

**UNIVERSIDAD CENTRAL DE VENEZUELA  
FACULTAD DE CIENCIAS  
POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN**



***INGENIERÍA DEL DOMINIO PARA LÍNEAS DE  
PRODUCTOS DE SOFTWARE ORIENTADAS A  
SERVICIOS EN EL MARCO DEL ESTÁNDAR  
ISO/IEC 26550***

**Tesis Doctoral presentado ante la ilustre Universidad  
Central de Venezuela por el M.Sc. Juan Carlos  
Herrera Ríos, para optar al título de Doctor en  
Ciencias de la Computación**

**Tutores:       Dra. Francisca Losavio  
                  Dr. Oscar Ordaz**

**Caracas – Venezuela  
Noviembre del 2017**



UNIVERSIDAD CENTRAL DE VENEZUELA  
FACULTAD DE CIENCIAS  
COMISIÓN DE ESTUDIOS DE POSTGRADO



Comisión de Estudios  
de Postgrado

VEREDICTO

Quienes suscriben, miembros del jurado designado por el Consejo de la Facultad de Ciencias y el Consejo de Estudios de Postgrado de la Universidad Central de Venezuela, para examinar la **Tesis Doctoral** presentada por el **MSc. Juan Carlos Herrera Ríos, C.I. 6.555.172** bajo el título **Ingeniería del Dominio para Líneas de Productos de Software Orientadas a Servicios en el marco del Estándar ISO/IEC 26550**, a fin de cumplir con el requisito legal para optar al grado académico de **Doctor en Ciencias de la Computación**, dejan constancia de lo siguiente:

1.- Leído como fue dicho trabajo por cada uno de los miembros del jurado, se fijó el día 10 de Noviembre de 2017, para que el **MSc. Juan Carlos Herrera** lo defendiera en forma pública, lo que Herrera hizo en el Auditorium Manuel Bemporad de la Escuela de Computación de la Facultad de Ciencias, mediante un resumen oral de su contenido, luego de lo cual **respondió satisfactoriamente** las preguntas que le fueron formuladas por el jurado, todo ello conforme con lo dispuesto en el Reglamento de Estudios de Postgrado.

2.- Finalizada la defensa del trabajo, el jurado decidió **aprobarlo**, por considerar, sin hacerse solidario con la ideas expuestas por **Herrera**, que está conforme a lo dispuesto y exigido en el Reglamento de Estudios de Postgrado.

Para dar este veredicto, el jurado estimó que el trabajo examinado constituye un aporte al área de la Ingeniería del Dominio (ID) de Líneas de Productos de Software (LPS) por las siguientes contribuciones: proponer un proceso de ID el cual enfoca una estrategia descendente para capturar los requisitos del dominio incluyendo los de calidad, los cuales son trasladados hasta la arquitectura de referencia de la LPS. Esta estrategia, que adapta la primera fase de alcance de un marco de referencia estándar de Ingeniería de LPS, permite disminuir el trabajo en las siguientes fases de ID. A partir de esta primera fase común, el proceso puede ser aplicado tanto a LPS como a LPS Orientadas a Servicios.

En fe de lo cual se levanta la presente ACTA, a los 10 días del mes de Noviembre del año 2017, conforme a lo dispuesto en el Reglamento de Estudios de Postgrado, actuó como **Coordinador** del Jurado el **Dr. Oscar Ordaz**.

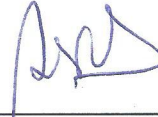
El presente trabajo fue realizado bajo la dirección de los tutores Dra. Francisca Losavio y Dr. Oscar Ordaz.

*Handwritten signatures and initials in blue ink, including 'BW', 'Definido', and 'Ordaz'.*



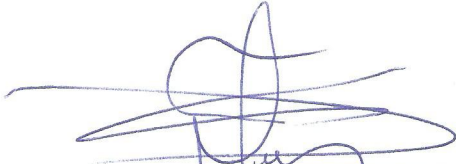
---

Dr. Christian Guillén Drija / C.I.  
10.524.455  
Universidad Metropolitana  
Jurado designado por el Consejo  
de la Facultad



---

Dra. Vanessa Leguizamo / C.I.  
13.309.245  
Universidad Central de Venezuela  
Jurado designado por el Consejo  
de la Facultad



---

Dr. Jean Carlos Guzmán / C.I.  
7.944.716  
Universidad Simón Bolívar  
Jurado designado por el Consejo  
de Estudios de Postgrado



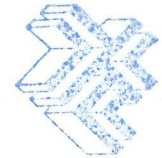
---

Dr. Eugenio Scalise / C.I. 10.184.983  
Universidad Central de Venezuela  
Jurado designado por el Consejo  
de Estudios de Postgrado



---

Dr. Oscar Ordaz / C.I. 2.963.836  
Universidad Central de Venezuela  
Tutor



POSTGRADO EN CIENCIAS  
DE LA COMPUTACION  
Universidad Central de Venezuela

CGD/VL/JCG/ES/OR/10-11-2017

# Resumen

---

Las *Líneas de Productos de Software (LPS)* y la *Arquitectura Orientada a Servicios*, conocidas también como *SOA: Service Oriented Architecture*) son enfoques para el desarrollo de software dirigidos a ser utilizados en la práctica industrial. Tienden a favorecer la reutilización de activos de software y capacidades existentes, en lugar de desarrollar de cero nuevos sistemas. Una LPS es un conjunto de sistemas complejos o intensivos de software, que comparten un conjunto común y organizado de características que satisfacen las necesidades de un sector específico del mercado o dominio. Estas características son desarrolladas a partir de un conjunto común de activos, que es reutilizado en diferentes productos o sistemas de software similares que conforman la familia LPS. La *Arquitectura de Referencia (AR)* es una arquitectura de software abstracta de carácter genérico, activo principal compartido por todos los productos de la LPS; abarca las partes comunes y variables de los productos de la familia y es utilizada como plantilla para producir nuevos productos. Por otra parte, SOA es un *modelo de referencia* de una arquitectura empresarial de negocios, donde la funcionalidad del negocio o lógica de la aplicación, es disponible como servicios reutilizables compartidos en una red. El objetivo de SOA es garantizar la transparente, flexible y dinámica interacción de los servicios prestados/consumidos, a través de la integración de múltiples dominios interconectados. SPL y SOA han mostrado su rentabilidad en el desarrollo de sistemas de software interoperables, reutilizables, adaptables y de rápido desarrollo, dando origen al enfoque de *Líneas de Productos de Software Orientadas a Servicios (LPSOS)*.

El objetivo principal de esta Tesis Doctoral, enmarcado en el primer ciclo de vida de la *Ingeniería del Dominio (ID)*, en la *Ingeniería de Líneas de Productos de Software (ILPS)*, es especificar y aplicar un proceso sistemático y repetible de ID para LPS, de acuerdo al estándar ISO/IEC 26550, que define un Modelo de Referencia para ILPS. El proceso explota los beneficios de SOA e ILPS, integrándolos en un proceso de diseño arquitectónico único. Para ello se definió un primer proceso **QuaDRA**: Arquitectura de Referencia Dirigida por la Calidad (*Quality-Design of Reference Architecture*), para construir un AR LPS. En este proceso todas las arquitecturas de software que se manejan son representadas por grafos no dirigidos y conexos  $G = (V, A)$ , siendo  $V$  los componentes de la arquitectura y  $A \subseteq V \times V$  los conectores entre los componentes. QuaDRA es adaptado luego en un segundo proceso **WSRA-SPL**: Arquitectura de Referencia de Servicios Web para LPS (*Web-Services Reference Architecture for SPL*), para construir una AR LPSOS. Ambos procesos se centran en el aseguramiento de la calidad del producto en etapas tempranas de desarrollo de la LPS; dos enfoques de desarrollo son combinados en la actividad Alcance del Dominio de la fase de Alcance de la SPL: proactivo (top-down)

y extractivo (bottom-up), con técnicas de reingeniería, en el Alcance del Portafolio de Productos, para así reducir el gran esfuerzo de desarrollo en las subsiguientes fases de la ID; se enfoca la calidad del software en etapas tempranas del desarrollo, la cual es especificada según el modelo de calidad estándar ISO/IEC 25010. QuaDRA y el WSRA-SPL integrado, serán aplicados al dominio de los Sistemas de Información Integrados de Salud.

*Palabras Clave:* Línea de Productos de Software (LPS), Línea de Productos de Software Orientada a Servicios (LPSOS); Ingeniería del Dominio (ID); Arquitectura de Referencia (AR); grafo conexo no dirigido; Modelo de Calidad; ISO/IEC 25010; Modelo de Referencia LPS; ISO/IEC 26550.

# Abstract

---

*Software Product Lines (SPL)* and *Service Oriented Architecture (SOA)* are software development approaches focusing the industrial practice. They favor reuse of software assets and existing capabilities, instead of developing new systems from scratch. SPL is a set of intensive or complex systems that share a common and organized set of features to satisfy the demand of a specific market sector or domain. These features are developed from a common set of assets that is reused in similar software products or systems, conforming the SPL family. The *Reference Architecture (RA)* is a generic and abstract software architecture that is the main asset shared by all SPL products; it includes common and variable parts of the family products, and it is used as a schema to derive new products. On the other hand, SOA is a *reference model* of an enterprise business architecture, where the business logic is available as shared reusable services through a network. SOA main goal is to guarantee transparent, flexible and dynamic interaction of provided/required services. SPL and SOA have shown fruitful results to a fast development of reusable and adaptable software systems, giving origin to the *Service Oriented SPL (SOSPL)*.

The main goal of this Doctoral Thesis, in the context of the Domain Engineering (DE) lifecycle of *SPL Engineering (SPLE)*, is to specify and apply a systematic and repeatable process for SPL, according to the standard ISO/IEC 26550 Reference Model for SPLE. The process takes advantages of the benefits of SPL and SOA, integrating both approaches in a unique architectural design process. For this, a first process was defined, **QuaDRA**: Quality-Design of Reference Architecture, to build an SPL RA; all the software architectures managed in this process are represented by non directed connected graphs  $G = (V, A)$  where  $V$  are the components of the architecture and  $A \subseteq V \times V$  are the connectors between the components. QuaDRA is then adapted in a second process, **WSRA-SPL**: Web-Services Reference Architecture for SPL, to construct a SOSPL RA; assurance of software product quality is focused at early stages of SPLE. Two development strategies are combined in the Domain Scoping activity of the SPL Scoping phase: proactive or top-down and extractive bottom-up, which use reengineering techniques to build the Product Portfolio, in order to reduce the huge developing effort in the subsequent DE phases; software product quality is focused at early development stages and it is specified by the ISO/IEC 25010 quality model standard. QuaDRA and the integrated WSRA-SPL will be applied to the Healthcare Integrated Information Systems domain.

*Keywords:* Software Product Lines (SPL); Service Oriented Software Product Lines (SOSPL); Domain Engineering, Reference Architecture (RA); connected non directed graph; Quality Model, ISO/IEC 25010; Reference Model SPL; ISO/IEC 26550.

# ***Agradecimiento***

---

A Dios todopoderoso, por cuidarme y ser mi guía en mi camino.

A mis tutores, Francis Losavio y Oscar Ordaz, por todo el apoyo y consideración en este transitar hacia mi Doctorado. Siempre los tendré presentes en mi corazón.

A todos aquellos con quien tuve la oportunidad de compartir en este desafiante e interesante camino, como profesores, amigos, compañeros de estudio, compañeros de trabajo, familiares.

GRACIAS...

# ***Dedicatoria***

---

*A mis padres, por apoyarme en todos los momentos de mi vida, y a mi hija Gabriela por ser mi inspiración a lo largo de todo este camino.*

*Gracias*



## Contenido

Lista de Tablas	12
Lista de Figuras	14
Glosario de Siglas	16
<b>CAPÍTULO I. INTRODUCCIÓN</b>	<b>18</b>
1.1 Problemática y alcance de la investigación	20
1.1.1 Estrategias para la adopción del enfoque LPS	25
1.1.2 Aspectos a ser considerados para el desarrollo de una LPSOS	27
1.2 Objetivo de la Tesis	32
1.2.1 Objetivos específicos	33
1.3 Identificación de los problemas a resolver para conseguir el objetivo	34
1.4 Metodología de investigación	35
1.4.1 Resultados parciales obtenidos para alcanzar los objetivos específicos durante el proceso metodológico de investigación	37
<b>CAPÍTULO II. CONTEXTO Y MARCO TEÓRICO</b>	<b>50</b>
2.1 Líneas de Productos de Software (LPS) – Ingeniería de LPS (ILPS) – El Marco de Referencia para LPS del estándar ISO/IEC 26550	51
2.1.1 Líneas de Productos de Software (LPS)	51
2.1.2 Marco de Referencia para LPS del estándar ISO/IEC 26550	53
2.2 Líneas de Productos de Software Orientadas a Servicios (LPSOS)	62
2.2.1 Arquitectura de Software Orientadas a Servicios	63
2.2.2 Integración de los Enfoques de Líneas de Productos de Software y Arquitectura Orientada a Servicios	64
2.2.3. LPSOS dinámicas	65
2.3 Calidad del Software y calidad del producto de software según el estándar ISO/IEC 25010	66
2.3.1 Calidad del Software	66
2.3.2 Estándares de calidad del software	67
2.3.3 Marco de referencia del Modelo de Calidad ISO/IEC 25010	70
2.4 Dominio de los Sistemas de Información Integrados de Salud (SIS)	76
2.4.1 Sistemas integrados de salud	76
2.4.2 SIS y su relación con SOA	79
2.5 Ontologías en el desarrollo de LPS	80
2.5.1 ¿Qué es una ontología?	82
2.5.2 Una Simple Metodología de Ingeniería del Conocimiento	82
2.5.3 Representación mediante Ontología de una Arquitectura de Referencia	83
2.6 Revisión Documental Sistemática (RDS)	85
2.6.1 Introducción	85
2.6.2 Razones para la Realización de una RDS	88
2.6.3 Ventajas/Desventajas de la RDS	88
2.6.4 Estructura del proceso de revisión	89

2.6.5 Protocolo de la RDS	89
2.6.6 Estructura del documento para la RDS	90
<b>CAPÍTULO III. TRABAJOS RELACIONADOS - MÉTODOS DE DESARROLLO DE LPS Y LPSOS EN EL CICLO DE VIDA DE LA ID</b>	<b>98</b>
3.1 RDS: Diseño de AR y/o ALP	99
3.2 Métodos de desarrollo para LPS que abarcan el ciclo de vida ID	102
3.2.1 RA&NFR-VAR	102
3.2.2 RDS: Otros métodos de desarrollo basados en SOA y LPS que contemplan ID	105
3.3 RDS: Métodos de Diseño de Arquitecturas de Referencia para Líneas de Productos de Software Orientadas a Servicios	112
3.3.1 Aspectos a ser considerados en la RDS	112
3.3.2 Marcos para la evaluación de métodos para el desarrollo de AR-LPS, SOA, AR-SOA	115
3.3.3 Criterios y elementos del Marco Único de Evaluación basado en los tres marcos AR-LPS, SOA y AR-SOA	118
3.3.4 Realización de la RDS [HLO16]	120
3.4 Métodos de desarrollo basados solo en SOA	132
3.4.1 Metodologías SOA	132
3.4.2 SOMA (Service-oriented Modeling and Architecture)	136
3.4.3 Proceso SOMA Detallado	140
<b>CAPÍTULO IV. PROCESO DE INGENIERÍA DEL DOMINIO DE LÍNEAS DE PRODUCTOS</b>	<b>153</b>
4.1 Integración de enfoques para el desarrollo de la ID	154
4.1.1 Enfoques para la Ingeniería del Dominio	154
4.1.2 Integración del enfoque de Bjørner al Marco de Referencia ISO/IEC 26550 para LPS	157
4.2 Proceso PLScopP: Product Line Scoping Process	163
4.3 QuaDRA: Quality Driven Reference Architecture	165
4.3.1 El Proceso de QuaDRA	166
4.3.2 Diagrama SPEM de QuaDRA	174
4.3.3 Conclusión	174
<b>CAPÍTULO V. PROCESO DE INGENIERÍA DE LÍNEAS DE PRODUCTOS DE SOFTWARE ORIENTADAS A SERVICIOS</b>	<b>179</b>
5.1 Premisas	181
5.1.1 Programación robusta, Servicios Web y QoS	181
5.1.2 Modelo SOA y la Arquitectura de Servicios Web (WSA)	182
5.2 Diseño de Arquitecturas de Servicios Web	183
5.3 WSRA-SPL: Web Services Reference Architecture for SPL – Adaptación de QuaDRA a SOA	186
5.3.1 Supuestos Preliminares para WSRA-SPL	186
5.3.2 Proceso WSRA-SPL - Especificación textual	188
5.3.3 Diagrama SPEM - WSRA-SPL	194
<b>CAPÍTULO VI. APLICACIÓN DE LOS PROCESOS QuaDRA Y WSRA-SPL AL DOMINIO DE LOS SISTEMAS DE INFORMACIÓN</b>	<b>197</b>

INTEGRADOS DE SALUD (SIS)	
6.1 Alcance del Caso de Estudio	198
6.2 Aplicación paso a paso del proceso QuaDRA	199
6.2.1 Aplicación del Proceso de Diseño QuaDRA para el Dominio de los SIS	199
6.2.2 Documentación de la AR	229
6.2.3 Derivación de un Producto SIS a partir de la AR	231
6.3 Aplicación paso a paso del proceso WSRA-SPL	235
6.3.1 Aplicación del Proceso de Diseño WSRA-LPS para el Dominio de los SIS	235
6.3.2 Análisis de los Resultados	246
CAPÍTULO VII. CONCLUSIONES Y PERSPECTIVAS	250
7.1 Logros	251
7.2 Limitaciones	254
7.3 Perspectivas	255
APÉNDICES	258
Publicaciones	259

## Lista de Tablas

<b>Tabla 1.</b> Definición de características y sub-características del Modelo de Calidad del Producto ISO/IEC 25010 [ISO10]	74
<b>Tabla 2.</b> Proceso de revisión sistemática	89
<b>Tabla 3.</b> Construyen la Arquitectura (Enfoque Reactivo)	101
<b>Tabla 4.</b> Construyen la Arquitectura (Enfoque Proactivo)	102
<b>Tabla 5.</b> Propiedades de Extracción	106
<b>Tabla 6.</b> Relación de Estudios Localizados y Aceptados según la Fuente de Datos	106
<b>Tabla 7.</b> Etapas (Categorías) para el Desarrollo del Dominio	107
<b>Tabla 8.</b> Otras Categorías de Interés para la Investigación	107
<b>Tabla 9.</b> Actividades Realizadas en las Fases de Análisis, Diseño e Implementación de la LPSOS	108
<b>Tabla 10.</b> Artefactos Desarrollados en las Fases de Análisis, Diseño e Implementación de la LPSOS	108
<b>Tabla 11.</b> Técnicas Utilizadas en las Fases de Análisis, Diseño e Implementación de la LPSOS	109
<b>Tabla 12.</b> Criterios y Elementos de Evaluación para Métodos de Desarrollo de AR-LPS	118
<b>Tabla 13.</b> Criterios y Elementos de Evaluación para Métodos de Desarrollo de Sistemas bajo el Modelo SOA	119
<b>Tabla 14.</b> Criterios y Elementos de Evaluación para Métodos de Desarrollo de AR-SOA	120
<b>Tabla 15.</b> Relación de Estudios Localizados y Aceptados según la Fuente de Datos	122
<b>Tabla 16.</b> Relación de Estudios Primarios Relativos a Procesos de Desarrollo para LPSOS según Tipo y Año de Publicación	123
<b>Tabla 17.</b> Descripción de las Propiedades: Métodos para AR-LPS	123
<b>Tabla 18.</b> Descripción de las Propiedades: Métodos para SOA	124
<b>Tabla 19.</b> Descripción de las Propiedades: Métodos para AR-SOA	124
<b>Tabla 20.</b> Checklist de las Propiedades para Métodos AR-LPS Relevantes Identificadas en los Estudios Primarios sobre LPSOS	125
<b>Tabla 21.</b> Checklist de las Propiedades para Métodos bajo SOA y AR-SOA Relevantes Identificadas en los Estudios Primarios sobre LPSOS	125
<b>Tabla 22.</b> Estudios más Representativos entre los Primarios para LPSOS	126
<b>Tabla 23.</b> Características para el Análisis de las Fases de Análisis y Diseño de Metodologías SOA	132
<b>Tabla 24.</b> Análisis de Metodologías SOA Existentes	133
<b>Tabla 25.</b> Resumen de la Comparativa de Metodologías SOA	135
<b>Tabla 26.</b> Tareas y Roles en las Fases de SOMA	141

### Lista de Tablas (continuación)

<b>Tabla 27.</b> Correspondencia entre las Fases de la ID según Pohl et. al, ISO/IEC 26550 y Bjørner	156
<b>Tabla 28.</b> Especificación de las facetas (vistas o perspectivas) del dominio	160
<b>Tabla 29.</b> Intrínsecas para describir las facetas del dominio	161
<b>Tabla 30.</b> Correspondencia con SOMA, Diseño Arquitectural: RUP&SOMA, y WSRA-SPL: QuaDRA adaptado	184
<b>Tabla 31.</b> CCT: Componentes y Conectores de cada configuración arquitectural de productos	203
<b>Tabla 32.</b> MCE: Vista general de la configuración arquitectónica de la AC	205
<b>Tabla 33.</b> UDD para EHR-SIS de los Doctores	209
<b>Tabla 34.</b> UDD para EHR-HIS de los Ingenieros de Dominio	209
<b>Tabla 35.</b> UDD para EHR-HIS del punto de vista de los Directores de Instituciones Gubernamentales de la Salud.	210
<b>Tabla 36.</b> Especificación por Intrínsecas del punto de vista de los Doctores para la Faceta de Procesos de Negocio Servicios de Cita.	211
<b>Tabla 37.</b> Notación BPMN para la especificación de los Procesos de Negocios	213
<b>Tabla 38.</b> Especificación por intrínsecas del punto de vista de los Ingenieros de Dominio para la Faceta Soporte Tecnológico.	215
<b>Tabla 39.</b> Especificación por intrínsecas del punto de vista de los Directores de HGI para la Faceta Reglas & Regulaciones.	218
<b>Tabla 40.</b> Tabla MCE Actualizada (MCEA): Vista General de la AC completada (ACA)	224
<b>Tabla 41.</b> DOC-AR: Documentación de la AR	230
<b>Tabla 42.</b> Tabla MCE – Portafolio de Servicios	236
<b>Tabla 43.</b> Tabla MCEA – Portafolio de Servicios Actualizado con servicios para obtener la ACA con Servicios	240

## Lista de Figuras

<b>Figura 1.</b> Arquitectura de estilo híbrido eventos/SOA/Capas	32
<b>Figura 2.</b> Estructura de la Tesis	45
<b>Figura 3.</b> Modelo de Referencia para Líneas de Productos de Software y Sistemas del ISO/IEC 26550	54
<b>Figura 4.</b> Modelo de Referencia SOA	64
<b>Figura 5.</b> Modelo de calidad del producto del ISO/IEC 25010 (versión oficial)	68
<b>Figura 6.</b> Modelo de calidad del producto del ISO/IEC 25010 (traducción no oficial)	69
<b>Figura 7.</b> Alcance de las medidas de calidad	71
<b>Figura 8.</b> Propiedades del software	72
<b>Figura 9.</b> Estructura utilizada para el modelo de calidad	73
<b>Figura 10.</b> Modelo de calidad del producto para SIS	79
<b>Figura 11.</b> Vista OwIViz en Protegé de la HIS-RA mostrando la jerarquía de clases de componentes comunes, puntos de variación y soluciones arquitecturales; la clase SetB es utilizada en el ciclo IA para la derivación de productos, no se considera en el contexto de este trabajo	85
<b>Figura 12.</b> Vista funcional de una RDS expresada en diagrama de casos de uso en UML	91
<b>Figura 13.</b> Vista conceptual de una RDS expresada en un modelo conceptual	91
<b>Figura 14.</b> Vista de procesos de negocio de una RDS expresada en BPMN	92
<b>Figura 15.</b> Fases de SOMA, un modelo de desarrollo de software	138
<b>Figura 16.</b> Flujo de alto nivel del ciclo de vida SOMA	139
<b>Figura 17.</b> Modelo de Referencia Conceptual que integra los enfoques de la Tabla 27	159
<b>Figura 18.</b> Modelo Conceptual para Modelar un Dominio (sub-fase “Modelado del Dominio”)	162
<b>Figura 19.</b> Diagrama SPEM del Proceso PLScoP	164
<b>Figura 20.</b> Diagrama SPEM del Proceso QuaDRA	174
<b>Figura 21.</b> Capa de Servicios SOA	183
<b>Figura 22.</b> Proceso WSRA-SPL en SPEM	194
<b>Figura 23.</b> Las tres arquitecturas de SIS estudiadas originales, OpenEMR, PatientOS y Care2X	200
<b>Figura 24.</b> Las tres arquitecturas de SIS estudiadas, en UML OpenEMR, PatientOS y Care2X	201
<b>Figura 25.</b> AC para el dominio SIS-EHR	204
<b>Figura 26.</b> Punto de vista de los Doctores de la Faceta Procesos de Negocios para Servicios de Citas (UML)	212

### Lista de Figuras (continuación)

<b>Figura 27.</b> Procesos de negocios extraídos desde el punto de vista de los Doctores para la Faceta Procesos de Negocios (BPMN)	214
<b>Figura 28.</b> Punto de vista de los Ingenieros de Dominio para la Faceta Tecnologías Soportadas (UML).	216
<b>Figura 29.</b> Procesos de negocios extraídos desde el punto de vista de los Ingenieros de Dominio para la Faceta Tecnologías Soportadas (BPMN)	217
<b>Figura 30.</b> Punto de vista de los Directores IGS para la Faceta Reglas & Regulaciones (UML)	219
<b>Figura 31.</b> Procesos de negocios extraídos desde el punto de vista de los Directores de IGS para la Faceta Reglas & Regulaciones (BPMN)	220
<b>Figura 32.</b> Agregación de los procesos de negocios para las facetas estudiadas – Modelo del Dominio (BPMNAgg)	222
<b>Figura 33.</b> AC Completada (ACA)	227
<b>Figura 34.</b> AR	228
<b>Figura 35.</b> Derivación de un Producto a partir de la AR.	234
<b>Figura 36.</b> WSRA	245

## ***Glosario de Siglas***

<b><i>Sigla</i></b>	<b>Español <i>Término</i></b>	<b><i>Sigla</i></b>	<b>Inglés <i>Término</i></b>
AR	Arquitectura de Referencia	RA	Reference Architecture
ALP	Arquitectura de Líneas de Productos	PLA	Product Line Architecture
AC	Arquitectura Candidata	CA	Candidate Architecture
ACA	AC Actualizada	UCA	Updated CA
AR-SIS	Arquitectura de Referencia de Sistemas de Información Integrados de Salud	HIS-RA	Healthcare Integrated Information Systems Reference Architecture
AROS	Arquitectura de Referencia Orientada a Servicios	SORA	Service Oriented Reference Architecture
BG	Objetivos de Negocios	BG	Business Goals
BPEL	Lenguaje de Ejecución de Procesos de Negocios	BPEL	Business Process Execution Language
DD	Diseño del Dominio	DD	Domain Design
EBSE	Metodología de IS basada en Evidencias	EBSE	Evidence-based Software Engineering
HCE	Historias Clínicas Electrónica	EHR	Electronic Healthcare Information Systems
IA	Ingeniería de Aplicación	AE	Application Engineering
ID	Ingeniería del Dominio	DE	Domain Engineering
IS	Ingeniería de Software	SE	Software Engineering
ILPS	Ingeniería de Línea de Productos de Software	SPLE	Software Product Line Engineering
IRD	Ingeniería de Requisitos del Dominio	DRE	Domain Requirements Engineering
LPS	Línea de Producto de Software	SPL	Software Product Line
LPSOS	Línea de Productos de Software Orientadas a Servicios	SOSPL	Service Oriented Software Product Lines
LPSD	Líneas de Productos de Software Dinámicas	DSPL	Dynamic Software Product Line
MCD	Modelo de Calidad del Dominio	DQM	Domain Quality Model



<i><b>Sigla</b></i>	<b>Español</b> <i><b>Término</b></i>	<i><b>Sigla</b></i>	<b>Inglés</b> <i><b>Término</b></i>
MCE	Modelo de Calidad Extendido	EQM	Extended Quality Model
MCEA	Modelo de Calidad Extendido Actualizado	UEQM	Updated Extended Quality Model
PLScop	Alcance de la Línea de Productos	PLScop	Product Line Scoping
PN	Proceso de Negocio	BP	Business Process
QoS	Calidad del Servicio	QoS	Quality of Service
QuaDRA	Diseño de Arquitectura de Referencia Dirigida por la Calidad	QuaDRA	Quality-Design of Reference Architecture (QuaDRA)
RAC	Repositorio de Activos Centrales	CAR	Core Asset Repository
RDS	Revisión Documental Sistemática	SDR	Systematic Document Review
RF	Requisitos Funcionales	FR	Functional Requirement
RNF	Requisitos No Funcionales	NFR	No-functional Requirement
SC	Servicios Candidatos	CS	Candidate Services
SIS	Sistemas de Información Integrados de Salud	HIS	Healthcare Integrated Information Systems
SIS-HCE	Sistemas de Información Integrados de Salud para HCE	EHR-HIS	Healthcare Integrated Information Systems for EHR
SOA	Arquitectura Orientada a Servicios	SOA	Service-Oriented Architecture
TCC	Tabla de Componentes y Conectores	CCT	Components and Connectors Table
UDD	Unidades de Descripción del Dominio	DDU	Domain Description Units
SW	Servicios Web	WS	Web Services
WBS	Estructura de Descomposición del Trabajo	WBS	Work Breakdown Structure
WSRA-SPL	Arquitectura de Referencia de Servicios Web para LPS	WSRA-SPL	Web-Services Reference Architecture for SPL

# ***INTRODUCCIÓN***

---

---

# CAPÍTULO I

## INTRODUCCIÓN

La presente Tesis Doctoral aborda la problemática de construir una *Línea de Productos de Software Orientada a Servicios (LPSOS)* [AG13] [EMA13] [QB09] [GB08] [INP+09] [LK10] [PBD09]; la *Arquitectura de Referencia (AR)* es aquí un elemento crucial para la derivación de productos concretos de software en un contexto de producción industrial de *Líneas de Productos de Software (LPS)* [Bos00] [CN01]. AR es una arquitectura de software definidas en términos de componentes y conectores con un comportamiento, en el sentido clásico de [SG96], pero abstracta, genérica e instanciable [PBL05] [ISO15], que representa una familia de sistemas de software o productos similares de un dominio o sector del mercado. El termino *dominio* es comúnmente utilizado para referirse a un área específica del conocimiento y un dominio de aplicación involucra cualquier aspecto en el cual se puede aplicar la computación [Ber92].

El objetivo general de esta Tesis, el cual será retomado en la Sección 1.2, es: “*Definir un Proceso de Ingeniería del Dominio para Líneas de Productos de Software Orientadas a Servicios, considerando aspectos de calidad del software*”. Ahora bien, el artefacto principal de una LPS es su AR, la cual se construye en el ciclo de Ingeniería del Dominio de la Ingeniería de la LPS [PBL05] [ISO15]. La AR para *Líneas de Productos Orientadas a Servicios (LPSOS)* o *AR-LPSOS*, objetivo central de esta investigación, será construida en un solo proceso, *WSRA-SPL: Arquitectura de Referencia de Servicios Web para LPS* o “*Web-Services Reference Architecture for SPL*” [HLO16c], el cual adapta un primer proceso, diseñado también en el marco de esta investigación, *QuaDRA: Arquitectura de Referencia Dirigida por la Calidad* o “*Quality-Design of Reference Architecture*” [HLO16b], en el cual se construye una AR para LPS o AR-LPS.

El enfoque LPS surge históricamente antes que el modelo de referencia de *Arquitectura Orientada a Servicios o "Service Oriented Software Architecture (SOA)"* [W3C04] base del enfoque de LPSOS, ambos enfoques fuertemente centrados en la reutilización. En particular, la LPS se centra en el desarrollo de software basado en la reutilización, que se fundamenta en la identificación y captura de las similitudes y variabilidades de productos de software similares dentro de un dominio o sector del mercado determinado [PBL05]; con ello se pretende reducir el tiempo, esfuerzo, costo, complejidad y una mejora de la productividad en el desarrollo y mantenimiento de sistemas de software complejos [NOB11].

En este primer capítulo introductorio se discute el problema planteado y el alcance de la investigación, y se justifica la selección de los enfoques y técnicas a integrar para el desarrollo de un proceso de Ingeniería de LPSOS, describiendo brevemente sus características principales; el contexto completo de la investigación será tratado luego en el Capítulo II. En una segunda sección se presenta el objetivo principal de la tesis y los objetivos específicos del mismo. En la tercera sección se describe el método de investigación utilizado, luego se presentan los principales resultados obtenidos durante la investigación, así como la organización de los restantes capítulos de esta Tesis Doctoral.

### **1.1 Problemática y alcance de la investigación**

En muchos dominios, el desarrollo de sistemas de software de gran complejidad y de gran escala representa siempre un desafío para los investigadores y desarrolladores en el área de la *Ingeniería de Software (IS)*, principalmente por la tarea no trivial de desarrollar pensando en la reutilización de componentes para la configuración de sistemas con un *núcleo básico* o "*core*" de artefactos reutilizables comunes y variables, denominados *activos* o "*assets*" compartidas por todos los productos similares de la familia LPS, [HHJ08] [INPJ09] [Ist09] [MSR10], favoreciendo así un desarrollo más rápido y confiable debido a la reutilización.

Desde finales de los '60 comienza el desarrollo basado en componentes o módulos, y aún más en la década de los '90 el paradigma de las LPS, fuertemente

promocionado por el *Software Engineering Institute (SEI)* de la universidad de Carnegie-Mellon, EEUU, ha cobrado impulso en la industria del software. En lugar de desarrollar sistemas a partir de cero, estos deben ser construidos a partir de partes reutilizables. En vez de componer un sistema siempre de la misma manera, éste debería ser adaptado a las necesidades de los clientes, quienes pueden elegir entre un conjunto de opciones de configuración [ABK+13] [LOJ16]. El enfoque LPS proporciona una forma de personalización en masa mediante la construcción de soluciones individuales basadas en un repositorio de activos o componentes de software reutilizables. La necesidad de adaptar soluciones individuales responde a diversas necesidades en el software con respecto a la funcionalidad, plataformas destino, y propiedades no funcionales, como por ejemplo, el rendimiento y espacio de memoria [ABK+13].

Las LPS prometen distintos beneficios [MS10] [ABK+13], de los cuales los más importantes son: a) adaptabilidad a las necesidades del cliente, b) reducción de costos, c) mejora de la calidad global, y d) disminución del tiempo de comercialización. Este enfoque representa un gran desafío para los investigadores y desarrolladores en el área de la IS, principalmente por la tarea de desarrollar artefactos (activos) suficientemente genéricos y de calidad para pensar en su reutilización y en la configuración de nuevos sistemas contextualizados, acorde con las necesidades de los clientes [Istoan 09] [MSR10].

Para el abordaje adecuado de este enfoque, se define un modelo de referencia de proceso para la *Ingeniería de la LPS (ILPS)* [PBL05] [ISO15], el cual propone lineamientos generales de desarrollo de LPS, que involucran métodos y técnicas en uso actualmente para el desarrollo de software para un solo sistema que generalmente deben ser adaptadas al contexto LPS. ILPS [ISO 13] establece como objetivos principales: a) apoyar arquitecturas de software configurables y evolutivas y b) permitir la personalización masiva de sistemas de software similares, miembro de una familia LPS [MAG+13], todo este inspirado en las cadenas de producción industrial automatizada, surgidas a principios del siglo '20.

Dado que las arquitecturas de las LPS son representaciones en el espacio del problema a un alto nivel de abstracción de aplicaciones, productos o sistemas de software similares en un dominio de interés, en el primer ciclo de vida de la Ingeniería del Dominio, dentro de la ILPS, es importante que se interrelacionen con los modelos del espacio de soluciones en el segundo ciclo de vida, la Ingeniería de la Aplicación, que permiten su puesta en funcionamiento real [MGH+13].

Por otra parte, SOA [W3C04] define una arquitectura o marco de referencia empresarial, centrada en la comunicación en redes a través de servidores Web que actúan como intermediarios para satisfacer demandas de servicios por clientes distribuidos en ubicaciones geográficamente distantes [GAT13]. Un *servicio* se define como una unidad discreta de la funcionalidad de negocio que está disponible a través de su interfaz por un contrato de servicios [RLS+08]; puede ser visto como un componente de software reutilizable. SOA proporciona un marco de referencia para la construcción de soluciones empresariales integradas basadas en servicios, que pueden automatizar los procesos de negocios. Particularmente se ocupa de la construcción independiente de servicios alineados al negocio que pueden ser combinados para dar soluciones a procesos de negocio de alto nivel para así obtener soluciones computacionales de más bajo nivel, en el contexto empresarial. El valor real de SOA no es solo la provisión adecuada de los servicios, sino más bien es cuando los servicios reutilizables, ágiles y flexibles, se combinan para “implementar” la parte computacional de los procesos de negocios. SOA ha demostrado su rentabilidad en el desarrollo de sistemas de software interoperables, reutilizables, adaptables y de rápido desarrollo, por lo cual existen importantes ventajas mutuas en la convergencia de SOA y LPS [GAT13] [CK10] [KLR09], dando origen al enfoque de LPSOS. La integración de los enfoques LPS y SOA no es tarea trivial; requiere que las organizaciones de desarrollo de LPS apliquen la gestión de la variabilidad respecto a los servicios que son vistos como componentes de software esenciales de SOA, y que los desarrolladores de sistemas orientados a servicios apliquen técnicas que usan los desarrolladores de LPS, como son la consideración de los servicios como componentes arquitecturales de alto nivel, compuestos mediante técnicas de

orquestración y coreografía. La necesidad de variación en cuanto a servicios, puede ser impulsada por las condiciones del mercado, las oportunidades del negocio, las nuevas tecnologías, las reglas de negocio, entre múltiples factores relacionados con la empresa. Esta necesidad puede expresarse en forma de objetivos de negocio establecidos por las organizaciones que desarrollan las LPSOS [CK10], las cuales utilizan los principios de SOA de composición de servicios para proporcionar una implementación concreta de los diferentes productos miembros de una familia LPS [MGH+13].

En los aspectos antes tratados, el aseguramiento de la calidad del producto de una LPS es una actividad crucial para su éxito, mantenimiento y evolución [ISO15], en un contexto de producción industrial del software; por lo tanto, cuando se trata del desarrollo de LPS, la reutilización masiva de los activos de software hace que los atributos de calidad (propiedades medibles de un artefacto de software) relativos a los requisitos no funcionales asociados a los requisitos de calidad [ISO11], impacten en la calidad de todos los productos de la LPS [Gon12]. En general, a nivel de diseño arquitectónico, la calidad no es fácil de medir, ya que el sistema aún no está construido y se utilizan evaluaciones más bien cualitativas (sujetas a la experticia del arquitecto) y no cuantitativas. Ahora bien, respecto a la evaluación de la calidad en una AR-LPSOS, que es una tarea difícil en arquitecturas de sistemas de software aislados y monolíticos, lo es aún más cuando se trata de familias de sistemas, ya que se requiere analizar las características de calidad, considerando propiedades funcionales y no funcionales, comunes y variantes en diferentes niveles de abstracción, como la calidad directamente ofrecida por los servicios y la calidad que ellos proporcionan como componentes arquitecturales de alto nivel para satisfacer las funcionalidades. Ahora bien, particularmente en Ingeniería de LPS, la determinación de la *calidad del servicio* o “*Quality of Service (QoS)*” es fundamental, debido a que varía acorde a características individuales, por lo cual los ingenieros del dominio deben asegurar que los servicios que van a formar parte del sistema satisfacen los rangos de valores establecidos por los QoS solicitados por las partes interesadas

[MGH+13], además de satisfacer las exigencias de calidad de alto nivel de las funcionalidades [HLO16].

Esto implica la necesidad de unificar criterios en el abordaje de los *Requisitos No Funcionales (RNF)* en etapas tempranas, ya que de ellos se derivan los atributos de calidad requeridos por los *Requisitos Funcionales (RF)*, que deben ser identificados y medidos, ya que condicionan la calidad global del sistema [ISO11].

El hecho de considerar estándares de calidad, representa un aspecto central en la tesis, acorde con las buenas prácticas de la IS, se justifica porque contribuyen a la construcción de un vocabulario común mediante la especificación de la calidad del producto mediante un modelo de calidad definido por un estándar [ISO11], constituyendo así un activo de software importante para la construcción de la AR; debe recordarse que esta arquitectura genérica abarca las partes comunes y variables de la familia que satisfacen RF y RNF y es utilizada como plantilla para la generación de los diferentes productos en la Ingeniería de la Aplicación, segundo ciclo de vida de la ILPS.

En dos Revisiones Documentales Sistemáticas (RDS) [HLM14] [HLM+14] realizadas, según [Kit07], se han detectado las siguientes evidencias de problemas aún no completamente resueltos, para algunos de los cuales esta tesis proporciona una solución:

- No se describen en detalle las actividades a realizar y artefactos que se producen en los procesos y/o métodos para la adquisición del conocimiento del dominio, para ninguna de las ingenierías LPS y LPSOS.
- No se describen diferentes perspectivas o vistas del dominio que pueden ser analizadas, como por ejemplo las facetas del dominio utilizadas por el enfoque de Bjørner [Bjø06].
- Los RF son los abordados en general, pero los RNF y/o requisitos de calidad (nótese que en la literatura estos términos con frecuencia se



usan indistintamente) son dejados de lado, desde las siguientes dos perspectivas [MGH+13]: características de calidad del software detectadas durante el proceso de desarrollo de la LPS (mantenibilidad, complejidad, eficiencia, confiabilidad, etc.) y características de calidad específicas del dominio de la aplicación o calidad global (disponibilidad, seguridad, compatibilidad, etc.)

- La determinación de las características de calidad no es realizada en las etapas tempranas del proceso de Ingeniería del Dominio, siendo estos aspectos cruciales para el diseño de la arquitectura [HLO+16] y de vital importancia para derivar de forma adecuada el conjunto de productos de la LPSOS.
- El problema de la variabilidad no funcional no está aún resuelto en LPS, y menos aún en LPSOS.

### 1.1.1 Estrategias para la adopción del enfoque LPS

En la mayoría de los casos, una empresa que se mueve hacia un desarrollo LPS tiene ya algunos productos del dominio de aplicación en su cartera de productos. En estos casos, se considera una transición a una LPS, más que de un desarrollo a partir de cero; un enfoque de diseño de la AR *ascendente* o “*bottom-up*” podría ser utilizado. En cambio, cuando no hay aún productos hechos en un dominio dado, podría usarse el enfoque *descendente* o “*top-down*” [ABK+13]. A continuación Krueger citado en [HLM14], distingue tres caminos de adopción diferentes:

- El enfoque *proactivo o descendente* (“*top-down*”) desarrolla una línea de productos a partir de cero utilizando métodos de análisis y diseño generalmente tradicionales o adaptados para LPS.

- El enfoque *extractivo o ascendente* (“*bottom-up*”) comienza con una colección de productos existentes e incrementalmente se refactorizan<sup>1</sup> para formar una línea de productos.
- El enfoque *reactivo*, también considerado en la literatura como un *bottom-up*, comienza con una pequeña línea de productos (posiblemente consistiendo de un solo producto construido), fácil de manejar y se extiende gradualmente con nuevas características y artefactos de implementación, ampliando así el alcance de la LPS.

La transición o migración de las organizaciones al paradigma LPS puede lograrse a través de los tres enfoques antes mencionados, de amplia adopción, proactivo, reactivo, extractivo [ABK+13] e híbrido [HLO16a] [HLO16c]:

El enfoque proactivo es apropiado cuando los requisitos para el conjunto de los productos necesarios y futuros, están bien definidos y estables. Este enfoque requiere un esfuerzo considerable para la organización en cuanto a costo y recursos, pero esto disminuye considerablemente una vez que la línea de producción se ha completado [KBL 05] [Kru02]. Las tareas de alto nivel que se llevan a cabo son: a) realizar un análisis de dominio y del alcance para identificar la variación que será soportada en la línea de producción; b) modelar la arquitectura de la línea de productos para soportar todos los productos en la LPS y c) diseñar las partes comunes y variables del sistema [Kru02].

En el enfoque reactivo, la organización ya tiene una línea de productos y la quiere extender, debido a la demanda de nuevos productos o cuando surgen nuevos requisitos respecto a los productos existentes. Las tareas de alto nivel que se llevan a

---

<sup>1</sup> El término refactorización se deriva de las matemáticas, cuando una expresión se convierte en otra equivalente que da el mismo resultado. En ingeniería del software se ha usado mucho a nivel de código; se realizan modificaciones al código, pero sin cambiar su objetivo inicial para obtener el mismo resultado. Refactorización se refiere al proceso de reescribir y reorganizar el texto para abreviarlo preservando su contenido. Por ejemplo, la expresión  $x^2-1$  puede ser factorizado como  $(x+1)(x-1)$ , revelando una estructura interna que no era visible previamente (como las dos raíces en  $-1$  y  $+1$ ; nótese que el resultado de la expresión matemática no cambia, pero la forma de presentarla sí (Wikipedia). En nuestro contexto significa que las arquitecturas de cada producto puede ser transformada en otra equivalente.

cabo son: a) caracterizar los requisitos para el nuevo producto en relación con los actualmente soportados en la línea de producción y b) si el nuevo producto está dentro del alcance de la línea de producción actual, se crea la definición del producto, en caso contrario, se amplía el alcance de la línea de producción actual para incluir los nuevos requisitos [Kru02].

En el enfoque extractivo, la organización aprovecha los sistemas de software existentes en el dominio o construidos ya por la organización, mediante la extracción de lo común y lo diverso, a partir del código fuente o de documentación existente, para sintetizarlos en una sola línea de producción. Requiere utilizar técnicas de reingeniería para reconstruir la arquitectura del sistema. Las tareas de alto nivel que se llevan a cabo son: a) identificar elementos comunes y la variación en los sistemas existentes y b) factorizarlos en una sola arquitectura para la LPS [Kru02].

También se tiene el enfoque denominado *híbrido* o “*middle-out*” o “*meet-in-the-middle*” [HLO16c] se parte de la especificación de los procesos de negocio como orquestaciones de servicios y se relacionan con los componentes de una AR ya existente; es muy utilizado para el desarrollo de sistemas basados en SOA [HLO16c].

En esta tesis se combinan los enfoques proactivo y extractivo en forma similar al middle-out. El enfoque extractivo es considerado similar al enfoque reactivo, ya que en ambos se reutilizan los sistemas existentes [KBL05] ya desarrollados por la empresa y/o LPS existentes en la organización.

### **1.1.2 Aspectos a ser considerados para el desarrollo de una LPSOS**

El enfoque LPSOS requiere considerar los siguientes elementos: la *Gestión de Procesos de Negocios (GPN)* o “*Business Process Management (BPM)*”, la gestión de la variabilidad funcional y no funcional y su respectiva calidad, la representación de una AR Orientada a Servicios. Estos aspectos son considerados prioritarios en una RDS realizada en [DF13] que presenta enfoques para la aplicación conjunta de LPS y GPN, incluyendo el soporte de SOA para GPN, en la cual se concluye que la combinación LPS, BPM y SOA es factible y beneficiosa para la construcción de un

proceso sistemático integrado para elaborar una LPSOS. A continuación se describen brevemente cada uno de estos aspectos.

### ***A. Gestión de Procesos de Negocios***

GPN en [AMK+09] define un proceso de negocio como un conjunto de actividades que se realizan en coordinación con un entorno organizacional y técnico. Estas actividades se desempeñan de manera conjunta para realizar los objetivos del negocio. Cada proceso de negocio es generado por una sola organización, pero puede interactuar con los procesos de negocios generados por otras organizaciones. Las partes fundamentales de GPN son su ciclo de vida y el *Lenguaje de Modelado de Procesos de Negocio* o “*Business Process Management Language (BPML)*” [AMK+09]. Existen varios BPMLs concebidos en la academia y la industria que se utilizan para representar el flujo de trabajo y los artefactos y/o producidos utilizados durante la realización de las actividades en el ciclo de vida. Algunos de estos son el *Diagrama de Actividades* (UML<sup>2</sup> 2.0) [OMG05] y la *Notación de Procesos de Negocios* o “*Business Process Modeling Notation (BPMN)*” [OMG11], los cuales fueron evaluados por un framework en [LK06]. Esta evaluación determinó que el BPMN es uno de los BPMLs más utilizado en la fase de diseño de un ciclo de vida GPN. Representa los procesos como actividades centradas en el control de flujo. El lenguaje de modelado proporciona elementos notacionales y mecanismos de apoyo a aspectos tales como la especificación de las actividades, el flujo de secuencia, el flujo de datos y los eventos; en cambio, no soporta especificaciones de aspectos o requisitos no funcionales. Por otra parte, la ejecución de los procesos de negocio, además de la selección de una plataforma de ejecución, como por ejemplo SOA, requiere de otros lenguajes. Por ejemplo el lenguaje “*Business Process Execution Language (BPEL)*” es utilizado para la ejecución de los procesos de negocio a través de la composición de servicios por orquestación [AMK+09]. En la actualidad, la mayoría de los enfoques de GPN enfatizan el desarrollo de un proceso de negocio, y a

---

<sup>2</sup> Unified Modeling Language

partir de este mismo se derivan muchas variantes, que son especializadas según diversas necesidades de la organización. Los procesos de negocio pueden tener piezas comunes para un grupo o familia, de diferentes casos de aplicación, implicando cierta variabilidad en la selección de las actividades involucradas en el proceso. Este aspecto podría ser utilizado para LPS, donde una funcionalidad común puede ser desarrollada como un activo reutilizable para la creación de nuevas variantes de los procesos de negocio en un dominio [CK10] [AMK+09].

### ***B. Gestión de la variabilidad en LPS y SOA***

La *Gestión de la Variabilidad* o “*Variation Management*” [CK10] se aplica a LPS y debe ser aplicada también a SOA; los puntos de variación [PBL05] pueden ser implementados en LPSOS ya sea por un único servicio compuesto (donde una interfaz de servicios puede ofrecer parametrización por QoS (o algún otro mecanismo) o a través de servicios similares que traten cada variante y que son luego integrados en un servicio compuesto como un punto de variación. En [GAT13] se indica que la variabilidad es manejada agrupando varios servicios que tienen la misma funcionalidad, es decir similitud en las tareas que desempeñan, pero difieren en su QoS.

### ***C. Arquitecturas de referencia en LPS y SOA***

Otro aspecto importante corresponde a la construcción de la AR, y en particular a las *Arquitecturas de Referencia Orientadas a Servicios (AROS)* que apoyen la reutilización sistemática de los activos principales para la configuración de productos de software concretos bajo un contexto LPS. Existen diferencias entre *Arquitecturas de LPS (ALP)* y AR: ALP tienden a ser menos abstractas que las AR, pero más abstractas que las arquitecturas de software concretas; es decir, las ALP son un tipo de AR [GAT13]. Con respecto a nuestra RDS [HLM14] que será estudiada en el Capítulo III, la AR y la ALP son consideradas similares. Las AROS se han propuesto como un tipo de AR; lo que debe estar claro es que una AROS en el contexto SOA es un marco o modelo de referencia para una arquitectura empresarial para la integración de aplicaciones en la empresa, no es una arquitectura de software

“real” con componentes, conectores y comportamiento, en el sentido de [SG96]. En cambio una AROS en el contexto LPS es una arquitectura de software abstracta, genérica e instanciable a partir de un modelo de variabilidad, para generar sistemas concretos orientados a servicios de la familia LPS. Una AROS-LPS será entonces para nosotros una AR en un contexto de LPSOS. Varias AROS bajo SOA han sido propuestas para sistemas de gobierno, entornos de trabajo colaborativo, o sistemas de atención de la salud; generalmente no son AROS “puras” solo conformadas por servicios, sino que incluyen componentes que no son servicios. Una RDS realizada sobre las AROS bajo LPS en [DF13] indica que faltan directrices sobre cómo definir las AROS, y en particular como definir las AROS que soporten un alto grado de variabilidad en este contexto [GAT13].

#### ***D. Gestión de la calidad en Servicios***

En los actuales momentos no se concibe un sistema de software que no tenga la calidad necesaria que satisfaga las expectativas de los usuarios finales. Más aun, las propiedades de calidad para una LPSOS son fundamentales para garantizar su completitud y carácter evolutivo. Como ya se mencionó, es vital abordar las características de calidad de estos sistemas desde una etapa temprana en el ciclo de vida del desarrollo de software. En nuestro caso, esto se debe hacer en el ciclo de vida de la Ingeniería del Dominio en el proceso de una *ILPS Orientada a Servicios (ILPSOS)*. Los servicios exponen propiedades de calidad de bajo nivel a las que responden en tiempo de ejecución, para intercomunicar con otros servicios; los valores de los QoS se ofrecen a través de parámetros. En general, lo que ocurre con las propiedades de calidad de alto nivel a las cuales también deben responder los servicios compuestos, cuando representan componentes funcionales en el caso de AROS, no se trata en detalles y no queda clara su trazabilidad respecto a la funcionalidad. Los *Servicios Web* o “*Web Services*” (*WS*) son tipos de servicios que pueden ser descubiertos, especificados y accedidos utilizando XML<sup>3</sup> y protocolos Web estándar. El elemento central de un WS está en la definición estándar de su

---

<sup>3</sup> eXtensible Markup Language

interfaz en WSDL<sup>4</sup>, que contiene entre otras informaciones, valores sobre los QoS inherentes al servicio, como por ejemplo tiempo de respuesta, máxima capacidad, etc. La descripción de un WS tiene dos partes: - una definición abstracta que describe el servicio como un componente, con su interfaz (donde se ofrecen también los QoS) y operaciones; - una definición que describe los enlaces concretos de las operaciones hacia puntos de acceso, que contienen una descripción de los datos, una dirección física y la información del protocolo de comunicación [GB08]. En un entorno de una aplicación que utiliza WS, se distinguen tres funciones: intermediario, proveedor y consumidor de servicios. El proveedor registra un servicio en el intermediario o “*bróker*”, por ejemplo en un repositorio UDDI<sup>5</sup>. El consumidor descubre un servicio gestionado por el intermediario y llama al servicio del proveedor. Todas las partes utilizan protocolos tipo SOAP<sup>6</sup>, un protocolo estándar basado en XML para el manejo de respuestas y solicitudes de WS [GB08]. Está claro que un WS, como componente de una aplicación, también debe responder a propiedades de calidad de alto nivel, las cuales no necesariamente son observables en ejecución, por ejemplo persistencia de la información y políticas de confidencialidad; este tipo de propiedades de calidad de alto nivel es la que se tratarán a nivel de diseño arquitectónico. En cambio las propiedades de calidad de bajo nivel inherentes a los QoS, se consideran durante el proceso de combinación de servicios, en el momento de evaluar la selección de un servicio particular a ser combinado y no serán tratadas en este trabajo.

### ***E. Dominio de aplicación***

El dominio de la LPSOS que se considera en esta tesis es aplicado a los *Sistemas de Información Integrados de Salud (SIS)* [LOS15], que son sistemas intensivos por su complejidad y restricciones de seguridad, interoperabilidad y disponibilidad. Son sistemas de información cuya arquitectura está organizada clásicamente en estilo capas y en estilo basado en eventos, bajo el modelo de

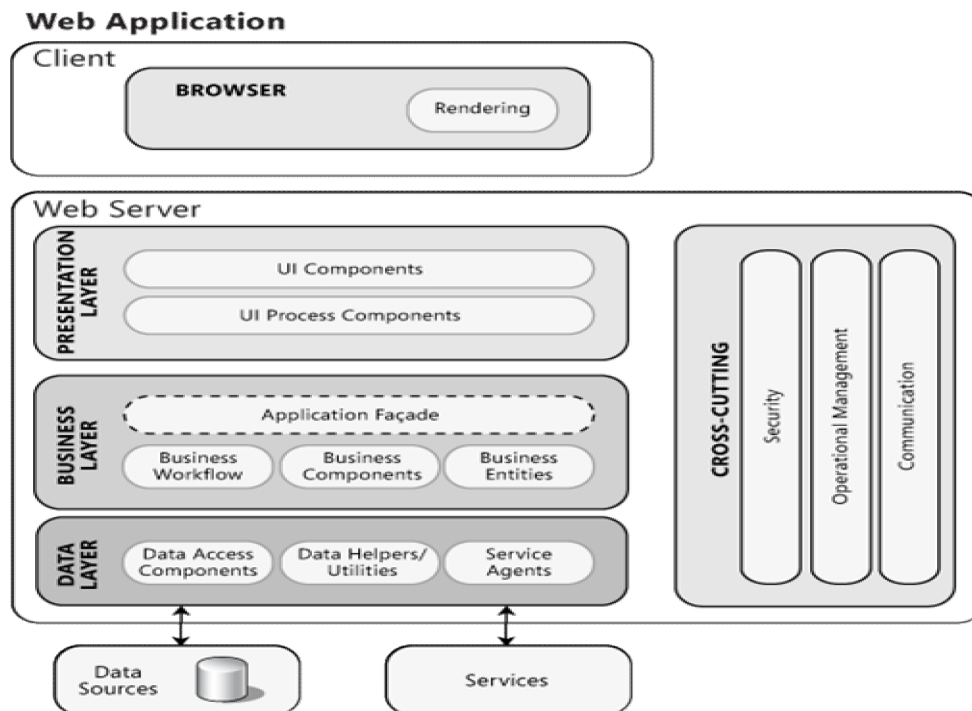
---

<sup>4</sup> Web Services Definition Language

<sup>5</sup> Universal Description, Discovery and Integration

<sup>6</sup> Simple Object Access Protocol

distribución cliente-servidor; SOA<sup>7</sup> se utiliza para la comunicación transversal entre capas y con el entorno de red, lo cual lo convierte en un estilo híbrido eventos/SOA/Capas, ver Figura 1, tomada de Wikipedia.



**Figura 1.** Arquitectura de estilo híbrido eventos/SOA/Capas [HLO16a]

## 1.2 Objetivo de la tesis

Por lo discutido en la sección anterior en lo que sigue se destaca el objetivo de la presente Tesis Doctoral.

En vista de los avances del trabajo obtenidos en [HLM14], [HLM+14] y de la consideración de un nuevo estándar ISO/IEC 26550 [ISO15] que surge como marco de referencias de las LPS, cuyo texto pudimos obtener completo solo en 2015 [HLO15], se precisó más el alcance de la investigación, la cual en el objetivo original aprobado en el proyecto [Her15] se concentraba solamente en la etapa de Análisis del Dominio. En consecuencia, el nuevo objetivo general de la tesis es:

<sup>7</sup> Service-Oriented Architecture, <http://www.w3.org/TR/wsa-reqs>



*“Definir un Proceso de Ingeniería del Dominio para Líneas de Productos de Software Orientadas a Servicios, considerando aspectos de calidad del software”* el cual amplía el objetivo original [Her15] *“Definir un Proceso de Análisis del Dominio para el desarrollo de Líneas de Productos Orientadas a Servicios, considerando aspectos de calidad del software”*, limitado solo a la etapa de Análisis del Dominio, incluida en el ciclo de Ingeniería del Dominio.

Para alcanzar el nuevo objetivo, se plantean los siguientes objetivos específicos o parciales:

### **1.2.1 Objetivos específicos**

1. Realizar una Revisión Documental Sistemática (RDS) [Kit07] del enfoque de Líneas de Productos de Software y modelos de referencias. Los resultados para el logro de este objetivo se presentaron en [HLM14].
2. Realizar una Revisión Documental Sistemática (RDS) bajo los lineamientos propuestos en [Kit07] del enfoque de LPSOS y métodos de desarrollo. Para alcanzar este objetivo se realizaron dos RDS cuyos resultados se presentaron en [HLM+14] [HLO16c] (ver Capítulo III).
3. Estudiar ontologías en el ámbito de LPSOS, para modelar los conceptos del dominio y la arquitectura de referencia.

Se discute en el Capítulo II el trabajo [LO16], en el cual una AR para LPS en el dominio SIS se representa mediante la ontología HIS-RA Ontology. Aún considerando su factibilidad, en esta investigación no se trataron representaciones ontológicas para AROS-LPS, dejándose este punto como perspectiva para un futuro trabajo.

4. Adaptar la fase de Alcance del Dominio del nuevo estándar ISO/IEC 26550 [ISO15] considerando también el modelo del negocio y aspectos de calidad. Los resultados para el logro de este objetivo se contemplan en [HLO15] [HLO+16].

5. Definir fases, actividades y artefactos involucradas en el Proceso de ID. Este objetivo se logró completamente en los trabajos [HLO15] [HLO16a] [HLO16b] [HLO+16] [HLO16d] (ver Capítulo IV y Capítulo V).

6. Definir como caso de estudio el dominio de los Sistemas de Información Integrados de Salud y validar el proceso.

Este objetivo también se alcanzó plenamente con los trabajos [HLO16a] [HLO16b] [HLO+16] [HLO16d] (ver Capítulo VI).

### 1.3 Identificación de los problemas a resolver para conseguir el objetivo

Ahora bien, de la discusión realizada en la Sección 1.1 y en los numerosos trabajos analizados en la literatura en el contexto de LPSOS, para la construcción de una AROS, deben manejarse en particular tres problemas principales:

- *Representación de la AR respecto a servicios*: definir una representación de la AR, conformada por componentes y conectores [SG96] genéricos, es decir instanciables, en la cual los componentes son servicios y los conectores son mensajes entre servicios, incluyendo la correspondencia entre servicios y componentes arquitecturales;
- *Satisfacción de las propiedades de calidad por las respectivas funcionalidades que las requieren representadas por servicios*: la gestión de las propiedades de calidad requeridas por los componentes que representan funcionalidades para asegurar en la AR la completitud (todas las funcionalidades prioritarias están presentes), la idoneidad funcional (las funcionalidades cumplen a cabalidad las tareas para las cuales fueron diseñadas), la idoneidad no funcional (las propiedades de calidad requeridas por las funcionalidad son satisfechas) y la evolución (capacidad de ser modificada en el tiempo);
- *Gestión de la variabilidad en la AR con servicios*: debe ser manejada la variabilidad funcional y no funcional.

Las soluciones a estos tres problemas determinarán los pasos a seguir para definir el Proceso de ID planteado en esta Tesis, en el cual se destaca un método de diseño de una AROS en un contexto LPS.

#### **1.4 Metodología de investigación**

Con fin de alcanzar los objetivos planteados, se desarrolló un proceso de investigación basado en [Hur07], partiendo de una revisión documental, de tal manera de recopilar las posturas teóricas que permitieron sustentar la investigación (etapa de *Investigación exploratoria*); continuando con una etapa de *investigación descriptiva* y finalizando con la etapa de *Investigación proyectiva*. Según [Hur07], la investigación descriptiva tiene como objetivo la descripción precisa del evento de estudio y está asociada al diagnóstico de la situación; la *investigación proyectiva* implica explorar, describir, explicar y proponer alternativas de cambio, más no necesariamente ejecutar la propuesta.

El proceso de investigación como tal, se dividió en cuatro grandes etapas, incluyendo una etapa final de validación de resultados, que permitieron alcanzar en forma metódica el objetivo general propuesto.

Una primera etapa de *investigación exploratoria* documental; se justificó en la realización de tres RDS [HLM14] [HLM+14] [HLO16c], siguiendo la técnica de [Kit07], en busca de antecedentes que dieran sustento teórico sobre LPS, SOA, LPSOS, procesos y métodos de Ingeniería de Dominio, técnicas de modelación del dominio y de su variabilidad, en particular para la realización la primera fase de la ID, el Alcance del Dominio, y finalmente la especificación de RNF. En esta etapa se lograron los objetivos 1, 2 completamente; en cuanto al objetivo 3, solo se estudió una ontología para AR-LPS, dejándose el caso de la AROS-LPS para futuras investigaciones.

Una segunda etapa de *investigación descriptiva*, se basa en la comparación de los resultados obtenidos a partir de las tres RDS realizadas y otras recientes RDS encontradas en la literatura; se identificaron los diferentes modelos para la

especificación de la variabilidad para la fase de Ingeniería de Requisitos del Dominio, según ISO/IEC 26550 y en particular aquellos modelos que contemplan RNF. Como parte de esta etapa descriptiva se consideró el análisis de los resultados de las tres RDS [HLM14] [HLO15] [HLO16c] contemplados en los objetivos 1 y 2.

Una tercera etapa de *investigación proyectiva* donde se adaptó la fase de Alcance de LPS [ISO15] planteada en el objetivo 4, considerando en [HLO16a] y [HLO+16] la calidad como una nueva faceta y el modelado del negocio según [Bj06]. Además se definieron dos procesos de ID, *QuaDRA* [HLO16b] y *WSRA-SPL* [HLO16d], para cumplir con el objetivo 5. QuaDRA es un Proceso de ID donde se diseña una AR para LPS dirigida por la calidad, que combina dos enfoques de desarrollo de LPS en la fase de Alcance [ISO15]: el proactivo (top-down) en el Alcance del Dominio inspirado en [Bj06] y el extractivo (bottom-up), con técnicas de reingeniería, mediante la refactorización de arquitecturas de productos existentes representadas por grafos conexos no dirigidos, inspirado en [LOE15], durante el Alcance del Portafolio de Productos, para así reducir el esfuerzo de de las fases subsiguientes.

En nuestro contexto "refactorización" se refiere a realizar modificaciones (agregar nuevos componentes funcionales o no funcionales o agrupar componentes existentes en una arquitectura), manteniéndose siempre los objetivos (funcionales o no funcionales) de los componentes, ubicados siempre en una configuración arquitectónica válida (grafo conexo y no dirigido). Es la forma de como se combinan los componentes semánticamente "similares" en una sola componente a la cual se le puede asignar un nombre diferente, por ejemplo. El objetivo funcional de esa agrupación de componentes similares en un nuevo componente, no cambia, pero la topología de la arquitectura si se transforma.

WSRA-SPL es un proceso de ID para diseñar una AR para LPSOS, con bases en el primer proceso QuaDRA. Ambos procesos contemplan el modelo de variabilidad del dominio, a partir de una primera arquitectura candidata obtenida automáticamente mediante un análisis previo de similitud semántica de componentes

arquitecturales de productos existentes (enfoque de refactorización bottom-up extractivo, inspirado en [LOE15] y completada utilizando diferentes técnicas encontradas en la etapa de investigación descriptiva, para obtener así una AROS evolutiva que responda a la calidad exigida por el dominio de las LPSOS.

Finalmente, en la cuarta etapa de *validación de resultados*, respondiendo al objetivo 6, se aplicaron los dos procesos, QuaDRA, para construir un AR-LPS, y WSRA-SPL, para construir una AROS-LPS, para el dominio de los SIS con el propósito de validar la propuesta mediante este caso de estudio en el dominio de los SIS.

Como productos de esta investigación se obtuvieron resultados parciales relevantes, que fueron publicados y que muestran la evolución del trabajo; el texto completo de estos trabajos puede verse en el Apéndice, sección Publicaciones; éstos fueron los siguientes:

#### **1.4.1 Resultados parciales obtenidos para alcanzar los objetivos específicos durante el proceso metodológico de investigación**

- *Revisión Documental Sistemática de Enfoques y Técnicas para la Construcción de Arquitecturas en un Contexto de Líneas de Productos de Software*, por Herrera, Losavio y Matteo y Losavio [HLM14]. Artículo “in extenso” publicado en la Revista Venezolana de Computación ReVeCom (ISSN: 2244-7040) – SVC (Sociedad Venezolana de Computación), Vol. 1, No. 1, pp. 17-25. Junio 2014. Selección de los Mejores Artículos de CoNCISa (Conferencia Nacional de Computación, Informática y Sistemas), Puerto Azul, Naiguatá, Venezuela 2013.

El objetivo de esta RDS es identificar aspectos claves que deben estar presentes en un proceso general de diseño de una AR, por lo que se hizo una RDS [Kit07] de la literatura referente a los enfoques, modelos, métodos y técnicas vigentes. Finalmente, a partir del estudio realizado, se obtuvo una primera propuesta para la construcción de una arquitectura para LPS. Las LPS son un enfoque centrado en el desarrollo de software basado en la reutilización, mediante el proceso de

ingeniería de líneas de productos y particularmente en la fase de ingeniería del dominio es donde se diseña la arquitectura de referencia y/o la arquitectura de líneas de productos, para ello dos enfoques son utilizados en la industria, partiendo de un amplio conocimiento del dominio de la aplicación se encuentra el enfoque proactivo y partiendo de los sistemas existentes se encuentra el enfoque reactivo, es de particular interés identificar los artefactos utilizados en las estrategias utilizadas por ambos enfoques diferenciándolos ya sea por el tipo de reutilización seleccionado o por la incorporación de nuevos artefactos en cada uno de los métodos propuestos, en consecuencia una revisión documental sistemática de los estudios que abordan el tema es indispensable para determinar los alcances y limitaciones de las propuestas encontradas.

- *Revisión Documental Sistemática en el Ámbito de la Ingeniería del Dominio para Líneas de Productos de Software Orientados a Servicios*, por Herrera, Losavio, Matteo y Ordaz [HLM+14]. Artículo “in extenso” publicado en la Revista Venezolana de Computación ReVeCom (ISSN: 2244-7040) – SVC, Vol. 1, No. 2, pp. 23-33. Diciembre 2014. Selección de los Mejores Artículos de CoNCISa, UCAB, Caracas, Venezuela, 2014.

El objetivo principal del trabajo fue realizar una RDS en el tema de ID para LPSOS, quien combina los enfoques de LPS y de SOA. Esta investigación pretende determinar trabajos recientes existentes en la literatura en cuanto a actividades, artefactos y técnicas para un proceso de Desarrollo del Dominio, según Björner [Bjö06] y de Análisis del Dominio, según Pohl et al. [PBL05]. Como resultado de esta revisión se detectó, combinando los enfoques mencionados, que las fases principales de un proceso general de ID para LPSOS son: análisis, diseño e implementación, inspiradas en las fases definidas en [PBL05] para la ID en un contexto de LPS. Dentro de estas fases se identificaron las actividades, artefactos y técnicas más usadas en la práctica. Estos resultados fueron utilizados para definir un proceso sistemático de Análisis del Dominio para LPSOS, que además contemplara el tratamiento de requisitos de calidad, aspecto relevante para un contexto de

producción industrial de software y no tratado sistemáticamente en los enfoques revisados.

- *Ingeniería del Dominio con el Estándar ISO/IEC 26550 para Líneas de Productos de Software Considerando la Faceta Calidad*, por Herrera, Losavio y Ordaz [HLO15]. Trabajo “in extenso” en actas de CoNCISa (Conferencia Nacional de Computación, Informática y Sistemas), Universidad de Carabobo (UC), Valencia, Venezuela, 2015.

El objetivo de este trabajo de investigación, enmarcada en la ID, primer ciclo de vida de la Ingeniería de la LPS (ver Figura 1), es describir el nuevo estándar ISO/IEC 26550 [ISO15] que define un Modelo de Referencia para LPS e incorporar en la definición de su primera fase, el Alcance de la LPS, el concepto de *faceta* de Bjørner [Bjø06], o conjunto finito de formas genéricas de modelar un dominio; cada faceta representa una vista particular del dominio; la unión de estas vistas constituye el dominio completo. Un aporte de este trabajo es incluir aspectos de calidad del producto de software como una nueva faceta; la calidad en LPS es considerada por el nuevo estándar como crucial para el diseño de una AR, sin embargo es difícil de atacar y no se especifican técnicas precisas para hacerlo [ISO15] [HLO+16]; en nuestro artículo la calidad del producto es especificada por el estándar ISO/IEC 25010 [ISO11] y debe ser considerada explícitamente, para garantizar que la arquitectura de referencia, artefacto clave de la LPS, tenga la calidad adecuada como arquitectura evolutiva.

- *Product Line Scoping for Healthcare Information Systems using the ISO/IEC 26550 Reference Model. Alcance de Líneas de Productos para Sistemas de Información de Salud usando el Modelo de Referencia ISO/IEC 26550*, por Herrera, Losavio y Ordaz [HLO16a]. Artículo “in extenso” publicado en la Revista Venezolana de Computación ReVeCom (ISSN: 2244-7040) – SVC (Sociedad Venezolana de Computación), Vol. 3, No. 1, pp. 38-50. Julio 2016. Selección de los Mejores Artículos del IV Simposio Científico y Tecnológico en Computación (SCTC

2016) (ISBN: 978-980-12-8407-9), pp. 33-46, Escuela de Computación, Facultad de Ciencias, UCV, Caracas, Venezuela.

El objetivo de este trabajo es presentar un proceso para determinar el alcance de una LPS; es decir, se quieren determinar sus limitaciones y posibilidades respecto a los productos que de ella pueden ser derivados, a partir de los activos de software reutilizables y necesarios para la producción y garantizar así su evolución y los posibles riesgos económicos, en un dominio dado. La fase de Alcance es la primera del ciclo de vida ID de la ILPS (ver Figura 1). El proceso propuesto para determinar el alcance de la SPL, denominado *PLScoP*, es una adaptación del proceso general *PL Scoping* del nuevo estándar ISO/IEC 26550 [ISO15], el cual como ya se mencionó, describe un modelo de referencia o framework para la ILPS. *PLScoP* se centra en considerar temprano en la ILPS la calidad del software y está dirigido a reducir el esfuerzo de desarrollo de las subsiguientes fases de la ID, es decir la Ingeniería de Requisitos del Dominio, donde se captura el modelo de variabilidad y el Diseño del Dominio, donde se construye la AR; debe notarse que es en la ID donde se centra el mayor esfuerzo de desarrollo de la LPS. En este trabajo, el paso Alcance del Dominio de *PLScoP* será aplicado al dominio de los SIS como caso de estudio.

- *Product Lines Scoping using ISO/IEC 26550 Reference Model considering Software Quality*, por Herrera, Losavio, Ordaz y Bøegh [HLO+16]. Artículo en ingles, “in extenso”, presentado por J. Bøegh y publicado en las Actas de la International Conference on Applied Social Science and Information Technology (ASSIT2016) July 24-25, 2016, Bangkok, Thailand. Fue publicado bajo el nombre de otra conferencia de mayor alcance y prestigio (2016 2nd International Conference on Humanity and Social Science (ICHSS 2016) pp. 194-200). Published by the DEStech Publications and indexed in Thomson Reuters Web of Science CPCI-S (ISTP).

En este trabajo tres enfoques para el desarrollo de la ID fueron comparados, el clásico ID de Björner (2006), la ILPS para ID de Pohl et al (2005) y el marco de referencia ISO/IEC 26550 (2013) para ILPS. El presente trabajo también se enmarca en el ciclo de vida de la ID de ILPS, donde los mayores esfuerzos se centran en la



construcción de la AR de la LPS y el Repositorio de Activos. El objetivo principal es introducir técnicas para la sub-fase de *Alcance del Dominio* o “*Domain Scoping*”, de la fase Alcance de la Línea de Productos (*Product Line Scoping, PLScop*), primera fase de la ID, definida en el ISO/IEC 26550 [ISO15] (ver Figura 1). Según el nuevo estándar, la consideración de la calidad del software es crucial para el éxito de la LPS, sin embargo no se especifican técnicas precisas para hacerlo y se introduce solo en la fase DD, muy tarde en el desarrollo; en nuestro trabajo la calidad fue especificada como una nueva faceta intrínseca de Bjørner. El proceso de PLScop para el Alcance del Dominio es descrito e ilustrado con un caso de estudio en el dominio de los SIS.

- *QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines based on ISO/IEC 26550*, por Herrera, Losavio y Ordaz [HLO16b]. Artículo en inglés, “in extenso”, publicado en la Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6(1), pp. 20-38, enero-junio 2016.

El objetivo de este trabajo, también enmarcado en el primer ciclo de vida de la ID, de la ILPS, es presentar y aplicar el proceso QuaDRA, sistemático y repetible para el diseño de una AR, adaptando los lineamientos del estándar ISO/IEC 26550 que define un Modelo de Referencia para la ILPS. La AR es el activo principal compartido por todos los productos de una familia LPS; abarca las partes comunes y variables de la familia y es usada como una plantilla para producir nuevos productos en el segundo ciclo de vida de la ILPS, la Ingeniería de la Aplicación (ver Figura 1). QuaDRA se considera un enfoque de desarrollo proactivo o top-down de LPS ya que sigue la ILPS; además es orientado a la calidad; ésta se toma en cuenta desde la primera fase de Alcance de la LPS en la ID, proporcionando así clara trazabilidad de los requisitos de calidad en todo el proceso, siendo éstos especificados como descriptores intrínsecos para todas las facetas del dominio, garantizando el carácter evolutivo de la AR. Los requisitos de calidad, especificados aquí por ISO/IEC 25010 [ISO11], en general son poco considerados en los enfoques ILPS, siendo dejados para la fase final de diseño de la AR [HLO+16]; sin embargo son los mayores

responsables de la variabilidad de la LPS. El estándar ISO/IEC 26550 promueve la primera fase de la ID, Alcance de la LPS o *PLScop*, que incluye las sub-fases Alcances del Portafolio, del Dominio y de Activos, donde los productos, los riesgos y la factibilidad económica de la LPS deben ser evaluados. QuaDRA combina dos enfoques de desarrollo de LPS en la fase de Alcance: el proactivo (top-down) en el Alcance del Dominio y el extractivo (bottom-up), con técnicas de reingeniería, durante el Alcance del Portafolio de Productos, para así reducir el gran esfuerzo de desarrollo en las subsiguientes actividades de ID. Por lo tanto nuestro enfoque contribuye globalmente a asegurar la calidad de la AR y a reducir el esfuerzo de desarrollo. QuaDRA será aplicado a un caso de estudio en el dominio de los SIS.

- *Diseño de Arquitecturas de Referencia para Líneas de Productos de Software Orientadas a Servicios: Revisión Documental Sistemática*, por Herrera, Losavio y Ordaz [HLO16c]. Artículo “in extenso” publicado en la Revista Venezolana de Computación ReVeCom (ISSN: 2244-7040) – SVC (Sociedad Venezolana de Computación), Vol. 3, No. 2, pp. 13-25. Diciembre 2016. Selección de los Mejores Artículos de CoNCISa (Conferencia Nacional de Computación, Informática y Sistemas), Colegio Universitario de Caracas (CUC), Caracas, Venezuela 2016.

El objetivo del trabajo es realizar una RDS en el tema de Líneas de Productos de Software Orientadas a Servicios (LPSOS), el cual integra los enfoques de Líneas de Productos de Software (LPS) y de Arquitecturas Orientadas a Servicios (SOA). La intención es que a partir de la RDS se identifiquen métodos (fases, etapas, actividades, artefactos y roles) para el diseño del artefacto principal de las LPS, la AR siguiendo un modelo de arquitectura empresarial SOA y conformada por Servicios Web. Los resultados de la RDS servirán de guía para definir un proceso sistemático para el diseño de una AR en el contexto LPSOS. Esta arquitectura contemplará el tratamiento de requisitos de calidad en las etapas tempranas del desarrollo de LPSOS para garantizar su evolución, y el modelado de la variabilidad para la derivación de productos concretos, aspectos esenciales en el contexto de producción industrial de software.

- Web-services reference architecture for software product lines: A quality-driven approach por Herrera, Losavio y Ordaz [HLO16b]. Artículo en inglés “in extenso”, publicado en la Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6(2), pp. 7-21, julio-diciembre 2016.

Las Líneas de Productos de Software (LPS) y las Arquitecturas Orientadas a Servicios (SOA), son dos enfoques para el desarrollo de software que favorecen la reutilización, en lugar de re-desarrollar nuevos sistemas. El objetivo principal del trabajo es explotar los beneficios de ambos enfoques e integrarlos en un único proceso de diseño arquitectural para LPS, denominado *WSRA-SPL: Web-Services Reference Architecture for SPL*, centrado en asegurar la calidad del software en etapas tempranas del ciclo de vida del desarrollo del software y la construcción de la AROS-LPS.

Ahora bien, para ilustrar de manera precisa todo el proceso de investigación llevado a cabo y los resultados obtenidos en esta Tesis Doctoral, se estructura el presente documento en siete capítulos:

En este **primer capítulo** se ofreció una introducción general de la investigación, el planteamiento del problema, los objetivos a alcanzar, la metodología de investigación utilizada, así como los productos obtenidos durante la investigación.

Un **segundo capítulo** contempla los principales fundamentos teóricos sobre los cuales se sustenta la presente investigación. Los siguientes aspectos son abordados: Líneas de Productos de Software (LPS), Ingeniería de LPS (ILPS), El Modelo de Referencia para LPS del estándar ISO/IEC 26550; Líneas de Productos de Software Orientadas a Servicios (LPSOS); Calidad del Software y calidad del producto de software según el estándar ISO/IEC 25010; Dominio de los Sistemas de Información Integrados de Salud (SIS); Ontologías en el desarrollo de LPS; y Revisión Documental Sistemática (RDS).

En el **tercer capítulo** se presentan trabajos relacionados sobre los métodos de desarrollo de LPS y LPSOS en el ciclo de vida de la ID. Para los métodos LPS se

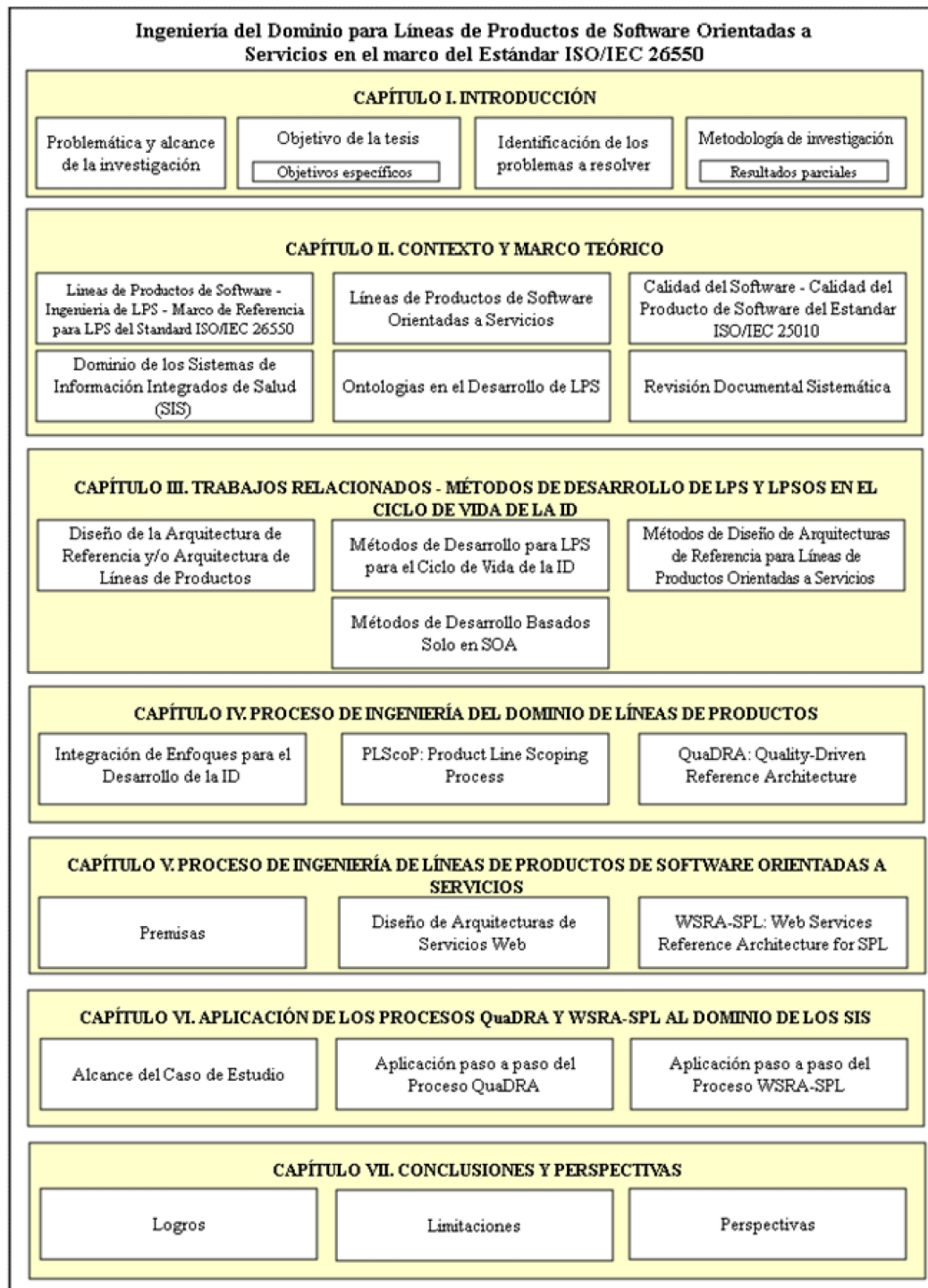
presentan los resultados de dos RDS [HLM+14] [HLO16c] que esbozan las fases, actividades, técnicas y artefactos utilizados; y el Proceso RA&NFR-VAR: Proceso Bottom-Up de Modelado de la Variabilidad de RF y RNF para la Construcción de una AR [LOE15], como antecedente directo de nuestro trabajo. Para los métodos de desarrollo basados solo en SOA, se presenta una tabla resumen donde son comparados: IBM/RUP SOMA (Service-oriented Modeling and Architecture), Marco Arquitectural Orientado a Servicio (SOAF), Metodología de Papazoglou, Metodología de Thomas Erl, Proceso Unificado Orientado a Servicio (SOUP). Finalmente, para los métodos de desarrollo LPSOS se presentan los resultados de una RDS donde se caracterizan a través de un marco de evaluación único los métodos de desarrollo.

En el **cuarto capítulo** se presenta la especificación del proceso completo de Ingeniería de Líneas de Productos (ILPS), QuaDRA [HLO16b].

En el **quinto capítulo** se presenta la especificación del proceso completo de ID para LPSOS, WSRA-SPL [HLO16b].

En el **sexto capítulo** se aplican los procesos QuaDRA y WSRA-SPL al dominio de los Sistemas de Información Integrados de Salud (SIS).

Por último, el **séptimo capítulo** es dedicado a las conclusiones, destacándose los aportes del trabajo, las limitaciones y las perspectivas, como líneas abiertas de investigación en el área. Finalmente se encuentran los apéndices respectivos. Nótese que cada capítulo tiene sus respectivas referencias bibliográficas. La Figura 2 ilustra gráficamente la organización del documento.



**Figura 2.** Estructura gráfica de la Tesis

# Referencias

---

- [ABK+13] S. Apel, D. Batory, C. Kästner, and G. Saake, *Feature-Oriented Software Product Lines*. Springer, 2013.
- [AG13] M. Abu-Matar, and H. Gomaa, “An Automated Framework for Variability Management of Service-Oriented Software Product Lines”. In *Service Oriented System Engineering (SOSE)*, IEEE 7th International Symposium on (pp. 260-267). IEEE, 2013.
- [AMK+09] M. Asadi, B. Mohabbati, N. Kaviani, D. Gašević, M. Bošković, and M. Hatala, *Model-driven development of families of Service-Oriented Architectures*, In *Proceedings the 1th International Workshop on Feature-Oriented Software Development*, pp. 95-102, ACM, 2009.
- [Ber92] E. Berard, *Essays in Object-Oriented Software Engineering*, Nueva York: Prentice Hall, 1992.
- [Bjø06] Bjørner, D. (2006). *Software Engineering 3 Domains, Requirements, and Software Design*. Texts in Theoretical Computer Science. An EATCS Series. Editors: W. Brauer G. Rozenberg A. Salomaa. Springer-Verlag Berlin Heidelberg.
- [Bos00] Bosch J. (2000) *Design and Use of Software Architectures – Adopting and evolving a product-line approach*, Addison-Wesley
- [CK10] S. Cohen, and R. Krut, *Managing variation in services in a software product line context* (No. CMU/SEI-2010-TN-007), Carnegie-Mellon Univ Pittsburgh PA Software Engineering Institute, 2010.
- [CN01] Clements P. and Northrop L. (2001) *SPL: practices and patterns*, 3rd ed. Readings, MA, Addison Wesley.
- [DF13] R. Dos Santos, and M. Fantinato, “The use of software product lines for business process management: A systematic literature review,” *Information and Software Technology* 55(8), pp. 1355-1373, 2013.
- [EMA13] A. Ezenwoke, S. Misra, and M. Adigun, “An Approach for e-Commerce On-Demand Service-oriented Product line Development”, *Acta Polytechnica Hungarica*, 10(2), 2013.
- [GAT13] M. Galster, P. Avgeriou, and D. Tofan, “Constraints for the design of variability-intensive service-oriented reference architectures—An industrial case study,” *Information and Software Technology*, 55(2), pp. 428-441, 2013.
- [GB08] S. Günther, and T. Berger, “Service-oriented product lines: towards a development process and feature management model for web services,” In *SPLC (2)*, pp. 131-136, October 2008.

- [Gon12] González H., J. (2012). Integration of Quality Attributes in Software Product Line Development. Master Thesis, Tesina de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información (ISMFSI).
- [Her15] Herrera R., Juan C., Análisis del Dominio para Líneas de Productos de Software Orientadas a Servicios. Proyecto Aprobado por la Comisión de Estudios de Postgrado N° CmEPG: 018-2015 (28-1-2015). Aprobado en sesión del 24/02/2015. Facultad de Ciencias. UCV.
- [Hur07] Hurtado, J. (2007). Metodología de la Investigación. Una Comprensión Holística. Ediciones Quirón-Sypal. Caracas.
- [HHJ08] A. Helferich, G. Herzwurm & S. Jesse, Software Product Lines and Service-Oriented Architecture: A Systematic Comparison of Two Concepts, Proceedings of the First Workshop on Service-Oriented Architectures and Software Product Lines, May 2008, CMU/SEI-2008-SR-006
- [HLM14] J. Herrera, F. Losavio, and A. Matteo, RDS de enfoques y técnicas para la construcción de arquitecturas en un contexto de líneas de productos de software, Revista Venezolana de Ciencias de la Computación ReVeCom, Vol. 1, No. 1, pp. 17-25, Junio 2014.
- [HLM+14] J. Herrera, F. Losavio, A. Matteo, and O. Ordaz, Revisión Documental Sistemática en el Ámbito de la Ingeniería del Dominio para Líneas de Productos de Software Orientados a Servicios, ReVeCom, Vol. 1, No. 2, pp. 23-33, Diciembre 2014.
- [HLO15] J. Herrera, F. Losavio, and O. Ordaz, Ingeniería del Dominio con el Estándar ISO/IEC 26550 para Líneas de Productos de Software Considerando la Faceta Calidad, Tercera Conferencia de la Sociedad Venezolana de Computación, CoNCISa, pp. 107-118, Universidad de Carabobo (UC), Valencia, Octubre 2015.
- [HLO16a] J. Herrera, F. Losavio, and O. Ordaz, Product Line Scoping for Healthcare Information Systems Using the ISO/IEC 26550 Reference Model, ReVeCom, Vol. 3, No. 1, pp. 38-50, Junio 2016.
- [HLO+16] ASSIT 2016, J. Herrera, F. Losavio, and O. Ordaz, Product Lines Scoping using ISO/IEC 26550 Reference Model considering Software Quality, 2nd International Conference on Humanity and Social Science (ICHSS 2016), (pp. 194-200), DEStech Publications, Inc. USA.
- [HLO16b] J. Herrera, F. Losavio, and O. Ordaz, QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines based on ISO/IEC 26550, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 1, pp. 20-38, enero-junio 2016.
- [HLO16c] J. Herrera, F. Losavio, and O. Ordaz, Diseño de Arquitecturas de Referencia para Líneas de Productos de Software Orientadas a Servicios: Revisión Documental Sistemática, ReVeCom, Vol. 3, No. 2, pp. 13-25, Diciembre 2016.
- [HLO16d] J. Herrera, F. Losavio, and O. Ordaz, Web-services reference architecture for software product lines: A quality-driven approach, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 2, pp. 7-21, julio-diciembre 2016.

- [INP+09] P. Istoan, G. Nain, G. Perrouin, J. Jezequel. “Dynamic Software Product Lines for Service-Based Systems”, INRIA, France, 9th IEEE International Conference on Computer and Information Technology, 2009.
- [ISO11] ISO/IEC 25010, Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, ISO/IEC JTC1/SC7/WG6, 2011.
- [ISO15] ISO/IEC NP 26550: Software and Systems Engineering – Reference Model for Software and Systems Product Lines. ISO/IEC JTC1/SC7, 2015.
- [Ist09] Paul Istoan, Software Product Lines for creating Service Oriented Applications, Masters internship at IRISA Rennes research institute. TRISKELL research team, Rennes June 3, 2009.
- [Kit07] Kitchenham, B and Charters, S. (2007). Guidelines for Performing Systematic Literature Reviews In Software Engineering. Keele University and University of Durham, Technical report EBSE-2007-01.
- [KLR09] G. Kotonya, J. Lee, and D. Robinson, “A consumer-centred approach for service-oriented product line development,” In European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference, pp. 211-220, IEEE, 2009.
- [Kru02] C. W. Krueger, “Easing the transition to software mass customization”, In Software Product-Family Engineering. Springer Berlin Heidelberg, 2002, pp. 282-293.
- [LK06] B. List, and B. Korherr, “An evaluation of conceptual business process modelling languages,” In Proceedings of the 2006 ACM symposium on Applied computing, pp. 1532-1539, ACM, 2006.
- [LK10] J. Lee, and G. Kotonya, “Combining service-orientation with product line engineering”. Software, IEEE, 27(3), 35-41, 2010.
- [LO16] Losavio, F., and Ordaz, F. (2016) Reference Architecture Representation by an Ontology for Healthcare Information Systems Software Product Line, en el IV Simposio Científico y Tecnológico en Computación / SCTC 2016 / ISBN: 978-980-12-8407-9, UCV, Caracas, Venezuela, 2016.
- [LOE15] Losavio, F., Ordaz, O. y Esteller, V. (2015a). Refactoring-based Design of Reference Architecture, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 5, No. 1, pp. 32-48.
- [LOJ16] Losavio F., Ordaz O., Jean S., Ontological approach to derive product configurations from a Software Product Line Reference Architecture, Ciencia y Tecnología (CyT, Universidad de Palermo, Argentina), N° 16, 2016, pp. 91-127, ISSN 1850-0870.
- [LOS15] Losavio, F., Ordaz, O. y Santos, I. (2015) Proceso de Análisis del Dominio Ágil de Sistemas Integrados de Salud en un Contexto Venezolano. Enl@ce Revista Venezolana de Información, Tecnología y Conocimiento, 12 (1), 101-134
- [MSR10] Medeiros, F. M., de Almeida, E. S., & de Lemos Meira, S. R. (2010). Designing a set of service-oriented systems as a software product line. In Software Components, Architectures and Reuse (SBCARS), 2010 Fourth Brazilian Symposium on (pp. 70-79). IEEE.



- [MS10] Munir, Q., & Shahid, M. (2010). Software Product Line: Survey of Tools (Doctoral dissertation, Master's thesis, Linköping University, Department of Computer and Information Science).
- [MAG+13] Mohabbati, B., Asadi, M., Gašević, D., Hatala, M., & Müller, H. (2013). Combining service-orientation and software product line engineering: A systematic mapping study. *Information and Software Technology*, 55(11), 1845-1859.
- [MGH+13] B. Mohabbati, D. Gasevic, M. Hatala, and H. Muller, A Quality Model and Evaluation Method for Service-Oriented Software Product Line and Configurable Business Processes, *ACM Transactions on Software Engineering Methodology*, September 2013.
- [NOB11] E. Nakagawa, P. Oliveira, and M. Becker, "Exploring the use of reference architectures in the development of product line artifacts," *Proceedings of the 15th International Software Product Line Conference*, Vol 2, ACM, 2011.
- [OMG05] OMG Object Management Group. (2005). Unified Modelling Language Superstructure, version 2.0 (formal/05-07-04), August, [www.omg.org/spec/UML/2.0](http://www.omg.org/spec/UML/2.0)
- [OMG11] Object Management Group (OMG). Business Process Model and Notation (BPMN), Version 2.0. 2011.
- [PBD09] C. Parra, X. Blanc, and L. Duchien, "Context awareness for dynamic service-oriented product lines", In *Proceedings of the 13th International Software Product Line Conference* (pp. 131-140). Carnegie Mellon University, 2009.
- [PBL05] K. Pohl, G. Boeckle, and F. van der Linden, *Software Product Line Engineering - Foundations, Principles, and Techniques*, Springer Verlag, Berlin / Heidelberg, 2005.
- [QB09] P. G. G. Queiroz, and R. T. V. Braga, "Uma abordagem de desenvolvimento de linha de produtos com uma arquitetura orientada a serviços", Doctoral dissertation, Master's thesis, ICMC-Universidade de Sao Paulo, Sao Carlos, SP, orientadora: RTV Braga, 2009.
- [RLS+08] Rosen, M., Lublinsky, B., Smith, K., & Balcer, M. (2008). *Applied SOA: service-oriented architecture and design strategies*. John Wiley & Sons.
- [SG96] M. Shaw, and D. Garlan, *Software architecture: perspectives on an emerging discipline*. Prentice Hall, 1996.
- [W3C04] World Wide Web Consortium (2004). *Web Services Architecture Requirements*. W3C Working Group Note, 11 February 2004. Copyright © 2004 W3C® (MIT, ERCIM, Keio) <http://www.w3.org/TR/wsa-reqs>.

# ***CAPÍTULO II***

## ***CONTEXTO Y MARCO TEÓRICO***

---

---

## CAPÍTULO II

### CONTEXTO Y MARCO TEÓRICO

En este capítulo se discute el contexto y los principales fundamentos teóricos sobre los cuales se sustenta la presente investigación: primero se tratan las LPS y el nuevo marco de referencia ISO/IEC26550, después se ofrece una visión general sobre las LPSOS, luego se aborda la calidad del producto de software; estos tópicos fueron tratados muy brevemente en la Introducción, Capítulo I, al describir la problemática que trata de resolver esta Tesis. Finalmente se presenta el dominio de aplicación, los SIS y se abordan las técnicas para la realización de las RDS planteadas en [Kit04].

#### **2.1 Líneas de Productos de Software (LPS) – Ingeniería de LPS (ILPS) – Marco de Referencia para LPS del estándar ISO/IEC 26550**

##### ***2.1.1 Líneas de Productos de Software (LPS)***

El enfoque LPS proporciona una forma de personalización en masa mediante la construcción de soluciones individuales basadas en un repositorio de componentes de software reutilizables o activos de software. Esto introduce el individualismo en la producción de software, pero aún conserva las ventajas de la producción en masa en la cual el dominio global y los segmentos de mercado pueden ser atendidos. La necesidad para el individualismo o variabilidad, responde a diversas necesidades en el software con respecto a la funcionalidad, plataformas destino, y las propiedades no funcionales, por ejemplo, seguridad, interoperabilidad y recursos en tiempo como rendimiento y espacio de memoria [ABK+13].

Comparado con el desarrollo de productos de software individuales a partir de cero y el desarrollo de productos estandarizados de talla única para todos, las LPS prometen beneficios distintos [ABK+13], de los cuales los más importantes son los siguientes:

***Hecho a la medida (personalización).*** En lugar de proporcionar un producto estandarizado (o un pequeño conjunto de productos pre-configurados), un proveedor de software puede producir toda una serie de productos diseñados de manera diferente adaptados a las necesidades particulares de cada cliente.

***Reducción de costos.*** Mientras que proporciona a cada cliente su solución, los proveedores de la LPS no tienen que pagar el costo de diseño y desarrollo de cada producto desde cero. En lugar de ello, se desarrollan las partes reutilizables que se pueden combinar de diferentes maneras, minimizando el costo global de desarrollo del producto por cliente.

***Mejora de la calidad.*** La producción masiva industrial ha mejorado en general la calidad del producto manufacturado, porque las partes individuales estandarizadas y reutilizables pueden comprobarse de forma sistemática y ser evaluadas en muchos productos.

***Tiempo de comercialización.*** Mientras que el software estándar es fácilmente disponible, los productos de software artesanales requieren gastos significativos, y a veces es más importante el tiempo, antes de que puedan ser liberados. Una arquitectura genérica o AR bien diseñada, que debe ser evolutiva, es decir extensible para nuevos productos y cambios tecnológicos, promete la posibilidad de reaccionar rápidamente a los cambios del mercado.

Este enfoque, inspirado en las cadenas automotrices de principios del siglo '20, se inicia a principios del siglo '21 [Bos00] [CN01]; representa un gran desafío para los investigadores y desarrolladores en el área de la IS, principalmente por la tarea nada trivial de desarrollar artefactos pensando en su reutilización (activos) y reutilizar los activos existentes en la configuración de nuevos sistemas contextualizados acorde a las necesidades de los clientes [Ist09] [MSR10].

Los objetivos que persigue la *Ingeniería de LPS (ILPS)* [PBL05] son:

- ✓ Promover la reutilización a través de la explotación de los puntos en común entre los productos de la línea [HHJ08].
- ✓ Captar los conceptos esenciales de interés común y la variabilidad del conjunto de productos en un mismo dominio [INP+09].
- ✓ Promover la construcción de arquitecturas de software configurables o Arquitecturas de Referencias [MAG+13].
- ✓ Permitir la personalización masiva de los sistemas de software intensivos [MAG+13].
- ✓ Mejorar la calidad del software y reducir el tiempo de comercialización de los nuevos productos de software [PBL05].

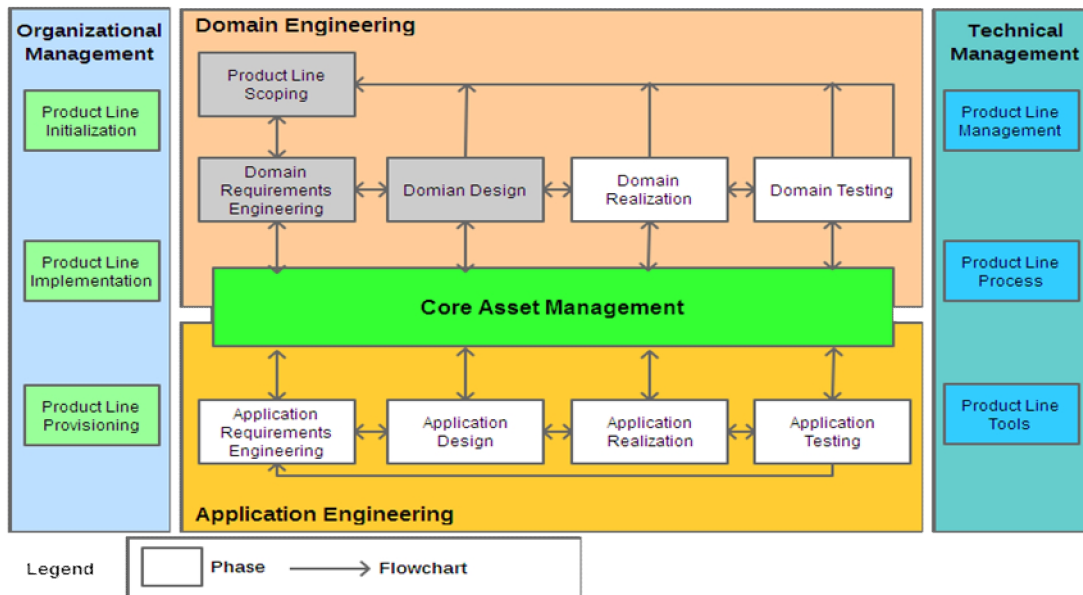
Los productos de software derivados de una LPS se distinguen en función de sus características incluidas; una característica refleja las necesidades de las partes interesadas o requisitos del cliente. Se trata de un incremento en la funcionalidad del producto y ofrece una opción de configuración [PBL05]. Esto puede implicar extender la AR si alguna de las características del cliente no están presentes, o considerar que solo pueden incluirse las características presentes en la AR.

### ***2.1.2 Marco de Referencia para LPS del estándar ISO/IEC 26550***

Es a través de la *Ingeniería de LPS (ILPS)* donde la reutilización es planificada y aplicada [PBL05]. El modelo de referencia estándar de ILPS que se muestra en la Figura 3, comprende dos ciclos de vida: la *Ingeniería del dominio (ID)* o “*Domain Engineering (DE)*” que es responsable de establecer la plataforma reutilizable y de definir las similitudes y variabilidades de la LPS y la *Ingeniería de la aplicación (IA)* o “*Application Engineering (AE)*” que se encarga de la derivación de nuevos sistemas o productos concretos a partir del marco de desarrollo establecido en la ID.

El estándar ISO/IEC 26550 proporciona un marco de referencia o “*framework*” para la ILPS, es decir, una representación abstracta de la ID y de la IA,

junto con la gestión de activos principales, la gestión organizacional y la gestión técnica de los procesos para la ILPS; este “*framework*” que se muestra en la Figura 3, constituye un marco o modelo de referencia de la ILPS, es decir una representación abstracta del ciclo de vida de toda la ILPS. Como ya se mencionó en la Introducción, incluye dos ciclos de vida principales, la ID y la IA donde se derivan los productos concretos miembros de la familia LPS a partir de la AR; el repositorio de donde se extraen los activos básicos para la derivación de los productos se construye en paralelo.



**Figura 3.** Modelo de Referencia para Líneas de Productos de Software y Sistemas del ISO/IEC 26550 [ISO15].

### A. Ingeniería de LPS (ILPS)

La ILPS [RSC+13] [ISO15], promocionada industrialmente por el *Software Engineering Institute (SEI)* de la Universidad de Carnegie-Mellon [CN01] [PBL05], se ocupa de los beneficios técnicos y económicos y alcanza la reutilización estratégica del software a través de la LPS mediante:

- La captura de características comunes y la factorización de las variaciones entre el dominio(s) o sub-dominio(s) de una LPS;

- El desarrollo de los principales activos utilizados en la construcción de los sistemas de la LPS;
- La divulgación y el hacer cumplir de manera prescrita la construcción de activos y sistemas de la LPS, y
- La evolución de los principales activos y de los productos en la LPS para mantener su aplicabilidad.
- La derivación de productos concretos a partir de la AR.

Un factor clave de éxito para la ILPS es establecer un enfoque adecuado sobre un dominio particular, bien definido y bien delimitado (ámbito) [ABK+13]. El conocimiento del dominio es factor importante para la reutilización, por lo tanto, se requiere de técnicas y métodos para capturarlo. Las LPS han sido desarrolladas para una amplia variedad de dominios, incluyendo entre otros, los sistemas operativos, sistemas de bases de datos, middleware, software de automoción, compiladores. Cuanto más amplio sea el dominio de una LPS, más grande es el número de requisitos posibles de los grupos de interés que pueden ser cubiertos en forma de productos adaptados individualmente. Sin embargo, el dominio más pequeño es el conjunto de similitudes entre los productos. Las características específicas de las LPS llevan a una separación entre la ID y la IA y entre el espacio del problema y el espacio de la solución [PBL05], [ABK+13].

La ILPS [PBL05] pretende ser una metodología para el desarrollo más rápido, a menor costo, con la mejor calidad, y con una mayor satisfacción del usuario final de productos de software y sistemas intensivos. Como resultado, se ha ganado una creciente atención mundial durante esta década hacia la ILPS. Se diferencia del desarrollo de un sistema único en dos aspectos principales [ISO15]:

1. Utiliza dos grandes procesos de desarrollo diferentes o ciclos de vida: ID e IA. La ID define y da cuenta de la similitud y la variabilidad de una LPS, estableciendo así una plataforma de software común para el desarrollo rápido de aplicaciones de alta calidad dentro de la línea. La IA deriva aplicaciones o

sistemas específicos al reutilizar estratégicamente la *plataforma LPS* que incluye básicamente la AR con el modelo de variabilidad o esquema instanciable y el repositorio de activos.

2. Es necesario definir y gestionar la variabilidad explícitamente en la Arquitectura de Referencia. Durante la ID, la variabilidad se introduce en todos los artefactos de dominio como requisitos, modelos arquitectónicos, componentes y casos de prueba.

Una razón para la introducción de la ILPS es la reducción de costos a través de la reutilización de los activos comunes de los diferentes productos. Dado que los múltiples productos que comparten características similares deben ser considerados en la ILPS, la complejidad de este proceso es alta comparada con la ingeniería de un producto de software único e implica por lo tanto un gran esfuerzo de trabajo conceptual de preparación de la plataforma LPS. El desarrollo de activos para ser reutilizados en múltiples productos a menudo no es fácil en las etapas iniciales del desarrollo de la LPS. Por lo tanto, el soporte de herramientas y métodos para la especificación, configuración y realización de estos activos es esencial en la ILPS. La norma [ISO15] de la Figura 3 se refiere a las capacidades de las herramientas y métodos necesarios para la ILPS. Para proporcionar los puntos focales para el uso y mejora de herramientas y métodos, esta norma ha identificado los insumos, los resultados y las tareas que deben realizarse.

A continuación se detalla el ciclo de vida la ID de la ILPS:

### ***A.1 Ingeniería del Dominio (ID)***

La ID es el proceso de analizar el dominio de una LPS para lograr el desarrollo de artefactos reutilizables o activos. No da lugar a un producto de software específico, sino prepara artefactos para ser utilizados en múltiples productos de una LPS. Un objetivo de la ID, como ya hemos mencionado, es el desarrollo para la reutilización. Se enfoca en proveer una manera apropiada para reutilizar dichos artefactos en la construcción de nuevos sistemas. Construye una AR y un repositorio de activos,



como una plataforma común de la LPS incluyendo la variabilidad clave identificada, que permanece durante todo el ciclo de vida de la LPS, para permitir a más largo plazo, el desarrollo de sistemas complejos basados en la reutilización. Herramientas y métodos para la ID facilitan esta tarea. En cambio la IA tiene el objetivo de desarrollar un producto específico para las necesidades de un cliente en particular (u otra de las partes interesadas). Se corresponde con el proceso de desarrollo de una aplicación única en la IS tradicional, pero vuelve a utilizar los artefactos de la ID donde sea posible. Su objetivo es el desarrollo con reutilización; la IA se repite para cada producto de la LPS que se va a derivar y realimenta el repositorio de activos. Todo el desarrollo ILPS se rige por el modelo de la fuente, originalmente planteado por [HE90].

Bjørner [Bjø06], el cual plantea un enfoque “clásico” de ID no orientado a LPS, también realiza una distinción entre el *espacio del problema* y el *espacio de la solución*, poniendo de relieve dos puntos de vista diferentes. El espacio del problema adopta la perspectiva de los actores y sus problemas, requisitos y puntos de vistas sobre todo el dominio y los productos individuales. Las facetas o *features* en términos de Pohl et al. [PBL05] son, de hecho, abstracciones del dominio que caracterizan el espacio del problema. Por el contrario, el espacio de soluciones representa las perspectivas de los desarrolladores y proveedores. Se caracteriza por la terminología del desarrollador, que incluye los nombres de funciones, clases, y los parámetros del programa. El espacio de la solución abarca el diseño, la implementación, la validación y verificación de las características y sus combinaciones en las formas más adecuadas (consistentes) para facilitar la reutilización sistemática en los productos específicos de la LPS.

Particularmente, en esta Tesis se abordan las tres primeras fases del ciclo de vida de la ID: *Alcance de la Línea de Productos* o “*Product Line Scoping (PL Scoping)*”, *Ingeniería de Requisitos del Dominio (IRD)* o “*Domain Requirements Engineering (DRE)*” y *Diseño del Dominio* o “*Domain Design (DD)*”. La Figura 3

presentada en la Sección 2.1.2 mostró estas fases resaltadas en gris, que se detallan a continuación y son tomadas del ISO/IEC 26550 [ISO15]:

#### ***A.1.1 Alcance de la línea de productos o “Product Line Scoping”***

Define el *Portafolio de Productos* el cual contiene los productos que constituirán la LPS y las principales características comunes y variables (visibles externamente) entre los productos, analiza los productos desde el punto de vista económico, y controla el desarrollo, producción y comercialización de la LPS con sus productos. El resultado principal de esta fase con respecto al framework ISO/IEC 26550 es la propuesta de activos. Esta incluye los activos principales (áreas funcionales y características comunes y variables de alto nivel de todas las aplicaciones) que se incluirán en una LPS con sus costos cuantificados y resultados de estimación de beneficios.

- El *Alcance del Portafolio de Productos* determina la definición del portafolio de productos, es decir 1) los productos de la LPS que la organización deberá desarrollar, producir, comercializar y vender; 2) las características comunes y variables que los productos deben proporcionar a fin de alcanzar los objetivos de negocio a corto y a largo plazo de la LPS de la organización, y 3) un calendario para la introducción de los productos al mercado.
- El *Alcance del Dominio* identifica y limita las áreas funcionales o funcionalidades, que son importantes para la LPS prevista, y proporcionar suficiente potencial de reutilización para justificar la creación de la LPS. El Alcance del Dominio se basa en las definiciones de las categorías de productos producidos por el Alcance del Portafolio de Productos.
- El *Alcance de los activos* se utiliza para identificar los activos reutilizables y calcular el costo/beneficio estimado de cada activo con el fin de determinar si una organización debe poner en marcha una LPS.

#### ***A.1.2 La Ingeniería de Requisitos del Dominio o “Domain Requirements Engineering”***

Comienza con la *hoja de ruta del producto* o “roadmap”, que es una planificación del desarrollo del producto con los objetivos a corto y largo plazo, y la lista de artefactos; captura integralmente los requisitos comunes y variables para los productos de la LPS, y construye una especificación inicial de requisitos incluyendo

un primer esbozo del modelo de variabilidad. También proporciona información para la gestión del producto con respecto a futuros cambios requeridos en el conjunto de características y la hoja de ruta del producto en su conjunto. La IRD tiene que adherirse a la especificación de las características de alto nivel de la LPS proporcionados por el alcance de la LPS. Sobre la base de estas características, se crean detallados requisitos comunes y variables suficientes para guiar el diseño del dominio (y por lo tanto también la realización, así como también las pruebas).

- *Licitación de requisitos del dominio* captura las variaciones que se prevén durante la vida útil previsible de la LPS de forma explícita. La licitación de requisitos del dominio se centra en el alcance, capturando de forma explícita la variación esperada por la aplicación de técnicas de obtención del dominio, y la incorporación de las partes interesadas del dominio.
- *Análisis de requisitos del dominio* busca las similitudes e identifica variaciones. Este también puede involucrar un mecanismo de retroalimentación más vigoroso a los solicitantes, señalando donde un sistema particular puede alcanzar la viabilidad económica si este fuera capaz de utilizar más requisitos comunes y pocos requisitos únicos. El análisis de requisitos del dominio tiene el alcance de la LPS como una de sus entradas; realiza un análisis de las similitudes y la variabilidad sobre la LPS licitada necesaria con el fin de identificar las oportunidades para la reutilización a mayor escala dentro de la LPS.
- *Especificación de requisitos del dominio* documenta un conjunto de requisitos de toda la LPS; incluirá marcadores simbólicos que los diversos documentos de requisitos específicos del producto cumplimentarán, expandirán e instanciarán.
- *Validación de requisitos del dominio* ofrece un amplio conjunto de revisores, y se produce en etapas. En primer lugar, los requisitos de toda la LPS deben ser verificados. Luego, en la tarea de validación de requisitos de la aplicación, los requisitos específicos del producto deben ser verificados. Sin embargo, los requisitos de toda la LPS deben también ser verificados para asegurarse de que tengan sentido para el producto.
- *Gestión de requisitos del dominio* hace provisiones por la naturaleza dual de la fase de IRD y por la naturaleza (común y específica) del servicio. Las políticas de gestión del cambio debe proporcionar un mecanismo formal para proponer cambios en la LPS y para soportar una evaluación sistemática del impacto de los cambios propuestos en la línea de productos. Las políticas de gestión del cambio gobiernan cómo los cambios en los requisitos de la LPS son propuestos, analizados y revisados. El acoplamiento entre los requisitos de

la LPS y los activos centrales (artefactos reutilizables) es aprovechado por el uso de enlaces de trazabilidad entre esos requisitos y sus asociados activos centrales del repositorio. Los cambios en los requisitos, entonces pueden desencadenar los cambios apropiados en los activos centrales del repositorio relacionados.

### ***A.1.3 El Diseño del Dominio o “Domain Design”***

Se basa en las especificaciones para desarrollar una arquitectura de línea de productos que permita la realización de la similitud y la variabilidad prevista en la LPS. Su objetivo principal es producir la AR de la LPS, definiendo su estructura (topología) y organización general, incluyendo su modelo de variabilidad. La AR refleja la variabilidad interna adicional introducida por la solución técnica, además de la variabilidad externa es decir, la similitud y variabilidad en los requisitos funcionales y no funcionales. Las principales preocupaciones del diseño del dominio se pueden dividir en tres partes, la construcción de la AR, la evaluación de la AR, y la gestión del Diseño del Dominio.

- *Construcción de la Arquitectura de Referencia.* El soporte de la variabilidad es crucial para el diseño de la AR. Debido a que todos los requisitos comunes y variables no pueden ser reflejados, una de las reglas básicas para añadir la variabilidad en el diseño de la arquitectura del dominio es dar prioridad a los requisitos. Los requisitos que son comunes a todas las aplicaciones deben ser satisfechos por la arquitectura de referencia, mientras que los requisitos variables no pueden ser a causa de la conflictividad entre algunos de ellos. La estructura y textura de la arquitectura de referencia soporta tales variabilidades.
- *Evaluación de la Arquitectura de Referencia* es una técnica de aseguramiento de la calidad. Para la ingeniería de LPS, una evaluación de la arquitectura de referencia es crucial, al menos para los requisitos de calidad relacionados con la LPS. Sólo una arquitectura que soporta suficientemente los requisitos de calidad sobrevivirá el tiempo suficiente como AR evolutiva.
- *Gestión del Diseño del Dominio* se enfoca en la gestión del desarrollo de la AR y el mantenimiento de los artefactos bajo cambios. Las principales actividades relacionadas con la gestión del diseño del dominio se pueden dividir en tres actividades: gestión de configuración y cambio para la gestión del diseño y trazabilidad del dominio. Las relaciones entre los requisitos comunes y/o variables y la AR no son simples asignaciones uno a uno, por lo

tanto, las trazabilidades entre los requisitos comunes/variables y los activos arquitecturales deben ser mantenidos a un nivel comprensible.

### ***A.2 Ingeniería de aplicación o “Application Engineering”***

La IA desarrolla sistemas individuales de acuerdo al esquema proporcionado por la AR. La ingeniería de aplicaciones y su modelo de ciclo de vida subyacente necesitan lidiar con menos complejidad y tiempos de desarrollo más cortos que la ID ya que una gran parte del esfuerzo de desarrollo y la complejidad se ha trasladado a la ID. Por otro lado, la IA está directamente involucrada con los clientes y por lo tanto, a menudo tendrá que lidiar con los cambios de las necesidades del cliente y con las restricciones que éstos puedan causar, implicando inclusive posibles cambios en la AR.

### ***B. Gestión organizacional (Organizational Management)***

Las fases de la gestión organizacional son aquellas fases necesarias para la orquestación del esfuerzo de la entera LPS. La introducción e institucionalización del enfoque LPS en las organizaciones no es una fase de un solo paso, sino que requiere preparación y planificación, ejecución e implementación.

### ***C. Gestión técnica (Technical Management)***

La gestión técnica es necesaria para diseñar la creación y evolución de los activos y productos básicos.

### ***D. Gestión de Activos Básicos o “Core Asset Management”***

La Gestión de Activos Básicos es una fase para almacenar, extraer y administrar los activos comunes resultantes de la ingeniería de dominio de una línea de productos de software. Si bien sólo las relaciones entre activos se gestionan en el desarrollo de productos individuales, la gestión de los activos básicos en un contexto de LPS es más compleja porque implica tanto los activos básicos como las relaciones entre ellos. La gestión de activos básicos gestiona y controla la plataforma LPS (AR y

Repositorio de Activos) y las configuraciones básicas de activos, incluidos los atributos para la minería y las anotaciones necesarias para reutilizar los activos principales (por ejemplo, enlaces, procesos y sus descripciones adjuntas a los activos centrales que prescriben cómo utilizar los activos), solicitud y retroalimentación del cambio, y el versionado de los activos básicos después de haber sido parte de la línea base o “baseline”, que en nuestro contexto comprende la AC inicial.

## 2.2 Líneas de Productos de Software Orientadas a Servicios (LPSOS)

La *Arquitectura Orientada a Servicios* o “*Service Oriented Architecture (SOA)*” [W3C04] es un marco de referencia empresarial para sistemas distribuidos, que básicamente sigue el estilo de eventos [SG95] y el modelo cliente-servidor para la distribución y comunicación; es centrada en la comunicación en redes a través de servidores Web que actúan como intermediarios para satisfacer las demandas de servicios por clientes distribuidos en ubicaciones geográficamente distantes. SOA ha cobrado enorme auge en la práctica reciente de desarrollo de software rápido, a bajo costo y es ampliamente tratada actualmente a nivel de investigación en la IS; constituye un paradigma para el diseño, desarrollo e implementación de aplicaciones de computación distribuida e independientes de la tecnología, por ejemplo, los Servicios Web pueden ser implementado mediante distintos lenguajes de programación como Java<sup>8</sup>, Python<sup>9</sup>, PHP<sup>10</sup>, etc., basada en estándares como WSDL<sup>11</sup>, XML<sup>12</sup>, UDDI<sup>13</sup>, etc., para el descubrimiento, el enlazado y ensamblaje de servicios de software débilmente acoplados [GAT13]. Se integra muy bien con el estilo clásico de capas [SG95] de los sistemas de información, permitiendo realizar integración de sistemas empresariales, ver Figura 1. El paso automático o semiautomático de procesos de negocios a composiciones de servicios es un área activa de investigación y una práctica de desarrollo.

---

<sup>8</sup> <https://www.java.com/es/>

<sup>9</sup> <https://www.python.org/>

<sup>10</sup> <http://php.net/>

<sup>11</sup> Web Services Definition Language

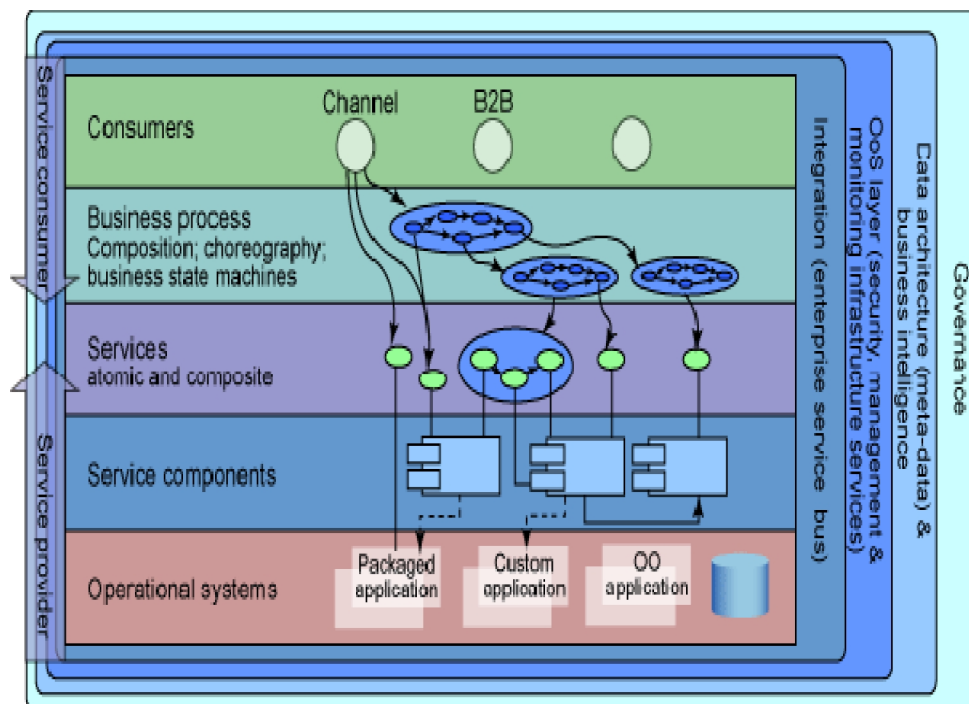
<sup>12</sup> eXtensible Markup Language

<sup>13</sup> Universal Description, Discovery and Integration

### 2.2.1 *Arquitectura de Software Orientadas a Servicios*

Una *arquitectura de software* es definida clásicamente en términos una configuración o topología de elementos arquitecturales (componentes y conectores) con un comportamiento específico, para cumplir determinados objetivos de alto nivel de abstracción [SG04] y es expresadas por múltiples vistas, incluyendo una vista lógica [Kru95]; una *Arquitectura basada en Servicios Web* o “*Web Services Architecture (WSA)*” también se expresa en [W3C04] mediante múltiples vistas o modelos, tales como mensajes, servicios, recursos y políticas. Se hablará indistintamente de servicios, *Servicios Web (SW)* o “*Web Services (WS)*” aunque su implementación puede ser diferente, de acuerdo al modelo de comunicación débilmente acoplado que se utilice, como síncrono, asíncrono, basado en mensajes, etc. Sin embargo una vista lógica de WSA considerando los servicios como componentes arquitecturales y sus conexiones (por mensajes, protocolos, etc.) no ha sido utilizada aún con frecuencia [Mil01]. El modelo de referencia SOA de la Figura 4 muestra las capas de Servicios y Componentes. Los servicios de la Capa de Servicios se hacen corresponder con componentes arquitecturales en la Capa de Componentes, después de pasar por una Capa de Procesos de Negocios con la composición de servicios, pero como se realiza y organiza en general esta correspondencia no es tratada en la literatura, sino en ejemplos muy particulares.

Una LPSOS es considerada como un conjunto de sistemas similares orientados a servicios que soportan los procesos de negocio de un dominio específico y que pueden ser desarrollados a partir de una plataforma común o conjunto de activos principales [MSR10]. Permiten la reutilización sistemática de componentes de software/servicios y ofrecen la flexibilidad en tiempo de ejecución requerida para el desarrollo de aplicaciones de alta calidad y eficacia en cuanto al costo y tiempo de entrega [EMA13].



**Figura 4.** Modelo de Referencia SOA [IBM09]

### 2.2.2 Integración de los Enfoques de Líneas de Productos de Software y Arquitectura Orientada a Servicios

Un aspecto de gran importancia que debe ser considerado al analizar las LPSOS, es el de la variabilidad. La *variabilidad* describe la capacidad de un sistema de software o artefacto para ser cambiado (ampliado, personalizado o configurado) para su uso en un contexto específico. Este especifica las soluciones arquitectónicas o computacionales que no son completamente definidas durante el diseño inicial. La variabilidad se introduce a través de los *puntos de variación* o “*placeholders*” (es decir, ubicaciones predefinidas en la arquitectura en la que se puede producir el cambio mediante una instanciación). En estos puntos de variación, que contienen conjuntos de componentes no comunes o variantes, las variantes son elegidas al crear instancias para un sistema de software o producto concreto. Hay diferentes tipos de variabilidad, tales como la variabilidad en las características funcionales o no funcionales o en los procesos de negocio. La variabilidad en las características o aspectos tanto funcionales como no funcionales que deben estar presente en la LPS,



es resuelta mediante la ILPS; la variabilidad en los procesos de negocio es resuelta a través de SOA.

Para que la integración de las LPS y SOA sea un éxito, es necesario que se resuelvan y/o se superen los siguientes desafíos [LK10]:

- Identificación apropiada de servicios,
- Adaptación del análisis de características para complementar las técnicas de identificación de servicio,
- Adaptación de la LPS a la orientación a servicios con respecto al descubrimiento y negociación de los servicios requeridos para mejorar la flexibilidad en tiempo de ejecución de la LPS,
- Incorporación de puntos de variación en los componentes que dan soporte a los servicios para controlar explícitamente las posibles configuraciones de la línea de productos.

### ***2.2.3. LPSOS dinámicas***

Por su parte, la *Orientación a Servicios (OS)* basada en el modelo SOA ya mencionado, es un paradigma que emerge rápidamente para el diseño y desarrollo de sistemas de software adaptables y dinámicos. La OS se centra en la creación y desarrollo de soluciones de aplicaciones mediante la utilización de servicios como bloques de construcción de software que encapsulan la funcionalidad y proporcionan flexibilidad a través del enlace dinámico. La promesa visionaria de la OS es que las aplicaciones se componen a la perfección por servicios, cuya comunicación se rige por modelos débilmente acoplados, con el fin de crear sistemas adaptables y dinámicos [MAG+13]. Por ello, recientemente existen estudios [INP+09] [PBD09] [LK10] [Par11] [BGP12] [SEK13] [SF13] que abordan el problema proponiendo lo que denominan: *Líneas de Productos de Software Dinámicas (LPSD)*. Las LPSD extienden las LPS para soportar nuevas configuraciones en tiempo de ejecución, adoptando SOA como arquitectura alternativa para hacer viable este requerimiento. Este tipo de enfoque está relacionado con el desarrollo de sistemas que pueden reaccionar a los cambios de su medio ambiente, también denominados sistemas

sensibles al contexto “*context-aware systems*” [PBD09]. Particularmente, el presente trabajo de investigación no sigue este enfoque de desarrollo, donde la configuración de nuevos sistemas es realizado en tiempo de ejecución.

En conclusión, SOA ha demostrado su rentabilidad en el desarrollo de sistemas de software flexibles y dinámicas, por lo cual existen importantes ventajas mutuas en la convergencia de SOA y LPS. Los servicios simplifican la integración de los sistemas complejos y permiten que se adecuen de manera más flexible al cambio. El acoplamiento débil y el uso de técnicas dinámicas o de enlace tardío hacen que los servicios más apropiados estén disponibles en tiempo de ejecución durante una situación determinada [BGP12]. Sin embargo, como veremos en el Capítulo III, no hay muchos métodos de desarrollo bien definidos y completamente adoptados en la industria, y menos aún, estándares, para la ILPSOS.

### **2.3. Calidad del Software y calidad del producto de software según el estándar ISO/IEC 25010**

En los aspectos antes tratados, el aseguramiento de la calidad del producto de una LPS es una actividad crucial para su éxito y su evolución en el tiempo. Actualmente no se concibe un sistema de software que no tenga la calidad necesaria que satisfaga las expectativas de sus requisitos funcionales y de los usuarios finales. Más aun, las propiedades de calidad para una LPSOS son fundamentales para garantizar su completitud, idoneidad y carácter evolutivo [ISO15] [HLO+16]. Por ello, en general y sobre todo en LPS, es vital abordar las características de calidad desde una etapa temprana en el ciclo de vida de desarrollo de software.

#### **2.3.1 Calidad del Software**

La calidad ha sido típicamente definida como un nivel de excelencia, de conformidad con las especificaciones, de satisfacción de requisitos, de atributos distintivos, de estar libre de defectos, deficiencias y de las significativas variaciones para cumplir con las expectativas del cliente [AAZ11].

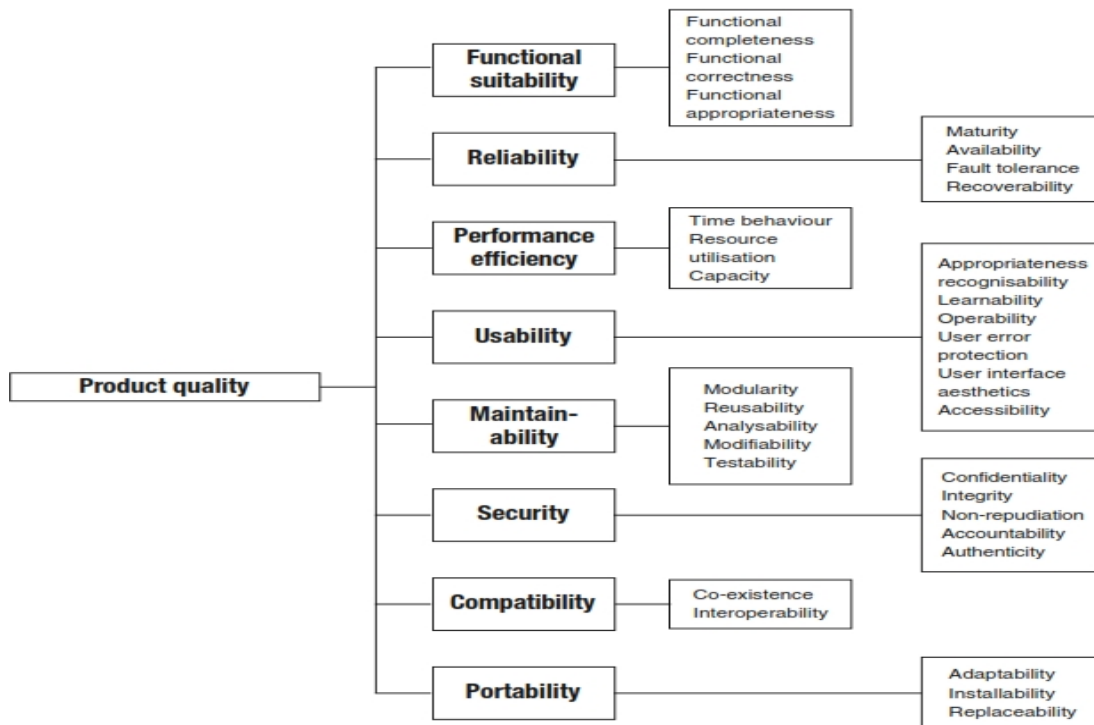
Establecer un modelo de calidad para el desarrollo del dominio de la aplicación que involucra los procesos de identificación de los stakeholders, adquisición del dominio, análisis del dominio y modelado del dominio, planteados en Bjørner y Pohl et al. [Bjø06] [PBL05], garantizaría la validez y confiabilidad del dominio desarrollado. El aseguramiento de la calidad del producto es una actividad crucial para el éxito de la industria del software, pero es, si cabe, más importante cuando se trata del desarrollo de la LPS, dado que la reutilización masiva de activos de software hace que los atributos de calidad (propiedades medibles de un artefacto de software) de los activos de software impacten en la calidad de todos los productos de la LPS [Gon12]. La calidad se ha convertido en un aspecto crítico de los productos de software ya que su ausencia produce pérdidas financieras cuantiosas, de salud, y a veces de vida. Al mismo tiempo la definición, o alcance, del dominio de la calidad del software ha evolucionado continuamente desde una perspectiva técnica a una perspectiva que abarca aspectos humanos, tales como la facilidad de uso y la satisfacción [Sur14].

Un reconocimiento cada vez mayor en relación con la importancia de la calidad del software también ha hecho que la IS cambie "su centro de gravedad" de la creación de una solución artesanal a una solución de ingeniería hacia la satisfacción de las partes interesadas [Sur14]. Así que la especificación de la calidad en la ingeniería aplicada al software, sistemas y servicios relacionados, tiene la intención de ayudar a los desarrolladores en la construcción de productos que responden adecuadamente a las exigencias de los clientes y para protección contra los productos defectuosos y los proveedores no profesionales [Sur14].

### ***2.3.2 Estándares de calidad del software***

El estándar internacional que directamente es el más aplicable al control de calidad del producto de software es la serie de estándares SQuaRE de la Organización Internacional de Estándares (ISO) [ISO05]. SQuaRE representa la Calidad, Requisitos y Evaluación para los productos de Software. Es un proyecto con mucho tiempo en ejecución de la ISO para consolidar y sustituir a la antigua norma ISO/IEC

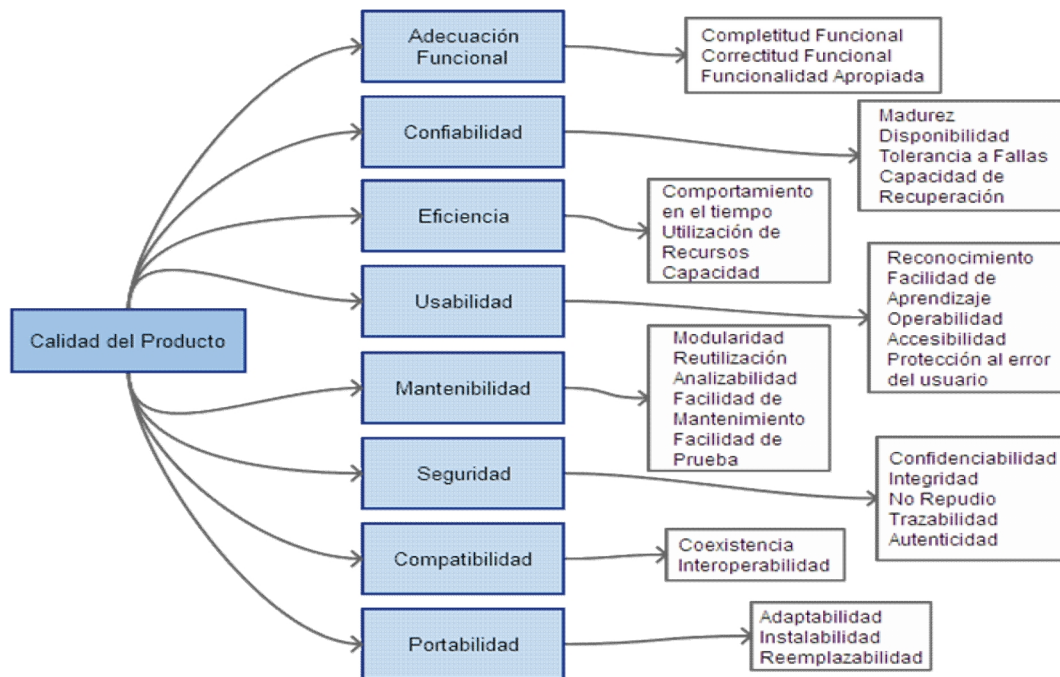
9126-1, que solía ser la norma principal para definir la calidad del producto de software, con la actual norma ISO/IEC 25010 [ISO11]. La serie SQuaRE [ISO05] incrementa la compatibilidad con otras normas ISO que se enfocan en la medición y la calidad del proceso [Wag13]. El elemento central de toda la serie es el *modelo de calidad* en el estándar ISO/IEC 25010. Este estándar describe un framework de modelo de calidad, que es un meta-modelo de los modelos de calidad que deben ser utilizados conforme al estándar. Este define que la calidad debe ser descompuesta en varios niveles de características de calidad o factores de calidad. Además, el estándar define dos modelos de calidad del software: calidad del producto y calidad en uso. Ambos modelos son versiones ligeramente modificadas de los modelos de calidad de la antigua norma ISO/IEC 9126-1. Particularmente, para esta investigación, solo se utilizará el modelo de calidad del producto (Figura 5).



**Figura 5.** Modelo de calidad del producto del ISO/IEC 25010 [ISO11] (versión oficial)

En la Figura 6, se muestra nuevamente el modelo de calidad ISO/IEC 25010, en una traducción no oficial, por ello se dejó la versión original en la Figura 5.

El estándar no establece cuándo se debe utilizar alguno de los modelos, aparte de indicar que: "el modelo de calidad del producto se centra en el sistema de cómputo destino que incluye el producto de software destino, y el modelo de calidad en uso se centra en todo el sistema hombre-máquina que incluye el sistema de cómputo y el producto de software destino", es decir: el modelo de calidad del producto se refiere a todo lo que involucra el desarrollo del software, pero cuando éste aún no está entregado a sus usuarios finales; el modelo de calidad en uso en cambio, se refiere al software el cual ya está en su ambiente de ejecución y es manejado por sus usuarios finales. Es importante entender, sin embargo, que estos modelos son taxonomías. Ellos describen una estructuración jerárquica de la calidad; el conjunto de características de alto nivel describen la calidad global de un sistema de software o producto. A partir de las sub-características de calidad se pueden derivar otras sub-características para así llegar a las requeridas en un cierto dominio; es de hacer notar que estos modelos de calidad requieren una adaptación cuidadosa para el dominio de interés, que debe ser realizada por un ingeniero de software y/o un ingeniero de calidad experto, conocedor del dominio.



**Figura 6.** Modelo de calidad del producto del ISO/IEC 25010 [ISO11] (traducción no oficial)

### ***2.3.3 Marco de referencia del Modelo de Calidad ISO/IEC 25010***

#### ***A. Modelos de calidad***

La calidad de un sistema es el resultado de la calidad de los elementos del sistema y su interacción. La calidad del software es el grado en que el producto de software satisface las necesidades establecidas e implícitas cuando es utilizado bajo condiciones específicas. El modelo de calidad del software define ocho características de calidad del software: idoneidad funcional, fiabilidad, eficiencia de rendimiento, operabilidad, seguridad, compatibilidad, facilidad de mantenimiento y transferibilidad [ISO11].

El modelo calidad en uso define tres características a nivel del sistema: facilidad de uso, flexibilidad en uso y seguridad en el uso que puede ser utilizado para especificar y evaluar los requisitos para el efecto de la calidad del software en contextos de uso especificados.

Las características de calidad tienen definidas subcaracterísticas y el estándar permite sub-subcaracterísticas definidas por el usuario en una estructura jerárquica. Las características de calidad definidas pueden ser utilizadas como una lista de verificación para asegurar una cobertura completa de la calidad [ISO11].

La calidad en uso es una medida de la calidad global del sistema en su entorno operativo para usuarios específicos, para llevar a cabo tareas específicas. Desde una perspectiva del software, la calidad en uso puede ser utilizada para medir la capacidad del software para permitir la calidad en uso en su entorno operativo, para llevar a cabo tareas específicas por usuarios específicos.

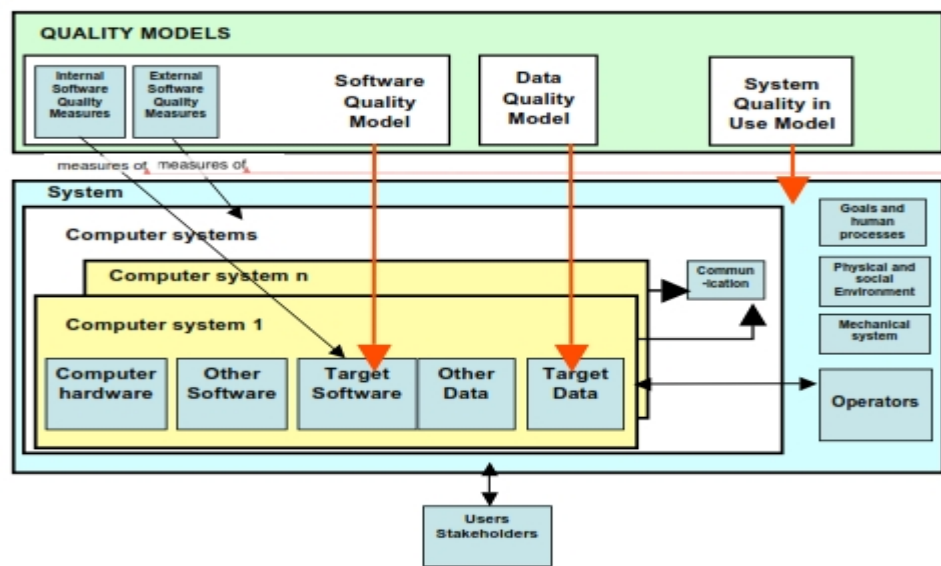
La calidad del software externa proporciona una vista de caja negra del software y aborda las propiedades relacionadas con la ejecución del software en el hardware y el sistema operativo.

La calidad del software interna proporciona una vista de caja blanca del software y aborda las propiedades del producto de software que normalmente están disponibles durante el desarrollo. La calidad del software interna está relacionada principalmente con las propiedades estáticas del software. La calidad del software

interna tiene un impacto sobre la calidad del software externa, que a su vez tiene un impacto sobre la calidad en uso. La Figura 7 muestra los diferentes tipos de medidas de calidad. La calidad de los datos está descrita en la norma ISO/IEC 25012.

En nuestro caso solo consideraremos la calidad interna del producto, porque estamos situados en una fase de diseño y el sistema final aún no se ha construido.

Los modelos de calidad sirven como un marco para garantizar que todos los aspectos de la calidad sean considerados desde el punto de vista interno, externo, y de la calidad en uso [ISO11].



**Figura 7.** Alcance de las medidas de calidad [ISO11]

### ***B. Propiedades del software***

Algunas propiedades del software son *inherentes* al producto de software, generalmente dependientes del sistema; algunas están *asignadas* al producto de software. La calidad de un producto de software en un contexto de uso particular está determinada por sus propiedades inherentes [ISO11].

- NOTA 1: "inherente" significa existente en algo, especialmente como una característica o rasgo permanente.
- NOTA 2: Ejemplos de propiedades inherentes son el número de líneas de código y la precisión de un cálculo numérico proporcionado por el software.

Ejemplos de propiedades asignadas son las del propietario de un producto de software y el precio de un producto de software.

Las propiedades inherentes pueden ser clasificadas como propiedades funcionales o como propiedades de calidad. Las propiedades funcionales determinan lo que el software es capaz de hacer (requisitos funcionales). Las propiedades de calidad determinan que tan bien el software las realiza o ejecuta (requisitos no funcionales). En otras palabras, las propiedades de calidad muestran el grado en que el software es capaz de proporcionar y mantener sus servicios especificados. Las propiedades de calidad son inherentes a un producto de software y al sistema asociado. Una propiedad asignada por lo tanto no es considerada como una característica de calidad del software, ya que se puede cambiar sin cambiar el software. La Figura 8 ilustra esta clasificación de las propiedades de software [ISO11].

Software properties	Inherent properties	Domain-specific functional properties
		Quality properties (functional suitability, reliability, performance efficiency, operability, security, compatibility, maintainability, transferability)
	Assigned properties	Managerial properties like for example price, delivery date, product future, product supplier

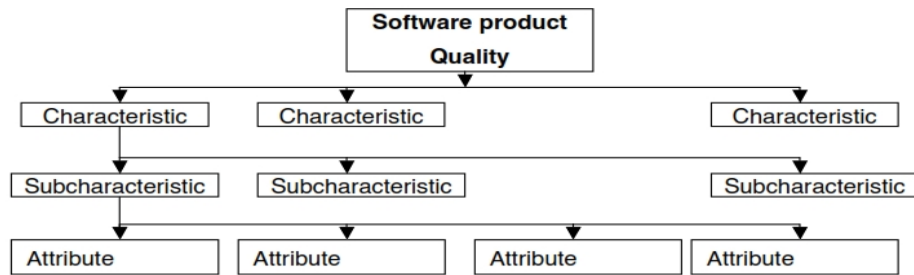
**Figura 8.** Propiedades del software [ISO11]

### ***C. Estructura utilizada para los modelos de calidad***

Los modelos de calidad SQuaRE categoriza la calidad del producto en características que se subdividen en subcaracterísticas y atributos de calidad (Figura 9) [ISO11].

El modelo de calidad SQuaRE consta de dos partes, el modelo de Calidad del Software Interno y Externo y el modelo de Calidad en Uso.





**Figura 9.** Estructura utilizada para el modelo de calidad [ISO11]

#### ***D. Utilizando un modelo de calidad***

La calidad del producto de software deberá ser evaluada utilizando un modelo de calidad definido. El modelo de calidad deberá ser utilizado al establecer los requisitos de calidad para los productos de software y productos intermedios. La calidad del producto de software deberá estar descompuesta en forma jerárquica en un modelo de calidad compuesto por características y subcaracterísticas que se pueden utilizar como una lista de chequeo de asuntos relacionados con la calidad.

No es posible en la práctica medir todas las subcaracterísticas internas y externas para todas las partes de un producto de software de gran tamaño. Del mismo modo que no suele ser práctico medir la calidad en uso de todos los posibles escenarios de las tareas del usuario. La importancia relativa o prioridad de las características de calidad dependerá del dominio de la aplicación y del producto. Por lo que el modelo deberá adaptarse antes de su uso, y los recursos para la evaluación asignados entre los diferentes tipos de mediciones dependerán de los objetivos del negocio y de la naturaleza de los procesos de diseño y del producto [ISO11].

#### ***E. Características y sub-características del Modelo de Calidad del Producto ISO/IEC 25010***

ISO/IEC 25010 considera una estructura jerárquica y propone ocho características de calidad de alto nivel: adecuación funcional, eficiencia, compatibilidad, usabilidad, confiabilidad, seguridad, mantenibilidad y portabilidad. Estas son refinadas en sub-características y definidas en su último nivel por un conjunto de atributos (o elementos medibles) de un producto de software, para el cual

la calidad es descrita y evaluada. La Tabla 1 muestra una traducción al castellano no oficial de las definiciones de las características y sub-características del estándar [ISO11].

**Tabla 1.** Definición de características y sub-características del Modelo de Calidad del Producto ISO/IEC 25010 [ISO11]

<b>Característica</b>	<b>Definición</b>
<b>Adecuación funcional</b>	Grado con el cual un sistema cumple funciones que corresponden a necesidades establecidas o implícitas, cuando es usado bajo ciertas condiciones.
<b>Sub-características</b>	
<ul style="list-style-type: none"> <li>• <i>Compleitud</i>: grado en que el producto ofrece las funciones que cumplen las necesidades implícitas y explícitas cuando el producto sea utilizado bajo las condiciones especificadas.</li> <li>• <i>Correctitud o precisión</i>: grado con el cual el sistema proporciona resultados correctos con el debido grado de precisión.</li> <li>• <i>Ser apropiado</i>: grado con el cual el conjunto de funciones facilita el cumplimiento de objetivos y tareas específicas.</li> </ul>	
<b>Eficiencia</b>	Eficiencia relativa al conjunto de recursos usados bajo condiciones dadas.
<b>Sub-características</b>	
<ul style="list-style-type: none"> <li>• <i>Comportamiento en tiempo</i>: el tiempo de procesamiento y de respuesta, y las tasas de <i>throughput</i> del sistema cuando realiza sus funciones, bajo condiciones establecidas.</li> <li>• <i>Utilización de recursos</i>: cantidad y tipo de recursos utilizados cuando el producto realiza su función en las condiciones indicadas en relación con el <i>benchmark</i> establecido.</li> <li>• <i>Capacidad</i>: grado con el cual los máximos límites de los parámetros del sistema cumplen con los requisitos.</li> </ul>	
<b>Compatibilidad</b>	Grado con el cual un sistema puede intercambiar información con otros sistemas y desempeñar las funciones requeridas compartiendo el mismo ambiente de hardware o software.
<b>Sub-características</b>	
<ul style="list-style-type: none"> <li>• <i>Co-exigencia</i>: Capacidad en que el producto puede coexistir con otros productos independientes en un ambiente común compartiendo de recursos comunes sin ningún impacto perjudicial.</li> <li>• <i>Interoperabilidad</i>: Capacidad en que dos o más sistemas o componentes pueden intercambiar información y usar la información que ha sido intercambiada.</li> </ul>	
<b>Usabilidad</b>	Grado con el cual un sistema puede ser usado por usuarios específicos, para cumplir con sus objetivos con efectividad, eficiencia y satisfacción, en un determinado contexto de uso.
<b>Sub-características</b>	
<ul style="list-style-type: none"> <li>• <i>Capacidad de ser aprendido</i>: grado con el cual un sistema puede ser usado por usuarios específicos con el objeto de aprender el sistema con efectividad, eficiencia, satisfacción y sin riesgos, en un determinado contexto de uso.</li> <li>• <i>Ser reconocido como apropiado</i>: grado con el cual los usuarios reconocen si un sistema</li> </ul>	

	<p>es apropiado para sus necesidades.</p> <ul style="list-style-type: none"> <li>• <i>Capacidad de ser operado</i>: grado con el cual un sistema posee atributos que lo hacen fácil de operar y controlar.</li> <li>• <i>Protección de errores del usuario</i>: grado con el cual el sistema protege al usuario de cometer errores.</li> <li>• <i>Estética de la interfaz usuario o apariencia</i>: grado con el cual la interfaz usuario permite hacer agradable y satisfacer la interacción con el usuario.</li> <li>• <i>Capacidad de ser de fácil acceso</i>: grado con el cual un sistema puede ser accedido por una gran variedad de personas, en un determinado contexto de uso.</li> </ul>
<b>Confiabilidad</b>	Grado con el cual un sistema realiza funciones específicas, en condiciones específicas, para un periodo de tiempo específico.
	<p><b>Sub-características</b></p> <ul style="list-style-type: none"> <li>• <i>Madurez</i>: grado con el cual un sistema cumple con sus funciones bajo condiciones normales de operación; trata de la frecuencia de fallas.</li> <li>• <i>Disponibilidad</i>: grado con el cual un sistema es operacional y accesible cuando se requiere usarlo; está relacionada con la madurez, la tolerancia a fallas y la recuperabilidad.</li> <li>• <i>Tolerancia a fallas</i>: grado con el cual un sistema sigue operando como convenido en presencia de fallas de hardware o software.</li> <li>• <i>Capacidad de recuperarse o recuperabilidad</i>: grado con el cual un sistema, en presencia de un evento de interrupción o falla, puede recuperar los datos afectados y restablecer el estado deseado del sistema.</li> </ul>
<b>Seguridad</b>	El grado con el cual la información o datos son protegidos de forma tal que personas o sistemas no puedan leerlos o modificarlos y que en cambio, personas o sistemas autorizados tengan acceso permitido a esa información o datos.
	<p><b>Sub-características</b></p> <ul style="list-style-type: none"> <li>• <i>Confidencialidad</i>: Grado en que la información y la data sean protegidos contra la divulgación no autorizada, ya sea accidental o deliberada.</li> <li>• <i>Integridad</i>: Capacidad de un sistema o componente en impedir el acceso no autorizado o la modificación de datos o programas computacionales.</li> <li>• <i>No repudio</i>: Grado en que las acciones o eventos que han tenido lugar pueden ser probados, de modo que no pueden ser repudiadas tardíamente.</li> <li>• <i>Auditable</i>: Grado en que las acciones de una entidad pueden ser trazadas inequívocamente hacia esa entidad. Es la capacidad de ser auditado.</li> <li>• <i>Autenticidad</i>: Grado en que la identidad de un sujeto o recurso puede demostrarse al ser reclamado.</li> </ul>
<b>Mantenibilidad</b>	Grado de eficacia y eficiencia con que el producto puede ser modificado.
	<p><b>Sub-características</b></p> <ul style="list-style-type: none"> <li>• <i>Modularidad</i>: grado con el cual un sistema o programa está compuesto de componentes discretas, de forma tal que un cambio a un componente tiene un impacto mínimo sobre los demás componentes.</li> <li>• <i>Reutilización</i>: grado con el cual un cierto activo puede ser usado en más de un sistema o para construir otros activos.</li> <li>• <i>Capacidad de ser analizado</i>: facilidad con la cual el impacto de un cierto cambio puede ser comprobado sobre el resto del producto, o el producto puede ser diagnosticado por defectos o causas de fallas, o en que las partes a ser modificadas puedan ser identificadas.</li> </ul>

<ul style="list-style-type: none"> <li>• <i>Capacidad de ser modificado</i>: grado con el cual un sistema puede ser efectivamente y eficientemente modificado sin introducir defectos o disminuir su eficiencia.</li> <li>• <i>Capacidad de ser probado</i>: facilidad con la cual criterios de prueba puedan ser establecidos para un sistema o componente y pruebas pueden ser realizadas y determinar si esos criterios se han cumplido.</li> </ul>	
<b>Portabilidad</b>	Capacidad de un sistema o componente de ser eficaz y eficiente al ser transferido a un hardware, software u ambiente operativo diferente.
<b>Sub-características</b>	
<ul style="list-style-type: none"> <li>• <i>Adaptabilidad</i>: grado con el cual un sistema o componente puede ser efectivamente y eficientemente adaptado de un ambiente de hardware, software u operativo, a otro.</li> <li>• <i>Capacidad de ser instalado</i>: facilidad con la cual un sistema puede ser efectivamente instalado o desinstalado en un ambiente específico.</li> <li>• <i>Capacidad de ser reemplazado</i>: grado en que el producto puede ser utilizado en lugar de otro producto de software especificado para el mismo fin en el mismo entorno.</li> </ul>	

A continuación se presenta el dominio de los *Sistemas de Información Integrados de Salud* o “*Healthcare Integrated Information Systems*”, que será utilizado para aplicar los procesos QuaDRA y WSRA-SPL en el Capítulo VI de esta investigación.

## 2.4 Dominio de los Sistemas de Información Integrados de Salud (SIS)

### 2.4.1 *Sistemas integrados de salud*

Según la definición de la Organización Mundial de la Salud, “Un sistema de salud es la suma de todas las organizaciones, instituciones y recursos cuyo objetivo principal consiste en mejorar la salud. Un sistema de salud necesita personal, financiación, información, suministros, transportes y comunicaciones, así como una orientación y una dirección generales. Además tiene que proporcionar buenos tratamientos y servicios que respondan a las necesidades de la población y sean justos desde el punto de vista financiero” [AAE+12]. En cuanto a la calidad de los SIS. La *interoperabilidad, disponibilidad y seguridad* son propiedades de calidad prioritarias. La interoperabilidad para el manejo de la información médica entre diferentes instituciones de salud, la disponibilidad porque estos sistemas deben estar disponibles la mayor parte del tiempo por la atención a pacientes y la seguridad porque no todo el personal de salud puede tener acceso a la información médica de un paciente, la cual

es confidencial. Se agrega la propiedad de *persistencia* a las mencionadas anteriormente, porque la información médica de un paciente no puede ser destruida.

El avance en el desarrollo de las Tecnologías de *Información y la Comunicación (TIC)* y los avances en la comunicación de datos a través de Internet proveen la plataforma ideal para el acceso universal globalizado de la información, así como de los servicios. Este contexto conduce a la identificación de la *salud electrónica o “e-Health”*, también denominada ahora *Telesalud*, como un área que se caracteriza por utilizar y combinar las TIC para almacenar y compartir datos con objetivos clínicos, administrativos y educacionales entre centros de salud ubicados en localidades geográficamente distantes [AAE+12].

En nuestro contexto, un SIS es un sistema de gestión médica que soporta la interconexión en red, remota y local (internet e intranet) y la interoperabilidad manifiesta en el intercambio de información digital, que satisfaga las necesidades de la población en materia de salud.

La *Historia Clínica (HC) o Health Record (HR)* es una funcionalidad clave en un SIS que gestiona el repositorio con la información, ordenada cronológicamente, de los episodios (eventos) clínicos registrados para una persona. Por lo tanto, es un instrumento imprescindible para que el profesional de la salud pueda llevar a cabo su actividad y prestar al paciente la mejor atención posible en cada momento. Su utilidad trasciende los fines asistenciales, y puede añadir funciones de planificación, gestión administrativa de instituciones de salud, investigación y educación. En la actualidad, la mayoría de las HC se almacenan en papel, con las desventajas que esto genera tanto a nivel de consulta, almacenamiento, personal a cargo, deterioro del papel, ilegibilidad de la letra, así como también de seguridad y confidencialidad de los datos [AAE+12].

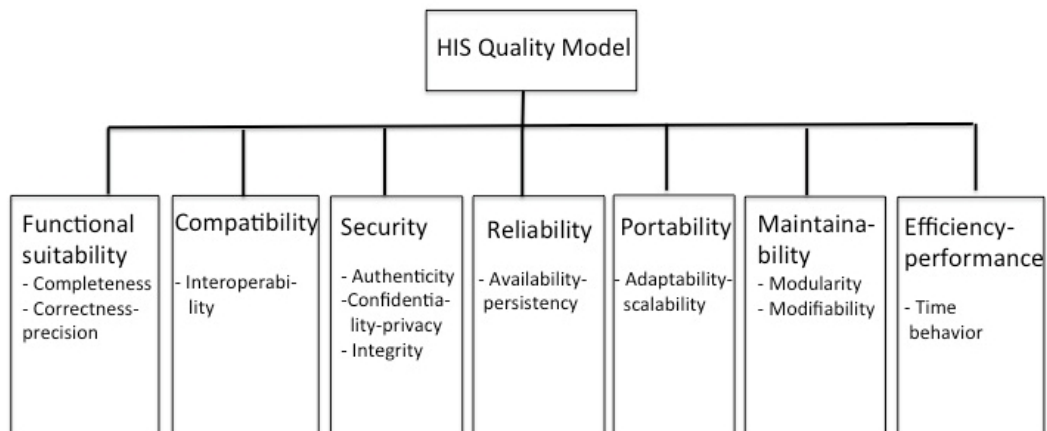
En consecuencia surge la *Historia Clínica Electrónica (HCE) o Electronic Health Record (EHR)*, que permite solucionar las carencias anteriormente descritas y propone los siguientes beneficios [AAE+12]:

- ✓ A nivel de accesibilidad y disponibilidad debido a que más de una persona a la vez puede acceder a la HCE y a su vez desde distintos lugares físicos,

- ✓ A nivel de visualización en donde dependiendo de las necesidades del usuario se ajustará el formato con el que se visualiza la información,
- ✓ A nivel de comunicación permitiendo la misma entre profesionales de forma similar a un correo electrónico o mensajería instantánea vinculado a la salud de un paciente,
- ✓ A nivel de agregación de datos permitiendo crear resúmenes según la información almacenada, hacer gestión clínica e investigación clínica y
- ✓ A nivel de la toma de decisiones colaborando con el proceso asistencial brindando soporte a los profesionales proponiendo alternativas posibles.

La HCE es el sistema medular en cualquier sistema informático de salud moderno. Allí se encuentran los registros de cada episodio clínico protagonizado por un paciente dado, así como sus enfermedades y hasta sus antecedentes familiares. La HCE permite a cualquier médico conocer a fondo a su paciente sin necesidad de hacerle un cuestionario que a veces, por la distancia entre los léxicos hablados por el profesional y el paciente, no conduce a una visión completa y correcta. El desafío de una HCE es almacenar la mayor cantidad de información a la vez de brindar distintas vistas (perspectivas) de esta información de forma eficiente. Por lo tanto, es el lugar primario para la carga de toda la información clínica [AAE+12]. En la actualidad, la HCE es la funcionalidad más importante de un SIS. Sin embargo, la mayor parte de las organizaciones que prestan servicios de salud, almacenan las HCE en todo tipo de formatos propietario, y son gestionadas en una multitud de sistemas de información médica disponibles en el mercado. La interoperabilidad de la información médica también se enfrenta con problemas psicosociales ocasionados por la praxis médica. Esta situación se convierte en un problema serio para la interoperabilidad de la información en el campo de la informática médica [LVV+08] [LMO+14].

El modelo de calidad completo para SIS o “*Healthcare Integrated Information Systems (HIS)*”, obtenido instanciando el modelo ISO/IEC 25010, se presenta en la Figura 10:



**Figura 10.** Modelo de calidad del producto para SIS [LOE15] [ISO11]

#### 2.4.2 SIS y su relación con SOA

Sobre este particular, SOA plantea un enfoque para la implementación de los SIS en [SMM+09], donde se describe el rol de SOA en los SIS en Telemedicina, indicando que la interoperabilidad en un sistema de telemedicina es una de las principales preocupaciones. Es difícil diseñar una arquitectura interoperable y flexible en telemedicina que transmita datos y permita el intercambio de información entre los sistemas. SOA está jugando un papel importante en el desarrollo de tales sistemas para facilitar el intercambio de la información entre diferentes aplicaciones de telemedicina mediante WS [KMM08] [NM07], quienes fungen como primordiales para la interoperabilidad y la flexibilidad a los cambios [MRS+07]. Se ha demostrado que las organizaciones de salud han obtenidos algunos beneficios con el uso de SOA; las organizaciones que han optado por utilizar las soluciones de SOA, lo han hecho para mejorar su capacidad evolutiva (responder a las necesidades cambiantes), para desarrollar con mayor eficacia, mejorar su rendimiento, y la consistencia en el desarrollo de SIS [HSS+08]. El aspecto de seguridad es uno de los problemas importantes a ser tratado en este dominio y en particular se vislumbra ahora la solución de la nube como viable también para los sistemas de salud; sin embargo esta solución presenta el grave problema de la disponibilidad, la cual no puede ser asegurada 100% con el uso de la nube.

A continuación se presenta brevemente el enfoque ontológico, el cual puede ser utilizado en nuestro contexto como una especificación alternativa de una RA.

### ***2.5 Ontologías en el desarrollo de LPS***

Una ontología define un vocabulario común para los investigadores que necesitan compartir información en un dominio. Incluye definiciones interpretables por la máquina de conceptos básicos en el dominio y las relaciones entre ellos [NM01].

Razones para el desarrollo de una ontología son [NM01]:

- Compartir la común comprensión de la estructura de la información entre las personas o agentes de software
- Permitir la reutilización del conocimiento del dominio
- Para hacer explícitos los supuestos del dominio
- Para separar el conocimiento del dominio del conocimiento operacional
- Analizar el conocimiento del dominio

1. Compartir la común comprensión de la estructura de la información entre las personas y los agentes de software es uno de los objetivos más comunes en el desarrollo de ontologías. Por ejemplo, supongamos que varios sitios Web contienen información médica o prestan servicios médicos electrónicos. Si estos sitios Web comparten y publican la misma ontología subyacente de los términos que todos ellos usan, entonces los agentes informáticos podrán extraer y agregar información de estos diferentes sitios. Los agentes pueden utilizar esta información agregada para responder a consultas del usuario o como entrada de datos a otras aplicaciones [NM01].

2. Permitir la reutilización del conocimiento del dominio fue uno de los impulsores del aumento reciente en la investigación de la ontología. Por ejemplo, modelos para muchos dominios diferentes necesitan representar la noción de tiempo. Esta representación incluye las nociones de intervalos de tiempo, puntos en el tiempo, medidas relativas de tiempo y así sucesivamente. Si un grupo de investigadores desarrolla tal ontología en detalle, simplemente otros pueden reutilizarlos para sus



dominios. Además, si tenemos que construir una gran ontología, podemos integrar varias ontologías existentes describiendo porciones del gran dominio. También se puede reutilizar una ontología general, y extenderla para describir nuestro dominio de interés [NM01].

3. Hacer explícitas las suposiciones del dominio subyacente a una implementación hace posible cambiar estos supuestos fácilmente si nuestro conocimiento sobre el dominio cambia. Suposiciones difíciles de codificar a cerca del mundo del código del lenguaje de programación hace que estos supuestos no sólo sean difíciles de encontrar y entender sino también difíciles de cambiar, en particular para alguien sin conocimientos de programación. Además, especificaciones explícitas de conocimiento del dominio son útiles para los nuevos usuarios que deben aprender el significado de los términos en el dominio [NM01].

4. Separar el conocimiento del dominio del conocimiento operacional es otro uso común de las ontologías. Podemos describir una tarea de configuración de un producto a partir de sus componentes según una especificación requerida e implementar un programa que haga esta configuración independiente de los productos y sus componentes. Entonces podemos desarrollar una ontología de componentes y características de la PC y aplicar el algoritmo para configurar PCs a la medida [NM01].

5. Analizar el conocimiento del dominio es posible cuando existe una especificación declarativa de los términos [NM01]. El análisis formal de los términos es extremadamente valioso tanto para intentar reutilizar ontologías existentes como para extenderlas.

A menudo, una ontología del dominio no es un objetivo en sí mismo. Desarrollar una ontología es similar a la definición de un conjunto de datos y su estructura para que lo utilicen otros programas. Métodos de resolución de problemas, aplicaciones independiente del dominio y agentes de software usan ontologías y bases de conocimiento construidos a partir de ontologías como datos [NM01].

### **2.5.1 ¿Qué es una ontología?**

Una ontología es una descripción formal explícita de los conceptos en un dominio del discurso (clases, a veces llamados conceptos), propiedades de cada concepto que describen diferentes características y atributos del concepto (ranuras, a veces llamadas roles o propiedades) y las restricciones sobre las ranuras (facetas, a veces llamadas restricciones de rol). Una ontología junto con un conjunto de instancias individuales de clases constituye una base de conocimiento. En realidad, existe una línea muy fina entre donde la ontología termina y donde la base de conocimientos comienza [NM01]. Para la IS, una ontología es una especificación formal explícita de cómo representar los fenómenos, conceptos y otras entidades que se supone que existen en algún área de interés (un universo del discurso) y las relaciones que mantienen entre ellos [Bjø06]. En otras palabras, una ontología es un catálogo de conceptos y sus relaciones, incluidas las propiedades así como las relaciones con otros conceptos) [Bjø06].

### **2.5.2 Una Simple Metodología de Ingeniería del Conocimiento**

Muchas metodologías existen para el desarrollo de ontologías, sin embargo no se ha desarrollado aún una metodología estándar; a continuación se discuten cuestiones generales a considerar y se ofrece un posible proceso para el desarrollo de una ontología.

Se describe un enfoque iterativo para el desarrollo de la ontología, esbozando algunas decisiones de modelado que un diseñador tiene que tener en consideración. En primer lugar, se destacan algunas reglas fundamentales en el diseño de la ontología, aunque estas reglas pueden parecer algo dogmáticas. Sin embargo, puede ayudar a tomar decisiones de diseño en muchos casos [NM01]. En segundo lugar, se enuncian los pasos a seguir para su desarrollo.

- 1) No existe una manera correcta para modelar un dominio, siempre hay alternativas viables. La mejor solución depende casi siempre de la aplicación que se tiene en mente y las extensiones que se puede anticipar.
- 2) El desarrollo de la ontología es necesariamente un proceso iterativo.

- 3) Conceptos en la ontología deben estar cerca de los objetos (físicos o lógicos) y las relaciones en el dominio de interés. Estos suelen ser sustantivos (objetos) o verbos (relaciones) en las oraciones que describen el dominio.

Pasos a seguir para el desarrollo de ontologías [NM01]:

Paso 1. Determinar el dominio y el alcance de la ontología

Paso 2. Considerar la reutilización de ontologías existentes

Paso 3. Enumerar términos importantes en la ontología

Paso 4. Definir las clases y la jerarquía de clases

Paso 5. Definir las propiedades de las clases, las ranuras

Paso 6. Definir las facetas de las ranuras

Paso 7. Crear instancias

No obstante, independientemente de las reglas y sugerencias, una de las cosas más importante a recordar es la siguiente: no hay una única ontología correcta para cualquier dominio. El diseño de ontologías es un proceso creativo y no dos ontologías diseñadas por diferentes personas serían similares. Las potenciales aplicaciones de la ontología y la comprensión y visión del diseñador del dominio afectarán, sin duda, las opciones de diseño de ontologías. Se puede evaluar la calidad de una ontología sólo si se utiliza en las aplicaciones para las cuales fueron diseñadas [NM01].

### ***2.5.3 Representación mediante Ontología de una Arquitectura de Referencia***

La elicitación del conocimiento del dominio es crucial en el contexto LPS, para definir una familia LPS con un grado adecuado de generalidad. La AR es la estructura subyacente común a todos los miembros de la familia, que contiene las componentes comunes y variantes funcionales y no funcionales, es decir, el modelo de variabilidad de la AR [3]. Un producto concreto de la familia puede ser derivado instanciando los puntos de variación [3] del modelo de variabilidad de la AR, para seleccionar diferentes variantes o soluciones arquitecturales. Por otra parte, los componentes funcionales deben cumplir con un cierto grado de calidad a fin de realizar correctamente sus tareas [LO16]. Puede haber varias soluciones variantes para satisfacer una propiedad no funcional, dado que esta puede ser resuelta por diferentes

mecanismos técnicos o herramientas en el mercado, y la elección debe ser hecha para seleccionar una solución conveniente para un producto concreto con respecto a otras propiedades no funcionales y/o el costo de la herramienta.

En consecuencia, como ya hemos visto, la elicitación del conocimiento del dominio es una ardua tarea en la ILPS, la mayor parte de él es embebido dentro de la AR, tal como la funcionalidad común central, las propiedades de calidad del dominio, los estilos arquitecturales y las reglas del negocio [LO16].

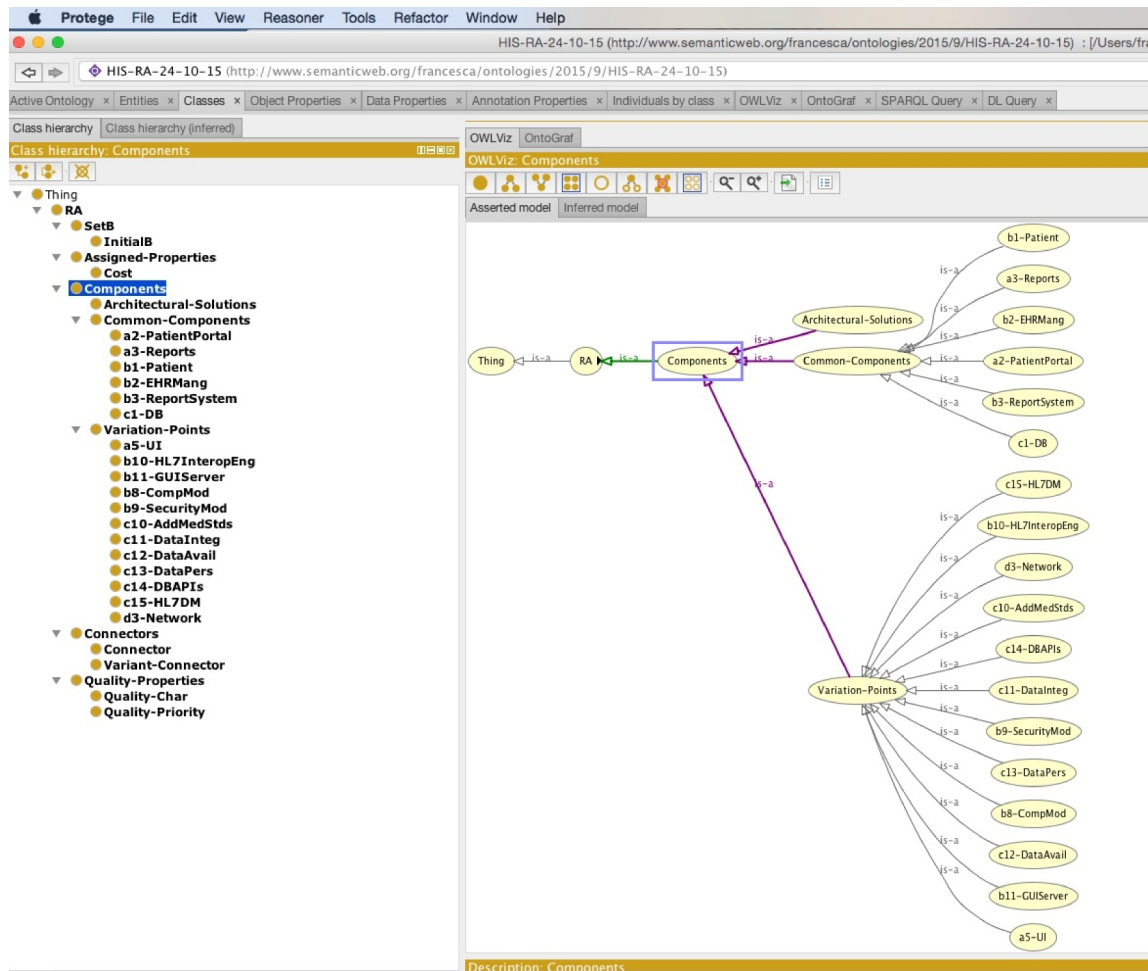
Por otra parte, las ontologías que representan el conocimiento organizado jerárquicamente, han sido utilizadas para especificar y verificar la consistencia del conocimiento del dominio en la LPS [LO16] [LOJ16] y para validar la consistencia en modelos de características tipo FODA<sup>14</sup> [KCH+90]. Una ontología es una especificación explícita de una conceptualización. Las ontologías son ampliamente utilizadas para capturar el conocimiento del dominio de una manera organizada y estructurada; utilizada como herramientas en la ILPS para comprobar la consistencia de las configuraciones de productos concretos derivados de la AR [LO16] [LOJ16].

La ontología *HIS-RA* o “*Healthcare Integrated Information Systems Reference Architecture*”, que se muestra en la Figura 11, ha sido definida para especificar el conocimiento embebido dentro de la AR para el dominio SIS (AR-SIS).

HIS-RA es otra representación de la AR-SIS además de la notación gráfica en UML, y es utilizada como una herramienta para deducir reglas de consistencia para derivar configuraciones arquitecturales validas o soluciones factibles (SF) para generar productos concretos al instanciar la AR-SIS en el ciclo de vida de la IA [LO16][LOJ16]. La representación ontológica HIS-RA, es utilizada para facilitar el razonamiento sobre la consistencia de las relaciones entre los componentes de la AR-SIS, derivando lógicamente reglas de consistencia correctas, para realizar la selección de una configuración arquitectural “conveniente” para un producto concreto, satisfaciendo los requisitos del dominio y del cliente [LOJ16].

---

<sup>14</sup> Feature Oriented Domain Analysis



**Figura 11.** Vista OwlViz en Protegé<sup>15</sup> de la HIS-RA mostrando la jerarquía de clases de componentes comunes, puntos de variación y soluciones arquitecturales; la clase SetB es utilizada en el ciclo IA para la derivación de productos, no se considera en el contexto de este trabajo [LOJ16]

## 2.6 Revisión Documental Sistemática (RDS)

### 2.6.1 Introducción

Debido a que las disciplinas relacionadas con la computación e informática tienen una trayectoria reciente respecto de otras disciplinas de la ciencia, no existen metodologías que guíen el desarrollo de revisiones documentales sistemáticas de la

<sup>15</sup> Protegé 5 en Mac OS X El Capitan

literatura en éstas áreas, como ocurre en disciplinas maduras, como por ejemplo la medicina. En consecuencia, Kitchenham propone un método para realizar *revisiones documentales sistemáticas (RDS)* [Kit04] que se basa en directrices desarrolladas para la investigación médica y que fueron adaptadas para ser usadas por los investigadores en el contexto de la IS [CRC+05].

Antes de emprender una revisión sistemática, el investigador debe asegurarse de que ésta es necesaria. En particular, es recomendable identificar y analizar cualquier revisión sistemática existente acerca del fenómeno o tema de interés con un criterio de evaluación apropiado. Para este fin, se sugiere usar listas de verificación como la siguiente [CRC+05]:

- ¿Cuáles son los objetivos de la revisión?
- ¿Qué fuentes fueron buscadas para identificar estudios primarios? ¿Había alguna restricción?
- ¿Qué criterios se incluyeron o excluyeron y cómo fueron aplicados?
- ¿Qué criterios fueron usados para evaluar la calidad de los estudios primarios y cómo fueron aplicados?
- ¿Cómo fueron extraídos los datos de los estudios preliminares?
- ¿Cómo fueron sintetizados los datos? ¿Cómo se diferencian los estudios investigados? ¿Cómo fueron combinados los datos? ¿Era razonable combinar los estudios?

Las razones más frecuentes que justifican la necesidad de una RDS son [Kit04]:

- ✓ Resumir la evidencia existente concerniente a una tecnología.
- ✓ Identificar algún un vacío en la investigación actual con el objeto de sugerir áreas para investigaciones futuras.
- ✓ Proveer un marco de trabajo y/o los antecedentes necesarios con el objeto de posicionar nuevas actividades de investigación.

También se deben identificar claramente los recursos o fuentes de información con que inicialmente se cuenta para llevar a cabo la revisión (por ejemplo: internet, revistas electrónicas, etc.).

En un estudio [KBB+09] que evaluó el impacto de las revisiones sistemáticas de la literatura recomiendan que los investigadores en el área de IS debieran adoptar una *metodología de IS basada en Evidencia* o “*Evidence-based Software Engineering (EBSE)*” para la acumulación de evidencias. Este método fue inicialmente desarrollado en la medicina, debido a que el asesoramiento médico basado en la opinión de expertos no era tan fiable como el asesoramiento basado en la acumulación de resultados a partir de experimentos científicos. Desde entonces, muchos campos han adoptado este método, por ejemplo, Criminología, Economía, Enfermería, etc.

El objetivo de la EBSE es: “Proporcionar los medios por el cual la evidencia más actualizada desde la investigación puede ser integrada con la experiencia práctica y los valores humanos en el proceso de toma de decisiones con respecto al desarrollo y mantenimiento de software”. En este contexto, la *evidencia* se define como una síntesis de los estudios de mayor calidad científica sobre un tema específico o preguntas de investigación [KBB+09].

Una RDS es una forma definida y metódica de identificar, evaluar y analizar los estudios primarios publicados con el fin de investigar un tema específico, lo que la convierte en una técnica que proporciona una evaluación completa y justa de las evidencias relacionadas con un tema de interés, lo que hace posible obtener de manera sistemática la revisión de la literatura y utilizarla para sintetizar, evaluar e interpretar la relevancia de todas las evidencias relacionadas con un tema específico, área temática o fenómeno de interés [SN07] [BMN+05] [KC07] [KBB+09].

En este contexto, una *evidencia individual* (por ejemplo, un estudio de caso o un estudio experimental divulgado en una publicación de artículo que contribuye a una RDS se llama *estudio primario*, mientras que el resultado de una RDS es un *estudio secundario*. La RDS tiene por objeto proporcionar una visión general de un área de investigación para evaluar la cantidad, calidad y tipo de estudios primarios existentes sobre un tema de interés [KBB+09].

La importancia de una RDS reside en que la mayoría de las investigaciones se inician con una revisión de la literatura de algún tipo. Sin embargo, a menos que la

revisión de la literatura sea exhaustiva y justa, tendrá poco valor científico. Esta es la razón principal para la realización de una RDS, la cual se rige por una metodología sistemática y sólida. Una RDS sintetiza el trabajo existente de manera que es justa y se puede validar que sea justa [KC07].

### ***2.6.2 Razones para la Realización de una RDS***

Hay muchas razones para llevar a cabo una RDS. Las razones más comunes son [Kit04] [KC07]:

- Para resumir la evidencia existente sobre un tratamiento o tecnología por ejemplo, para resumir la evidencia empírica de los beneficios y limitaciones de un método ágil específico.
- Identificar las lagunas en la investigación actual para sugerir áreas para una mayor investigación.
- Proporcionar un marco referencial con el fin de posicionar adecuadamente nuevas actividades de investigación.

Sin embargo, las RDS también pueden llevarse a cabo para examinar la medida en que se apoya la evidencia empírica, se contradicen hipótesis teóricas, o incluso para ayudar a la generación de nuevas hipótesis, lo cual es un caso frecuente.

### ***2.6.3 Ventajas/Desventajas de la RDS***

- Una metodología bien definida hace que sea menos probable que los resultados de la revisión de la literatura estén prejuiciados
- No protege contra el sesgo en los estudios primarios
- Información acerca de los efectos de un fenómeno a través de una amplia gama de ajustes y métodos empíricos
- En el caso de estudios cuantitativos, que también pueden involucrar comparaciones, es posible combinar los datos obtenidos por la RDS utilizando técnicas de meta-análisis y técnicas como DESMET [Kit96]



- Requieren considerablemente más esfuerzo que una revisión de la literatura tradicional, pero los resultados pueden ser más confiables y se presentan con mejor organización.

#### 2.6.4 Estructura del proceso de revisión

Una RDS de la literatura involucra varias actividades discretas [KC07]. En este documento se resumen estas etapas en tres fases principales: *Planificación de la revisión*, *Realización de la revisión*, *Informe de la revisión*.

En la Tabla 2 se describen las fases, las etapas o actividades que incluye cada fase y los artefactos utilizados y generados en cada una de las fases.

**Tabla 2.** Proceso de revisión sistemática

<b>Fases</b>	<b>Etapas o actividades</b>	<b>Artefacto generado</b>
Planificación de la revisión	Justificación de la necesidad de una revisión.	Protocolo
	Especificación de la(s) interrogante(s) de investigación.	
	Desarrollo del protocolo de revisión.	
	Evaluación del protocolo (opcional).	
Realización de la revisión	Identificación de los estudios relacionados.	Estudios primarios aceptados
	Selección de los estudios primarios.	
	Evaluación de la calidad de los estudios.	
	Extracción de datos.	Modelo de datos
	Síntesis de los datos.	Tablas
		Modelo conceptual (opcional)
Informe de la revisión	Especificación de los mecanismos de difusión.	Tablas resumen
	Presentación de los resultados	
	Evaluación del informe (opcional).	

#### 2.6.5 Protocolo de la RDS

El protocolo para una revisión sistemática debe describir la razón de la revisión; los objetivos y los métodos que se utilizarán para localizar, seleccionar y evaluar críticamente los estudios, y para recopilar y analizar los datos de los estudios incluidos. Trata sobre cómo el protocolo aborda las interrogantes de investigación, las estrategias de búsqueda, criterios y procesos de selección, los criterios de evaluación de la calidad, el modelo y proceso de extracción de datos, y el plan de análisis de los

datos. En otras palabras, un protocolo es un plan o conjunto de pasos a seguir en un estudio [KBB+09].

### **2.6.6 Estructura del documento para la RDS**

#### **Planificación**

- Establecimiento de la necesidad de una revisión sistemática
- Preguntas de investigación
  - Identificación de revisiones sistemáticas anteriores
- Desarrollo del protocolo de revisión
  - Selección de las fuentes
  - Selección de estudios
    - Expresiones de búsqueda
    - Criterios de Inclusión y Exclusión
  - Modelo y proceso de extracción de datos
- Revisión del protocolo

#### **Realización**

- Ejecución de la selección
  - Selección de los estudios
  - Evaluación de la Calidad de los Estudios
  - Revisión de la Selección
- Extracción de la información
  - Extracción resultados objetivos
  - Extracción resultados subjetivos
  - Resolución de las divergencias entre los revisores
  - Evaluación de la ejecución

#### **Informe**

- Resumen de los Resultados
  - Revisión de la Selección
    - Presentación de resultados en tablas
    - Presentación de resultados en modelos conceptuales
- Comentarios finales
  - Número de estudios
  - Sesgos en la Búsqueda, Selección y Extracción
  - Variación entre revisores
  - Aplicación de los resultados
  - Recomendaciones

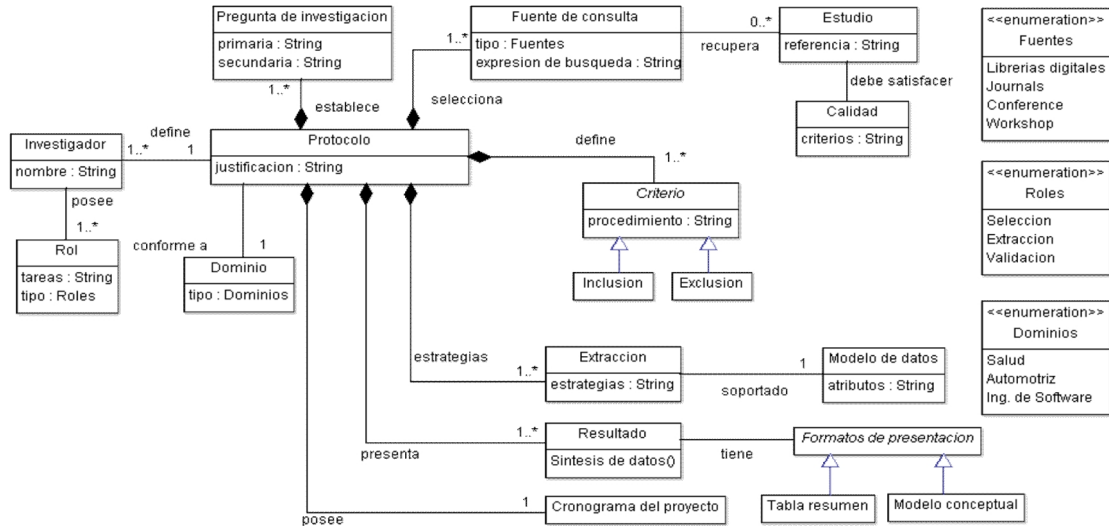
A continuación en las Figuras 12, 13, 14 se presentan tres (3) perspectivas (vistas) del proceso de elaboración de una RDS:

- las funcionalidades principales expresadas como un modelo de casos de uso y representado por un diagrama UML (ver Figura 12);

- los componentes y sus relaciones, expresados como un modelo conceptual, representado en UML (ver Figura 13);
- los pasos a seguir por cada actor (rol), expresados como procesos de negocio para cada una de las fases de la RDS, representados en BPMN (ver Figura 14).



**Figura 12.** Vista funcional de una RDS expresada en diagrama de casos de uso en UML; Fuente: propia



**Figura 13.** Vista conceptual de una RDS expresada en un modelo conceptual; Fuente: propia

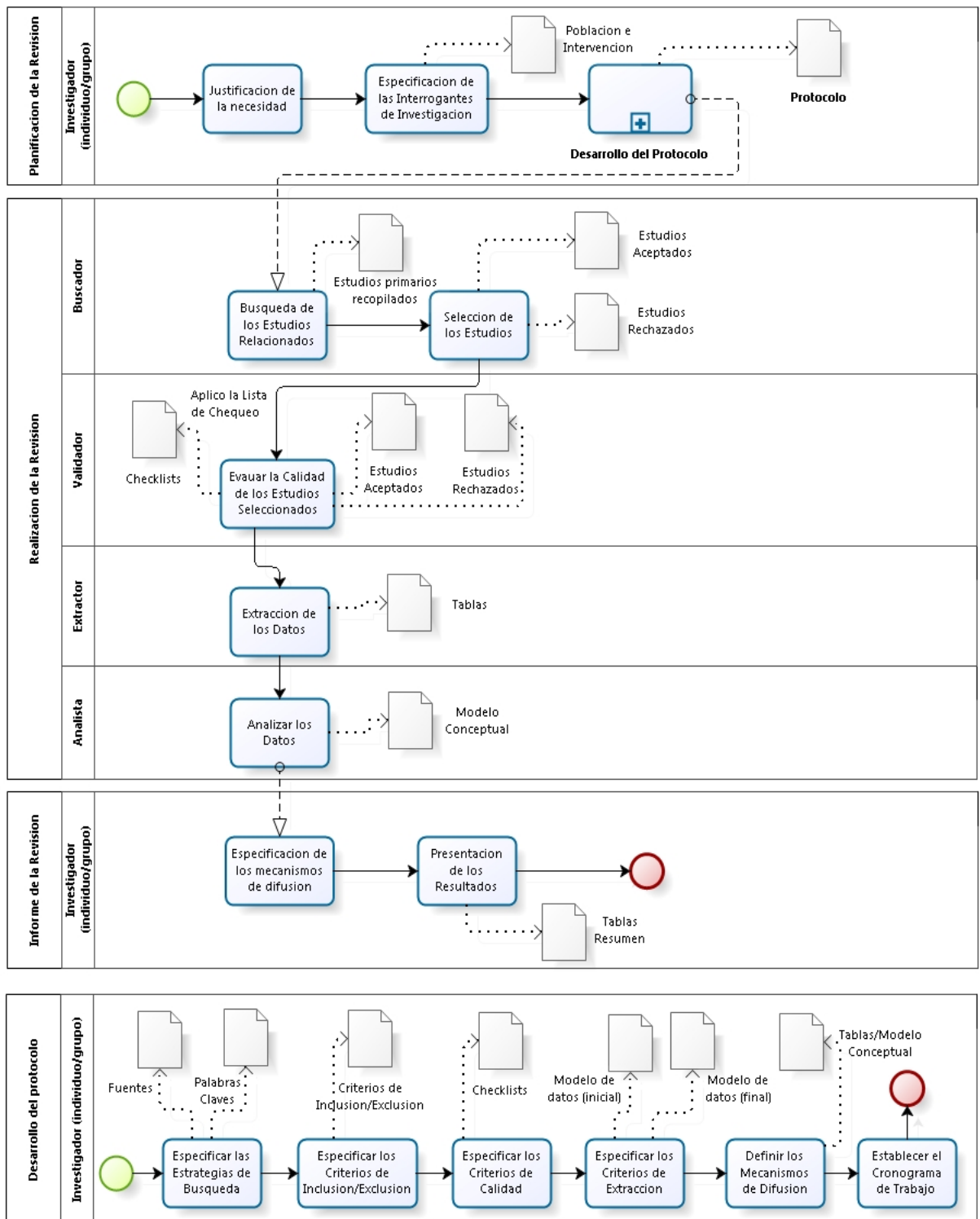


Figura 14. Vista de procesos de negocio de una RDS expresada en BPMN; Fuente: propia

# Referencias

---

- [AAE+12] Aguerre E., Alonso Y, Estala I, Rapetti K. (2012). Arquitectura de Referencia para ASSE (Administración de Servicios de Salud del Estado), Proyecto de Grado, Facultad de Ingeniería, UdelaR, Montevideo, Octubre 2012
- [AAZ11] Allauddin, M., Azam, F., & Zia, M. J. (2011). A Survey of Quality Assurance Frameworks for Service Oriented Systems. *International Journal of Advancements in Technology*, 2(2), 188-198.
- [ABK+13] S. Apel, D. Batory, C. Kästner, and G. Saake, *Feature-Oriented Software Product Lines*. Springer, 2013.
- [AMS07] Asikainen T., Mannisto T., Soininen T. (2007) Kumbang: A domain ontology for modelling variability in software product families *Advanced Engineering Informatics* 21, pp 23–40. Elsevier
- [Bj06] Bjørner, D. (2006). *Software Engineering 3 Domains, Requirements, and Software Design*. Texts in Theoretical Computer Science. An EATCS Series. Editors: W. Brauer G. Rozenberg A. Salomaa. Springer-Verlag Berlin Heidelberg.
- [BMN+05] Biolchini, J., Mian, P., Natali, A. & Travassos, G. (2005). *Systematic Review in Software Engineering*, Technical Report ES67905, PESC - COPPE/UFRJ.
- [Bos00] Bosch J. (2000) *Design and Use of Software Architectures – Adopting and evolving a product-line approach*, Addison-Wesley
- [CN01] Clements P. and Northrop L. (2001) *SPL: practices and patterns*, 3rd ed. Readings, MA, Addison Wesley.
- [CRC+05] Caro, M., Rodríguez, A., Calero, C., Fernández-Medina, E., Piattini, M. (2005). Análisis y revisión de la literatura en el contexto de proyectos de fin de carrera: Una propuesta. *Revista Sociedad Chilena de Ciencia de la Computación*. Noviembre 2005.
- [GAT13] M. Galster, P. Avgeriou, and D. Tofan, “Constraints for the design of variability-intensive service-oriented reference architectures—An industrial case study,” *Information and Software Technology*, 55(2), pp. 428-441, 2013.
- [Gon12] González H., J. (2012). *Integration of Quality Attributes in Software Product Line Development*. Master Thesis, Tesina de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información (ISMFSI).
- [HE90] Henderson-Sellers, B., & Edwards, J. M. (1990). The object-oriented systems life cycle. *Communications of the ACM*, 33(9), 142-159.

- [HHJ08] A. Helferich, G. Herzworm & S. Jesse, Software Product Lines and Service-Oriented Architecture: A Systematic Comparison of Two Concepts, Proceedings of the First Workshop on Service-Oriented Architectures and Software Product Lines, May 2008, CMU/SEI-2008-SR-006
- [HLO+16] ASSIT 2016, J. Herrera, F. Losavio, and O. Ordaz, Product Lines Scoping using ISO/IEC 26550 Reference Model considering Software Quality, 2nd International Conference on Humanity and Social Science (ICHSS 2016), (pp. 194-200), DEStech Publications, Inc. USA.
- [HSS+08] The Practical Guide for SOA in Health Care. A real-world approach to planning, designing, and deploying SOA. Version 1.0. 2008. Healthcare Services Specification Project, Health Level Seven, Object Management Group. All rights reserved.
- [IBM09] IBM. (2009). IBM Service-oriented Architecture (SOA) Solutions. IBM SOA Foundation.
- [INP+09] P. Istoan, G. Nain, G. Perrouin, J. Jezequel. "Dynamic Software Product Lines for Service-Based Systems", INRIA, France, 9th IEEE International Conference on Computer and Information Technology, 2009.
- [ISO05] ISO/IEC 25000. Software and system engineering—Software product Quality Requirements and Evaluation (SQuaRE). Guide to SQuaRE. International Organization for Standardization.
- [ISO11] ISO/IEC 25010, Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, ISO/IEC JTC1/SC7/WG6, 2011.
- [ISO15] ISO/IEC NP 26550: Software and Systems Engineering – Reference Model for Software and Systems Product Lines. ISO/IEC JTC1/SC7, 2015.
- [Ist09] Paul Istoan, Software Product Lines for creating Service Oriented Applications, Masters internship at IRISA Rennes research institute. TRISKELL research team, Rennes June 3, 2009.
- [KBB+09] Kitchenham, B., Brereton, O., Budgen, D., Turner, M., Bailey, J. & Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic review literature. Journal Information and Software Technology 51 7–15.
- [KC07] Kitchenham, B., Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering, Version 2.3, EBSE Technical Report, EBSE-2007-01, Software Engineering Group, School of Computer Science and Mathematics, Keele University, Keele, UK and Department of Computer Science, University of Durham, UK.
- [KCH+90] Kang, K., Cohen, S., Hess, J., Nowak, W. and Peterson, S. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report, CMU/SEI-90-TR- 21, Software Engineering Institute (Carnegie Mellon), Pittsburgh, PA 15213, 1990.
- [Kit04] Kitchenham, B. (2004). "Procedures for performing systematic reviews," Keele University and National ICT Australia Ltd, Tech. ReportTR/SE-0401 and NICTA TR 0400011T.1.

- [Kit96] Kitchenham B. (1996) DESMET: A method for evaluating Software Engineering methods and tools, Tech. Report TR96-09, ISSN: 1353-7776, Department of Computer Science University of Keele, Staffordshire ST5 5BG U.K.
- [KMM08] Kart, F., Moser, L. E., & Melliar-Smith, P. M. (2008). Building a distributed e-healthcare system using SOA. *IT professional*, 10(2), 24-30.
- [Kru95] P. Kruchten, "Architectural blueprints—The "4+1" view model of software architecture," In proceedings of the Conference on TRI-Ada'95, Anaheim, CA, USA, November 1995.
- [LK10] J. Lee, and G. Kotonya, "Combining service-orientation with product line engineering". *Software, IEEE*, 27(3), 35-41, 2010.
- [LMO+14] Losavio, F., Marchan, E., Ordaz, O., & Santos, I. (2014). Arquitectura de Referencia para Sistemas de Salud con Historias Clínicas Electrónicas Interoperables considerando un Modelo de Calidad Estándar, en actas de SCTC 2014, Escuela de Computación, Facultad de Ciencias, UCV, Caracas.
- [LO16] Losavio, F. & Ordaz, O. Reference Architecture Representation by an Ontology for Healthcare Information Systems Software Product Line. IV Simposio Científico y Tecnológico en Computación / SCTC 2016 / ISBN: 978-980-12-8407-9. Universidad Central de Venezuela, Caracas, Venezuela - 09 al 11 de mayo de 2016
- [LOE15] Losavio, F., Ordaz, O. y Esteller, V. (2015a). Refactoring-based Design of Reference Architecture, *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS)*, Vol. 5, No. 1, pp. 32-48.
- [LOJ16] Losavio F., Ordaz O., Jean S., Ontological approach to derive product configurations from a Software Product Line Reference Architecture, *Ciencia y Tecnología (CyT, Universidad de Palermo, Argentina)*, N° 16, 2016, pp. 91-127, ISSN 1850-0870.
- [LOS15] Losavio, F., Ordaz, O. y Santos, I. (2015) Proceso de Análisis del Dominio Ágil de Sistemas Integrados de Salud en un Contexto Venezolano. *Enl@ce Revista Venezolana de Información, Tecnología y Conocimiento*, 12(1), 101-134
- [LVV+08] Lugo, E., Villegas, H., Villegas, A., & Pacheco, J. (2008, January). Lector de historias clínicas electrónicas codificadas en el estándar Health Level 7/Clinical Document Architecture para su aplicación en servicios de telemedicina. In *IV Latin American Congress on Biomedical Engineering 2007, Bioengineering Solutions for Latin America Health* (pp. 904-907). Springer Berlin Heidelberg.
- [MAG+13] Mohabbati, B., Asadi, M., Gašević, D., Hatala, M., & Müller, H. (2013). Combining service-orientation and software product line engineering: A systematic mapping study. *Information and Software Technology*, 55(11), 1845-1859.
- [MGH+13] B. Mohabbati, D. Gasevic, M. Hatala, and H. Muller, A Quality Model and Evaluation Method for Service-Oriented Software Product Line and Configurable Business Processes, *ACM Transactions on Software Engineering Methodology*, September 2013.

- [Mil01] Milanovic, N. (2001). Contract-based Web Service Composition Framework, Doctoral Dissertation, Humboldt University, Berlin.
- [MRS+07] Mykkänen, J., Riekkinen, A., Sormunen, M., Karhunen, H., & Laitinen, P. (2007). Designing web services in health information systems: From process to application level. *International journal of medical informatics*, 76(2), 89-95.
- [MSR10] Medeiros, F. M., de Almeida, E. S., & de Lemos Meira, S. R. (2010). Designing a set of service-oriented systems as a software product line. In *Software Components, Architectures and Reuse (SBCARS)*, 2010 Fourth Brazilian Symposium on (pp. 70-79). IEEE.
- [NM01] Noy, Natalya F., & McGuinness, Deborah L. *Ontology Development 101: A Guide to Creating Your First Ontology*: Knowledge Systems Laboratory, Stanford University. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001.
- [NM07] Nadkarni, P. M., & Miller, R. A. (2007). Service-oriented architecture in medical software: promises and perils. *Journal of the American Medical Informatics Association*, 14(2), 244-246.
- [PBD09] C. Parra, X. Blanc, and L. Duchien, "Context awareness for dynamic service-oriented product lines", In *Proceedings of the 13th International Software Product Line Conference* (pp. 131-140). Carnegie Mellon University, 2009.
- [PBL05] K. Pohl, G. Boeckle, and F. van der Linden, *Software Product Line Engineering - Foundations, Principles, and Techniques*, Springer Verlag, Berlin / Heidelberg, 2005.
- [SMM+09] Shaikh, A., Memon, M., Memon, N., & Misbahuddin, M. (2009, March). The role of service oriented architecture in telemedicine healthcare system. In *Complex, Intelligent and Software Intensive Systems*, 2009. CISIS'09. International Conference on (pp. 208-214). IEEE.
- [SN07] Staples, M. & Niazi, M. (2007). Experiences using systematic review guidelines. *Journal of System and Software*.
- [Sur14] Suryan, W. (2014). *Software quality engineering: a practitioner's approach*. IEEE. Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
- [W3C04] World Wide Web Consortium (2004). *Web Services Architecture Requirements*. W3C Working Group Note, 11 February 2004. Copyright © 2004 W3C® (MIT, ERCIM, Keio) <http://www.w3.org/TR/wsa-reqs>.
- [Wag13] Wagner S. (2013). *Software Product Quality Control*. Springer.



***CAPÍTULO III***

***TRABAJOS RELACIONADOS: MÉTODOS  
DE DESARROLLO DE LPS Y LPSOS EN EL  
CICLO DE VIDA DE LA ID***

---

---

## CAPÍTULO III

### TRABAJOS RELACIONADOS: MÉTODOS DE DESARROLLO DE LPS Y LPSOS EN EL CICLO DE VIDA DE LA ID

En este capítulo se discuten los trabajos directamente relacionados con los temas tratados en la presente investigación y que sirvieron de base para identificar, analizar y describir métodos de desarrollo de LPS y LPSOS en el ciclo de vida de la ID:

En primer lugar, se aborda el tema del diseño de la AR y/o ALP mediante una primera RDS sobre Enfoques y Técnicas para la Construcción de Arquitecturas en un Contexto de Líneas de Productos de Software [HLM14]; se plantean dos enfoques utilizados en la industria; partiendo de un amplio conocimiento del dominio de la aplicación se encuentra el enfoque proactivo y partiendo de los sistemas existentes se encuentra el enfoque reactivo; es de particular interés identificar los artefactos utilizados por ambos enfoques, diferenciándolos ya sea por el tipo de reutilización seleccionado o por la incorporación de nuevos artefactos en cada uno de los métodos propuestos.

En segundo lugar, se abordó el tema sobre métodos de desarrollo para LPS que abarcan el ciclo de vida ID, sobre este particular se describe el método RA&NFR-VAR: Proceso Bottom-Up de Modelado de la Variabilidad de RF y RNF para la Construcción de una Arquitectura de Referencia que fue utilizado como referente previo a nuestra propuesta [LOE15]; y una segunda RDS realizada en el Ámbito de la Ingeniería del Dominio para Líneas de Productos de Software Orientados a Servicios [HLM+14]; con la finalidad de identificar actividades, artefactos y técnicas para definir un proceso sistemático de Análisis del Dominio para LPSOS, con técnicas específicas para la adquisición y análisis del dominio y cómo es realizado el tratamiento de los requisitos de calidad, aspecto relevante en un contexto de producción industrial de software e indispensable para nuestro proceso.

En tercer lugar, se abordó el tema sobre métodos de diseño de arquitecturas de referencia para líneas de productos de software orientadas a Servicios, sobre este particular, una tercera RDS fue realizada sobre Métodos para el Diseño de Arquitecturas de Referencia para LPSOS, el cual integra los enfoques de LPS y SOA [HLO16c]; la intención es la identificación de métodos (fases, etapas, actividades, artefactos y roles) para el diseño del artefacto principal de las LPS, la AR siguiendo un modelo de arquitectura empresarial SOA y conformada por Servicios Web; estos métodos deben incluir en las fases del ciclo de vida ID de LPSOS los siguientes aspectos: - la consideración de las propiedades de calidad, - la identificación, especificación y realización de los servicios como componentes arquitecturales, - la trazabilidad entre los componentes de servicios funcionales y no funcionales, y - la gestión de la variabilidad en cuanto a cómo es resuelta; en nuestro trabajo la gestión de la variabilidad es clave en un contexto SOA y LPS.

Finalmente, otro aspecto de sumo interés para la investigación, son los métodos de desarrollo basados solo en SOA, para ello se describen las diversas metodologías de desarrollo SOA existentes que abordan el diseño arquitectural, comparándolos lo que permitió identificar sus fortalezas y debilidades.

### **3.1 RDS: Diseño de AR y/o ALP [HLM13]**

Para el estudio de la literatura referente a los métodos, técnicas y artefactos para construir la AR y/o la *Arquitectura de Líneas de Productos (ALP)* o "*Product Line Architecture (PLA)*" que es el artefacto medular en el ciclo de vida de la ID para el desarrollo de las LPS y tema central en esta Tesis Doctoral, se siguió la metodología RDS [Kit07] [KBB+09] [Kit04], discutida en el Capítulo II, por sus ventajas en el área de la Ingeniería de Software para explorar, organizar, resumir y analizar las principales tendencias que se han propuesto o utilizado en la construcción de las AR y/o ALP.

En la literatura se pueden encontrar dos tipos de arquitecturas de software para el dominio: AR y ALP. Ambas apuntan a mejorar el desarrollo del software mediante

la estandarización de las arquitecturas de un conjunto de sistemas de software [NOB11b].

Una AR, como ya fue mencionado, acumula conocimiento de un dominio, identifica soluciones abstractas de un problema y promueve la reutilización de las experiencias en diseño alcanzando una sólida y bien reconocida comprensión y conocimiento de un dominio, ofreciendo soluciones estandarizadas para un dominio de aplicación más amplio [NOB11b][GON11]. Según [RRR11] es generalmente construida siguiendo un enfoque *descendente o proactivo*, o “*top-down*”.

Teniendo en cuenta la importancia de las AR, diferentes dominios de aplicaciones, como el automotriz, la aeronáutica y la robótica, han propuesto y utilizado estas arquitecturas [NOB11b].

La ALP, en cambio, es más especializada, centrándose en un subconjunto específico de los sistemas de software existentes de un dominio y ofrecen soluciones estandarizadas para una familia pequeña de sistemas [NOB11b], lo que permite crear sistemas de forma más confiable y rentable [NOB11a].

Una diferencia esencial es que las ALP tratan con la variabilidad entre productos, y se sitúan en un nivel de abstracción más bajo en comparación con las AR [NOB11b].

La ALP es también una arquitectura genérica, pero es construida a partir de productos existentes, por lo tanto “hereda” de cierta forma las propiedades arquitectónicas, por ejemplo el estilo arquitectónico, y las funcionalidades relevantes de cada producto considerado. Por eso es construida mediante un enfoque denominado *ascendente, extractivo o reactivo*, o “*bottom-up*”, ver Capítulo I, Sección 1.1.1 [RRR11] [Kru02], que considera técnicas de reingeniería en diferentes etapas del desarrollo. Es considerado un enfoque útil en la práctica, cuando en la industria no se tiene una LPS y se quiere construir una, a partir de productos que funcionan y han sido elaborados anteriormente, o se tiene un pequeña LPS y se quiere extender con nuevos productos.

A pesar de que las ALP son generalmente construidas mediante un enfoque extractivo/reactivo, el término más común en LPS es AR, por lo tanto en este trabajo es el término que se utilizará.

A partir de aquí, el análisis y alcance del desarrollo de la AR o ALP en esta investigación se caracteriza por identificar los métodos y los artefactos que se utilizan y/o reutilizan para construir la arquitectura a partir de los dos tipos de enfoques para el desarrollo de LPS: proactivo y reactivo [Kru02], su justificación se basa en las nuevas tendencias en el desarrollo de LPS como medio para la personalización masiva del software y de allí los dos enfoques para la construcción de las arquitecturas, estos modelos son: el enfoque proactivo, y el enfoque reactivo [RRR11] [Kru02].

En consecuencia el interés de este trabajo consiste en considerar la AR o ALP, centrándose en los enfoques de desarrollo de la LPS proactivo y reactivo. Esta revisión puede verse completa en el Apéndice [HML13]. Aquí se muestran algunos resultados relevantes que se obtuvieron en las Tablas 3 y 4, para los cuales se localizaron 465 estudios, de los cuales solo 69 estudios fueron aceptados para su análisis, como resultado de la búsqueda en las base de datos de publicaciones, a través de las expresiones de búsqueda:

**Tabla 3.** Construyen la arquitectura por el enfoque reactivo

Estudio	Método	Tipo de Arquitectura
E1 [TYL12]	Extra view to the QADA	ALP
E2 [DMR+05]	Framework RA aplicando SOA	AR
E3 [GG05]	RA based in Reverse Engineering	AR
E4 [MMA+05]	RA basado en patrones	AR
E5 [Eix98]	ARES (Architecture Reference) approach	AR
E6 [GA11]	Empirically-Grounded RA approach	AR
E7 [BFG+04]	PuLSE-DSSA approach	AR
E8 [Jir12]	Prototype of reengineering approach	ALP
E9 [BHR06]	Layered PLA approach	ALP
E10 [AAM03]	KobrA method	ALP
E11 [Har02]	FAST (Family-Oriented Abstraction, Specification, and Translation)	ALP
E12 [DSG+08]	An MDE-based PLA	ALP
E13 [PGG+04]	RA based in Prior Systems	AR
E14 [Hoe04]	Any-time variability approach	ALP

**Tabla 4.** Construyen la Arquitectura por el Enfoque Proactivo

Estudio	Método	Tipo de Arquitectura
E15 [IAO+09]	Framework ACROSeT	RA
E16 [AG08]	ERA (E-contracting Reference Architecture) approach	RA
E17 [PD05]	Interface-Activities Model (IAM) RA	RA
E18 [YBM09]	ProSA-RA	RA
E19 [Pan11]	A component-oriented approach centered on MDE	RA
E20 [DN08]	Cross-domain RA approach	RA
E21 [NUM+11]	ASRA (Agent System Reference Architecture) approach	RA
E22 [OCC+12]	Proactive Design Method for PLAs	ALP
E23 [DRG07]	Decision-oriented approach	ALP
E24 [BGH+99]	SEPA Methodology	RA
E25 [SLC+10]	NEXOF-RA approach	RA
E26 [KKJ11]	Process for Managing Variability	ALP
E27 [DN07]	UML profile approach for variations	ALP
E28 [Lal99]	Style-specific approach	ALP
E29 [ZLZ+07]	Feature oriented approach	ALP

### 3.2 Métodos de desarrollo para LPS que abarcan el ciclo de vida ID:

#### 3.2.1 RA&NFR-VAR [Est16]

A continuación se describe un método para el desarrollo de LPS que abarca el ciclo de vida ID, que fue utilizado como referente para la elaboración de esta Tesis, *RA&NFR-VAR: Proceso Bottom-Up de Modelado de la Variabilidad de RF y RNF para la Construcción de una Arquitectura de Referencia* [LOE15][Est16].

En este proceso se trata de resolver el problema de construir el modelo de variabilidad para una AR, considerando la variabilidad de RF y RNF. Todas las arquitecturas manejadas aquí están representadas por un grafo conexo y no dirigido  $(P, R)$ , donde  $P$  son los nodos o componentes y  $R$  es una relación simétrica que representa los lados o conectores de la arquitectura [LOL+13]; la propiedad de conectividad implica que no hay componentes aislados en una arquitectura. Un componente proporciona/requiere interfaz hacia/desde otro componente; en esta propuesta no se consideran puertos, ni otros tipos de relaciones entre componentes. Solo la vista lógica (componentes y conectores) [Kru95] que expresa aspectos estáticos de la arquitectura [SG96], y será especificada en UML 2.X [OMG05].

En este trabajo se aborda en particular la variabilidad no funcional en un contexto LPS. Un modelo de variabilidad es el mecanismo que permite realizar la

instanciación de la AR; está constituido por componentes arquitecturales “abstractas” o puntos de variación, también denominados en la literatura “*placeholders*”, los cuales son conjuntos que contienen componentes “concretas” (variantes), que serán las instancias del punto de variación; una configuración arquitectónica de un producto específico concreto, se obtiene manteniendo los componentes comunes de la AR, instanciando los puntos de variación [PBL05] y considerando la satisfacción de los requisitos demandados por el cliente.

Es importante destacar que los aspectos no funcionales y sus requisitos de calidad no son tratados en profundidad como los aspectos funcionales en los enfoques conocidos de desarrollo de LPS [Mat04], como por ejemplo los que utilizan los modelos de características tipo FODA o “*Feature-Oriented Domain Analysis*” [KCH+90], que se basa en los requisitos directamente percibidos por el usuario; en cambio los RNF no son generalmente percibido por los usuarios y son más difíciles de detectar dependiendo de la experticia del ingeniero del dominio, del ingeniero calidad o del arquitecto .

El proceso RA&NFR-VAR es general para cualquier dominio y es aplicado al dominio de sistemas de información integrados de salud, con arquitectura de estilo híbrido Eventos/Capas y que sigue un modelo cliente/servidor para la comunicación.

RA&NFR-VAR contempla dos fases principales: *Análisis del Dominio* y *Diseño del Dominio*. La fase de Análisis del Dominio es breve, respecto a la realizada en un enfoque proactivo, porque RA&NFR-VAR es extractivo o bottom-up, es decir que se reutiliza el conocimiento sobre productos existentes del dominio, refactorizando sus arquitecturas. A continuación se presenta el proceso RA&NFR-VAR descrito en forma textual; en [Est16] puede verse el proceso con todos los detalles:

#### ***Análisis del Dominio***

*Entrada:* Descripción del Dominio (RF, RNF, reglas del negocio y propiedades del estilo, descripción informal de las arquitecturas de productos existentes);

*Actividades:*

- *Análisis de Similitud de Componentes:*

A partir de la Descripción del Dominio, se realiza un análisis de similitud semántico de los componentes respecto a las tareas que realizan y se renombran de acuerdo a este análisis, se

construye el *grafo*  $(P, R)$  *conexo*, asociado a cada producto y una *Tabla de Similitud*, con componentes y conectores comunes y variantes;

- *Especificación de las Arquitecturas de los Productos iniciales:*

A partir de los *grafos*  $(P, R)$  asociados a cada producto, se construyen sus representaciones arquitecturales en UML 2.X

- *Construcción automática de la Arquitectura Candidata (AC) inicial:*

Se realiza la unión de los *grafos*  $(P, R)$  de cada producto, preservando los conectores, para determinar el “core” de componentes comunes; las no comunes son consideradas variantes; se construye el *grafo*  $(P, R)$  de una *Arquitectura Candidata (AC)* inicial y se genera automáticamente su representación arquitectónica en UML 2.X. Nótese que AC es un *grafo* *conexo*, dado que el análisis de similitud asegura que la intersección de los *grafos* que la conforman es no vacía.

- *Construcción del Modelo de Calidad del Dominio:*

A partir de la Descripción del Dominio, se extraen las propiedades de calidad. Se adecúan a la nomenclatura del estándar ISO/IEC 25010. Se construye el *Modelo de Calidad del Dominio* o “*Domain Quality Model (DQM)*”, especificado por [ISO11];

- *Construcción del Modelo de Calidad Extendido:*

Se estudian los componentes arquitecturales de la Tabla de Similitud, analizando sus posibles metas no funcionales (propiedades o requisitos de calidad a las cuales los componentes deben responder); estas propiedades pueden no estar presentes en el DQM, si son derivadas de componentes funcionales; se asignan soluciones arquitecturales o mecanismos computacionales, para cada propiedad de calidad, provenientes de la tecnología o de catálogos de patrones y se asignan prioridades de acuerdo a su importancia respecto al dominio; se construye un *Modelo de Calidad Extendido* o “*Extended Quality Model*”, representado por la *Tabla EQM* que corresponde a los “escenarios” comúnmente utilizados en métodos de desarrollo de software conocidos, como por ejemplo ADD<sup>16</sup> [BKB02];

- *Construcción del modelo de variabilidad (Construcción del SIG):*

Se construye un *Grafo de Interdependencia de Objetivos “blandos” o No Funcionales* o “*Softgoals Interdependency Graph (SIG)*”, para cada propiedad de calidad prioritaria. SIG determina la trazabilidad entre un componente funcional y la propiedad de calidad que lo realiza o softgoal; algunas de estas soluciones pueden estar presentes en la Tabla EQM; SIG es un proceso complejo, que puede ser realizado en forma semiautomática; se basa en el enfoque de metas [CNY+00], pero requiere la experticia del arquitecto de software y/o ingeniero de calidad. SIG puede introducir nuevos componentes variantes para satisfacer exhaustivamente los requisitos de calidad del EQM, asignando a cada componente una “operacionalización” o solución arquitectural, que podría no estar presente entre las disponibles en la Tabla EQM, por lo tanto ésta se debe actualizar en caso de haber nuevas operacionalizaciones;

- *Completar AC:*

A partir de la Tabla EQM actualizada se completa la AC inicial con los nuevos componentes variantes introducidos por el SIG y sus conectores; la AC completada es también un *grafo*  $(P, R)$  y su conectividad es asegurada en su construcción; se genera automáticamente su representación en UML 2.X;

### ***Diseño del Dominio***

*Entrada: AC completada, Tabla EQM*

*Actividades:*

- *Construcción de la AR:*

Se agrupan las variantes de la AC completada que realizan tareas similares, en puntos de variación (proceso que se denomina *factorización*), manteniéndose los componentes comunes y sus conectores. AR es un *grafo*  $(P, R)$  *conexo* por construcción. Se genera automáticamente el diagrama UML 2.X;

---

<sup>16</sup> Attribute Driven Design Method, SEI, Carnegie-Mellon



- *Documentación de la AR:*

Se especifican en una tabla las componentes comunes, los puntos de variación con sus variantes y sus conectores correspondientes; se colocan comentarios para cada componente y punto de variación indicando restricciones y opcionalidad. Esta tabla de documentación de AR constituye una entrada a la derivación de productos concretos a partir de AR, en la fase de IA en este proceso bottom-up sustituye al modelo FODA clásico [KCH+90] utilizado en el enfoque top-down. Esta tabla también puede ser representada por una ontología, como se mostró en el Capítulo II, Sección 2.6.

### **3.2.2 Otros métodos de desarrollo basados en SOA y LPS que contemplan ID: RDS [HLM+14]**

La presente RDS [HML+14], al igual que RA&NFR-VAR [Est16], también considera el ciclo ID [PBL05] para LPS que incluye el Análisis del Dominio. Se identifican las actividades, artefactos y técnicas utilizadas en un proceso completo de análisis del dominio, diferenciándolas: a) por los artefactos de entrada y de salida, b) por el enfoque de diseño utilizado (proactivo o reactivo) para el desarrollo de la AR, con el fin de describir como se analiza y realiza en general en la literatura un proceso de análisis del dominio en la ID y c) también es de especial interés identificar las notaciones, como *UML (Unified Modeling Language)* [OMG05], las técnicas como FODA [KCH+90], y los marcos de referencia como SOA, utilizados para obtener los principales artefactos.

El objetivo de esta RDS es entonces identificar las actividades, artefactos y técnicas comúnmente utilizadas en la ID de LPS, y eventualmente con SOA, encontrados en la literatura reciente, para luego integrarlos para definir un proceso único de análisis del dominio para LPSOS.

#### ***A. Realización de la RDS***

Como resultado de la búsqueda mediante las preguntas de investigación y las fuentes de datos, 231 estudios fueron localizados, de los cuales solo 67 estudios fueron aceptados para su análisis. En las estrategias de búsqueda, teniendo en cuenta las preguntas, se utilizaron las palabras clave o descriptores:

*Software Product Line, Service-Oriented Architecture, Domain Engineering, Domain Stakeholders, Domain Acquisition, Domain Analysis, Domain Design, Domain Modeling, Domain Verification, Domain Validation, Analysis Tool*

combinándolos con operadores lógicos AND, OR; para capturar los estudios primarios.

Para la selección final de los estudios aceptados, se realizó un proceso de filtrado que consistió en la aplicación de los criterios de inclusión y de exclusión; en la Tabla 6 se muestra un resumen de los resultados indicando la cantidad de estudios aceptados y rechazados. En la Tabla 5, se describen las propiedades de extracción que serán utilizados a lo largo de la revisión sistemática.

**Tabla 5.** Propiedades de Extracción

<b>Propiedades</b>	<b>Descripción</b>
Estudio	Identificación del artículo objeto de estudio.
Definición	Concepto utilizado en el estudio para definir la ID.
Etapas	Indica las etapas utilizadas en los trabajos analizados para descubrir el proceso de desarrollo del dominio.
Actividades	Indica las actividades realizadas en cada una de las etapas anteriores.
Propósito	Objetivo o meta propuesta por la actividad a realizar.
Artefactos de entrada	Artefactos utilizados como insumos para poder realizar una actividad.
Artefactos de salida	Artefactos producidos como consecuencia de haber realizado una actividad.
Enfoque	Enfoque utilizado para el desarrollo del dominio; puede ser Proactivo o Reactivo.
Comentarios	Aspectos de interés reflejados en la investigación.

**Tabla 6.** Relación de Estudios Localizados y Aceptados según la Fuente de Datos

<b>Fuentes de datos</b>	<b>Artículos</b>	
	<i>Aceptados</i>	<i>Localizados</i>
Google Academic	20	36
Scirus	30	167
ResearchGate	8	10
ScienceDirect	9	18
Total	67	231

De los estudios localizados se descartaron los artículos repetidos con nombres iguales; también se descartaron aquellos artículos, que al examinar sus contenidos se encontró que eran idénticos, aun cuando sus nombres eran diferentes; quedando un total de 67 estudios aceptados. Entre ellos, solo 10 estudios fueron seleccionados [MAL10] [AG13] [QB09] [GB08] [Dol06] [Lap11] [Par11] [SL09] [BGM+11] [LMN10], debido a que abordan y/o describen un proceso de desarrollo para la construcción de LPSOS, que es el objetivo principal de interés del estudio;

A continuación en las Tablas 7 y 8 se muestra la distribución de los estudios aceptados agrupados por categorías o ámbitos de análisis, basados en la terminología utilizada en [Bj06] en las etapas de desarrollo del dominio de aplicación.

**Tabla 7.** Etapas (Categorías) para el Desarrollo del Dominio [Bj06]

Desarrollo del Dominio		
Categorías de análisis		Total
1	Stakeholders Identification	2
2	Domain Acquisition	2
3	Domain Analysis and Concept Formation	32
4	Domain Modeling: (domain design)	18
5	Domain Validation and Verification	0
6	Domain Theory	0
7	Domain Documentation	1
Total		55

**Tabla 8.** Otras Categorías de Interés para la Investigación

Otras categorías de análisis	Total
Domain-Driven Design	2
Quality Model	1
Analysis Tool	0
Design Tool	0
Domain Architecture	9
Total	12

A continuación, un análisis más detallado se llevó a cabo en cada estudio incluido y el informe de la revisión.

### ***B. Informe de la RDS***

En esta última fase, se presentan los resultados analíticos de la revisión sistemática. La extracción de los datos obtenidos y la síntesis del conocimiento considerando cada pregunta de investigación se discute a continuación.

Las respuestas a las preguntas de investigación versan sobre los aportes de la investigación realizada por Dines Bjørner [Bj06] para el desarrollo del dominio de propósito general, del trabajo realizado por Pohl et al. [PBL05] sobre el proceso de ID para LPS, y de los 10 trabajos seleccionados, que particularmente utilizan como arquitectura el marco de referencia SOA.

*Con Relación a las Preguntas “PI1 y PI2”:*

***¿Cuáles son las actividades comúnmente realizadas en las etapas de análisis, diseño e implementación en el proceso de ID bajo el enfoque de LPSOS?***

***Desde el punto de vista de las LPS implementadas con SOA ¿Cuáles son los artefactos comúnmente utilizados y técnicas utilizadas para su construcción en la fase de ID para LPSOS?***

Para responder a estas dos preguntas, se utilizaron las siguientes dos estrategias; primero, de los estudios aceptados inicialmente, se seleccionaron aquellos estudios que abordan o describen un proceso de desarrollo para LPSOS; segundo, para representar los resultados del análisis de estos estudios, y mostrarlos de una forma más amigable, se utilizó un modelo de datos para la sistematización de los datos.

Las Tablas 9, 10 y 11 describen el conjunto de actividades, artefactos generados y técnicas utilizadas respectivamente en la construcción de LPSOS a partir de los 10 estudios analizados. Los 10 estudios analizados abordan el proceso de ID bajo el enfoque LPSOS, basándose en las siguientes 3 etapas: análisis, diseño e implementación.

**Tabla 9.** Actividades Realizadas en las Fases de Análisis, Diseño e Implementación de la LPSOS

<b>Etapas</b>	<b>Actividades</b>
Análisis	Análisis de factibilidad de la LPS Análisis del dominio Análisis de características: similitudes y variabilidades de la LPS Análisis de los activos existentes Análisis de los requerimientos de la familia
Diseño	Identificación de los elementos arquitecturales Especificación de la arquitectura Definición de la técnica de implementación de la variabilidad
Implementación	Construcción de los componentes/servicios Plan de producción o "Roadmap" de los productos de la LPS

**Tabla 10.** Artefactos Desarrollados en las Fases de Análisis, Diseño e Implementación de la LPSOS

<b>Etapas</b>	<b>Artefactos</b>
Análisis	Documento de factibilidad Modelo conceptual del dominio Modelo de características Listado de componentes/servicios disponibles Tabla de identificación de requerimientos
Diseño	Servicios/Componentes Arquitectura de referencia Modelo de configuración Modelo de componentes/servicios

	Modelo de estado Modelo de colaboración, interacción, comunicación. Mecanismos de implementación de la variabilidad.
Implementación	Componentes SPL desarrollados, reutilizados, o comprados a terceros. Composición de los componentes/servicios

**Tabla 11.** Técnicas en las Fases de Análisis, Diseño e Implementación de la LPSOS

<b>Etapas</b>	<b>Técnicas</b>
Análisis	Diagrama de clases (UML) Notación de Modelado de Procesos de Negocios (BPMN) Diagrama de actividades (UML) Diagrama de Casos de Uso (UML) FODA, variantes de FODA Tablas Requisitos Funcionales – No Funcionales
Diseño	Service-Oriented Architecture (SOA) Service-Component Architecture (SCA) Vistas arquitecturales: Diagrama de clases (UML), Diagrama de componentes (UML), Diagrama de estado (UML), Diagrama de colaboración (UML). Lenguaje de Variabilidad Común (LVC) Mecanismos de configuración, generación
Implementación	SCA, SOA Orquestado/configuración/composición

*Con Relación a la Pregunta “PI3”:*

***¿Desde el punto de vista del ciclo ID, cual es el enfoque utilizado para el diseño de la arquitectura de referencia y/o arquitectura de líneas de productos?***

Desde el punto de vista de la ID, el enfoque utilizado para el diseño de la AR y/o PLA [HLM13] por la mayoría de los estudios analizados, fue el enfoque proactivo.

*Con Relación a la Pregunta “PI4”:*

***¿Existen vistas de calidad que tomen en consideración algún estándar o modelo de calidad del producto?***

Existe muy poca evidencia de algún estudio que proponga o utilice alguna vista de calidad, modelo de calidad o implemente algún estándar de calidad del producto, utilizando ambos enfoques LPS y SOA. Solo un trabajo [MGH+13] implementa estrategias que incorporan características de calidad en un enfoque LPSOS, es decir que consideran la calidad desde la perspectiva de QoS para la concepción de la LPS y toman en consideración la arquitectura SOA, sin embargo, aunque esta investigación hace un análisis superficial del modelo de calidad ISO/IEC

9126-1 y del ISO/IEC 25010 [ISO11], argumentando que debe adecuarse a las particulares características de las LPSOS, no se propone una vista de calidad basada en alguno de los estándares antes mencionados.

Por otra parte, se evidenciaron dos trabajos independientes que abordan o proponen un modelo de calidad, uno de ellos describe un procedimiento para incorporar un modelo de calidad para una LPS [TP03], el segundo si aborda un modelo de calidad basado en el ISO/IEC 25010, pero solo para SOA, bajo el paradigma de la *Computación Orientada a Servicios* o “*Service-Oriented Computing, (SOC)*” [GL11].

*Con Relación a la Pregunta “PI5”:*

***¿Existen herramientas de análisis o diseño que apoyen el proceso de ID para el desarrollo de LPSOS?***

No se describen herramientas y/o prototipos que apoyen o aborden el proceso de desarrollo de la LPSOS, ni que utilicen alguna de vista de calidad del dominio aplicando un modelo de calidad o estándar (ISO9126, ISO25010, entre otros); sin embargo existen algunos estudios que abordan el proceso, por ejemplo desde la perspectiva de la LPS [MS10][Gon12][Mon09], particularmente se encuentra una herramienta en línea a través del enlace: [www.splot-research.org](http://www.splot-research.org).

Por otra parte, en el estudio [SBR+07] que aborda el análisis de las características de las herramientas del tipo open-source que apoyan procesos de LPS, muestra que la mayoría de las propuestas se basan o parten del modelado de las características de la LPS [ACL+13][BGP12][LMN08].

### ***C. Discusión***

Es conveniente y necesario describir aquellos aspectos que resaltan de cada uno de los 10 procesos analizados, aparte de las actividades realizadas en las etapas de análisis, diseño e implementación del dominio; entre ellas se destacan las siguientes evidencias: existen muy pocos estudios que abordan adecuadamente la identificación de los *actores* o “*stakeholders*” que son *partes interesadas del dominio*. Particularmente no se identifican claramente los actores involucrados para

efectuar la adquisición del dominio, aspecto fundamental según Bjørner [Bj06], para obtener una visión global del dominio de la aplicación.

Solo algunos estudios [MAL10] [Par11] identifican los roles de los participantes o “*stakeholders*” y en que etapas del proceso de desarrollo del dominio (análisis, diseño e implementación) actúan o toman decisiones.

También existen pocos estudios que aborden algún método, técnica o procedimiento relacionado con la adquisición del conocimiento sobre el dominio.

En los 10 estudios seleccionados para la investigación, se observó que ninguno valida y/o verifica el dominio de la solución desarrollado por los ingenieros de software (arquitectura de referencia, componentes/servicios) contra el dominio especificado por los expertos del dominio del problema (conocedores de los procesos de negocios en la organización). La mayoría de los estudios analizados, solo abordan las fases de análisis y modelado del dominio. Solo algunos describen como implementar el dominio (arquitectura de referencia [NBG11]).

Los autores que describen la AR, no toman en cuenta alguna la vista de calidad del dominio, como la descrita en [LM13], no toman en consideración ningún estándar de calidad, como ISO/IEC9126-1 o su versión actualizada ISO/IEC25010 [ISO11].

Se evidencia en los estudios, que los conocedores del dominio por lo general no son del área de la informática o afines, si la organización requiere una LPS soportada por SOA, no se describe o enuncia como se realiza la transformación de las decisiones del negocio a una notación de modelado de negocio [OMG11], *Modelo de Características* o “*Feature Model (FM)*” [ABK+13] en el caso de la LPS y los servicios que ha de soportar la arquitectura, particularmente para SOA; existe mucha ambigüedad y poca claridad en los estudios analizados.

Solo los stakeholders conocedores del dominio del problema con experiencia o estudios en el campo de la informática tendrían las habilidades o capacidades

necesarias para realizar un ID con éxito o por lo menos con más precisión desde el punto de vista del desarrollo de LPSOS.

Los 10 estudios abordan parcialmente la documentación total del dominio, según lo planteado por Bjørner [Bjø06], referente a las facetas del dominio.

Ninguno de los 10 estudios analizados aborda las 6 etapas (identificación de los stakeholders, adquisición del dominio, análisis del dominio, diseño del dominio, validación/verificación del dominio y desarrollo de la teoría del dominio) para el desarrollo del dominio propuesta por Bjørner [Bjø06].

Se destaca que los 10 estudios abordan el proceso de desarrollo de la ID de forma diferente, es decir, aun cuando los procesos utilizan algunas actividades y artefactos comunes, cada uno de ellos realiza el proceso visto de forma global, con actividades, artefactos y técnicas diferentes y pobremente detalladas.

### **3.3 Métodos de Diseño de Arquitecturas de Referencia para Líneas de Productos de Software Orientadas a Servicios: RDS [HLO16]**

Para el estudio de la literatura referente al tema de LPSOS, el cual integra los enfoques de LPS y SOA, se siguió también la metodología RDS [Kit07] [KBB+09] [Kit04] vista en el Capítulo II, la intención es que a partir de esta RDS se identifiquen métodos (fases, etapas, actividades, artefactos y roles) para el diseño de la AR, artefacto principal de la LPS, siguiendo el modelo SOA y conformada por Servicios Web como componentes.

#### **3.3.1 Aspectos a ser considerados en la RDS**

El enfoque LPSOS requiere considerar los siguientes elementos: la *Gestión de Procesos de Negocios (GPN)* o “*Business Process Management (BPM)*”, la gestión de la variabilidad funcional y no funcional, la representación de una Arquitectura de Referencia Orientada a Servicios (AROS) y la gestión de la calidad. Estos aspectos son considerados prioritarios en la RDS de Dos Santos Rocha y Fantinato [DF13] que presenta enfoques para la aplicación conjunta de LPS y GPN, incluyendo el soporte de SOA para GPN, y en la cual se concluye que la combinación LPS, BPM y SOA es



factible y beneficiosa para la construcción de un proceso sistemático integrado para elaborar una LPSOS. A continuación se describen brevemente cada uno de estos aspectos.

### **A. Gestión de Procesos de Negocios**

GPN en [AMK+09] define un proceso de negocio como un conjunto de actividades que se realizan en coordinación con un entorno organizacional y técnico. Estas actividades se desempeñan de manera conjunta para realizar los objetivos del negocio. Cada proceso de negocio es generado por una sola organización, pero puede interactuar con los procesos de negocios generados por otras organizaciones. Las partes fundamentales de GPN son su ciclo de vida y el *lenguaje de modelado de procesos de negocio o “Business Process Management Language (BPML)”* [AMK+09]. Existen varios BPMLs concebidos en la academia y la industria que se utilizan para representar el flujo de trabajo y los artefactos producidos y/o utilizados durante la realización de las actividades en el ciclo de vida. Algunos de estos son el Diagrama de Actividades e UML y la *Notación de Modelado de Procesos de Negocios o “Business Process Modelling Notation (BPMN)”* [OMG11], los cuales fueron evaluados por un framework en [LK06]. Esta evaluación determinó que *BPMN* es uno de los BPMLs más utilizado en la fase de diseño de un ciclo de vida GPN.

### **B. Gestión de la Variabilidad**

La *Gestión de la Variabilidad* o *“Variation Management”* [CK10] se aplica tanto a SOA como a LPS; los puntos de variación pueden ser implementados ya sea por un único servicio (donde una interfaz de servicios puede ofrecer parametrización por QoS o algún otro mecanismo, o a través de servicios similares que traten cada variación. Mohabbati [GAT13], indica que la variabilidad (vista a un bajo nivel de abstracción) es gestionada teniendo varios servicios que tienen la misma funcionalidad, pero difieren en su QoS.

### ***C. Arquitecturas de Referencia en LPS y SOA***

Otro aspecto importante corresponde a la construcción de la AR, y en particular a las AROS, que apoyen la reutilización sistemática de los activos principales para la configuración de productos de software concretos. Las AROS se han propuesto como un tipo de AR; lo que debe estar claro es que las AROS en el contexto SOA son un modelo de arquitecturas empresariales para la integración de aplicaciones en la empresa, no son arquitecturas de software “reales” en el sentido de Garlan y Shaw [SG96]. En cambio en el contexto LPS, AROS son arquitecturas de software genéricas e instanciables a partir de un modelo de variabilidad, para generar sistemas concretos orientados a servicios de la familia LPS. Una AROS-LPS será entonces para nosotros una AR en un contexto de LPSOS. Varias AROS bajo SOA han sido propuestas para sistemas de gobierno, entornos de trabajo colaborativo, o sistemas de atención de la salud; generalmente no son AROS “puras” solo conformadas por servicios, sino que incluyen componentes que no son servicios. Una RDS realizada sobre las AROS bajo LPS en [DF13], indica que faltan directrices sobre cómo definir las AROS, y en particular como definir las AROS que soporten un alto grado de variabilidad para que se adecúen a un contexto LPS [GAT13].

### ***D. Gestión de la Calidad en Servicios***

En los actuales momentos no se concibe un sistema de software que no tenga la calidad necesaria que satisfaga las expectativas de los usuarios finales. Más aun, las propiedades de calidad para una LPSOS son fundamentales para garantizar su completitud y carácter evolutivo. Por ello, es vital abordar las características de calidad de estos sistemas desde una etapa temprana en el ciclo de vida del desarrollo de software. Particularmente esto se debe hacer en el ciclo de vida de la ID en el proceso de una *ILPS Orientada a Servicios (ILPSOS)*. Los servicios exponen propiedades de calidad de bajo nivel a las que responden en tiempo de ejecución, para intercomunicar con otros servicios; los valores de los QoS se ofrecen a través de parámetros. En general, lo que ocurre con las propiedades de calidad de alto nivel a las cuales también deben responder los servicios compuestos, cuando representan componentes funcionales en el caso de AROS, no se trata en detalles. Los WS son

tipos de servicios que pueden ser descubiertos, especificados y accedidos utilizando XML y protocolos Web estándar. El elemento central de un WS está en la definición estándar de su interfaz en WSDL<sup>17</sup>, que contiene entre otras informaciones, los QoS. Para más detalles sobre la descripción de un WS, ver la sección 1.1.2 parte D, sobre la gestión de la calidad en servicios.

### ***3.3.2 Marcos para la evaluación de métodos para el desarrollo de AR-LPS, SOA, AR-SOA***

#### ***A. Marcos para la Evaluación de los Métodos***

El marco de evaluación o framework, constituido por las Tablas 12, 13 y 14, es utilizado como herramienta de análisis. La Tabla 12 muestra los criterios para evaluar métodos de diseño de AR-LPS. La Tabla 13 muestra criterios para evaluar métodos de desarrollo de sistemas orientados a servicios bajo SOA. La Tabla 14 presenta criterios para evaluar métodos de diseño de AR-SOA. El objetivo de este framework de evaluación no es la comparación de métodos, sino proporcionar un panorama de métodos actuales para tomar decisiones en cuanto a la definición de un método general para el diseño de una arquitectura AROS-LPS.

AROS contempla tres estrategias de desarrollo:

- *bottom-up* o *ascendente (reactivo o extractivo)*: derivar los servicios a partir de componentes existentes, es decir de aplicaciones heredadas o de una AR ya realizada, como por ejemplo la identificada a partir de un proceso “bottom-up” que considera la refactorización de productos existentes del mercado.
- *top-down* o *descendente* o *proactivo*: derivar los servicios a partir de la especificación del modelo del dominio expresado mediante procesos de negocios.
- *middle-out* o *meet-in-the-middle* o *híbrido*: partir de la especificación de los procesos de negocio como orquestaciones de servicios y relacionarlos con los componentes de una AR ya existente.

Para la RDS de este trabajo, se buscan los métodos o enfoques existentes (estado del arte) que describan de forma sistemática las etapas, actividades, roles y artefactos necesarios para realizar una AR en un contexto LPSOS (AROS-LPS),

---

<sup>17</sup> Web Services Definition Language

utilizando cualquiera de los tres frameworks mencionados [Mat04] [KKC+09] [OFF+10].

La aplicación del marco de evaluación proporcionará la base para la definición de categorías detalladas de elementos o aspectos presentes en un criterio. Mediante un conjunto de preguntas, este estudio intenta abordar el alcance de los métodos para identificar las diferencias y similitudes que estos presentan. Como no existe un marco específico para la evaluación de enfoques LPSOS, se van a combinar los frameworks de las Tablas 12, 13 y 14, en un marco único que incluya todos los criterios de evaluación. A continuación se describen en detalles los criterios:

#### ***A.1 Criterios del marco de evaluación para métodos de desarrollo de AR-LPS [Mat04]:***

Hay cuatro criterios esenciales para la evaluación de estos métodos:

- *Contexto del método*: situación del problema, objetivo específico, ciclo de vida abordado.
- *Usuario del método*: personas que solucionan el problema, que rol desempeña, guía proporcionada para aplicar el método.
- *Componentes del método o proceso* para la resolución del problema: 1) modelos subyacentes, 2) lenguaje, 3) definición de los pasos, su secuencia, 4) artefactos, 5) vistas arquitecturales, como el modelo “4+1 vistas” de Kruchten [Kru95]), 6) variabilidad (nivel de abstracción donde la variabilidad es manifiesta: a nivel de proceso de negocio, a nivel de los componentes), 7) herramientas que facilitan la realización del método.
- *Gestión de la calidad (nuevo criterio)*: ¿Cómo el método soporta la especificación de las propiedades de calidad y/o requisitos no funcionales?; el aseguramiento de la calidad del producto es una actividad crucial para el éxito de la industria del software, pero es más importante cuando se trata del desarrollo de la LPS, dado que la reutilización masiva de activos de software hace que sus atributos de calidad (o propiedades medibles de un artefacto de software) impacten en la calidad de los productos de la LPS [Gon12].
- *Validación del método*: casos de estudios donde se compruebe la efectividad del método, el estilo arquitectural elegido satisface las expectativas de los usuarios finales.

#### ***A.2 Criterios del marco de evaluación para métodos de desarrollo de sistemas bajo el modelo SOA [KKC+09]:***

Hay cinco criterios esenciales para la evaluación de un método para desarrollo de sistemas o aplicaciones aisladas bajo el modelo SOA de arquitectura empresarial,

que no involucran AR, sino una arquitectura software concreta de un solo producto o sistema aislado:

- *Concepto de servicio*: para analizar si los conceptos de servicio, servicios de negocios y software, son soportados; un *servicio de negocio* es un conjunto específico de acciones que son llevadas a cabo por una organización, un *servicio de software* expone las funcionalidades de la aplicación que pueden ser reutilizados y compuestos basados en las necesidades u objetivos del negocio, por lo tanto un servicio de software soporta la ejecución de un servicio de negocio.
- *Estrategia de realización o desarrollo*: *top-down*, *bottom-up*, *middle-out* o *meet-in-the-middle* que combina ambos; esta última estrategia es la más recomendada para implementar SOA.
- *Cobertura del ciclo de vida* o fases principales para el desarrollo bajo SOA; según SOMA [Ars04] son: identificación, especificación y realización.
- *Grado de granularidad del análisis*: el nivel de detalle utilizado para describir el método, que va desde un nivel bastante descriptivo, con gran cantidad de detalles, hasta un nivel más abstracto, para describir un proceso más general, flexible y adaptable.
- *Accesibilidad y validez*: Indica que el método debe proveer documentación accesible y poder ser validado.

### ***A.3 Criterios del marco de evaluación para métodos de desarrollo de AR-SOA [OFF+10]:***

El marco de evaluación AR-SOA surge de responder la pregunta de investigación:

***¿Cuáles características de SOA han sido consideradas durante el diseño y desarrollo de la AR-SOA?***

Se presentan tres criterios:

- *Publicación del servicio*: La AR-SOA debe permitir la publicación de la descripción de los servicios directamente a los consumidores, a través de intermediarios, por ejemplo UDDI.
- *Calidad del servicio (QoS)*: La AR-SOA debe permitir utilizar mecanismos para capturar, controlar, registrar y señalar el incumplimiento de requisitos no funcionales descritos en los acuerdos de servicio por su interfaz. Estos acuerdos especifican las características de calidad que proporciona el servicio en ejecución, como el tiempo de respuesta, rendimiento, disponibilidad, entre otros. Estos mecanismos deben ser enlazados a los componentes arquitecturales de alto nivel, a través del Repositorio de Activos.
- *Composición de los servicios*: es el proceso de construir un nuevo servicio a partir de servicios existentes [W3C04]. Todos los servicios se comunican por paso de mensajes, a través de protocolos. La AR-SOA debe permitir la composición de

los servicios: a través de los procesos de negocio, mediante una *orquestración* coordinada de servicios que enfatiza el paso de mensajes, o mediante la *coreografía*, más orientada hacia los aspectos funcionales que el servicio debe proporcionar.

### 3.3.3 Criterios y elementos del Marco Único de Evaluación basado en los tres marcos AR-LPS, SOA y AR-SOA

El objetivo del *Marco Único de Evaluación (MUE)* no es solo proporcionar una visión general de los actuales métodos de ingeniería para la LPSOS, sino además indagar cómo los métodos difieren en cuanto a criterios de diseño. Esto es especificado en la sección del informe de la RDS, donde se muestran los resultados de nuestra evaluación. La aplicación de MUE proporciona la base para la definición detallada de los elementos presentes en los criterios. Mediante preguntas, este marco intenta abordar por ejemplo, madurez, sentido práctico y alcance de los métodos para encontrar las similitudes y diferencias, además, de cómo se identifican los servicios a partir del modelo de procesos de negocio, y cuáles son los componentes que constituirán una AROS-LPS y que implementarán los servicios, incluyendo la relación entre estos componentes y el modelo de variabilidad.

A continuación, las Tablas 12, 13, 14 describen criterios y elementos que conforman el MUE, con sus respectivas preguntas de análisis. Estas preguntas ayudan a la captura de información relevante para cada uno de los criterios de evaluación. El MUE se deriva integrando todos los criterios de evaluación y será utilizado en la RDS para efectuar el análisis de los artículos seleccionados como estudios primarios: son aquellas investigaciones relevantes que quedan para el análisis, luego de haberse aplicado los criterios RDS de inclusión, exclusión y calidad:

**Tabla 12.** Criterios y Elementos de Evaluación para Métodos de Desarrollo de AR-LPS, Adaptado de [Mat04]

Criterios	Elementos	Preguntas de investigación
Dominio de la LPS	Ámbito	¿Cuál es/son el/los dominio(s) de la LPS en la que se enfoca el método? Ejemplo: dominio de los sistemas de información.
Contexto del método	Objetivo específico	¿Cuál es el objetivo específico del método?
	Fase(s) de línea de productos	¿Qué fases de la LPS cubre el método? Por ejemplo, el método abarca solamente las fases de análisis y diseño en el ciclo de vida de la ID.

	Entradas del método	¿Cuál es (son) la (s) entrada (s) del método?
	Salidas del método	¿Cuáles son la (s) salida (s) del método?
Usuario del método	Grupo destinatario	¿Quiénes son los actores que intervienen en el método?
	Motivación	¿Cuáles son los beneficios que percibe el usuario al utilizar el método?
	Habilidades necesarias	¿Qué habilidades necesita el usuario para cumplir las tareas requeridas por el método?
	Orientación	¿Cómo el método guía al usuario mientras se aplica el método?
Componentes del método	Estructura del método	¿Cuáles son los pasos, fases y/o actividades del proceso de diseño que se utilizan para llevar a cabo un objetivo específico del método?
	Artefactos	¿Cuáles son los artefactos creados y gestionados por el método?
	Vistas arquitecturales	¿Cuáles son las vistas arquitecturales que se consideran en el método?
	Lenguaje/notación	¿El método define un lenguaje o notación para representar los modelos, diagramas y otros artefactos que produce?
	Variabilidad	¿Cómo el método soporta la expresión de la variabilidad funcional y no funcional?
	Herramientas	¿Cuáles son las herramientas y/o técnicas que apoyan el método?
Gestión de la calidad en el método	Propiedades de calidad	¿Cómo el método soporta la especificación de las propiedades de calidad y/o requisitos no funcionales?
Validación del método	Madurez	¿Se ha validado el método en casos de estudios prácticos industriales?
	Calidad de la arquitectura	¿Cómo el método valida la calidad del diseño de la AR-LPS?

**Tabla 13.** Criterios y Elementos de Evaluación para Métodos de Desarrollo de Sistemas bajo el Modelo SOA, Adaptado de [KKC+09]

<b>Criterios</b>	<b>Elementos</b>	<b>Preguntas de investigación</b>
Concepto de servicio	Servicios de negocios	¿Cuál es el concepto de servicio soportado?
	Servicios de software o WS	
	Ambos	
Estrategia de realización o desarrollo	Top-down	¿Cuál es el tipo de estrategia utilizada para el desarrollo bajo SOA?
	Bottom-up	
	Middle-out	
Cobertura del ciclo de vida	Identificación	¿Cuál es el grado de cobertura con relación al ciclo de vida SOA?
	Especificación	
	Realización	
Grado de granularidad del análisis	Bajo	¿Cuál es el grado de granularidad del nivel de abstracción del método?
	Moderado	
	Alto	
Accesibilidad y validez	Documentación no accesible	¿La documentación del método es apropiada y sirve de guía para su uso en la práctica?
	Parcialmente documentado	
	Bien documentado (Caso de estudio)	

**Tabla 14.** Criterios y Elementos de Evaluación para Métodos de Desarrollo de AR-SOA, Adaptado de [OFF+10]

<b>Criterios</b>	<b>Elementos</b>	<b>Preguntas de investigación</b>
Publicación del servicio	Directamente a los consumidores	¿La AR-SOA permite la publicación de la descripción de los servicios?
	A través de intermediarios	
Composición de los servicios	Orquestación de servicios	¿La AR-SOA permite que el desarrollo de software pueda ser construido por medio de la composición de servicios?
	Coreografía de servicios	
Calidad del Servicio	Cumplimiento de los requisitos no funcionales de alto nivel	¿La AR-SOA permite el uso de mecanismos para validar que se cumplan los requisitos no funcionales de alto nivel requeridos por la funcionalidad que un servicio desempeña?
	Cumplimiento de los requisitos no funcionales de bajo nivel (QoS)	¿La AR-SOA permite el uso de mecanismos para validar que se satisfagan los requisitos no funcionales de bajo nivel (QoS) descritos en los acuerdos de su interfaz?

### **3.3.4 Realización de la RDS [HLO16]**

En la Tabla 2 se describen las fases, las etapas de cada fase y los artefactos generados en el proceso de la RDS, ver Capítulo II.

#### **A. Planificación de la RDS**

Esta fase define los objetivos de la investigación y el protocolo en que la revisión se ejecutará.

*Identificación de la Necesidad:* Revisar los procesos de desarrollo actuales con el fin de proporcionar una guía para el diseño de una AR para LPSOS, en nuestro caso. Además, de identificar problemas pendientes por resolver y posibles áreas para la mejora de los procesos de desarrollo.

*Desarrollo del Protocolo:* El protocolo es el plan o conjunto de pasos a seguir en un estudio, constituido por las preguntas de investigación, las estrategias de búsqueda, los criterios y procesos de selección, los criterios de evaluación de la calidad, el modelo y proceso de extracción de datos, y el plan de análisis y síntesis de los datos.

Las siguientes Preguntas de Investigación (PI) fueron consideradas en inglés; aquí las colocamos en castellano:



- *PI1. ¿Cuáles métodos que siguen enfoques “bottom-up” y/o “top-down” existen (estado del arte) para pasar de (conversión) una arquitectura basada en componentes a una arquitectura basada en WS o como llevar a cabo esta correspondencia?*
- *PI2. ¿cómo se expresan los requisitos de calidad de las funcionalidades respecto a los QoS de los WS?*

Con relación a la estrategia de búsqueda, las fuentes de búsqueda seleccionadas para encontrar los estudios primarios son los siguientes: Google Academic y ResearchGate.

Se identificaron las palabras claves de búsqueda o descriptores:

*Software Product Line, Service-Oriented Architecture, Methods and Tools Design, Reference Architecture, Web Services, Business Process, Non-Functional Properties, Quality of Service*

combinándolas con operadores lógicos AND, OR; para capturar los estudios primarios.

Los Criterios de Inclusión (CI) utilizados para incluir los estudios relevantes en esta RDS son:

- CI1: Estudios publicados entre 2008 y 2015
- CI2: Que aborde la arquitectura para el desarrollo de Líneas de Productos de Software para Familias de Productos Orientados a Servicios utilizando SOA.
- CI3: Que en su contenido incluya al menos el ciclo de vida de la ID para LPSOS.
- CI4: Que en su contenido describa un método de desarrollo para LPSOS.

Los Criterios de Exclusión (CE) son:

- CE1: Que no aborde la integración de LPS y SOA como alternativa para el desarrollo de productos de software.
- CE2: Que esté escrito en un lenguaje diferente al inglés o castellano y que no esté disponible en formato PDF; o que tenga como contenido solo una presentación en un congreso o workshop realizado por los autores.
- CE3: Si existen trabajos repetidos, se elige uno de ellos.

En la selección inicial de las publicaciones entre los resultados de búsqueda, se considera la lectura de los títulos, palabras claves y resúmenes de las publicaciones encontradas. Las publicaciones serán almacenadas en carpetas independientes de

acuerdo a la expresión de búsqueda utilizada, y de esta forma facilitar la ubicación de los estudios para realizar el análisis con mayor fluidez y comodidad.

Con respecto a la extracción de datos y método de síntesis, se prevé la construcción de tablas para la obtención de datos relacionados con las preguntas de investigación.

Para estandarizar la forma en que la información estará representada, se definió un formato o modelo de datos para recopilación de los datos en los estudios seleccionados. Las Tablas 17, 18, 19 muestran la descripción de las propiedades de extracción o “checklist” que serán utilizadas a lo largo de la RDS. Las Tablas 20, 21 muestran los hallazgos encontrados de la extracción de los datos relevantes en los estudios.

El modelo de datos será utilizado en la fase de Realización de la RDS, para seleccionar los mejores estudios, y luego en la fase de Informe de la revisión, estos estudios serán analizados a través del MUE que se describe en la Sección 3.3.2.

### ***B. Realización de la RDS***

En esta fase se sigue el protocolo previamente establecido. Como resultado de la búsqueda, 331 estudios fueron localizados, de los cuales solo 21 fueron aceptados inicialmente para su análisis a través de tres “checklists”, las Tablas 17, 18 y 19 respectivamente. La Tabla 15 a continuación, muestra un resumen de los resultados indicando la cantidad de estudios localizados, aceptados y rechazados.

**Tabla 15.** Relación de Estudios Localizados y Aceptados según la Fuente de Datos

Fuentes de datos	Artículos		
	<i>Aceptados</i>	<i>Rechazados</i>	<i>Localizados</i>
Google Academic	12	283	295
ResearchGate	9	27	36
Total	21	309	331

De los estudios localizados se descartaron aquellos artículos cuyos contenidos eran idénticos, aun cuando sus identificadores eran diferentes. También, se identificaron estudios similares, que abordaban el mismo proceso y en los cuales participaban los mismos autores, aunque el título del estudio era distinto; para estos

casos, se tomó la decisión de elegir el estudio más completo y actualizado. La Tabla 16 muestra los estudios primarios aceptados según el tipo de evento y año de publicación.

**Tabla 16.** Relación de Estudios Primarios Relativos a Procesos de Desarrollo para LPSOS según Tipo y Año de Publicación

Año	Tipo de evento			Total
	Conferencia	Revista	Tesis	
2008	2	1		3
2009	2			2
2010	2	2	1	5
2011	5	1		6
2012	1			1
2013			1	1
2014	1	1		2
2015		1		1
Total	13	6	2	21

Los 21 estudios primarios describen un proceso para el desarrollo de una LPSOS. A continuación, un análisis más detallado se llevó a cabo en cada uno de ellos. Este análisis se realizó mediante la aplicación de dos “checklists” (Tablas 20 y 21) respecto a las propiedades que nos interesan evidenciar en los estudios (Tablas 17, 18 y 19). Esto permite caracterizarlos con relación a las propiedades presentes en sus respectivos contenidos. Se utiliza la nomenclatura (++), (+), (-), que significa que la calidad de la documentación fue buena, regular con un valor en ambos casos de 1 punto c/u o que carece de esa propiedad con un valor de 0 punto, respectivamente. Este aspecto también influyó en la decisión sobre cuales estudios fueron seleccionados para ser evaluados por el MUE.

**Tabla 17.** Descripción de las Propiedades: Métodos para AR-LPS

Propiedad	Descripción
ID	Ciclo de Vida de Ingeniería del Dominio (ID)
An	Fase de Análisis
Di	Fase de Diseño
Re	Fase de Realización
IA	Ciclo de Vida de Ingeniería de la Aplicación (IA)
<b>Estrategia Desarrollo</b>	<b>Estrategia utilizada para abordar el método [Sva12]:</b>
Td	Top-down (proactivo)
Bu	Bottom-up (extractivo/reactivo)
Hi	Middle out o Meet-in-the-middle (híbrido)
Método	El método describe un proceso sistemático:
Act	Actividades a seguir
E/S	Entradas y salidas

Art	Artefactos producidos/utilizados
Rol	Roles de participación en el proceso
Arq	Vistas arquitecturales
CasoEst	Caso de estudio que valida el proceso
ModVariab	Modelado de la variabilidad funcional y no funcional (calidad) [RLA+10]
Car	En el modelo de características [LKL02]
Arq	En el modelo arquitectural
PN	En el modelo de procesos de negocio
WS	En servicios: un servicio proporciona variantes mediante la parametrización; o varios servicios ofrecen la misma funcionalidad, pero con diferentes QoS (implementaciones)
Estd	Uso de un Modelo de Calidad estándar para especificar las propiedades de calidad [MMR14]: ISO-9126, ISO-25010, McCall, Boehm, FUR PS, Dromey, etc.
EspArq	El método considera la especificación de la AR
Tech	Enfoques y Tecnologías: SCA18; MDA19; MDE20; SOC21
Ont	Ontologías: expresan el conocimiento del dominio; expresan la AR como medio para relacionar el dominio del problema con el dominio de la solución con una terminología común.
Herr	Herramientas: grado en que el método es automatizado; si una herramienta es propuesta o implementada

**Tabla 18.** Descripción de las Propiedades: Métodos para SOA

Propiedad	Descripción
GPN	incluye el ciclo de vida de GPN [AMK+09]
NotacPN	Notación (BPMN, etc.) para el modelado de los procesos de negocios
LengEj	Utiliza el lenguaje BPEL para implementar (ejecutar) los procesos de negocios en la etapa de implementación del ciclo de vida de GPN
EstiloArq	El estilo (modelo) arquitectural es SOA
WS	WS como tecnología para realizar SOA
<b>Métodos de conversión</b>	<b>Método para identificar, especificar y realizar los servicios bajo un enfoque SOA</b>
Car/S	Actividades de como traducir del Modelo de Características a Servicios (“top-down”)
PN/S	Actividades de como traducir de un Modelo de Procesos de Negocios a Servicios (“top-down”)
WS/Arq	Actividades de como traducir los Servicios a Componentes arquitecturales (“top-down”)
Arq/WS	Actividades de como traducir una AR de software (componentes, conectores) a una AR cuyo componentes son Servicios (“bottom-up”)
Métodos de desarrollo	Métodos de desarrollo SOA para las fases de Análisis y Diseño [Sva12]: IBM RUP/SOMA; SOAF; SOUP; SOMA; Erl methodology; Papazoglou methodology
ModRef/ ModConcept	Modelos de Referencia/Modelos conceptuales SOA [MM13]: SOA-RM <sup>22</sup> , SOA-RFA <sup>23</sup> , SOA Ontology <sup>24</sup> ; SOMF <sup>25</sup> ; PIM4SOA <sup>26</sup> ; SoaMI <sup>27</sup>

**Tabla 19.** Descripción de las Propiedades: Métodos para AR-SOA

Propiedad	Descripción
Pub	Permite la publicación de la descripción de los servicios directamente a los consumidores, a través de intermediarios
Comp	Permite que el software pueda ser construido por medio de la composición de los servicios
RNF	Cumplimiento de los requisitos no funcionales de alto nivel y/o de bajo nivel

<sup>18</sup> Service Component Architecture

<sup>19</sup> Model Driven Architecture

<sup>20</sup> Model Driven Engineering

<sup>21</sup> Service Oriented Computing

<sup>22</sup> OASIS Reference Model for SOA

<sup>23</sup> OASIS Architecture Foundation for SOA

<sup>24</sup> Open Group SOA Ontology

<sup>25</sup> Service-oriented Modeling Framework

<sup>26</sup> Platform-independent Model for SOA

<sup>27</sup> SOA Modeling Language

**Tabla 20.** Checklist de las Propiedades para Métodos AR-LPS Relevantes Identificadas en los Estudios Primarios sobre LPSOS

Estudios Primarios	AR-LPS																				Total	
	ID			IA	Estrategia Desarrollo			Método					ModVariab				Esp Arq	Tech	Ont	Herr		
	An	Di	Re		Td	Bu	Hi	Act	E/S	Art	Rol	Arq	Caso Est	Car	Arq	PN	WS					
[ZSC+08]	++	++	++	-	++	-	-	++	++	++	-	++	++	++	-	-	-	-	-	-	++	11
[GB08]	++	+	+	-	-	++	-	++	++	-	-	-	++	++	-	-	-	-	-	-	-	8
[IJP09]	++	++	++	++	++	-	-	++	++	++	-	-	++	++	-	++	-	-	MDE <sup>28</sup>	-	-	12
[RLA+10]	+	-	++	-	++	-	-	++	++	++	-	-	-	-	-	-	++	-	-	-	-	7
[CK10]	++	-	-	-	-	++	-	++	-	-	-	-	++	-	-	-	++	-	-	-	-	5
[GE11]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
[BGF+11]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
[EMA13]	++	+	-	-	++	-	-	++	++	++	-	++	++	++	-	-	-	++	-	-	-	10
[Med10]	++	++	++	-	++	-	-	++	++	++	++	++	++	-	++	-	-	++	-	-	-	12
[BGM+11]	++	++	++	++	++	-	-	++	++	++	++	-	++	++	-	-	-	-	-	++	++	13
[DRH+14]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
[PJ15]	++	++	++	++	-	++	-	++	++	++	++	++	++	++	-	-	-	-	-	-	++	13
[SS12]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
[AG11]	++	++	++	++	++	-	-	++	++	++	++	++	++	++	-	++	++	-	-	-	-	14
[BCC+08]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
[MAG+14]	++	++	++	++	++	-	-	++	++	++	-	++	++	++	-	++	++	++	-	-	-	14
[LMN10]	++	++	++	++	++	-	-	++	++	++	-	++	++	++	-	++	-	-	-	-	-	12
[Gal10]	++	+	-	-	-	++	-	++	++	++	++	-	++	-	-	-	-	-	-	-	-	8
[AMK+09]	++	++	++	++	++	-	-	++	++	++	++	++	++	++	-	++	++	++	MDE	-	++	17
[NTG11]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
[ALH+11]	++	++	++	++	-	++	-	++	++	++	-	++	++	++	++	-	++	++	-	-	-	14
21	21	13	12	8	16	5	0	21	14	13	6	9	14	11	2	5	6	5	2	1	4	

**Tabla 21.** Checklist de las Propiedades para Métodos bajo SOA y AR-SOA Relevantes Identificadas en los Estudios Primarios sobre LPSOS

Estudios Primarios	SOA										AR-SOA			Total				
	GPN		Estilo Arq	WS	Métodos de conversión a Servicios				Métodos de Desarrollo	ModRef/ModConcept	Pub	Comp	RNF					
	NotacPN	LengEj			Car/S	PN/S	WS/Arq	Arq/WS										
[ZSC+08]	++	++	++	++	++	++	++	-	-	-	-	-	-	-	++	-	-	8
[GB08]	-	-	++	-	++	-	-	-	-	-	-	-	-	-	-	-	-	2
[IJP09]	++	-	++	-	++	++	-	-	-	-	-	-	-	-	-	++	-	5
[RLA+10]	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
[CK10]	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
[GE11]	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
[BGF+11]	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
[EMA13]	++	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2
[Med10]	-	-	++	++	++	-	-	-	-	RUP/SOMA	-	-	-	-	+	++	++	7
[BGM+11]	-	-	++	++	++	-	-	-	-	-	-	-	-	-	-	++	-	4
[DRH+14]	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
[PJ15]	-	-	++	++	++	-	++	-	-	-	-	-	-	-	-	++	-	5
[SS12]	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
[AG11]	++	-	++	-	++	++	-	-	-	-	-	-	-	SoaML <sup>29</sup>	-	++	-	6
[BCC+08]	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
[MAG+14]	++	-	++	-	-	++	-	-	-	-	-	-	-	-	-	++	++	5
[LMN10]	++	-	++	-	++	++	++	-	-	-	-	-	-	-	-	++	-	6
[Gal10]	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
[AMK+09]	++	-	++	-	++	++	-	-	-	-	-	-	-	-	-	++	-	5

<sup>28</sup> Model Driven Engineering<sup>29</sup> <http://www.omg.org/spec/SoaML/>

[NTG11]	-	-	++	-	-	-	-	-	-	-	-	-	-	1
[ALH+11]	-	-	++	++	++	++	++	-	-	-	-	++	-	6
21	7	1	21	5	10	7	4	0	1	1	1	10	2	

Luego de la captura de los datos, reflejados a través de las propiedades identificadas en las Tablas 20, 21, aquellos estudios que obtuvieron el mayor número de propiedades consideradas, fueron seleccionados para ser evaluados utilizando las categorías, elementos y preguntas del MUE descrito en la sección 3.3.2 (Tablas 12, 13, 14).

### C. *Discusión*

A continuación se presenta el informe de los resultados analíticos de la RDS, considerando cada pregunta de investigación y analizados a través de la aplicación del MUE, expresado a través de sus criterios y elementos de análisis. La Tabla 22 identifica los 10 estudios más representativos relacionados con procesos de desarrollo de LPSOS, de los 21 estudios primarios inicialmente aceptados y que luego de ser analizados, son los que satisfacen el mayor número de propiedades encontradas en los métodos, este valor es obtenido de la sumatoria de los totales de las Tablas 20, 21. MUE se aplicó a estos 10 estudios, que tratan explícitamente procesos de desarrollo para LPSOS.

**Tabla 22.** Estudios más Representativos entre los Primarios para LPSOS

Estudios	Número de propiedades que satisfacen
ZGCAL [ZSC+08]	19
Istoan [IJP09]	17
Med [Med10]	19
BMKARGBH [BGM+11]	17
PJ [PJ15]	18
AG [AG11]	20
MAGL [MAG+14]	19
LMN [LMN10]	18
AMKGBH [AMK+09]	22
ALHG [ALH+11]	20

### ***B.1 Con respecto a Métodos para AR-LPS (ver Tabla 12)***

#### ***Criterio Dominio:***

Los estudios tratan los dominios: gobierno electrónico o “*e-government*”, comercio electrónico o “*e-business*”, y reservaciones o “*e-travel*”.

#### ***Criterio Contexto del Método: objetivos, fases, entradas, salidas***

*Objetivo:* Se destacan dos propuestas para el desarrollo de LPSOS [IJP09] [AMK+09] que utilizan el paradigma de la ingeniería dirigida por modelos, en inglés Model Driven Engineering (MDE). La principal diferencia entre ambos, es que aplican el enfoque de forma diferente: [AMK+09] lo aplica para desarrollar el ciclo de vida ID para LPS completo; y [IJP09] lo utiliza en la etapa final del ciclo IA, para la generación de los productos de la línea.

*Fases:* Todas las propuestas incluyen el ciclo de vida de la ID para LPS.

*Entradas:* Todas las propuestas parten de que el dominio del problema es conocido.

*Salidas:* Modelo de características [ABK+13] [AMK+09] [ZSC+08] [PJ15] [LMN10] [LKL02], modelo de variabilidad [PBL05], modelo de procesos de negocio [MAG+14] [DF13] [AMK+09] [AG11] [BCC+08], modelo de servicios [IJP09], modelo de componentes [Med10] [LMN10] y AR [GAT13] [OFF+10] [AMK+09] [ALH+11].

#### ***Criterio Usuarios del Método: grupo, motivación habilidades, orientación***

*Grupo:* La mayoría de las propuestas señalan como participantes o actores a ingenieros del dominio, arquitectos e ingenieros de software y analistas SOA. Los roles más representativos son: “*Business Analyst*” o Analista de Negocio que es el responsable de verificar que los candidatos a orquestación de servicios representen de forma precisa la lógica del negocio; “*Domain Architect*” o Arquitecto del Dominio para la identificación de los componentes que proveen la implementación de las operaciones expuestas por los servicios; “*SOA Architect*” o Arquitecto SOA para la identificación de los servicios candidatos; “*Domain Designer*” o Diseñador del Dominio and “*Service Designer*” o Diseñador de Servicios. Para [Med10] los más importantes son: Analista de Negocio, Arquitecto SOA y Arquitecto del Dominio.

*Orientación:* Todas las propuestas indican al usuario como realizar las actividades en cada una de las fases del método.

*Motivación y habilidades:* Ninguna de las propuestas describe las habilidades requeridas por los usuarios de los métodos para poder realizar de forma efectiva las actividades presentadas.

### ***Criterio Componentes del Método: estructura, artefactos, vistas, lenguajes, variabilidad y herramientas***

*Estructura:* En cuanto a la definición de los pasos y su secuencia, existe consenso en determinar el alcance de la LPS mediante el análisis de sus características expresado a través del modelo de características, esto es desde el punto de vista estructural, y desde el punto de vista de su comportamiento, se realiza un análisis de los procesos de negocios utilizando por lo general BPMN para expresarlos, ambos conceptos son muy utilizados en la especificación de la variabilidad en la LPSOS.

En [AMK+09] para lograr la integración LPS-MDE-SOA, se comienza por la definición de los procesos de negocio y por la especificación del conjunto de actividades coordinadas para el cumplimiento de los objetivos organizacionales. Luego, los elementos de software necesarios para lograr estos objetivos (los servicios) son identificados, y para cada servicio, se identifican y desarrollan el conjunto de interfaces realizando la especificación del servicio, así como las correspondientes implementaciones de servicio.

*Artefactos:* el modelo de características [LKL02] es el más utilizado, seguido del modelo de procesos de negocios [MAG+14] [AG11], el modelo de variabilidad [ZSC+08] [PJ15] [LMN10], y la AR [ALH+11].

*Variabilidad:* [Med10] indica que la gestión de la variabilidad es una actividad clave en ILPSOS, en la cual los servicios proporcionan la flexibilidad de las LPS y de sus productos, diferenciándolos en el momento de establecer el enlace de los servicios: en tiempo de diseño (estáticamente) y en tiempo en ejecución (dinámicamente). La variabilidad en la mayoría de los estudios analizados se realiza en el modelado de características, en algunos estudios ésta es realizada a través de los procesos de negocios [AMK+09], y en un solo estudio [ALH+11] es realizada a nivel de los servicios y de los componentes, expresados mediante una AR.

También en [AMK+09] se indica que la mayoría de los enfoques de gestión de procesos de negocio enfatizan el desarrollo de un proceso de negocio, y a partir del mismo, se derivan muchas variantes, especializadas según diversas necesidades de la organización. Los procesos de negocio pueden tener algunas piezas comunes para un grupo de diferentes casos de aplicación, donde implica cierta variabilidad en la selección de las actividades de un proceso de negocio. Una funcionalidad común puede ser desarrollada como un activo reutilizable para la creación de nuevas variantes de los procesos de negocio en un dominio. Además, los puntos de variación, derivados de las propiedades no funcionales y tecnológicas de las actividades, proporcionan la funcionalidad del negocio con diferentes propiedades no funcionales y tecnológicas. Tal variabilidad facilita la configuración flexible del grupo de procesos de negocio dirigida por la calidad [AMK+09].

*Vistas:* Con relación a las vistas arquitecturales, la especificación de la arquitectura es la actividad, en la cual la arquitectura es documentada utilizando diferentes vistas y



notaciones para representar los intereses de los diferentes actores involucrados. Existen cuatro estudios [Med10] [ZSC+08] [AG11] [LMN10] que proponen la utilización de vistas arquitecturales similares al enfoque del 4+1 vistas [Kru95].

En [AG11] se especifican 5 vistas: Vista de Características, Vista de Contrato de Servicio, Vista de Procesos de Negocio, Vista de Interface de Servicios y la Vista de Coordinación de Servicios. En [Med10] las 4 vistas arquitecturales consideradas son: Vista de Capas; Vista de Integración; Vista de Componente y Vista de Interacción. En [ZSC+08] se especifican 3 vistas arquitecturales: Vista de Coordinación, Vista de Aplicación, Vista de Datos. En [LMN10] se propone un estilo arquitectural heterogéneo, con tres niveles de descomposición asociados a tres estilos arquitecturales representados mediante 3 vistas: Vista Arquitectural de Servicios, Vista Arquitectural de Comunicación entre Procesos, Vista Arquitectural (en el ADL<sup>30</sup> C2). Solo los estudios [Med10] [LMN10] consideran la Vista de Componentes, donde existe una correspondencia entre las interfaces de los servicios y los componentes que las implementan.

*Lenguajes:* Los lenguajes utilizados por el método para el modelado arquitectural, [Med10] [ZSC+08] [AG11] [LMN10] utilizan la notación UML para expresar algunas de las vistas arquitecturales, especialmente [ALH+11] la utiliza para especificar la AR; para modelar los procesos de negocios, en [AMK+09] [BGM+11] utilizan BPMN y en [MMR14] [AG11] el diagrama de actividades se expresa en UML.

*Herramientas:* En cuanto a las herramientas de soporte al proceso utilizadas, solo cuatro trabajos mencionan alguna [AMK+09] [ZSC+08] [BGM+11] [PJ15], que abordan parcialmente alguna parte del método propuesto; en [AMK+09] se utiliza FeatureMapper<sup>31</sup> para especificar el modelo de características y relacionar las características con los artefactos de software que las implementan; en [BGM+11] utilizan un enfoque basado en ontologías, donde utilizan el modelo de características [LKL02] para la generación automática de composiciones de servicios ejecutables, para ello usan un lenguaje semántico de WS y utilizan un framework que soporta la composición de WS a nivel semántico, denominado Web Service Modelling Ontology (WSMO), donde las composiciones de servicios son codificadas en el lenguaje WSML, este framework tiene una herramienta que ejecuta estas composiciones, denomina Web Service Modelling eXecution environment (WSMX)<sup>32</sup>, que permite generar y ejecutar la orquestación y coreografía de los servicios. También utilizan ATLAS<sup>33</sup> Transformation Language (ATL) para traducir del modelo de características al WSML. En [ZSC+08] Feature Plug-in<sup>34</sup> y [PJ15] FeatureIDE<sup>35</sup> son utilizadas para especificar el modelo de características.

---

<sup>30</sup> Architecture Description Language

<sup>31</sup> <http://featuremapper.org/>

<sup>32</sup> <http://www.wsmx.org/>

<sup>33</sup> <https://eclipse.org/atl/>

<sup>34</sup> <http://gsd.uwaterloo.ca/fmp>

<sup>35</sup> [http://www.witi.cs.uni-magdeburg.de/iti\\_db/research/featureide/](http://www.witi.cs.uni-magdeburg.de/iti_db/research/featureide/)

### ***Criterio Gestión de la Calidad en el Método: propiedades de calidad***

*Propiedades de calidad:* Lamentablemente, la calidad en general es muy poco abordada en la LPSOS y tratada tarde en la etapa de diseño, solo dos estudios [Med10] [MAG+14] trataron este aspecto de forma más precisa. En [MAG+14] se indica que en la fase de diseño del dominio se debe incluir la especificación de las propiedades no funcionales, debido a que los NFR están entrelazados y relacionados con los requisitos funcionales, y que puede haber varios servicios que proporcionan la misma funcionalidad, pero que difieren de la implementación de su QoS. Para ello, las características en el modelo de características tipo FODA [LKL02] son etiquetadas con los rangos de calidad que serán soportados por la arquitectura de la línea de productos, ayudando al ingeniero de servicios y de software a evaluar el impacto de las características variantes seleccionadas acorde a las características de calidad que los servicios proveen. Por su parte [Med10] presenta una técnica para identificar servicios a partir de los atributos de calidad, analizando escenarios de atributos de calidad para identificar los servicios que ayuden al cumplimiento de los atributos de calidad arquitectural, para ello, dispone de un conjunto de patrones de diseño SOA que permiten satisfacer algunos atributos de calidad. Ninguna propuesta utiliza un estándar para especificar la calidad.

### ***Criterio Validación del Método: madurez, calidad de la arquitectura***

*Madurez:* Todas las propuestas analizadas presentaban un caso de estudio, donde se evaluaba su factibilidad.

*Calidad de la arquitectura:* Solo dos propuestas evalúan la calidad arquitectural: [MAG+14] lo hace desde el punto de vista de los usuarios finales del método, mediante la evaluación del desempeño y escalabilidad del método; [Med10] presenta métricas para medir la cohesión y el acoplamiento entre los componentes a nivel arquitectural.

## ***B.2 Con respecto a Métodos para SOA (ver Tabla 13)***

### ***Criterio Concepto de Servicio: negocio, WS***

En términos generales la mayoría de los métodos abordan ambas definiciones: servicios de negocios y WS, primero identifican cuales son los servicios necesarios para dar soporte a las funcionalidades del negocio y luego identifican cuales WS son requeridos.

### ***Criterio Estrategia de realización o desarrollo: top-down, bottom-up, middle-out***

“Top-down” fue la más utilizada, con un 76,2% del total de estudios analizados, seguido del “bottom-up” con un 23,8%; en ninguno de los métodos se abordó un enfoque híbrido.

***Criterio Cobertura del ciclo de vida: Identificación, Especialización, Realización***

Solo un estudio [ALH+11] abarcó en su análisis el ciclo de vida completo SOA, el resto de los estudios solo se enfocan en la presentación de propuestas para la fase de identificación de los servicios.

***Criterio Grado de granularidad del análisis: Bajo, Moderado, Alto***

El nivel de abstracción en general es alto, la identificación de los servicios es la actividad más realizada en los estudios [Med10] [AMK+09] [BGM+11] [PJ15] [AG11], la especificación es muy poco abordada y la realización casi inexistente. Sin embargo, dos estudios ameritan mencionarse: en [AMK+09], en el paradigma SOA, las actividades en un grupo de procesos de negocio pueden ser implementadas por servicios o delegadas a los usuarios. Cada actividad, puede tener uno o más servicios que pueden realizarla. Los servicios pueden ser comunes y reutilizados entre diversos procesos de negocios. Diferentes servicios pueden proporcionar funcionalidad del negocio para soportar distintos requisitos tecnológicos, como la comunicación a través de diversos medios de transporte. Además, pueden proporcionar funcionalidad del negocio con diferentes propiedades no funcionales, como seguridad y rendimiento, expuestos por diferentes implementaciones de interfaces de servicios y composición de servicios. En [ALH+11], la variabilidad de una LPS es realizada a nivel de los servicios y componentes, expresados en una AR. Ninguno de los trabajos revisados aborda tener una AR y luego adaptarla a servicios.

***Criterio Accesibilidad y validez: Documentación no accesible, parcialmente accesible, bien accesible***

Todos los métodos analizados presentan una buena documentación y presentan un caso de estudio, donde se detallaba las actividades a seguir y se evaluaba su validez.

***B.3 Con respecto a Métodos para AR-SOA (ver Tabla 14)******Criterio Publicación del servicio: directamente a consumidores, por intermediarios***

Este criterio no fue abordado en los métodos analizados.

***Criterio Composición de los servicios: orquestación, coreografía***

La mayoría de las propuestas presentan como alternativa para la composición de los servicios la orquestación, implementada a través de los procesos de negocios, quienes coordinan el llamado a los servicios.

***Criterio Calidad del Servicio: cumplimiento de requisitos no funcionales de alto nivel y de bajo nivel***

Ninguno de los métodos evalúa la calidad de servicio a un alto nivel de abstracción en las arquitecturas presentadas, como por ejemplo desempeño, escalabilidad, interoperabilidad.

**3.4 Métodos de desarrollo basados solo en SOA [Sva12] [AGA+08] [MR12]**

En esta sección primero se aborda el análisis donde se evalúan las características que satisfacen varias metodologías para desarrollar sistemas de software bajo SOA [Sva12], luego se describe la metodología RUP/SOMA de IBM [AGA+08] debido que es la más divulgada y completa, y por último el trabajo realizado en [MR12] que utiliza la metodología RUP/SOMA, que es donde se encuentra el proceso de desarrollo mucho más detallado con un caso de estudio.

**3.4.1 Metodologías SOA [Sva12]**

Este trabajo tiene como objetivo presentar el estado del arte de las metodologías más ampliamente conocidas enfocadas en el marco de referencia empresarial SOA, describiendo las soluciones propuestas para las fases de análisis y diseño. Se discuten las características según las cuales se comparan estas metodologías. Se analizan las metodologías SOA existentes. Se proporcionan los resultados de la comparación. Tablas 23, 24 y 25 respectivamente.

**Tabla 23.** Características para el análisis de las Fases de Análisis y Diseño de Metodologías SOA [Sva12]

<b>Características</b>	<b>Descripción</b>
<b>Estrategia</b> de análisis y diseño SOA	<b>Estrategias de desarrollo</b> (descendente “ <i>top-down</i> ”, ascendente “ <i>bottom-up</i> ” e híbrido “ <i>meet-in-the-middle</i> ”)
<b>Cobertura</b> de análisis y diseño	Las fases de análisis y diseño orientadas a servicios de las metodologías SOA que serán analizadas y comparadas pueden ser divididas en <i>cinco actividades</i> principales que fueron <i>refinados</i> de nuevo en <i>pasos</i> . Los siguientes pasos serán <i>utilizados para la evaluación de la cobertura del análisis y diseño SOA</i> .
	<b>Paso 1: Análisis del Negocio de la Organización Objetivo</b> (objetivos y metas del negocio)
	<b>Paso 2: Planificación de Proyectos SOA</b>

	(alcance del proyecto SOA y estrategia de entrega de los servicios: creándolos desde cero, a partir de componentes existentes, comprando a terceros)
	<b>Paso 3: Identificación del Servicio</b> (Identificación de los servicios candidatos; todos los requisitos funcionales y no funcionales para el desarrollo de SOA son recolectados.
	<b>Paso 4: Análisis y Especificación de los Servicios</b> (Seleccionar cuales servicios candidatos serán desarrollados; los servicios son agrupados por funcionalidad: entidad de negocio, aplicación, proceso de negocio). Las especificaciones de los procesos de negocio serán agrupados con los servicios que fueron creados.
	<b>Paso 5: Decisiones de la Realización del Servicio</b> (Documentar las decisiones de realización).
<b>Grado de prescripción:</b>	Las metodologías SOA varían desde la más a la menos descriptiva. Los parámetros disponibles son: fases, actividades, pasos y entradas, salidas (artefactos) para cada paso.
<b>Adopción de técnicas y notaciones existentes:</b>	La mayoría de metodologías SOA se basan en técnicas como OOAD, CBM, BPM, EA y en notaciones como UML y BPMN, mientras que otras no abordan notaciones y técnicas específicas.

Este trabajo contribuye a esbozar las fortalezas y debilidades de las metodologías SOA propuestas y se centran en las fases de análisis y de diseño de SOA, proporcionando un análisis en profundidad y una comparativa de acuerdo con las características especificadas [Sva12].

**Tabla 24.** Análisis de Metodologías SOA Existentes [Sva12]

<b>Metodología</b>	<b>Descripción</b>
<b>IBM RUP/SOMA</b> [AGA+08]	Es una metodología integral desarrollada por IBM en un esfuerzo de aportar aspectos únicos de SOMA a RUP <sup>36</sup> . Sin embargo, debido a que SOMA es una metodología propietaria de la IBM, su especificación completa no está disponible. La metodología consiste de cuatro fases: <i>Análisis de transformación del negocio, identificación, especificación, y realización de los servicios.</i>
<b>Marco Arquitectural Orientado a Servicio (SOAF)</b> [EAK06]	Esta metodología consta de cinco fases principales: <i>obtención de la información, identificación del servicio, definición del servicio, realización del servicio, hoja de ruta y planificación.</i> SOAF pretende facilitar las

<sup>36</sup> Rational Unified Process

	actividades de identificación, definición y realización de los servicios mediante la combinación de un modelado descendente “top-down” de los procesos de negocio existentes con un análisis ascendente “bottom-up” de las aplicaciones existentes.
<b>Metodología de Papazoglou</b> [PH06]	Proporciona una metodología de desarrollo basada en SOA que abarca un ciclo de vida completo. Este se basa en RUP, Desarrollo Basado en Componentes (CBD) y BPM. El modelo de proceso es iterativo e incremental y comprende una planificación anticipada o preparatoria y ocho fases principales: <i>Análisis del Servicio, Diseño del Servicio, Construcción del Servicio, Pruebas del Servicio, Aprovisionamiento del Servicio, Despliegue del Servicio, Ejecución del Servicio y Monitoreo del Servicio.</i>
<b>Metodología de Thomas Erl</b> [Erl05]	Es una guía paso a paso a través de dos fases principales: análisis y diseño orientado a servicios. El <i>análisis orientado a servicios</i> comprende tres pasos principales: <i>definir requerimientos del negocio, identificar sistemas de automatización existentes y modelar servicios candidatos</i> y puede ser dividido en dos partes principales: la primera parte en la que los requisitos del negocio son definidos y la segunda parte en que los servicios candidatos son modelados.
<b>Proceso Unificado Orientado a Servicio (SOUP)</b> [Mit17]	SOUP es una metodología de IS híbrida que está dirigida a proyectos SOA. Como su nombre lo indica esta metodología se basa principalmente en RUP. Su ciclo de vida consta de seis fases: <i>Incepción, Definición, Diseño, Construcción, Despliegue y Soporte.</i> La metodología SOUP puede ser utilizada en dos variantes ligeramente diferentes: una adoptando RUP para proyectos SOA iniciales y otra adoptando una mezcla de RUP y XP para el mantenimiento de las aplicaciones SOA existentes.

### ***A. Comparación de Metodologías SOA***

Las metodologías SOA fueron comparadas utilizando las características descritas en la Tabla 23 al esbozar las principales diferencias, ventajas e inconvenientes. La comparación resultó en las siguientes apreciaciones [Sva12]:

**Tabla 25.** Resumen de la Comparativa de Metodologías SOA [Sva12]

Metodología	Categorías de Análisis			
	<i>Estrategia</i>	<i>Cobertura</i>	<i>Grado de prescripción</i>	<i>Adopción de técnicas y notaciones existentes</i>
IBM RUP/SOMA	hibrida	5 pasos definidos en tabla 23	mayor	BPM, UML, BPEL, WSDL, WS-BPEL
Thomas Erl	descendente “top-down”	Paso 3, 4 y 5 definidos en tabla 23	buen grado	BPM, WSDL, WS-BPEL, WS-* specifications (WS-Security, WS-Addressing, etc.)
SOUP	hibrida	Paso 3 y 4 incompleto	carece	carece
SOAF	hibrida	Paso 3 y 4 incompleto	carece	carece
Papazoglou	hibrida	Solo recomendaciones para el diseño y especificación de los servicios	carece	CDB, BPM, BPMN, WSDL, BPEL, UML

### ***B. Conclusiones***

El objetivo del trabajo de [Sva12] fue comparar las más ampliamente conocidas y populares metodologías de desarrollo SOA al proporcionar un análisis profundo de las fases de análisis y diseño orientadas a servicios. En este trabajo se analizan y comparan las siguientes metodologías SOA: IBM RUP/SOMA, SOAF, metodología de Thomas Erl, metodología de Papazoglou y SOUP.

La investigación mostró que: las metodologías SOA analizadas varían en el grado de prescripción de las más prescriptivas, a las menos prescriptivas permitiendo al usuario personalizar y adaptar la metodología para concretar el alcance del proyecto. Además, la mayoría de las metodologías analizadas SOA se basan en incorporar existentes y probadas técnicas, notaciones, tales como: OOAD, CDB, BPM, WSDL, BPEL, UML, lo que significa que las técnicas son todavía aplicables y

se ofrecen nuevas para el desarrollo de SOA, pero carece de un nuevo método para organizar el proceso de desarrollo SOA [Sva12].

La mayoría de las metodologías SOA analizadas proponen la estrategia híbrida o middle-out para el análisis orientado a servicios, lo que significa que la mayoría de los proyectos SOA no comienzan en un lugar vacío “desde cero” y la mayoría de ellas está dirigida a cambiar sistemas heredados o “*legacy*” [Sva12].

En conclusión, podemos decir que mucho ya se ha hecho en esta área, pero todavía hay una falta de madurez descriptiva y validada de caso de estudios en prueba de conceptos, de las metodologías SOA no propietarias [Sva12].

### **3.4.2 SOMA (*Service-oriented Modeling and Architecture*) [AGA+08]**

SOMA es un método de desarrollo de software creado y desarrollado inicialmente en IBM para el diseño y construcción de soluciones basadas en el marco de referencia SOA. Este método define claves técnicas y proporciona prescriptivas tareas y orientación normativa detallada para el análisis, diseño, implementación, prueba y despliegue de servicios, componentes, flujos, información y políticas necesarias para diseñar con éxito y construir una solución SOA robusta y reutilizable en una empresa [AGA+08]. Es importante señalar que SOMA no es un método para desarrollo de LPS, sino es orientado al desarrollo de un solo sistema o producto, en el ámbito de sistemas orientados a servicios.

Muchas definiciones se han propuesto para los servicios [AGA+08]. Desde una perspectiva empresarial, un servicio es una capacidad alineada al negocio, bien definida, encapsulada, y reutilizable. Una operación de servicio es la parte primaria de un servicio y especifica las respectivas entradas, su propósito (función, actividad u obligaciones), y las salidas (artefactos, productos, resultados o entregables). Un servicio es completamente definido por una descripción del servicio, un documento publicado o artefacto que describe el objetivo general del servicio y sus entradas, propósito, salidas, alcance, responsabilidad, gobernanza, sostenibilidad (período de suministro, mantenimiento y reparación), y las cualidades de la prestación del



servicio [AGA+08]. Desde una perspectiva de las TIC, un servicio es un recurso de software detectable, invocable que tiene una descripción del servicio y la interfaz y es configurable través de directivas (políticas). La descripción del servicio se encuentra disponible para la búsqueda, enlace, e invocación por un consumidor de servicios. La implementación de la descripción del servicio es realizada a través de un proveedor de servicios que ofrece requisitos de calidad de servicio (QoS) para el consumidor de servicios [AGA+08].

### ***Vista General del Método SOMA***

Como un método del ciclo de vida de desarrollo software para el desarrollo de soluciones basadas en SOA, o cualquier solución utilizando principios orientados a servicios, SOMA define técnicas claves y describe los roles en un proyecto SOA y una Estructura de Descomposición del Trabajo (EDT) en inglés “Work Breakdown Structure (WBS)”. Esta estructura incluye tareas, los artefactos de entrada y salida para las tareas y una guía prescriptiva con los consejos necesarios para un análisis detallado, diseño, implementación y despliegue de los servicios, componentes y flujos necesarios para construir un ambiente de SOA robusto y reutilizable. El método SOMA incluye siete fases principales que se muestran en la Figura 15.

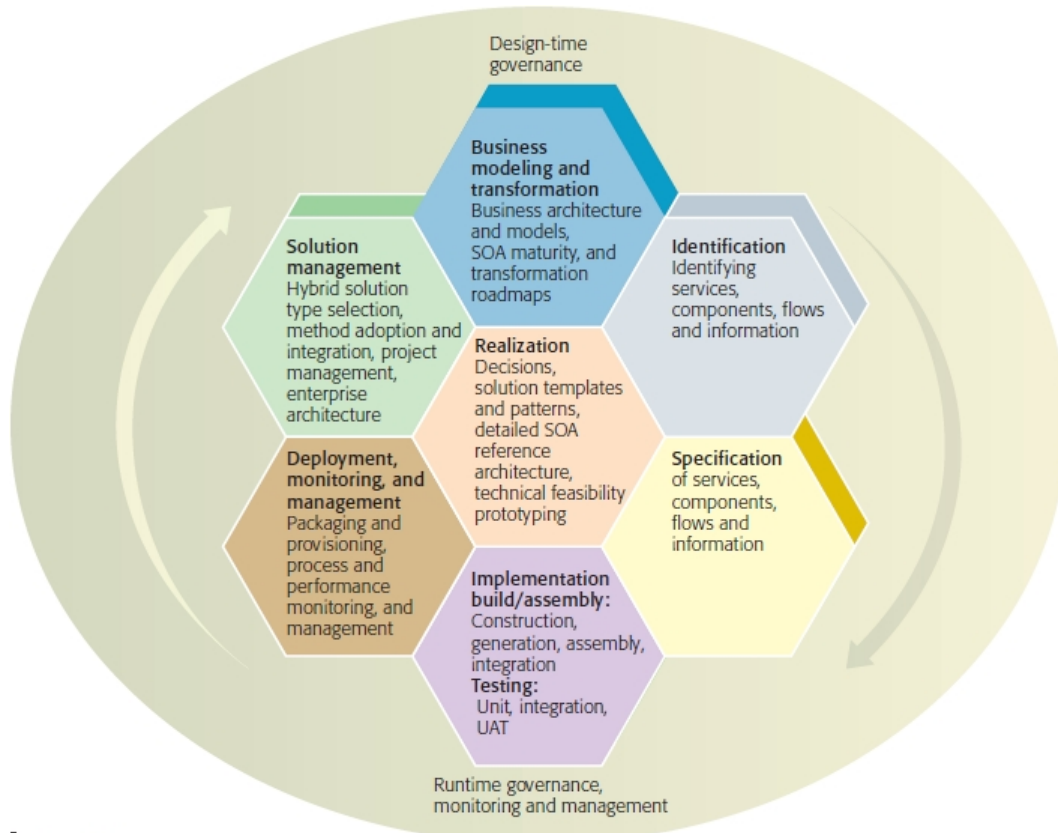
#### ***Fase 1: Fase de transformación y modelado del negocio***

En esta fase, el negocio es modelado, simulado y optimizado, y es identificada un área focal “ámbito” para la transformación que conducirá a una serie de proyectos posteriores utilizando el conjunto de fases que se muestran en la Figura 16 y que serán brevemente discutidos en las secciones siguientes. Esta fase no es estrictamente necesaria pero es altamente recomendable.

#### ***Fase 2: Fase de gestión de la solución***

Las soluciones SOA son híbridas en su naturaleza y suelen incluir varios tipos de solución. Esto es porque los servicios identificados y especificados durante las primeras fases en SOMA pueden ser realizados en las fases posteriores de SOMA por diferentes escenarios, tales como el desarrollo personalizado, integración y transformación de productos heredados, y la integración de un paquete aplicación. SOMA está diseñado para soportar la naturaleza híbrida de soluciones SOA y prescribe actividades específicas y orientaciones para implementar cada uno de los tipos anteriores de solución. En SOMA, el método contiene aspectos comunes a todos los tipos de solución SOA. El método contiene: tareas, artefactos, roles y orientación, específicos a cada uno de los tipos de solución que es definida y exteriorizada como

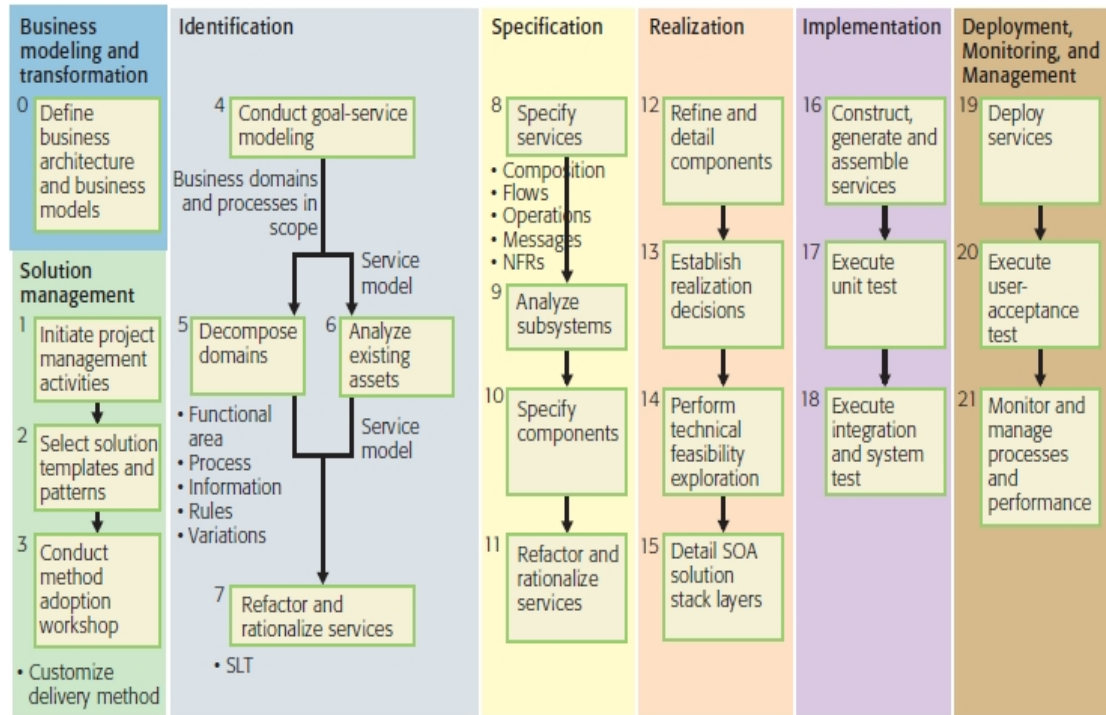
un plantilla del método denominado Plantilla de Solución “*solution template*”. Al principio de una contratación y, posteriormente, cuando se toman las decisiones de la realización de los servicios, típicamente se descubren y seleccionan los tipos de solución necesarios para construir una solución SOA para un cliente. Las actividades y tareas de la plantilla de solución seleccionada se insertan en puntos de extensión predefinidos en el método SOMA para proporcionar un proceso integral para la ejecución de la contratación.



**Figura 15.** Fases de SOMA, un modelo de desarrollo de software, tomado de [AGA+08]

### **Fase 3: Fase de identificación**

La fase de identificación se refiere a la identificación de los tres constructos fundamentales de SOA: servicios, componentes y flujos. La identificación en SOMA es el proceso de identificar los servicios candidatos y crear un portafolio de servicios de negocio alineados a servicios de Tecnología de Información (TI) que soportan colectivamente los procesos de negocio y objetivos de la organización. Esto es realizado a través del proceso de evaluación de la funcionalidad existente para ver si se puede colocar en el modelo de servicio y mediante la determinación de las capacidades de TI faltantes necesarias para apoyar la alineación de los procesos, objetivos y estrategia de negocios.



**Figura 16.** Flujo de alto nivel del ciclo de vida SOMA, tomado de [AGA+08]

#### **Fase 4: Fase de especificación**

En la fase de especificación de SOMA, la SOA es diseñada. El diseño de alto nivel así como partes importantes del diseño detallado de los componentes de servicio es completado en esta fase. Durante la fase de especificación, se aprovechan los activos existentes y se elaboran los servicios, flujos los componentes a partir de la fase de identificación de manera iterativa e incremental para ayudar a alcanzar la decisión de realización. El modelo de servicio es posteriormente elaborado en términos de las dependencias de servicio, flujos y composición, eventos, reglas y políticas, operaciones, mensajes, requisitos no funcionales y las decisiones de gestión del estado.

#### **Fase 5: Fase de realización**

Validamos las decisiones de realización mediante la exploración de la factibilidad técnica que pretende ejercer las decisiones arquitectónicas y factores de riesgo del proyecto mediante prototipos extensibles diseñados y desarrollados tempranamente.

#### **Fase 6: Fases de implementación y despliegue, supervisión y gestión**

En la fase de implementación, se construye, genera y se ensamblan los componentes funcionales y técnicos que realizan los servicios, y flujos. Se crean los necesarios mecanismos por el cual un componente existente puede participar en la realización de un servicio. En algunos casos, los activos existentes son refactorizados y refinados así que pueden ser utilizados para realizar un servicio. Se realizan las pruebas de unidad, integración y de sistema para los servicios, componentes y flujos.

SOMA utiliza una variedad de plantillas de solución que incluyen el soporte para el desarrollo a la medida, integración de aplicaciones empaquetadas, e integración y transformación de aplicaciones heredadas, el diseño y uso del ESB<sup>37</sup> y los servicios de negocios compuestos. La fase de implementación es extendida por las actividades y tareas de estas plantillas de solución para los diferentes tipos de solución en la solución global. Dependiendo de los tipos de solución, actividades complementarias y tareas de las plantillas solución correspondiente son ejecutadas para crear y actualizar los artefactos necesarios.

La fases de despliegue, monitoreo y gestión, se centran en el empaquetado, aprovisionamiento, y ejecución de las pruebas de aceptación de los usuarios y el despliegue de servicios en el entorno de producción. Así, como en la fase de implementación, la fase de despliegue es también extendida por actividades y tareas de las plantillas de solución para los diferentes tipos de solución en la solución global.

En conclusión, SOMA tiene una rica estructura de trabajo que comprende flujos de actividades, tareas y las correspondientes entradas y salidas. El alcance para la organización puede reducirse a la de un simple proyecto para la prueba de nuevos WS. Puede configurarse para ser adaptado (SOMA Agile), para soportar proyectos enfocados únicamente en la integración o comprobación de WS, o puede configurarse para un método más riguroso para soportar transformaciones de negocio o empresarial de gran envergadura (SOMA completo). SOMA puede ser utilizado para soportar la identificación y priorización de servicios de alta prioridad, o puede ser utilizado para desarrollar una aplicación o un sistema adoptando SOA.

### **3.4.3 Proceso SOMA Detallado [MR12]**

SOMA se refiere al ámbito más general de modelado de servicios necesarios para diseñar y crear soluciones SOA. SOMA abarca un ámbito amplio y pone en práctica el *análisis y diseño orientado a servicios* o “*Service Oriented Analysis and Design (SOAD)*” a través de la identificación, especificación y realización de los servicios, los componentes que realizan estos servicios (también conocido como "componentes de servicio"), y los flujos de procesos que pueden ser utilizados para la composición de estos servicios [MR12].

---

<sup>37</sup> Enterprise Service Bus

En la Tabla 26 se describen las fases, tareas y roles involucrados en un proceso SOMA.

**Tabla 26.** Tareas y Roles en las Fases de SOMA [MR12]

<b>Fases</b>	<b>Tareas</b>	<b>Roles</b>
Identificación	Descomposición del dominio	Analista del negocio
	Modelado de servicio – objetivo	Analista de procesos
	Análisis de activos existentes	Arquitecto de software
Especificación	Especificación de los servicios	Arquitecto de software
	Análisis del subsistema	Administrador de servicios
	Especificación de los componentes	Desarrollador
Realización	Decisiones de realización	Arquitecto de software

Uno de los principales resultados del método SOMA es un modelo de servicio [BLJ+08]. Se recomienda que un modelo de servicio constituya los siguientes artefactos acerca de los servicios:

- Portafolio de Servicios: lista de todos los servicios de la organización
- Jerarquía de servicios: categorización de los servicios en grupos de servicios
- Exposición de servicios: análisis y racionalización de cuales servicios deben ser expuestos y cuáles no
- Dependencias de servicio: representación de las dependencias de servicios en otros servicios
- Composición de servicio: como los servicios toman parte en las composiciones para realizar los flujos de procesos del negocio
- NFRs de servicio: requisitos no funcionales que un servicio debe cumplir
- Administración de estado: diversos tipos de estados que un servicio debe mantener y poner en práctica (este aspecto no fue realizado en [MR12])
- Decisiones de realización: decisiones arquitecturales, para cada servicio, y todo el mecanismo justificado para implementar el servicio.

A continuación se describe en detalle el proceso SOMA [MR12]:

### ***Proceso SOMA [MR12]***

#### **1. Identificación:**

*Input:* Documentación informal proporcionada por la organización que requiere de un sistema orientado a servicios.

##### *1.1 Analizar los procesos existentes:*

*Input:* Entrevistas con los stakeholders conocedores del dominio

*Identificar los procesos de negocio existentes en la organización*

*Output:* Descripción no formal de los procesos de negocios

##### *1.2 Realizar una descomposición de los procesos:*

*Input:* Descripción no formal de los procesos de negocios

*Analizar los procesos de negocios*

*Output: Procesos de negocio modelados con la notación BPMN  
(Descripción del Dominio)*

**1.3 Identificación de los servicios:**

*Input: Procesos de negocio modelados con la notación BPMN (Descripción del Dominio)*

*Identificación de los Servicios y Operaciones del proceso de negocios*

*Output: Tabla de servicios y operaciones*

**1.4 Relación entre Servicios y Entidades**

*Input: Procesos de negocio modelados con la notación BPMN (Descripción del Dominio)*

*Modelado de entidades correspondiente al proceso de negocio  
Identificación de las entidades que participan en cada servicio*

*Output: Modelo de entidades; Tabla de entidades correspondientes a los servicios*

**1.5 Relación entre Servicios y Entidades Propias**

*Input: Tabla de entidades correspondientes a los servicios*

*Identificación de entidades redundantes y entidades no asociados a algún servicio*

*Output: Tabla de entidades-servicios (consolidada)*

**1.6 Relación entre Servicios y Requerimientos Claves Técnicos**

*Input: Listado de requerimientos del negocio*

*Definición de los requerimientos técnicos que debe satisfacer el sistema*

*Descripción de requisitos técnicos funcionales (del negocio)*

*Descripción de requisitos técnicos no funcionales (de las agregaciones)*

*Output: Tabla de requerimientos claves técnicos funcionales y no funcionales*

**1.7 Priorización de Servicios**

*Input: Objetivos del negocio; Lista de servicios candidatos*

*Construir la matriz procesos de negocio – objetivos*

*Construir la tabla de objetivos priorizados*

*Identificar que servicios participan en el cumplimiento de los objetivos y su importancia dentro del cumplimiento del mismo*

*Construir la matriz objetivos-servicios*

*Output: Lista de Servicios Priorizados; Tabla de servicios-operaciones; Modelo de entidades*

*Output: Lista de Servicios Priorizados*

**2. Especificación:**

*Input: Lista de Servicios Priorizados; Tabla de Servicios-Operaciones; Modelo de Entidades.*

**2.1 Portafolio de servicios**

*Input: Lista de Servicios (portafolio inicial)*

*Lista de servicios clasificados según los siguientes criterios:*

*Priorización, Estado, Propietario, Origen*

*Construir el portafolio de servicios ordenados por prioridad*

*Lista de especificaciones de servicio (para los RF)*

*Se realizará una lista de todas las operaciones suministradas por la especificación*

*Identificación de los servicios que satisfacen una regla del negocio (RNF)*

*Se realizará una lista de todas las operaciones por regla de negocio*

*Output: Tabla de especificaciones de servicios (proceso de negocio); Tabla de especificaciones de servicios (regla de negocio); Portafolio de servicios actualizado*

### 2.2 Dependencias entre Servicios

*Input: Tablas de especificaciones de servicios*

*Identificación de los tipos de dependencia entre servicios (clasificación de dependencias)*

*Dependencias entre los servicios*

*Output: Tabla de dependencias entre servicios*

### 2.3 Categorización de Servicios

*Input: Portafolio de servicios actualizado*

*Categorización por zonas: presentación, integración, proveedores, infraestructura.*

*Output: Tabla de categorización de servicios por zona; gráfica que relaciona los servicios atómicos, compuestos y los procesos*

### 2.4 Requerimientos No Funcionales (QoS)

*Input: DQM-HIS*

*Especificación de los requerimientos técnicos no funcionales*

*Output: Listado de los requerimientos técnicos que deben satisfacer los servicios*

*Output: Portafolio de Servicios Actualizado*

## 3. Realización:

*Input: Portafolio de Servicios Actualizado*

### 3.1 Componentes de Implementación

*Input: Lista priorizada de los servicios*

*Especificación de los componentes que deben satisfacer las capas de los servicios de presentación, procesos, datos, infraestructura*

*Output: Componentes de la Arquitectura (Inicial)*

### 3.2 ATAM (Architecture Tradeoff Analysis Method)

*Input: DQM-HIS*

*Especificación de las decisiones arquitectónicas*

*Construir el Árbol de Atributos de Calidad*

*Análisis de Escenarios*

*Output: Especificación de los escenarios;*

*Output: Componentes de la arquitectura (Completada)*

Nótese que en el trabajo [MR12] las actividades que aparecen en la Fase de Realización, se realizan en la Fase de Especificación. La Fase de Realización la utilizan para decidir sobre que herramientas o componentes funcionales o no funcionales (tecnologías) son utilizadas para implementar las capas. Por ejemplo, para la capa de procesos, los autores sugieren el uso de la herramienta *Intalio Designer* e *Intalio Server*, para la orquestación de los servicios para dar cumplimiento a los procesos de negocio.

# Referencias

---

- [AAM03] M. Anastasopoulos, C. Atkinson, and D. Muthig, A Concrete Method for Developing and Applying Product Line Architectures, in *Objects, Components, Architectures, Services, and Applications for a Networked World*, Springer Berlin Heidelberg, pp. 294-312, 2003.
- [ABK+13] S. Apel, D. Batory, C. Kästner, and G. Saake, *Feature-Oriented Software Product Lines*, Springer, 2013.
- [ACL+13] M. Acher, P. Collet, P. Lahire, and R. B. France, Familiar: A Domain-Specific Language for Large Scale Management of Feature Models, *Science of Computer Programming*, vol. 78, no. 6, pp. 657-681, 2013.
- [AG08] S. Angelov and P. Grefen, An E-Contracting Reference Architecture, *Journal of Systems and Software*, vol 81, no. 11, pp. 1816-1844, 2008.
- [AG11] M. Abu-Matar, and H. Gomaa, "Feature based variability for service oriented architectures," In *Software Architecture (WICSA)*, 2011 9th Working IEEE/IFIP Conference, pp. 302-309, IEEE, June 2011.
- [AG13] M. Abu-Matar and H. Gomaa, An Automated Framework for Variability Management of Service-Oriented Software Product Lines, in *proceedings of the IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*, pp. 260-267. San Francisco, California, USA, March 2013.
- [AGA+08] Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., Holley, K.: SOMA: A method for developing service-oriented solutions. *IBM Systems Journal* 47(3), 377–396, 2008
- [ALH+11] I. Achour, L. Labed, R. Helali, and H. B. Ghazela, "A service oriented product line architecture for E-Government," *The 2011 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE'11)*. 2011.
- [AMK+09] M. Asadi, B. Mohabbati, N. Kaviani, D. Gašević, M. Bošković, and M. Hatala, Model-driven development of families of Service-Oriented Architectures, In *Proceedings the 1th International Workshop on Feature-Oriented Software Development*, pp. 95-102, ACM, 2009.
- [Ars04] A. Arsanjani, *Service-oriented modeling and architecture How to identify, specify, and realize services for your SOA*. IBM Software Group, November 2004.
- [BCC+08] N. Boffoli, D. Caivano, D. Castelluccia, F. M. Maggi, and G. Visaggio, "Business process lines to develop service-oriented architectures through the software product lines paradigm," In *SPLC (2)* pp. 143-147, 2008.



- [BFG+04] J. Bayer, T. Forster, D. Ganesan, J. F. Girard, I. John, J. Knodel, R. Kolb, and D. Muthig, Definition of Reference Architectures Based on Existing Systems, Lifecycle and Process for Family Integration, Fraunhofer IESE-Report no. 2004, vol. 34.
- [BGF+11] M. H. ter Beek, S. Gnesi, A. Fantechi, and J. L. Fiadeiro, "Variability and rigour in service computing engineering," In Software Engineering Workshop, 2011 34th IEEE, pp. 122-127, June 2011.
- [BGH+99] K. S. Barber, T. Graser, J. Holt, and C. Silva, Representing Domain Reference Architectures by Extending the UML Metamodel, in Twelfth International Conference on Software Engineering and Knowledge Engineering, pp. 256, 1999.
- [BGP12] L. Baresi, S. Guinea, and L. Pasquale, Service-Oriented Dynamic Software Product Lines, *Computer*, vol. 45, no. 10, pp. 42-48, 2012.
- [BGM+11] M. Bošković, D. Gašević, B. Mohabbati, M. Asadi, M. Hatala, N. Kaviani, J. Rusk, and E. Bagheri, "Developing families of software services: a semantic web approach," *Journal of Research & Practice in Information Technology*, 43(3), 2011.
- [BKB02] Bass L., Klein M., Bachmann F. Attribute Driven Design Method (ADD), SEI, CMU, 2002.
- [BHR06] M. C. Bastarrica, N. Hitschfeld-Kahler, and P. O. Rossel, Product Line Architecture for a Family of Meshing Tools, in Reuse of Off-the-Shelf Components, Springer Berlin Heidelberg, pp. 403-406, 2006.
- [Bjø06] D. Bjørner, *Software Engineering 3: Domains, Requirements, and Software Design*, Texts in Theoretical Computer Science, Springer-Verlag Berlin Heidelberg, 2006.
- [CK10] S. Cohen, and R. Krut, Managing variation in services in a software product line context (No. CMU/SEI-2010-TN-007), Carnegie-Mellon Univ Pittsburgh PA Software Engineering Institute, 2010.
- [DF13] R. Dos Santos, and M. Fantinato, "The use of software product lines for business process management: A systematic literature review," *Information and Software Technology* 55(8), pp. 1355-1373, 2013.
- [DMR+05] N. Duro, F. Moreira, J. Rogado, J. Reis, and Nestor Peccia, Developing a Reference Architecture for the Ground Segment Software, IEEEAC paper #1134, Version 1.14, IEEE, 2005.
- [DN07] L. Dobrica and E. Niemelä, Modeling Variability in the Software Product Line Architecture of Distributed Middleware Services, VTT Technical Research Centre of Finland, Oulu, Finland, 2007.
- [DN08] L. Dobrica and E. Niemelä, An Approach to Reference Architecture Design for Different Domains of Embedded Systems, in *Software Engineering Research and Practice*, pp. 287-293, 2008.
- [DRH+14] N. C. Das, S. Ripon, O. Hossain, and M. S. Uddin, "Requirement analysis of product line based semantic web services," *Lecture Notes on Software Engineering*, 2(3), 210, 2014.
- [Dol06] P. Dolog, *Engineering Adaptive Web Applications*, Doctoral Dissertation, University of Hanover, Germany, 2006.

- [DRG07] D. Dhungana, R. Rabiser, and P. Grünbacher, Decision-Oriented Modeling of Product Line Architectures, in the Working IEEE/IFIP Conference Software Architecture (WICSA'07), pp. 22, 2007.
- [DSG+08] G. Deng, D. C. Schmidt, A. Gokhale, J. Gray, Y. Lin, and G. Lenz, Evolution in Model-Driven Software Product-Line Architecture, Designing Software-Intensive Systems: Methods and Principles, 2008.
- [EAK06] A. Erradi, S. Anand, and N. Kulkarni, SOAF: an architectural framework for service definition and realization. In: IEEE International Conference on Services Computing (SCC 2006), 18-22 September 2006, Chicago, Illinois, USA. IEEE Computer Society, 2006, 151-158.
- [Eix98] W. Eixelsberger, Recovery of a Reference Architecture, A Case Study, in proceedings of the third International Workshop on Software Architecture, ACM, pp. 33-36, 1998.
- [EMA13] A. Ezenwoke, S. Misra, and M. O. Adigun, "An approach for e-commerce on-demand service-oriented product line development," Acta Polytechnica Hungarica, 10(2), pp. 69-87, 2013.
- [Erl05] T. Erl, Service-Oriented Architecture: Concepts, Technology and Design. Prentice Hall PTR, 2005.
- [Est16] Esteller L., Víctor A. Proceso de modelado de la variabilidad en el análisis del dominio que contempla requisitos funcionales y no funcionales para la construcción de una arquitectura de referencia. Tesis Doctoral. Ciencias de la Computación, Universidad Central de Venezuela., Marzo 2016.
- [GA11] M. Galster and P. Avgeriou, Empirically-Grounded Reference Architectures, A Proposal, in proceedings of the joint ACM SIGSOFT conference--QoSA and ACM SIGSOFT symposium--ISARCS on QoSA and architecting critical systems—ISARCS, ACM, pp. 153-158, 2011.
- [Gal10] M. Galster, "Describing variability in service-oriented software product lines," In Proceedings of the Fourth European Conference on Software Architecture, pp. 344-350, ACM, August 2010.
- [GAT13] M. Galster, P. Avgeriou, and D. Tofan, "Constraints for the design of variability-intensive service-oriented reference architectures—An industrial case study," Information and Software Technology, 55(2), pp. 428-441, 2013.
- [GB08] S. Günther and T. Berger, Service-Oriented Product Lines: Towards a Development Process and Feature Management Model for Web Services, in proceedings of the 12th International Software Product Line Conference (SPLC'08), pp. 131-136, Limerick, Ireland, September 2008.
- [GE11] M. Galster, and A. Eberlein, "Identifying potential core assets in service-based systems to support the transition to service-oriented product lines," In Engineering of Computer Based Systems (ECBS), 2011 18th IEEE International Conference and Workshops, pp. 179-186, IEEE, April 2011.
- [Gon12] H. J. González, "Integration of quality attributes in software product line development," Tesina de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información (ISMFSI), 2012.

- [GG05] A. Grosskurth and M. W. Godfrey, A Reference Architecture for Web Browsers, in proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05), pp. 661-664, 2005.
- [GL11] A. Goeb and K. Lochmann, A Software Quality Model for SOA, in proceedings of the 8th International Workshop on Software Quality (WoSQ'11), pp. 18-25, Szeged, Hungary, September 2011.
- [Gon12] H. J. González, Integration of Quality Attributes in Software Product Line Development, Master Tesis en Ingeniería del Software, Métodos Formales y Sistemas de Información (ISMFSI), 2012.
- [GON11] M. Guessi, L. B. R. de Oliveira, and E. Y. Nakagawa., Representation of Reference Architectures: A Systematic Review, Department of Computer Systems, University of São Paulo, SP, Brazil, 2011.
- [Har02] M. Harsu, FAST Product-Line Architecture Process, Tampere University of Technology, 2002.
- [HLM13] Herrera J., Losavio F., & Matteo A. (2013). Revisión Documental Sistemática de Enfoques y Técnicas para la Construcción de Arquitecturas en un contexto de Líneas de Productos de Software. Revista Venezolana de Ciencias de la Computación ReVeCom, Vol 1, No. 1, (pp. 17-25), Junio 2014.
- [HLM14] J. Herrera, F. Losavio, and A. Matteo, "RDS de enfoques y técnicas para la construcción de arquitecturas en un contexto de líneas de productos de software," Revista Venezolana de Ciencias de la Computación ReVeCom, Vol. 1, No. 1, pp. 17-25, Junio 2014.
- [HLM+14] Herrera J., Losavio F., Matteo A., & Ordaz O. (2013). Revisión Documental Sistemática en el Ámbito de la Ingeniería del Dominio para Líneas de Productos de Software Orientados a Servicios, ReVeCom, Vol 1, No. 2, (pp. 23-33), Diciembre 2014.
- [HLO16c] J. Herrera, F. Losavio, and O. Ordaz, Diseño de Arquitecturas de Referencia para Líneas de Productos de Software Orientadas a Servicios: Revisión Documental Sistemática, ReVeCom, Vol. 3, No. 2, pp. 13-25, Diciembre 2016.
- [Hoe04] A. van der Hoek, Design-Time Product Line Architectures for Any-Time Variability, Science of Computer Programming, vol. 53, no. 3, pp. 285-304, 2004.
- [IAO+09] A. Iborra, D. Alonso, F. J. Ortiz, J. Pastor, P. Sánchez, and B. Álvarez, Design of Service Robots - Experiences using Software Engineering, Robotics & Automation Magazine, IEEE, vol. 16, no. 1, pp. 24-33, 2009.
- [IJP09] P. Istoan, J. M. Jézéquel, and G. Perrouin, "Software product lines for creating service-oriented applications," Doctoral dissertation, Irisa Rennes Research Institute, 2009.
- [ISO11] ISO/IEC 25010, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuARE) - System and software quality models, ISO/IEC JTC1/SC7/WG6, 2011.
- [Jir12] W. Jirapanthong, Experience on Re-Engineering Applying with Software Product Line, International Journal of Computer Engineering Science (IJCES), vol. 2, no. 5, 2012.

- [KBB+09] B. Kitchenham, O. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, Systematic Literature Reviews in Software Engineering, a Systematic Literature Review, *Information and software technology*, vol. 51, no. 1, pp. 7-15, 2009.
- [KCH+90] Kang, K., Cohen, S., Hess, J., Nowak, W. and Peterson, S. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report, CMU/SEI-90-TR- 21, Software Engineering Institute (Carnegie Mellon), Pittsburgh, PA 15213, 1990.
- [Kit04] B. Kitchenham, Procedures for Performing Systematic Reviews, Keele University and National ICT Australia Ltd, Tech. Report TR/SE-0401 and NICTA TR 0400011T.1, 2004.
- [Kit07] B. Kitchenham, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Version 2.3, EBSE Technical Report, Software Engineering Group, School of Computer Science and Mathematics, Keele University, UK and Department of Computer Science, University of Durham, UK, 2007.
- [KKC+09] T. Kohlborn, A. Korthaus, T. Chan, and M. Rosemann, “Identification and analysis of business and software services - a consolidated approach,” *Services Computing*, IEEE Transactions, 2(1), pp. 50-64, 2009.
- [KKJ11] Y. G. Kim, S. Kee Lee, and S. B. Jang, Variability Management for Software Product-Line Architecture Development, *International Journal of Software Engineering and Knowledge Engineering*, vol. 21, no. 07, pp. 931-956, 2011.
- [Kru02] C. W. Krueger, Easing the Transition to Software Mass Customization, in *Software Product-Family Engineering*, Springer Berlin Heidelberg, 2002, pp. 282-293.
- [Kru95] P. Kruchten, “Architectural blueprints—The “4+1” view model of software architecture,” In proceedings of the Conference on TRI-Ada’95, Anaheim, CA, USA, November 1995.
- [Lal99] P. Lalanda, Style-Specific Techniques to Design Product-Line Architectures, Thomson-CSF Corporate Research Laboratory, Domaine de Corbeville, Orsay, France, 1999.
- [Lap11] A. Lapouchnian, Exploiting Requirements Variability for Software Customization and Adaptation, Doctoral Dissertation, University of Toronto, Canada, 2011.
- [LK06] B. List, and B. Korherr, “An evaluation of conceptual business process modelling languages,” In Proceedings of the 2006 ACM symposium on Applied computing, pp. 1532-1539, ACM, 2006.
- [LKL02] K. Lee, K. Kang, and J. Lee, “Concepts and guidelines of feature modeling for product line software engineering,” LNCS 2319, pp. 62-77, 2002.
- [LM13] F. Losavio, and A. Matteo, Reference Architecture Design Using Domain Quality View, *Journal of Software Engineering & Methodology*, vol. 3, no. 1, 2013.

- [LMN08] J. Lee, D. Muthig, and M. Naab, An Approach for Developing Service Oriented Product Lines, in proceedings of the 12th International Software Product Line Conference (SPLC'08), Limerick, Ireland, September 2008.
- [LMN10] J. Lee, D. Muthig, and M. Naab, A Feature-Oriented Approach for Developing Reusable Product Line Assets of Service-Based Systems, *Journal of Systems and Software*, vol. 83, no. 7, pp. 1123-1136, 2010.
- [LOE15] Losavio, F., Ordaz, O. y Esteller, V. Refactoring-based Design of Reference Architecture, *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS)*, Vol. 5, No. 1, pp. 32-48, 2015.
- [LOL+13] Losavio, F., Ordaz, O., Levy, N. & Baiotto, A. Graph Modeling of a Refactoring Process for Product Line Architecture Design, XXXIX CLEI 2013, (1) 2-13, 7-11 Oct. 2013, Puerto Azul, Naiguatá, Venezuela, 2013.
- [MAG+14] B. Mohabbati, M. Asadi, D. Gašević, and J. Lee, "Software product line engineering to develop variant-rich web services," In *Web Services Foundations*, pp. 535-562, Springer, New York, 2014.
- [MAL10] F. M. Medeiros, E. S. de Almeida, and S. R. de Lemos Meira, Designing a Set of Service-Oriented Systems as a Software Product Line, in proceedings of the Fourth Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), pp. 70-79, Salvador, Bahia, Brazil, September 2010.
- [Mat04] M. Matinlassi, "Comparison of software product line architecture design methods: COPA, FAST, FORM, KobrA and QADA," In *Proceedings of the 26th International Conference on Software Engineering*, pp. 127-136, IEEE Computer Society, 2004.
- [Med10] F. M. Medeiros, "SOPLE-DE: an approach to design service-oriented product line architectures," *Dissertação (Mestrado)*. Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife, 2010.
- [MGH+13] B. Mohabbati, D. Gasevic, M. Hatala, and H. Muller, A Quality Model and Evaluation Method for Service-Oriented Software Product Line and Configurable Business Processes, *ACM Transactions on Software Engineering Methodology*, September 2013.
- [Mit17] K. Mittal, Service Oriented Unified Process (SOUP). [Citado Abril 2017]. Disponible de: <http://www.kunalmittal.com/html/soup.html>.
- [MM13] M. Mohammadi, and M. Mukhtar, "A review of SOA modeling approaches for enterprise information systems," *Procedia Technology*, 11, pp. 794-800. 2013.
- [MMA+05] J. Muñoz, J. Muñoz, F. J. Alvarez, and M. Romo, *Aplicación de un Método para Generar una Arquitectura de Referencia que Integre Sistemas Heredados y Bases de Datos*, Universidad Autónoma de Aguascalientes, México, 2005.
- [MMR14] J. P. Miguel, D. Mauricio, and G. Rodríguez, "A review of software quality models for the evaluation of software products," *International Journal of Software Engineering & Applications (IJSEA)*, Vol. 5, No. 6, November 2014.
- [MR12] Martínez Aguilar, N. E., & Román Urbietta, Á. J. *Diseño de una arquitectura orientada a servicios para un establecimiento de salud de nivel de complejidad*

- I-3 (Doctoral disertación, Universidad Peruana de Ciencias Aplicadas-UPC), 2012.
- [MS10] Q. Munir and M. Shahid, Software Product Line: Survey of Tools, Master's thesis, Linköping University, Department of Computer and Information Science, 2010.
- [Mon09] S. Montagud, Un Método para la Evaluación de la Calidad de Líneas de Productos Software basado en SQuaRE, Master's Thesis, (In Spanish), Master en Ingeniería del Software Métodos Formales y Sistemas de Información. Universidad Politécnica de Valencia, Spain, 2009.
- [NBG11] M. Njima, M. ter Beek, and S. Gnesi, Product Line Architectures for SOA, in proceedings of the 2011 International Conference on Software Engineering Research and Practice (SERP'11), Las Vegas, Nevada, USA, July 2011.
- [NOB11a] E. Nakagawa, P. Oliveira, and M. Becker, Exploring the Use of Reference Architectures in the Development of Product Line Artifacts, in proceedings of the 15th International Software Product Line Conference, Vol 2, ACM, 2011.
- [NOB11b] E. Nakagawa, P. Oliveira, and M. Becker, Reference Architecture and Product Line Architecture: A Subtle but Critical Difference, Software Architecture. Springer Berlin Heidelberg, pp. 207-211, 2011.
- [NTG11] M. Njima, M. H. Ter Beek, and S. Gnesi, "Product line architectures for SOA," In Proceedings of the 11th Conference on Software Engineering Research and Practice, pp. 227-232. 2011.
- [NUM+11] D. N. Nguyen, K. Usbeck, W. M. Mongan, C. T. Cannon, R. N. Lass, J. Salvage, W. C. Regli, I. Mayk, and T. Urness, A Methodology for Developing an Agent Systems Reference Architecture, in Agent-Oriented Software Engineering XI, Springer Berlin, pp. 177-188, 2011.
- [OCC+12] W. N. Oizumi, A. C. Contieri, G. G. Correia, T. E. Colanzi, S. Ferrari, I. M. S. Gimenes, E. A. Oliveira, A. F. Garcia, and P. C. Masiero, On the Proactive Design of Product-Line Architectures with Aspects, An Exploratory Study, in proceedings of the 36th Computer Software and Applications Conference (COMPSAC 2012), pp. 273-278, 2012.
- [OFF+10] L. B. R. de Oliveira, K. R. Felizardo, D. Feitosa, and E. Y. Nakagawa, "Reference models and reference architectures based on service-oriented architecture: a systematic review," In Software Architecture, pp. 360-367, Springer Berlin Heidelberg, 2010.
- [OMG05] OMG Object Management Group. Unified Modelling Language Superstructure, version 2.0 (formal/05-07-04), August, [www.omg.org/spec/UML/2.0](http://www.omg.org/spec/UML/2.0), 2005.
- [OMG11] Object Management Group (OMG). Business Process Model and Notation (BPMN), Version 2.0. 2011.
- [Pan11] M. Panunzio, Definition, Realization and Evaluation of a Software Reference Architecture, Technical Report UBLCS-2011-07, Department of Computer Science, University of Bologna, Italy, 2011.
- [Par11] C. Parra, Towards Dynamic Software Product Lines: Unifying Design and Runtime Adaptations, Doctoral Dissertation, Université des Sciences et Technologie de Lille, Lille I, France, 2011.

- [PBL05] K. Pohl, G. Bockle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*, Springer, 2005.
- [PD05] J. Pollard and R. Duke, A Reference Architecture for Instructional Educational Software, in *ACM International Conference Proceeding Series*, pp. 43-52, 2005.
- [PGG+04] M. Pinzger, H. Gall, J. F. Girard, J. Knodel, C. Riva, W. Pasman, C. Broerse, and J. G. Wijnstra, *Architecture Recovery for Product Families*, in *Software Product-Family Engineering*, Springer Berlin Heidelberg, pp. 332-351, 2004.
- [PH06] M. P. Papazoglou and W.-J. van den Heuvel, Service-oriented design and development methodology, *International Journal of Web Engineering and Technology* 2(4), 412-442, 2006.
- [PJ15] C. Parra, and D. Joya, "SPLIT: an automated approach for enterprise product line adoption through SOA," *Journal of Internet Services and Information Security (JISIS)*, 5(1), pp. 29-52, 2015.
- [QB09] P. G. G. Queiroz and R. T. V. Braga, *Uma Abordagem de Desenvolvimento de Linha de Produtos com uma Arquitetura Orientada a Serviços*, Master's thesis, ICMC-Universidade de Sao Paulo, Sao Carlos, Sao Paulo, Brazil, November 2009.
- [RLA+10] H. G. B. Ribeiro, S. R. de Lemos Meira, E. S. de Almeida, D. Lucredio, A. Alvaro, V. Alves, and V. Garcia, "An assessment on technologies for implementing core assets in service-oriented product lines," In *Software Components, Architectures and Reuse (SBCARS)*, 2010 Fourth Brazilian Symposium, pp. 90-99, IEEE, 2010.
- [RRR11] A. Rashid, J.C. Royer, and A. Rummler, *Aspect-Oriented Model-Driven Software Product Lines. The AMPLE Way*, Cambridge University Press, Cambridge, 2011.
- [SBR+07] S. Segura, D. Benavides, A. Ruiz-Cortés, and P. Trinidad, *Open Source Tools for Software Product Line Development, Open Source and Product Lines*, 2007.
- [SG96] Shaw, M. y Garlan D. *Software Architecture: Perspectives of an emerging discipline*, Prentice-Hall, 1996.
- [SL09] D. Smith and G. Lewis, *Service-Oriented Architecture and Software Product Lines: Pre-Implementation Decisions*, in *proceedings of the 3rd International Workshop on Service Oriented Architectures and Product Lines (SOAPL'09)*, pp. 166-171, San Francisco, California, USA, September 2009.
- [SLC+10] V. Stricker, K. Lauenroth, P. Corte, F. Gittler, S. De Panfilis, and K. Pohl, *Creating a Reference Architecture for Service-Based Systems: A Pattern-Based Approach*, in *proceeding of Towards the Future Internet - Emerging Trends from European Research*, 2010.
- [SS12] H. Serajzadeh, and F. Shams, "An approach for discovering services for a service-oriented product line," *Global Journal on Technology*, Vol. 1, 2012.
- [Sva12] S. Svanidzaitė, "A comparison of SOA methodologies analysis & design phases," *Databases and Information Systems BalticDB&IS '2012*, 202, 2012.

- [TLY12] L. Tan, Y. Lin, and H. Ye. Quality-Oriented Software Product Line Architecture Design, *Journal of Software Engineering and Applications*, vol. 5, no. 7, pp. 472-476, 2012.
- [TP03] A. Trendowicz and T. Punter, Quality Modeling for Software Product Lines, in proceedings of the 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'03), 2003.
- [W3C04] World Wide Web Consortium, Web Services Architecture Requirements, W3C Working Group Note, 11 February, 2004.
- [YBM09] E. Y. Nakagawa, E. F. Barbosa, and J. C. Maldonado, Exploring Ontologies to Support the Establishment of Reference Architectures, An Example on Software Testing, in *European Conference on Software Architecture WICSA/ECSA 2009*, pp. 249-252, 2009.
- [ZLZ+07] C. Zhu, Y. Lee, W. Zhao, and J. Zhang, A Feature Oriented Approach to Mapping from Domain Requirements to Product Line Architecture, in *Software Engineering Research and Practice*, pp. 219-225, 2007.
- [ZSC+08] F. Zaupa, I. M. de Souza Gimenes, D. D. Cowan, P. S. Alencar, and C. J. P. de Lucena, "A service-oriented process to develop web applications," *Journal UCS*, 14(8), pp. 1368-1387. 2008.



***CAPÍTULO IV***

***PROCESO DE INGENIERÍA DEL***

***DOMINIO DE LÍNEAS DE PRODUCTOS***

---

## CAPÍTULO IV

# PROCESO DE INGENIERÍA DEL DOMINIO DE LÍNEAS DE PRODUCTOS

En este capítulo se tratará el problema general del desarrollo de Líneas de Productos, en el ciclo de vida de la Ingeniería del dominio de la ILPS. Como ya se señaló en la Introducción, Capítulo I, el objetivo principal de esta Tesis Doctoral es “Definir un Proceso de Ingeniería del Dominio para Líneas de Productos de Software Orientadas a Servicios, considerando aspectos de calidad del software”. En el presente capítulo se abordará el problema de la ID en LPS y en el próximo capítulo se tratará el problema de la ID en LPSOS. Se consideran aquí tres secciones: en la primera sección se describen las características de tres enfoques para el proceso ID, comparándolos y señalando sus similitudes y diferencias, con el fin de adaptarlos para definir nuestro proceso de ID que integre también la calidad del software; la segunda sección describe *PLScoP*, la primera fase del ciclo ID, para determinar el *Alcance de la LPS*, que es una etapa recomendada en ILPS para reducir el esfuerzo en las subsiguientes fases del desarrollo; en la tercera sección se especifica el proceso QuaDRA [HLO16b] para la ID en LPS, que integra lo enunciado en las dos secciones previas y concluye con una AR-LPS. Posteriormente, en el Capítulo V se tratará el proceso WSRA-SPL [HLO16d] que adapta QuaDRA y concluye con la AR-LPSOS.

### **4.1 Integración de enfoques para el desarrollo de la ID [HLO15] [HLO+16]**

A continuación se presentan tres enfoques para realizar la ID; dos contemplan el enfoque LPS [PBL05] [ISO15] y otro no está orientado al contexto LPS [Bjø06]. Además se describe como se realiza la integración de los tres enfoques.

#### ***4.1.1 Enfoques para la Ingeniería del Dominio***

La ILPS, según Pohl et al. [PBL05] trata los elementos comunes que integrarán los sistemas o productos de una familia LPS y sistemáticamente se encarga

de la variabilidad (es decir, las diferencias) entre esos sistemas. La similitud es una propiedad compartida por toda la familia LPS, que comparten siempre un mismo dominio, y su variabilidad define cómo las diferentes aplicaciones, sistemas o productos miembros de la familia LPS pueden variar [MP14].

Según Bjørner [Bjø06], antes de diseñar el software, se deben comprender sus requisitos y, antes de que se puedan prescribir los requisitos, se debe comprender el dominio de la aplicación que se va a construir; estos tres elementos lo denomina el Paradigma Trípico de la IS: - para comprender el dominio, se debe realizar su modelado desde las perspectivas de las facetas, bajo la óptica de *los grupos de interés* o "*stakeholders*". La ID o *Desarrollo del Dominio* propuesta por Bjørner reduciría el esfuerzo de trabajo relacionado con algunos aspectos vinculados con la Ingeniería de Requisitos (IR) clásica. Muchos autores abogan que la IR sería mucho más fácil de abordar (o de plano, podría "minimizarse") una vez que se haya estudiado exhaustivamente el dominio. Los artefactos más importantes de la ID de [Bjø06], son el *Modelo del Dominio* y la Teoría (formal) del Dominio asociada (que no será tratada aquí), sin embargo no se asocia con un contexto LPS, pero puede ser dirigido al desarrollo de una familia de sistemas; la Tabla 27 muestra la relación entre las fases de la ID definidas por Pohl et al. [PBL05], el estándar ISO/IEC 26550 [ISO15] y Bjørner [Bjø06]; esta correspondencia es un aporte de la presente investigación y fue reportado en [HLO15] [HLO+16].

**Tabla 27.** Correspondencia entre las Fases de la ID según Pohl et. al, ISO/IEC 26550 y Bjørner [HLO 15][HLO+16]

ISPL - Pohl et al. [PBL05]	Modelo de Referencia LPS - ISO/IEC 26550 [ISO15]	Ingeniería de Software (IS) - Bjørner [Bjø06]
<p><i>Proceso de Ingeniería del Dominio (ID) o "Domain Engineering Process (DE)":</i> Se consideran <b>5 sub-procesos</b> clave para anticipar los cambios para tomar en cuenta la evolución de la LPS, se desarrollan en el orden que establezca la organización que requiere la LPS:</p> <p>1. <i>Gestión del producto o "Product Management":</i> Se tratan aspectos económicos y estrategias de mercado; se ocupa de la <i>gestión del portafolio de productos o "product portfolio management"</i> de la LPS. Es un proceso de decisión dinámico, donde la lista de productos activos y/o proyectos es constantemente revisada y actualizada. La entrada son las <i>metas</i> de la organización y la salida es la <i>hoja de ruta o "roadmap"</i> que determina las características comunes y variables más importantes de los productos de la futura LPS, así como la planificación para la entrega de la LPS; las principales <b>actividades</b> de este sub-proceso son:</p> <ul style="list-style-type: none"> <li>o <i>Alcance del Portafolio de Productos o "Product Portfolio Scoping":</i> Define los productos a ser desarrollados por la LPS, y sus características clave; también se estudian productos existentes en el dominio;</li> <li>o <i>Alcance del Dominio o "Domain Scoping":</i> Define los límites del dominio de la LPS</li> <li>o <i>Alcance de Activos o "Assets Scoping":</i> Identifica componentes específicos a ser reutilizados o desarrollados para ser reutilizados;</li> </ul>	<p><i>Ciclo de vida de la Ingeniería del Dominio (ID) o "Domain Engineering(DE) lifecycle":</i> Se consideran <b>5 fases</b> en el orden establecidos por la organización, como en [PBL05]:</p> <p>1. <i>Alcance de la LP o "Product Line Scoping (PLScop)":</i> Define los límites de la LPS, identificando características comunes y variables más importantes. También considera productos existentes del dominio u organización. Se analizan aspectos económicos y se planifica la comercialización de la familia de productos de la LPS; PLScop es responsable de toda la gestión de la LPS e involucre <b>3 sub-fases</b>:</p> <ul style="list-style-type: none"> <li>- <i>Alcance del Portafolio de Productos o "Product Portfolio Scoping":</i> 1. Identificar productos a ser desarrollados, producidos, comercializados o vendidos por la LPS (<i>hoja de ruta del producto o "product roadmap"</i>); 2. Estudio de características comunes y variables de productos para ayudar al satisfacer las metas del negocio y enfrentar la evolución; 3. Planificación para la comercialización de productos. Es la entrada para la fase siguiente de Alcance del Dominio. Algunas técnicas propuestas son: métodos de resolución de problemas, survey, modelado de características, proceso de decisión en grupo.</li> <li>- <i>Alcance del Dominio o "Domain Scoping":</i> identificar y acotar las áreas funcionales que puedan proporcionar suficiente reutilización para justificar la creación de la LPS. Algunas técnicas propuestas son: ingeniería del conocimiento ("<i>Task Force Team</i>"), método de descomposición funcional ("<i>top-down</i>"), revisiones, métodos de decisión de grupos;</li> <li>- <i>Alcance de Activos o "Assets Scoping":</i> identificar los límites del núcleo básico de activos, proporcionar una primera visión de activos comunes y variables. Algunas técnicas propuestas son: ingeniería del conocimiento ("<i>Task Force Team</i>"), métodos de resolución de problemas, "<i>Data Mining</i>"; estimación del esfuerzo para productos existentes, estimación/evaluación de costo y riesgo, análisis de <i>retorno de inversión o "Return On Investment (ROI)"</i>, métodos de decisión de grupos;</li> </ul> <p>Salida de PLScop: <i>Propuesta de Activos o "asset proposal"</i>; incluye los principales activos (áreas funcionales y características comunes y variantes de alto nivel de todos los productos) que serán incluidos en la LPS con los resultados cuantificados de su estimación de costos y beneficios.</p>	<p>La IS contempla aquí <b>3 fases</b>:</p> <ul style="list-style-type: none"> <li>• <i>La Ingeniería del Dominio o Desarrollo del Dominio o "Domain Engineering or Domain Development (DE)";</i> es la más extensa y considera <b>6 etapas</b>; un <i>dominio</i> se define como cualquier área de interés y un <i>dominio de aplicación</i> es un dominio en el cual se aplica la computación, el <i>Modelo del Dominio o "Domain Model"</i> es la principal salida; describe un familia de sistemas, no solo una instancia del dominio;       <ol style="list-style-type: none"> <li>1. <i>Identificación de los Stakeholders o "Stakeholders Identification":</i> se identifican y clasifican como <i>grupos de interés</i>, todas las personas relevantes en los diferentes equipos de trabajo dentro de la organización; en [ISO15] estos se identifican en la fase DRE;</li> <li>2. <i>Adquisición del Dominio o "Domain Acquisition":</i> se captura, describe y clasifica información sobre el dominio a partir de los <i>puntos de vista o "viewpoints"</i> de los stakeholders; un punto de vista es conformado por las declaraciones realizadas por el grupo de interés sobre el dominio donde realiza sus actividades y son registradas como <i>Unidades de Descripción del Dominio (UDD) o "Domain Description Units (DDU)"</i> muchas UDD son la salida;</li> <li>3. <i>Análisis del Dominio o "Domain Analysis":</i> se analizan las DDU para estudiar los conflictos, inconsistencias y falta de completitud que pueden haber surgido en la <i>Adquisición del Dominio</i>; para este análisis se utilizan las <i>facetas o "facets"</i>;</li> <li>4. <i>Modelado del Dominio o "Domain Modelling":</i> se construye el <i>Modelo del Dominio o "Domain Model"</i>, es decir el significado de la descripción del dominio; deben capturarse los <i>intrínsecos o "intrinsic"</i>, faceta común a todas las facetas: <i>procesos de negocio, gestión &amp; organización, las reglas &amp; regulaciones, tecnologías de soporte y comportamiento humano</i>; la salida es el <i>Modelo del Dominio</i>; introducimos aquí la nueva <i>Faceta Intrínseca Descriptora de la Calidad o "Quality Intrinsic Facet Descriptor"</i> para especificar la calidad del dominio; un <i>Modelo de Calidad del Dominio o "Domain Quality Model (DQM)"</i> [LOE15], adaptado de [ISO11] se utilizará en esta investigación para especificar la calidad del producto;</li> <li>5. <i>Verificación/Validación del Dominio o "Domain Verification/Validation":</i> la especificación (formal, si se tiene) del Modelo del Dominio debe ser verificada con herramientas automáticas, o validada contra la opinión de stakeholders "<i>leaders</i>" de los diferentes equipos de trabajo</li> <li>6. <i>Teoría del Dominio o Domain Theory:</i> una teoría formal construida sobre una especificación formal del Modelo del Dominio.</li> </ol> </li> </ul>
<p>2. <i>Ingeniería de Requisitos del Dominio (IRD) o "Domain Requirements Engineering (DRE)":</i> Incluye todas las actividades clásicas de IR para elicitar y documentar requisitos comunes</p>	<p>2. <i>Ingeniería de Requisitos del Dominio (IRD) o "Domain Requirements Engineering (DRE)":</i> Debe conformarse a las características de alto nivel de la LPS dadas por PLScop. Sobre las bases de</p>	<ul style="list-style-type: none"> <li>• <i>Ingeniería de Requisitos (IR) o "Requirements Engineering (RE)":</i> Soporta las operaciones/procesos de la organización identificados en la fase anterior de ID; se especifican</li> </ul>

<p>y variables de la LPS. Entrada: la hoja de ruta. Salida: modelos de requisitos del dominio y el modelo de variabilidad de la LPS. Actividades:</p> <ul style="list-style-type: none"> <li>- <i>Elicitación de requisitos del dominio</i>: captura de requisitos</li> <li>- <i>Análisis de requisitos del dominio</i>: identifica y refina los requisitos elicitados; modelo de características; modelo de variabilidad.</li> <li>- <i>Especificación de requisitos</i>: documenta el modelo de variabilidad</li> <li>- <i>Validación</i> (“validation”) <i>de requisitos del dominio con los stakeholders</i></li> <li>- <i>Gestión de requisitos del dominio</i>: manejo de cambios de requisitos</li> </ul>	<p>estas características, se construyen requisitos comunes y variantes suficientes para guiar las fases siguientes del diseño de la AR, de realización y de pruebas.</p>	<p>los requisitos del software, es decir las <i>prescripciones de requisitos</i> o “<i>requirements prescriptions</i>”;</p> <p><b>Nota:</b> la IR se enfoca hacia un solo sistema, no sobre LPS, sin embargo [Bjø06] afirma que podría ser enfocada también para familias de sistemas, con lo cual se acercaría más a nuestro contexto LPS.</p>
<p>1. <i>Diseño del Dominio</i> o “<i>Domain Design (DD)</i>”:</p> <p>Se diseña la arquitectura del dominio o <i>Arquitectura de Referencia (AR)</i> o “<i>Reference Architecture (RA)</i>”. El arquitecto determina como se reflejan los requisitos en la AR, incluyendo el modelo de variabilidad. Se refiere a las actividades “tradicionales” de diseño arquitectónico: <i>Construir la Arquitectura, reducir la complejidad, modelado, simulación/prototipaje, validación, asegurar la calidad.</i></p> <p><b>Nota 1:</b> El aspecto crucial de la calidad es dejado completamente al arquitecto.</p> <p><b>Nota 2:</b> de los 5 sub-procesos de ID, solo se consideran aquí los 3 primeros; los sub-procesos de Realización y Pruebas pueden verse en [PBL05]</p>	<p>1. <i>Domain Design (DD)</i>:</p> <p>Trata de la especificación de una arquitectura para la LPS que permita la realización de las características comunes y variantes identificadas en las fases anteriores. El objetivo principal es producir la <i>Arquitectura de Referencia (AR)</i> o “<i>Reference Architecture (RA)</i>”, definiendo la estructura de la LPS. La actividad <i>Evaluación de la AR</i> o “<i>RA Evaluation</i>”, trata de técnicas para asegurar la calidad de la AR. Para la ILPS asegurar la calidad de la AR respecto a los requisitos de calidad de la LPS es crucial. Solo una AR que soporte suficientemente los requisitos de calidad será de carácter evolutivo. La actividad <i>Gestión del Diseño del Dominio</i> o “<i>Domain Design Management</i>” se refiere a garantizar la trazabilidad entre RF y RNF.</p> <p><b>Nota 1:</b> Las consideraciones sobre calidad son prácticamente ausentes en [PBL05] y en <i>Gestión del Diseño del Dominio</i> se dice que debe asegurarse la trazabilidad entre RF y RNF; claro está no de dice “como” hacerlo, siendo éste uno de los objetivos de la presente investigación.</p> <p><b>Nota 2:</b> en esta investigación solo se consideran las 3 primeras fases de ID; las fases de <i>Realización</i> y <i>Pruebas</i> pueden verse en [ISO15]</p>	<ul style="list-style-type: none"> <li>• <i>Diseño de Software</i> o “<i>Software Design</i>”</li> </ul> <p>Es el proceso de convertir todos los requisitos de software prescritos y su documentación en código ejecutable sobre una plataforma de hardware apropiada;</p> <ul style="list-style-type: none"> <li>• <i>Diseño de la arquitectura del Software</i> o “<i>Software Architecture Design</i>”:</li> </ul> <p>Se construye una primera especificación del software, indicando como manejar los requisitos en términos de componentes y sus interconexiones a un nivel alto de abstracción, sin detallar los componentes, produciendo la documentación apropiada</p> <p><b>Nota 1:</b> el Diseño de Software y de la Arquitectura se enfocan hacia un solo sistema, no hacia LPS.</p> <p><b>Nota 2:</b> solo las etapas 1, 2, 3 y 4 de la ID se consideran en esta investigación.</p>

#### 4.1.2 Integración del enfoque de Bjørner al Marco de Referencia ISO/IEC 26550 para LPS

En cuanto al enfoque de Pohl, la Tabla 27 muestra que está inmerso en el nuevo estándar y en [Käk10] se comprueba esta aserción; en consecuencia nos concentraremos solo en la integración de la fase ID o *Desarrollo del Dominio* de Bjørner [Bjø06] al estándar ISO/IEC 26550, en la fase de Alcance de la LPS en [ISO15]; para esta integración se contemplan dos aspectos básicos:

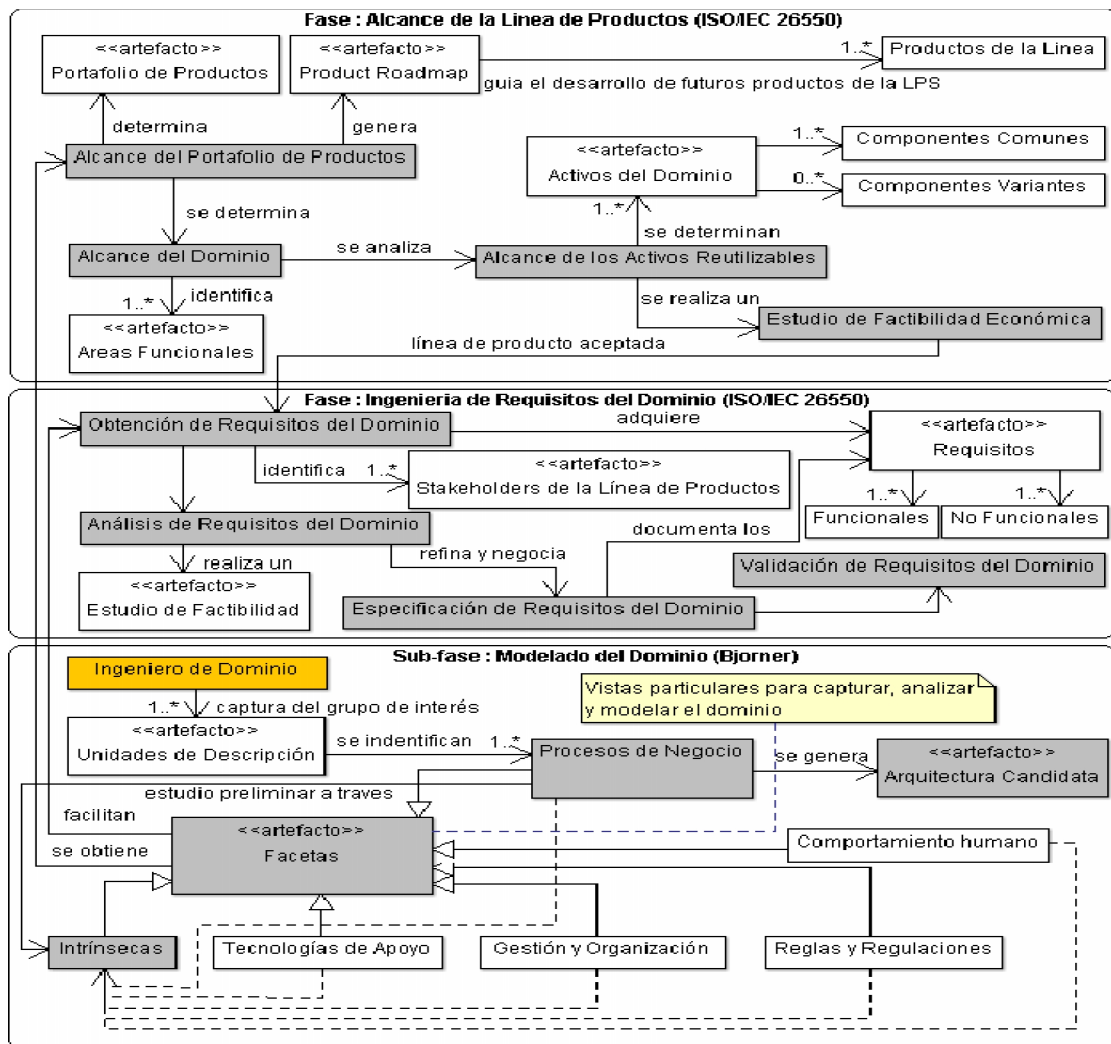
- Incluir en la fase 1. *Alcance de la LP* o “*Product Line Scoping (PLScop)*” de [ISO15], las 4 primeras etapas del *Desarrollo del Dominio* de [Bjø06]:

*Identificación de Stakeholders, Adquisición (captura) del Dominio, Análisis y Modelado del Dominio*, como actividades de la sub-fase *Alcance del Dominio* o “*Domain Scoping*”.

- Incluir explícitamente en la etapa *Modelado del Dominio* [Bjø06] (que será sub-fase de *Alcance del Dominio* en [ISO15]), el nuevo descriptor “*calidad*” en la *faceta intrínseca*, para la especificación de la calidad del dominio, respecto a la familia de productos de software en el contexto LPS, aspecto crucial en la fase de Alcance de la LPS y para los artefactos Modelo de Calidad del Dominio (MCD) y Arquitectura Candidata (AC). La inclusión del descriptor *calidad* en la *faceta intrínseca* constituye un aporte importante de nuestra investigación. Una *faceta* es definida en [Bjø06] como un conjunto finito de formas genéricas que describen un dominio desde diferentes perspectivas o *puntos de vista* de los grupos de interés, ver Tabla 28 [HLO15]. La noción de *faceta* no es nueva en la Ingeniería de Software [Kru95]; de acuerdo con [Bjø06], como cada *faceta* representa una vista del dominio desde diferentes perspectivas; la unión de estas vistas conforman la vista completa del dominio, denominado *Modelo del Dominio*; además, la *faceta especial intrínsecas* es una *faceta* que contiene descriptores o atributos (entidad, función, evento, comportamiento, y ahora también *calidad*) necesarios para describir todas las otras *facetas*, ver Tabla 29; con la inclusión del descriptor *calidad*, la *calidad del software* es ahora considerada para especificar todas las otras *facetas*. Para modelar una *faceta*, primero debemos considerarla, por ejemplo la *faceta Procesos del Negocio o Tecnologías Soportadas*; entonces debemos analizarla y formar conceptos, y luego se describe: (a) aproximadamente, (b) en términos de las entidades del dominio (terminología), (c) de forma narrativa y (d) posiblemente formalizando la *faceta* [Bjø06].
- Las técnicas en la fase de Desarrollo del Dominio [Bjø06] se organizarán para especificar sistemáticamente los procesos de negocio realizados por la organización y tendientes a ser automatizados, para integrarlas en las sub-fases

del nuevo estándar [ISO15]: *alcance del portafolio de productos, alcance del dominio, alcance de los activos reutilizables, y estudio de factibilidad económica*, para así determinar el alcance global de la LPS.

La fase *Desarrollo del Dominio* de [Bjø06] se integra al Modelo de Referencia del estándar ISO/IEC 26550, representada en el modelo conceptual UML [HLM+14] de la Figura 17, donde se muestran las fases, sub-fases y artefactos del Modelo de Referencia Integrado con [Bjø06].



**Figura 17.** Modelo Conceptual que integra los enfoques [ISO15] y [Bjø06] de la Tabla 27. Fuente: autor.

En la Figura 17, se destacan en color gris las sub-fases y artefactos más importantes del enfoque integrado, los alcances del portafolio de productos, de los activos reutilizables del dominio, la obtención de requisitos del dominio, el estudio de la factibilidad económica de la LPS, la faceta intrínseca. En color naranja se señala el rol del ingeniero del dominio quien es el actor más importante de todo el enfoque. La Tabla 28 muestra las facetas; la Tabla 29 muestra los descriptores de la faceta intrínsecas, incluyendo la calidad como un nuevo descriptor:

**Tabla 28.** Especificación de las facetas (vistas o perspectivas) del dominio [Bjø06]

Facetas para el Modelado del Dominio		Descripción	Notación/Técnica
1	Procesos del Negocio	Comprende la descripción de los procesos (comportamiento) en la cual las estrategias, tácticas y secuencias operacionales de transacciones de un negocio, una empresa, un componente de la infraestructura, son dadas por los stakeholders.	- Bocetos (Rough sketch) - Diagramas de flujo en BPMN [OMG11] - Redes de Petri - Storyboard
2	Intrínsecas	Son aquellos conceptos y fenómenos de un dominio (entidades, funciones, eventos y comportamientos) que son básicos para cualquiera de los diferentes puntos de vista del grupo de interés o stakeholders del dominio. Es una faceta base, en términos de los cuales se expresan todas las otras facetas, es decir guían la descripción de cada faceta. Nótese que el concepto de Intrínsecas corresponde a la noción de <i>Punto de Vista o "View Point"</i> definido en el estándar IEEE 1471-2000 [IEE00] para la descripción de la arquitectura de referencia de sistemas intensivos de software, que incluyen las LPS.	- Modelos conceptuales, por ejemplo en UML [OMG05]; los comportamientos pueden ser modelados en la notación BPMN [OMG11]
3	Tecnologías Soportadas	Tecnologías utilizadas por las partes interesadas para poder realizar (implementar) los procesos de negocios identificados. Estas tecnologías pueden ser implementadas de forma manual, mecánica, electromecánica, componentes electrónicos. Los estilos arquitectónicos se consideran parte de esta faceta.	- Diagrama de estados - Diagramas para descripción de las vistas arquitecturales, como diagramas de componentes en UML 2.X [OMG05]
4	Gestión y Organización	Consiste en la estructura gerencial, es decir, quien hace que/quien reporta a quien. Identifica quienes pueden realizar ciertas funciones.	- Organigrama de la estructura organizacional; jerarquía de las partes interesadas que intervienen en el dominio
5	Reglas y Regulaciones	Estándares utilizados en la organización relacionada o que afectan la toma de decisiones estratégicas, tácticas y operacionales. Son las reglas y regulaciones legales a	- Scripts (similar a los algoritmos o programas) que indican



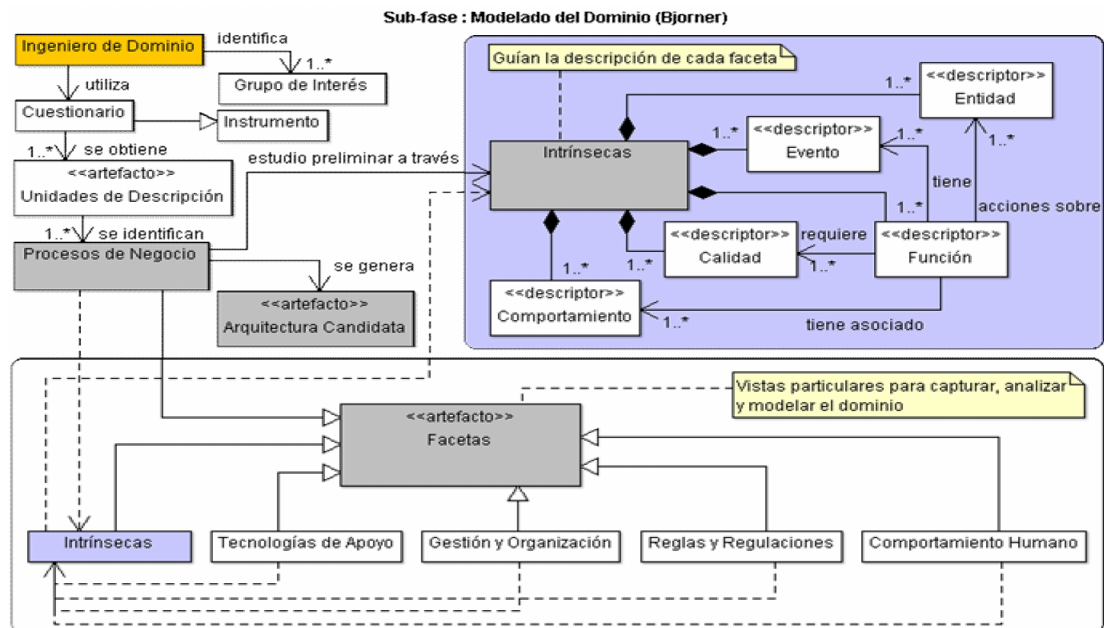
		nivel interinstitucional. El guion de acción (scripts), indica con claridad cuando una regla es aplicada y las acciones a seguir paso a paso.	paso a paso las acciones realizar. Pseudocódigo
6	Comportamiento Humano	Es una "forma" de representación de las entidades, ejecución de funciones, causando o reaccionando a eventos o participando en el comportamiento de procesos. Comprende cualquier escala de calidad de los seres humanos que llevan a cabo el trabajo asignado, es decir, como las personas realizan el proceso.	- Escenarios - Pseudocódigo (no se indica una notación en particular)

**Tabla 29.** Intrínsecas para describir todas las Facetas del Dominio con el Nuevo Descriptor Calidad

<b>Descriptores</b>	<b>Descripción</b>
Entidades	Representan los fenómenos y conceptos del dominio
Funciones	Operaciones (acciones) realizadas sobre las entidades
Eventos	Implica cambios en las entidades por invocaciones de funciones, es decir, acciones en el dominio
Comportamiento	Secuencias de acciones y eventos que afectan a las entidades del dominio
Calidad	Especificada por el modelo estándar de calidad ISO/IEC 25010 [ISO11]. Se asocia a la faceta Procesos de Negocio como objetivos de calidad (desde los RNF) requeridos por los RF (funciones, actividades o tareas), en la facetas Tecnología de Soporte como la calidad soportada por los estilos arquitectónicos, patrones o mecanismos involucrados, y Reglas&Regulaciones como la calidad requerida por las reglas de negocio del dominio [HLO15]. La calidad del producto debe ser especificada para garantizar que los productos de la LPS mantendrán los niveles de calidad industrial aceptables.

A continuación, en la Figura 18 se presenta el modelo conceptual basado en Bjørner [Bjø06] para la sub-fase *Modelado del Dominio* (Figura 17), que facilita la realización de la fase *Alcance de la LP* y de la siguiente fase *Ingeniería de Requisitos del Dominio*, durante el desarrollo de la ID para LPS. Este modelo conceptual permite identificar con claridad todos los elementos esenciales involucrados y sus relaciones: a) identificación de los grupos de interés, b) la captura (adquisición) del dominio, c) el análisis del dominio y d) el modelado del dominio. Las facetas son utilizadas para la captura, análisis y modelado del dominio.

En la Figura 18, los elementos resaltados en color gris (Facetas, Intrínsecas) son los componentes principales del proceso para la sub-fase: Modelado del Dominio. También se destaca en color azul el bloque de las componentes de la faceta intrínsecas (Entidad, Función, Evento, Comportamiento y Calidad). Sin dejar de lado al ingeniero de dominio en color naranja, que es el actor principal que realiza la mayor parte de las tareas o actividades en el proceso.



**Figura 18.** Modelo Conceptual para modelar el dominio en la sub-fase *Modelado del Dominio*. Fuente: Autor

El ingeniero de dominio para familiarizarse con el dominio objeto de estudio, debe realizar visitas in-situ y realizar entrevistas casuales con las partes interesadas o stakeholders del dominio, debe revisar la documentación asociada existente, crear un modelo rudimentario (boceto) del dominio y de allí construir un cuestionario para indagar sobre: entidades, funciones (acciones), eventos y comportamientos base (procesos de negocio), luego el cuestionario (instrumento) le es presentado a cada uno de los grupos de interés (partes interesadas en el dominio) [Jai06] [Bjø08] [Bjø09] [Bjø10]. La captura del conocimiento del dominio debe ser abordado a través de los descriptores de la faceta Intrínsecas, ver Tabla 29.

Los procesos de negocio son considerados como los comportamientos iniciales (base) del dominio; según [Bjø06], un proceso de negocio es también una faceta. En nuestro caso consideraremos los procesos de negocio sensibles a ser automatizados. Por regla general, la captura del conocimiento del dominio se inicia tomando como referente los procesos de negocio identificados a través de las visitas casuales a las partes interesadas del dominio, que conforman los grupos de interés en el dominio; a partir de los procesos de negocios identificados se realiza un estudio

preliminar utilizando la faceta Intrínsecas para comprenderlos y especificarlos; con ellos se van identificando las entidades, funciones, eventos y variantes en el comportamiento base identificado inicialmente.

Es importante destacar que en Bjørner [Bjø06], los descriptores (Tabla 29) son interpretados de la siguiente manera: el descriptor comportamiento comprende el conjunto de secuencias de acciones (funciones) y eventos que afectan a las entidades del dominio; sin embargo para esta investigación el descriptor función es mas relevante, y se interpreta de esta manera: las funciones tienen eventos, comportamientos y calidad asociadas que afectan a las entidades del dominio.

#### **4.2 Proceso PLScop: Product Line Scoping Process [HLO16a], [HLO+16]**

El *Alcance de la LP* es un activo central [NC12] [DeB99] [Sch00] de la LPS, luego se justifica su desarrollo. El proceso *PL Scoping (PLScop)* de [ISO15] será adaptado incluyendo las 4 primeras etapas del proceso de *Desarrollo del Dominio* de Bjørner en la etapa de *Alcance del Dominio* o "*Domain Scoping*". Nuestra adaptación, que consideramos un aporte importante de esta investigación, se denomina *Proceso de Alcance de la LP* o "*PLScop (PL Scoping Process)*", contempla técnicas precisas y sus artefactos. Nótese, que en general los estándares o marcos de referencia especifican el "*Qué hacer*", sin embargo, los detalles del "*Cómo hacerlo*" siempre tienen que ser definidos adecuadamente; incluso a pesar de que en [ISO15] se proporciona una lista de técnicas y métodos disponibles, cómo combinarlos o adaptarlos al contexto LPS para alcanzar una actividad en particular no es especificado.

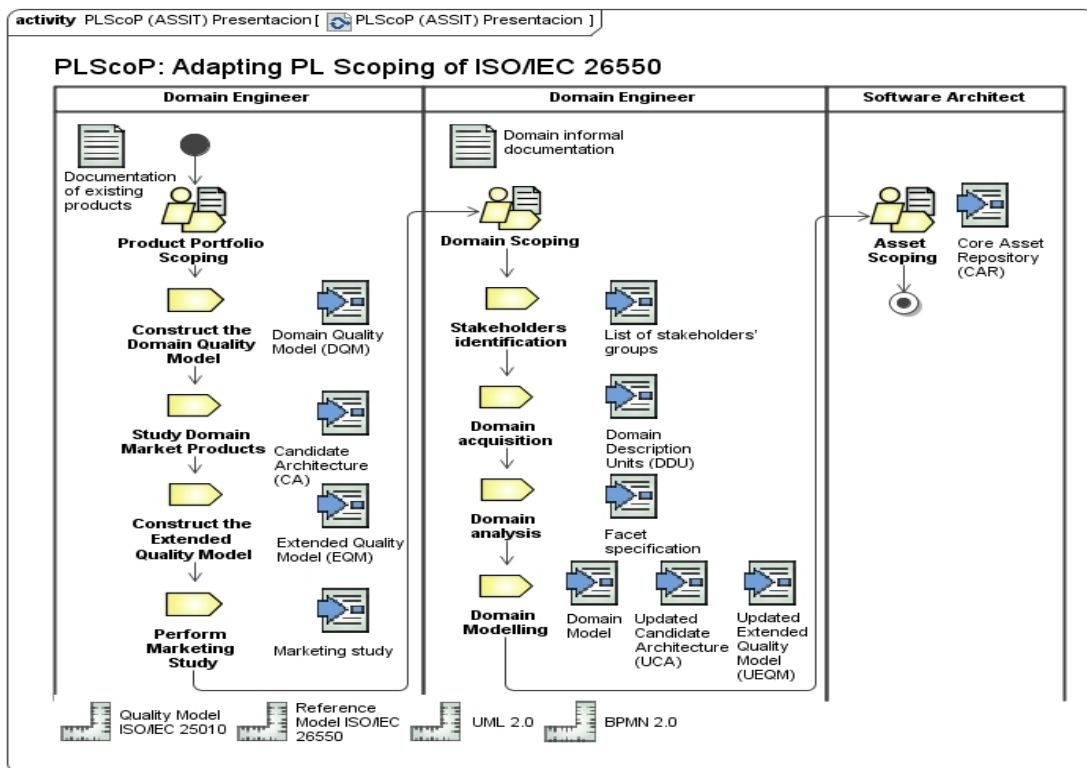
##### **A. PLScop: Aportes importantes**

- *Complementa el marco de referencia estándar ISO/IEC 26550 para LPS integrando el Desarrollo del Dominio de Bjørner [HLO15][HLO+16]*
- *Reduce el esfuerzo de las fases subsiguientes de IDR y DD: incluye un proceso extractivo bottom-up [LOE15][Est16] para el estudio de productos existentes en el Alcance del Portafolio de Productos, para producir automáticamente una arquitectura candidata (AC) inicial*

con un primer modelo de variabilidad, representada por un grafo conexo no dirigido

- Las propiedades de calidad son consideradas temprano en esta fase de alcance, como recomiendan [ISO15] [ABK+13] [SRK+12], pero no lo realizan, porque delegan estos aspectos a la fase DD [HLO+16]; las propiedades de calidad son especificadas por el estándar ISO/IEC 25010 [ISO11]
- La notación estándar BPMN será utilizada para especificar los procesos de negocio que representan el Modelo del Dominio incluyendo las propiedades de calidad, aspectos ausentes en esta notación

La adaptación de las directrices del *Alcance de la LP* en ISO/IEC 26550, es representado a continuación en la Figura 19 por el diagrama en SPEM<sup>38</sup> del proceso PLScoP, el cual será presentado también en lenguaje pseudo-formal, integrado al proceso QuaDRA en la siguiente sección.



**Figura 19.** Diagrama SPEM del Proceso PLScoP

<sup>38</sup> Software & Systems Process Engineering Metamodel Specification, <http://www.omg.org/spec/SPEM/2.0/>

## **B. PLScoP: Ventajas y Limitaciones**

- *La etapa del Alcance del Portafolio de Productos puede ser transformada en un proceso para realizar la evolución de la AR, es decir, gerencia de cambios, en las fases IRD y DD, por tener una AC inicial. El análisis de los productos existentes en el mercado o de la organización sin embargo no es una tarea fácil, y depende mucho de la documentación, lo que requiere un esfuerzo considerable para aplicar técnicas de reingeniería [LOE15]. La AR puede tener limitaciones si no se estudia un número suficientes de productos o si éstos no son muy representativos para el dominio.*
- *Se han combinado los enfoques proactivo “top-down” y extractivo “bottom-up” para especificar el dominio: top-down es considerado en el enfoque de Bjørner [Bjø06], que comienza con la descomposición del dominio en los procesos de negocio de la organización especificados por los descriptores intrínsecos, y es el enfoque generalmente utilizado en ILPS [PBL05] [ISO15]. El enfoque bottom-up se utiliza en la etapa del Alcance del Portafolio también en [PBL05] [ISO15], para tener un panorama amplio de los productos actuales y futuros de la LPS, mediante el estudio de los productos existentes del dominio. En este sentido, nuestra propuesta toma las ventajas de la combinación de [Bjø06] y [ISO15], reduciendo las debilidades generales del ciclo de vida de la ID.*
- *Las herramientas automáticas de soporte al proceso son necesarias y están aún en fase de diseño.*

### **4.3 QuaDRA: Quality Driven Reference Architecture [HLO16b]**

El objetivo de esta sección es presentar un proceso sistemático y repetible, denominado QuaDRA o “*Quality-oriented Design of Reference Architecture*” para la ID, de acuerdo al estándar ISO/IEC 26550 para ILPS. QuaDRA se considera un enfoque de desarrollo proactivo (top-down) ya que sigue la ILPS; además es orientado a la calidad; ésta se toma en cuenta desde la primera fase de Alcance de la LP, proporcionando así clara trazabilidad de los requisitos de calidad en todo el proceso, siendo éstos especificados como descriptores intrínsecos para todas las facetas del dominio, facilitando la evolución de la AR. QuaDRA combina dos enfoques de desarrollo de LPS en la fase de Alcance: el proactivo (top-down) en el Alcance del Dominio [Bjø06] y el extractivo (bottom-up), con técnicas de reingeniería, durante el Alcance del Portfollio de Productos, para así reducir el gran esfuerzo de desarrollo en las subsiguientes actividades de ID. Por lo tanto nuestro

enfoque contribuye globalmente a asegurar la calidad de AR y su carácter evolutivo y a reducir el esfuerzo global de desarrollo.

QuaDRA combina dos tendencias de desarrollo de la LPS: enfoque proactivo o top-down de Bjørner [Bjø06] en el Alcance del Dominio, con el enfoque extractivo o bottom-up en el Alcance del Portafolio de Productos [LOE15], que utiliza una técnica de reingeniería para llevar a cabo el estudio de los productos del mercado existentes en el dominio, suponiendo que los productos en el dominio estén disponibles. La combinación de los enfoques top-down y bottom-up reducirá el esfuerzo en las posteriores fases de la IRD y DD, donde se concentra el mayor esfuerzo de desarrollo en la ILPS.

#### ***4.3.1 El proceso de QuaDRA***

En lo que sigue, el proceso QuaDRA es presentado en lenguaje pseudo-formal, considerando sus tres fases:

- I. Alcance de la LP (PLScoP)
- II. Ingeniería de Requisitos del Dominio (DRE)
- III. Diseño del Dominio (DD)

##### ***Proceso QuaDRA***

##### ***inicio QuaDRA***

##### ***I. Fase Alcance de la LP (PLScoP);***

##### ***fase PLScoP;***

##### ***inicio PLScoP;***

***1. Alcance del Portafolio de Productos*** - *determina los requisitos para los productos futuros de la LPS:*

##### ***Inicio Alcance del Portafolio de Productos;***

***Entrada:*** documentación sobre productos o sistemas existentes;

***1.1 Estudio de los productos existentes del mercado en el dominio*** para inferir acerca de los productos de la LPS que pueden ser construidos:

- análisis de la similitud semántica de los componentes de los productos en el dominio es realizada sobre la documentación disponible;
- se construyen por refactorización las arquitecturas en UML 2.X de los productos existentes considerados;
- la *Tabla de Componentes y Conectores* o "*Components and Connectors Table (CCT)*" es construida, mostrando los

componentes y conectores de las arquitecturas de los productos considerados;

- una *Arquitectura Candidata (AC)* o "Candidate Architecture (CA)" es construida a partir de las arquitecturas de los productos existentes mediante la aplicación de un enfoque bottom-up extractivo automático [LOE15];

- esta técnica bottom-up extractiva es aplicada sobre una estructura de grafo conectado y no dirigido que representa las arquitecturas de los productos;

- la AC también es un grafo obtenido por la *unión* de los grafos que representan las configuraciones arquitecturales de los productos considerados; La AC es representada en UML 2.X;

Observación. Recordemos que: dados los grafos  $G_1 = (V_1, A_1)$  y  $G_2 = (V_2, A_2)$  donde  $V_i$  representa los nodos o vértices y  $A_i$  son los lados o arcos, su *unión*, denotada por  $G_1 \cup G_2$  se define como el grafo  $G'=(V', A')$  donde  $V'= V_1 \cup V_2$  y  $A'=A_1 \cup A_2$ . De igual forma, se define en forma iterativa la unión de más de dos grafos. Si los grafos  $G_1$  y  $G_2$  son conexos y  $V_1 \cap V_2 \neq \emptyset$ , entonces su unión es conexa. Recuerde que un grafo es conexo si para cada dos vértices existe un camino que los une: si los dos vértices están en el mismo grafo existe un camino por ser conexo. Si están en diferentes grafos se fabrica un camino pasando por la parte común. En nuestro contexto las arquitecturas de los productos de software son grafos conexos y por ser de un mismo dominio, deben tener componentes en común, esto es la intersección de las arquitecturas de productos de un mismo dominio tienen intersección no vacía;

- construir el *Modelo de Calidad del Dominio (MCD)* o "Domain Quality Model (DQM)", con respecto a los productos existentes, instanciando el modelo de calidad ISO/IEC 25010 para el dominio [ISO11];

- construir la *Tabla Modelo de Calidad Extendido (MCE)* o "Extended Quality Model (EQM) Table", donde la AC es documentada haciendo una lista de los componentes comunes y variantes de las propiedades de calidad requeridas/proporcionadas y sus restricciones; MCE es una vista diferente de la AC y también puede ser especificada como una ontología;

**Salida:** arquitecturas de productos existentes; CCT; AC; MCD; MCE;

**Notación:** arquitecturas de productos existentes tipo: grafo, diagrama UML; CCT tipo: tabla; AC tipo: grafo, diagrama UML u ontología; MCD tipo: tabla o gráfico; MCE tipo: tabla;

## 1.2 Elaborar un estudio de mercado para los productos de la LPS;

**Entrada:** información del mercado; AC; MCE;

- Realizar un estudio de mercado,

- los factores económicos también deberán ser analizados para la viabilidad de la LPS.

**Salida:** Estudio de Mercado, Portafolio de Productos

**Notación:** Estudio de Mercado tipo: documento; Portafolio de Productos tipo: texto;

### **1.3 Describir la similitud y la variabilidad para los futuros productos de la LPS.**

**Entrada:** AC; MCE;

- La similitud y la variabilidad de los futuros productos de la LPS serán especificados por la *Tabla MCE Actualizada (MCEA)* o "*Updated Extended Quality Model (UEQM) Table*" en el paso 2.4;

- Cada componente de las configuraciones de los futuros productos deberá ser descrita por un documento informal;

**Salida:** Documento de los Futuros Productos;

**Notación:** Documento de los Futuros Productos tipo: texto

#### **Fin Alcance del Portafolio de Productos;**

**Salida:** MCD; AC; MCE; Estudio de Mercado; Portafolio de Productos; Documento de Futuros Productos;

## **2. Alcance del Dominio - adaptado del Desarrollo del Dominio de Bjørner [Bj06];**

### **Inicio Alcance del Dominio;**

**Entrada:** MCD; MCE;

#### **Inicio Alcance del Dominio;**

##### **2.1 Identificación de las partes interesadas (Stakeholders):**

**Entrada:** visitas; entrevistas; talleres; cuestionarios (técnicas específicas para cada una de estas actividades deberán ser especificadas);

- Identificación de los grupos de partes interesadas con intereses similares en la organización que requiere la LPS;

**Salida:** listado de los grupos de partes interesadas

**Notación:** listado de los grupos de partes interesadas tipo: tabla;

##### **2.2 Adquisición del Dominio:**

**Entrada:** Listado de los grupos de partes interesadas;

- Captura y recopilación de información de cada grupo de interés en el dominio, en declaraciones para construir las *Unidades de Descripción del Dominio (UDD)* o "*Domain Description Units (DDU)*" para cada punto de vista de las partes interesadas;

**Salida:** UDD;

**Notación:** UDD tipo: tabla;

##### **2.3 Análisis del Dominio:**

**Entrada:** UDD;

- analizar las UDD, estudiar posibles inconsistencias; los Procesos de Negocio (PN) son extraídos de las UDD [Bj06], y son representados como comportamientos en la especificación de la faceta por descriptores intrínsecos desde un punto de vista de las



partes interesadas relevantes para el dominio; la especificación de las facetas tiene tres representaciones: tabla, diagrama UML para representar el modelo conceptual de la tabla, y diagramas BPMN para representar los procesos;

- la calidad del producto de software [ISO11] es incluida como un nuevo descriptor de la faceta intrínseca;

- se recomienda en [Bjø06] especificar primero la Faceta de Procesos de Negocios; las otras facetas consideradas relevantes para el dominio también son especificadas de la misma forma, en nuestro caso ellas serán Soporte Tecnológico y Reglas & Regulaciones;

**Salida:** Especificación de la faceta;

**Notación:** Especificación de la Faceta tipo: tabla, diagrama UML, diagrama BPMN;

#### **2.4 Modelado del Dominio;**

**Entrada:** Especificación de la faceta; AC; MCE; Documento de Futuros Productos;

- un Modelo del Dominio es obtenido a partir de la especificación de cada faceta; está es representada por diagramas BPMN:

- Integrar la especificación BPMN por cada punto de vista de las partes interesadas mediante la agregación de las actividades de los carriles "lanes" de cada contenedor "pool" del PN en BPMN; recuerde que una "Aggregation" BPMN agrupa actividades en un proceso;

- Adjuntar a la respectiva agregación, la información sobre las propiedades de calidad relacionadas con cada actividad, como una nueva actividad para ser realizada por la agregación para cumplir con la calidad respectiva;

- Obtener un nuevo diagrama de agregación BPMN (*BPMNAgg*) para cada punto de vista de las partes interesadas considerando las agregaciones;

- considera las agregaciones como posibles nuevos componentes o subcomponentes funcionales de la AC; conectándolos a los apropiados elementos en la configuración de la AC;

- por ejemplo, si el nuevo componente es el Sistema de Asistencia al Diagnóstico, éste debe ser resuelto para conectarlo al sistema de gestión EHR o tenerlo como un sistema separado en la Capa de Procesos, y en este caso la Interfaz de Usuario SIS deberá ser modificada en la Capa de Presentación; nótese que la sintaxis y la semántica de los elementos de la AC deberán mantenerse;

- las propiedades de calidad relacionadas con la agregación deberán ser incluidas como variantes de los nuevos componentes, proporcionando la calidad requerida; lo que significa establecer una nueva conexión en la AC entre estos componentes;

- estudiar el MCE, el Documento de Futuros Productos (ver Paso 1.3), y las Agregaciones *BPMNAgg*, para considerar si los nuevos componentes encontrados contribuyen a los futuros productos de la

LPS;

- Actualizar la AC (ACA) o "Updated CA (UCA)" (en todas sus representaciones);

- Actualizar el MCE (MCEA);

**Salida:** Modelo del Dominio: BPMNagg; ACA, MCEA;

**Notación:** Modelo del Dominio tipo: BPMN; MCEA tipo: tabla; ACA tipo: grafo, diagrama UML;

**fin Alcance del Dominio;**

**3. Alcance de los Activos** - para determinar la propuesta de Activos Centrales en la LPS

**Inicio Alcance de los Activos;**

**Entrada:** ACA; MCEA; MCD; BPMNagg;

- Información para la propuesta de activos centrales será extraída de la entrada del paso 3, para la construcción del *Repositorio Activos Centrales (RAC)*;

**fin Alcance de los Activos;**

**Salida:**

RAC;

**Notación:** RAC tipo: base de datos;

**fin PLScOP;**

## **II. Ingeniería de Requisitos del Dominio (IRD)**

**fase IRD**

**inicio IRD;**

**1. Análisis de Requisitos del Dominio:**

**Inicio Análisis de Requisitos del Dominio;**

**Entrada:** ACA; MCEA;

- ACA y MCEA contienen información precisa sobre los componentes comunes y variantes, incluyendo los aspectos de calidad;

**Salida:** ACA; MCEA;

**Fin Análisis de Requisitos del Dominio;**

**2. Especificación de Requisitos del Dominio:**

**Inicio Especificación de Requisitos del Dominio;**

**Entrada:** ACA; MCEA;

- ACA y MCEA contienen la especificación completa de los requisitos;

**Salida:** ACA, MCEA;

**Fin Especificación de Requisitos del Dominio;**

**3. Validación de Requisitos del Dominio:**

**Inicio Validación de Requisitos del Dominio**

**Entrada:** ACA; MCEA;

- consultar la ontología ACA, si está presente, para la comprobación de consistencia, de lo contrario otra técnica deberá ser utilizada;

- actualizar nuevamente MCEA y ACA;

- construir el grafo y diagrama UML 2.X correspondiente a ACA

**Salida:** ACA; MCEA;

**notación:** ACA tipo: ontología, grafo, diagrama UML; MCEA: tabla;  
**fin;**

#### **4. Gestión de Requisitos del Dominio:**

**Inicio Gestión de Requisitos del Dominio;**

**Entrada:** ACA; MCEA;

- actualizar la ACA para algún cambio en los requisitos;

- actualizar la MCEA; el paso 3 deberá ser aplicado de nuevo para validar los cambios;

**Salida:** ACA; MCEA;

**notación:** ACA tipo: ontología, grafo, diagrama UML; MCEA: tabla;

**fin Gestión de Requisitos del Dominio;**

**fin IRD;**

### **III. Diseño del Dominio (DD)**

**fase DD:**

**inicio DD:**

**Entrada:** ACA; MCEA;

**1. Diseño de la AR:** "... El soporte de la Variabilidad es crucial para el diseño de la AR, ya que todos los requisitos comunes y variables no pueden ser reflejados; una de las reglas básicas de la adición de la variabilidad en la AR es priorizando los requisitos. Los requisitos comunes a todas las aplicaciones deben ser satisfechas por la AR, mientras que los requisitos variables no pueden ser, a causa de la conflictividad entre algunos de ellos. La estructura y textura de la arquitectura de referencia deben ser compatibles con dicha variabilidad..." [ISO15].

**Inicio Diseño de la AR:**

**Entrada:** ACA; MCEA;

- la AR, representada por un grafo conexo, está conformada por los componentes comunes y puntos de variación [PBL05], que son componentes construidos agrupando variantes en la ACA que realizan tareas semánticamente similares; a estos <<vp>> se asignan nuevos nombres, diferentes a los de las variantes que los conforman, <<nombre del componente <<vp>>; los conectores de RA se construyen de tres formas: - tomando los conectores de CC en AC (que también están en ACA; - un conector existe entre dos <<vp>> si hay un conector entre una variante de un <<vp>> y una variante del otro <<vp>>; -

existe un conector entre un componente de CC y un <<vp>> si existe un conector entre el componente común y una variante del <<vp>>.

- La AR es un grafo conectado, no dirigido, especificado en UML 2.X; también puede ser especificada como una ontología;

**Salida:** Tabla MCEA; diagrama UML 2.X de AR

- **Documentación de la AR:**

**Inicio:**

**Entrada:** Tabla MCEA; Diagrama UML de la AR

Se especifican en una tabla (DOC-RA) las componentes comunes, los puntos de variación con sus variantes y sus conectores correspondientes; se colocan comentarios para cada componente y punto de variación <<vp>> indicando restricciones y opcionalidad. Esta tabla de documentación de AR constituye una entrada a la derivación de productos concretos a partir de la AR, en la fase de IA y sustituye al modelo de características comúnmente usado en ILPS clásica [referencia FODA].

**Salida:** Tabla DOC-AR;

**fin Diseño de la AR;**

**Salida:** AR; MCEA; DOC-RA;

**Notación** AR tipo: grafo, diagrama UML, ontología; MCEA: tabla;

**2. Evaluación de la AR:** es una técnica de control de calidad. "... Para ILPS, una evaluación de la AR es crucial, al menos para los requisitos de calidad relacionados con la LPS. Sólo un AR que soporte suficientemente los requisitos de calidad será evolutiva... "[ISO15].

**Inicio Evaluación de la AR;**

**Entrada:** AR

- la AR cumple con los requisitos de calidad por construcción; cada componente de la AR cumple con la calidad requerida; por otra parte, la AR satisface los requisitos de calidad globales del estilo arquitectural del dominio;

**Salida:** AR; documento con la evaluación

**fin Evaluación de la AR;**

**3. Gestión del Diseño del Dominio:** "... tiene por objeto la gestión del desarrollo y mantenimiento de la AR para los artefactos bajo cambios. Las principales actividades relacionadas con la gestión del diseño del dominio se pueden dividir en tres actividades: - Configuración y - gestión del cambio para el diseño del dominio y - gestión de la trazabilidad. Las relaciones entre los requisitos comunes y/o variables y la AR no son simples asignaciones "mapping" uno-a-uno, por lo tanto, la trazabilidad

entre los requisitos comunes/variables y los activos de la AR debe ser mantenida a un nivel comprensible... "[ISO15].

**Inicio *Gestión del Diseño del Dominio*;**

**Entrada:** AR

- Requisitos de trazabilidad son resueltos por construcción en la tabla MCEA;

**fin *Gestión del Diseño del Dominio*;**

**Salida:** AR; MCEA

**Notación** AR tipo: grafo, diagrama UML, ontología; MCEA: tabla

**fin DD;**

**fin *QuaDRA*;**

Todas las tablas y los diagramas de las arquitecturas mencionadas en el proceso QuaDRA pueden verse en el Cap. VI, que trata la aplicación del proceso al caso de estudio en el dominio de los Sistemas de Información Integrados de Salud.

A grandes rasgos, a partir del número de las actividades en QuaDRA mostradas en la IRD y DD, se puede apreciar que el esfuerzo requerido ha sido reducido considerablemente.

A continuación, en la Figura 20 se presenta el proceso QuaDRA expresado en la notación SPEM<sup>39</sup> [OMG08].

---

<sup>39</sup> Software & Systems Process Engineering Meta-Model Specification

### 4.3.2 Diagrama SPEM de QuaDRA

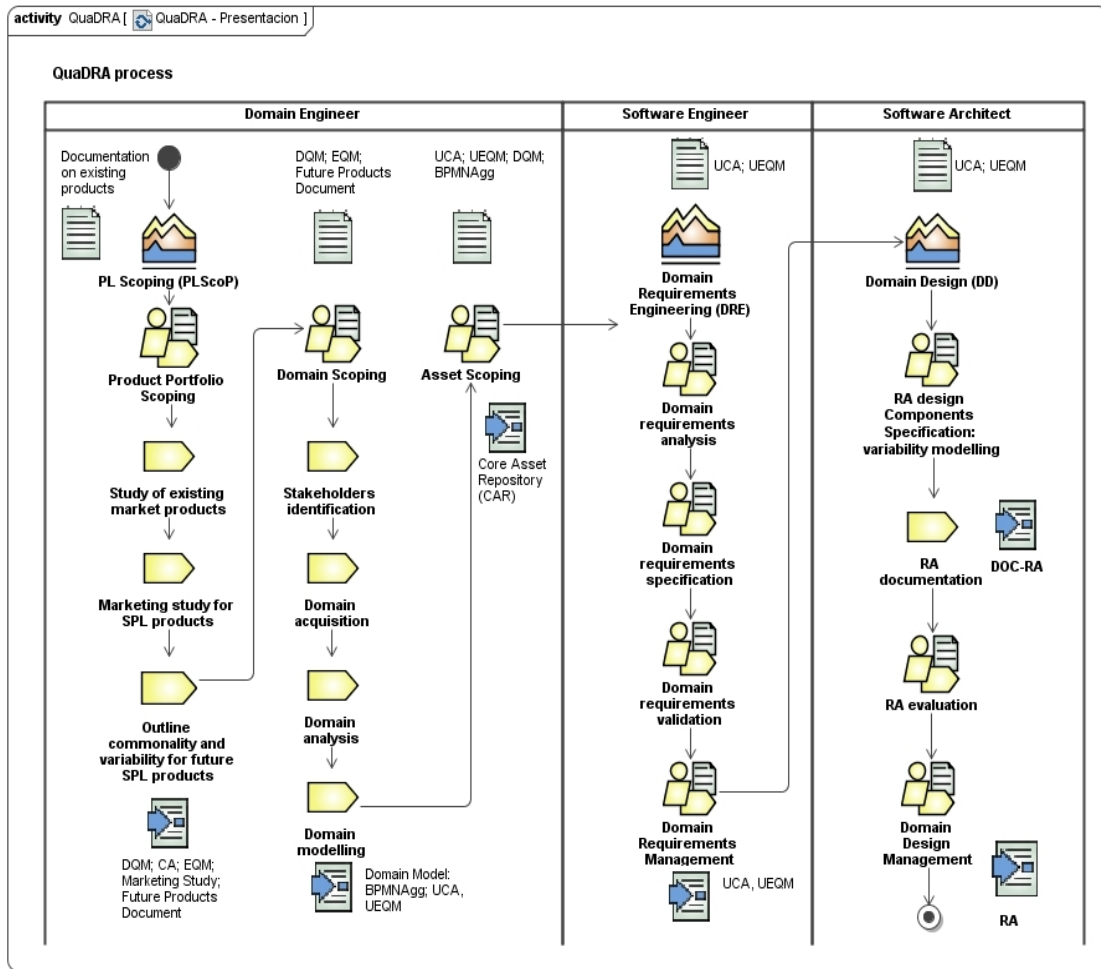


Figura 20. Diagrama SPEM del Proceso QuaDRA

### 4.3.3 Conclusión

La idea básica de nuestro enfoque, que no es nuevo en el diseño arquitectónico para los sistemas individuales, es considerar las principales funcionalidades en la AR, que requieren propiedades de calidad para ser funcionalmente adecuados, incluyendo componentes variantes proporcionando potenciales "implementaciones" para satisfacer esta calidad ofrecidas por los componentes tecnológicos disponibles del mercado. Esto es facilitado por el proceso bottom-up propuesto para la construcción de forma automática de la AC, por la unión de los grafos que representan las arquitecturas de los productos, respetando las

propiedades del estilo de arquitectural del dominio y el núcleo de funcionalidades básicas.

Para reducir la brecha encontrada en el estándar ISO/IEC 26550 sobre el tratamiento tardío de los aspectos de calidad, ésta es introducida temprano como un descriptor intrínseco en las especificaciones de la faceta en el Alcance del Dominio, y luego se hace corresponder dentro los procesos de negocio, que se aglomeran para conformar nuevos componentes arquitectónicos para los futuros productos de la LPS. El enorme esfuerzo requerido en las fases de IRD y DD es entonces reducido por la combinación de los enfoques bottom-up y top-down, siendo esta una contribución importante. Está claro que el ingeniero de dominio, arquitecto o experto juega un papel importante en las especificaciones de la tabla EQM y o UEQM; está claro que un conocimiento profundo sobre la tecnología disponible dará lugar a una AR más flexible y evolutiva. La recomendada representación ontológica de la AR se basa en las tablas EQM-UEQM, que recolectan “escenarios” sobre el conocimiento del dominio, respecto a que componente requiere y/o proporciona las propiedades de calidad; con esto se determina la trazabilidad entre componentes que resuelven funcionalidades y componentes que proporcionan soluciones o mecanismos para satisfacer la calidad requerida por esas funcionalidades, para que su comportamiento sea idóneo. La ontología de la AR será necesaria para comprobar la coherencia (consistencia) de las restricciones en los requisitos y para la futura derivación de los productos concretos de la LPS en el ciclo de vida de IA.

Por otra parte, hay que recordar que la calidad del software no es considerada temprano en [Käk10] [Kor15] [PBL05] [ISO15], siendo dejada en la etapa de Evaluación del Dominio, en la fase DD; además en [PBL05] es dejada completamente al criterio del ingeniero de dominio o a la experticia del arquitecto; nuestra contribución es haber incluido la calidad del software explícitamente en el Alcance del Dominio, aprovechándose de la especificación de la faceta intrínseca, que se aplica a todas las facetas del dominio; la realización de los aspectos de calidad juega un papel importante para el éxito de la LPS [ISO15]. Además, aún cuando los

componentes de software sean desarrollados respetando sus requisito de calidad, y el producto concreto de software es construido a partir de estos componentes “fiables”, la calidad del producto completo no puede ser garantizada; sin embargo, nosotros consideramos la calidad global del producto, en cierta medida, asegurando que las propiedades de calidad del dominio sean siempre satisfechas; pero esto sigue siendo un tema de investigación abierto.

En el próximo capítulo el proceso QuaDRA será adaptado para las LPS Orientadas a Servicios.



# Referencias

---

- [ABK+13] Apel, S., Batory, D., Kästner, C. & Saake, G. (2013). Feature-Oriented Software Product Lines. Springer.
- [Bjø06] Bjørner, D. (2006). Software Engineering 3 Domains, Requirements, and Software Design. Texts in Theoretical Computer Science. EATCS Series. Editors: W. Brauer G. Rozenberg A. Salomaa. Springer-Verlag Berlin Heidelberg.
- [Est16] Esteller L., Víctor A. Proceso de modelado de la variabilidad en el análisis del dominio que contempla requisitos funcionales y no funcionales para la construcción de una arquitectura de referencia. Tesis Doctoral. Ciencias de la Computación, Universidad Central de Venezuela, Marzo 2016.
- [HLO15] J. Herrera, F. Losavio, and O. Ordaz, Ingeniería del Dominio con el Estándar ISO/IEC 26550 para Líneas de Productos de Software Considerando la Faceta Calidad, Tercera Conferencia de la Sociedad Venezolana de Computación, CoNCISa, pp. 107-118, Universidad de Carabobo (UC), Valencia, Octubre 2015.
- [HLO16a] J. Herrera, F. Losavio, and O. Ordaz, Product Line Scoping for Healthcare Information Systems Using the ISO/IEC 26550 Reference Model, ReVeCom, Vol. 3, No. 1, pp. 38-50, Junio 2016.
- [HLO16b] J. Herrera, F. Losavio, and O. Ordaz, QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines based on ISO/IEC 26550, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 1, pp. 20-38, enero-junio 2016.
- [HLO16d] J. Herrera, F. Losavio, and O. Ordaz, Web-services reference architecture for software product lines: A quality-driven approach, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 2, pp. 7-21, julio-diciembre 2016.
- [HLO+16] J. Herrera, F. Losavio, and O. Ordaz, Product Lines Scoping using ISO/IEC 26550 Reference Model considering Software Quality, 2nd International Conference on Humanity and Social Science (ICHSS 2016), (pp. 194-200), DEStech Publications, Inc. USA.
- [ISO15] ISO/IEC. (2015). ISO/IEC NP 26550: Software and Systems Engineering – Reference Model for Software and Systems Product Lines. ISO/IEC JTC1/SC7 WG4.
- [ISO11] ISO/IEC. (2011). ISO/IEC 25010: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuARE) -- System and software quality models. ISO/IEC JTC1/SC7/WG6, Draft.

- [Käk10] Käkölä, T. (2010). Standards initiatives for software product line engineering and management within the international organization for standardization. In System Sciences (HICSS), 43rd Hawaii International Conference on (pp. 1-10). IEEE.
- [Kor15] Korff, A. (2015). Implementing ISO 26550: 2013 model-based. Complex Systems Design & Management, 346.
- [LOE15] Losavio, F., Ordaz, O. & Esteller, V. (2015). Refactoring-Based Design of Reference Architecture, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS) 5(1) pp. 32-48.
- [OMG05] Object Management Group (OMG). (2005). Unified Modelling Language Superstructure, version 2.0 (formal/05-07-04), August.
- [OMG08] OMG Object Management Group. (2008). Software & Systems Process Engineering Meta-Model Specification Version 2.0. MA: Object Management Group. Document number: formal/2008-04-01.
- [OMG11] Object Management Group (OMG). (2011). Business Process Model and Notation (BPMN), Version 2.0. Available in: <http://www.omg.org/spec/BPMN/2.0/>
- [PBL05] Pohl, K., Bockle, G. & Van Der Linden, F. (2005). Software Product Line Engineering: Foundations, Principles, and Techniques. Springer.
- [SRK+12] Siegmund, N., Rosenmuller, M., Kuhlemann, M., Kåstner, C., Apel, S. & Saake, G. (2012). SPL-Conqueror: Toward Optimization of Non-Functional Properties in SPL, Software Quality Journal.

***CAPÍTULO V***

***PROCESO DE INGENIERÍA DE LÍNEAS  
DE PRODUCTOS DE SOFTWARE  
ORIENTADAS A SERVICIOS***

---

---

## CAPÍTULO V

### PROCESO DE INGENIERÍA DE LÍNEAS DE PRODUCTOS DE SOFTWARE ORIENTADAS A SERVICIOS

En este capítulo se tratará el problema del desarrollo de Líneas de Productos de Software Orientadas a Servicios. Se abordarán las siguientes secciones: la primera sección describe las premisas o consideraciones previas que caracterizan el desarrollo de la ID para LPSOS; la segunda sección esboza el proceso WSRA-SPL, que es la adaptación del proceso QuaDRA para su adecuación al contexto de LPSOS, que constituye la segunda parte del proceso que concluye con la Arquitectura de Referencia Orientada a Servicios (AROS) o AROS-LPS, recordando que QuaDRA es un proceso general para el desarrollo de LPS; finalmente la tercera sección presenta en lenguaje pseudo-formal y el correspondiente diagrama SPEM, el proceso WSRA-SPL [HLO16d] que integra lo enunciado en las dos secciones previas.

Se ha visto que LPS y SOA son enfoques para el desarrollo de software utilizado en la práctica industrial favoreciendo la reutilización de activos (recursos) y capacidades existentes, en lugar de re-desarrollar nuevos sistemas. El objetivo principal del proceso WSRA-SPL consiste en explotar los beneficios de ambos enfoques e integrarlos en un único proceso de diseño arquitectónico. WSRA-SPL, al integrar QuaDRA, incorpora el aseguramiento de la calidad del software en etapas tempranas.

Mientras la LPS tiene como objetivo identificar la variabilidad y similitudes en una familia de productos desarrollada por un solo productor, SOA apunta a modelar y construir servicios independientes desarrollados por diversas empresas que, al combinarse, forman Sistemas Orientados a Servicios (SOS) complejos [CD15] [HHJ+07]. La gestión de la variabilidad de los servicios debe ser manejada [Mil01] [CD15] [CK10] [GZ14] para integrar LPS y SOA. Si bien los objetivos son similares en ambos enfoques, estos conducen a diferentes necesidades de variación; ambos

tipos de sistemas deben cumplir con las propiedades de calidad que pueden variar en la LPS o SOA.

Una *Arquitectura de Referencia de Servicios Web* o “*Web Services Reference Architecture (WSRA)*” para LPS está conformada por componentes arquitectónicos (servicios) y conectores (mecanismos de intercambio de mensajes entre WS). Una WSRA está construida bajo el modelo SOA soportando mantenibilidad (modificabilidad-extensibilidad, reutilización), fiabilidad, seguridad, escalabilidad y consistencia-corrección. La interoperabilidad es proporcionada por las interfaces estándar de los servicios. Sin embargo, hay que distinguir entre el modelo de referencia SOA de Arquitectura Empresarial, y la Arquitectura de Referencia LPS, donde el modelo de variabilidad [PBL05] debe estar presente para permitir la instanciación de la AR-LPS.

Del análisis de la literatura relacionada, se identificaron tres problemas principales para tratar la integración de enfoques de desarrollo de software LPS y SOA:

- 1) la representación de la AR-LPS con servicios
- 2) la satisfacción de las propiedades de calidad requeridas por las funcionalidades para comportarse correctamente
- 3) la gestión de la variabilidad.

El proceso WSRA-SPL propuesto pretende dar una solución a estos problemas.

## **5.1 Premisas**

### ***5.1.1 Programación robusta, Servicios Web y QoS***

Según el enfoque de ILPS dirigido por la calidad [LOE15], cada componente arquitectónico de alto nivel que resuelve una funcionalidad del sistema o RF, tiene asociados propiedades de calidad o RNF, para trabajar correctamente, es decir, que sea funcionalmente adecuado o idóneo [ISO11]. El componente implementado por un servicio debe satisfacer los objetivos de calidad de alto nivel requeridos. Para un

estilo de programación robusto [DD83] [Bro12], la calidad debe ser tomada en cuenta temprano, como una capacidad incorporada de la AR-LPS.

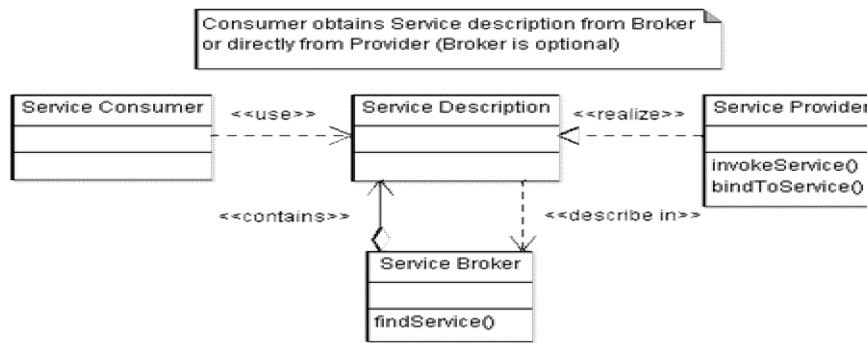
Por otra parte, en la programación de WS, la QoS (Calidad del Servicio, “Quality of Service”) representan RNF de bajo nivel sobre el comportamiento del servicio, relacionado sobre todo con el rendimiento y la seguridad, expresado como atributos de calidad o elementos medibles [ISO11] [ISO98]. Las QoS son especificados en la interfaz estándar del servicio, separada de la implementación, escrita en XML o en *Lenguaje de Descripción de Servicios Web* o “*Web Service Description Language (WSDL)*” [W3C04] para conseguir la interoperabilidad del servicio. La calidad de servicio (QoS) se define como un conjunto de cualidades relacionadas con los comportamientos colectivos de uno o más objetos. Los servicios pueden implementar componentes funcionales complejos (servicios compuestos), pero también deben satisfacer objetivos específicos de calidad, como cualquier otra funcionalidad, además de su QoS interno. La QoS no debe ser confundida con las características de calidad de alto nivel [ISO11] que el servicio puede “implementar”, como un componente de software cuya responsabilidad principal es cumplir cierta funcionalidad.

### **5.1.2 Modelo SOA y la Arquitectura de Servicios Web (WSA)**

Recordemos que una arquitectura de software es definida clásicamente por una configuración de elementos arquitectónicos, componentes, conectores y comportamiento, para alcanzar un conjunto deseado de propiedades [SG96]; varias vistas son propuestas para representar una configuración arquitectónica, incluyendo una vista lógica clásica, estática [Kru00]. Una *arquitectura basada en WS* o “*Web Services Architecture (WSA)*”, es expresada en [W3C04] por múltiples vistas o modelos, tales como mensajes, servicios, recursos y políticas. Sin embargo, todavía falta una vista lógica de la WSA considerando los servicios como componentes arquitecturales y sus conexiones (por mensajes, protocolos, etc.), y la composición podría facilitarse mediante esta vista lógica [Mil01]. El modelo de referencia SOA en capas que puede verse en la Figura 4 del Cap. II, muestra las capas de Componentes y

Servicios; la Figura 21 muestra detalles de la Capa de Servicios; los servicios de la Capa de Servicios están relacionados con los componentes arquitecturales, pero cómo esto es realizado y cómo están organizados no está muy detallado en la literatura.

SOA sigue un estilo arquitectónico basado en eventos [SG96], estableciendo la distribución y la comunicación a través de una red utilizando un modelo cliente-servidor; las aplicaciones son descompuestas en redes gestionadas de servicios que inter-operan, los cuales pueden ser implementados en una variedad de tecnologías, siendo los WS la más popular; utilizar SOA no significa tener una WSA “pura”, otros componentes pueden no ser WS [OMG08a].



**Figura 21.** Capa de Servicios SOA [Ars04]

## 5.2 Diseño de Arquitecturas de Servicios Web

La Tabla 30 presenta una correspondencia de tres principales enfoques de diseño de la AR que involucran el desarrollo de WS; el proceso WSRA-SPL completo será especificado en la sección 5.3 en lenguaje pseudo-formal, y un vistazo rápido del proceso en SPEM, se puede apreciar en la Figura 22.

SOMA toma como insumo el análisis y “componentización” del negocio. Un componente de negocio es descrito por los servicios de negocio que este ofrece. Un servicio de negocio está basado en la agrupación de las funcionalidades del negocio mediante flujos de trabajo, tareas, actividades, a menudo incluyendo reglas implícitas y explícitas.

**Tabla 30.** Correspondencia con SOMA [Ars04], Diseño Arquitectural: RUP&SOMA [MR12], y WSRA-SPL: QuaDRA adaptado a SOA [HLO16b], [HLO16d]

SOMA (SOA) [Ars04]		Diseño Arquitectural con RUP & SOMA (SOA) [MR12]	WSRA-SPL: QuaDRA (SPL) Adaptado a SOA [HLO16d]
<i>Fases</i>	<i>Tareas Principales</i>	<i>Actividades Principales</i>	<i>Fases y actividades principales</i>
Identificación (Servicios Candidatos o "Candidate Services (CS)", Componentes, Flujos)	Descomposición del dominio (Enfoque top-down)	- Analizar los procesos de negocios (BP) e identificar los <i>objetivos de negocios</i> o "Business Goals (BG)"	- Fase <i>PLScoP</i>  - Estudio de los productos del mercado; <i>nótese que este paso es realizado primero en la fase de Alcance de la LP</i> : - construcción de la Arquitectura Candidata (AC) inicial;  - identificación de los BG desde los componentes principales de la AC;  - asociar los BG con CS;  - exponer los futuros productos de la LPS como BG adicional; - asociar estos BG a CS; - construir el MCD;  - construir el MCE (Portafolio de Servicios);  - construir el Portafolio de Productos;
	Modelado Servicio-Objetivo (Enfoque middle-out)	- Asociar los servicios con los objetivos de negocio (BG) funcionales y no funcionales - Determinar los Servicios Candidatos (CS)	- Fase <i>PLScoP</i>  - <i>Modelado del Dominio</i> : - obtener el BP en BPMN, incluyendo los RNF enlazados (atados) a las actividades del BP; - identificar BG desde BP; BG funcionales son las principales funcionalidades del dominio (desde la AC); BG no funcionales son propiedades de calidad de prioridad del dominio (desde el MCD); - CS son asociados a éstos BG;
	Análisis de los sistemas existentes (Enfoque de bottom-up)	- Estudio de los sistemas existentes para extraer los CS - Relacionar los CS con entidades en el BP - Agrupar entidades de acuerdo con el servicio; cada servicio está relacionado a RF y RNF - Construir el Portafolio de Servicios con los CS, el cual es actualizado si nuevos servicios son encontrados	- Fase <i>PLScoP</i>  - <i>Modelad del Dominio (cont.)</i> : - agregar BP; - establecer nuevos BG desde las agregaciones BP;
Especificación (de Servicios, Componentes, Flujos)	Especificación de los servicios	Composición de los servicios, identificación de los RF, RNF y Calidad del Servicio (QoS) para los servicios en el Portafolio	- Fase <i>PLScoP</i>  - <i>Modelado del Dominio (cont.)</i> : <i>Composición de Servicios</i> : - realizar el diseño de los servicios funcionales



			<p>y no funcionales (verificar RF y RNF para cada composición de servicios desde el MCE);</p> <ul style="list-style-type: none"> <li>- asociar servicios compuestos a BG en las agregaciones;</li> <li>- comprobar los CS ya identificados de la AC contra los nuevos servicios compuestos, para evitar la redundancia;</li> <li>- determinar si éstos CS son servicios compuestos;</li> <li>- actualizar MCE (MCEA: Portafolio de Servicios);</li> <li>- actualizar la AC (ACA)</li> </ul>
	Análisis de Subsistemas	Asociar servicios a los componentes arquitectónicos	<p>- Fase DRE</p> <p>- <i>Asociar servicios a los componentes arquitectónicos:</i></p> <ul style="list-style-type: none"> <li>- servicios compuestos de las agregaciones y los futuros servicios de la LPS, son los principales componentes WSRA, conformando la ACA;</li> <li>- establecer conexiones entre los componentes de la ACA para especificar la trazabilidad desde la composición hasta los servicios simples mediante la comprobación de la MCEA para consistencia (coherencia);</li> </ul>
	Especificación de los componentes	Especificación de componentes arquitectónicos	<p>- Fase DD</p> <p>- <i>Modelado de la variabilidad:</i></p> <ul style="list-style-type: none"> <li>- determinar el modelo de variabilidad mediante la agrupación de los servicios compuestos que realizan tareas similares en puntos de variación;</li> <li>- Construir la WSRA;</li> </ul>
Realización (Decisiones)	Realización de las decisiones	Selección de específicos servicios disponibles en el mercado o desarrollo de servicios específicos	Esto no se considerará aquí; los mecanismos identificados durante el estudio de los productos existentes y otros requeridos en la ACA estarán disponibles en el repositorio de activos.

### 5.3 WSRA-SPL: Web Services Reference Architecture for SPL – Adaptación de QuaDRA a SOA

#### *Supuestos preliminares*

Se utilizan para mejorar la legibilidad de la especificación del proceso WSRA-SPL presentado en esta sección y que se detallan a continuación:

#### *5.3.1 Supuestos Preliminares para WSRA-SPL*

Utilizaremos la definición clásica de la *arquitectura de software* [SG96], componentes y conectores; los componentes son representados como servicios y sus conexiones serán la comunicación entre ellos, es decir, los mensajes SOAP intercambiados vía HTTP, HTTPS, SMTP, FTP, etc. a través de la red. El servicio, visto como un componente, es una vista abstracta, lógica de sistemas, módulos, programas, bases de datos, etc., que se define en términos de lo que hace, por lo general para llevar a cabo una operación a nivel empresarial o funcionalidad, especificada en los procesos de negocio. Por lo tanto, un modelo de negocio debe estar disponible.

La *composición de los servicios* a partir de los flujos de trabajo o “*workflows*” de los procesos de negocio (orquestración de los procesos de negocio), se supone correcto, siendo ésta una responsabilidad del middleware que se utilice, por ejemplo del bus de servicios, y/o herramientas de traducción automática como Intalio, etc. [MR12] en lenguajes ejecutables, como “*Business Process Execution Language (BPEL)*”. Lo que es importante es que los servicios actúan juntos en una composición como una sola aplicación [IBM09], que suponemos sea correcta.

En la AR-LPS definida en [HLO16b], los componentes derivados de las funcionalidades implícitas [ISO11] proporcionan soluciones o mecanismos de bajo nivel para satisfacer una propiedad de calidad de alto nivel requerida por un componente para resolver una funcionalidad del sistema o *componente funcional*, son diseñados como *servicios simples*; los componentes funcionales en cambio, serán diseñados como *servicios compuestos*; un *servicio compuesto* es entonces

conformado a su vez por servicios compuestos o por servicios simples. La trazabilidad entre los servicios compuestos y sus propiedades de calidad requeridas, representada por servicios simples, está dada por la *coreografía* de los servicios simples y compuestos involucrados en el diseño del componente funcional.

La composición de servicios, denominada también *coreografía u orquestación* [W3C04] es el proceso de construcción de un nuevo servicio a partir de los ya existentes, centrándose más en la funcionalidad del servicio (coreografía) o enfatizándose más en el intercambio de mensajes (orquestación). El umbral entre estas dos formas de composición no está muy claro en la literatura [W3C04]. En nuestro caso, queremos garantizar como el servicio se comportará con respecto a las propiedades funcionales (lo que entregará) y a las propiedades no funcionales (cómo y en qué condiciones se entregará) [Mil01]; por lo tanto, desde el punto de vista arquitectónico, no estamos interesados en el intercambio de mensajes, siendo esta una responsabilidad del software intermediario o “*middleware*” que se utilice como proveedor de servicios (por ejemplo un bus), sino en las propiedades de un componente y los servicios compuestos que la conforman y en sus relaciones (coreografía). Sin embargo, la calidad de servicio (QoS) ofrece la posibilidad de gestionar diferentes opciones de composición de servicios (gestión de la variabilidad) a un nivel más bajo de abstracción. Por ejemplo, en el caso de un servicio simple S, cuyo propósito principal (o funcionalidad implícita) es satisfacer la *Disponibilidad-Persistencia* de los datos, la elección entre la replicación o un mecanismo espejo (APIs), se podría hacer mediante la evaluación de los atributos de *Comportamiento temporal* o *Capacidad*, los cuales son especificados como parámetros de calidad de servicio en la interfaz WSDL de S. En el caso de la AR-LPS, S sería un punto de variación [PBL05] para resolver la *Disponibilidad*, conteniendo como variantes las APIs mencionadas.

### 5.3.2 Proceso WSRA-SPL - Especificación textual

#### **WSRA-SPL: Proceso de Diseño de Arquitectura de Referencia de Servicios Web para SPL**

El proceso de WSRA SPL es general y puede aplicarse a cualquier dominio, ver la sección 5.3.3; se presenta aquí una traducción al castellano del proceso WSRA-SPL original en inglés, la cual puede verse en [HLO16d].

#### **Proceso WSRA-SPL (QuaDRA adaptado)**

Las modificaciones con respecto a QuaDRA [HLO16b] han sido resaltadas en color gris en el proceso WSRA-LPS, presentado a continuación en una notación pseudo-formal; WSRA-LPS es representado gráficamente en SPEM en la Figura 22:

#### **WSRA-SPL**

**Inicio WSRA-SPL;**

**I. Fase de alcance de la LP (PLScoP)**

**fase PLScoP;**

**Inicio PLScoP;**

**1. Alcance del Portafolio de Productos** - determina los requisitos para los productos futuros de la LPS basados en WS;

**Inicio Alcance del Portafolio de Productos;**

**Entrada:** documentación sobre productos o sistemas existentes;

#### **1.1 Estudio de los productos existentes en el mercado;**

*1.1.1 SOMA - Análisis de los sistemas existentes en el dominio para inferir acerca de los productos basados en WS de la LPS que pueden ser construidos:*

- Análisis de la similitud semántica de los componentes de los productos; se hace sobre la documentación disponible;
- se construyen por refactorización las arquitecturas en UML 2.X de los productos existentes considerados mediante la aplicación de un enfoque bottom-up extractivo automático [LOE15]
- La tabla CCT o "Components and Connectors Table" [LOE15] es construida;
- Una Arquitectura Candidata (AC) es automáticamente construida a partir de las configuraciones arquitectónicas de productos representadas por un grafo conexo y no dirigido;
- La AC también es un grafo y es obtenido por la unión de los grafos que representan las arquitecturas de los productos;

**Observación:** Recordemos que: dados los grafos  $G_1 = (V_1, A_1)$  y  $G_2 = (V_2, A_2)$  donde  $V_i$  representa los nodos o vértices y  $A_i$  son los lados o arcos, su unión, denotada por  $G_1 \cup G_2$  se define como el grafo  $G' = (V', A')$  donde  $V' = V_1 \cup V_2$  y  $A' = A_1 \cup A_2$ . De

igual forma, se define en forma iterativa la unión de más de dos grafos. Si los grafos  $G_1$  y  $G_2$  son conexos y  $V_1 \cap V_2 \neq \emptyset$ , entonces su unión es conexa. Recuerde que un grafo es conexo si para cada dos vértices existe un camino que los une: si los dos vértices están en el mismo grafo existe un camino por ser conexo. Si están en diferentes grafos se fabrica un camino pasando por la parte común. En nuestro contexto las arquitecturas de los productos de software son grafos conexos y por ser de un mismo dominio, deben tener componentes en común, esto es la intersección de las arquitecturas de productos de un mismo dominio tienen intersección no vacía; La AC es también representada en UML 2.X y puede también ser especificada como una ontología;

#### 1.1.2 SOMA - Modelado de los Objetivos de Servicios:

- Los componentes de la AC representan los principales objetivos del negocio o "Business Goals (BG)" funcionales; estos componentes son asociados a los Candidatos a Servicios o "Candidate Services (CS)"; sin embargo, algunos BG no funcionales también podrían estar presentes en la AC, dependiendo de las características extraídas de los productos estudiados; estos también serán asociados como CS:

- Construir el Modelo de Calidad del dominio (MCD) para identificar los BG no funcionales, con respecto a los productos existentes, instanciando el modelo de calidad ISO/IEC 25010 para el dominio [ISO11];

- Identificación de los CS: construir la Tabla MCE (Modelo de Calidad Extendido), a partir de la AC y el MCD; La AC es documentada en el MCE haciendo una lista de los componentes comunes y variantes como CS, con los escenarios de las propiedades de calidad requeridas/proporcionadas y sus restricciones;

- El MCE contiene esta documentación, y es una vista diferente de la AC; conforma ahora el portafolio de servicios;

**Salida:** CCT; AC; MCD; MCE (Portafolio de Servicios);

**Notación:** AC tipo: grafo, diagrama UML u ontología; MCE tipo: tabla; MCD tipo: tabla o gráfico; CCT tipo: tabla;

#### 1.2 Elaborar un estudio de mercado para los productos de la LPSOS;

**Entrada:** información del mercado; AC; MCE;

- Realizar un estudio de mercado,  
- los factores económicos también deberán ser analizados para la viabilidad de la LPS.

**Salida:** Estudio de Mercado;

**Notación:** Estudio de Mercado tipo: documento;

#### 1.3 Describir la similitud y la variabilidad para los futuros productos de la LPSOS;

**Entrada:** AC; MCE;

- La similitud y la variabilidad de los futuros productos de la LPS serán especificados por la Tabla MCE actualizada (MCEA) en el paso 2.4;

- Cada componente de las configuraciones de los futuros productos será descrita por un documento informal;
- Salida:** Documento de los Futuros Productos;
- Notación:** Documento de los Futuros Productos tipo: texto;

**Salida:** MCD; AC; MCE (Portafolio de Servicios); Estudio de Mercado; Documento de Futuros Productos;

**Fin Alcance del Portafolio de Productos;**

## 2. Alcance del Dominio - adaptado de la ID de Bjørner [Bj06]

### **Inicio del Alcance del Dominio;**

**Entrada:** MCD; MCE; Documento de Futuros productos;

**Inicio Alcance del Dominio;**

### 2.1 Identificación de las partes interesadas (Stakeholders):

- Entrada:** visitas; entrevistas; talleres; cuestionarios (técnicas específicas para cada una de estas actividades deberán ser especificadas);
- Identificación de los grupos de partes interesadas con intereses similares en la organización que requiere la LPS;
- Salida:** listado de los grupos de las partes interesadas
- Notación:** listado de los grupos de las partes interesados tipo: tabla;

### 2.2 Adquisición del Dominio:

- Entrada:** Listado de los grupos de partes interesadas;
- Captura y recoge información de cada grupo de interés en el dominio, en declaraciones para construir las *Unidades de Descripción del Dominio (UDD)* o "*Domain Description Units (DDU)*" para cada punto de vista de las partes interesadas;
- Salida:** UDD;
- Notación:** UDD tipo: tabla;

### 2.3 Análisis del Dominio:

- Entrada:** UDD;
- Analizar las UDD, estudiar posibles inconsistencias; los Procesos de Negocio (PN) son extraídos de las UDD, y son representados como comportamientos en la especificación de la faceta por descriptores intrínsecos desde el punto de vista de las partes interesadas relevantes para el dominio; la especificación de las facetas tiene tres representaciones: tabla, diagrama UML para representar el modelo conceptual de la tabla, y diagramas BPMN para representar el PN. Los comportamientos son utilizados para identificar los BG; notase que a partir de la AC en el paso 1.1, la funcionalidad principal y eventualmente algunos BG no funcionales ya han sido identificados y pueden ser verificados a través del PN en BPMN; adicionales BG también pueden aparecer de esta especificación del dominio más completa;
  - La calidad del software [ISO11] es incluida como un nuevo descriptor de la faceta intrínseca.
  - Se recomienda en [Bj06] especificar primero la faceta PN. Otros facetas consideradas relevantes para el dominio son también especificadas en la misma forma, es decir, Soporte Tecnológico y Reglas & Regulaciones.

Salida: Especificación de la faceta;  
notación: Especificación de la Faceta tipo: tabla, diagrama  
UML, diagrama BPMN;

#### **2.4 Modelado del Dominio - SOMA - Descomposición del Dominio;**

##### *2.4.1 Especificación de Servicios SOMA - identificación de adicionales BG;*

**Entrada:** Especificación de la faceta; AC; MCE; Documento de Futuros Productos;

Un Modelo del Dominio parcial es obtenido a partir de la especificación de cada faceta; que está representada por el PN en BPMN:

- Integrar el PN en BPMN por cada punto de vista de las partes interesadas mediante la agregación de las actividades en los carriles "lanes" de cada contenedor "pool" del PN; recuerde que un BPMN "Aggregation" agrupa actividades del PN; estas agregaciones serán asignadas "mapped" a servicios compuestos, y representan los BG funcionales;
- Adjuntar a la respectiva agregación, la información sobre todas las propiedades de calidad para cada actividad, como una nueva actividad para ser realizada por la agregación para cumplir su calidad global "overall"; esta información corresponde a las propiedades de calidad de alto nivel de los servicios compuestos que representan los BG no funcionales; nótese que ellas no son calidad de servicio "QoS" que se consideran en la coreografía del servicio compuesto y que en la composición se supone que es correcta;
- Obtener un nuevo diagrama de agregación de BPMN (BPMNAgg) para cada punto de vista de las partes interesadas considerando las agregaciones; estos son los servicios compuestos con sus propiedades de calidad requeridas;

##### *2.4.2 SOMA - Análisis de Subsistemas - en nuestros términos: considerar agregaciones, actualizar AC, actualizar MCE, asignar agregaciones a servicios compuestos;*

- considera las agregaciones "aggregations" (servicios compuestos) como posibles nuevos componentes o subcomponentes funcionales de la AC; proporciona comunicación con los servicios apropiados en la configuración de la AC; por ejemplo, si el nuevo servicio compuesto es el Sistema de Asistencia al Diagnóstico, éste debe ser resuelto para conectarlo al servicio compuesto Sistema de Gestión EHR o tenerlo como un servicio compuesto separado, y en este caso la Interfaz de Usuario SIS deberá ser modificada; nótese que la sintaxis y la semántica de los elementos de la AC deberán mantenerse; las conexiones arquitecturales proporcionada/requerida entre dos servicios compuestos significa que deberán interactuar por intercambios de mensajes, como es el usual comportamiento de un SW.
- servicios simples para la resolución de las propiedades de calidad relacionadas con la agregación (composición de servicios) deberán ser incluidas como variantes de los

nuevos componentes que proporcionan la calidad requerida; lo que significa establecer una nueva conexión (comunicación) entre estos servicios simples en la AC;

- estudiar el MCE, el documento de Futuros Productos (ver Paso 1.3), y las agregaciones BPMNAgg, para considerar si los nuevos servicios compuestos encontrados contribuyen a los futuros productos de la LPS;
- Actualizar la AC (ACA) (en todas sus representaciones);
- Actualizar el MCE (MCEA - Actualizar el Portafolio de Servicios);

**Salida:** Modelo del Dominio: BPMNAgg; ACA, MCEA;

**Notación:** Modelo del Dominio tipo: BPMN; MCEA tipo: tabla; ACA tipo: grafo, diagrama UML y/o ontología;

**fin Alcance del Dominio;**

### **3. Alcance de los Activos - para determinar la propuesta de Activos Centrales en la LPS;**

**Entrada:** ACA; MCEA; MCD; BPMNAgg;

**Inicio Alcance de los Activos;**

- Información para la propuesta de activos centrales serán extraídos a partir de la entrada para la construcción de un Repositorio Activos Centrales (RAC);

**fin Alcance de los Activos;**

**Salida:** RAC;

**Notación:** RAC tipo: base de datos;

**fin PLScoP;**

## **II. Ingeniería de Requisitos del Dominio (IRD)**

**fase IRD;**

**inicio IRD;**

### **1. Análisis de Requisitos del Dominio (RD);**

**Inicio Análisis RD;**

**Entrada:** ACA; MCEA;

- ACA y MCEA contienen información precisa sobre los servicios simples o compuestos comunes y variantes, incluyendo los problemas de calidad;

**Salida:** ACA; MCEA;

**fin Análisis RD;**

### **2. Especificación de Requisitos del Dominio;**

**Inicio Especificación RD;**

**Entrada:** ACA; MCEA;

- ACA y MCEA contienen la especificación de los requisitos de la WSRA completa;

**Salida:** ACA, MCEA;

**Fin Especificación RD;**

### **3. Validación de Requisitos del Dominio:**

**Inicio Validación RD;**

**Entrada:** ACA; MCEA;

- consultar la ontología ACA, si está presente, para la comprobación de coherencia "consistencia", de lo contrario otra técnica deberá ser utilizada;
- actualizar el MCEA;

**Salida:** ACA; MCEA;



**Notación:** ACA tipo: ontología;  
**fin Validación RD;**

#### **4. Gestión de Requisitos del Dominio**

**inicia gestión RD;**

**Entrada:** ACA; MCEA;

- actualizar la ACA para algún cambio en los requisitos;
- actualizar la MCEA; el paso 3 deberá ser aplicado de nuevo para validar los cambios;

**Salida:** ACA; MCEA;

**fin gestión RD;**

**fin IRD;**

### **III. Diseño del Dominio (DD)**

**fase DD;**

**Inicio DD;**

**1. Diseño WSRA-SPL - SOMA - Especificación de los Componentes:**  
 modelado de la variabilidad en la LPS;

**Inicio Diseño WSRA-SPL;**

**Entrada:** ACA; MCEA;

- la WSRA está conformada por servicios compuestos y/o simples comunes y puntos de variación [PBL05], construidos mediante la agrupación de las variantes en la ACA (servicios compuestos y/o simples) que realizan tareas semánticamente similares; los conectores de RA se construyen de tres formas: - tomando los conectores de CC en AC (que también están en ACA; - un conector existe entre dos <<vp>> si hay un conector entre una variante de un <<vp>> y una variante del otro <<vp>>; - existe un conector entre un componente de CC y un <<vp>> si existe un conector entre el componente común y una variante del <<vp>>.
- Actualizar el MCEA con la información sobre los servicios.
- WSRA es un grafo conexo, no dirigido, especificado en UML 2.X y/o como una ontología;

**- Documentación de la WSRA:**

**Inicio:**

**Entrada:** Tabla MCEA; Diagrama UML de la WSRA

Se especifican en una tabla las componentes comunes, los puntos de variación <<vp>> con sus variantes y sus conectores correspondientes; se colocan comentarios para cada componente y punto de variación indicando restricciones y opcionalidad. Esta tabla de documentación de WSRAA constituye una entrada a la derivación de productos concretos a partir de la AR, en la fase de IA y sustituye al modelo de características comúnmente usado en ILPS clásica [referencia FODA [referencia];

**Salida:** Tabla DOC-WSRA;

**Salida:** WSRA; MCEA; DOC-WSRA

**Notación:** WSRA tipo: grafo, diagrama UML y/o ontología;

**Fin Diseño WSRA-SPL;**

**2. Evaluación de la WSRA:** es una técnica de control de calidad;  
**inicio Evaluación de la WSRA;**

**Entrada:** WSRA;

- WSRA cumple con los requisitos de calidad de alto nivel por construcción; cada componente WSRA-SPL (servicio simple o compuesto) cumple con la calidad requerida; por otra parte, la WSRA satisface los requisitos de calidad globales del estilo arquitectural del dominio;

**Salida:** documento de evaluación

**fin Evaluación de la WSRA;**

### **3. Gestión del Diseño del Dominio**

**inicio Gestión del Diseño del Dominio**

**Entrada:** WSRA;

- Requisitos de trazabilidad son resueltos por construcción;

**Salida:** WSRA;

**fin Gestión del Diseño del Dominio;**

**fin DD;**

**fin WSRA-SPL;**

Todas las tablas y los diagramas de las arquitecturas mencionadas en el proceso WSRA-SPL pueden verse en el Capítulo VI, que trata la aplicación del proceso al caso de estudio en el dominio de los Sistemas de Información Integrados de Salud. A continuación, en la Figura 22 se presenta el proceso WSRA-SPL expresado en la notación SPEM [OMG08b].

#### **5.3.3 Diagrama SPEM de WSRA-SPL**

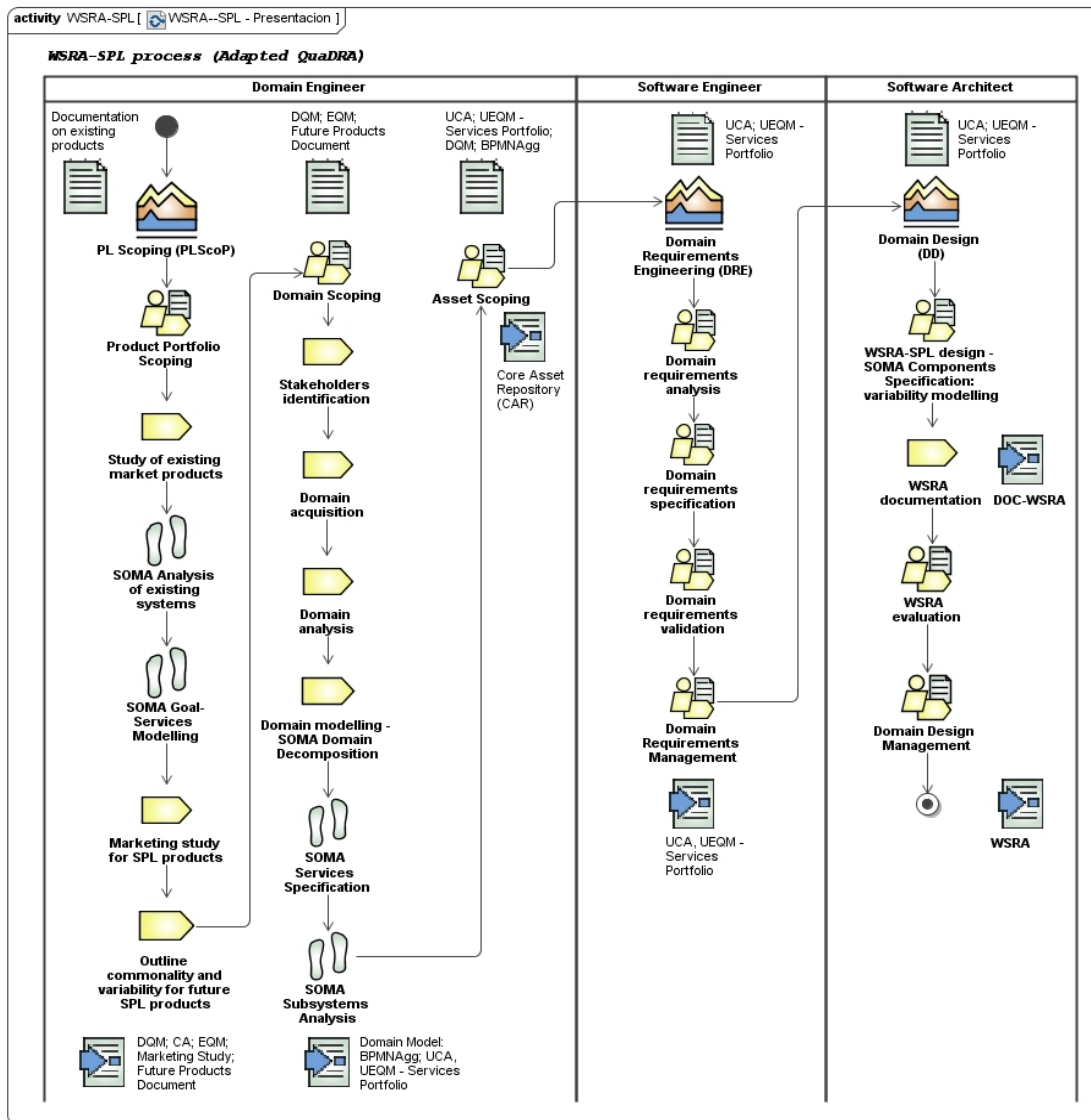


Figura 22. Proceso WSRA-SPL en SPEM; fuente: autor

# Referencias

---

- [Ars04] Arsanjani, A. (2004). Service-oriented modelling and architecture How to identify, specify, and realize services for your SOA. IBM Software Group. November.
- [Bjø06] Bjørner, D. (2006). Software Engineering 3 Domains, Requirements, and Software Design. Texts in Theoretical Computer Science. EATCS Series. Editors: W. Brauer G. Rozenberg A. Salomaa. Springer-Verlag Berlin Heidelberg.
- [Bro12] Brosgol, B. (2012). ADA 2012: The joy of contracts, Electronic Design.
- [CD15] Carlos, P. & Diego, J. (2015). SPLIT: An Automated Approach for Enterprise Product Line Adoption Through SOA, Journal of Internet Services and Information Security (JISIS), (5) 1 pp. 29-52, Feb.
- [CK10] Cohen, S. & Krut, R. (2010). Managing Variation in Services in a SPL Context, Technical Note. CMU/SEI-2010-TN-007. Carnegie Mellon University.
- [DD83] Department of Defense of the U.S.A. (1983). ADA Programming Language ANSI/MIL-STDS 1815-A, USA. January 1983.
- [GZ14] Gómez, J. D. & Zapata, C. (2014). SPL como complemento para el desarrollo de soluciones orientadas a servicios (SOA): Una revisión de la literatura, TG Especialista en Ingeniería de Software, Universidad de Medellín, Colombia.
- [HHJ+07] Helferich, A., Herzwurm, G., Jesse, S. & Mikusz, M. (2007). Software Product Lines, Service-Oriented Architecture and Frameworks: Worlds Apart or Ideal Partners? Draheim and Weber (Eds), TEAA 2006, Berlin Heidelberg, LNCS 4473, pp. 187-201.
- [HLO16b] J. Herrera, F. Losavio, and O. Ordaz, QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines based on ISO/IEC 26550, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 1, pp. 20-38, enero-junio 2016.
- [HLO16d] J. Herrera, F. Losavio, and O. Ordaz, Web-services reference architecture for software product lines: A quality-driven approach, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 2, pp. 7-21, julio-diciembre 2016.
- [IBM09] IBM. (2009). IBM Service-oriented Architecture (SOA) Solutions. IBM SOA Foundation.
- [ISO11] ISO/IEC. (2011). ISO/IEC 25010: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models. ISO/IEC JTC1/SC7/WG6, Draft.

- [ISO98] ISO/IEC. (1998). ISO/IEC 13236: Information Technology – Quality of service: Framework, ISO/IEC JTC 1/SC 6.
- [Kru00] Kruchten, P. (2000). Architectural Blueprints—The Rational Unified Process An Introduction. 2nd Edition.
- [LOE15] Losavio, F., Ordaz, O. & Esteller, V. (2015). Refactoring-Based Design of Reference Architecture, *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS)* 5(1) pp. 32-48.
- [Mil01] Milanovic, N. (2001). Contract-based Web Service Composition Framework, Doctoral Dissertation, Humboldt University, Berlin.
- [MR12] Martínez Aguilar N.E. & Román Urbietta A. Jr. (2012). Diseño de una arquitectura orientada a servicios para un establecimiento de salud de nivel de complejidad I-3, Tesis para Ingeniero de Software, Facultad de Ingeniería, Universidad Peruana de Ciencias Aplicada, Lima.
- [OMG08a] OMG. (2008). OMG - HL7, Practical Guide to SOA in Healthcare, Healthcare Services Specification Project, Health Level Seven, OMG.
- [OMG08b] OMG Object Management Group. (2008). Software & Systems Process Engineering Meta-Model Specification Version 2.0. MA: Object Management Group. Document number: formal/2008-04-01.
- [PBL05] Pohl, K., Bockle, G. & Van Der Linden, F. (2005). *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer.
- [SG96] Shaw, M. & Garlan, D. (1996). *Software architecture: perspectives on an emerging discipline*. Prentice Hall.
- [W3C04] W3C Working Group. (2004). Web Services Architecture Note 11, Feb. 2004.

***CAPÍTULO VI***

***APLICACIÓN DE LOS PROCESOS***

***QuaDRA Y WSRA-SPL AL DOMINIO DE***

***LOS SISTEMAS DE INFORMACIÓN***

***INTEGRADOS DE SALUD (SIS)***

---

---

## CAPÍTULO VI

### APLICACIÓN DE LOS PROCESOS QuaDRA y WSRA-SPL AL DOMINIO DE LOS SISTEMAS DE INFORMACIÓN INTEGRADOS DE SALUD (SIS)

De acuerdo a las actividades principales que integran cada una de las fases de los procesos QuaDRA y WSRA-SPL para la construcción de una AR en contextos LPS y LPSOS respectivamente, mostrando las técnicas utilizadas y los artefactos producidos, se desarrolla a continuación el siguiente caso de estudio en el dominio de los *Sistemas de Información Integrados de Salud (SIS)* o "*Healthcare Integrated Information Systems (HIS)*". Los SIS, como ya se mencionó en el Capítulo 2 son sistemas complejos que responden a las propiedades de calidad prioritarias disponibilidad, interoperabilidad y seguridad. En la práctica médica actual, LPS para SIS aún no han sido completamente definidas, desarrolladas y adoptadas; la falta de acuerdo sobre las normas médicas y las cuestiones psicosociales dificulta la interoperabilidad de la información médica, en particular de las *Historias Clínicas Electrónica* o "*Electronic Healthcare Information Systems (EHR)*", y su adopción general sigue siendo difícil, aún si se han promulgado a niveles gubernamentales, leyes y reglamentos específicos para alcanzar estos objetivos a nivel mundial.

#### **6.1 Alcance del Caso de Estudio**

La mayoría de los doce SIS identificados en [MID10] para establecer los principales requisitos de un SIS, son de código abierto y han sido construidos teniendo en cuenta el episodio de la interacción médico-paciente; todos ellos se centraron en el manejo de la *Historia Clínica Electrónica (HCE)* o "*Electronic Health Records (EHR)* para *Healthcare Integrated Information Systems (EHR-HIS)*", como la funcionalidad principal. La presente investigación se basa en esta premisa. El dominio de los SIS para nuestra LPS se limita entonces a sus funcionalidades básicas para la gestión de HCE, la atención al paciente con

planificación de citas y captura de datos demográficos, la emisión de órdenes médicas e informes y los servicios administrativos básicos, como la facturación por atención de pacientes. Los servicios de imagen y laboratorio, gestión de salas del hospital, servicios de enfermería, urgencias, administración general del hospital, etc. no se han considerado en estos primeros estudios [LOE15] [LOS15]. Se ha simplificado el caso de estudio, ya que han sido tratados los elementos de SIS a un alto nivel de detalles como corresponde al diseño arquitectónico, evitando detalles de bajo nivel, para facilitar la ilustración de nuestro enfoque, siguiendo también el espíritu de la fase del Alcance de la LP.

## **6.2 Aplicación paso a paso del proceso QuaDRA**

### ***6.2.1 Aplicación del Proceso de Diseño QuaDRA para el Dominio de los SIS***

QuaDRA será aplicado paso a paso a continuación:

#### ***I. Fase Alcance de la LP (PLScOP)***

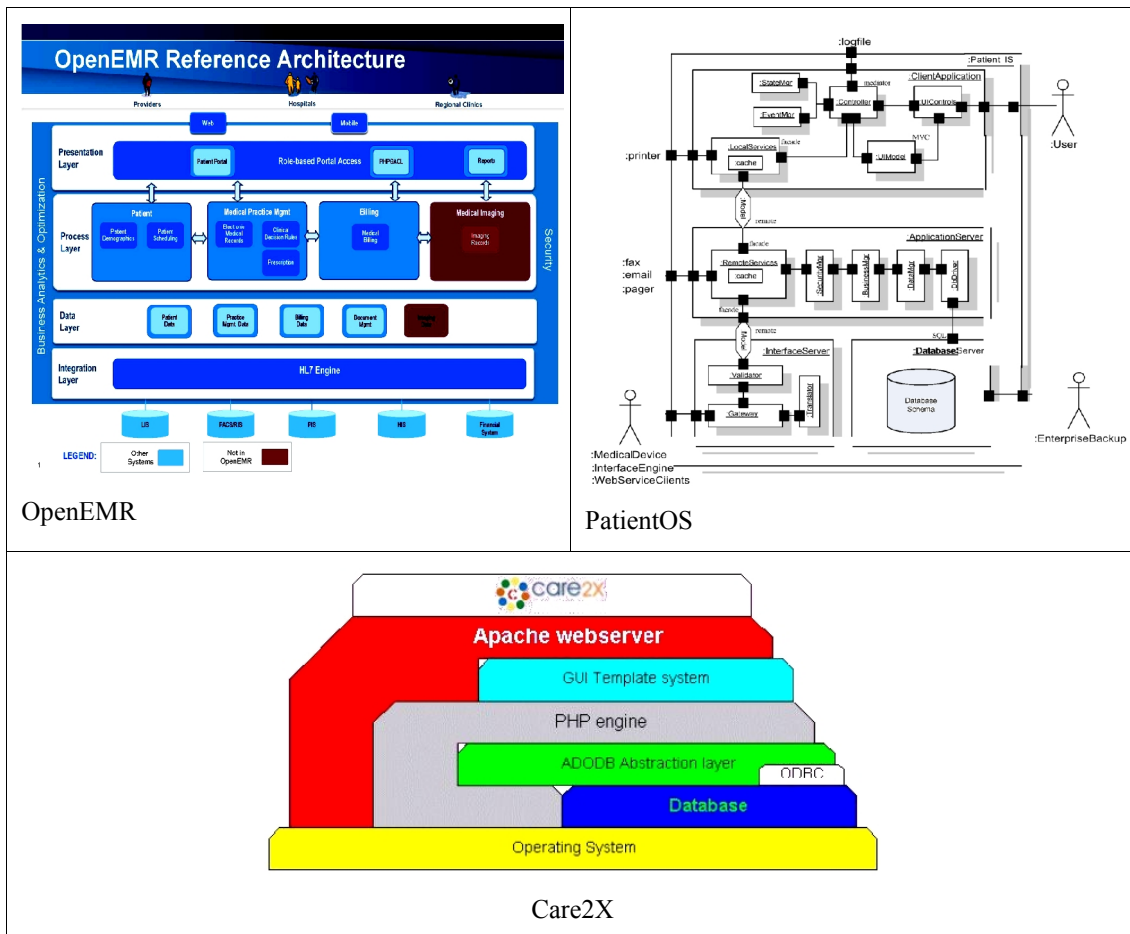
*1. Alcance del Portafolio de Productos - determina los requisitos para los productos futuros de la LPS:*

##### *1.1 Estudio de los productos existentes del mercado en el dominio*

La mayoría de los doce sistemas encontrados en [MID10] se centraron en el manejo del EHR como funcionalidad principal y muestra claras limitaciones, a saber, no pueden ser modificadas para requisitos particulares, pobre interfaces usuario como “*front-ends*”, dificultad de ser instalados, no hay suficiente tecnología para soportar las capacidades de telemedicina, etc. Todos ellos consideran el episodio físico de encuentro médico-paciente. Para esta investigación, cuatro sistemas de código abierto actualmente en el mercado, discutidos en [MID10], fueron ampliamente estudiados para producir la arquitectura candidata inicial (AC). Según el estudio en [Sam10], OpenEMR [Paz10] y PatientOS [Cau13] presentan un uso de 90% y 92% respectivamente, según datos del 2010. OpenEMR es un sistema basado en la Web de gestión de expediente médico electrónico (EMR), ofreciendo servicios EHR-HIS habituales para clientes distribuidos geográficamente y diferentes plataformas, garantizando portabilidad e interoperabilidad a través de Internet, apoyando también



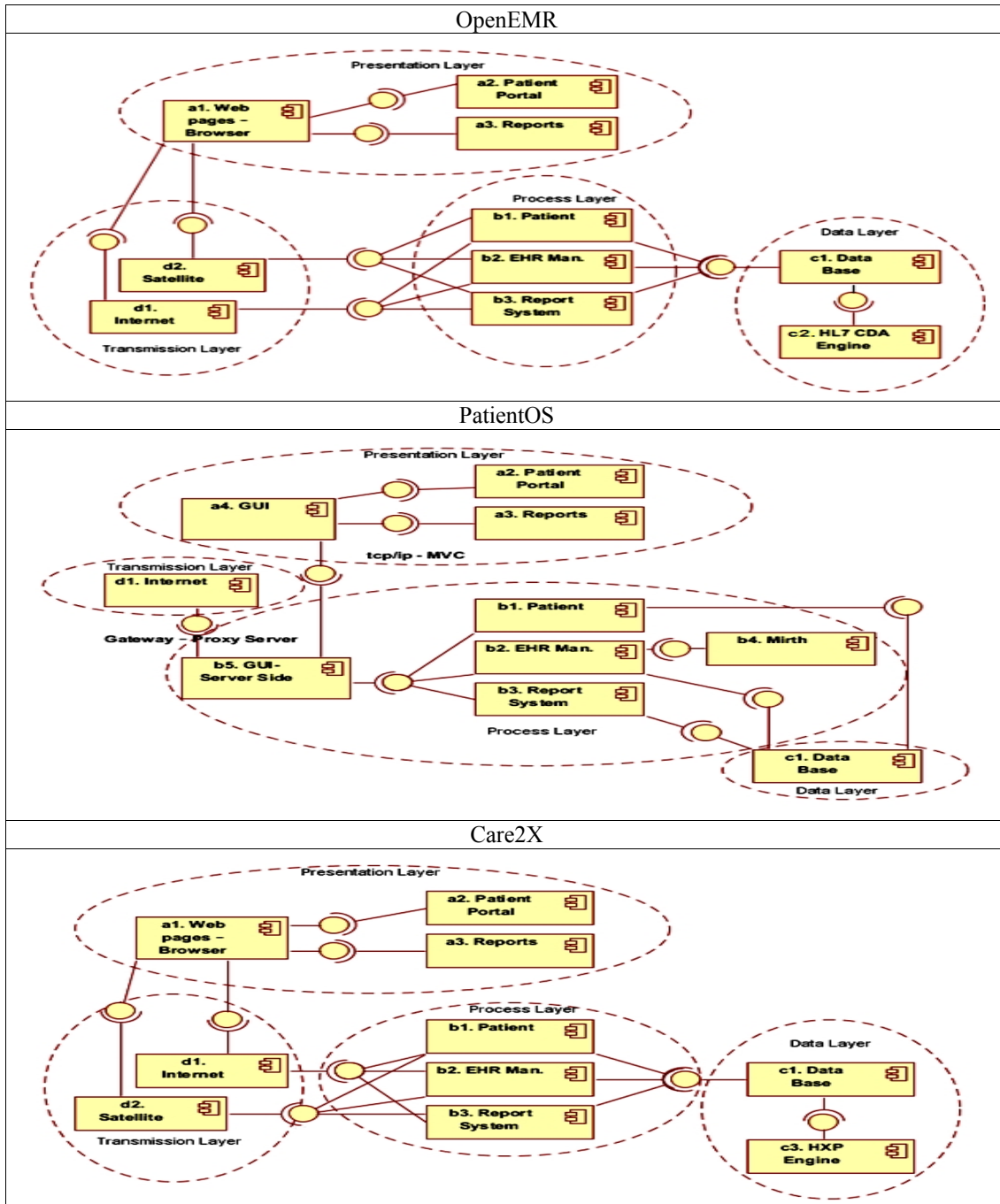
los dispositivos móviles. PatientOS es un sistema stand-alone, con el front-end de OpenEHR, siguiendo un modelo distribuido cliente/servidor de sistemas de información de 3 capas; tiene una interfaz gráfica usuario (GUI) basada en MVC para asegurar la mantenibilidad y recupera servicios remotos de un servidor Web a través de la capa de proceso. A continuación, en la Figura 23 se presentan las tres arquitecturas analizadas, tal como se encontraron en la literatura, OpenEMR, PatientOS y Care2x.



**Figura 23.** Las tres arquitecturas de SIS estudiadas originales, OpenEMR, PatientOS y Care2X [Paz10] [Cau13] [HBM09]

Otros sistemas de código abierto utilizados en proyectos nacionales recientes fueron Care2x [COS05], muy similar a OpenEMR y GNU-Health muy similar a PatientOS; GNU-Health no se incluyó en nuestra AC debido a la pobre documentación

arquitectónica. Luego, se construyen por refactorización las arquitecturas en UML 2.X de los productos existentes considerados, en la Figura 24 se presentan los diagramas UML de los 3 productos considerados: OpenEMR, PatientOS y Care2x.



**Figura 24.** Las tres arquitecturas de SIS estudiadas, en UML OpenEMR, PatientOS y Care2X [LOE15]

En conclusión, se encuentran pocos sistemas de salud de código abierto maduros disponibles en los mercados y menos aún una LPS para SIS. En consecuencia, la construcción de una LPS-SIS, es una meta factible e interesante a nivel de investigación. En la práctica industrial, la producción adaptable de sistemas SIS podría ser también un reto interesante, realizando estudios de costos-beneficios apropiados.

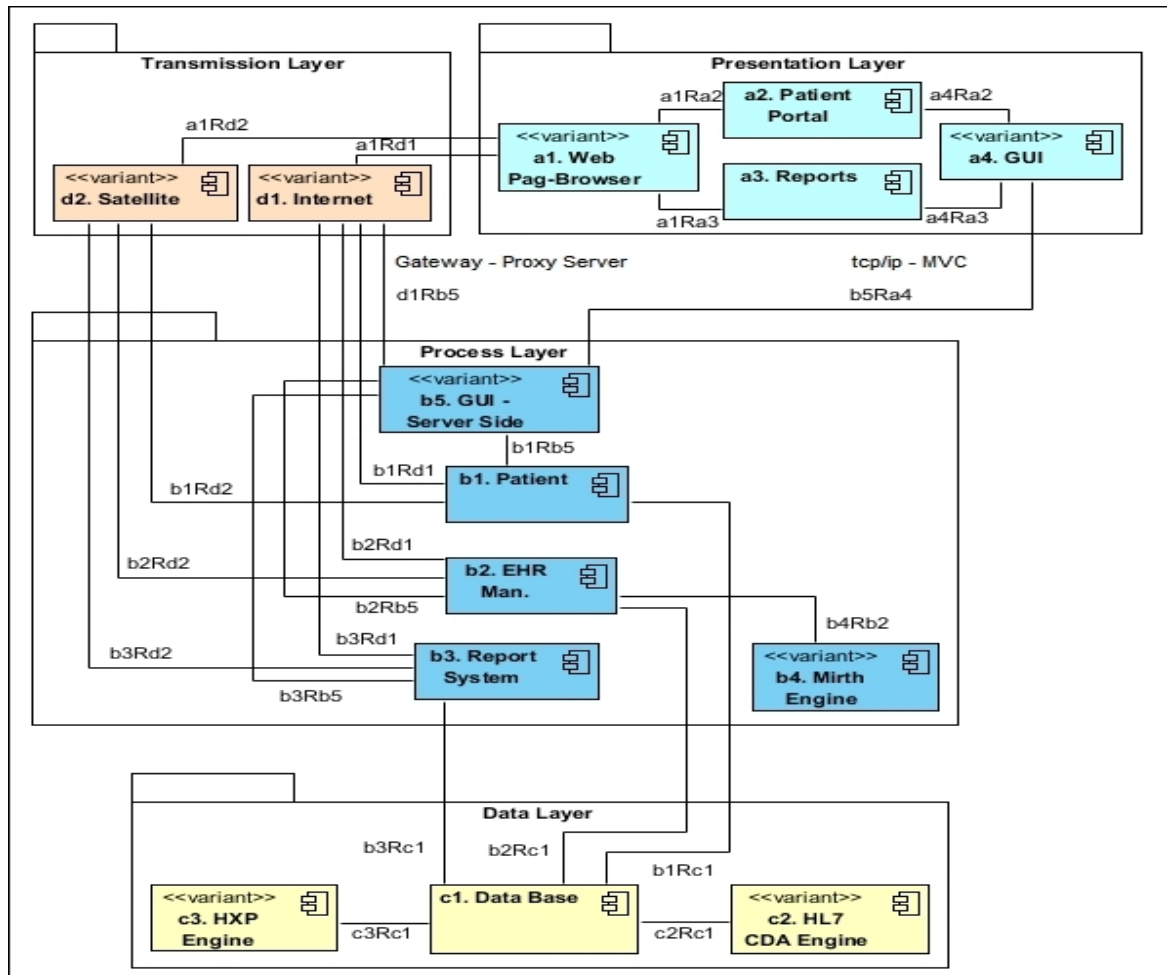
La Tabla 31 presenta los componentes de las arquitecturas después de la unificación de nombres de los componentes, respetando la semántica de las funciones lo más lejos posible, de los tres sistemas de código abierto OpenEMR, PatientOS y Care2X. La semejanza semántica de los componentes fue estudiada en cada producto, sobre la base de la documentación disponible, para determinar la base común de funcionalidades.

Los grafos denotados por  $(P_i, R_i)$ ,  $i=1, 2, 3$  de las configuraciones arquitectónicas (conectadas) de los tres productos considerados, donde  $P_i$  es el conjunto de componentes o nodos y  $R_i$  es el conjunto de aristas o conectores. Se denota  $AC = (P, R)$  como la arquitectura inicial obtenida automáticamente, donde  $P = \cup P_i$  y  $R = \cup R_i$ ,  $i=1, 2, 3$ . Una arista  $aRb \in R$  es variante y será denotada por  $aRb$  si y solamente si  $\exists i, j$  tal que  $a, b \in P_i \cap P_j$  y  $\exists a R_i b$  y  $\neg \exists a R_j b$ . El hecho de que  $a, b \in P_i$  y  $\neg \exists a R_j b$  para alguna  $i$  (ninguna conexión entre  $a$  y  $b$ ), fue señalada en la Tabla 31 por "X". Los componentes y conectores comunes y variantes son también señalados; el símbolo "-" indica la ausencia de un componente o la ausencia de un conector específico, ya que uno de sus componentes está ausente. Los componentes comunes se resaltan el gris en la Tabla 31.

**Tabla 31.** CCT: Componentes y Conectores de cada configuración arquitectural de productos [LOE15]

OpenEMR	PatientOS	Care2x
<i>a. Capa de Presentación</i>	<i>a. Capa de Presentación</i>	<i>a. Capa de Presentación</i>
<i>Componentes:</i>	<i>Componentes:</i>	<i>Componentes:</i>
a1. Web pages-Browser	-	a1
a2. Patient Portal	a2	a2
a3. Reports	a3	a3
-	a4. GUI	-
<i>Conectores:</i>	<i>Conectores:</i>	<i>Conectores:</i>
a1Ra2	-	a1Ra2
a1Ra3	-	a1Ra3
a1Rd1	-	a1Rd1
a1Rd2	-	a1Rd2
-	a4Ra2	-
-	a4Ra3	-
-	a4Rb5	-
<i>b. Capa de Procesos</i>	<i>b. Lógica del Negocio</i>	<i>b. Capa de Procesos</i>
<i>Componentes:</i>	<i>Componentes:</i>	<i>Componentes:</i>
b1. Patient	b1. Patient business model	b1
b2. EHR Management	b2	b2
b3. Report Systems	b3	b3
-	b4. Mirth (HL7 Engine)	-
-	b5. GUI-Server Side	-
<i>Conectores:</i>	<i>Conectores:</i>	<i>Conectores:</i>
b1Rc1	b1Rc1	b1Rc1
b1Rd1	X	b1Rd1
b1Rd2	-	b1Rd2
-	b1Rb5	-
b2Rc1	b2Rc1	b2Rc1
b2Rd1	X	b2Rd1
b2Rd2	-	b2Rd2
-	b2Rb4	-
-	b2Rb5	-
b3Rc1	b3Rc1	b3Rc1
b3Rd1	X	b3Rd1
b3Rd2	-	b3Rd2
-	b3Rb5	-
-	b5Rd1	-
<i>c. Capa de Datos</i>	<i>c. Capa de Datos</i>	<i>c. Capa de Datos</i>
<i>Componentes:</i>	<i>Componentes:</i>	<i>Componentes:</i>
c1. Data Base	c1	c1
c2. HL7 CDA Engine	-	-
-	-	c3. HXP Engine
<i>Conectores:</i>	<i>Conectores:</i>	<i>Conectores:</i>
c1Rc2	-	-
-	-	c1Rc3
<i>d. Capa de Transmisión</i>	<i>d. Capa de Transmisión</i>	<i>d. Capa de Transmisión</i>
<i>Componentes:</i>	<i>Componentes:</i>	<i>Componentes:</i>
d1. Internet	d1	d1
d2. Satellite	-	d2

La AC existente para SIS se reutilizó para este caso de estudio [LOE15], véase la Figura 25, construida mediante la unión de los grafos correspondiente a los productos OpenEMR, PatientOS y Care2X (ver figura 24) de la Tabla 31 CCT, aplicando la etapa de Análisis del Dominio del proceso RA&NFR-VAR [LOE15].



**Figura 25.** AC para el dominio SIS – unión de grafos [LOE15]

El MCD ha sido presentado en el Capítulo II, consulte la Figura 10.

Ahora se construirá el modelo de calidad extendido, la tabla MCE, con las soluciones arquitecturales requeridas por los componentes funcionales para satisfacer las propiedades de calidad; esta se muestra en la Tabla 32 como una documentación textual de la AC, donde se especificarán los componentes con la calidad requerida, incluyendo las prioridades y soluciones arquitectónicas

proporcionando estas cualidades. MCE equivale a presentar los escenarios comúnmente usados en métodos de diseño arquitectónicos centrados en atributos de calidad, pero dirigidos a un solo sistema, por ejemplo ADD [BKB02] y es una forma de representar el modelo FODA [LKK02].

En el MCE se recolectan “escenarios” sobre el conocimiento del dominio, respecto a que componente requiere y/o proporciona las propiedades de calidad; con esto se determina la trazabilidad entre componentes que resuelven funcionalidades y componentes que proporcionan soluciones o mecanismos para satisfacer la calidad requerida por esas funcionalidades, para que su comportamiento sea idóneo.

El MCE se construye de la siguiente manera: se especifican por capas cada uno de los componentes de la AC inicial; se especifican por cada componente de la AC las propiedades de calidad del MCD que requiere; se asignan por cada propiedad de calidad el (los) mecanismo (s) o solución (es) arquitectural (es) que satisfaga (n) la propiedad de calidad (estilos/patrones, componentes tecnológicos del mercado); y se enuncian las posibles restricciones que deben ser consideradas al momento de establecer una configuración arquitectónica válida.

**Tabla 32.** MCE: Vista general de la configuración arquitectónica de la AC

Componentes Comunes (CC)	Componente Variante <<variant>>	Descripción componente	Propiedades de Calidad del Componente – Prioridad se muestra en (); 1 alta, 2 media, 3 baja;		Restricciones
			Requerida (s)	Proporcionada (s)	
	<b>a1-WebPage</b> (Browser)	- Interfaz de Usuario en páginas Web (IU) sobre un Browser; contiene botones a2 y a3 para acceder las funcionalidades HCE-SIS	- Autenticidad (1), - Confidencialidad (1), - Integridad (1), - Disponibilidad-Persistencia (1), - Comportamiento en Tiempo (2), - Adaptabilidad (2), - Capacidad-Escalabilidad (2), - Interoperabilidad (1), - Usabilidad	- d1 y/o d2          - no es considerada	<b>a1, a4</b> no pueden estar a la vez presentes en una configuración arquitectónica; uno de ellos es <b>obligatorio RF CC</b> . Comportamiento en Tiempo, autenticidad, confidencialidad, integridad, son resueltos por los protocolos de Internet (HTTP, HTTPS) por d1 o d2; Disponibilidad y Capacidad-escalabilidad dependen de la conectividad de la red. Capacidad de adaptación depende de los servicios Web a través de d1 o d2.
	<b>a4-GUI</b> (stand alone)	- Stand-alone IU; contiene botones (o menús) a2 y a3 para acceder las funcionalidades EHR-	- Autenticidad, - Confidencialidad, - Integridad,  - Disponibilidad-	- d1 y/o d2       - b5 vía TCP/IP,	

	HIS	Persistencia, - Comportamiento en Tiempo  - Modificabilidad (3), - Modularidad (3),  - Usabilidad	proxy, o gateway en la Capa de Procesos  - b5 vía patrón MVC en la Capa de Procesos  - no es considerada	Capacidad-Escalabilidad depende de la capacidad de la red para transmitir datos. Disponibilidad y Comportamiento en Tiempo para a4 depende del lado del servidor b5, responsable de las comunicaciones. La usabilidad no depende de la arquitectura o el estilo, no ha sido considerado en este contexto y no es parte del MCD-SIS. Compleitud está asegurada por la construcción.
<b>a2-Patient Portal</b>	- botón para acceder las funcionalidades comunes b1, b2 del EHR-HIS en la Capa de Procesos - botón para acceder la funcionalidad b3	- las propiedades de calidad anteriores también son requeridas por a2, a3	- los componentes anteriores también proporcionan la calidad para a2, a3	<b>a2, a3 son obligatorias CC FR</b> ; ellos son IU de botones, actualmente son parte de a1 y a4.
<b>a3-Reports</b>				
<b>b1-Patient</b>	- servicios para la admisión del paciente a un centro de salud, en una cita programada, asistido por el personal médico autorizado: recuperación, almacenamiento, modificación, eliminación de datos personales y demográficos del paciente, resultados de imágenes/laboratorio, y servicios de programación de citas	- Correctitud-Precisión (2), - Disponibilidad-Persistencia,  - Autenticidad, - Confidencialidad, - Integridad  - Adaptabilidad - Capacidad-Escalabilidad  - Interoperabilidad,	- componentes en la Capa de Procesos  - d1 o d2 o por componentes en las Capas de Procesos y/o Datos  - componentes en la Capa de Datos	<b>b1, b2, b3 son obligatorias CC FR</b> - si los servicios EHR son implementados como servicios Web, la Adaptabilidad es resuelta; en caso contrario la Adaptabilidad y Capacidad-Escalabilidad pueden ser resueltas por c1 en la Capa de Datos
<b>b2-EHRMang</b>	- servicios básicos para la gestión del EHR: recuperación, creación, almacenamiento, modificación, compartición	- Disponibilidad-Persistencia, - Autenticidad, - Confidencialidad, - Integridad	- b4 in Process Layer y/o c2 o c3 in Data Layer  - d1 o d2 o por componentes en la Capa de Procesos y/o Datos	<b>- Interoperabilidad es obligatoria RNF SIS</b> resuelta por b4 en la Capa de Procesos; c2 o c3 son soluciones opcionales para la Interoperabilidad en la Capa de Datos;
<b>b3-Report System</b>	- servicios para la emisión, recuperación, modificación, y almacenamiento de documentos relacionados con la práctica médica (reportes, ordenes, facturación)	- Adaptabilidad, - Capacidad-Escalabilidad - Autenticidad, - Confidencialidad, - Integridad, - Disponibilidad-Persistencia  - Correctitud-Precisión	- componentes en la Capa de Datos  - d1 o d2 o por componentes en la Capa de Procesos y/o Datos  - componentes en la Capa de Procesos	
<b>b4-MirthEng</b>	- servicios de traducción para EHR hacia el formato estándar HL7	- Interoperabilidad	- engines disponibles en el mercado	<b>b4 es obligatorio</b> para resolver el RNF de interoperabilidad en SIS

	<b>b5-GUI Server Side</b>	- permite comunicación cliente/servidor local y remota	- Comportamiento en Tiempo - Modificabilidad - Modularidad	- protocolos - módulos	- TCP/IP, proxy o gateway  - patron MVC
<b>c1-DB</b>		- servicios usuales proporcionados por un SGBD (Sistema de Gestión de Base de Datos)	- Capacidad- Escalabilidad - Integridad - Disponibilidad- Persistencia  - Adaptabilidad - Interoperabilidad	- módulos, mecanismos  - APIs - c2, c3	- <b>c1 es obligatoria CC FR;</b>
	<b>c2-HL7Eng</b> <b>c3-HXPeng</b>	- servicios de traducción a formatos estándares para la adaptación o adecuación de documentos EHR	- Interoperabilidad	- módulos, mecanismos o engines	- c2, c3 are opcional; al menos b4 debe estar presente en una configuración arquitectural
	<b>d1-Internet</b>	- servicios de comunicación cableados/no cableados de área amplia	- Comportamiento en Tiempo, - Disponibilidad- Persistencia, - Adaptabilidad, - Capacidad- Escalabilidad - Autenticidad, - Confidencialidad, - Integridad	- hardware /software (protocolos, servicios, etc.) infraestructura de red cableada/no cableada	<b>d1 es obligatorio CC FR</b> , d2 es un RF opcional; ellos pueden estar presentes en una configuración arquitectural. Adaptabilidad depende de los servicios Web; Disponibilidad depende de la conectividad en la red;
	<b>d2-Satellite</b>	- servicios de comunicación móvil	- Las propiedades anteriores son mantenidas.	- los componentes anteriores son mantenidos.	Capacidad-Escalabilidad depende de la capacidad de la red para transportar los datos; los protocolos HTTP/HTTPS resuelven limitada seguridad; HTTP solo no es posible debido a los requerimientos de seguridad SIS; estos protocolos no son considerados aquí separadamente, ellos son utilizados en la Capa de Procesos para incrementar los niveles de seguridad.

### 1.2 Elaboración del estudio de mercado para la viabilidad de la LPS.

El estudio de mercado no se realizará explícitamente en este trabajo, porque se realizó en [LOE15].

### 1.3 Esbozo de la similitud y variabilidad para futuros productos de la LPS.

Nuevas características, no consideradas entre las funcionalidades básicas de la AC, pueden ser añadidas para conformar futuros productos EHR-HIS para la LPS a construirse:



- un sistema de imagen (LabImage System) para manejar los resultados de laboratorio y estudios de imágenes radiológicas, vinculándolos al EHR del paciente, necesarios, almacenados con formatos apropiados para ser compartidos por el personal médico e instituciones local o distante, y de rápida recuperación para visualización.
- un asistente de diagnóstico (Diagnosis) para ayudar a los médicos a realizar diagnosis en línea: recuperación de catálogos médicos para la identificación de enfermedades, protocolos para ser aplicados a una determinada enfermedad, etcétera.
- Facilidad para servicios de citas en línea para permitir que los pacientes programen sus citas en línea, seleccionen las instituciones de salud y especialistas en particular; consulta de la sintomatología en línea (SymptomAssistant) para verificar la sintomatología para seleccionar un especialista en particular; recuperar, modificar, quitar y almacenar datos personales y demográficos en línea en el EHR del paciente.

Posible similitud y variabilidad para los nuevos productos de la LPS, incluyendo las características anteriores son destacadas en la siguiente fase del Alcance del Dominio.

## ***2. Aplicación del Alcance del Dominio***

Siguiendo los pasos definidos en el alcance del dominio, se tiene:

### *2.1 Identificación de las Partes Interesadas (Stakeholders):*

La información fue obtenida de las visitas, entrevistas, talleres o cuestionarios en diferentes instituciones relacionadas con los grupos de interés: las instituciones de salud, las instituciones gubernamentales relacionadas con los desarrolladores de software y salud [LOS15]. Diferentes puntos de vista fueron identificados para conformar los grupos de interés (stakeholders):

Médicos, Ingenieros de Dominio, Directores de Instituciones gubernamentales de salud.

### *2.2 Adquisición del Dominio:*

Sentencias o declaraciones formuladas por los diferentes actores para ilustrar sus puntos de vista sobre el dominio EHR-HIS fueron capturadas. Recuerde que el alcance del dominio SIS considerado se limita al episodio de médico-paciente

encontrada bajo citas programadas “*scheduled appointments*”, que se llamará brevemente "Appointment Services" [MID10], y que se centra en el manejo del EHR.

### 2.3 Análisis del Dominio

Las declaraciones son agrupadas dentro de la Unidad de Descripción de Dominio (UDD) [Bjø06] y son representadas textualmente como una tabla (ver Tablas 33, 34 y 35), para cada grupo de interés identificados.

- *Especificación de las UDD*

**Tabla 33.** UDD para EHR-HIS de los Doctores

<b>Unidad de Descripción de Dominio EHR-HIS</b>
Punto de vista del grupo de partes interesadas: Médicos
<b>Declaraciones</b>
Paciente es atendido en el hospital bajo cita previa
Si es el primera cita del paciente, un nuevo EHR es creado por la enfermera, en caso contrario se recupera el EHR existente del paciente
El EHR del paciente es proporcionado por la enfermera y accesado por el médico
Nuevos exámenes y resultados de laboratorios pueden ser añadidos al EHR del paciente por el médico, si es el caso
Diagnóstico y ordenes/informes/facturación médica para el paciente son producidos por el médico para concluir la atención médica
Una nueva cita es programada por la enfermera si es requerido por el médico
Material y equipos médicos pueden ser requeridos por el médico para proporcionar atención médica adecuada

Esta UDD se refiere a los servicios de cita del paciente involucrando a las enfermeras, pacientes y médicos presentes en la institución de salud, haciendo hincapié en la gestión del EHR, según el alcance de este caso de estudio.

**Tabla 34.** UDD para EHR-HIS de los Ingenieros de Dominio

<b>Unidad de Descripción de Dominio EHR-HIS</b>
Grupo de punto de vista de partes interesadas: Ingenieros de Dominio
<b>Declaraciones</b>
El EHR-SIS es soportado por una arquitectura híbrida SOA/Capas
El estilo arquitectónico EHR-SIS soporta modificabilidad, interoperabilidad, rendimiento (comportamiento en tiempo); servicios de seguridad son proporcionados por los protocolos de Internet, transversalmente a todas las capas; sin embargo la disponibilidad depende de la conexión a internet; la seguridad también puede ser proporcionada o por módulos en la Capa de Proceso

La Capa de Transmisión es gestionada por un servidor Web que se comunica a todas las demás Capas
Acceso al EHR del cliente por un navegador en la Capa de Presentación, que se conecta a la Capa de Transmisión a través de un servidor Web
Unidades médicas deben tener amplio acceso a internet, intranet o vía satélite
Las principales funcionalidades EHR-SIS deben ser soportadas: servicios de cita del paciente y captura de datos demográficos, gestión del EHR y emisión de órdenes/informes/facturación médica.

**Tabla 35.** UDD para EHR-HIS de los Directores de Instituciones Gubernamentales de la Salud.

<b>Unidad de Descripción de Dominio EHR-HIS</b>
Grupo de punto de vista de partes interesadas: Directores de las instituciones gubernamentales de salud (HGI)
<b><i>Declaraciones</i></b>
Digitalizar la EHR con formato estándar para lograr el intercambio entre los médicos y las instituciones sanitarias nacionales e internacionales geográficamente distantes
Tener una base de datos de especialistas nacionales e internacionales
Desarrollar una plataforma Web para gestionar los servicios de citas en línea “on-line”

El conocimiento del dominio inicial es capturado desde la UDD en las Tablas 33, 34 y 35 y descrito textualmente en las Tablas 36, 38 y 39 por funciones intrínsecas, para cada faceta considerada.

- *Especificación de las Facetas*

Las facetas que se especificarán en este trabajo son Procesos de Negocio, Soporte Tecnológico y Reglas & Regulaciones. Los puntos de vista considerados para las tres facetas son: Grupo de médicos para los Procesos de Negocio, grupo de Ingenieros de Dominio para el Soporte Tecnológico y Directores de instituciones gubernamentales de salud (IGS) para Reglas & Regulaciones. Facetas, grupos de partes interesadas y puntos de vista fueron seleccionados sobre las bases del proceso de desarrollo en la ID para LPS, cuyo principal objetivo es diseñar una Arquitectura de Referencia.

Diversas notaciones pueden ser utilizadas para especificar las facetas desde los puntos de vistas de las partes interesadas con descriptores intrínsecos, cada uno ofreciendo diferentes detalles “granularidad” de especificaciones y capacidades

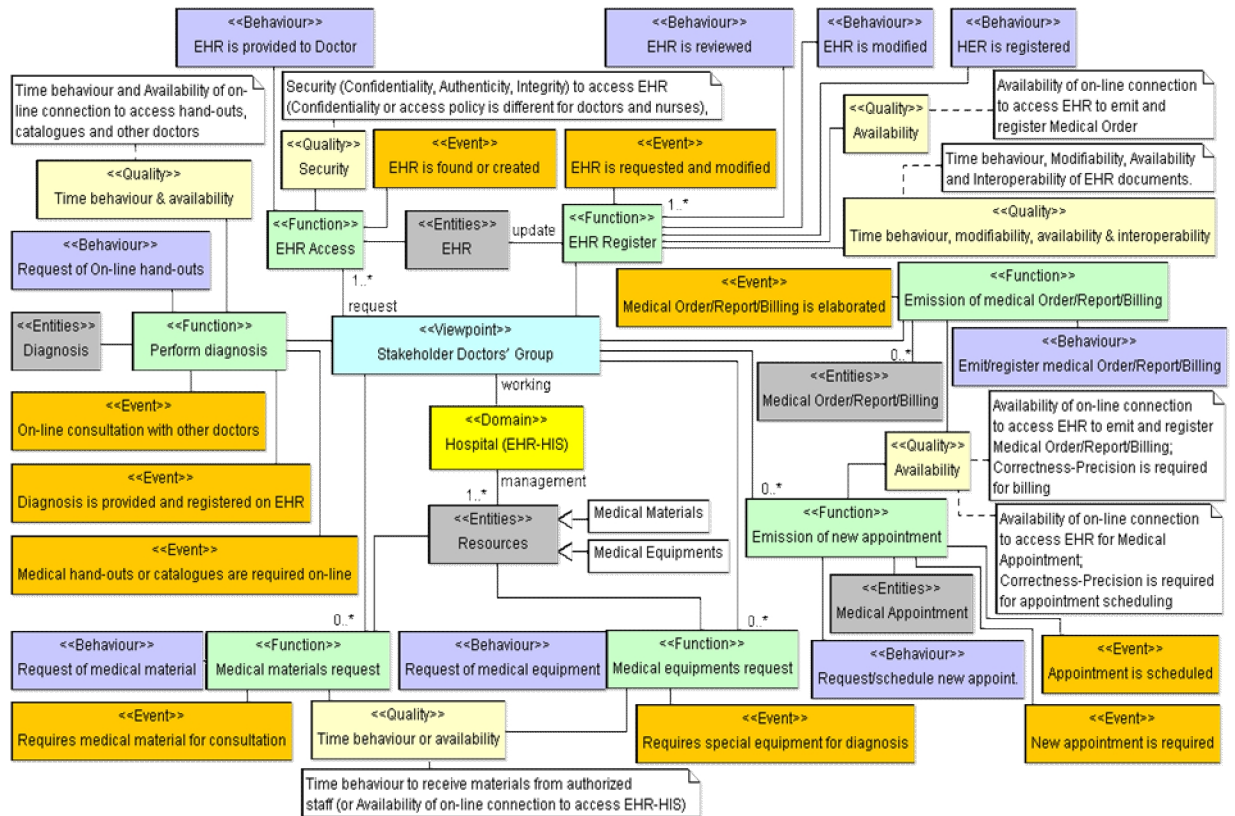
[Bjø06]; de cada especificación, mas detalles son extraídos; en este trabajo se utilizarán las siguientes notaciones:

- tablas para la especificación textual informal para las intrínsecas, para todas las facetas (ver Tablas 36, 38 y 39);
- diagramas UML [OMG05] (ver Figuras 26, 28 y 30) para todas las facetas;
- diagramas BPMN [OMG11] para todos los procesos de negocio (ver Figuras 27, 29, 31 y 32).

**Tabla 36.** Especificación por Intrínsecas del punto de vista de los Doctores para la Faceta de Procesos de Negocio Servicios de Cita.

<b>Dominio EHR-HIS</b>				
Punto de Vista del Grupo de Stakeholders: Doctores				
Servicios de Citas Faceta Procesos de Negocio				
<i>Entidades</i>	<i>Funciones</i>	<i>Eventos</i>	<i>Comportamiento</i>	<i>Calidad</i>
EHR	Acceso EHR	<i>EHR es encontrado o creado</i>	Enfermera confirma la existencia del EHR del paciente o crea un EHR para el nuevo paciente; la enfermera proporciona el EHR al médico <i>(EHR es proporcionado al médico)</i>	<i>Seguridad (confidencialidad, autenticidad, integridad)</i> para acceder el EHR: <i>confidencialidad</i> o políticas de acceso es diferente para las enfermeras y doctores, <i>autenticidad</i> es necesaria para unívocamente identificar al usuario, e <i>integridad</i> es requerida para la consistencia de los datos. <i>Disponibilidad</i> de la conexión en línea para acceder al EHR, <i>Comportamiento en tiempo</i> para el rápido acceso al EHR, <i>Modificabilidad</i> para cambiar el EHR, <i>Disponibilidad-Persistencia</i> para recuperar siempre el EHR, e <i>Interoperabilidad</i> para compartir el EHR.
	Registro EHR	<i>EHR es solicitado por el doctor; el doctor atiende al paciente medicamente y el EHR es modificado</i>	<i>EHR es accedido</i> y revisado por el doctor; puede añadir resultados de laboratorio y/o exámenes, si existe un EHR del paciente; el paciente es atendido médicamente; el EHR es modificado <i>(EHR es revisado, modificado, registrado)</i>	
Diagnóstico	Realizar diagnóstico	<i>Catálogos o prospectos Médicos son requeridos en línea; consulta en línea con otros doctores en diferentes instituciones de salud o localmente para realizar el diagnóstico; diagnóstico es proporcionado y registrado en el EHR</i>	Médico solicita en línea prospectos o catálogos; puede consultar en línea con otros médicos en diferentes instituciones de salud o a médicos locales, y el EHR del paciente debe ser compartida por otros médicos (situación ideal, ya que los factores psicosociales a menudo evitan la realización de esta actividad); revisar de resultados de laboratorio y/o exámenes; el médico produce el diagnóstico <i>(Solicitud de prospectos en línea)</i>	<i>Comportamiento en Tiempo y Disponibilidad</i> de la conexión en línea para tener rápido acceso a la consulta, catálogos y otros doctores.
Recursos: Materiales Médicos (*)	Solicitud de materiales médicos	<i>Doctor requiere material médico para la consulta</i>	Médico solicita materiales médicos llamando al personal autorizado (o utilizando el sistema EHR-SIS si este servicio está disponible) <i>(Solicitud de material médico)</i>	<i>Disponibilidad</i> física de materiales, <i>Comportamiento en tiempo</i> para recibir materiales de las personas autorizadas (o <i>Disponibilidad</i> de conexión en línea para acceder el sistema HER-SIS para la solicitud de materiales, en caso de que esta facilidad esté soportada)
Recursos: Equipos Médicos (*)	Solicitud de equipos médicos	<i>Doctor requiere equipo especial para el diagnóstico</i>	Médico solicita equipo médico llamando al personal autorizado (o utilizando el sistema EHR-SIS si este servicio está disponible)	<i>Disponibilidad</i> física de equipos, <i>Comportamiento en tiempo</i> para recibir equipos de personas autorizadas (o <i>Disponibilidad</i> de

			(Solicitud de equipo médico)	conexión en línea para acceder el sistema HER-SIS para la solicitud de equipos, en caso de que esta facilidad esté soportada)
Cita Medica	Emisión de una nueva cita	<i>Nueva cita es requerida</i> para el paciente, si es necesario; la cita es agendada	Doctor registra al paciente para una nueva cita; la cita es agendada (programada). <i>(Solicita/Programa una nueva cita)</i>	<i>Disponibilidad</i> de conexión en línea para acceder EHR para Citas Medicas; <i>Correctitud-precisión</i> es requerida para la agenda de citas
Orden/Reporte/ Factura Medica	Emisión de Orden/Reporte/ Factura medica	<i>Orden/Reporte/Factura es elaborada</i> para la presente consulta	Doctor emite Orden/Reporte/Factura medica <i>(Emite/registra Orden/Reporte/Factura medica)</i>	<i>Disponibilidad</i> de conexión en línea para acceder HER para emitir y registrar Orden/Reporte/Factura Medicas; <i>Correctitud-precisión</i> es requerida para la facturación
Nota: (*) Estos servicios no serán considerados para el EHR-HIS en este trabajo; el nombre del comportamiento es especificado dentro de ( ) en <i>itálico</i>				



**Figura 26.** Punto de vista de los Doctores de la Faceta Procesos de Negocios para Servicios de Citas (UML). Fuente: [HLO16b]

**Tabla 37.** Notación BPMN para la especificación de los Procesos de Negocios [OMG11]

BPMN Notación	Símbolo	Descripción	Interpretación (Figuras 5, 7, y 9)
	Tarea	Actividad simple o atómica representando el trabajo en una organización; ellas consumen recursos; no se detalla más.	Ellos pueden ser mapeadas (relacionadas) a componentes o servicios arquitecturales en la AR-LPS.
	Compuerta exclusiva	Elemento de control del flujo de trabajo de los datos; Un único camino es seleccionado entre varias alternativas.	Representa variabilidad en la LPS.
	Compuerta inclusiva	Elemento de control del flujo de trabajo de los datos; Uno o más caminos pueden ser seleccionados de varias alternativas.	Representa variabilidad en la LPS.
	Evento de inicio	Inicia un proceso.	Representa el punto de partida o de entrada de un proceso de negocio.
	Evento de inicio de mensaje	Un procesos es iniciado cuando un mensaje es recibido,	Representa los eventos que surgen de un proceso de negocio activo.
	Evento de finalización	Indica el final de un flujo de trabajo.	Representa el final de algún comportamiento en el proceso de negocio.
	Evento de finalización definitiva	Indica que un proceso termina, incluso si hay flujos de trabajo activos.	Representa el final de un proceso de negocio.
	Evento de enlace intermedio	Permite la conexión de dos secciones del proceso.	Permite una conexión entre dos secciones del proceso de negocios.
	Swimlanes (Pool)	Es un contenedor de procesos; representa un participante, entidad o roll.	Representa el punto de vista de los stakeholders.
	Swimlanes (Lane)	Son subdivisiones del contenedor de procesos; que representas los participantes dentro de una organización.	Representa los diferentes comportamientos del proceso de negocio.
	Objeto conector: Secuencia	Representa el flujo de trabajo y la secuencia de actividades.	Representa el flujo de ejecución de las actividades en un proceso de negocio.
	Objeto conector: Mensaje	Representa las interacciones entre procesos o contenedores.	Representa cambios de los puntos de vista de las partes interesadas.
	Asociaciones	Se utilizan para relacionar información adicional sobre el proceso.	Relacionan las entidades, actividades o funciones con las propiedades de calidad requeridas.
	Artefacto: objeto de datos	Ellos proporcionan información adicional sobre el proceso; muestran la información requerida por una actividad, tales como su entrada/salida.	Documento que especifica la calidad requerida por las entidades, actividades o funciones, en cada comportamiento.
	Agregación de actividades incluidas en un proceso	Permiten agrupar las actividades de un proceso (sub-proceso)	Representa una funcionalidad, sistema o subsistema; que puede ser mapeado (relacionado) a un componente o servicio de la AR LPS.
	Regla de Negocio	Permite especificar las reglas de negocio	Representa la calidad requerida por la agregación, "sintetizando" lo que se adjunta como comentarios.

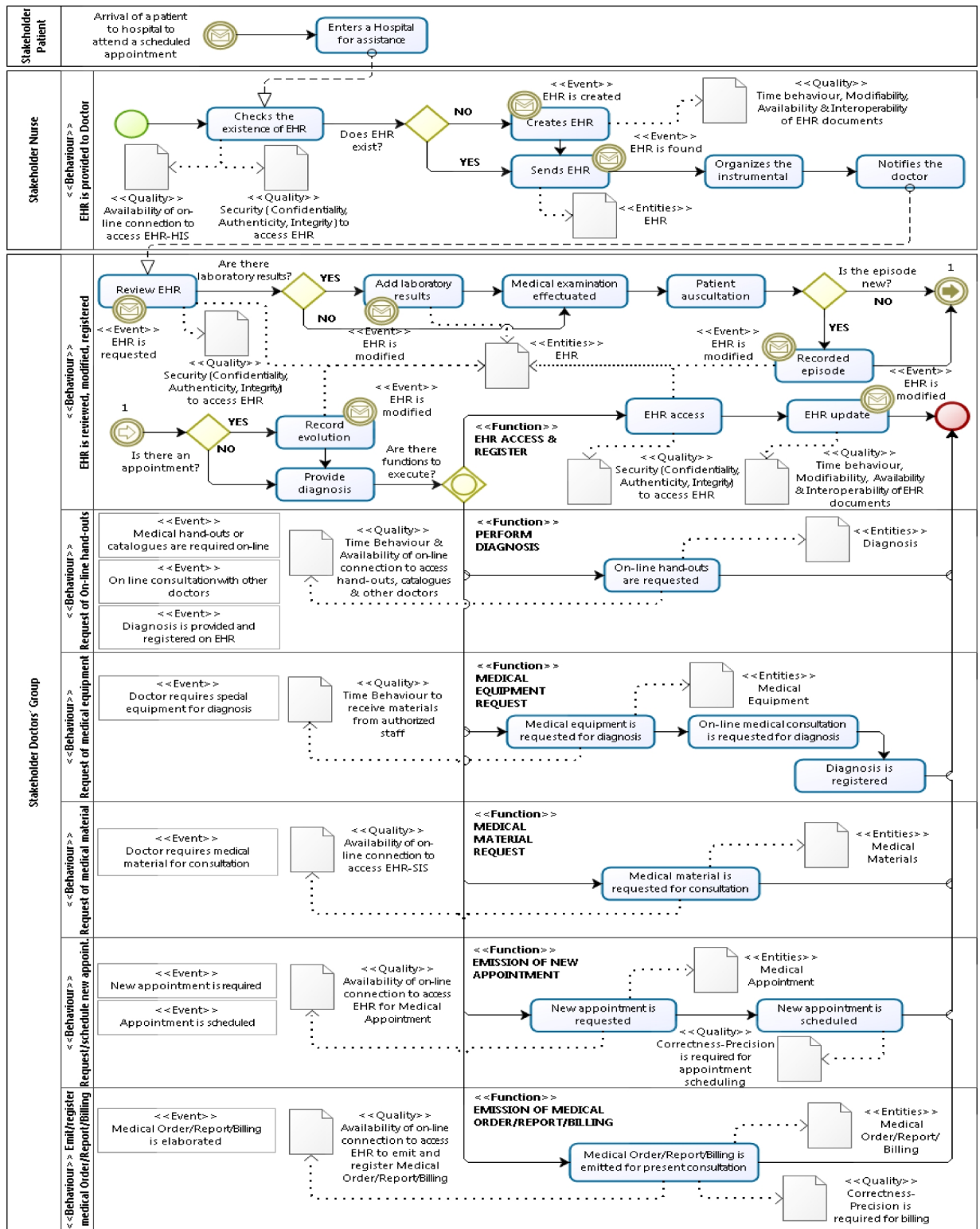


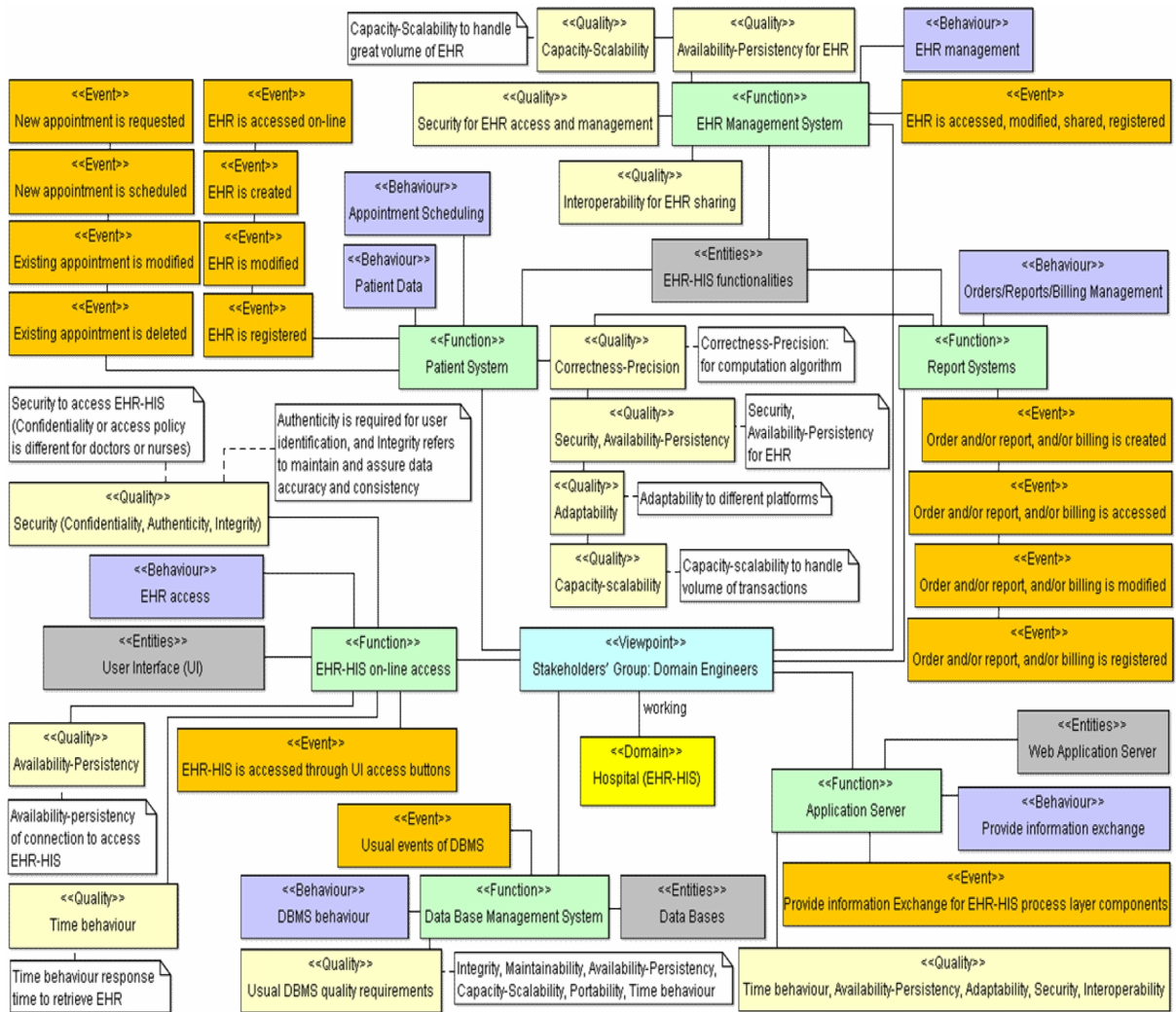
Figura 27. Procesos de negocios extraídos desde el punto de vista de los Doctores para la Faceta Procesos de Negocios (BPMN). Fuente: [HLO16b]

**Tabla 38.** Especificación por intrínsecas del punto de vista de los Ingenieros de Dominio para la Faceta Soporte Tecnológico.

<b>Dominio EHR-HIS</b>				
Punto de Vista Grupo de Stakeholders: Ingenieros de Dominio				
Faceta Soporte Tecnológico				
<b>Entidades</b>	<b>Funciones</b>	<b>Eventos</b>	<b>Comportamiento</b>	<b>Calidad</b>
Capa de Presentación: Interfaz de Usuario (IU)	Acceso en línea del EHR-HIS	- EHR-HIS es accedido a través de botones de la interfaz de usuario (IU)	- Se accede al EHR ( <i>EHR Access</i> )	- <i>Seguridad (Confidencialidad, Autenticidad, Integridad)</i> para acceder al EHR-SIS (Confidencialidad o políticas de acceso es diferente para los médicos o enfermeras); <i>Disponibilidad-Persistencia</i> de la conexión para acceder a la EHR-SIS; Se requiere Autenticidad para la identificación del usuario; La <i>Integridad</i> se refiere a mantener y asegurar la precisión y la coherencia de los datos; <i>Comportamiento en el tiempo</i> es el tiempo de respuesta para recuperar el EHR
Capa de Procesos: Funcionalidades EHR-SIS	Sistema del Paciente	- <i>Nueva cita es solicitada</i> para el paciente por el personal médico autorizado: - <i>Nueva cita está programada</i> - <i>Cita existente es modificada</i> - <i>Cita existente es eliminada</i>  - EHR del paciente si está presente es accedido en línea por parte del personal médico autorizado para comprobar los datos personales y demográficos del paciente; en caso contrario, el EHR del paciente es creado por el personal médico autorizado - EHR es modificado por el personal médico autorizado - EHR es registrado por el personal médico autorizado - EHR es accedido, modificado, compartido, registrado por el personal médico autorizado	- Servicios de programación para citas de los pacientes ( <i>Programación de la cita</i> )  - Datos personales y demográficos del paciente son registrados ( <i>Datos del Paciente</i> )	- <i>Correctitud-Precisión</i> : para los algoritmos de cálculo, - <i>Seguridad, Disponibilidad-Persistencia</i> para el EHR, - <i>Adaptabilidad</i> a diferentes plataformas, - <i>Capacidad-Escalabilidad</i> para manejar volumen de transacciones
	Sistema de Gestión EHR		- EHR es accedido, modificado, registrado durante la atención médica del paciente - EHR es visualizado por otro personal médico autorizado local o distante para la consulta en línea, para ayudar a la emisión del diagnóstico del paciente - EHR es modificado por el personal médico autorizado para añadir diagnóstico - EHR es modificado por el personal médico autorizado para añadir nuevos resultados de laboratorio/imagen ( <i>Gestión del EHR</i> )	- <i>Interoperabilidad</i> para compartir el EHR, <i>Seguridad</i> para el acceso y la gestión del EHR, <i>Disponibilidad-Persistencia</i> del EHR, porque no pueden ser destruidos y deben estar siempre disponibles, - <i>Capacidad-Escalabilidad</i> para manejar un gran volumen de EHR
	Sistema de Reporte	- Orden y/o informe, y/o facturación está creada (emitida) - Orden y/o informe, y/o facturación es accedida - Orden y/o informe, y/o facturación es modificada - Orden y/o informe, y/o facturación es registrada	Gestión médica de las órdenes/informes/facturación: edición, acceso, modificación, registro, en el EHR del paciente ( <i>Gestión de Órdenes/Informes/Facturación</i> )	- <i>Correctitud-Precisión</i> : para el algoritmo de cálculo, - <i>Seguridad, Disponibilidad, Persistencia,</i> - <i>Adaptabilidad</i> a diferentes plataformas, - <i>Capacidad-Escalabilidad</i> para manejar volumen de transacciones



Capa de Datos: Base de Datos	Sistema de Gestión de Base de Datos (SGBD) para cada base de datos EHR-HIS: EHRDB, AppointmentDB, etc.	- Eventos usuales de un SGBD	- Comportamiento habitual de un SGBD (DBMS behaviour)	- Requisitos de calidad habituales en un SGBD: <i>Integridad, Mantenibilidad, Disponibilidad, Persistencia, Capacidad-Escalabilidad, Portabilidad, Comportamiento en el tiempo</i>
Capa de Transmisión o comunicación: Servidor de Aplicaciones Web	Servidor de Aplicaciones	Proporcionar Intercambio de Información del EHR-SIS para los componentes en la Capa de Procesos	Permite el intercambio eficiente, confiable, seguro de información a través del servidor Web; la información es intercambiada de forma independiente de la plataforma ( <i>Proporcionar intercambio de información</i> )	- <i>Comportamiento en el tiempo, disponibilidad-persistencia, adaptabilidad, seguridad, interoperabilidad</i>



**Figura 28.** Punto de vista de los Ingenieros de Dominio para la Faceta Tecnologías Soportadas (UML). Fuente: [HLO16b]

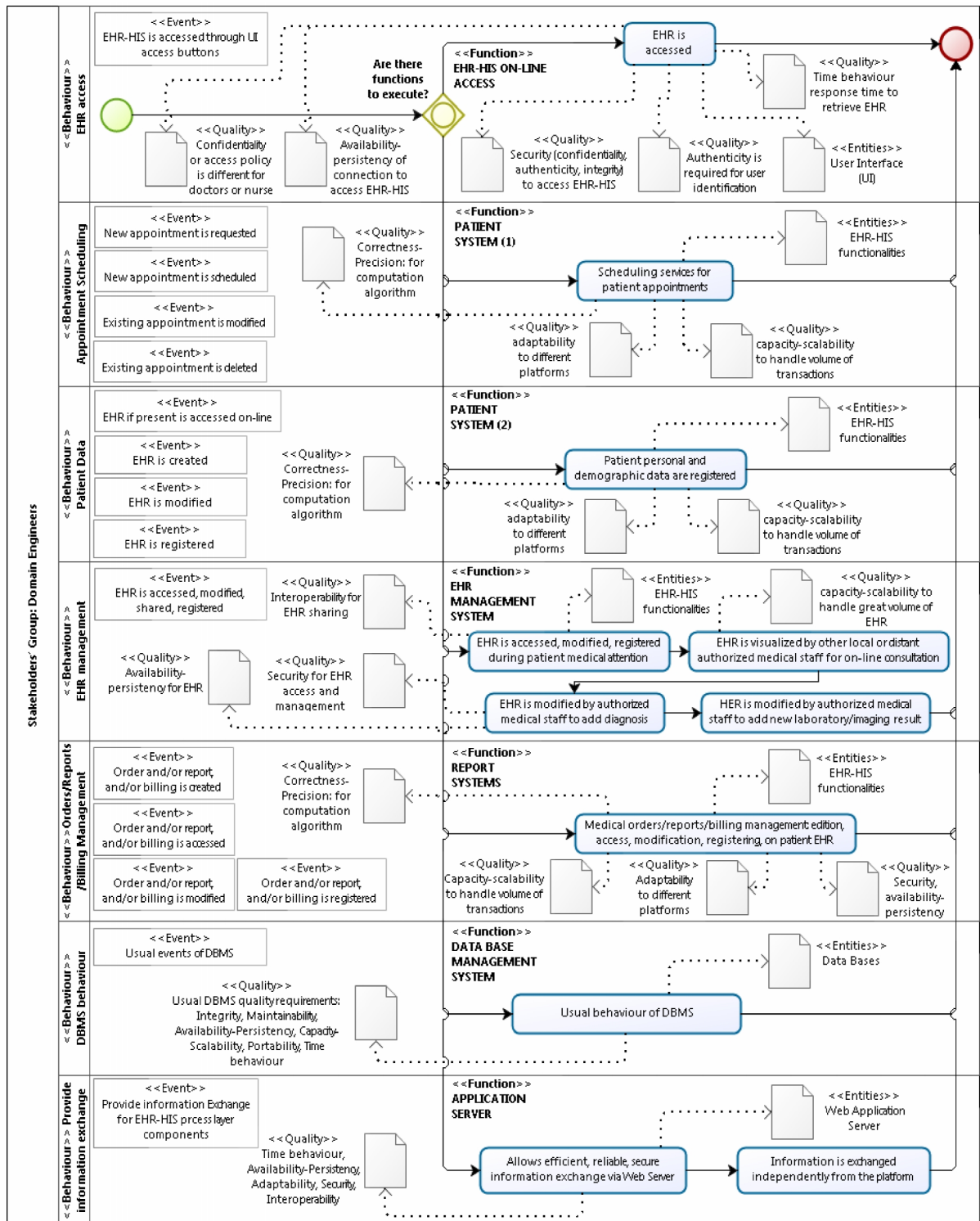
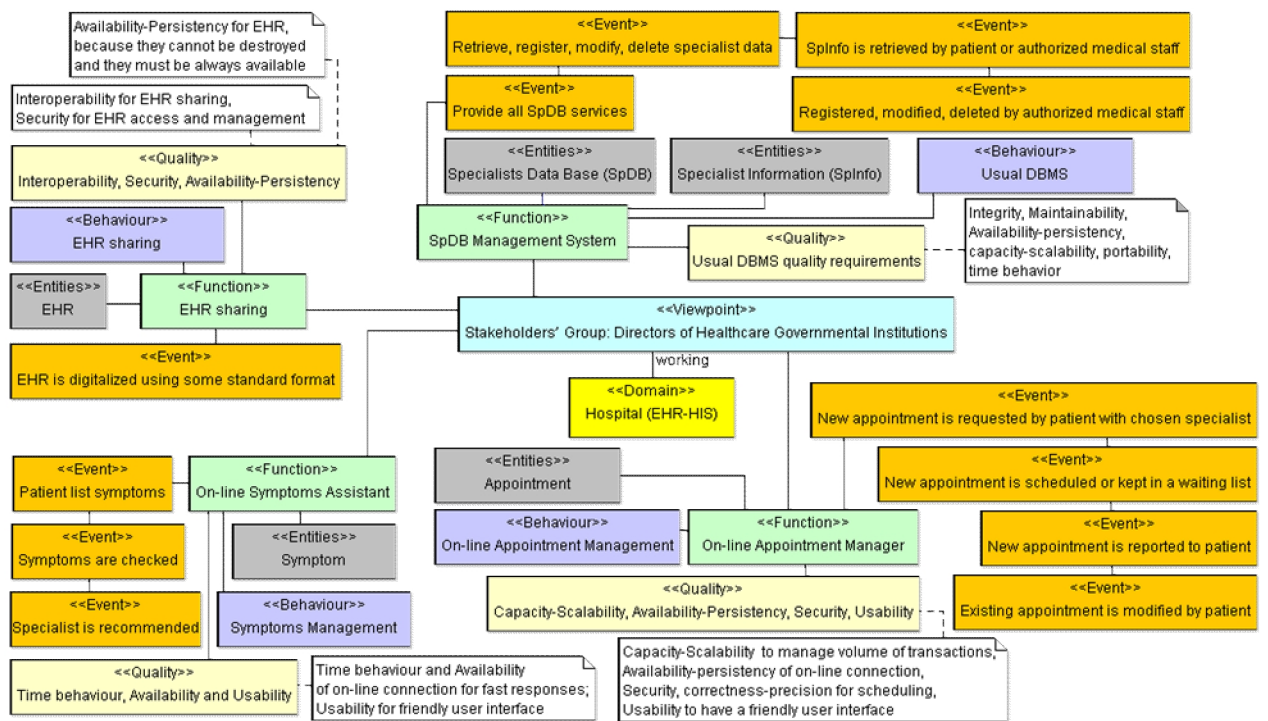


Figura 29. Procesos de negocios extraídos desde el punto de vista de los Ingenieros de Dominio para la Faceta Tecnologías Soportadas (BPMN). Fuente: [HLO16b]

**Tabla 39.** Especificación por intrínsecas del punto de vista de los Directores de HGI para la Faceta Reglas & Regulaciones.

<b>Dominio EHR-SIS</b>				
Punto de Vista del Grupo de Stakeholders: Directores de Instituciones Gubernamental de Salud (HGI)				
Faceta Reglas & Regulaciones				
<i>Entidades</i>	<i>Funciones</i>	<i>Eventos</i>	<i>Comportamiento</i>	<i>Calidad</i>
EHR	Compartir EHR	EHR es digitalizado usando algún formato estándar	Registros EHR son compartidos ( <i>Compartir EHR</i> )	- <i>Interoperabilidad</i> para compartir el EHR, <i>Seguridad</i> para el acceso y la gestión del EHR, <i>Disponibilidad-Persistencia</i> de la EHR, que no puede ser destruida y ellos deben estar siempre disponibles
- Base de Datos de Especialistas (SpDB), -SpInfo: Información de Especialistas	Gestión del Sistema SpDB (SpDBMS)	Proporcionar todos los servicios SpDBMS; recuperar, registrar, modificar, borrar datos de especialistas. SpInfo es recuperada por el paciente o por el personal médico autorizado; este es registrado, modificado, eliminado por el personal médico autorizado	Actividades habituales en un SGBD ( <i>Usual DBMS</i> )	Requisitos de calidad habituales en un SGBD: <i>Integridad, Mantenibilidad, Disponibilidad-Persistencia, Capacidad-Escalabilidad, Adaptabilidad, Comportamiento en el tiempo</i>
Síntoma	Asistencia para Síntomas en Línea	- Lista de síntomas de pacientes - Los síntomas son comprobados - Un especialista es recomendado	- Los síntomas son listados en línea por el paciente y son comprobados automáticamente en catálogos médicos en línea - Un especialista es recomendado en línea para el paciente, si se detectan síntomas, en caso contrario, el paciente debe volver a escribir en línea una nueva lista de síntomas ( <i>Gestión de Síntomas</i> )	- <i>Comportamiento en el tiempo</i> y - <i>Disponibilidad</i> de la conexión en línea para respuestas rápidas - <i>Usabilidad</i> para una interfaz de usuario amigable y fácil de usar
Cita	Gestionar Citas en Línea	- Nueva cita es solicitada por el paciente con el especialista elegido - Nueva cita es programada o mantenerse en una lista de espera - Es informada al paciente - Cita existente es modificada por el paciente	- Proporcionar servicios de citas precisas con el especialista; manejar el volumen de solicitudes, proporcionar un acceso seguro a los servicios de citas; contar con instalaciones de conexión a la red; tener una interfaz fácil de usar - La cita con el especialista, es solicitada, programada, reportada, eliminada o modificada en línea ( <i>Gestión de citas en línea</i> )	- <i>Capacidad-Escalabilidad</i> para gestionar el volumen de transacciones, <i>Disponibilidad-Persistencia</i> de la conexión en línea, <i>Seguridad, Correctitud-Precisión</i> para la programación de citas, <i>Usabilidad</i> para tener una interfaz fácil de usar y amigable



**Figura 30.** Punto de vista de los Directores IGS para la Faceta Reglas & Regulaciones (UML). Fuente: [HLO16b]

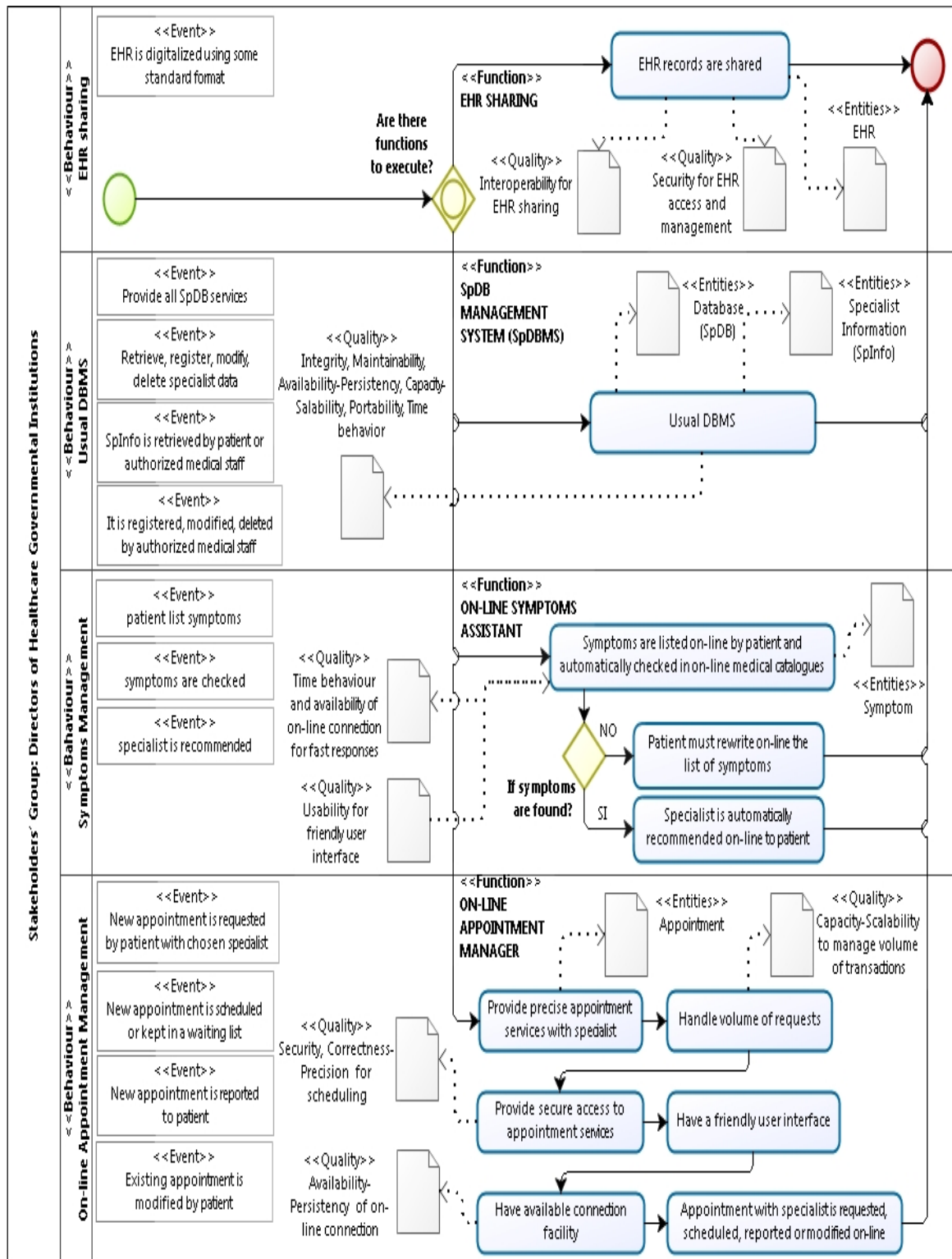


Figura 31. Procesos de negocios extraídos desde el punto de vista de los Directores de IGS para la Faceta Reglas & Regulaciones (BPMN). Fuente: [HLO16b]

#### *2.4 Modelado del Dominio*

La representación BPMN del Modelo del Dominio de la Figura 32 muestra agrupaciones “Aggregations” de los procesos de negocio identificados para los tres diferentes grupos de puntos de vistas para las facetas de Procesos de Negocio, Soporte Tecnológico y Reglas y Regulaciones. Esta información contiene los potenciales nuevos componentes a ser incluidos en la AC; el MCE actualizado (MCEA) en la Tabla 40, especifica estos nuevos componentes, seleccionados por el ingeniero de dominio o experto del dominio, que serán añadidos a la AC, según las consideraciones sobre los nuevos productos establecidos en el paso 1.3 en el Documento Productos Futuros.

La AC Completada (ACA), ver Figura 33 y el MCE, consulte la Tabla 32, son utilizados para construir el MCEA, ver Tabla 40, donde también se muestran los puntos de variación agrupando nuevas variantes; para restringir la presentación; realmente serán conformados en el paso 3.7 de la IRD. Los nombres utilizados en el MCEA para cada componente, reflejan similitudes funcionales semánticas encontradas en las agregaciones de los procesos estudiados para cada punto de vista de las partes interesadas; recordar también que no se consideran procesos no automáticas que no implican interacción con entidades digitales, por ejemplo, la entrada del paciente en el hospital.

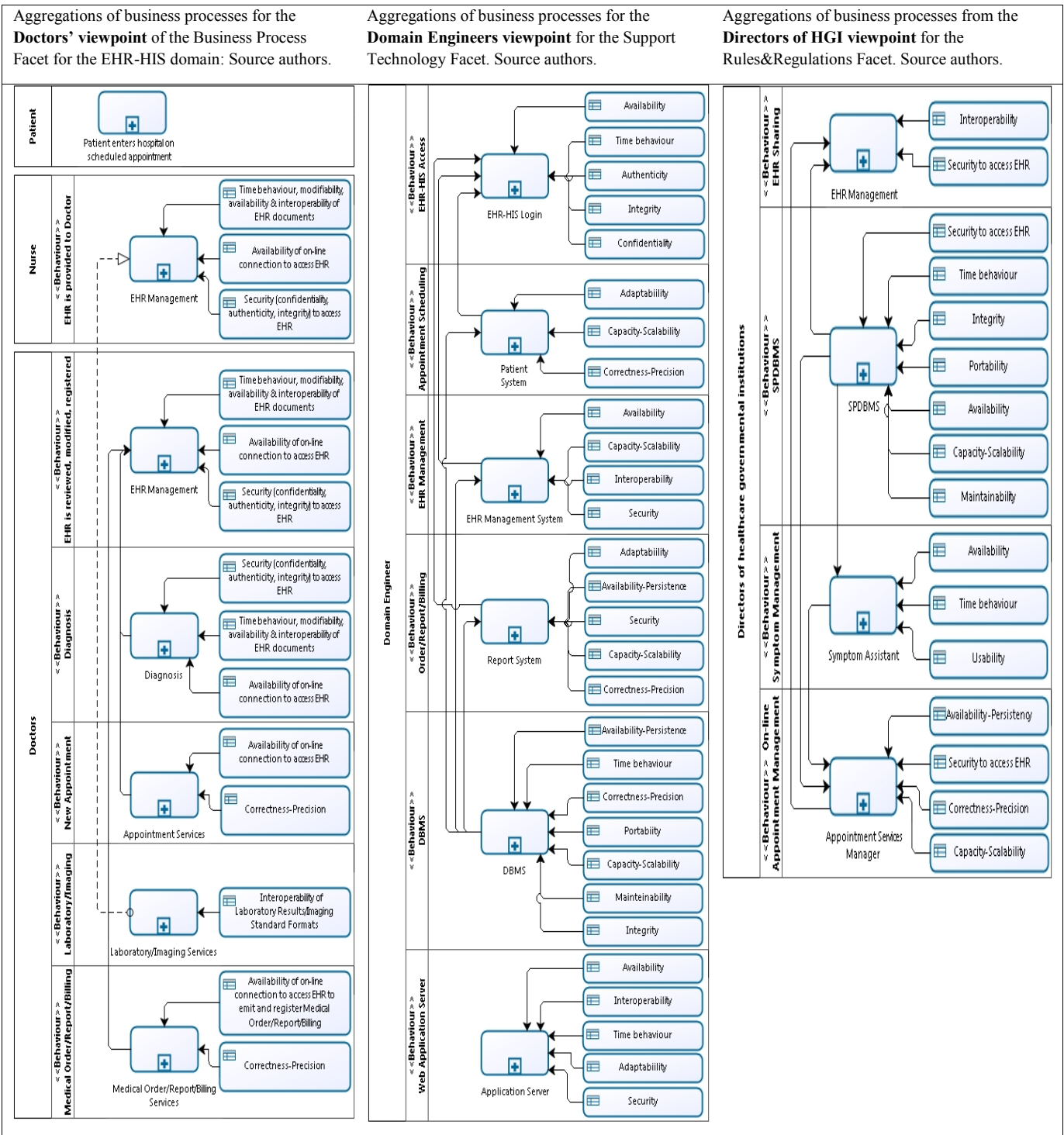


Figura 32. Agregación de los procesos de negocios para las facetas estudiadas – Modelo del Dominio (BPMNAgg). Fuente: [HLO16b]

Las modificaciones del MCE son:

- CC b1-Patient ahora está dedicado solamente a la atención de pacientes, incluyendo los sub-componentes:
  - b18-AppointService, encontrado en todos puntos de vista examinados;
  - b19-SymptomAssistant sólo se encontró en el punto de vista de los Directores de HGI para obtener servicios de cita en línea; sin embargo, este mecanismo puede también ser usado por los médicos para el diagnóstico;
  - b20-DemographicData también se menciona en todos los puntos de vista;
- CC b2- Medical Practice ahora incluye funciones directamente relacionadas con la práctica médica, con nuevos sub-componentes:
  - b12-EHRManager para la gestión del EHR, cambia el nombre anterior a b2-EHR Mang;
  - b13-Diagnosis para asistencia de los diagnósticos;
  - b14 – Orders&Reports para la emisión de las recetas médicas y gestión de informes;
- CC b3-AdminSystem cambia el nombre de b3-Reports incluyendo sólo los servicios administrativos;
- CC b15-LabImageSystem es una nueva funcionalidad para laboratorio y proyección de imagen y gestión de resultados;
- CC a6-Lab&Imaging es un nuevo botón que se agregó para acceder a la nueva funcionalidad b15-LabImageSystem.

Nótese que *cl. DB* en la AC inicial incluye todas las bases de datos; en la AC completada las principales bases de datos utilizadas se muestran como subcomponentes, pero no serán tratadas por separado. Las semánticas de los nuevos componentes serán proporcionadas en la Tabla 40.

### *3. Alcance de los Activos*

En el paso Aplicación del Alcance de los Activos, el Repositorio de Activos Centrales (RAC) está conformado por MCD, CCT, AC, ACA, MCE, MCEA, BPMNs, y BPMNAggs, que son los resultados “outputs” básicos de las sub-fases 1 y 2; sin embargo, la estructura del RAC tiene que estar correctamente diseñada, lo que está fuera del alcance de esta investigación.



**Tabla 40.** Tabla MCE Actualizada (MCEA): Vista General de la AC completada (ACA); los cambios son resaltados en gris.

Componentes Comunes (CC)	Componente Variante <<variant>>	Descripción componente	Propiedades de calidad componente – Prioridad se muestra en (); 1 alta, 2 media, 3 baja		Restricciones
			Requerida (s)	Proporcionada (s)	
a2. Patient Portal a3. Reports		- botón para acceder las funcionalidades comunes b1, b2 del HCE-SIS en la Capa de Procesos - botón para acceder la funcionalidad b3	Ver Tabla 32	- los componentes anteriores también proporcionan la calidad para a2, a3	a2, a3 son obligatorias CC FR; ellos son IU de botones, son parte de a1 y a4.
	a1. WebPage a4. GUI	- Interfaz de usuario (IU) de páginas Web en un navegador - IU Stand-alone	ver Tabla 32 ver Tabla 32	ver Tabla 32 ver Tabla 32	ver Tabla 32 ver Tabla 32
a6. Lab&Imagen		- Botón de IU para acceder a la funcionalidad b15	- las mismas propiedades de calidad especificadas en la Tabla 32 para a2, a3 se mantienen	- los mismos componentes especificados en la Tabla 32 para a2, a3 se mantienen	a6 es obligatoria CC FR; esta es parte de a1 y a4.
b1-Patient  - b18- AppointmentService		- programación de citas en el sitio y/o en línea	- Correctitud-Precisión (2), - Disponibilidad-Persistencia (1),  - Autenticidad (1), - Confidencialidad (1), - Integridad (1) - Adaptabilidad (2) - Capacidad-Escalabilidad (2)	- b6  - c7, c9a, c9b - b7 - c8 - c5, c6 - c4	b1 es obligatoria CC FR; b18, b19 y b20 son sub-componentes de b1 y ellos también son obligatorios CC; c8 es un mecanismo adicional para la integridad; c5, c6 son alternativas para la portabilidad de base de datos comerciales a diferentes plataformas
	- b19- SymptomAssistan t	- Asistencia a la búsqueda de sintomatología para encontrar especialista médico	Las propiedades anteriores son válidas para sub-componentes b19, b20	Los componentes anteriores son válidos para subcomponentes b19, b20	
	- b20- DemographicDat a	- Recuperación, almacenamiento, modificación, eliminación de los datos personales y demográficos del paciente, de laboratorio / imágenes			
b2- MedicalPractice  - b12- EHRManager		- Recuperación, creación, almacenamiento, modificación, compartición del EHR	- Interoperabilidad (1), - Disponibilidad-Persistencia, - Autenticidad, - Confidencialidad, - Integridad	- b4 y/o c2 o c3 - c7, c9a, c9b  - b7-HTTP, b7-HTTPS, HTTPS - c8	b2 es un RF CC obligatorio; b12, b13 y b14 son subcomponentes de b2 y también son CC obligatorios - Interoperabilidad es un RNF SIS obligatorio resuelto por b4; C2 o C3 son soluciones opcionales

- <i>b13-Diagnosis</i>		- Recuperación de catálogos médicos, protocolos; búsqueda de sintomatología	- Disponibilidad-Persistencia, - Autenticidad, - Confidencialidad, - Integridad - Comportamiento en el tiempo (2)	- c7, c9a, c9b  - b7-HTTP, b7-HTTPS, HTTPS - c8 - protocolos en la Capa de Transmisión	para la Interoperabilidad en la Capa de Datos;
- <i>b14-Orders&amp;Reports</i>		- Servicios de emisión, recuperación, modificación y almacenamiento de órdenes e informes médicos	- Autenticidad, - Confidencialidad, - Integridad, - Disponibilidad-Persistencia - Capacidad-Escalabilidad,	- b7-HTTP, b7-HTTPS, HTTPS - c8 - c7, c9a, c9b  - c4	
<i>b3-AdminSystem</i>		- servicios administrativos básicos: facturas, facturación, informes estadísticos	- Autenticidad, - Confidencialidad, - Integridad, - Disponibilidad-Persistencia, - Correctitud-Precisión	- b7-HTTP, b7-HTTPS, HTTPS - c8 - c7, c9a, c9b  - b6	b3 es un RF CC obligatorio;
<i>b-15 LabImageSystem</i>		- recuperación, almacenamiento, modificación de estudios de laboratorio y de imágenes radio gráficas	- Interoperabilidad - Disponibilidad-Persistencia, - Autenticidad, - Confidencialidad, - Integridad, - Correctitud-Precisión	- b16, b17 - c7, c9a, c9b  - b7-HTTP, b7-HTTPS, HTTPS - c8 - b6	b15 es un RF CC obligatorio
	<i>b16-LOINC</i> <<variant>>	- Base de datos incluyendo un estándar para la identificación de las observaciones de laboratorio médico	- Interoperabilidad	- <i>base de datos (repositorio)</i>	RNF obligatorio para b15; b16, b17; ambos deben estar presentes en una configuración
	<i>b17-DICOM</i> <<variant>>	- Mecanismo para la conversión a formatos gráficos de imágenes estándar; conformada por un archivo de formato y un protocolo común; los archivos se intercambian a través de TCP/IP	- Interoperabilidad	- <i>Engine de traducción del mercado</i>	
	<i>b6-Algor*</i> (*) <i>multiplicidad</i>	- Algoritmos para cálculos	- Correctitud-Precisión	- <i>modules</i>	RNF obligatorio para b1, b3 y b15
	<i>b7. Data Security</i>	- Servicios o algoritmos para la identificación del usuario (Autenticidad) y políticas de derechos de acceso del usuario (Confidencialidad) de la información médica	- Autenticidad, - Confidencialidad, - Integridad	- <i>b7-HTTP</i> - <i>HTTPS</i> - <i>b7-HTTPS</i>	- RNF SIS obligatorio; alternativas que combinan algoritmos de seguridad en b7 Capa de Procesos con los protocolos de seguridad HTTP/HTTPS en la Capa de Transmisión; La integridad también es tratada en la Capa de Datos; sólo una alternativa se sostiene
	<i>b4. Mirth</i>	- Servicios de traducción	- Interoperabilidad	- engine de	b4 es obligatorio para

		para EHR al formato estándar HL7		traducción del marcado	resolver el RNF SIS de interoperabilidad
	b5. GUIServerSide <<variant>>	- soporta la comunicación local y remota cliente/servidor	- Comportamiento en el tiempo - Modificabilidad (3) - Modularidad (3)	- protocolos - módulos	- TCP/IP, proxy o gateway - patrón MVC
c1-DB - c16-EHRDB - c17-LABIMDB - c18-SPDB - c19-CATDB		- servicios de bases de datos a través de los SGBD habituales	Lo mismo que fue especificado en la Tabla 32 para c1-DB	Lo mismo que fue especificado en la Tabla 32 para c1-DB	c1 es un RF CC obligatorio, entonces, todos los subcomponentes son también CC obligatorio; en este estudio sólo se considera c1
	c4-NewMed Stds *	- Adición de nuevos estándares para la práctica médica: formatos, catálogos, protocolos médicos, etc.	- Capacidad-Escalabilidad	- módulos	RNF opcional
	c8-IntegMech *	- Proporciona integridad de los datos a las bases de datos	- Integridad (c8)	- mecanismos	RNF obligatorio para c1-DB
	c9a-Mirror c9b. MirrorReplc	- Proporciona copias de seguridad de bases de datos	- Disponibilidad-Persistencia	- mecanismos	RNF opcional para c1-DB; solamente una alternativa es válida
	c7- Hibernate	- Proporciona persistencia de datos a las bases de datos	- Disponibilidad-Persistencia	- engine	RNF obligatorio para c1-DB
	c5-JDBC c6-ODBCβ	- Portabilidad a diferentes plataformas de bases de datos comerciales	- Adaptabilidad	- APIs	RNF opcional para c1-DB
	c2-HL7Eng c3-HXPeng	- Interoperabilidad con los formatos estándares de HL7	- Interoperabilidad	- engines	RNF opcional para c1-DB; al menos b4 debe estar presente
d1. Internet	d2. Satellite	- Infraestructura de red, permiten la comunicación entre las capas	- Comportamiento en Tiempo, - Disponibilidad-Persistencia, - Adaptabilidad, - Capacidad-Escalabilidad, - Autenticidad, - Confidencialidad, - Integridad	- protocolos HTTP/HTTPS	d1 es común

## II. Fase Ingeniería de Requisitos del Dominio (IRD)

El resultado para la Fase II Aplicación de la Ingeniería de Requisitos del Dominio (IRD) es la ACA, mostrada en la Figura 33.

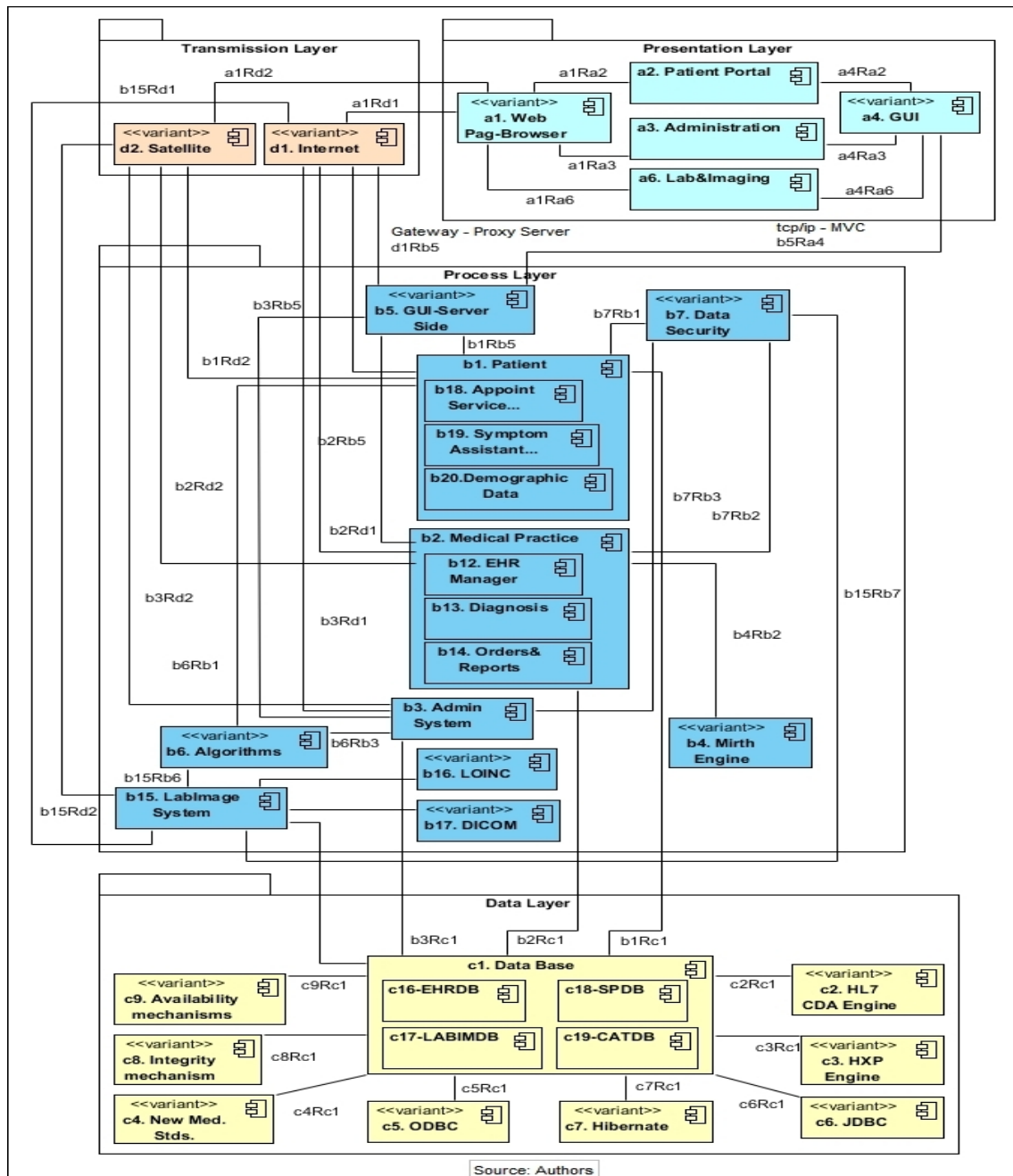


Figura 33. AC Completa (ACA). Fuente: [HLO16b]

### III. Fase Aplicación del Diseño del Dominio (DD)

El resultado para la Fase III Aplicación del Diseño del Dominio (DD) es la AR, mostrada en la Figura 34. Los componentes de la AR y la AR como artefacto son también añadidos al RAC.

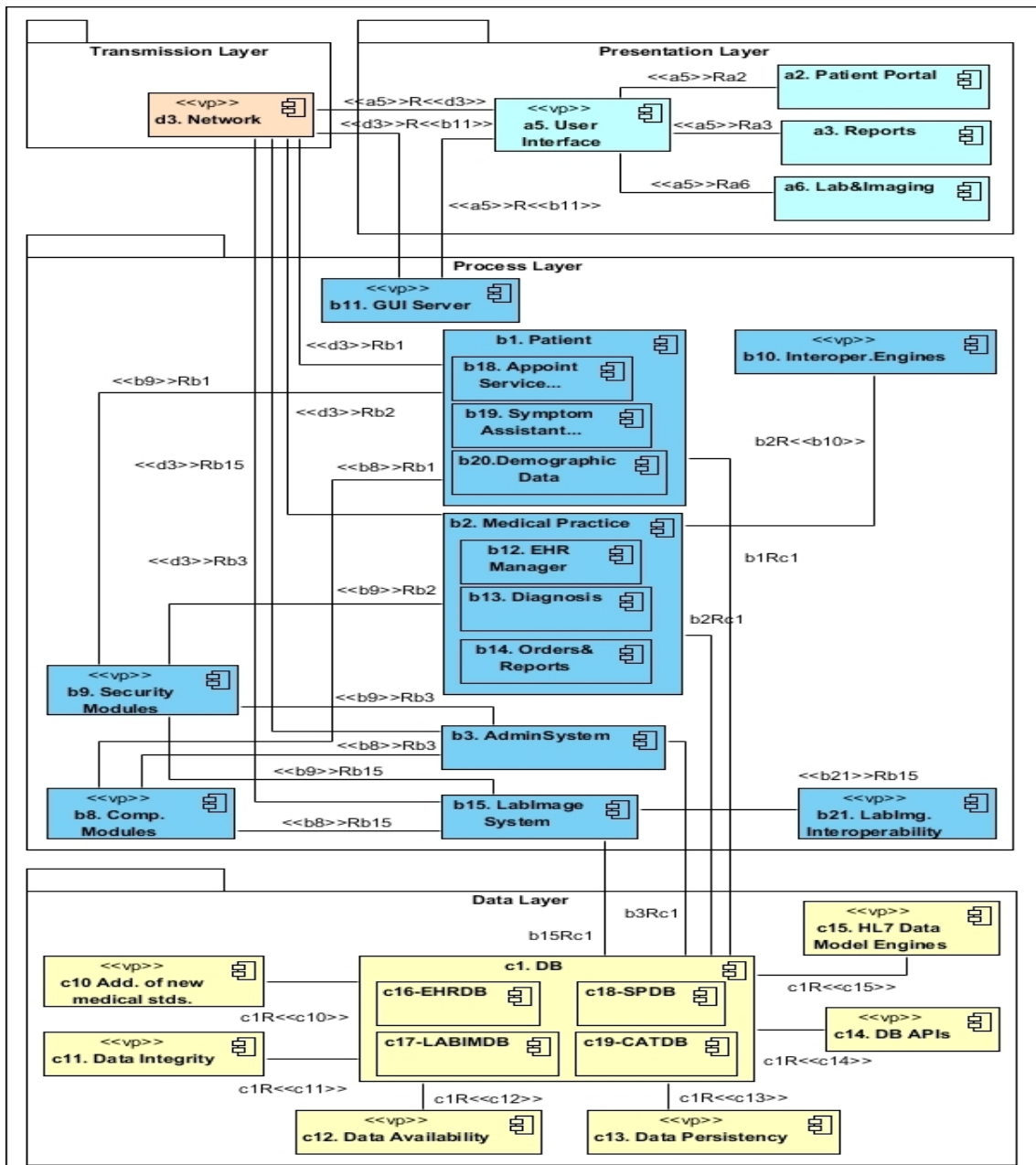


Figura 34. AR. Fuente: [HLO16b]

A continuación, se presenta la documentación de la AR, que es la base para la derivación de productos concretos a partir de la AR.

### 6.2.2 Documentación de la AR

La documentación de la AR (DOC-RA) constituye una entrada a la derivación de productos concretos a partir de AR, en el ciclo de la IA.

DOC-RA (ver Tabla 41), muestra la lista de componentes comunes, puntos de variación y sus correspondientes conectores, la terminología se ha colocado en inglés y español para facilitar la legibilidad; los puntos de variación de la AR son: <<a5>> *User Interface* = {a1, a4}, en la capa de presentación; <<d3>> *Network* = {d1, d2}, en la capa de transmisión, que incluye la conexión al servidor web y SSL con protocolos de seguridad de redes específicos; en la capa de proceso <<b10>> *Interoper. Engines* = {b4}, que indica la disponibilidad de mecanismos comerciales para la interoperabilidad; <<b11>> *GUI Server* = {b5} que incluye a la variante b5. *GUI Server-side*; <<b21>> *Labimg.Interoperability* = {b16, b17} para la interoperabilidad de las imágenes de laboratorio; en la capa de datos se tiene a <<c15>> *HL7 Data Model Engines* = {c2, c3} que indica diferentes mecanismos para HL7 CDA o modelos similares; otros puntos de variación son <<b8>> *Comp. Modules* = {b6} para manejar algoritmos computacionales para la precisión en los resultados de las consultas de pacientes y en los reportes, y <<b9>> *Security Modules* = {b7} para manejar aspectos de autenticidad y confidencialidad, junto con los protocolos de red HTTP/HTTPS.

Adicionalmente, la capa de datos contiene los puntos de variación <<c14>> *DB APIs* = {c5, c6}, para portabilidad a objetos, empleando diferentes tipos de bases de datos relacionales comerciales (MySQL, PostgreSQL, Oracle), <<c10>> *Add. of new medical stds.* = {c4} para la evolución en el tiempo; y finalmente los puntos de variación <<c11>> *Data Integrity* = {c8}, <<c12>> *Data Availability* = {c9}, y <<c13>> *Data Persistency* = {c7}.

Debe notarse que los <<vp>> que tienen una sola variante pueden ser extendidos con otras variantes, dependiendo de los componentes tecnológicos disponibles en el mercado. La Tabla DOC-RA se utiliza para la derivación de

productos en el ciclo IA. A continuación, se presenta un ejemplo de derivación de un producto para mostrar la utilidad de la AR.

**Tabla 41.** DOC-AR: Documentación de la AR

Componentes/Conectores de AR	Especificación de Componentes y Conectores de AR	Comentarios/Restricciones	
<b>a. Capa de presentación</b> Componentes <<a5>> User Interface = {a1, a4}	a1. Web pages – Browser	- Control de acceso de usuarios e interacción con las funcionalidades principales de SIS y/o con el servidor Web	
	a4. GUI	- Cliente GUI basada en MVC Note que a1 y a4 no pueden ejecutarse juntos; una de ellos es obligatorio	
a2, a3, a6	a2. Patient Portal, a3. Administration, a6. Lab&Imaging	- Componentes comunes obligatorias para acceder a las funcionalidades principales del SIS	
Conectores <<a5>>R<<d3>> <<a5>>Ra2 <<a5>>Ra3 <<a5>>Ra6 <<a5>>R<<b11>>	{a1Rd1, a1Rd2}; {a1Ra2, a4Ra2}; {a1Ra3, a4Ra3}; {a1Ra6, a4Ra6}; {a4Rb5};	Ejemplo de un desarrollo: <<a5>>R<<d3>> = {a1Rd1, a1Rd2}, {a4Rd1, a4Rd2} = {a1Rd1, a1Rd2}, porque los conectores {a4Rd1, a4Rd2} no existen ya que a4 se conecta solo a d1 y lo hace vía b5;	
<b>b. Capa de Proceso</b> Componentes <<b11>> GUI Server = {b5}	b5. GUI Server-side	- Acceso a funcionalidades en la Capa de Proceso y a servicios remotos de Internet	
	<<b8>> Computation modules = {b6}	- Los cálculos necesarios para obtener resultados precisos - Servicios o algoritmos para manejar tarjetas, contraseña, biometría, etc. Dispositivos de autenticación de usuario y políticas de acceso de usuario para la confidencialidad de la información médica	
	<<b9>> Security modules = {b7}		
	b1 - b18. Appoint. Service - b19. Symptom Assistant... - b20. Demographic Data b2 - b12. EHR Manager - b13. Diagnosis - b14. Orders&Reports b3 b15	b1. Patient, b2. Medical Practice, b3. Admin System, b15. LabImage System	- Componentes comunes obligatorias, principales funcionalidades del SIS
	<<b10>> Interoper. Engines = {b4}	b4. Mirth Engine	- Motor HL7 para interoperabilidad de HCE en Capa de Proceso
	<<b21>> LabImg.Interoperability = {b16, b17}	b16. LOINC b17. DICOM	- Mecanismos para la interoperabilidad de las imágenes de laboratorio
	Conectores b1R<<d3>> b2R<<d3>> b3R<<d3>> b15R<<d3>> b3R<<b8>> b1R<<b8>> b15R<<b8>> b1R<<b9>> b2R<<b9>> b3R<<b9>> b15R<<b9>> b15R<<b21>>	{b1Rd1, b1Rd2}; {b2Rd1, b2Rd2}; {b3Rd1, b3Rd2}; {b15Rd1, b15Rd2}; {b3Rb6}; {b1Rb6}; {b15Rb6} {b1Rb7}; {b2Rb7}; {b3Rb7}; {b15Rb7}; {b15Rb16, b15Rb17};	- b1Rd1, b2Rd1, b3Rd1, b15Rd1, (b1Rb5), {b2Rb5}, (b3Rb5} y (b5Rd1} son conectores variantes, ya que están ausentes en algunos de los productos

<i>b2R</i> << <i>b10</i> >> <i>b1R</i> << <i>b11</i> >>; <i>b2R</i> << <i>b11</i> >>; <i>b3R</i> << <i>b11</i> >>; << <i>b11</i> >> <i>R</i> << <i>d3</i> >>; <i>b1Rc1</i> ; <i>b2Rc1</i> ; <i>b3Rc3</i> ; <i>b15Rc1</i> ;	<i>(b2Rb4)</i> ; <i>(b1Rb5)</i> ; <i>(b2Rb5)</i> ; <i>(b3Rb5)</i> ; <i>(b5Rd1)</i> ; <i>{b1Rc1}</i> ; <i>{b2Rc1}</i> ; <i>{b3Rc1}</i> ; <i>{b15Rc1}</i> ;	- <i>d2</i> no puede ser utilizado con <i>b5</i> porque el producto PatientOS no tiene esa opción
<i>c. Capa de Datos</i> Componentes << <i>c10</i> >> <i>Addition of new medical standards</i> = <i>{c4}</i>	<i>c4. New medical standards</i>	- Adición de nuevos estándares médicos (catálogos de diagnóstico, etc.) para la evolución del sistema
<< <i>c11</i> >> <i>Data Integrity</i> = <i>{c8}</i>	<i>c8. Integrity mechanisms</i>	- Soluciones de integridad de datos, APIs específicas para cada Base de Datos (BD)
<< <i>c12</i> >> <i>Data Availability</i> = <i>{c9}</i>	<i>c9. Availability mechanisms</i>	- Mecanismos de la disponibilidad de datos
<< <i>c13</i> >> <i>Data Persistency</i> = <i>{c7}</i>	<i>c7. Hibernate</i>	- Mecanismos de persistencia de datos para la plataforma Java
<< <i>c14</i> >> <i>DBAPIs</i> = <i>{c5, c6}</i>	<i>c5. ODBC</i>	- Mecanismos de portabilidad a objetos de bases de datos relacionales
	<i>c6. JDBC</i>	- Mecanismos de portabilidad a objetos Java de bases de datos relacionales - <i>c5</i> o <i>c6</i> son obligatorios
<< <i>c15</i> >> <i>HL7 Data Model Engine</i> = <i>{c2, c3}</i>	<i>c2. HL7 CDA Engine</i> <i>c3. HXP Engine</i>	- se asegura la Interoperabilidad con el estándar HL7 en Capa de datos - <i>c2</i> o <i>c3</i> deben ser obligatorios si se realiza esta propiedad en Capa de datos
<i>c1. DB</i> - <i>c16-EHRDB</i> - <i>c17-LABIMDB</i> - <i>c18-SPDB</i> - <i>c19-CATDB</i>	<i>c1. Data Base</i>	- Bases de datos comerciales; <i>c1</i> es obligatorio
Conectores <i>c1R</i> << <i>c10</i> >>; <i>c1R</i> << <i>c11</i> >>; <i>c1R</i> << <i>c12</i> >>; <i>c1R</i> << <i>c13</i> >>; <i>c1R</i> << <i>c14</i> >>; <i>c1R</i> << <i>c15</i> >>;	<i>{c1Rc4}</i> ; <i>{c1Rc8}</i> ; <i>{c1Rc9}</i> ; <i>{c1Rc7}</i> ; <i>{c1Rc5, c1Rc6}</i> ; <i>{c1Rc2, c1Rc3}</i>	- <i>c7</i> , <i>c8</i> , <i>c9</i> mecanismos relativos a satisfacer propiedades de calidad inherentes a SMBD
<i>d. Capa de Transmisión</i> Componentes << <i>d3</i> >> <i>Network</i> = <i>{d1, d2}</i>	<i>d1. Internet</i> <i>d2. Satellite</i>	- Solicitud/respuesta para tener servicios eficientes, adecuados, disponibles, seguros e interoperables a través de protocolos de Internet.  Note que <i>d1</i> es un componente obligatorio, sin embargo, se ha incluido como variante; <i>d2</i> es una variante opcional y comparten el mismo punto de variación << <i>d3</i> >> <i>Network</i> . <i>d1</i> es obligatorio, <i>d2</i> es opcional pero no puede ser utilizado con <i>b5</i> ; Los protocolos HTTP o HTTPS son opcionales para <i>d1</i> y <i>d2</i> ; La presencia de HTTP es imposible sin <i>b7</i> , debido a los requisitos de SIS y hay que complementar con <i>b7</i>

### 6.2.3 Derivación de un Producto SIS a partir de la AR

La derivación de un producto a partir de la AR consiste en la instanciación de sus puntos de variabilidad, manteniendo siempre los componentes comunes, considerando también la selección apropiada de conectores (ver Figura 35 y Tabla 41).



De acuerdo a los requisitos del cliente se requiere un producto SIS con las siguientes características: a) que sea de bajo costo; b) que tenga una interfaz de páginas Web; c) que permita el control de los pacientes, sus historias médicas, ordenes y reportes médicos, y una gestión administrativa básica; d) que sea flexible para incorporar nuevos estándares médicos en catálogos, etc.; e) que los datos sean portables entre base de datos que operen sobre la plataforma Java.

Los puntos de variación de la AR para SIS a ser instanciados para la derivación de este producto, considerando los requisitos del cliente anteriores y las propiedades de calidad requeridas/proporcionadas por los componentes especificados en la tabla DOC-AR (Tabla 41) son:

Capa de presentación:	<<a5>> <i>User interface = {a1, a4}</i> => a1 por requerir la interfaz de Páginas Web. Donde => indica instanciación
Capa de procesos:	<<b8>> <i>Comp. Modules = {b6}</i> la propiedad precisión no es demandada por el cliente pero es requerida por b1 y b3 para la corrección de los resultados (ver AC, Figura 33 y Tabla XX. <<b9>> <i>Security Modules = {b7}</i> , la propiedad seguridad no es demandada por el cliente pero es requerida por b2 para acceso controlado a personal de salud autorizado
Capa de datos:	<<b10>> <i>Interoper. Engines = {b4}</i> <<c15>> <i>HL7 Data Model Engines = {c2, c3}</i> , la propiedad de interoperabilidad no es demandada por el cliente pero es requerida por b2 para compartir información médica entre diferentes centros de salud <<c10>> <i>Add. of new medical stds. = {c4}</i> para satisfacer escalabilidad <<c12>> <i>Data Availability = {c9}</i> por ser propiedad de calidad prioritaria de SIS <<c13>> <i>Data Persistency = {c7}</i> por ser propiedad de calidad prioritaria de SIS <<c14>> <i>DB APIs {c5, c6}</i> => {c6} para satisfacer portabilidad a objetos Java
Capa de transmisión:	<<d3>> <i>Network = {d1, d2}</i> => d1 por requerir Internet

El *arquitecto de software o ingeniero de dominio* identificó los siguientes requisitos que corresponden a componentes funcionales como variantes:

- a1. *interfaz de página Web* para acceder al portal de pacientes para el control de pacientes, historias médicas y reportes médicos, estadísticas y gestión administrativa en b1, b2, b3 respectivamente.

También identificó los siguientes requisitos no funcionales y/o de calidad implícitos:

- para portabilidad de la base de datos sobre plataforma Java selecciona *c5. JDBC*;
- para seguridad de los datos y comunicación selecciona *b7* y con protocolos de comunicación en *d1*, *b7+HTTP*, *b7+HTTPS* y *HTTPS* para aumentar el nivel de seguridad;
- para precisión, toma *b6. Algorithms* donde se considerará un módulo específico de algoritmos a ser determinado;
- para interoperabilidad, considera *c2. HL7 CDA Engine*;
- para disponibilidad-persistencia, selecciona *c7. Hibernate* y *c9. Availability mech.*
- para la escalabilidad, por tener nuevos estándares médicos en catálogos, etc. toma *c4.New Med. Stds.*

Además debe notarse que por requerir el cliente un producto de bajo costo, se decidió realizar la interoperabilidad de HCE en la capa de datos con el componente *c2*, que resultó ser menos costosa en tiempo y recursos, que agregar un componente a nivel de capa de proceso.

De esta forma se establece la configuración arquitectónica de un nuevo producto SIS, incluyendo automáticamente el núcleo de componentes comunes y variantes; la Figura 35 muestra esta configuración.

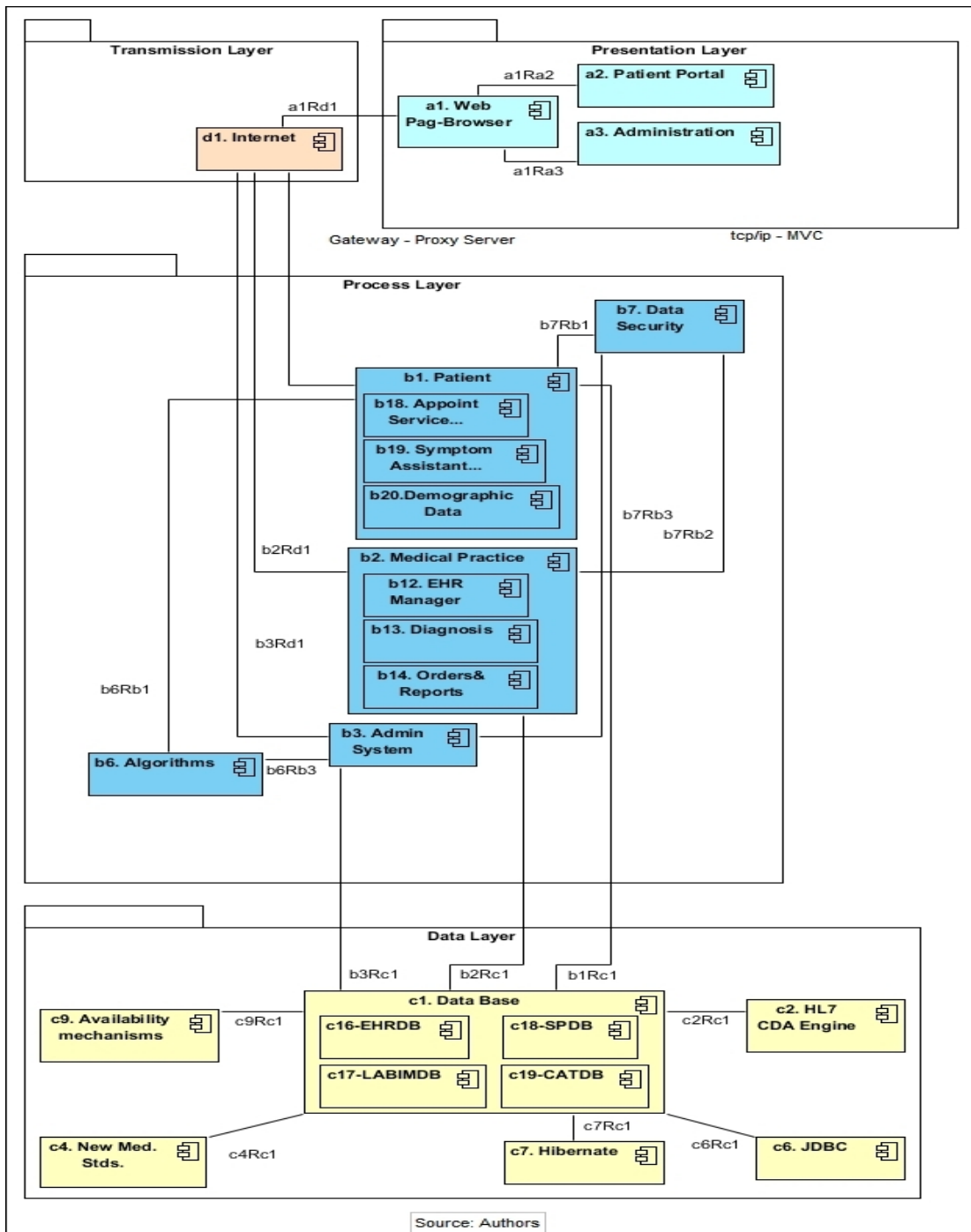


Figura 35. Derivación de un Producto a partir de la AR.

### 6.3 Aplicación paso a paso del proceso WSRA-SPL

Las Líneas de Productos de Software (LPS) y la Arquitectura Orientada a Servicios (SOA) son enfoques para el desarrollo de software utilizado en la práctica industrial favoreciendo la reutilización de activos (recursos) y capacidades existentes, en lugar de re-desarrollar la construcción de nuevos sistemas. El objetivo principal de este trabajo consiste en explotar los beneficios de ambos enfoques e integrarlos en un proceso de diseño arquitectónico único, Arquitectura de Referencia de Servicios Web para LPS (WSRA-SPL), centrada en el aseguramiento del software en etapas tempranas. En esta sección se aborda la aplicación de WSRA-SPL para el dominio de los Sistemas de Información Integrados de Salud (SIS) “*HIS Domain*”. En general, se reutilizan todos los artefactos desarrollados en el proceso QuaDRA para llegar a la AR. Lo que se hace es utilizar SOMA para agrupar los servicios en servicios simples o compuestos, de acuerdo a los objetivos del negocio (BG) que se extraen de las diferentes vistas agrupadas en el BPMNAgg (ver Figura 32). Algunas propiedades de calidad requeridas por las funcionalidades podrían ser modificadas respecto a las especificadas en QuaDRA, de ser el caso, pero las propiedades de calidad globales del estilo eventos/capas con modelo de comunicación para transmisión y distribución se mantiene.

#### 6.3.1 Aplicación del Proceso de Diseño WSRA-SPL para el Dominio de los SIS

WSRA-SPL será aplicado paso a paso a continuación:

##### ***I. Fase Alcance de la LP (PLScoP)***

*1. Alcance del Portafolio de Productos - para determinar los requisitos para futuros productos de la LPS:*

*1.1 Estudio de los productos existentes del mercado*

*1.1.1 SOMA Análisis de los sistemas existentes*

- La AC (ver Figura 25, *Sección 6.2.1*) y la CCT (ver Tabla 31, *Sección 6.2.1*) son construidas en la aplicación del proceso QuaDRA mostrado en la *Sección 6.2*.

Componentes comunes de la AC (funcionalidades principales) (CC), son: a1, a2, a3, d1; componentes variantes que no están presentes en todos los productos son: a4, b4, b5, c2, c3, d2. Ellos son todos servicios (simples y/o compuestos) potenciales, extraídos de los comportamientos o funciones (BG) de los procesos de negocios (BP), y conforman el MCE, y se hacen corresponder a los componentes funcionales y/o no funcionales en AC:

### 1.1.2 Modelado de Objetivos-Servicios SOMA: para construir el Portafolio de Servicios inicial

- El MCD (Figura 25) y el MCE (Tabla 31) con servicios son construidos:

El MCE (Portafolio de Productos) ahora con servicios, es obtenido desde el MCD, AC, y CCT; MCE será actualizada o transformada incluyendo servicios en el Paso 2.4.2 y será elaborada en paralelo con los Pasos 1.2 y 1.3.

- Los Servicios Candidatos (SC) son identificados:

Los SC son identificados a partir de las funcionalidades principales; se considera la *vista lógica* de un servicio [Mil01] como una funcionalidad y un servicio como tal debe satisfacer sus propias propiedades de calidad de alto nivel; se actualiza o transforma la MCE obtenida en el proceso QuaDRA (ver Tabla 32) para incluir servicios simples y compuestos como componentes arquitecturales; esta transformación se muestra en la Tabla 42.

**Tabla 42.** Tabla MCE – Portafolio de Servicios

CA Componentes Comunes (CC)	Servicio	Descripción	Propiedades de Calidad de Alto Nivel de Servicio – Prioridad en ( ); 1 alta, 2 media, 3 baja		Restricciones
			Requerida	Proporcionada	
			- a2. Patient Portal - a3. Reports	Servicios simples - botones	
a1-Web pages – Browser	Servicio compuesto	Servicio de interfaz web	- Disponibilidad-Persistencia (1) - Comportamiento en Tiempo (2) - Adaptabilidad (2), - Capacidad-Escalabilidad (2) - Interoperabilidad (1)	- conexión a la red - protocolos de red - por ser WS - protocolos de red por ser WS - por ser WS	- disponibles en Capa de transmisión;
a4. GUI	No es un servicio	Modulo de Interfaz Usuario stand-alone	- Autenticidad (1) - Confidencialidad (1), - Integridad (1) - Disponibilidad-	- módulos de seguridad - modulo que se	- desarrollados por programación - 100% disponible, no

			Persistencia (1)	despliega en el cliente	depende de la conexión en red
			- Usabilidad	- diseño de páginas Web	No es una propiedad arquitectural; no será tratada aquí, depende del diseño de las páginas
<i>b1. Patient</i>	<i>Servicio Compuesto</i>	- Programación de citas presenciales en el sitio de atención medica  - Recuperación, almacenamiento, modificación, eliminación de los datos personales y demográficos del paciente\	- Correctitud-precisión, - Disponibilidad-Persistencia,  - Autenticidad, - Confidencialidad, - Integridad  - Portabilidad	- modulo de cálculo - c1  - modulo de seguridad + HTTP/HTTPS  - c5, c6	- b1, b2, b3 son RF CC SIS obligatorios; son servicios compuestos; - no está presente, debe ser agregado - mecanismos de base de datos para la persistencia;  - el modulo seguridad no está presente, debe ser agregado en Capa de transmisión, combinado con protocolos de red HTTP/HTTPS existentes; algunos de estos mecanismos de seguridad deben estar presentes, por lo menos uno es obligatorio por ser un RNF SIS prioritario
<i>b2. MedicalPractice</i>	<i>Servicio compuesto:</i>	- Todo lo relativo a la atención medica presencial al paciente	- Capacidad-Escalabilidad	- c1	- c5, c6 son servicios simples alternativos para la portabilidad de bases de datos comerciales a diferentes plataformas;  - c1 es un servicio complejo común, un Sistema Manejador de Base de Datos (SMBD) relacional
- <i>b12-EHRManager</i>	<i>Servicio compuesto:</i>	- Recuperación, creación, almacenamiento, modificación, compartición del EHR	- Disponibilidad-Persistencia  - Autenticidad, - Confidencialidad, -Integridad	- c1  - modulo de seguridad + HTTP/HTTPS	- b2 CC obligatorio, se ha subdividido en sub-componentes b12 y b14 que son también servicios compuestos y son CC obligatorios;
- <i>b14.Orders&amp;Reports</i>	<i>Servicio compuesto</i>	- Servicios de emisión, recuperación, modificación y almacenamiento de órdenes e informes médicos	- Interoperabilidad  - Comportamiento en Tiempo (2)	- b4 y/o c2 o c3  - protocolos de red	- interoperabilidad es un RNF SIS obligatorio que es resuelto por b4. Mirth en proceso; c2 o c3 son soluciones opcionales para Interoperabilidad en datos - disponibles en Capa de transmisión
<i>b3. AdminSystem</i>	<i>Servicio compuesto</i>	- servicios administrativos básicos: facturas, facturación, informes	- Autenticidad, - Confidencialidad, -Integridad	- modulo de seguridad + HTTP/HTTPS  - c1	- b3 RF CC obligatorio

		estadísticos	- Disponibilidad- Persistencia - correctitud- precisión	- c1  - modulo de cálculo	
<i>b4. Mirth</i>	<i>Servicio simple: b4</i> <<variant>>	- Servicios de traducción para EHR al formato HL7	- Interoperabilidad	- <i>Motor del mercado de traducción a HL7</i>	b4 es variante, pero obligatorio para resolver el RNF SIS de interoperabilidad
<i>b5. GUI Server Side</i>	- Servicio compuesto para comunicación cliente/ servidor local y remota	- Comportamiento en Tiempo - Modificabilidad- Modularidad	- protocolos de red  - modulo	- TCP/IP, proxy o gateway - implementa el patrón MVC	- disponibles en Capa de transmisión - disponible por programación
<i>c1-DB</i>	<i>Servicio Compuesto</i>	- <i>servicios de bases de datos a través de los SGBD habituales</i>	- Capacidad- Escalabilidad - Integridad - Disponibilidad- Persistencia - Adaptabilidad - Interoperabilidad	- módulos, mecanismos	c1 es un RF CC SIS obligatorio, entonces, todos los subcomponentes son también CC obligatorios;
- <i>c2. HL7Eng</i> - <i>c3. HXPEng</i>	<i>Conjunto de servicios simples: c2, c3</i> <<variants>>	- <i>proporcionan interoperabilidad con HL7 en Capa de datos</i>	- Interoperabilidad	- motores	RNF opcional; al menos b4 debe estar presente en una configuración arquitectural
- <i>d1. Internet</i> - <i>d2. Satellite</i>	<i>Servicios Compuestos: d1; d2;</i> <<variant>>	- Infraestructura de red, permite la comunicación entre las capas representadas como Servidores Web	- Comportamiento en Tiempo, - Disponibilidad- Persistencia, - Adaptabilidad, - Capacidad- Escalabilidad - Autenticidad, - Confidencialidad, - Integridad	- protocolos HTTP/HTTPS	d1 es RF CC, d2 es variante

### 1.2 Elaborar el estudio de mercado para la LPSOS

El estudio de mercado es considerado en el análisis de los productos del mercado, que se supone que existen en el dominio (Tabla 25), para garantizar la viabilidad de la LPS.

### 1.3 Esbozo de la similitud y variabilidad para futuros productos de la LPSOS.

Nuevas características para los nuevos productos en la familia LPS SIS son especificados. La Tabla 25 mostró las similitudes y la variabilidad de los tres productos estudiados y la Tabla 42 describió las funcionalidades del EHR-SIS para LPSOS para conformar el Documento de los Productos Futuros:

- *sistema de imagen (LabImage System)*: resultados de imágenes de laboratorio.
- *asistente de diagnóstico (Diagnosis)*: diagnóstico en línea.
- *servicios de citas medica on-line*: programar citas de los pacientes on-line, selección de las instituciones de salud y de los especialistas.

## 2. Alcance del Dominio

Los pasos 2.1, 2.2 y 2.3 son detallados en la sección 6.2.2 y en [HLO16b]; ellos definen el Modelo del Dominio representado por los BP en BPMN.

### 2.4 Modelado del dominio- descomposición del dominio - SOMA

#### 2.4.1 Especificación de Servicios - SOMA - identificación de adicional BG

En las Figuras 27, 29 y 31 los comportamientos o funciones (BG), denotadas por <<function>> son: acceso y registro del EHR “EHR ACCESS&REGISTER”, realizar diagnóstico “PERFORM DIAGNOSIS”, emisión de nuevas citas “EMISSION OF NEW APPOINTMENT”, emisión de Ordenes/Informes/Facturación “EMISSION OF MEDICAL/ ORDER/ REPORT/ BILLING”; estas funciones son SC desde el punto de vista de los médicos “doctores”. Otros puntos de vista de las otras partes interesadas y las facetas son estudiadas para obtener BG adicionales: sistema paciente “PATIENT SYSTEM”, sistema de gestión de EHR “EHR MANAGEMENT SYSTEM”, sistema de reportes “REPORT SYSTEM”, DBMS, servidor de aplicaciones “APPLICATION SERVER”, intercambio del EHR “EHR SHARING”, SpDBMS, asistente de síntomas en línea “ON-LINE SYMPTOMS ASSISTANT”, gestión de citas en línea “ON-LINE APPOINTMENT MANAGEMENT”. Los SC son aquellos que satisfacen estos BG y que son agrupadas en BPMNAgg (Figura 32) para conseguir los componentes arquitecturales de la AC.

MCE (Portafolio de servicios, Tabla 42 se actualiza en MCEA con los nuevos servicios en la Tabla 43).

#### 2.4.2 Análisis de Subsistemas SOMA – en nuestros términos: agregación, AC actualizada, actualización de MCE (MCAE) (portafolio de servicios actualizado), mapa de agregaciones a servicios compuestos

Las actividades y funciones de las vistas de las partes interesadas y facetas se encuentran agregadas en BPMNAgg para tener un Modelo de Dominio único (Figura 32) con comportamientos funcionales agregados (RF) y sus propiedades de calidad



(RNF), representados por BG y asignado a SC, que serán los componentes arquitecturales de la ACA con servicios, cuyo diagrama UML no se mostrará aquí, por ser similar a la Figura 33; el MCE (Portafolio de Servicios) en la Tabla 42 es actualizado en MCEA (Portafolio de Servicios Actualizado) en la Tabla 43; los servicios mencionados como futuros productos a ser desarrollados en la Sección 1.3 son los nuevos servicios agregados.

El Documento Productos Futuros (Paso I.1.3) y el BPMNAgg son estudiados para considerar si los nuevos servicios compuestos contribuyen a los futuros productos de la LPS; cambios de la AC son asignados (vinculados) a nuevos servicios compuestos (resaltados en color gris en la Tabla 43).

**Tabla 43.** MCEA (Portafolio de Servicios Actualizado)

CA Componentes Comunes (CC)	Servicio	Descripción	Propiedades de Calidad de Alto Nivel de Servicio – Prioridad en ( ); 1 alta, 2 media, 3 baja		Restricciones
			Requerida	Proporcionada	
- a2. Patient Portal - a3. Reports	<i>Servicios compuesto</i> - a1. Web Pages- Browser <<variant>>	Interfaz usuario para HCE-SIS	- Autenticidad (1), - Confidencialidad (1), - Integridad (1), - Disponibilidad-Persistencia (1), - Comportamiento en Tiempo (2), - Adaptabilidad (2) - Capacidad- Escalabilidad (2),	- HTTP/HTTPS  - protocolos de red - por ser WS  - protocolos de red por ser WS  - por ser WS	RF variante
	<i>No es un servicio</i> - a4. GUI <<variant>>	Interfaz usuario stand-alone para HCE-SIS	- Autenticidad (1) - Confidencialidad (1), - Integridad (1) - Disponibilidad-Persistencia (1)	- módulos de seguridad	RF variante  - 100% por ser stand-alone
	<i>Servicios simples</i> - botones	Acceso a las funcionalidades HCE-SIS	- Autenticidad (1) - Confidencialidad (1), - Integridad (1)  - Disponibilidad-Persistencia (1)	- conexión a los servidores por d1, d2 o b5 vía HTTP/HTTPS  - conexión a la red	RF CC obligatorios
			- Usabilidad	- diseño de páginas Web	Esta no es una propiedad arquitectural, y no será tratada aquí.

<i>b1-Patient</i>	<i>Servicio Compuesto por los nuevos servicios: b18, b19, b20</i>				<ul style="list-style-type: none"> <li>- b1 es un RF CC SIS obligatorio;</li> <li>- b18, b19 y b20 son servicios compuestos de b1, y también son servicios CC obligatorios;</li> </ul>
- b18- <i>AppointService</i>	<i>Servicio compuesto: b18</i>	- programación de citas en el sitio y/o en línea	- Correctitud-Precisión, - Disponibilidad-Persistencia, - Autenticidad, - Confidencialidad, - Integridad - Portabilidad - Capacidad-Escalabilidad	- b6  - c7, c9a, c9b	- Nuevo, módulos para algoritmos de cálculo; - mecanismos de base de datos para la persistencia - protocolos de red combinados con mecanismo de seguridad b7, el cual es un módulo adicional de seguridad para políticas de acceso que puede agregarse para mayor seguridad además de los protocolos HTTP/HTTPS
- b19- <i>SymptomAssistant</i>	<i>Servicio compuesto: b19</i>	- Asistencia a la búsqueda de sintomatología para encontrar especialista médico	- Las propiedades anteriores se mantienen para los servicios b19, b20	- b7-HTTP, b7-HTTPS, HTTPS  - c8 - c5, c6 - c4	- c8 es un servicio simple o mecanismo para proporcionar integridad adicional, es opcional; - c5, c6 son servicios simples alternativos para la portabilidad de bases de datos comerciales a diferentes plataformas; - c4 es un servicio simple para añadir nuevos estándares médicos.
- b20- <i>DemographicData</i>	<i>Servicio compuesto: b20</i>	- Recuperación, almacenamiento, modificación, eliminación de los datos personales y demográficos del paciente		Los servicios anteriores son mantenidos para b19, b20	- c8 es un servicio simple o mecanismo para proporcionar integridad adicional, es opcional; - c5, c6 son servicios simples alternativos para la portabilidad de bases de datos comerciales a diferentes plataformas; - c4 es un servicio simple para añadir nuevos estándares médicos.
<i>b2-MedicalPractice</i>	<i>Servicio Compuesto por los servicios: b12, b13, b14</i>	- Todo lo relativo a la práctica medica presencial			
- b12. <i>EHRManager</i>	<i>Servicio compuesto: b12</i>	- Recuperación, creación, almacenamiento, modificación, compartición del EHR	- Interoperabilidad, - Disponibilidad-Persistencia, - Autenticidad, - Confidencialidad,	- b4 y/o c2 o c3 - c7, c9a, c9b  - b7-HTTP, b7-HTTPS, HTTPS	- b2 es un RF CC obligatorio; b12, b13 y b14 son también servicios compuestos de b2, y ellos son también CC obligatorios;
- b13. <i>Diagnosis</i>	<i>Servicio compuesto:</i>	- Recuperación de catálogos médicos, protocolos; búsqueda de sintomatología	- Integridad	- c8 -	- HTTP/HTTPS protocolos en la Capa de Transmisión + b7 en proceso
- b14. <i>Orders&amp;Reports</i>	<i>Servicio compuesto: b14</i>	- Servicios de emisión, recuperación, modificación y almacenamiento de órdenes e informes médicos	- Disponibilidad-Persistencia, - Autenticidad, - Confidencialidad, - Capacidad-Escalabilidad	c7, c9a, c9b  - b7-HTTP, b7-HTTPS, HTTPS  - c1	- b12 es un RNF SIS obligatorio resuelto por b4 en proceso; c2 o c3 son soluciones opcionales para Interoperabilidad en datos
<i>b3. AdminSystem</i>	<i>Servicio compuesto: b3</i>	- servicios administrativos básicos: facturas, facturación, informes estadísticos	Autenticidad, - Confidencialidad,  - Integridad, - Disponibilidad-Persistencia - Correctitud-Precisión	- b7-HTTP, b7-HTTPS, HTTPS  - c8 - c7, c9a, c9b  - b6	- b3 es un RF CC obligatorio;

<b>b15.</b> <b>LabImageSystem</b>	<i>Servicio compuesto:</i> <i>b15</i>	- recuperación, almacenamiento, modificación de estudios de laboratorio y de imágenes radio gráficas	- Interoperabilidad - Disponibilidad-Persistencia, - Autenticidad, - Confidencialidad,  - Integridad, - Correctitud-Precisión	- b16, b17 - c7, c9a, c9b  - b7-HTTP, b7-HTTPS, HTTPS  - c8 - b6	b15 es un RF CC obligatorio
<b>b16. LOINC</b>  <b>b17. DICOM</b>	<i>Conjunto de servicios simples:</i> <i>b16, b17</i> <i>&lt;&lt;variant&gt;&gt;</i>	- Base de datos incluyendo un estándar para la identificación de las observaciones de laboratorio médico  - Mecanismo para la conversión a formatos gráficos estándar de imágenes; conformada por un archivo de formato y un protocolo común; los archivos se intercambian a través de TCP/IP	- Interoperabilidad  - Interoperabilidad	- <i>base de datos (repositorio)</i>  - <i>Motor de traducción de formatos de imágenes del mercado</i>	Interoperabilidad es un RNF obligatorio para b15; b16, b17; ambos pueden estar presentes en una configuración; uno por lo menos es mandatorio  - traducción a estándares de formatos de imágenes
<b>b6. Algor*</b>  <i>(*) multiplicidad</i>	<i>Conjunto de servicios simples:</i> <i>b6</i> <i>&lt;&lt;variant&gt;&gt;</i>	- Algoritmos para cálculos	- Correctitud-Precisión	- <i>módulos</i>	Correctitud-Precisión es un RNF obligatorio para b1, b3 y b15
<b>b7. Data Security</b>	<i>Conjunto de servicios simples:</i> <i>b7</i> <i>&lt;&lt;variant&gt;&gt;</i> - <i>b7-HTTP</i> <i>&lt;&lt;variant&gt;&gt;</i> - <i>HTTPS</i> <i>&lt;&lt;variant&gt;&gt;</i> - <i>b7-HTTPS</i> <i>&lt;&lt;variant&gt;&gt;</i>	- Servicios o algoritmos para la identificación del usuario (Autenticidad) y políticas de derechos de acceso del usuario (Confidencialidad) de la información médica	- Autenticidad, - Confidencialidad, - Integridad	- <i>b7-HTTP</i> - <i>HTTPS</i> - <i>b7-HTTPS</i>	- La seguridad es un RNF SIS obligatorio; alternativas para algoritmos de seguridad en b7 combinados con protocolos de seguridad HTTP / HTTPS; la integridad también es abordada en la base de datos; sólo una alternativa es válida
<b>b4. Mirth</b>	<i>Servicio simple:</i> <i>b4</i> <i>&lt;&lt;variant&gt;&gt;</i>	- Servicios de traducción para EHR al formato estándar HL7	- Interoperabilidad	- <i>Engine de traducción a HL7 del mercado</i>	b4 es obligatorio para resolver el RNF SIS de interoperabilidad
<b>b5. GUI Server Side</b>	- Servicio compuesto para comunicación cliente/ servidor local y remota	- Comportamiento en Tiempo - Modificabilidad-Modularidad	- protocolos de red  - módulos	- TCP/IP, proxy o gateway - patron MVC	- disponibles  - disponible por programación
<b>c1. DB</b> - <i>c16-EHRDB</i> - <i>c17-LABIMDB</i> - <i>c18-SPDB</i> - <i>c19-CATDB</i>	<i>Conjunto de servicios compuestos:</i> <i>c16, c17, c18, c19</i>	- <i>servicios de bases de datos a través de los SGBD habituales</i>	- Capacidad-Escalabilidad - Integridad - Disponibilidad-Persistencia - Adaptabilidad - Interoperabilidad	- módulos, mecanismos	c1 es un RF CC obligatorio, entonces, todos los subcomponentes son también CC obligatorios; en este estudio sólo c1 es considerada
<b>c4. NewMedStds *</b>	<i>Conjunto de servicios simples:</i> <i>c4</i> <i>&lt;&lt;variant&gt;&gt;</i>	- <i>Adición de nuevos estándares para la práctica médica: formatos, catálogos, protocolos médicos, etc.</i>	- Capacidad-Escalabilidad	- módulos	RNF opcional
<b>c8. IntegMech *</b>	<i>Conjunto de servicios simples:</i> <i>c8</i> <i>&lt;&lt;variant&gt;&gt;</i>	- <i>Proporciona integridad de los datos para bases de datos</i>	- Integridad	- mecanismos	RNF obligatorio para c1-DB

c9. AvailabMech - c9a. Mirror - c9b. MirrorReplie	Conjunto de servicios simples: c9, c9a, c9b <<variant>>	- Proporciona copias de seguridad para bases de datos	- Disponibilidad-Persistencia	- mecanismos	RNF obligatorio para c1-DB; solo una alternativa es valida
c7. Hibernate	Conjunto de servicios simples: c7	- Proporciona persistencia de datos a las bases de datos	- Disponibilidad-Persistencia	- motor	RNF obligatorio para c1. DB
c5. JDBC c6. ODBC	Conjunto de servicios simples: c5, c6 <<variants>>	- Portabilidad a diferentes plataformas de bases de datos comerciales	- Adaptabilidad	- APIs	RNF opcional para c1. DB
c2. HL7Eng c3. HXPENg	Conjunto de servicios simples: c2, c3 <<variants>>	- proporciona interoperabilidad de los datos	- Interoperabilidad	- engines	RNF opcional; al menos b4 debe estar presente en una configuración arquitectural
- d1. Internet - d2. Satellite	Servicios Compuestos: d1; d2; <<variant>>	- Infraestructura de red, permite la comunicación entre las capas representadas como Servidores Web	- Comportamiento en Tiempo, - Disponibilidad-Persistencia, - Adaptabilidad, - Capacidad-Escalabilidad - Autenticidad, - Confidencialidad, - Integridad	- protocolos HTTP/HTTPS	d1 es RF CC, d2 es variante

### 3. Alcance de los Activos - para determinar la Propuesta de Activos Base de la LPS

La propuesta de Activos Base está conformada por la AC, ACA, MCE, MCEA, MCD, BPMNagg, relativos a servicios. Los diagramas BPMN no presentan cambios respecto a los obtenidos en el la Sección 6.2 para proceso QuaDRA, por los tanto no serán mostrados aquí.

## II. Fase de Ingeniería de Requisitos del Dominio (DRE).

Los pasos II.1, II.2, II.3 y II.4 para análisis de requisitos, especificación, validación y gestión han sido considerados en la ACA y en el MCEA; si la ACA es representada por una ontología [LO16], herramientas de razonamiento pueden ser utilizadas para validar la consistencia de requisitos mediante la formulación correcta de reglas de consistencia acorde a las restricciones obligatorias y opcionales en el MCEA (Tabla 43). El enfoque del modelo orientado a características (FODA) [LKK02] no es utilizado aquí como tal porque se sustituye por la Tabla 43, para considerar las opciones obligatorias y opcionales del modelo de variabilidad WSRA (Tabla 43).

### ***III. Fase de Diseño del Dominio (DD).***

#### *1. Diseño de la WSRA-LPS - Especificación Componentes SOMA: modelado de la variabilidad*

La WSRA está conformada por CC (servicios compuestos) y por <<vp>> (conjunto de servicios simples o compuestos) que agrupan variantes de la ACA (servicios simples o compuestos) realizando tareas semánticamente similares (ver Tabla 41 de documentación y MCEA, Tabla 43).

- La WSRA (Figura 36) es un grafo conectado, no dirigido, construido a partir del MCEA; este también puede ser especificado como una ontología [LO16], para lo cual se utilizan las tablas 41 y 43.

- *Documentación de la WSRA:*

Como última actividad, se produce la Tabla de Documentación del Dominio para la WSRA (DOC-WSRA), debe ser construida aquí, pero no presenta cambios respecto a la Tabla 41 DOC-RA, salvo por el hecho de que como se manejan servicios, el componente *a4. GUI* que no es un servicio, debería pasar por un proceso de “wrapping” para ser transformado en WS.

#### *2. Evaluación de la WSRA: es una técnica de aseguramiento de la calidad.*

La WSRA cumple con los requisitos de calidad por construcción; cada componente WSRA cumple con la calidad requerida; por otra parte, la WSRA satisface los requisitos de calidad del estilo arquitectural del dominio.

#### *3. Gestión de Diseño del Dominio:*

Los requisitos de trazabilidad son resueltos por construcción, para garantizar la evolución de la WSRA; todas las propiedades de calidad de alto nivel requeridas por cada servicio compuesto en la arquitectura son especificadas y resuelto por servicios simples.

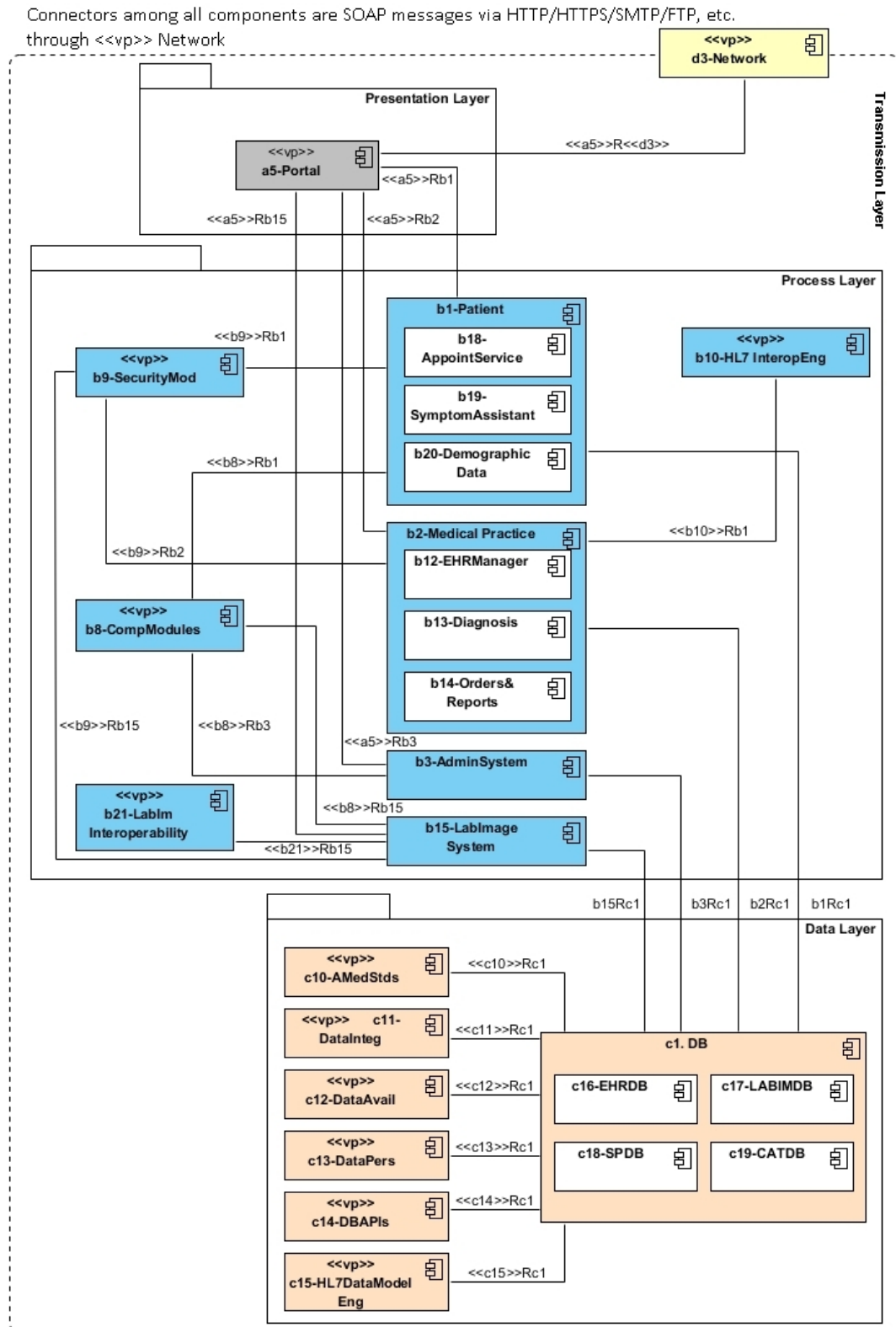


Figura 36. WSRA; Fuente: autor

### **6.3.2 Análisis de los Resultados**

Como se ha señalado en este trabajo, LPS y SOA son enfoques de desarrollo de software utilizados en la práctica industrial favoreciendo la reutilización de los activos y capacidades existentes, en lugar de re-desarrollar para construir nuevos sistemas. Este trabajo explota los beneficios de ambos enfoques integrándolos en un único proceso de diseño arquitectónico de LPS orientado a servicios denominado WSRA-LPS, especificado en el capítulo 5, que es nuestro principal resultado. En la literatura discutida en el capítulo 3, no se encontró un proceso completo para el diseño de un AR considerando WS. Sólo el proceso SOMA, discutido en la sección 4, se encontró, pero construye una arquitectura para un sistema único y no para una familia de sistemas, como es el caso de la LPS. Esta fue la principal motivación para especificar nuestro proceso WSRA-LPS.

En la sección 6.2 de este capítulo, el proceso QuaDRA [HLO16b] fue definido para construir una AR para LPS sin considerar explícitamente los WS como componentes de la AR, por lo tanto, QuaDRA fue adaptado en este presente trabajo incorporando el método SOMA para conformar el proceso de diseño WSRA-LPS. Fue utilizada la definición de la arquitectura de software clásica [SG96]; donde los componentes son representados como servicios y sus conexiones es la comunicación entre ellos, es decir, los mensajes SOAP intercambiados vía HTTP, HTTPS, SMTP, FTP; en la actualidad SOAP ha sido desplazado por mensajes REST que usan los comandos de HTTP (GET/PUT/DELETE/...) como operaciones; los servicios web basados en SOAP hoy en día son considerados legacy.

Los WS son considerados en la WSRA-LPS como componentes conformando la vista lógica, abstracta de un sistema, módulos, programas, bases de datos, etc., llevando a cabo una operación del negocio o funcionalidad, especificado por los procesos de negocios. Por lo tanto un modelo de negocio fue considerado en la primera fase de WSRA-LPS, el Alcance de la LP (PLScoP).

Otra importante consideración incluida en WSRA-LPS fue que la orquestación de los WS a partir de los flujos de trabajo de los procesos de negocio fue

asumida como correcta, siendo esta responsabilidad del middleware bus de servicios y/o de herramientas de traducción automática; lo que es importante para WSRA-LPS es que los WS actúan juntos en una composición como una sola aplicación [IBM09], y se supone que es correcta. Por lo tanto, desde la perspectiva arquitectónica, no se aborda el intercambio de mensajes, sino en su coreografía representando las propiedades de un componente y las relaciones entre los servicios que la conforman.

Con respecto a incluir explícitamente los problemas de calidad, como en la configuración arquitectónica producida por QuaDRA, los servicios simples son también considerados elementos arquitectónicos procedentes de las funcionalidades implícitas proporcionando soluciones o mecanismos de bajo nivel para satisfacer las propiedades de calidad de alto nivel requeridas por los componentes más complejos resolviendo las funcionalidades del sistema, designados como servicios compuestos.

En consecuencia, los tres problemas principales encontrados al combinar los enfoques LPS y SOA en un único proceso de diseño de arquitectura de referencia, fueron resueltos en esta investigación:

- la representación de la AR LPS con servicios, representados en la notación UML estándar;
- la satisfacción de las propiedades de calidad requeridas por las funcionalidades para comportarse correctamente, obtenida por la construcción en los primeros pasos del proceso;
- finalmente, la gestión de la variabilidad requerida en la LPS, la cual es obtenida conformando la WSRA con servicios compuestos comunes y puntos de variabilidad [PBL05] o con un conjunto de servicios que contienen variantes (servicios simples o compuestos), fueron construidas al agrupar las variantes de la ACA (servicios simples o compuestos) realizando tareas semánticamente similares.



# Referencias

---

- [Bjø06] Bjørner, D. (2006). Software Engineering 3 Domains, Requirements, and Software Design. Texts in Theoretical Computer Science. EATCS Series. Editors: W. Brauer G. Rozenberg A. Salomaa. Springer-Verlag Berlin Heidelberg.
- [BKB02] Bass, L., Klein, M., & Bachmann, F. (2002). Quality attribute design primitives and the attribute driven design method. Software Product-Family Engineering, 323-328.
- [Cau13] Caulton G. (2013). PatientOS – an Open Source (GPL) Health Care Information System. <http://www.patientsos.org>
- [COS05] COSGov Vietnam. (2005). Care2X, International Conference and Expo, Hanoi, September.
- [HLO16b] J. Herrera, F. Losavio, and O. Ordaz, QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines based on ISO/IEC 26550, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 1, pp. 20-38, enero-junio 2016.
- [IBM09] IBM. (2009). IBM Service-oriented Architecture (SOA) Solutions. IBM SOA Foundation.
- [LO16] Losavio, F. & Ordaz, O. (2016). Reference Architecture Representation by an Ontology for Healthcare Information Systems Software Product Line, 4to Simposio SCTC 2016, pp. 20-32, ISBN: 978-980-12-8407-9, Escuela de Computación, Universidad Central de Venezuela, 9-11 Mayo.
- [LOE15] Losavio, F., Ordaz, O. & Esteller, V. (2015). Refactoring-Based Design of Reference Architecture, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS) 5(1) pp. 32-48.
- [LOS15] Losavio, F., Ordaz, O. & Santos, I. (2015). Proceso de análisis del dominio ágil de sistemas integrados de salud en un contexto venezolano, ENL@CE, Vol. 12, No.1, pp.101-134.
- [MID10] Morrison, C., Iosif, A. & Danka, M. (2010). Report on existing open-source electronic medical records, University of Cambridge, Computer Laboratory, Technical Report No. 768, UCM-CL-TR-768, ISSN 1476-2986.
- [OMG05] Object Management Group (OMG). (2005). Unified Modelling Language Superstructure, version 2.0 (formal/05-07-04), August.
- [OMG11] Object Management Group (OMG). (2011). Business Process Model and Notation (BPMN), Version 2.0. Available in: <http://www.omg.org/spec/BPMN/2.0/>

- [Paz10] Pazos P., Taller de OpenEHR (2010). Un Estándar para crear HCEs. JAIIO CAIS. <http://www.slideshare.net/pablitox/taller-open-ehr-cais-2010-pablopazos>
- [PBL05] Pohl, K., Bockle, G. & Van Der Linden, F. (2005). Software Product Line Engineering: Foundations, Principles, and Techniques. Springer.
- [Sam10] Samilovich, S. (2010). OpenEMR – Historia Clínica Electrónica de código abierto y distribución gratuita, apta para su uso en el sistema de salud Argentina, JAIIO CASI
- [SG96] Shaw, M. & Garlan, D. (1996). Software architecture: perspectives on an emerging discipline. Prentice Hall.

## ***CAPÍTULO VII***

### ***Conclusiones y Perspectivas***

---

---

## CAPÍTULO VII

### CONCLUSIONES Y PERSPECTIVAS

En este capítulo se presentan las conclusiones de la investigación resumiendo el cumplimiento de cada uno de los objetivos planteados, con el fin de esbozar los aportes obtenidos, su originalidad y algunas de las limitaciones del trabajo. Así mismo se plantean perspectivas con algunas consideraciones sobre el trabajo futuro para continuar la presente investigación y dar inicio a nuevas investigaciones en esta área.

#### 7.1 Logros

Basados en el planteamiento inicial de la investigación, se pueden enumerar los resultados más importantes obtenidos:

1. Se realizaron tres Revisiones Documentales Sistemáticas (RDS) en las etapas de investigación exploratoria y descriptiva; debe observarse que los temas generales de las RDS fueron sugeridas por los tutores, las RDS fueron completamente desarrolladas por J. C. Herrera:
  - Una primera RDS [HLM14] sobre Enfoques y Técnicas para la Construcción de Arquitecturas en un Contexto de LPS, la cual permitió:
    - proponer una *estrategia de desarrollo híbrida* para LPS y LPSOS: top-down proactivo en la fase de Análisis del Dominio y bottom-up extractivo en la fase de Estudio de Productos Existentes para desarrollar automáticamente una arquitectura candidata inicial [LOE15].
  - Una segunda RDS [HLO15] en el Ámbito de la Ingeniería del Dominio para LPSOS, la cual permitió:

- estudiar el modelo de referencia estándar ISO/IEC 26550 para ILPS, y compararlo con otros procesos similares [PBL05] y [Bj06], observando deficiencias importantes como la consideración tardía del aseguramiento de la calidad respecto a la arquitectura de referencia, producto central de la LPS.
  - Una tercera RDS [HLO16c] sobre métodos para el Diseño de la AR para LPSOS, el cual integra los enfoques LPS y SOA, la cual permitió:
    - Determinar el modelo de variabilidad de la AROS-LPS utilizando servicios simples y compuestos como variantes.
2. En la etapa de investigación proyectiva se realizaron cuatro trabajos importantes [HLO16a] [HLO+16] [HLO16b] [HLO16d], que contemplan:
- definir un proceso sistemático de Análisis del Dominio para LPS y LPSOS [HLO16a][HLO+16], con técnicas específicas basadas en las teorías clásicas de Bjørner [Bj06] (contribución original de J. C. Herrera), incorporando el tratamiento temprano de los requisitos de calidad, dentro de la primera fase del ciclo ID de la ILPS (contribución y originalidad compartida por F. Losavio, O. Ordaz y J. Bøegh), la fase de Alcance de LPS del modelo de referencia estándar ISO/IEC 26550 [ISO15] (contribución original de J. C. Herrera). En esta fase se contempla la construcción automática de la arquitectura candidata inicial [LOE15] a partir de la unión de los grafos conexos y no dirigidos que representan las arquitecturas de productos existentes (contribución y originalidad compartida por F. Losavio, O. Ordaz y V. Esteller).
  - definir el proceso QuaDRA [HLO16b], centrado en la calidad, sistemático y repetible, incluyendo la fase de Alcance de la LPS definida en [HLO16a][HLO+16] para disminuir el trabajo de las fases subsiguientes del ciclo ID de la ILPS (contribuciones y originalidades compartidas entre todos los autores).

- definir el proceso WSRA-SPL [HLO16d], adaptado de QuaDRA incorporando partes del método SOMA utilizado para el desarrollo SOA, adaptándolo al enfoque LPS, para obtener así una AROS-LPS (contribuciones y originalidades compartidas entre todos los autores).
3. Fueron resueltos los siguientes problemas:
- representar la vista lógica [Kru95] de la AROS, como una arquitectura de software "real" (componentes y conectores), aspecto ausente en la literatura; se consideró la composición por coreografía de los servicios como un conjunto de componentes en la AROS, conectados por el usual intercambio de mensajes entre servicios. Debe recordarse que no solo la AROS, sino todas las arquitecturas manejadas en los procesos propuestos en esta investigación son representadas por grafos conexos no dirigidos, para facilitar así la construcción automática de una primera arquitectura de referencia base o inicial, la arquitectura candidata AC; la representación de las arquitecturas por grafos por lo tanto ha permitido reducir el esfuerzo global de desarrollo de la ID en la ILPS.
  - Considerar en el proceso WSRA-SPL los requisitos de calidad: las características de calidad de alto nivel requeridas por los componentes funcionales o servicios compuestos, como servicios simples o compuestos (mecanismos, API, motores o herramientas tecnológicas, etc.) proveedores de estas cualidades.
  - Construir el modelo de variabilidad en la WSRA teniendo en cuenta los puntos de variación integrados “*embedded*” por construcción dentro de la WSRA, teniendo en cuenta tareas semánticamente similares en cuanto a sus objetivos funcionales o no funcionales.
4. La Tabla de Documentación de la AROS constituye la entrada al ciclo de vida de la IA. Contiene los elementos comunes, variantes y las restricciones exigidas por la AROS; a partir de estas restricciones habría que formular

reglas de consistencia para la derivación de configuraciones arquitectónicas válidas de productos concretos; nótese que la conectividad y el cumplimiento de las propiedades de calidad exigidas por las funcionalidades y por el dominio, se cumplen por construcción. Una representación ontológica de la AROS puede ser utilizada en este sentido [LOJ 16].

5. Para validar la propuesta, esta se aplicó al dominio de los Sistemas de Información integrados de Salud (SIS), permitiendo ilustrar cada uno de las fases, actividades y artefactos utilizados en el proceso, validando así la construcción del Modelo del Dominio y la obtención de la AROS. Este dominio es de particular interés tanto desde el punto de vista social como del punto de vista informático, ya que son sistemas complejos con importante exigencias de calidad, su adopción global es aún problemática a nivel mundial y no hay aún LPSOS completamente definida en este dominio.

## 7.2 Limitaciones

- Aún, cuando los componentes de software son desarrollados respondiendo a algún requisito de calidad, y el producto de software es construido a partir de estos componentes, la calidad de todo el producto no puede ser garantizada; sin embargo, nosotros consideramos la calidad global del producto en cierta medida, asegurando que las propiedades de calidad del estilo arquitectural del dominio sean satisfechas; pero esto sigue siendo un tema de investigación abierto.
- Los QoS, que son valores de bajo nivel de abstracción, también pueden ser utilizados para la selección de variantes en el ciclo de vida de la IA, donde los nuevos productos de la familia LPS son derivados; sin embargo el problema de los QoS está fuera del alcance de nuestro trabajo, ya que la WSRA-LPS se refiere al ciclo de vida de la Ingeniería de Dominio.
- La falta de herramientas automáticas de soporte a los proceso QuaDRA y WSRA-SPL

### 7.3 Perspectivas

El proceso WSRA-SPL ha sido especificado como una herramienta reutilizable, precisando artefactos y técnicas, y se ha aplicado al dominio de los SIS y puede ser repetido siguiendo su especificación. Los pasos automáticos en el proceso de WSRA-LPS son la construcción de la AC inicial y la generación de la WSRA a partir de la AC actualizada; sin embargo, consideramos necesarios crear herramientas de software para apoyar los pasos semiautomáticos de la WSRA-LPS, siendo este un trabajo en curso.

Adicionalmente, se encuentra en fase de definición de una ontología para representar la AROS, que permitirá validar la consistencia de las configuraciones arquitecturales para la derivación de productos concretos de la AROS, en el ciclo IA.

Otros aspectos, que no se han tocado en este trabajo y que aún son problemas abiertos de investigación, son los siguientes:

- la gestión y construcción sistemática del Repositorio de Activos Base de la LPSOS;
- evaluar la escalabilidad de la AROS respecto al número y calidad de los productos que se pueden derivar a partir de ella, para una familia LPSOS determinada.



# Referencias

---

- [Bjø06] Bjørner, D. (2006). Software Engineering 3 Domains, Requirements, and Software Design. Texts in Theoretical Computer Science. An EATCS Series. Editors: W. Brauer G. Rozenberg A. Salomaa. Springer-Verlag Berlin Heidelberg.
- [HLM14] J. Herrera, F. Losavio, and A. Matteo, RDS de enfoques y técnicas para la construcción de arquitecturas en un contexto de líneas de productos de software, Revista Venezolana de Ciencias de la Computación ReVeCom, Vol. 1, No. 1, pp. 17-25, Junio 2014.
- [HLM+14] J. Herrera, F. Losavio, A. Matteo, and O. Ordaz, Revisión Documental Sistemática en el Ámbito de la Ingeniería del Dominio para Líneas de Productos de Software Orientados a Servicios, ReVeCom, Vol. 1, No. 2, pp. 23-33, Diciembre 2014.
- [HLO15] J. Herrera, F. Losavio, and O. Ordaz, Ingeniería del Dominio con el Estándar ISO/IEC 26550 para Líneas de Productos de Software Considerando la Faceta Calidad, Tercera Conferencia de la Sociedad Venezolana de Computación, CoNCISa, pp. 107-118, Universidad de Carabobo (UC), Valencia, Octubre 2015.
- [HLO16a] J. Herrera, F. Losavio, and O. Ordaz, Product Line Scoping for Healthcare Information Systems Using the ISO/IEC 26550 Reference Model, ReVeCom, Vol. 3, No. 1, pp. 38-50, Junio 2016.
- [HLO+16] ASSIT 2016, J. Herrera, F. Losavio, and O. Ordaz, Product Lines Scoping using ISO/IEC 26550 Reference Model considering Software Quality, 2nd International Conference on Humanity and Social Science (ICHSS 2016), (pp. 194-200), DEStech Publications, Inc. USA.
- [HLO16b] J. Herrera, F. Losavio, and O. Ordaz, QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines based on ISO/IEC 26550, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 1, pp. 20-38, enero-junio 2016.
- [HLO16c] J. Herrera, F. Losavio, and O. Ordaz, Diseño de Arquitecturas de Referencia para Líneas de Productos de Software Orientadas a Servicios: Revisión Documental Sistemática, ReVeCom, Vol. 3, No. 2, pp. 13-25, Diciembre 2016.
- [HLO16d] J. Herrera, F. Losavio, and O. Ordaz, Web-services reference architecture for software product lines: A quality-driven approach, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 2, pp. 7-21, julio-diciembre 2016.

- [ISO15] ISO/IEC NP 26550: Software and Systems Engineering – Reference Model for Software and Systems Product Lines. ISO/IEC JTC1/SC7, 2015.
- [Kru95] P. Kruchten, “Architectural blueprints—The “4+1” view model of software architecture,” In proceedings of the Conference on TRI-Ada’95, Anaheim, CA, USA, November 1995.
- [LOE15] Losavio, F., Ordaz, O. & Esteller, V. (2015). Refactoring-Based Design of Reference Architecture, *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS)* 5(1) pp. 32-48.
- [LOJ16] F. Losavio, O. Ordaz, and S. Jean, Ontological approach to derive product configurations from a Software Product Line Reference Architecture, *Revista CyT, UP, Argentina, N° 16*, pp. 91-127, ISSN 1850-0870, 2016, [http://www.palermo.edu/ingenieria/pdf2016/CyT\\_16\\_07.pdf](http://www.palermo.edu/ingenieria/pdf2016/CyT_16_07.pdf)
- [PBL05] K. Pohl, G. Boeckle, and F. van der Linden, *Software Product Line Engineering - Foundations, Principles, and Techniques*, Springer Verlag, Berlin / Heidelberg, 2005.

# ***Apéndices***

---

---

## **Publicaciones**

---

En esta sección se presentan los trabajos de investigación que fueron publicados en Revistas especializadas y Editoriales realizados por el autor de esta Tesis Doctoral; por razones de espacio en cuanto al tamaño de la tesis, solo se presentan la primera página de cada una de las publicaciones. Las siguientes referencias indican el nombre de las publicaciones junto con el enlace donde se podrá acceder y descargar el trabajo completo.

- [HLM14] J. Herrera, F. Losavio, and A. Matteo, RDS de enfoques y técnicas para la construcción de arquitecturas en un contexto de líneas de productos de software, Revista Venezolana de Ciencias de la Computación ReVeCom, Vol. 1, No. 1, pp. 17-25, Junio 2014. <http://www.svc.net.ve/revecom-vol1-no1>
- [HLM+14] Herrera J., Losavio F., Matteo A., & Ordaz O. (2013). Revisión Documental Sistemática en el Ámbito de la Ingeniería del Dominio para Líneas de Productos de Software Orientados a Servicios, ReVeCom, Vol 1, No. 2, (pp. 23-33), Diciembre 2014. <http://www.svc.net.ve/revecom-vol1-no2>
- [HLO16a] J. Herrera, F. Losavio, and O. Ordaz, Product Line Scoping for Healthcare Information Systems Using the ISO/IEC 26550 Reference Model, IV Simposio Científico y Tecnológico en Computación, SCTC 2016, ISBN: 978-980-12-8407-9, Universidad Central de Venezuela, Mayo 2016, Caracas, Venezuela. ReVeCom, Vol. 3, No. 1, pp. 38-50, Junio 2016. <http://www.svc.net.ve/revecom-vol3-no1>
- [HLO+16] ASSIT 2016, J. Herrera, F. Losavio, and O. Ordaz, Product Lines Scoping using ISO/IEC 26550 Reference Model considering Software Quality, 2nd International Conference on Humanity and Social Science (ICHSS 2016), (pp. 194-200), DEStech Publications, Inc. USA. <http://www.destechpub.com>
- [HLO16b] J. Herrera, F. Losavio, and O. Ordaz, QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines based on ISO/IEC 26550, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), Vol. 6, No. 1, pp. 20-38, enero-junio 2016. <http://fundacioniai.org/raccis/v6n1/n10a3.pdf>
- [HLO16c] J. Herrera, F. Losavio, and O. Ordaz, Diseño de Arquitecturas de Referencia para Líneas de Productos de Software Orientadas a Servicios: Revisión Documental Sistemática, ReVeCom, Vol. 3, No. 2, pp. 13-25, Diciembre 2016. <http://www.svc.net.ve/revecom-vol3-no2>

- [HLO16d] J. Herrera, F. Losavio, and O. Ordaz, Web-services reference architecture for software product lines: A quality-driven approach, *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS)*, Vol. 6, No. 2, pp. 7-21, julio-diciembre 2016.  
<http://fundacioniai.org/raccis/v6n2/n11a1.pdf>

# Revisión Documental Sistemática de Enfoques y Técnicas para la Construcción de Arquitecturas en un Contexto de Líneas de Productos de Software

Juan Herrera<sup>1</sup>, Alfredo Matteo<sup>2</sup>, Francisca Losavio<sup>2</sup>  
jchr1982@gmail.com, alfredojose.matteo@gmail.com, francislosavio@gmail.com

<sup>1</sup> PFG Informática para la Gestión Social, Universidad Bolivariana de Venezuela, Caracas, Venezuela

<sup>2</sup> Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela

---

**Resumen:** Las líneas de productos de software son un enfoque centrado en el desarrollo de software basado en la reutilización, mediante el proceso de ingeniería de líneas de productos y particularmente en la fase de ingeniería del dominio es donde se diseña la arquitectura de referencia y/o la arquitectura de líneas de productos, para ello dos enfoques son utilizados en la industria, partiendo de un amplio conocimiento del dominio de la aplicación se encuentra el enfoque proactivo y partiendo de los sistemas existentes se encuentra el enfoque reactivo, es de particular interés identificar los artefactos utilizados en las estrategias utilizadas por ambos enfoques diferenciándolos ya sea por el tipo de reutilización seleccionado o por la incorporación de nuevos artefactos en cada uno de los métodos propuestos, en consecuencia una revisión documental sistemática de los estudios que abordan el tema es indispensable para determinar los alcances y limitaciones de las propuestas encontradas.

**Palabras Clave:** Líneas de Productos de Software; Arquitectura de Referencia; Arquitectura de Líneas de Productos; Proactivo; Reactivo; Revisión Documental Sistemática.

**Abstract:** The software product lines are a software development approach based on reuse, through the process of product lines engineering; particularly in the domain engineering phase is where the reference architecture and/or product line architecture is designed. For this purpose, two approaches are commonly used in industry, one based on extensive knowledge of the application domain or proactive approach, and the other one, or reactive approach, based on building on existing systems; in particular, it is interesting to identify the artifacts used in the strategies employed by both approaches, differentiating either for the selected type of reuse or incorporation of new artifacts in each of the proposed methods. In consequence, a systematic literature review of studies that address this issue is essential to determine the scope and limitations of the proposals found.

**Keywords:** Software Product Line; Reference Architecture; Product Line Architecture; Proactive; Reactive; Systematic Literature Review.

---

## I. INTRODUCCIÓN

El enfoque denominado LPS (Líneas de Productos de Software) se centra en el desarrollo de software basado en la reutilización, que se fundamenta en la identificación y captura de las similitudes y variabilidades de productos de software dentro de un dominio determinado [1]; con ello se pretende reducir el tiempo, esfuerzo, costo, complejidad y una mejora de la productividad en el desarrollo y mantenimiento de sistemas de software complejos [2].

Es a través de la ILPS (Ingeniería de Líneas de Productos de Software) donde la reutilización es planificada y aplicada [1]. La ILPS ha demostrado ser un paradigma para el desarrollo de una diversidad de sistemas intensivos de software utilizando

tecnologías existentes para la construcción a gran escala de productos adaptados a las necesidades individuales de los clientes a un menor costo, en menor tiempo y con mayor calidad [1].

El paradigma de ILPS comprende dos grandes procesos [1]: ID (Ingeniería del dominio) que es responsable de establecer la plataforma reutilizable y de definir las similitudes y variabilidades de la línea de productos. IA (Ingeniería de la aplicación) que se encarga de derivar aplicaciones de líneas de productos a partir de la plataforma establecida en la ingeniería de dominio.

Las actividades incluidas en el proceso de ID son la gestión del producto, la ingeniería de requisitos del dominio, el diseño del

dominio, la realización del dominio y las pruebas del dominio [3]. La gestión del producto se ocupa de los aspectos económicos, en particular de las estrategias de mercado relacionadas con las líneas de productos. La ingeniería de requisitos del dominio es responsable de la identificación de los requisitos comunes y variables. El diseño del dominio define la arquitectura de referencia de la línea de productos proporcionando una estructura común de alto nivel para todas las aplicaciones derivadas. La realización del dominio se ocupa del diseño detallado y la implementación de componentes de software reutilizables. Las pruebas del dominio revelan la evidencia de los defectos en los artefactos del dominio y crea artefactos de prueba reutilizables para la evaluación de las aplicaciones. De allí, para el propósito de esta investigación, solo es de interés la actividad relacionada con el diseño del dominio y particularmente con el artefacto que se deriva de esta actividad que es la *arquitectura del dominio*. El *dominio* es un campo de estudio que define un conjunto de requisitos comunes, la terminología y la funcionalidad de cualquier programa de software construido para resolver un problema.

En este contexto, se pueden encontrar dos tipos de arquitecturas de software para el dominio: AR (Arquitectura de Referencia) y ALP (Arquitectura de Líneas de Productos). Ambas apuntan a mejorar el desarrollo del software mediante la estandarización de las arquitecturas de un conjunto de sistemas de software [4].

En general, una *arquitectura de referencia* acumula conocimiento de un dominio, identifica soluciones abstractas de un problema y promueve la reutilización de las experiencias en diseño alcanzando una sólida y bien reconocida comprensión y conocimiento de un dominio, ofreciendo soluciones estandarizadas para un dominio de aplicación más amplio [4][5]. Es generalmente construida siguiendo un enfoque denominado top-down, descendente o proactivo [6].

Teniendo en cuenta la importancia de las ARs, diferentes dominios de aplicaciones, como el automotriz, la aeronáutica y la robótica, han propuesto y utilizado estas arquitecturas [4].

La *arquitectura de líneas de productos*, en cambio, es más especializada, centrándose en un subconjunto específico de los sistemas de software existentes de un dominio y ofrecen soluciones estandarizadas para una familia pequeña de sistemas [4], lo que permite crear sistemas de forma más confiable y rentable [2].

Una diferencia esencial es que las ALPs tratan con la variabilidad entre los productos, y se sitúan en un nivel de abstracción más bajo en comparación con las ARs [4].

La ALP es también una arquitectura genérica, pero es construida a partir de productos existentes, por lo tanto “hereda” de cierta forma las propiedades arquitectónicas, por ejemplo el estilo arquitectónico, y las funcionalidades relevantes de cada producto considerado. Por eso es construida mediante un enfoque denominado bottom-up, ascendente, extractivo o reactivo [6][7], que considera técnicas de reingeniería en diferentes etapas del desarrollo. Es considerado un enfoque útil en la práctica, cuando en la industria no se tiene una LPS y se quiere construir una, a partir de productos que funcionan y han sido elaborados anteriormente; es a partir de ellos que se quiere generalizar y construir la ALP. Las ARs

están generalmente situadas en un nivel de abstracción más alto en comparación con las ALPs [4].

A partir de aquí, el análisis y alcance del desarrollo de la AR y ALP en esta investigación se caracteriza por identificar los métodos y los artefactos que se utilizan y/o reutilizan para construir la arquitectura (AR o ALP) a partir de dos tipos de enfoques: proactivo y reactivo [7], su justificación se basa en las nuevas tendencias en el desarrollo de las líneas de productos como medio para la personalización masiva del software y de allí los dos enfoques para la construcción de las arquitecturas, estos modelos son: el enfoque proactivo, y el enfoque reactivo [6][7].

En consecuencia el interés de este trabajo consiste en considerar la AR o ALP, centrándose en los enfoques proactivo y reactivo.

El objetivo principal de este trabajo es explorar, organizar, resumir y analizar las principales tendencias que se han propuesto o utilizado en la construcción de las ARs y/o ALPs. Para ello, se ha adoptado y llevado a cabo una revisión sistemática bibliográfica, mediante la guía proporcionada por Kitchenham para estudiar y analizar la literatura en el campo de la ingeniería del software [8][9][10].

Este trabajo está estructurado de la siguiente manera, además de esta introducción y las conclusiones: en la Sección II se presentan los enfoques y modelos a considerar; en la Sección III se presenta el análisis de la revisión documental sistemática realizada. En la Sección IV se discuten los resultados obtenidos. Finalmente, en la Sección V se da un resumen de los aportes de la investigación.

## II. ENFOQUES Y MODELOS A CONSIDERAR

### A. Personalización Masiva de Software y sus Beneficios

La *personalización masiva* del software es un enfoque que se concentra en los medios de producción eficiente y el mantenimiento de múltiples productos de software similares, aprovechando lo que tienen en común y gestionando las variantes entre ellos. El objetivo es que el enfoque cambie del desarrollo y mantenimiento de varios productos distintos, hacia el desarrollo y mantenimiento de una sola LPS [7]. Cabe decir, que la personalización masiva es la producción a gran escala de productos adaptados a las necesidades individuales de los clientes [1].

### B. Enfoques para la Adopción de la Personalización Masiva de Software

La transición de las organizaciones a la personalización masiva de software puede lograrse con uno de los tres enfoques de amplia adopción: proactivo, reactivo, y extractivo [7].

El *enfoque proactivo* es apropiado cuando los requisitos para el conjunto de los productos necesarios y futuros, están bien definidos y estables. Este enfoque requiere un esfuerzo considerablemente para la organización, pero esto cae bruscamente una vez que la línea de producción se ha completado [1][7]. Las tareas de alto nivel que se llevan a cabo son: a) realizar un análisis de dominio y del alcance para identificar la variación que será soportada en la línea de producción; b) modelar la arquitectura de la línea de productos para soportar todos los productos en la línea de producción y c) diseñar las partes comunes y variables del sistema [7].

En el *enfoque reactivo*, la organización aumenta su línea de productos debido a la demanda de nuevos productos o cuando surgen nuevos requisitos a los productos existentes. Las tareas de alto nivel que se llevan a cabo son: a) caracterizar los requisitos para el nuevo producto en relación con los actualmente soportados en la línea de producción y b) si el nuevo producto está dentro del alcance de la línea de producción actual, se crea la definición del producto, en caso contrario, se amplía el alcance de la línea de producción actual para incluir los nuevos requisitos [7].

En el *enfoque extractivo*, la organización aprovecha los sistemas existentes de software mediante la extracción de lo común y lo diverso del código fuente, para sintetizarlos en una sola línea de producción. Requiere utilizar técnicas de reingeniería para reconstruir la arquitectura del sistema a partir del código fuente, o de la documentación existente. Las tareas de alto nivel que se llevan a cabo son: a) identificar elementos comunes y la variación en los sistemas existentes y b) factorizar en una sola línea de productos [7].

En esta investigación solo se utilizarán los enfoques proactivo y reactivo. El enfoque extractivo es considerado como similar al enfoque reactivo, ya que en ambos se reutilizan los sistemas existentes [1]: sistemas de software ad hoc y/o LPS presentes en la organización.

Las tareas que van a servir de base en el proceso de la toma de decisiones para determinar en cada uno de los estudios si el método de construcción de la arquitectura utiliza el enfoque proactivo (similar al enfoque top-down, según el proceso de desarrollo de software tradicional [11]) o por el contrario, utiliza el enfoque reactivo (similar al enfoque bottom-up [11]) son:

Se considera que el método pertenece al enfoque proactivo si describe las siguientes tareas en este orden: 1) análisis del dominio, 2) definición del alcance determinado por la variedad de productos que la organización considera que van pertenecer a ese dominio y 3) el modelado de la arquitectura.

Se considera que el método pertenece al enfoque reactivo si describe las siguientes tareas en este orden: 1) elige el conjunto de sistemas existentes que formarán parte de la línea de productos, 2) identifica las similitudes y diferencias entre los sistemas existentes y 3) el modelado de la arquitectura.

### III. REVISIÓN DOCUMENTAL SISTEMÁTICA

La revisión sistemática se realizó en el contexto de la AR y ALP, con el objetivo de identificar todos los estudios relevantes relacionados con los métodos de construcción de estas arquitecturas. Se llevó a cabo en el primer trimestre del 2013. Kitchenham indica que la gestión del proceso de revisión sistemática se fundamenta en tres fases principales: Planificación de la revisión, Realización de la revisión e Informe de la revisión [8][9][10].

En esta revisión documental sistemática se pretende conocer y comprender los enfoques y/o técnicas de diseño de las ARs y ALPs, particularmente el interés se centra en si aplican el enfoque proactivo o reactivo en la construcción de estas arquitecturas, además, de determinar el nivel de reutilización con relación a métodos y artefactos arquitecturales tradicionales o por el contrario si los métodos describen nuevos métodos y/o artefactos para la construcción de estas. Por

último identificar las similitudes y diferencias entre las ARs y ALPs.

En la Tabla I se describen las fases, las etapas que incluyen cada fase y los artefactos generados en el proceso de revisión sistemática.

**Tabla I:** Proceso de Revisión Documental Sistemática

Fases	Etapas	Artefactos
Planificación de la revisión	Justificación de la necesidad de una revisión.	Protocolo
	Especificación de la(s) interrogante(s) de investigación.	
	Desarrollo del protocolo de revisión.	
Realización de la revisión	Identificación de los estudios relacionados.	Estudios aceptados
	Selección de los estudios primarios.	
	Evaluación de la calidad de los estudios.	
	Extracción de los datos.	Modelo de datos
	Síntesis de los datos.	
Informe de la revisión	Especificación de los mecanismos de difusión.	Tablas resumen
	Presentación de los resultados	

#### A. Planificación de la Revisión

Esta fase define los objetivos de la investigación y el protocolo en que la revisión se ejecutará.

1) *Identificación de la Necesidad para una Revisión Sistemática:* Como parte de un proyecto de investigación mayor que investiga la identificación y caracterización de las ARs y ALPs debido a que ambas arquitecturas tienen definidos sus roles en lo que respecta a objetivos específicos de las arquitecturas de software y en particular para el desarrollo de las LPS. La identificación y comprensión de enfoques y/o técnicas existentes que apoyan la construcción de estas arquitecturas es relevante para comprender sus alcances y limitaciones.

2) *Desarrollo del Protocolo:* El protocolo aborda las preguntas de investigación, las estrategias de búsqueda, criterios y procesos de selección, los criterios de evaluación de la calidad, el modelo y proceso de extracción de datos, y el plan de análisis de los datos. En otras palabras, el protocolo es un plan o conjunto de pasos a seguir en un estudio.

Las preguntas de investigación tienen el objetivo de encontrar todos los estudios para comprender y resumir las evidencias acerca de los enfoques propuestos o utilizados para construir las ARs y ALPs, las siguientes PIs (Preguntas de Investigación) son consideradas:

- PI1. ¿Cuáles son los métodos utilizados para el diseño de las ARs y ALPs tomando en consideración el enfoque proactivo (top-down) o reactivo (bottom-up) como estrategias para el diseño de las arquitecturas?
- PI2. ¿Los métodos reutilizan o adaptan algún patrón, estilo o modelo arquitectónico tradicional o describen nuevos métodos o artefactos para la construcción de las arquitecturas?



- PI3. ¿Cuáles son las diferencias encontradas entre las ARs y las ALPs?

Adicionalmente, otras preguntas fueron formuladas:

- PI2.1. ¿Cuál es la forma o estrategia de reutilización más utilizada?
- PI2.2. ¿Cuál de los dos modelos (proactivo y reactivo) es el más utilizado?
- PI2.3. ¿Cuáles otros aspectos fueron evidenciados en los métodos analizados?

Con relación a la estrategia de búsqueda, primero se establecieron las fuentes de búsqueda para encontrar los estudios primarios, particularmente se seleccionó el motor de búsqueda ResearchGate<sup>1</sup> como la única fuente de búsqueda por ser una herramienta de investigación científica, que permite a los investigadores buscar contenido en revistas arbitradas, conferencias, workshop y repositorios institucionales con información relevante para la realización de la revisión sistemática.

Para establecer la estrategia de búsqueda, teniendo en cuenta las preguntas de investigación, inicialmente se identificaron las principales palabras clave “Reference Architecture” y “Product Line Architecture”. En la Tabla II se encuentran todas las expresiones de búsqueda utilizadas en la revisión sistemática.

**Tabla II:** Expresiones de Búsqueda en la Revisión Sistemática

Fuente consultada = “http://www.researchgate.net”		
Expresiones de búsqueda	Hits	Aceptados
(“product line architecture” OR “product line architectures”) AND (“approach”)	91	29
(“product line architecture” OR “product line architectures”) AND (“method” OR “methods”)	48	
(“product line architecture” OR “product line architectures”) AND (“proactive”)	5	
(“product line architecture” OR “product line architectures”) AND “reactive”	2	
(“product line architecture” OR “product line architectures”) AND “top-down”	0	
(“product line architecture” OR “product line architectures”) AND “bottom-up”	0	
<b>Total</b>	<b>146</b>	
(“reference architecture” OR “reference architectures”) AND (“approach”)	210	40
(“reference architecture” OR “reference architectures”) AND (“method” OR “methods”)	100	
(“reference architecture” OR “reference architectures”) AND “proactive”	1	
(“reference architecture” OR “reference architectures”) AND “reactive”	2	
(“reference architecture” OR “reference architectures”) AND “top-down”	3	
(“reference architecture” OR “reference architectures”) AND “bottom-up”	3	
<b>Total</b>	<b>319</b>	
<b>Total general</b>	<b>465</b>	<b>69</b>

Con relación a los criterios de inclusión y exclusión. Criterios por los cuales los estudios serán evaluados para decidir si deben ser seleccionados o no en el contexto de la revisión sistemática.

Por lo tanto, los CIs (Criterios de Inclusión) que se utilizan para incluir los estudios relevantes en esta revisión sistemática fueron:

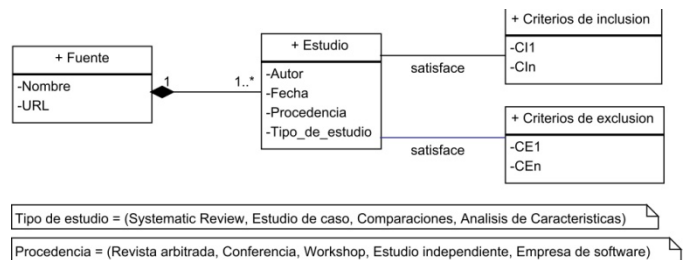
- CI1: El estudio define AR y/o ALP en el ámbito de las LPS.
- CI2: El estudio propone o emplea un método para construir las ARs y/o ALPs en el ámbito de las LPSs.

Los CEs (Criterios de Exclusión) que se utilizan para excluir los estudios que no contribuyen a responder a las interrogantes de la investigación son:

- CE1: El estudio no propone o utiliza el modelo proactivo (top-down) y/o reactivo (bottom-up) para construir las ARs y/o ALPs;
- CE2: El estudio donde su texto completo no esté disponible o solo aparece el resumen; y
- CE3: El estudio que esté escrito en un idioma diferente al inglés o español.

Con respecto a la extracción de datos y método de síntesis. Se prevé la construcción de tablas para la obtención de datos relacionados con las preguntas de investigación, en caso de que sea aplicable.

En la selección inicial de las publicaciones de entre los resultados de búsqueda, se dio lectura a los títulos y resúmenes de las publicaciones encontradas. En la Figura 1 se muestra el modelo de datos inicial utilizado en la revisión sistemática. Las publicaciones fueron almacenadas en carpetas independientes de acuerdo a la expresión de búsqueda utilizada, de esta forma facilitar la ubicación de los estudios para realizar el análisis con mayor fluidez y comodidad.



**Figura 1:** Modelo de Datos Inicial para la Revisión Sistemática

De las 69 publicaciones definitivamente seleccionadas se procede a la extracción de la información relevante utilizando el modelo de datos que será utilizado en las fases de realización e informe de la revisión representado en la Figura 2, y que incluye la lectura de sus contenidos. En la recolección de datos, las citas textuales de los artículos analizados se conservarán en el idioma original (inglés) para evitar el posible sesgo al traducir y posteriormente parafrasear según la interpretación del investigador sobre la cita leída. Esto permite una vez realizados los aportes a la investigación, poder referirse a las citas por posibles malas interpretaciones de las mismas, para validar y para futuras referencias. No obstante, al final de la revisión sistemática los resultados finales reflejados en la fase de informe de la revisión se colocarán en español que es el idioma nativo del investigador.

<sup>1</sup> <http://www.researchgate.net>

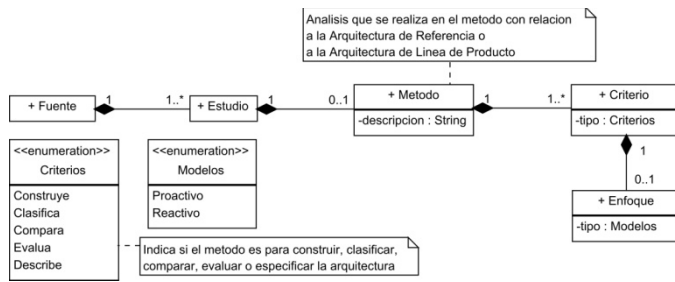


Figura 2: Modelo de Datos para el Vaciado de los Datos

Para estandarizar la forma en que la información estará representada, se definió un formato para recopilar datos de los estudios seleccionados. La Tabla III muestra el formato que será utilizado para registrar los resultados de la extracción de los datos, tomando en cuenta el modelo de datos de la Figura 2.

Tabla III: Formato para la Recolección de Datos

Criterio	Estudio		Método	TA	Asuntos de interés en relación a los artefactos utilizados por el método		
	Año	Ref			Reutiliza y adapta	Reutiliza tal cual	Agrega artefacto

La Tabla IV describe las propiedades de extracción que serán utilizados a lo largo de la revisión sistemática.

Tabla IV: Propiedades de Extracción

Propiedad	Valor	Descripción
Criterios	Construye	Construyen la arquitectura bajo el enfoque reactivo (bottom-up); o proactivo (top-down).
	Clasifica	Clasifican las arquitecturas
	Compara	Comparan métodos de construcción arquitectural.
	Evalúa	Evalúan la calidad arquitectural.
	Especifica	Describen formalmente la arquitectura.
Estudios	Año	Año de publicación del estudio.
	Ref.	Referencia bibliográfica y/o webgráfica.
Método	Nombre	Nombre del método identificado en el estudio.
TA	Tipo de arquitectura	Indica si el método se basa en la AR o ALP.
Asuntos de interés en relación a los artefactos utilizados por el método	Reutiliza y adapta	Indica si el método, técnica, patrón, estilo, estándar o framework es reutilizado y adaptado para construir la arquitectura.
	Reutiliza tal cual	En este caso los artefactos son reutilizados sin modificarlos para construir la arquitectura.
	Agrega artefacto	En este caso se crea o propone algún nuevo método, técnica, patrón, estilo, estándar o framework para construir la arquitectura.

### B. Realización de la Revisión

En esta fase, los estudios son identificados, seleccionados y evaluados de acuerdo con el protocolo previamente establecido. Como resultado de la búsqueda en las base de datos de publicaciones con las expresiones de búsquedas, 465 estudios fueron localizados, de los cuales solo 69 estudios fueron aceptados para su análisis.

Para la selección final de los estudios aceptados se realizó un proceso filtrado que consistió en la aplicación de los criterios de inclusión y de exclusión, en la Tabla V se muestra un resumen de los resultados indicando la cantidad en números y porcentajes de los estudios y que a manera de resumen fueron

agrupados en los dos elementos fundamentales comunes en los criterios de búsqueda expresados en la Tabla II.

Tabla V: Relación de Estudios por Criterios de Búsqueda

Criterios de búsqueda	Artículos			
	Incluidos	Excluidos	Total de aciertos	% Incluidos
Reference Architecture	40	279	319	57.97
Product Line Architecture	29	117	146	42.03
<b>Total</b>	<b>69</b>	<b>396</b>	<b>465</b>	

La Tabla VI presenta la relación de los estudios cuantificados por año y tipo de publicación, de los 69 estudios que se consideraron pertinentes para esta revisión documental sistemática.

Tabla VI: Relación de Estudios por Años y Tipos de Publicación

Años	Tipo de publicación			
	Revistas	Conferencias y talleres	Otros <sup>2</sup>	Total
1998	1	1		2
1999	3	1	1	5
2000	1		1	2
2001	1		1	2
2002	1	2	1	4
2003	3		1	4
2004	3	3	1	7
2005	1	6	1	8
2006	2		1	3
2007	2	2		4
2008	3	1	1	5
2009	1	6	1	8
2010	1	1	1	3
2011	2	5	1	8
2012	2	1	1	4
<b>Total</b>	<b>27</b>	<b>29</b>	<b>13</b>	<b>69</b>

Resalta el hecho que desde el año 2004 hasta la fecha, es donde se concentra el mayor número de investigaciones, observándose un gran interés en los investigadores del área de la ingeniería de software en realizar estudios relacionados con las ARs y ALPs para LPS.

A continuación, un análisis más detallado se llevó a cabo en cada estudio incluido y el informe de la revisión.

### C. Informe de la Revisión

En esta última fase, se presentan los resultados analíticos de la revisión sistemática. La extracción de los datos obtenidos y la síntesis del conocimiento considerando cada pregunta de investigación se discute a continuación.

1) Con Relación a la Pregunta PII: ¿Cuáles métodos utilizan como estrategia el enfoque proactivo o el enfoque reactivo para el diseño de la AR y ALP?

La Tabla VII resume los enfoques o técnicas que abordan el proceso de construcción de las arquitecturas desde la perspectiva del enfoque reactivo o *bottom-up*.

<sup>2</sup> Otros se refiere a los repositorios que tienen algunas universidades donde almacenan las investigaciones de sus estudiantes

**Tabla VII:** Construyen la Arquitectura (Enfoque Reactivo)

E <sup>3</sup>	Método (autor(es)) (ref.)	TA <sup>4</sup>
E1	Extra view to the QADA (Tan et al., 2012) [12]	ALP
E2	Framework RA aplicando SOA (Duro et al., 2005) [13]	AR
E3	RA based in Reverse Engineering (Grosskurth & Godfrey, 2005) [14]	AR
E4	RA basado en patrones (Mnñoz et al., 2005) [15]	AR
E5	ARES (Architecture Reference) approach [16]	AR
E6	Empirically-Grounded RA approach (Galster & Avgeriou, 2011) [17]	AR
E7	PuLSE-DSSA approach (Bayer et al., 2004) [18]	AR
E8	Prototype of reengineering approach (Jirapanthong, 2012) [19]	ALP
E9	Layered PLA approach (Bastarrica, 2006) [20]	ALP
E10	KobrA method (Anastasopoulos, 2002) [21]	ALP
E11	FAST (Family-Oriented Abstraction, Specification, and Translation) (Harsu, 2002) [22]	ALP
E12	An MDE-based PLA (Deng, 2009) [23]	ALP
E13	RA based in Prior Systems (Pinzger et al., 2004) [24]	AR
E14	Any-time variability approach (Hoek, 2004) [25]	ALP

La Tabla VIII resume los métodos que abordan el proceso de construcción de las arquitecturas desde la perspectiva del enfoque proactivo o *top-down*.

**Tabla VIII:** Construyen la Arquitectura (Enfoque Proactivo)

E <sup>3</sup>	Método (autor(es)) (ref.)	TA <sup>4</sup>
E15	Framework ACROSeT (Iborra et al., 2009) [26]	RA
E16	ERA (E-contracting Reference Architecture) approach (Angelov & Grefen, 2008) [27]	RA
E17	Interface-Activities Model (IAM) RA (Pollard & Duke, 2005) [28]	RA
E18	ProSA-RA (Nakagawa et al., 2009) [29]	RA
E19	A component-oriented approach centered on MDE (Panunzio, 2011) [30]	RA
E20	Cross-domain RA approach (Dobrica & Niemelä, 2008) [31]	RA
E21	ASRA (Agent System Reference Architecture) approach (Nguyen et al., 2010) [32]	RA
E22	Proactive Design Method for PLAs (Oizumi et al., 2012) [33]	ALP
E23	Decision-oriented approach (Dhungana, 2007) [34]	ALP
E24	SEPA Methodology (Barber & Graser, 1999) [35]	RA
E25	NEXOF-RA approach (Stricker et al., 2010) [36]	RA
E26	Process for Managing Variability (Kim et al., 2011) [37]	ALP
E27	UML profile approach for variations (Dobrica & Niemelä, 2007) [38]	ALP
E28	Style-specific approach (Lalanda, 1999) [39]	ALP
E29	Feature oriented approach (Zhu et al., 2007) [40]	ALP

En la Tabla IX, se resumen los resultados obtenidos en las Tablas VII y VIII, mostrando que no existe una tendencia o preferencia muy marcada en utilizar el enfoque proactivo o el reactivo, independientemente si es para diseñar la AR o ALP.

2) *Con Relación a la Pregunta PI2:* Identificar si los métodos utilizan algún patrón, estilo o modelo arquitectónico en particular como artefacto para la construcción de las ARs y ALPs, particularmente en lo referente a que artefactos reutiliza y adapta, solamente reutiliza o por el contrario agrega o incorpora algún nuevo artefacto en el proceso de construcción de estas arquitecturas.

En la Tabla IX se muestra la distribución de los estudios según el enfoque de construcción caracterizado por el tipo de arquitectura y asuntos de interés relacionados con las técnicas utilizadas por el método desde el punto de vista de la reutilización de artefactos arquitectónicos existentes.

**Tabla IX:** Distribución de Estudios por Enfoque de Desarrollo y Asuntos de Interés

Enfoque o modelo	Tipo de Arquitectura			Asuntos de interés en relación a artefactos utilizados en el método		
	AR <sup>5</sup>	ALP <sup>6</sup>	Total	Reutiliza y adapta	Reutiliza tal cual	Agrega artefacto
Proactivo	9	6	15	7	6	8
Reactivo	7	7	14	7	11	3
<b>Total</b>	<b>16</b>	<b>13</b>	<b>29</b>	<b>14</b>	<b>17</b>	<b>11</b>

Los artefactos son aquellas técnicas y/o elementos arquitecturales (estilo arquitectónico, patrón arquitectónico, patrón de diseño, etc.) que usan y/o implementan los métodos para la construcción de las arquitecturas.

Al realizar un análisis de las tendencias utilizadas en la construcción de las arquitecturas en los estudios analizados, según los datos suministrados por la Tabla IX, las siguientes preguntas fueron resueltas:

PI2.1. ¿Cuál es la forma de reutilización más utilizada en la construcción de estas arquitecturas?

Es evidente que la tendencia es la de reutilizar los artefactos existentes, en algunos casos se le realizan modificaciones a estos artefactos, en otros casos se incorporan en el método tal cual como originalmente fueron definidos.

PI2.2. ¿Cuál de los dos modelos (proactivo y reactivo) es el más utilizado en la construcción de estas arquitecturas?

La evidencia indica la tendencia en utilizar ambos enfoques, el proactivo y el reactivo por igual.

PI2.3. ¿Cuáles otros aspectos fueron evidenciados en los métodos analizados para la construcción de las arquitecturas?

La Tabla X muestra otras evidencias que reflejan el interés que tienen algunos autores, no solo en la construcción de las arquitecturas, sino también en clasificarlas, compararlas, evaluarlas y especificarlas formalmente.

**Tabla X:** Distribución de los Estudios por Criterios de Análisis

Criterios	TA		Total
	AR	ALP	
Construyen	16	13	29
Clasifican	3	1	4
Comparan	2	3	5
Evalúan	9	10	19
Especifican	0	3	3

Se destaca que los estudios [12][18][21][27][29][33][41], satisfacen dos de los criterios, particularmente describen un método para construir la arquitectura y luego la evalúan a través de características de calidad. Solo 1 estudio [42] mostró un enfoque para clasificar y evaluar las ARs.

<sup>3</sup> Estudios

<sup>4</sup> Tipo de arquitectura

<sup>5</sup> Arquitectura de referencia

<sup>6</sup> Arquitectura de líneas de productos

3) *Con Relación a la Pregunta "PI3": ¿Cuáles son las semejanzas y diferencias entre la AR y la ALP?*

La Tabla XI resume las similitudes y las diferencias entre ambas arquitecturas [4][12][14][16][17][19][30][31][36][43][44][45][46][47][48][49].

**Tabla XI:** Semejanzas y Diferencias entre AR y ALP

Semejanzas
Garantiza la interoperabilidad de los sistemas a través de la estandarización.
Facilita la creación de instancias de nuevas arquitecturas.
La flexibilidad y la interoperabilidad son dos de los requisitos no funcionales que guían su diseño.
Determinan el ámbito de la línea de productos que limita los productos que se pueden construir.
Se producen durante la actividad del diseño del dominio en la fase de ingeniería del dominio.
Comparten una arquitectura y un conjunto de componentes reutilizables entre una familia de productos.
Describe elementos básicos de la arquitectura de software para los sistemas que se derivan del mismo dominio.
Son arquitecturas de software que no contiene detalles de implementación.
Diferencias
La AR requiere la existencia de dominios bastantes maduros debido a su alto nivel de abstracción.
AR tiene mayor nivel de abstracción con respecto a la ALP.
Las ALP tienen que ver con la variabilidad entre los productos.
Las ALPs representan un grupo de sistemas que comparten elementos similares, pero producidos independientemente por una sola organización.
Las ARs representan el espectro visible y futuro (conjunto, gama, rango, variedad) de los sistemas en un dominio.
Las ALPs son menos abstractas que la RA, pero más abstractas que las arquitecturas concretas, es decir se sitúan en un nivel intermedio entre la AR y el producto concreto.
Las ALP son un tipo de arquitectura de referencia.
La ALP puede ser instanciada en una arquitectura de software mediante el refinado, extendiéndola, y en particular, especificando decisiones de implementación.
El rol de las ARs es capturar el conocimiento de las arquitecturas existentes. Sobre la base de la misión, visión y estrategias, y sobre las necesidades futuras de los clientes, y proporciona orientación a las múltiples organizaciones que evolucionan o crear nuevas arquitecturas.
La AR se crea mediante la captura de los elementos esenciales de las arquitecturas existentes y teniendo en cuenta las necesidades futuras y oportunidades, que van desde las tecnologías específicas, a los patrones de los modelos de negocio y segmentos de mercado.
La ALP especifica la descomposición de un sistema en subsistemas, la interacción entre subsistemas y la distribución de funcionalidad entre ellas.

#### IV. DISCUSIÓN

Los resultados de la revisión sistemática indican que, a pesar del gran interés en las AR y en las ALPs, la mayoría de los estudios aplican diferentes criterios en la forma o manera de concebir o diseñar la AR y las ALPs por lo cual ampliar la investigación en cuanto a la unificación de criterios en esta área sería de gran aporte en la comunidad de desarrolladores.

La evidencia indica que el término AR no se origina por el surgimiento de la metodología de ingeniería de líneas de productos de software como otro enfoque para la reutilización y la mantenibilidad de familias de productos de software, ya anteriormente se hablaba de las arquitecturas de referencias, esto se ve reflejado por los 25 estudios de los 69 que abordan las AR y no las relacionan con el SPL de forma explícita.

Las LPS es otro enfoque que hace énfasis en la reutilización de los activos de software presentes en la organización, la

preferencia es la de reutilizar los artefactos existentes (estilo arquitectónico, patrón, etc.), modificándolos o no, y en pocos casos incorporando nuevos artefactos en los métodos de desarrollo definidos en la organización.

De las políticas de desarrollo de software en la organización a nivel industrial, la toma de decisión de utilizar el enfoque proactivo o el reactivo dependerá de: si se quiere construir una nueva LPS sin considerar los productos existentes, modificar una LPS existente o reutilizar los productos existentes y de allí construir una nueva LPS.

Aunque hay autores que indican que la AR y la ALP son diferentes en cuanto al objetivo para el cual fueron concebidas, concretamente en el alcance del dominio determinado por la organización, en la mayoría de los estudios analizados, los autores utilizan ambos términos de forma equivalente.

#### V. CONCLUSIONES

La principal contribución de este trabajo fue presentar una perspectiva detallada sobre los enfoques de desarrollo proactivo y reactivo propuestos para la construcción de las AR y PLA, así como determinar la forma de reutilización de los artefactos en la construcción de estas arquitecturas.

La elección de parte de la organización del enfoque proactivo o reactivo como estrategia para el desarrollo de la arquitectura, dependerá de los tiempos de entrega de los productos y de la capacidad de inversión disponible en la organización; si la organización dispone de los recursos suficientes entonces tendrá la capacidad de abordar el enfoque proactivo que requiere gran cantidad de talento humano y de tiempo para diseñar la arquitectura; si la organización tiene tiempo en el mercado produciendo productos de software, el enfoque reactivo suele ser el elegido por esta, sin embargo se ha de tener en cuenta que los productos existentes no fueron inicialmente pensados o concebidos bajo el esquema de LPS, por lo cual el proceso de reingeniería necesario puede ser considerable, dependiendo de la madurez que exista en la organización en sus procesos de desarrollo de software y del uso de estándares de calidad en los productos presentes.

Existe una tendencia a la reutilización de artefactos existentes, como por ejemplo: estilos y patrones arquitectónicos, patrones de diseños, etc., que la incorporación de nuevos artefactos para la construcción de la AR y/o ALP.

Existe una carencia en el uso de modelos de calidad, por ejemplo el ISO 9126 o su actualización el ISO 25010, por lo cual una revisión documental sistemática que responda a la pregunta de cuáles estrategias y/o técnicas se han aplicado para proveer de calidad a las ARs y/o ALPs, y a partir de allí desarrollar un modelo de calidad para estas arquitecturas.

Esta revisión sistemática provee a la comunidad de desarrolladores de un insumo que apoye la toma de decisiones en la elección de utilizar el enfoque proactivo o reactivo para el abordaje de las LPSs como práctica organizacional para el desarrollo de sus productos de software, así como utilizar la AR o ALP como punto de partida para el diseño, dependiendo de la madurez que tenga la organización en cuanto a la metodología utilizada para el desarrollo de software y de la visión empresarial en relación a los futuros desarrollos.

## REFERENCIAS

- [1] K. Pohl, G. Boeckle, and F. van der Linden, *Software Product Line Engineering - Foundations, Principles, and Techniques*, Springer Verlag, Berlin / Heidelberg, 2005.
- [2] E. Nakagawa, P. Oliveira, and M. Becker, *Exploring the Use of Reference Architectures in the Development of Product Line Artifacts*, in proceedings of the 15th International Software Product Line Conference, Vol 2, ACM, 2011.
- [3] B. Penzenstadler, *Evaluation and Testing of Approaches to Support Variants in Model-Based Design for Software Product Lines*, University of Passau, Thesis, 2006.
- [4] E. Nakagawa, P. Oliveira, and M. Becker, *Reference Architecture and Product Line Architecture: A Subtle but Critical Difference*, Software Architecture. Springer Berlin Heidelberg, pp. 207-211, 2011.
- [5] M. Guessi, L. B. R. de Oliveira, and E. Y. Nakagawa., *Representation of Reference Architectures: A Systematic Review*, Department of Computer Systems, University of São Paulo, SP, Brazil, 2011.
- [6] A. Rashid, J.C. Royer, and A. Rummler, *Aspect-Oriented Model-Driven Software Product Lines. The AMPLE Way*, Cambridge University Press, Cambridge, 2011.
- [7] C. W. Krueger, *Easing the Transition to Software Mass Customization*, in Software Product-Family Engineering, Springer Berlin Heidelberg, 2002, pp. 282-293.
- [8] B. Kitchenham, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*, Version 2.3, EBSE Technical Report, Software Engineering Group, School of Computer Science and Mathematics, Keele University, UK and Department of Computer Science, University of Durham, UK, 2007.
- [9] B. Kitchenham, O. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, *Systematic Literature Reviews in Software Engineering, a Systematic Literature Review*, Information and software technology, vol. 51, no. 1, pp. 7-15, 2009.
- [10] B. Kitchenham, *Procedures for Performing Systematic Reviews*, Keele University and National ICT Australia Ltd, Tech. Report TR/SE-0401 and NICTA TR 0400011T.1, 2004.
- [11] B. Basem El-Haik and A. Shaout, *Software Design for Six Sigma: A Roadmap for Excellence*, John Wiley and Sons, 2010.
- [12] L. Tan, Y. Lin, and H. Ye. *Quality-Oriented Software Product Line Architecture Design*, Journal of Software Engineering and Applications, vol. 5, no. 7, pp. 472-476, 2012.
- [13] N. Duro, F. Moreira, J. Rogado, J. Reis, and Nestor Peccia, *Developing a Reference Architecture for the Ground Segment Software*, IEEEAC paper #1134, Version 1.14, IEEE, 2005.
- [14] A. Grosskurth and M. W. Godfrey, *A Reference Architecture for Web Browsers*, in proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05), pp. 661-664, 2005.
- [15] J. Muñoz, J. Muñoz, F. J. Alvarez, and M. Romo, *Aplicación de un Método para Generar una Arquitectura de Referencia que Integre Sistemas Heredados y Bases de Datos*, Universidad Autónoma de Aguascalientes, México, 2005.
- [16] W. Eixelsberger, *Recovery of a Reference Architecture, A Case Study*, in proceedings of the third International Workshop on Software Architecture, ACM, pp. 33-36, 1998.
- [17] M. Galster and P. Avgeriou, *Empirically-Grounded Reference Architectures, A Proposal*, in proceedings of the joint ACM SIGSOFT conference--QoSA and ACM SIGSOFT symposium--ISARCS on QoSA and architecting critical systems--ISARCS, ACM, pp. 153-158, 2011.
- [18] J. Bayer, T. Forster, D. Ganesan, J. F. Girard, I. John, J. Knodel, R. Kolb, and D. Muthig, *Definition of Reference Architectures Based on Existing Systems, Lifecycle and Process for Family Integration*, Fraunhofer IESE-Report no. 2004, vol. 34.
- [19] W. Jirapanthong, *Experience on Re-Engineering Applying with Software Product Line*, International Journal of Computer Engineering Science (IJCES), vol. 2, no. 5, 2012.
- [20] M. C. Bastarrica, N. Hirschfeld-Kahler, and P. O. Rossel, *Product Line Architecture for a Family of Meshing Tools*, in Reuse of Off-the-Shelf Components, Springer Berlin Heidelberg, pp. 403-406, 2006.
- [21] M. Anastasopoulos, C. Atkinson, and D. Muthig, *A Concrete Method for Developing and Applying Product Line Architectures*, in Objects, Components, Architectures, Services, and Applications for a Networked World, Springer Berlin Heidelberg, pp. 294-312, 2003.
- [22] M. Harsu, *FAST Product-Line Architecture Process*, Tampere University of Technology, 2002.
- [23] G. Deng, D. C. Schmidt, A. Gokhale, J. Gray, Y. Lin, and G. Lenz, *Evolution in Model-Driven Software Product-Line Architecture*, Designing Software-Intensive Systems: Methods and Principles, 2008.
- [24] M. Pinzger, H. Gall, J. F. Girard, J. Knodel, C. Riva, W. Pasman, C. Broerse, and J. G. Wijnstra, *Architecture Recovery for Product Families*, in Software Product-Family Engineering, Springer Berlin Heidelberg, pp. 332-351, 2004.
- [25] A. van der Hoek, *Design-Time Product Line Architectures for Any-Time Variability*, Science of Computer Programming, vol. 53, no. 3, pp. 285-304, 2004.
- [26] A. Iborra, D. Alonso, F. J. Ortiz, J. Pastor, P. Sánchez, and B. Álvarez, *Design of Service Robots - Experiences using Software Engineering*, Robotics & Automation Magazine, IEEE, vol. 16, no. 1, pp. 24-33, 2009.
- [27] S. Angelov and P. Grefen, *An E-Contracting Reference Architecture*, Journal of Systems and Software, vol 81, no. 11, pp. 1816-1844, 2008.
- [28] J. Pollard and R. Duke, *A Reference Architecture for Instructional Educational Software*, in ACM International Conference Proceeding Series, pp. 43-52, 2005.
- [29] E. Y. Nakagawa, E. F. Barbosa, and J. C. Maldonado, *Exploring Ontologies to Support the Establishment of Reference Architectures, An Example on Software Testing*, in European Conference on Software Architecture WICSA/ECSA 2009, pp. 249-252, 2009.
- [30] M. Panunzio, *Definition, Realization and Evaluation of a Software Reference Architecture*, Technical Report UBLCS-2011-07, Department of Computer Science, University of Bologna, Italy, 2011.
- [31] L. Dobrica and E. Niemelä, *An Approach to Reference Architecture Design for Different Domains of Embedded Systems*, in Software Engineering Research and Practice, pp. 287-293, 2008.
- [32] D. N. Nguyen, K. Usbeck, W. M. Mongan, C. T. Cannon, R. N. Lass, J. Salvage, W. C. Regli, I. Mayk, and T. Urness, *A Methodology for Developing an Agent Systems Reference Architecture*, in Agent-Oriented Software Engineering XI, Springer Berlin, pp. 177-188, 2011.
- [33] W. N. Oizumi, A. C. Contieri, G. G. Correia, T. E. Colanzi, S. Ferrari, I. M. S. Gimenes, E. A. Oliveira, A. F. Garcia, and P. C. Masiero, *On the Proactive Design of Product-Line Architectures with Aspects, An Exploratory Study*, in proceedings of the 36th Computer Software and Applications Conference (COMPSAC 2012), pp. 273-278, 2012.
- [34] D. Dhungana, R. Rabiser, and P. Grünbacher, *Decision-Oriented Modeling of Product Line Architectures*, in the 6th Working IEEE/IFIP Conference of Software Architecture (WICSA'07), Mumbai, India, January 2007.
- [35] K. S. Barber, T. Graser, J. Holt, and C. Silva, *Representing Domain Reference Architectures by Extending the UML Metamodel*, in 20th International Conference on Software Engineering and Knowledge Engineering, San Francisco, USA, July 1999.
- [36] V. Stricker, K. Lauenroth, P. Corte, F. Gittler, S. De Panfilis, and K. Pohl, *Creating a Reference Architecture for Service-Based Systems: A Pattern-Based Approach*, in proceeding of Towards the Future Internet - Emerging Trends from European Research, 2010.
- [37] Y. G. Kim, S. Kee Lee, and S. B. Jang, *Variability Management for Software Product-Line Architecture Development*, International Journal of Software Engineering and Knowledge Engineering, vol. 21, no. 07, pp. 931-956, 2011.
- [38] L. Dobrica and E. Niemelä, *Modeling Variability in the Software Product Line Architecture of Distributed Middleware Services*, VTT Technical Research Centre of Finland, Oulu, Finland, 2007.
- [39] P. Lalanda, *Style-Specific Techniques to Design Product-Line Architectures*, Thomson-CSF Corporate Research Laboratory, Domaine de Corbeville, Orsay, France, 1999.
- [40] C. Zhu, Y. Lee, W. Zhao, and J. Zhang, *A Feature Oriented Approach to Mapping from Domain Requirements to Product Line Architecture*, in Software Engineering Research and Practice, pp. 219-225, 2007.
- [41] J. Bayer, O. Flege, and C. Gacek, *Creating Product Line Architectures*, in Software Architectures for Product Families, Springer Berlin Heidelberg, pp. 210-216, 1999.
- [42] C. Maga and N. Jazdi, *A Survey on Determining Factors for Modeling Reference Architectures*, in proceedings of OOPSLA, Florida, USA, 2009.
- [43] G. Muller and P. van de Laar, *Researching Reference Architectures and their Relationship with Frameworks, Methods, Techniques, and Tools*, 7th Annual Conference on Systems Engineering Research (CSER 2009), S08-46, Loughborough University, UK, 2009.

- [44] P. Avgeriou, *Describing, Instantiating and Evaluating a Reference Architecture: A Case Study*, Enterprise Architect Journal, 2003.
- [45] B. Graaf, H. van Dijk, and A. van Deursen, *Evaluating an Embedded Software Reference Architecture - Industrial Experience Report*, in proceedings of the 9th European Conference of Software Maintenance and Reengineering (CSMR 2005), pp. 354-363, 2005.
- [46] U. Eklund, Ö. Askerdal, J. Granholm, A. Alminger, and J. Axelsson, *Experience of Introducing Reference Architectures in the Development of Automotive Electronic Systems*, 2005.
- [47] S. Giesecke, W. Hasselbring, and C. von Ossietzky, *Taxonomy of Architectural Style Usage*, in proceedings of the 2006 conference on Pattern Languages of Programs, ACM, 2006.
- [48] B. P. Gallagher, *Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study*, CMU/AEI-2000-TN-007, Carnegie-Mellon University, Pittsburgh Software Engineering Institute, 2000.
- [49] J. Bayer, D. Ganesan, J. F. Girard, J. Knodel, R. Kolb, and K. Schmid, *Definition of Reference Architectures based on Existing Systems*, Fraunhofer IESE-Report, vol. 34, 2004.

# Revisión Documental Sistemática en el Ámbito de la Ingeniería del Dominio para Líneas de Productos de Software Orientados a Servicios

Juan Herrera<sup>1</sup>, Francisca Losavio<sup>2</sup>, Alfredo Matteo<sup>2</sup>, Oscar Ordaz<sup>3</sup>

jchr1982@gmail.com, francislosavio@gmail.com, alfredojose.matteo@gmail.com, oscarordaz55@gmail.com

<sup>1</sup> PFG Informática para la Gestión Social, Universidad Bolivariana de Venezuela, Caracas, Venezuela

<sup>2</sup> Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela

<sup>3</sup> Escuela de Matemática, Universidad Central de Venezuela, Caracas, Venezuela

---

**Resumen:** Una Revisión Documental Sistemática (RDS) permite extraer conocimiento sobre un tema de investigación; esto se hace mediante preguntas adecuadas, a partir del gran volumen de información disponible en internet. RDS se enfoca en una pregunta de investigación que trata de identificar, evaluar, seleccionar y sintetizar evidencias de investigaciones de alta calidad relevantes para esa pregunta. El objetivo principal del trabajo es realizar una RDS en el tema de la Ingeniería del Dominio (ID) de Líneas de Productos de Software Orientados a Servicios (LPSOS), quien combina los enfoques de Líneas de Productos de Software (LPS) y de Arquitecturas Orientadas a Servicios o del inglés Service-Oriented Architecture (SOA). Esta investigación pretende determinar trabajos recientes existentes en la literatura en cuanto a actividades, artefactos y técnicas para un proceso de desarrollo del dominio, según Björner y de análisis del dominio según Pohl et al.. Como resultado de esta revisión se detectó, combinando los enfoques mencionados, que las fases principales de un proceso general de ID para LPSOS son: análisis, diseño e implementación, inspiradas en las fases definidas por Pohl et al. para la ID en un contexto de LPS. Dentro de estas fases se identificaron las actividades, artefactos y técnicas más usadas en la práctica. Estos resultados serán utilizados para definir un proceso sistemático de Análisis de Dominio para LPSOS, que además contemplará el tratamiento de requisitos de calidad, aspecto relevante para un contexto de producción industrial de software y no tratado sistemáticamente en los enfoques revisados.

**Palabras Clave:** Ingeniería del Dominio; Línea de Productos de Software; Arquitectura Orientada a Servicios; Línea de Productos de Software Orientadas a Servicios; Revisión Documental Sistemática; Calidad del Software.

**Abstract:** A Systematic Literature Review (SLR) allows to extract knowledge on a research topic; this is done through appropriate questions, from the large volume of information available on the internet. SLR is focused on a research question that tries to identify, appraise, select and synthesize high quality research evidence relevant to that question. The main goal of this work is to perform an SLR on the subject of Domain Engineering (DE) of Service-Oriented Software Products Line (SOSPL), who combines the approaches of Software Products Line (SPL) and Service-Oriented Architecture (SOA). This research aims to determine existent recent works in the literature regarding activities, artifacts, and techniques for development of the domain, according to Björner and domain analysis according to Pohl et al.. As a result of this review, main phases of a general DE process of for SOSPL were identified: analysis, design and implementation, inspired in the phases defined by Pohl et al. for DE in an SPL context. Within these phases activities, artifacts and techniques most used in practice were identified. These results will be used to define a systematic process of Domain Analysis for SOSPL, which also includes the treatment of quality requirements, relevant aspect for industrial software production the context, not systematically treated in the revised approaches.

**Keywords:** Domain Engineering; Software Products Line; Service-Oriented Architecture; Service-Oriented Software Products Line; Systematic Literature Review; Software Quality.

---

## I. INTRODUCCIÓN

El desarrollo de sistemas de software de gran complejidad y de gran escala representa un desafío para los investigadores y desarrolladores en el área de la IS (Ingeniería de Software); surge entonces la tendencia al desarrollo basado en

componentes reutilizables, para de ahí configurar nuevos sistemas con rapidez y de más bajo costo [1][2][3].

Desde finales de los '60, y aun más en la década de los '90, el paradigma de las LPS (Líneas de Productos de Software) ha

cochado impulso en la industria del software. En lugar de desarrollar sistemas a partir de cero, estos deben ser contruidos a partir de partes reutilizables. En vez de componer un sistema siempre de la misma manera, éste debería ser adaptado a las necesidades de los clientes (del inglés, “stakeholders”), donde éstos pueden elegir entre un conjunto de opciones de configuración. El enfoque de LPS permite la construcción de soluciones individuales basadas en un repositorio de componentes de software reutilizables. La necesidad de proporcionar soluciones individuales responde a diversas necesidades en el software con respecto a la funcionalidad, las plataformas destino y las propiedades no funcionales, como por ejemplo rendimiento y espacio de memoria [4].

Las LPS prometen distintos beneficios [4][5], de los cuales los más importantes son: a) adaptabilidad a las necesidades del cliente, b) reducción de costos, c) mejora de la calidad global y d) disminución del tiempo de comercialización. Este enfoque representa un gran desafío para los investigadores y desarrolladores en el área de la IS, principalmente por la tarea de desarrollar artefactos (activos o del inglés “assets”) suficientemente genéricos para pensar en su reutilización y en la configuración de nuevos sistemas contextualizados acorde con las necesidades de los clientes [2][3].

Para el abordaje adecuado de este enfoque, la ILPS (Ingeniería de Líneas de Productos de Software) ofrece métodos y técnicas eficaces para la reutilización sistemática en el desarrollo de software con el fin de: a) apoyar arquitecturas de software configurables y b) permitir la personalización masiva de los sistemas de software [6][7]. Ha sido reconocida como un enfoque exitoso para la gestión de la variabilidad y la ingeniería de la reutilización. La ILPS consta de dos ciclos de vida principales: ID (Ingeniería del Dominio) e IA (Ingeniería de la Aplicación) [8]. La ID abarca el análisis y la identificación del ámbito de aplicación de la línea de productos, que incluye la captura de todo el conocimiento sobre el dominio de interés a través del modelado de partes comunes (del inglés “commonality”) y de puntos de variación (del inglés “variability points”) que están presentes en la arquitectura de la línea de productos.

Al considerar las fases del proceso de ID desde la perspectiva de las LPS [8], el *análisis del dominio* permite la captura y análisis del conocimiento sobre el dominio. El resultado principal del análisis del dominio es un artefacto denominado *modelo del dominio*. Este modelo representa las propiedades comunes y variables de la familia de sistemas en el dominio y las relaciones entre ellas. Conceptos del dominio y sus relaciones mutuas son analizados y modelados. Los conceptos del dominio representan el vocabulario del dominio y pueden ser representados por ontologías. Cada concepto luego es extendido por sus características comunes y variables y las dependencias entre ellos. Las características variables determinan el espacio de configuración para la familia de sistemas que se resuelve luego en la IA. El *diseño del dominio* produce la arquitectura abstracta genérica o de referencia (del inglés: Product Line Architecture, PLA) o RA (Reference Architecture) para la familia de sistemas, de acuerdo con los patrones arquitectónicos comúnmente aceptados (modelo-vista-controlador, en capas, etc.). Un ciclo completo de ID

construye el conocimiento de las necesidades específicas de los diferentes grupos de interés para los que se lleva a cabo la adaptación y configuración de la LPS.

Por otra parte, la *Arquitectura Orientada a Servicios* (del inglés: Service Oriented Architecture, SOA) [9][10] es una arquitectura centrada en la comunicación en redes a través de servidores Web que actúan como intermediarios para satisfacer demandas de servicios por clientes distribuidos en ubicaciones geográficamente distantes. SOA ha cobrado enorme auge en la práctica reciente de desarrollo de software rápido, a bajo costo y es ampliamente tratada actualmente a nivel de investigación en la ingeniería de software; constituye un paradigma para el diseño, desarrollo e implementación de aplicaciones de computación distribuida e independiente de la tecnología basada en estándares para el descubrimiento, el enlazado y ensamblaje de servicios de software débilmente acoplados [11]. Un *servicio* se define como una unidad discreta de la funcionalidad de negocio que está disponible a través de un contrato de servicios [12]; puede ser visto como un componente de software reutilizable. SOA es una arquitectura para la construcción de soluciones empresariales basadas en servicios [13]. Particularmente se ocupa de la construcción independiente de servicios alineados al negocio que pueden ser combinados en procesos de negocio de alto nivel y soluciones computacionales en el contexto empresarial. El valor real de SOA no es solo la provisión adecuada de los servicios, sino más bien es cuando los servicios reutilizables se combinan para crear procesos de negocios ágiles y flexibles. SOA ha demostrado su rentabilidad en el desarrollo de sistemas de software interoperables, reutilizables, adaptables y de rápido desarrollo, por lo cual existen importantes ventajas mutuas en la convergencia de SOA y LPS [1][14][15][16][17][18][19], dando origen al enfoque de LPSOS (Líneas de Productos de Software Orientadas a Servicios).

En todos los puntos antes tratados, el aseguramiento de la calidad del sistema como producto de software, es una actividad crucial para el éxito, ya que se ubican en un contexto de producción industrial; por lo tanto, cuando se trata del desarrollo de LPS, la reutilización masiva de los activos de software hace que los atributos de calidad (propiedades medibles de un artefacto de software) impacten en la calidad de todos los productos de la LPS [20]. La evaluación de la calidad es una tarea difícil en los sistemas de software monolíticos, y lo es aún más cuando se trata de LPSOS, ya que requiere analizar las características de calidad de una LPS bajo SOA, considerando propiedades funcionales y no funcionales comunes y variantes [21][22]. Por otra parte, la evaluación de la calidad es una parte esencial en la optimización y configuración de la LPS, ya que debe proporcionar métricas cuantitativas globales para la calidad de la familia de sistemas, generalmente basadas en la especificación de la arquitectura, como por ejemplo: costo, desempeño, disponibilidad, confiabilidad, escalabilidad y seguridad [23], pero también específicas para un sistema miembro de la familia [24].

Ahora bien, en un contexto de LPSOS, particularmente durante el ciclo de ID, la determinación de la calidad del servicio (del inglés: Quality of Service, QoS) es fundamental, debido a que varía acorde a características individuales, por lo cual los ingenieros del dominio deben asegurar que los servicios que



van a formar parte del sistema satisfacen los rangos de valores establecidos por los QoS solicitados por las partes interesadas [23].

En los estudios anteriormente referenciados se han detectado las siguientes evidencias de problemas aún no completamente resueltos, para algunos de los cuales este trabajo pretende proporcionar una solución:

- No se describen en detalle las actividades a realizar para la adquisición del conocimiento del dominio.
- No se describen diferentes perspectivas en el dominio que puede ser analizadas, por ejemplo, las facetas del dominio, utilizadas en el enfoque de Bjorner [25].
- Los requisitos funcionales son los abordados en general, pero los requisitos no funcionales y/o requisitos de calidad son dejados de lado, desde las siguientes perspectivas [23]: características de calidad específicas de la línea de productos (mantenibilidad, complejidad, etc.) y características de calidad específica del dominio o calidad global (disponibilidad, seguridad, etc.).
- La determinación de las características de calidad no es realizada en las etapas tempranas del proceso de ID (análisis del dominio); se ha detectado que son de vital importancia para derivar de forma adecuada el conjunto de productos de la LPSOS.

Es de particular interés en el contexto de esta investigación, que se centra básicamente en la fase de Análisis del Dominio de la ID, por ser la más importante y previa a la derivación de los productos concretos de la LPS en la IA, identificar las actividades, artefactos y técnicas utilizadas en un proceso completo de análisis del dominio para el desarrollo de las LPSOS, diferenciándolas: a) por los artefactos de entrada y de salida, b) por el enfoque de diseño utilizado (proactivo o reactivo) para la arquitectura, con el fin de describir como se analiza y realiza el proceso de análisis del dominio en la ID y c) también es de especial interés identificar las técnicas (UML (Unified Modeling Language), FODA (Feature-Oriented Domain Analysis), etc.) utilizadas como notaciones para sus principales artefactos.

Una RDS (Revisión Documental Sistemática) [26] permite extraer el conocimiento sobre el estado del arte en un tema específico de investigación; este conocimiento es extraído, mediante preguntas adecuadas, a partir de un gran volumen de información disponible. En consecuencia una RDS de la literatura que aborda el tema de la ID de LPSOS es indispensable para determinar los alcances y limitaciones de las propuestas encontradas. El objetivo de este trabajo es utilizar la RDS para identificar las actividades, artefactos y técnicas comúnmente utilizadas en la ID encontrados en la literatura reciente, para luego integrarlos para definir un proceso único de análisis del dominio para LPSOS incorporando también aquellas actividades y/o artefactos faltantes.

Este trabajo está estructurado de la siguiente manera, además de esta introducción y las conclusiones: en la Sección II se presentan los enfoques a considerar; en la Sección III se presenta el análisis de la revisión documental sistemática

realizada. En la Sección IV se discuten los resultados obtenidos. Finalmente, en la Sección V se da un resumen de los aportes de la investigación.

## II. ENFOQUES A CONSIDERAR

### A. ID (Ingeniería del Dominio)

Un *dominio* es un área de conocimiento que utiliza conceptos comunes para la descripción de los fenómenos, requisitos, problemas, capacidades y soluciones. Se asocia generalmente con una terminología bien definida o parcialmente definida. Esta terminología se refiere a los conceptos básicos del dominio, sus definiciones (es decir, sus significados semánticos), y sus relaciones. También puede referirse a las conductas que se desean, prohibidas, o percibidas dentro del dominio [27]. Un *dominio de aplicación* comprende cualquier aspecto en el cual la computación pueda aplicarse. Según Bjorner [25], básicamente existen tres clases de dominios de aplicación:

- La clase de aplicaciones que pueden caracterizarse por el apoyo a la enseñanza o el estudio de un campo: el software educativo o de formación, software experimental de demostración de teoremas, o similares.
- La clase de aplicaciones que pueden caracterizarse por el apoyo al desarrollo de sistemas informáticos propios: compiladores, sistemas operativos, sistemas de gestión de bases de datos, sistemas de comunicación de datos, etc.
- La clase de aplicaciones que pueden caracterizarse por el apoyo al desarrollo de software comercial o industrial, esto incluye los sistemas de información en general y las aplicaciones empresariales, entre otras.

Particularmente, la última clase de estos dominios de aplicación son los de interés en el trabajo de Bjorner [25]. Por ello, la selección del trabajo de este autor es argumentada por la razón que las aplicaciones implementadas mediante SOA fueron diseñadas con la intención de apoyar a través de esta arquitectura basada en un intermediario para la comunicación, los dominios de aplicación relacionados con los sistemas de información gerencial y/o empresarial que utilizan un estilo clásico de capas, como por ejemplo, capa de presentación, lógica y datos.

La ID es el conjunto de actividades cuyo objetivo es desarrollar, mantener y administrar la creación y evolución de los dominios [25][27]. Ha sido de especial interés para los sistemas de información y de las comunidades de ingeniería de software, por varias razones: a) la necesidad de mantener y reutilizar los conocimientos existentes, b) la necesidad de gestionar los crecientes requisitos de la variabilidad de la información y de los sistemas de software, y c) la necesidad de obtener, formalizar, y compartir conocimientos especializados en diferentes ámbitos, en evolución [27].

La ID como disciplina tiene una importancia práctica, ya que debe proporcionar los métodos y técnicas que pueden ayudar a reducir el tiempo de entrega al mercado, los costos de desarrollo y los riesgos en los proyectos, además que deben ayudar a mejorar la calidad del producto y el rendimiento sobre

una base consistente [27]. La ID es utilizada, investigada y estudiada en diversas áreas, principalmente:

- ILPS (Ingeniería de línea de productos de software)
- ILED (Ingeniería de lenguaje específico de dominio)
- Modelado conceptual.

Según Bjorner [25], antes de diseñar el software, se deben comprender sus requerimientos, y antes que se puedan desarrollar los requerimientos, se debe comprender el dominio de la aplicación en que se va a construir. Para comprender el dominio de la aplicación, se debe realizar una etapa de análisis del dominio desde las perspectivas de las facetas del dominio bajo la óptica de los grupos de interés. Es decir, cada grupo de interés tendrá características particulares en cada una de las facetas que conforma el dominio de la aplicación. La ID reduce el esfuerzo de trabajo relacionado con algunos aspectos de la IS, especialmente los vinculados con la ingeniería de requisitos.

La Tabla I muestra la relación entre las fases de la ID expresada por Bjorner [25] y las fases expresadas por Pohl et al. [8]. Esto se hace con la intención de mostrar como los conceptos expresados por Bjorner en ID, complementan los de Pohl bajo el enfoque LPS. El enfoque propuesto por Bjorner fue seleccionado debido a que las fases del proceso para el desarrollo del dominio son claramente definidas y argumentadas, y debido al gran nivel de detalle en la descripción de las actividades realizadas, expresadas mediante las facetas (o vistas) del dominio.

El análisis del dominio basado en las facetas del dominio desde las perspectiva de los stakeholders, permitirá conocer el dominio de la aplicación al considerar de forma más adecuada los aspectos funcionales y no funcionales (requisitos de calidad) que deben ser satisfechas desde diferentes vistas.

**Tabla I:** Relación entre las Fases de la ID según Dines Bjorner [25] y Pohl et al. [8]

Perspectiva de Bjorner [25]	Perspectiva de Pohl et al. [8]
Proceso de desarrollo del dominio: Identificación de los stakeholders; Adquisición del dominio; Análisis del dominio y formación de conceptos	<i>Análisis del dominio:</i> modelo de procesos de negocios; modelo de características
Modelado del dominio (facetas del dominio): Procesos del negocio; Intrínsecos; Tecnologías soportadas; Organización y gerencia; Reglas y regulaciones; Scripts (guiones); Comportamiento humano	<i>Diseño del dominio:</i> arquitectura genérica abstracta; arquitectura de referencia (top-down) y/o arquitectura de línea de productos (botton-up)

**B. Proceso para el Desarrollo del Dominio, dentro de la ID Propuesto por Bjorner**

Contempla las siguientes fases o subprocesos:

- identificación de los stakeholders
- adquisición del conocimiento del dominio

- análisis del conocimiento del dominio y formación de conceptos
- modelado del dominio (diseño del dominio)
- verificación y validación del dominio y
- teoría del dominio.

1) *Los Stakeholders del Dominio y sus Perspectivas:* Los stakeholders del dominio se refieren a una persona, o grupo de personas unidas de alguna manera, o por una institución, empresa o un grupo de este tipo, caracterizado por su interés común en el dominio o dependencia del dominio. Por la perspectiva de los stakeholders del dominio lo comprendemos a él, o una comprensión del dominio compartido por un grupo de stakeholders específicamente identificado; una vista que puede diferir de un grupo de interés a otro grupo de interés del mismo dominio. La identificación de las perspectivas de los stakeholders (por ejemplo, vistas) incorpora el desarrollo de principios, técnicas y herramientas. Sin la clara identificación y enlace con todas las partes relevantes interesadas del dominio no se puede aspirar a construir un modelo de dominio creíble.

2) *Adquisición del Dominio:* La adquisición del dominio comprende el proceso de obtención de datos sobre el dominio, esto es, obtener (capturar) datos (hechos) de las partes interesadas del dominio, de redactar las descripciones acerca de ellos, y de su estructura (es decir, su organización y/o clasificación aproximada de estas descripciones). Esto incluye la recolección de datos, de la literatura y de nuestras observaciones acerca del conocimiento del dominio. Este conocimiento incluye entidades fenomenológicas, funciones, eventos y comportamientos.

3) *Análisis y del Dominio y Formación de Conceptos:* El análisis del dominio comprende el estudio de la adquisición del conocimiento del dominio, con los siguientes objetivos: a) descubrir inconsistencias, conflictos y la falta de completitud dentro de ellas, b) formar conceptos de estas declaraciones de adquisición del conocimiento del dominio. La formación de conceptos del dominio comprende la abstracción de los fenómenos del dominio, en conceptos.

4) *Modelado del Dominio (Modelo de Facetas del Dominio):* Las facetas del dominio comprende un conjunto finito de formas genéricas de analizar un dominio, cada forma genérica representa una vista particular del dominio o faceta del dominio, de tal modo que las diferentes facetas abarcan conceptualmente diferentes vistas, y juntas conforman el dominio. A continuación se enuncian las principales facetas del dominio según [25]:

- Procesos del negocio (conjuntos de acciones que se realizan en el dominio)
- Características intrínsecas (lo que es común a todas las facetas)
- Tecnologías de apoyo (implementación de las facetas)
- Organización y gestión

- Reglas y regulaciones (normativas)
- Comportamiento del recurso humano (stakeholders)

Un modelo de dominio se refiere al conjunto de uno o más modelos acordes con las facetas del dominio, estas pueden ser reescritas (y posiblemente formalizadas) en un modelo consolidado (un único modelo en donde confluyen todas las facetas).

Para modelar una faceta del dominio, primero hay que adquirirla; entonces se debe analizar lo que se ha adquirido, y formar conceptos de lo que se ha analizado, para luego describirla: a) aproximadamente, b) en términos de las entidades del dominio (terminología), c) de forma narrativa y d) posiblemente formalizando la faceta [25].

5) *Validación del Dominio*: La validación del dominio se refiere a la garantía, con los stakeholders, en particular los clientes, que las descripciones del dominio que se producen como resultado de la adquisición del dominio, análisis del dominio, la formación de conceptos y el modelado del dominio es acorde con la forma en que los grupos de interés miran el dominio.

### C. Arquitectura Orientada a Servicios (SOA)

SOA se ha convertido en un concepto ampliamente estudiado y utilizado en la investigación y la práctica de la ingeniería de software. SOA es una arquitectura para el diseño, desarrollo e implementación de aplicaciones de computación distribuida e independiente de la tecnología basada en estándares para el descubrimiento, el enlazado y ensamblaje de servicios de software débilmente acoplados que son transmitidos mediante una red [11].

SOA proporciona el marco conceptual para la realización de sistemas orientados al servicio al permitir que los sistemas de software puedan ser compuestos y reconfigurados dinámicamente usando servicios detectables en la red. Desde el punto de vista de la ILPS, este modelo de despliegue promete beneficios significativos sobre el modelo tradicional de implementación del software como un producto: en primer lugar, la naturaleza dinámica de SOA significa que puede soportar las necesidades y expectativas del usuario en un entorno de cambios continuos; en segundo lugar, los servicios pueden ser combinados con diferentes configuraciones y contextos simplificando el despliegue de variantes de los productos adaptados a las necesidades de los distintos clientes, pero sobre todo basados en los mismos servicios básicos [28].

Particularmente SOA se ocupa de la construcción independiente de servicios alineados al negocio que pueden ser combinados en significativos procesos de negocio de alto nivel y soluciones en el contexto empresarial [29]. El valor real de SOA no es solo la construcción de los servicios, sino más bien cuando los servicios reutilizables se combinan para crear procesos de negocios ágiles y flexibles. La agilidad y flexibilidad se producen cuando los nuevos procesos pueden rápidamente y eficientemente ser creados a partir del conjunto de servicios existente [12].

SOA permite el ensamblado, orquestación y el mantenimiento de las soluciones empresariales para reaccionar rápidamente a los cambiantes requisitos empresariales [30].

### D. Calidad del Software

La calidad ha sido típicamente definida como un nivel de excelencia, de conformidad con las especificaciones, de satisfacción de requisitos, de atributos distintivos, de estar libre de defectos, deficiencias y de las significativas variaciones para cumplir con las expectativas del cliente [31][32].

Establecer un modelo de calidad para el desarrollo del dominio de la aplicación que involucra los procesos de identificación de los stakeholders, adquisición del dominio, análisis del dominio y modelado del dominio, planteados en Bjorner [25], garantizaría la validez y confiabilidad del dominio desarrollado. El aseguramiento de la calidad del producto es una actividad crucial para el éxito de la industria del software, pero es, si cabe, más importante cuando se trata del desarrollo de líneas de producto software, dado que la reutilización masiva de activos de software hace que los atributos de calidad (propiedades medibles de un artefacto de software) de estos activos impacten en la calidad de todos los productos de una línea de producto [17].

La calidad se ha convertido en un atributo crítico de los productos de software ya que su ausencia produce pérdidas financieras, de salud, y a veces de vida. Al mismo tiempo la definición, o alcance, del dominio de la calidad del software ha evolucionado continuamente desde una perspectiva técnica a una perspectiva que abarca aspectos humanos tales como la facilidad de uso y la satisfacción [21][24].

## III. REVISIÓN DOCUMENTAL SISTEMÁTICA

La RDS se llevó a cabo en el primer y segundo trimestre del 2014. Kitchenham indica que la gestión del proceso de revisión sistemática se fundamenta en tres fases principales: Planificación de la revisión, Realización de la revisión e Informe de la revisión [26]. En la Tabla II se describen las fases, las etapas que incluyen cada fase y los artefactos generados en el proceso de revisión sistemática.

**Tabla II:** Proceso de Revisión Documental Sistemática

Fases	Etapas	Artefactos
Planificación de la revisión	Justificación de la necesidad de una revisión.	Protocolo
	Especificación de la(s) interrogante(s) de investigación.	
	Desarrollo del protocolo de revisión.	
Realización de la revisión	Identificación de los estudios relacionados.	Estudios aceptados
	Selección de los estudios primarios.	
	Evaluación de la calidad de los estudios.	
	Extracción de los datos.	Modelo de datos
Síntesis de los datos.		
Informe de la revisión	Especificación de los mecanismos de difusión.	Tablas resumen
	Presentación de los resultados	

### A. Planificación de la Revisión

Esta fase define los objetivos de la investigación y el protocolo en que la revisión se ejecutará.

1) *Identificación de la Necesidad para una Revisión Sistemática:* Estrategias y actividades utilizadas durante el proceso de ID para el desarrollo de LPSOS, con el fin de determinar los alcances y limitaciones de las propuestas encontradas.

2) *Objetivos:* Revisar los procesos de desarrollo actuales en el área de la ID para el enfoque de LPS para el desarrollo de Familias de Productos Orientadas a Servicios; Identificar las estrategias y actividades utilizadas en las fases de análisis, diseño e implementación durante el proceso de ID en el desarrollo de LPSOS; Recolectar evidencia acerca de las investigaciones actuales que indiquen sus implicaciones en la práctica; Identificar problemas pendientes por resolver y posibles áreas para la mejora de estos procesos.

3) *Desarrollo del Protocolo:* El protocolo es un plan o conjunto de pasos a seguir en un estudio, constituido por las preguntas de investigación, las estrategias de búsqueda, criterios y procesos de selección, los criterios de evaluación de la calidad, el modelo y proceso de extracción de datos, y el plan de análisis de los datos.

Las preguntas de investigación tienen el objetivo de encontrar todos los estudios para comprender y resumir las evidencias acerca de los enfoques propuestos o utilizados durante el desarrollo del proceso de ID, las siguientes PIs (Preguntas de Investigación) fueron consideradas:

- PI1: ¿Cuáles son las actividades comúnmente realizadas en las etapas de análisis, diseño e implementación en el proceso de ID bajo el enfoque de LPSOS?
- PI2: Desde el punto de vista de las LPS implementadas con SOA. ¿Cuáles son los artefactos comúnmente utilizados y técnicas utilizadas para su construcción en la fase de ID para LPSOS?
- PI3: Desde el punto de vista de la fase de ID. ¿Cuál es el enfoque utilizado para el diseño de la arquitectura de referencia y/o arquitectura de líneas de productos (desde la perspectiva de la investigación realizada en [33]: enfoques proactivo y reactivo)?
- PI4: ¿Existen vistas de calidad que tomen en consideración algún estándar o modelo de calidad del producto (ISO9126-1, ISO25010 [32]) que tomen en consideración las características de calidad particulares para el diseño de LPSOS?
- PI5: ¿Existen herramientas de análisis o diseño que apoyen el proceso de ID para el desarrollo de LPSOS?

Con relación a la estrategia de búsqueda, primero se establecieron las fuentes de búsqueda para encontrar los estudios primarios, particularmente se seleccionaron los siguientes gestores de búsqueda: Google Academic, Scirus, ResearchGate y ScienceDirect.

Para establecer la estrategia de búsqueda, teniendo en cuenta las preguntas de investigación, inicialmente se identificaron las principales palabras de acuerdo a la terminología utilizada por Bjorner [25], que a continuación se enuncian: Software Product Line, Service-Oriented Architecture, Domain Engineering, Domain Stakeholders, Domain Acquisition, Domain Analysis, Domain Design, Domain Modeling, Domain Verification, Domain Validation, Analysis Tool.

Con relación a los criterios de inclusión y exclusión, aquí se establecen los criterios por los cuales los estudios serán evaluados para decidir si deben ser seleccionados o no en el contexto de la revisión sistemática.

Por lo tanto, los CI (Criterios de Inclusión) que se utilizan para incluir los estudios relevantes en esta RDS:

- CII: Que el artículo aborde el área de la Ingeniería de Dominio para el desarrollo de Líneas de Productos de Software para Familias de Productos Orientados a Servicios utilizando SOA.

Los CE (Criterios de Exclusión) que se utilizan para excluir los estudios que no contribuyen a responder a las interrogantes de la investigación son:

- CE1: Que el artículo no aborde la integración de LPS y SOA como alternativa para el desarrollo de productos de software
- CE2: Que el artículo este escrito en un lenguaje diferente al inglés, portugués y español
- CE3: Que el artículo no esté disponible en el formato PDF
- CE4: Que el artículo tenga como contenido una presentación en un congreso o evento realizado por los autores.
- CE5: Que el artículo este repetido (el mismo artículo, pero con dos nombres diferentes), en este caso solo se acepta un solo ejemplar.

Con respecto a la extracción de datos y método de síntesis, se prevé la construcción de tablas para la obtención de datos relacionados con las preguntas de investigación, en caso de que sea aplicable.

En la selección inicial de las publicaciones de entre los resultados de búsqueda, se dio lectura a los títulos y resúmenes de las publicaciones encontradas. En la Figura 1 se muestra el modelo de datos inicial utilizado en la revisión sistemática.

Las publicaciones fueron almacenadas en carpetas independientes de acuerdo a la expresión de búsqueda utilizada, de esta forma facilitar la ubicación de los estudios para realizar el análisis con mayor fluidez y comodidad.

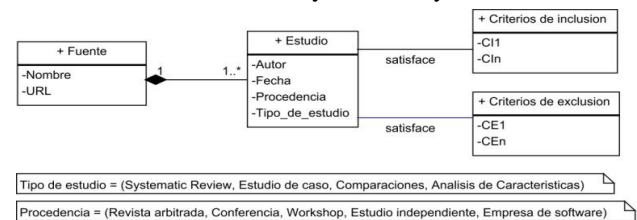


Figura 1: Modelo de Datos Inicial para la Revisión Sistemática

Del conjunto de publicaciones seleccionadas se procede a la extracción de la información relevante utilizando el modelo de datos que será utilizado en las fases de realización e informe de la revisión. En la recolección de datos, las citas textuales de los artículos analizados se conservarán en el idioma original (inglés) para evitar el posible sesgo al traducir y posteriormente parafrasear según la interpretación del investigador sobre la cita leída. Esto permite una vez realizados los aportes a la investigación, poder referirse a las citas por posibles malas interpretaciones de las mismas, para validar y para futuras referencias. No obstante, al final de la revisión sistemática los resultados finales reflejados en la fase de informe de la revisión se colocarán en español que es el idioma nativo del investigador.

Para estandarizar la forma en que la información estará representada, se definió un formato para recopilar datos de los estudios seleccionados. La Tabla III muestra el formato que será utilizado para registrar los resultados de la extracción de los datos.

**Tabla III:** Formato para la Recolección de Datos

Estudio				
Definición ID				
Etapas	Actividades	Propósito	Artefactos de entrada	Artefactos de salida
Análisis				
Diseño				
Implementación				
<b>Enfoque</b>				
<b>Comentarios</b>				

La Tabla IV describe las propiedades de extracción que serán utilizados a lo largo de la revisión sistemática.

**Tabla IV:** Propiedades de Extracción

Propiedades	Descripción
Estudio	Identificación del artículo objeto de estudio.
Definición	Concepto utilizado en el estudio para definir la ID.
Etapas	Indica las etapas utilizadas en los trabajos analizados para descubrir el proceso de desarrollo del dominio.
Actividades	Indica las actividades realizadas en cada una de las etapas anteriores.
Propósito	Objetivo o meta propuesta por la actividad a realizar.
Artefactos de entrada	Artefactos utilizados como insumos para poder realizar una actividad.
Artefactos de salida	Artefactos producidos como consecuencia de haber realizado una actividad.
Enfoque	Enfoque utilizado para el desarrollo del dominio; puede ser Proactivo o Reactivo.
Comentarios	Aspectos de interés reflejados en la investigación.

### B. Realización de la Revisión

En esta fase, los estudios son identificados, seleccionados y evaluados de acuerdo con el protocolo previamente establecido. Como resultado de la búsqueda 231 estudios fueron localizados, de los cuales solo 67 estudios fueron aceptados para su análisis.

Para la selección final de los estudios aceptados se realizó un proceso filtrado que consistió en la aplicación de los criterios de inclusión y de exclusión, en la Tabla V se muestra un

resumen de los resultados indicando la cantidad en números de los estudios aceptados y rechazados.

**Tabla V:** Relación de Estudios Localizados y Aceptados según la Fuente de Datos

Fuentes de datos	Artículos	
	Aceptados	Localizados
Google Academic	20	36
Scirus	30	167
ResearchGate	8	10
ScienceDirect	9	18
<b>Total</b>	<b>67</b>	<b>231</b>

En los estudios aceptados se descartaron los artículos repetidos donde los identificadores de los archivos (nombres) son iguales; también se descartaron aquellos artículos, que al examinar sus contenidos se encontró que eran idénticos, aun cuando sus identificadores eran diferentes.

A continuación en las Tablas VI y VII se muestra la distribución de los estudios aceptados agrupados por categorías o ámbitos de análisis, basados en la terminología utilizada en [25] en las etapas de desarrollo del dominio de aplicación.

**Tabla VI:** Etapas (Categorías) para el Desarrollo del Dominio [25]

Desarrollo del Dominio		
Categorías de análisis		Total
1	Stakeholders Identification	2
2	Domain Acquisition	2
3	Domain Analysis and Concept Formation	32
4	Domain Modeling: (domain design)	18
5	Domain Validation and Verification	0
6	Domain Theory	0
7	Domain Documentation	1
<b>Total</b>		<b>55</b>

**Tabla VII:** Otras Categorías de Interés para la Investigación

Otras categorías de análisis	Total
Domain-Driven Design	2
Quality Model	1
Analysis Tool	0
Design Tool	0
Domain Architecture	9
<b>Total</b>	<b>12</b>

A continuación, un análisis más detallado se llevó a cabo en cada estudio incluido y el informe de la revisión.

### C. Informe de la Revisión

En esta última fase, se presentan los resultados analíticos de la revisión sistemática. La extracción de los datos obtenidos y la síntesis del conocimiento considerando cada pregunta de investigación se discute a continuación.

Las respuestas a las preguntas de investigación versan sobre los aportes de la investigación realizada por Dines Bjorner [25] para el desarrollo del dominio de propósito general, del trabajo realizado por Pohl et al. [8] sobre el procesos de ID específicamente para las LPS, y de los trabajos realizados en [3][14][16][17][34][35][36][37][38][39] vinculados con las LPS para el desarrollo de FPOS (Familias de Producto Orientados a Servicios), que particularmente utilizan como arquitectura de implementación la SOA (Arquitectura Orientada a Servicios).

1) *Con Relación a la Pregunta P11 y P12: ¿Cuáles son las actividades comúnmente realizadas en las etapas de análisis, diseño e implementación en el proceso de ID bajo el enfoque de LPSOS?*

Para responder a estas dos preguntas, se utilizaron las siguientes dos estrategias; primero, de los estudios aceptados inicialmente, se seleccionaron aquellos estudios que abordan o describen un proceso de desarrollo para LPSOS; segundo, para representar los resultados del análisis de estos estudios, y mostrarlos de una forma más amigable, se utilizó un modelo de datos para la sistematización de los datos. De los 67 estudios aceptados, solo 10 estudios fueron seleccionados, debido a que abordan y/o describen un proceso de desarrollo para la construcción de LPSOS, que es el objetivo principal de interés del estudio.

Las Tablas VIII, IX y X describen el conjunto de actividades, artefactos generados y técnicas utilizadas respectivamente en la construcción de LPSOS a partir de los estudios analizados: [3][14][16][17][34][35][36][37][38][39] vinculados con las LPS para el desarrollo de Familias de Producto Orientados a Servicios. Los 10 estudios analizados abordan el proceso de ID bajo el enfoque de LPSOS basándose en las siguientes 3 etapas: análisis, diseño e implementación.

**Tabla VIII:** Actividades Realizadas en las Fases de Análisis, Diseño e Implementación de la LPSOS

Etapas	Actividades
<b>Análisis</b>	Análisis de factibilidad de la LPS Análisis del dominio Análisis de características: similitudes y variabilidades de la LPS Análisis de los activos existentes Análisis de los requerimientos de la familia
<b>Diseño</b>	Identificación de los elementos arquitecturales Especificación de la arquitectura Definición de la técnica de implementación de la variabilidad
<b>Implementación</b>	Construcción de los componentes/servicios Plan de producción ( <i>Roadmap</i> ) de los productos de la LPS

**Tabla IX:** Artefactos Desarrollados en las Fases de Análisis, Diseño e Implementación de la LPSOS

Etapas	Artefactos
<b>Análisis</b>	Documento de factibilidad Modelo conceptual del dominio Modelo de características Listado de componentes/servicios disponibles Tabla de identificación de requerimientos
<b>Diseño</b>	Servicios/Componentes Arquitectura de referencia Modelo de configuración Modelo de componentes/servicios Modelo de estado Modelo de colaboración, interacción, comunicación. Mecanismos de implementación de la variabilidad.
<b>Implementación</b>	Componentes SPL desarrollados, reutilizados, o comprados a terceros. Composición de los componentes/servicios

**Tabla X:** Técnicas Utilizadas en las Fases de Análisis, Diseño e Implementación de la LPSOS

Etapas	Técnicas
<b>Análisis</b>	Diagrama de clases (UML) Notación de Modelado de Procesos de Negocios (BPMN) Diagrama de actividades (UML) Diagrama de Casos de Uso (UML) FODA, variantes de FODA Tablas Requisitos Funcionales – No Funcionales
<b>Diseño</b>	Service-Oriented Architecture (SOA) Service-Component Architecture (SCA) Vistas arquitecturales: Diagrama de clases (UML), Diagrama de componentes (UML), Diagrama de estado (UML), Diagrama de colaboración (UML). Lenguaje de Variabilidad Común (LVC) Mecanismos de configuración, generación
<b>Implementación</b>	SCA, SOA Orquestado/configuración/composición

2) *Con Relación a la Pregunta “PI3”:* ¿Desde el punto de vista de la fase de ID, cual es el enfoque utilizado para el diseño de la arquitectura de referencia y/o arquitectura de líneas de productos?

Desde el punto de vista de la fase de ID, el enfoque utilizado para el diseño de la arquitectura de referencia y/o arquitectura de líneas de productos (desde la perspectiva de la investigación realizada en [33] por la mayoría de los estudios analizados fue el enfoque proactivo.

3) *Con Relación a la Pregunta “PI4”:* ¿Existen vistas de calidad que tomen en consideración algún estándar o modelo de calidad del producto?

No existe evidencia de algún estudio que proponga o utilice alguna vista de calidad, modelo de calidad o implemente algún estándar de calidad del producto, utilizando ambos enfoques: LPS y SOA. Solo un trabajo [23] implementa estrategias que incorporan características de calidad en un enfoque LPSOS, es decir que consideran la calidad desde la perspectiva de QoS para la concepción de la LPS y toman en consideración la arquitectura de implementación SOA, sin embargo, aunque esta investigación hace un análisis superficial del modelo de calidad ISO9126-1 y del ISO25010 [32], argumentando que debe adecuarse a las particulares características de las LPSOS, no se propone una vista de calidad basada en alguno de los estándares antes mencionados.

Por otro lado, se evidenciaron dos trabajos independientes que abordan o proponen un modelo de calidad, uno de ellos describe un procedimiento para incorporar un modelo de calidad para una LPS [40], el segundo si aborda un modelo de calidad basado en el ISO25010, pero solo para SOA bajo el paradigma de la Computación Orientada a Servicios (del inglés: Service-Oriented Computing, SOC) [41].

4) *Con Relación a la Pregunta “PI5”:* ¿Existen herramientas de análisis o diseño que apoyen el proceso de ID para el desarrollo de LPSOS?

No se describen herramientas y/o prototipos que apoyen o aborden el proceso de desarrollo de la LPSOS, ni que utilicen alguna de vista de calidad del dominio aplicando un modelo de calidad o estándar (ISO9126, ISO25010, entre otros); sin embargo existen algunos estudios que abordan el proceso, por ejemplo desde la perspectiva de la LPS [5][20][42], particularmente se encuentra una herramienta en línea a través del enlace: [www.splot-research.org](http://www.splot-research.org).

Por otra parte, en el estudio [43] que aborda el análisis de las características de las herramientas del tipo open source que apoyan procesos de LPS, muestra que la mayoría de las propuestas se basan o parten del modelado de las características de la LPS [44][45][46].

#### IV. DISCUSIÓN

Es conveniente y necesario describir aquellos aspectos que resaltan de cada uno de los 10 procesos analizados, aparte de las actividades realizadas en las etapas de análisis, diseño e implementación del dominio; entre ellas se destacan las siguientes evidencias:

Existen muy pocos estudios que abordan adecuadamente la identificación de los stakeholders (partes interesadas del dominio).

Particularmente no se identifican claramente los stakeholder (actores) involucrados para efectuar la adquisición del dominio, aspecto fundamental según Bjorner [25], para obtener una visión global del dominio de la aplicación.

Solo algunos estudios [3][36] identifican los roles de los stakeholders y en que etapas del proceso de desarrollo del dominio actúan o toman decisiones; pero solo en las fases de análisis, diseño e implementación del dominio.

También existen pocos estudios que aborden algún método, técnica o procedimiento relacionado con la adquisición del dominio.

En todos los estudios seleccionados para la investigación se observó que ninguno valida y/o verifica el dominio desarrollado por los ingenieros de software (arquitectura de referencia, componentes/servicios) contra el dominio especificado por los expertos del dominio del problema (conocedores de los procesos de negocios en la organización).

La mayoría de los estudios analizados, solo abordan las fases de análisis y modelado del dominio.

Solo algunos describen como implementar el dominio (arquitectura de referencia [47]).

La arquitectura de referencia descrita por los autores, no toman en cuenta alguna vista de calidad del dominio, como el descrito en [48], o toman en consideración algún estándar de calidad, por ejemplo: ISO9126 o la versión actualizada ISO25010 [32].

Se evidencia en los estudios, que los conocedores del dominio por lo general no son del área de la informática o afines, si la organización requiere una LPS implementada a través de SOA, no se describe o enuncia como se realiza la transformación de las decisiones del negocio al BPM (Modelo de Procesos de Negocios), FM (Modelo de Características) en el caso de la

LPS y los servicios que ha de soportar la arquitectura, particularmente para SOA; existe mucha ambigüedad y poca claridad en los estudios analizados.

Solo los stakeholders conocedores del dominio del problema con experiencia o estudios en el campo de la informática tendrían las habilidades o capacidades necesarias para realizar la tarea anterior con éxito o por lo menos con más precisión desde el punto de vista del desarrollo de LPSOS que se pretende realizar en la organización.

Tampoco se definen vistas de calidad y/o estándares de calidad para validar de manera formal el proceso de adquisición, análisis y diseño del dominio para la LPSOS [20][42].

Los 10 estudios abordan parcialmente la documentación total del dominio, según lo planteado por Bjorner [25], referente a las facetas del dominio.

Ninguno de los 10 estudios analizados aborda las 6 etapas (identificación de los stakeholders, adquisición del dominio, análisis del dominio, diseño del dominio, validación/verificación del dominio y desarrollo de la teoría del dominio) para el desarrollo del dominio propuesta por Bjorner [25].

Se destaca que los 10 estudios abordan el proceso de desarrollo de la ID de forma diferente, es decir, aun cuando los procesos utilizan algunas actividades y artefactos comunes, cada uno de ellos realiza el proceso visto de forma global, con actividades, artefactos y técnicas diferentes.

#### V. CONCLUSIONES

La principal contribución de este trabajo fue realizar una RDS para presentar una perspectiva detallada sobre las actividades, artefactos y técnicas que comúnmente se utilizan en un proceso de desarrollo para la ID de LPSOS. Existe una carencia en los estudios analizados en lo referente al uso de estándares de calidad en los procesos de desarrollo identificados y falta de trazabilidad entre los modelos utilizados, sobre todo en lo que respecta a los RNF (Requisitos No Funcionales) del modelo del negocio al modelo del sistema de software en un dominio dado. Esta revisión sistemática provee a la comunidad de desarrolladores el estado del arte en lo concerniente con el desarrollo completo de la disciplina de ID para LPSOS.

Actualmente se está definiendo un proceso de análisis de dominio para LPSOS, tomando en cuenta los análisis realizados a partir de esta RDS. Se ha seleccionado como caso de estudio para aplicar este proceso al dominio de los SIS (Sistemas Integrados de Salud). Este dominio es de particular importancia en esta investigación por tres aspectos claves: por una parte será utilizado para aplicar las principales actividades, artefactos y técnicas evidenciadas en la RDS para la ingeniería de requisitos de SIS, los cuales son sistemas de información, generalmente en 4-capas (presentación, proceso, datos y comunicación), orientados a servicios, ya que utilizan en su mayoría una arquitectura basada en SOA en su capa de comunicación. Por otra parte, en este dominio no se encuentra definida claramente una LPS y las arquitecturas tipo SOA que permiten la interoperabilidad de servicios, no abarcan funcionalidades esenciales de SIS, como por ejemplo la interoperabilidad de las historias clínicas del paciente, a menos que se tenga la adopción sistemática de estándares por parte de

la comunidad de salud; este requisito es uno de los más importantes del dominio. Por lo tanto, las tareas relacionadas con la salud del ciudadano, como son, por ejemplo, la de remitir un paciente a otros centros de salud especializados, se dificultan por la falta de interoperabilidad de las historias clínicas con otros centros de salud; se recalca que solo la adopción de estándares permitiría esta facilidad. A nivel internacional, para las historias clínicas electrónicas se encuentran ya definidos algunos estándares como el HL7, pero aún no comúnmente adoptado a nivel mundial. Finalmente, a nivel nacional tampoco hay estándares adoptados que permitan realizar la integración de estos sistemas. Es de hacer notar sin embargo, que ya fue aprobada como Ley Orgánica de la Nación la Ley de Telesalud, el 12 de Agosto del 2014<sup>1</sup>. El estudio de los requisitos planteados en esta Ley constituirá parte de las reglas del negocio a ser incluidas en el modelo del negocio del dominio de los SIS, del cual la LPSOS será derivada.

Con este caso de estudio se pretende demostrar las ventajas y/o fortalezas, y/o debilidades del proceso definido, el cual permitirá una primera validación del proceso.

#### AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por los proyectos grupales DISoFT No. 2011001343 del PEII-Fonacit; DesCCaP No. PG-03-8014-2011 y DARGRAF No. 03-8730-2013-1 del CDCH; además, por el Postgrado en Ciencias de la Computación, Facultad de Ciencias, Universidad Central de Venezuela y el Banco Central de Venezuela.

#### REFERENCIAS

- [1] P. Istoan, G. Nain, G. Perrouin, and J. Jezequel, *Dynamic Software Product Lines for Service-Based Systems*, in proceedings of the 2009 Ninth IEEE International Conference on Computer and Information Technology (CIT'09), Xiamen, China, October 2009.
- [2] P. Istoan, *Software Product Lines for Creating Service Oriented Applications*, Masters internship at IRISA Rennes Research Institute, TRISKELL Research Team, Rennes, France, June 2009.
- [3] F. M. Medeiros, E. S. de Almeida, and S. R. de Lemos Meira, *Designing a Set of Service-Oriented Systems as a Software Product Line*, in proceedings of the Fourth Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), pp. 70-79, Salvador, Bahia, Brazil, September 2010.
- [4] S. Apel, D. Batory, C. Kästner, and G. Saake, *Feature-Oriented Software Product Lines*, Springer, 2013.
- [5] Q. Munir and M. Shahid, *Software Product Line: Survey of Tools*, Master's thesis, Linköping University, Department of Computer and Information Science, 2010.
- [6] B. Mohabbati, M. Asadi, D. Gašević, M. Hatala, and H. Müller, *Combining Service-Oriented and Software Product Line Engineering: A Systematic Mapping Study*, Information and Software Technology, vol. 55, no. 11, pp. 1845-1859, 2013.
- [7] R. Dos Santos and M. Fantinato, *The Use of Software Product Lines for Business Process Management: A Systematic Literature Review*, Information and Software Technology, vol. 55, pp. 1355-1373, 2013.
- [8] K. Pohl, G. Bockle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*, Springer, 2005.
- [9] Wide Web Consortium, *Web Services Architecture Requirements*, W3C Working Group Note, 11 February, 2004.
- [10] J. Nicolai, *SOA in Practice: The Art of Distributed System Design*, O'Reilly, Sebastopol, California, USA, 2007.
- [11] M. Galster, P. Avgeriou, and D. Tofan, *Constraints for the Design of Variability-Intensive Service-Oriented Reference Architectures—An Industrial Case Study*, Information and Software Technology, vol. 55, no. 2, pp. 428-441, 2013.
- [12] M. Rosen, B. Lublinsky, K. Smith, and M. Balcer, *Applied SOA: Service-Oriented Architecture and Design Strategies*, Wiley & Sons, 2008.
- [13] J. Estefan, K. Laskey, F. McCabe, and P. Thornton, *Reference Architecture Foundation for Service Oriented Architecture Version 1.0*, OASIS Committee Draft, vol. 2, no. 14, 2009.
- [14] M. Abu-Matar and H. Gomaa, *An Automated Framework for Variability Management of Service-Oriented Software Product Lines*, in proceedings of the IEEE 7th International Symposium on Service Oriented System Engineering (SOSE), pp. 260-267. San Francisco, California, USA, March 2013.
- [15] A. Ezenwoke, S. Misra, and M. Adigun, *An Approach for e-Commerce On-Demand Service-Oriented Product Line Development*, Acta Polytechnica Hungarica, vol. 10, no. 2, 2013.
- [16] P. G. G. Queiroz and R. T. V. Braga, *Uma Abordagem de Desenvolvimento de Linha de Produtos com uma Arquitetura Orientada a Serviços*, Master's thesis, ICMC-Universidade de Sao Paulo, Sao Carlos, Sao Paulo, Brazil, November 2009.
- [17] S. Günther and T. Berger, *Service-Oriented Product Lines: Towards a Development Process and Feature Management Model for Web Services*, in proceedings of the 12th International Software Product Line Conference (SPLC'08), pp. 131-136, Limerick, Ireland, September 2008.
- [18] J. Lee and G. Kotonya, *Combining Service-Oriented with Product Line Engineering*, Software, IEEE, vol. 27, no. 3, pp. 35-41, 2010.
- [19] C. Parra, X. Blanc, and L. Duchien, *Context Awareness for Dynamic Service-Oriented Product Lines*, in proceedings of the 13th International Software Product Line Conference (SPLC'09), pp. 131-140, San Francisco, California, USA, August 2009.
- [20] H. J. González, *Integration of Quality Attributes in Software Product Line Development*, Master Tesis en Ingeniería del Software, Métodos Formales y Sistemas de Información (ISMFSI), 2012.
- [21] W. Suryn, *Software Quality Engineering: A Practitioner's Approach*, IEEE, Published by John Wiley & Sons, Inc., Hoboken, New Jersey, 2014.
- [22] S. Wagner, *Software Product Quality Control*, Springer, 2013.
- [23] B. Mohabbati, D. Gasevic, M. Hatala, and H. Muller, *A Quality Model and Evaluation Method for Service-Oriented Software Product Line and Configurable Business Processes*, ACM Transactions on Software Engineering Methodology, September 2013.
- [24] L. O'Brien, L. Bass, and P. Merson, *Quality Attributes and Service-Oriented Architectures*, Software Engineering Institute, Carnegie Mellon University, 449, September 2005.
- [25] D. Björner, *Software Engineering 3: Domains, Requirements, and Software Design*, Texts in Theoretical Computer Science, Springer-Verlag Berlin Heidelberg, 2006.
- [26] B. Kitchenham, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*, Version 2.3, EBSE Technical Report, Software Engineering Group, School of Computer Science and Mathematics, Keele University, UK and Department of Computer Science, University of Durham, UK, 2007.
- [27] I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen, and J. Bettin, *Domain Engineering. Product Lines, Languages and Conceptual Models*. Springer, 2013.
- [28] G. Kotonya, J. Lee, and D. Robinson, *A Consumer-Centred Approach for Service-Oriented Product Line Development*, in proceedings of Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA 2009), pp. 211-220, Cambridge, UK, September 2009.
- [29] Q. Gu and P. Lago, *On Service-Oriented Architectural Concerns and Viewpoints*, in proceedings of Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA 2009), pp. 289-292, Cambridge, UK, September 2009.

<sup>1</sup> <http://conocimientolibre.cenditel.gob.ve/files/2014/05/LEY-TELESALUD.pdf>



- [30] M. Asadi, B. Mohabbati, N. Kaviani, D. Gašević, M. Bošković, and M. Hatala, *Model-Driven Development of Families of Service-Oriented Architectures*, in proceeding of the First International Workshop on Feature-Oriented Software Development (FOSD'09), pp. 95-102, Denver, Colorado, USA, October 2009.
- [31] M. Allauddin, F. Azam, and M.J. Zia, *A Survey of Quality Assurance Frameworks for Service Oriented Systems*, International Journal of Advancements in Technology, vol. 2, no. 2, pp. 188-198, 2011.
- [32] ISO/IEC 25010. Software Engineering - *Software Product Quality Requirements and Evaluation (SQuaRE) - Quality Model*, International Organization for Standardization (ISO), 2010.
- [33] J. Herrera, F. Losavio, and A. Matteo, *Revisión Documental Sistemática de Enfoques y Técnicas para la Construcción de Arquitecturas en un Contexto de Líneas de Productos de Software*, Primera Conferencia Nacional de Computación, Informática y Sistemas (CoNCISa 2013). Naiguatá, Venezuela, Octubre 2013.
- [34] P. Dolog, *Engineering Adaptive Web Applications*, Doctoral Dissertation, University of Hanover, Germany, 2006.
- [35] A. Lapouchnian, *Exploiting Requirements Variability for Software Customization and Adaptation*, Doctoral Dissertation, University of Toronto, Canada, 2011.
- [36] C. Parra, *Towards Dynamic Software Product Lines: Unifying Design and Runtime Adaptations*, Doctoral Dissertation, Université des Sciences et Technologie de Lille, Lille I, France, 2011.
- [37] D. Smith and G. Lewis, *Service-Oriented Architecture and Software Product Lines: Pre-Implementation Decisions*, in proceedings of the 3rd International Workshop on Service Oriented Architectures and Product Lines (SOAPL'09), pp. 166-171, San Francisco, California, USA, September 2009.
- [38] M. Bošković, D. Gašević, B. Mohabbati, M. Asadi, M. Hatala, N. Kaviani, and E. Bagheri, *Developing Families of Software Services: A Semantic Web Approach*, Journal of Research & Practice in Information Technology, vol. 43, no. 3, 2011.
- [39] J. Lee, D. Muthig, and M. Naab, *A Feature-Oriented Approach for Developing Reusable Product Line Assets of Service-Based Systems*, Journal of Systems and Software, vol. 83, no. 7, pp. 1123-1136, 2010.
- [40] A. Trendowicz and T. Punter, *Quality Modeling for Software Product Lines*, in proceedings of the 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'03), 2003.
- [41] A. Goeb and K. Lochmann, *A Software Quality Model for SOA*, in proceedings of the 8th International Workshop on Software Quality (WoSQ'11), pp. 18-25, Szeged, Hungary, September 2011.
- [42] S. Montagud, *Un Método para la Evaluación de la Calidad de Líneas de Productos Software basado en SQuaRE*, Master's Thesis, (In Spanish), Master en Ingeniería del Software Metodos Formales y Sistemas de Información. Universidad Politécnica de Valencia, Spain, 2009.
- [43] S. Segura, D. Benavides, A. Ruiz-Cortés, and P. Trinidad, *Open Source Tools for Software Product Line Development*, Open Source and Product Lines, 2007.
- [44] M. Acher, P. Collet, P. Lahire, and R. B. France, *Familiar: A Domain-Specific Language for Large Scale Management of Feature Models*, Science of Computer Programming, vol. 78, no. 6, pp. 657-681, 2013.
- [45] L. Baresi, S. Guinea, and L. Pasquale, *Service-Oriented Dynamic Software Product Lines*, Computer, vol. 45, no. 10, pp. 42-48, 2012.
- [46] J. Lee, D. Muthig, and M. Naab, *An Approach for Developing Service Oriented Product Lines*, in proceedings of the 12th International Software Product Line Conference (SPLC'08), Limerick, Ireland, September 2008.
- [47] M. Njima, M. ter Beek, and S. Gnesi, *Product Line Architectures for SOA*, in proceedings of the 2011 International Conference on Software Engineering Research and Practice (SERP'11), Las Vegas, Nevada, USA, July 2011.
- [48] F. Losavio, and A. Matteo, *Reference Architecture Design Using Domain Quality View*, Journal of Software Engineering & Methodology, vol. 3, no. 1, 2013.

# Product Line Scoping for Healthcare Information Systems Using the ISO/IEC 26550 Reference Model

Juan Carlos Herrera<sup>1</sup>, Francisca Losavio<sup>2</sup>, Oscar Ordaz<sup>2,3</sup>  
jchr1982@gmail.com, francislosavio@gmail.com, oscarordaz55@gmail.com

<sup>1</sup>PFG Informática para la Gestión Social, Universidad Bolivariana de Venezuela, Caracas, Venezuela

<sup>2</sup>Escuela de Computación, Laboratorio MoST, Universidad Central de Venezuela, Caracas, Venezuela

<sup>3</sup>Escuela de Matemáticas, Universidad Central de Venezuela, Caracas, Venezuela

---

**Abstract:** The main goal of this work is to present a process for Software Product Lines (SPL) scoping focused on early consideration of software product quality. Product Line Scoping is the first phase of SPL Engineering (SPLE), in the Domain Engineering (DE) lifecycle, where the SPL long-term feasibility must be determined. The PLScoP process proposed here for SPL scoping, is an adaptation of the general PL Scoping phase defined in the new ISO/IEC 26550 standard describing a reference model or framework for SPLE, and it concerns three main stages, Portfolio Scoping, Domain Scoping and Asset Scoping. In this work, the complete PLScoP is outlined, but only the Domain Scoping phase will be detailed and applied. General guidelines on “What to do”, as the majority of standards offer, are defined in ISO/IEC 26550. Our PLScoP complements this framework by presenting the “How to do”, offering precise techniques and artefacts to be applied and constructed, and by considering early and systematically quality issues; this will allow reduction of the development effort in the subsequent DE phases, Domain Requirements Engineering and Domain Design, where the major effort is concentrated SPL development workload. The Domain Scoping step of PLScoP will be applied to the Healthcare Information Systems domain.

**Keywords:** Software Product Lines; Product Line Scoping, PLScoP; Domain Scoping; ISO/IEC 26550; Software Quality; ISO/IEC 25010; Healthcare Information Systems.

---

## I. INTRODUCTION

Software Product Lines (SPL), or simply Product Lines (PL), is an approach that provides a way of massive personalization of individual solutions from a repository of reusable software assets, in a particular domain; it is inspired in the Fordism technique used to increase production while lowering costs in early 20th century automotive industry. The term domain is often used in reference to a particular knowledge area; an application domain denotes any aspect where computing can be applied [1]; a *domain* is defined by Bérard as the minimal set of properties describing precisely a family of problems in which a computational application or system is involved for their solution [2], and it is the definition adopted in this context. The problem of software development based on reusing components or a core of software elements, favouring efficient and reliable development is not new [3][4][5][6] and it is a complex problem not yet completely solved in academic or industrial practices; moreover, there is still a huge gap between research results and their industrial application [7][8]. The SPL development, also called SPL Engineering (SPLE) [9], aims to promote maximal reuse exploiting common elements in similar products of the SPL family in a particular domain. The main idea, but not an easy task, is to capture the essential common

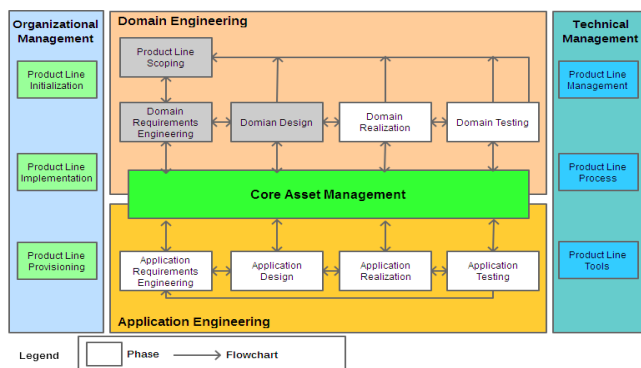
elements and possible variable issues to construct an evolutionary SPL, since it must manage changes and last over time to provide an economic payback. Instead of describing a single system, the SPL model describes a set or family of software systems, products or applications. Complex domain analysis must be achieved in order to specify and delimit the family that can be developed from the core of assets, with their commonality and variability [3][4][10].

This work is framed within the Domain Engineering (DE), first lifecycle of SPLE, where the major development effort is concentrated in constructing the SPL Reference Architecture (RA) and Core Asset Repository (see Figure 1). The main goal of this paper is to present and apply a PL Scoping Process (PLScoP), with precise techniques and artefacts, adapting the PL Scoping phase or initial DE phase, defined in the Reference Model of the new standard ISO/IEC 26550 [7][8][11]. Notice, in general that standards or general frameworks specify the “What to do”, however the details of the “How to do” have always to be properly defined; even if in [11] a list of available techniques and methods are provided, how to combine or adapt them to the SPL context to achieve a particular activity is not specified. Our work complements the SPL standard framework, offering explicit techniques to be applied to

perform PL Scoping activities. In particular, for the study of domain existing products in the Portfolio Scoping step, a bottom-up process [12] can be considered, providing as output an initial candidate architecture; moreover, the ISO/IEC 25010 quality model [13] is used to specify quality properties related to functional (FR) and non functional requirements (NFR), and BPMN <sup>1</sup>[14] is proposed to specify the domain model. PLScOP emphasizes software product quality assurance at early stages of SPLE; quality properties and their traceability w.r.t. FR and NFR requirements has in general been poorly considered in SPL development, playing however a major role in the SPL variability and evolutionary capacity. In this work, the Domain Scoping phase will be applied to the Healthcare Information Systems domain to obtain a first draft of a domain model.

Usual approaches focus more on products' features directly perceived by users [15], which is not the case of quality properties that appear as "implicit functionality" often later on during the SPL development; nevertheless, quality requirements are responsible of most of the SPL variability at the moment of deriving concrete SPL products during the Application Engineering (AE) or second SPLE lifecycle [16][17]. Figure 1 shows the ISO/IEC 26550 SPL Reference Model [11]. If quality properties are not considered early in the DE lifecycle of the SPL development, the global quality of the products derived from RA cannot be guaranteed, compromising organizational goals and the whole SPL ROI (Return of Inversion).

Besides this introduction and the conclusion, the structure of this paper is the following: some related works are discussed in Section 2; Section 3 describes the SPL Domain Engineering guidelines of the ISO/IEC 26550 standard; Section 4 presents PLScOP, as an adaption of the PI Scoping phase [11]. Finally, Section 5 is dedicated to the application of the Domain Scoping step of PLScOP to a case study in the Healthcare Information Systems domain.



**Figure 1:** ISO/IEC 26550 Reference Model for Software and Systems Product Lines

## II. RELATED WORKS

In what follows, some related works relevant to the subject of SPL scoping, context of this research, are discussed:

An SDR (Systematic Documental Review) [18] is presented in [19] to identify best practices, challenges and limitations of the

SPL scoping phase; this study points out that scoping is important and even essential for the achievement of the product line. Its main goal is to determine the feasibility of the SPL, identifying crucial aspects such as products that will conform the SPL, risks, potential reuse and costs to implement main assets. An important question related with our work is the following: *How is the SPL development affected by the scoping phase?* It is clear that the artefacts produced during this phase are input to the DRE phase and should reduce the effort also in subsequent phases. The study claims that the majority of the approaches reviewed do not have a clear understanding of the relation between scoping and DRE phases. In this sense, our proposition to use Bjørner's domain modelling [20] in PL Scoping will fill this gap.

A characterization of the benefits and weakness perceived in the Domain Scoping and DRE phases using the agile method is presented in [21]. The observed variables were the stakeholders' motivation, effort, communication and collaboration, iterative and adaptability aspects of the process, requirements and technological volatility. A question arisen from this study is: *How the effort to perform SPL scoping is characterized by the stakeholders?* The effort is measured in man-hours, and the answer was "great", due to the huge amount of domain documentation (often incomplete and inconsistent) that has to be analysed to capture enough domain knowledge to provide acceptable asset scoping and product portfolio, and the lack of domain experts. Activities identified for the scoping phase were: *Pre-scoping, Domain scoping, Product scoping, and Assets scoping*. Bottlenecks found were: the absence of domain and product experts to capture products' main functionalities, and the clear identification of features and their granularity. Moreover, variables stakeholders' communication and collaboration, iterative and adaptability aspects of the process, were also found to affect the effort. Our research do not use an agile method, nevertheless we claim to reduce the scoping effort by introducing the Bjørner's business process-centric technique of domain modelling [20], modified by introducing the quality intrinsic descriptor, to specify quality properties related to business processes functions at this very early stage.

A *framework* is described in [22] for SPL developments including the SPL scoping phase, to specify what the SPL can or cannot do, by defining those behaviours or aspects that will be incorporated or eliminated from the SPL. Scoping defines the long-term feasibility of the SPL; it starts with a broad document, which is being refined in the measure that more domain knowledge is captured. The goal is to establish a limit for the SPL to achieve business and market goals. Our research aims to establish a "structure" for this document (input/output artefacts and their structure, techniques used, etc.), since a precise definition was not found in the literature; it will be used as a valid input to reduce the effort in the DRE and DD phases. Hence the scoping document will result an important asset.

The proposed framework suggests the following activities for SPL scoping:

- Workshops and interviews with the stakeholders
- Examine existing products (bottom-up approach [23])

<sup>1</sup> Business process Modeling Notation

- Context diagrams
- Develop a matrix of attributes/products
- Develop SPL scenarios

where a context diagram represents relevant entities related with the SPL w.r.t the users of the products; an attributes/products matrix is used to define the variability of the SPL; scenarios are used to identify interactions that are common to all products and those that are variants.

The Bjørner's domain modelling [20] used in our Domain Scoping step is compliant with most of the above requirements, excepting for - Examine existing products, which is part of the Portfolio Scoping step, where a bottom-up approach will be used [12][23]; however, this point will not be treated here, being the object of an on-going work.

In [24], a specific method based on PuLSE, Pulse-Eco, is presented. This process uses a DE bottom-up development strategy; we are actually focusing on a DE top-down approach [23] combined with a bottom-up approach to study existing products [12], which is recommended in SPL scoping by the new standard [11]; this study however, can be performed on the basis of an "agile" bottom-up process [25].

In [26], three domain engineering approaches were compared, two of them related to SPL, namely, Pohl et al. [9] and the ISO/IEC standard [11], the other one, the classic Bjørner's DE approach [20] strongly based on business processes modelling. From this comparison, [20] was found suitable to integrate the Domain Scoping step of the ISO/IEC 26550 SPL Reference Model [11], and quality was included as a new facet; however, since quality is involved in all facets used in [20] to specify the domain model from different stakeholders' viewpoints; in our present work, quality is specified as an intrinsic facet descriptor and the whole approach is illustrated with a complete case study.

### III. SPL DOMAIN ENGINEERING GUIDELINES WITH THE NEW STANDARD ISO/IEC 26550

The complete DE lifecycle will be briefly described in what follows, according to the ISO/IEC 26550 guidelines [11]; however in this work, only the Domain Scoping step of the PL Scoping phase will be treated in details.

According to the SEI<sup>2</sup> definition [22][27][28], PL Scoping is itself a core asset. In SPL development, scoping is a fundamental activity that will determine the long-term viability of the SPL. Like scoping in general, PL Scoping determines what's "in" and what's "out" of the SPL. The scope definition identifies those entities with which products in the SPL will interact (that is, the product line context), and it also establishes the commonality and sets limits on the variability of the products in the SPL. The scope definition usually begins as a broad, general draft document that is refined as more knowledge is captured and more analysis is performed. For example, for an SPL for a Web-based system, browsers would definitely be "in". Aircraft flight simulators instead, would definitely be "out"; the PL scope may not come into sharp focus all at once. The goal of the scope definition is to draw the

boundary between "in" and "out" in such a way that the SPL satisfies its business and market goals.

Five phases are considered in ISO/IEC 26550; notice that phases and sub-phases can be performed in the order established by the company or organization requiring the SPL [11].

#### A. Product Line Scoping (PL Scoping)

It defines SPL boundaries for DE, envisioning major common and variable features to all products within the SPL. Economical aspects are analysed and the commercialization of the SPL product family is planned; PL Scoping is responsible of the whole SPL management and consequent evolution: it involves 3 sub-phases:

1) *Product Portfolio Scoping*: 1. Identify products that the SPL should be developing, producing, marketing and selling (product "roadmap"); 2. The study of common and variable feature of existing products should provide guidance to meet business objectives and face SPL evolution; 3. Schedule for introducing products to the market. It is input to Domain Scoping.

2) *Domain Scoping*: identify and bound functional or organizational areas that SPL will impact to provide sufficient reuse potential to justify the SPL creation.

3) *Assets Scoping*: identify the boundaries of core assets, providing a first glance at common and variable assets. It identifies reusable assets and calculate the cost/benefit estimated from each asset in order to determine whether an organization should launch an SPL.

Major outputs of PL Scoping are: the asset proposal; it includes major assets (functional areas and high-level common and variable features of all SPL products) that will be included in an SPL with their quantified costs and benefits estimation results. The features defined in the asset proposal directly affect Domain Requirements Engineering (DRE) and Application Requirements Engineering (ARE) shown in Figure 1. More than one asset proposal can be made to find out an optimal set of products and assets. The asset proposal defines also a schedule for delivering specific products to customers and for bringing them to market.

#### B. Domain Requirements Engineering (DRE)

It has to adhere to the specification of the SPL's high-level features provided by PL Scoping. Based on these features, it creates detailed common and variable requirements sufficient to guide subsequent Domain Design (where the SPL RA is designed), realization and testing phases. It involves 5 sub-phases: - Domain requirements elicitation, - Domain requirements analysis, - Domain requirements specification, - Domain requirements validation, and - Domain Requirements Management (to handle changes in requirements).

#### C. Domain Design (DD)

It draws upon the specifications to develop an SPL architecture that enables the realization of the planned commonality and variability within the SPL. The main goal is to produce the RA, defining general SPL structure and

<sup>2</sup> Software Engineering Institute, MIT, Carnegie Mellon

textures. RA reflects additional internal variability introduced by technical solutions besides the external variability, i.e., commonality and variability in the user's perceived requirements. It involves the sub-phases: - RA design, - RA evaluation (quality assurance technique), and - Domain Design Management (to handle changes in the RA design).

#### D. Domain Realization/Implementation

Design and implementation of reusable loosely coupled software components and configurable interfaces, implementing common and variable artefacts offered by RA. Domain realization includes configuration mechanisms to realize variability domain implementation, such as building and buying components supporting the RA infrastructure. They are not yet executable applications.

#### E. Domain Testing/Validation

It validates the domain artefacts created in previous phases and generates domain test artefacts that can be reused later on in Application Testing. Testing in this context means review, validation and verification of artefacts as well as eventually testing some available implementations.

In this work, only the Domain Scoping sub-phase, within the PL Scoping phase will be applied to a case study, to illustrate our approach.

### IV. THE PL SCOPING PROCESS: PLScoP

The adaptation of the ISO/IEC 26550 PL Scoping phase is constituted by the PLScoP process that is outlined in what follows:

#### A. PLScoP Context

The PL Scoping guidelines of the ISO/IEC 26550 standard were completed by integrating to the Domain Scoping sub-phase, the stages described by Bjørner [20] for classic DE for single software systems, not considered for an SPL context. However, they provide a nice technique to specify domain knowledge, based on facets and stakeholders' viewpoints [26].

A *facet* is defined in [20] as a finite set of generic forms of describing a domain from different stakeholders' perspectives or viewpoints, namely, business processes, support technology, management & organization, rules & regulations, human behaviour, and intrinsics; complete definitions of these terms were presented in [26].

The facet notion is not new in Software Engineering [29]; according to [20], each facet represents a view of the domain from different perspectives; the union of these views conforms the complete domain view, called Domain Model; moreover, the special *intrinsics facet* is a facet containing descriptors or attributes (entity, function, event, behaviour) necessary to describe all the other facets (see Table I); the intrinsics notion has allowed us to include software quality, specified by the ISO/IEC 25010 standard [13], as a new intrinsic descriptor. Software quality is then considered to specify all other facets.

**Table I:** Intrinsics to Describe all Domain Facets with the New Quality Descriptor

Intrinsics descriptors	Description
Entities	Represent the phenomena and concepts of the domain
Functions	Operations (actions) performed on the entities
Events	Imply changes in entities by function invocations, i.e., actions in the domain
Behaviour	Sequences of actions and events affecting domain entities
Quality	Specified by the standard quality model ISO/IEC 25010 [13]. It is associated to the Business Processes facet as quality goals (obtained from NFR) required by FR (functions, activities or tasks), to facets Support Technology as quality supported by architectural styles, patterns or mechanisms involved, and Rules & Regulations as quality required by domain business rules [26]. Product quality must be specified to guarantee that SPL products will hold acceptable industrial quality levels.

In particular, this work involves Business Processes, Support Technology, and Rules & Regulations facets, since the final aim of DE is to build an SPL reference architecture; Management & Organization and Human Behaviour facets can be also described in terms of quality, using models, such as CMMI<sup>3</sup> where organizational practices are deeply involved, but this topic is outside the scope of the present work.

Notice that in the Domain Scoping adaptation from [11] integrating Bjørner's domain development [20], a huge number of business processes can be derived from the so called Domain Description Units (DDU) specifications from the declarations of different stakeholders groups expressing their viewpoints. However, this complete specification is outside the PL Scoping spirit, which aims to offer a quick "glance" of the SPL feasibility and limitations, hence only the presentation of few basic modelling elements are considered sufficient to illustrate the "How to do" of our process.

#### B. PLScoP Context

The adaptation of the ISO/IEC 26550 PL Scoping guidelines, is presented in what follows; notice that Asset Scoping and Portfolio Scoping are left as the last steps, since according to [11], the order in which they are executed depends on the organization building the SPL, and we have major interest here in Domain Scoping; however, if a study of existing market products for the domain has to be done, Product Portfolio Scoping should be executed first to perform this study, and its output should be input to the Domain Scoping step.

#### PLScop process

1. *Domain Scoping:*  
*Input:* Domain informal documentation provided by the organization requiring the SPL, Domain Quality Model (DQM) specified by ISO/IEC 25010 [13].
  - a) *Stakeholders Identification:*

<sup>3</sup> Capability Maturity Model Integration

*Input:* visits, interviews, workshops, questionnaires (specific techniques for each one of these activities should be specified).

- Identify groups of stakeholders with similar interests in the organization requiring the SPL.

*Output:* list of stakeholders' groups type: text

b) *Domain Acquisition:*

*Input:* List of Stakeholders' groups

- Capture and gather information from stakeholders into *declarations* to build Domain Description Units (DDU) for each stakeholder viewpoint;

*Output:* DDU type: table

c) *Domain Analysis:*

*Input:* DDU

Analyse DDU, study possible inconsistencies, and business processes are extracted first from DDU [20], to specify the Business Processes facet from a stakeholder viewpoint relevant to the domain; it is represented by a table, UML [30] and BPMN diagrams, using intrinsic facet descriptors. Quality issues are included as a new intrinsic facet descriptor.

Then other facets, relevant to the domain, are also specified according to domain specific stakeholders' viewpoints; for example, in our case they will be *Support Technology* from the *Domain Engineers viewpoint*, and *Rules & Regulations* from the *Directors of healthcare governmental institutions viewpoint*.

The business processes extracted from the DDUs for these stakeholders' viewpoints are identified among the behaviours present in the facets, and specified as new Business Processes facets, considering the respective stakeholders' viewpoint.

*Output:* Facets specifications type: table; UML and BPMN diagrams for business processes

d) *Domain Modelling:*

*Input:* Facets specifications

A domain description is obtained from all the facets specifications by intrinsics; this document should be focused on commonality and variability of the entities involved. According to [20] a domain model is a meaningful domain description; it will be represented integrating the BPMN specifications for all business processes considered.

*Output:* Domain Model type: BPMN. To have a more general specification of the domain model, including all facets, an ontological approach could be considered.

2. *Asset Scoping*

*Input* Domain Model specification

Information on core assets will be extracted from domain model to conform the SPL Core Asset Repository, which will be informally described into the Asset Proposal document, analysing here also economical factors for the SPL feasibility.

*Output:* Asset Proposal document

3. *Product Portfolio Scoping*

*Input:* Available documentation on existing products

Existing products in the domain are assumed to exist; they will be studied to infer about the SPL products that can be developed, main capabilities and limitations; this study could be preformed applying an extractive or bottom-up process to construct automatically a draft *candidate architecture (CA)* [31] using reengineering techniques to handle similarity analysis of the products' components [12]. Notice that since in general widely used products on the

market are considered, results of the existing products study are included into the market study on the SPL feasibility.

*Output:* Product Roadmap: CA: type: graph or UML diagram

Notice that the Product Roadmap artefact includes the candidate architecture artefact which is a first broad draft of the RA that can be built [12]. This possibility should be considered, to have also an additional input to the DRE phase, thus reducing the required effort there. This initial candidate architecture has imbedded the domain knowledge, extracted from existing products about main common and variant components, which can be enriched with the more general information obtained from the Domain Model. It will become also part of the Asset Proposal.

C. *PLScoP: Advantages and Limitations*

If a first candidate architecture can be constructed or is available for a domain, our PLScoP, and in particular the Portfolio Scoping step, can be transformed into a process to perform the RA evolution, i.e., management of changes, in DRE and DD phases. The analysis of existing products is not an easy task, and it depends much on the available market products documentation, requiring a considerable effort to apply reengineering techniques [12]. If this draft architecture is not available, the Domain Scoping step of our PLScoP is still crucial to construct a detailed Domain Model that will help to delimit clearly the SPL scope and functional and non functional granularity, to reduce the effort in the subsequent DE phases where the RA is built.

The advantage of our process is to have combined top-down and bottom-up approaches to specify the domain: top-down is considered in the philosophical Børner's approach [20], which starts with the domain decomposition into organizational business processes specified by intrinsics descriptors, and it is generally used in SPLE [9][11]. The bottom-up approach is proposed to be used in the Portfolio Scoping step also in [9][11], to have a broad picture of the SPL present and future products, by studying domain existing products in the organization proposing the SPL or in different organizations with similar domains. In this sense, our proposition takes advantages from the combination of [9] and [11], reducing general weaknesses of DE lifecycle.

V. APPLICATION TO THE HEALTHCARE INFORMATION SYSTEMS DOMAIN

In what follows, the Healthcare Information Systems (HIS) Domain for the SPL will be briefly discussed to be applied to illustrate PLScoP.

A. *SPL Domain: Healthcare Information Systems (HIS)*

HIS [32] are software intensive systems, i.e., complex integrated information systems, generally located in different and distant institutions and with mandatory (priority) NFR, such as interoperability, availability and security. Interoperability (technical), the HIS crucial quality property for *Electronic Health Records (EHR)* management and sharing, is the ability of two or more systems or components to exchange information and to use the exchanged information; semantic interoperability refers to use a common terminology or

language to communicate systems; process interoperability incorporates business processes and healthcare professionals must standardise business rules to ensure that health information is properly recorded, such as the transfer of information between systems is consistent and complete [33]. The general architecture is a hybrid event-based style, SOA<sup>4</sup>/Layers [12][31], see Figure 2 (from Wikipedia). HIS must facilitate transparent sharing of different kinds of medical information such as EHR and laboratory&imaging results, offering also telemedicine services that can be performed online at remote locations, with wide support of information technology. The use of standards such as HL7, HL7 CDA, LOINC<sup>5</sup>, and DICOM<sup>6</sup> are mandatory for interoperability of EHR and laboratory&imaging results [25][32][33][34].

Nevertheless, in actual medical practice, SPL for HIS have not yet been completely defined, developed and adopted; the lack of agreement on medical standards and psychosocial issues makes difficult the interoperability of EHR, and HIS general adoption is still difficult, even if specific laws and regulations towards these goals have been promulgated worldwide.

### B. Software Quality Modeling in SPL Domain Scoping

Quality has been defined in general as a level of excellence, conformance with specifications, requirements satisfaction, defect free, accomplishment of customer demands [35], and also related to human aspects such as usability and satisfaction [36]. The early specification in PL Scoping of software quality will facilitate to map this quality into all subsequent DE activities; this information on domain quality will be specified as a Quality Model (DQM), reflected into the Domain Model, and registered in the Asset Core Repository. As we have already pointed out, quality assurance is crucial in an industrial software production context to guarantee SPL evolution and the massive assets reuse, impacting on the quality of all products of the SPL family [37].

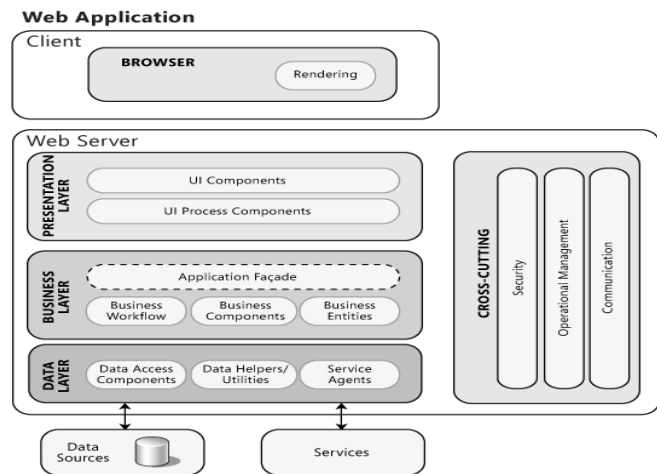


Figure 2: Hybrid Architecture SOA/Layers

The ISO/IEC 25010 Quality Model standard [13] to specify software product quality is part of the SquaRE series of

<sup>4</sup> Service-Oriented Architecture

<sup>5</sup> Logical Observation Identifiers Names and Codes

<sup>6</sup> Digital Communication in Medicine

standards of the International Standards Organization (ISO). This series focuses on quality, requirements and evaluation of software products. It states compatibility with other ISO standards on quality measures and process quality [38]. The central document of the series is the known ISO/IEC 25010 Quality Model [13], describing a hierarchical framework where quality is decomposed into levels of characteristics, sub-characteristics, etc., until the attributes or measurable elements. The Product Quality Model will be used here, since we are in a software development context; Figure 3 [25][32], shows its adaptation to specify the HIS domain quality w.r.t. the SPL family of software products.

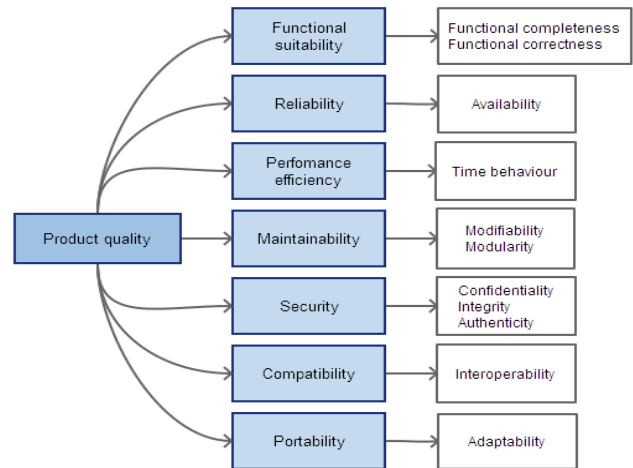


Figure 3: ISO/IEC 25010 HIS Domain Quality Model (HIS-DQM)

### C. Case Study Scope

For this work, the HIS domain for the SPL is restricted to its basic functionalities (EHR-HIS), namely EHR management, patient attention with patient appointment scheduling and capture of demographic data, emission of medical reports, basic administrative services for patient attention; imaging and laboratory services, hospital rooms management, nursing services, urgencies, general hospital administration, etc. will not be considered here [25][34]. Note that the example has been simplified, since only HIS elements at a high granularity level have been treated, avoiding low-level details to facilitate the illustration of our approach, following also the spirit of the PL Scoping phase.

On the other hand, the main facets that will be specified in this work are Business Processes, Support Technology, and Rules&Regulations. The viewpoints considered are: Doctors' group for Business Processes, Domain Engineers' group for Support Technology, and Directors of healthcare governmental institutions for Rules&Regulations. Facets, stakeholders' groups, and viewpoints were selected on the bases of the SPL DE development process, whose main goal is to design a Reference Architecture.

### D. Application of PLScop to the HIS Case Study

Following the basic steps defined in the *Domain Scoping* sub-phase, we have:

1) *Stakeholders Identification*: The technique used consists in doing the expertise of domain engineer extracted from visits,

interviews, workshops or questionnaires. Different viewpoints are identified to conform the stakeholders' groups: Doctors, Domain Engineers and Directors of Healthcare governmental institutions.

2) *Domain Acquisition*: The technique used consists in statements or Declarations formulated by stakeholders to illustrate their viewpoints.

*DDU construction:*

The declarations are grouped into the *Domain Description Unit (DDU)* and they are represented textually as a table (see Tables II, III, and IV) for each one of the identified stakeholders' groups.

**Table II:** DDU for EHR-HIS Domain from the Doctors' Viewpoint

EHR-HIS Domain Description Unit	
Viewpoint	Stakeholders' Group: Doctors
<i>Declarations</i>	
Patient is attended in hospital under scheduled appointment	
If it is the patient first appointment, a new EHR must be created by nurse, else nurse retrieves patient existing EHR	
Patient EHR is accessed by doctor	
New exams and laboratory results can be added to patient EHR by doctor, if it is the case	
Diagnosis and medical orders for patient are produced by doctor to conclude medical attention	
A new appointment is scheduled by nurse if required by Doctor	
Medical equipment and material can be required by doctor to provide adequate medical attention	

**Table III:**DDU for EHR-HIS Domain from the Domain Engineers' Viewpoint

EHR-HIS Domain Description Unit	
Viewpoint	Stakeholders' Group: Domain Engineers
<i>Declarations</i>	
HIS is supported by an hybrid architecture SOA/Layers	
The HIS architectural style supports mainly modifiability, interoperability, performance (time-behaviour), and security services are provided by Internet protocols, crosscutting all layers; availability however depends on internet connection;	
Transmission Layer is managed by a Web Server that communicates all other layers	
Clients access HIS by a browser in Presentation Layer, which connects to the Transmission Layer via a Web Server	
Medical units should have wide range internet and intranet access	
Main EHR-HIS functionalities must be supported: patient appointment services, EHR management and emission of medical reports.	

**Table IV:**DDU for EHR-HIS Domain from Directors of Healthcare Governmental Institutions' Viewpoint

EHR-HIS Domain Description Unit	
Viewpoint	Stakeholders' Group: Directors of Healthcare Governmental Institutions
<i>Declarations</i>	
Digitalize EHR with standard format to achieve sharing among doctors and national and international healthcare institutions	
Have Database of national and international specialists	
Develop a Web platform to manage on-line appointment services	

3) *Domain Analysis*: The technique used consists in the initial domain knowledge is captured from the DDU's (see Table II, III and IV, and described textually in Table III as business processes.

*Facets specification:*

In Table V, business processes specific to the Business Processes facet are derived first [20] from DDU, considering the Doctors' viewpoint. Tables VI and VII will specify business processes specific to facets Support Technology and Rules & Regulations, respectively; only one process will be specified for each stakeholder's group viewpoint to abridge this presentation.

**Table V:** Business Process Derived from the DDU of Doctors' Viewpoint

Viewpoint	Stakeholders' Group: Doctors
<i>Business Process</i>	<i>Description</i>
Appointment Services	From the arrival of a patient to hospital to attend a scheduled appointment, check or create new patient EHR including capture of demographic data and general patient information by nurse; EHR management, medical consultation, diagnosis, emission of medical order

**Table VI:** Business Process Derived from the DDU of Domain Engineers' Viewpoint

Viewpoint	Stakeholders' Group: Domain Engineers
<i>Business Process</i>	<i>Description</i>
EHR Management	Consider in the User Interface (UI) component in Presentation Layer, the access to the EHR Management system in Process Layer; provide EHR access, modification, sharing and registering in database in Data Layer. A Transmission Layers should be present for network services, and it crosscuts all other layers.

**Table VII:** Business Process Derived from the DD of Directors of Healthcare Governmental Institutions' Viewpoint

Viewpoint	Stakeholders' Group: Directors of Healthcare Governmental Institutions
<i>Business Process</i>	<i>Description</i>
On-line appointment services	Provide precise appointment services with specific specialist; handle requests' volume, provide secure access to appointment services; have wide-range and reliable connection facility; have a friendly user interface

Different notations can be used to specify facets from stakeholders' viewpoints with intrinsic descriptors, each one offering different specification granularity; from each specification, more details are extracted; in this work the following notations will be used:

- informal textual specification by tables (see Tables VIII, X, and XI),
- semi-formal UML [30] diagrams (see Figure 4, 6, and 7) for all facets,
- semi-formal BPMN [14] diagrams for business processes in the Business Process facet (see Figure 5 [39]).

In this context, a semi-formal notation language means that it has well-defined syntax and semantics; however it cannot be verified mathematically. Formal languages, such as VDM, Z, B, and RAISE/RSL, also mentioned in [20], are used to specify high assurance systems, to reduce errors in requirement definitions of safety-critical software systems. The HIS domain is basically constituted by non-safety-critical integrated enterprise systems; hence we chose UML and BPMN standard

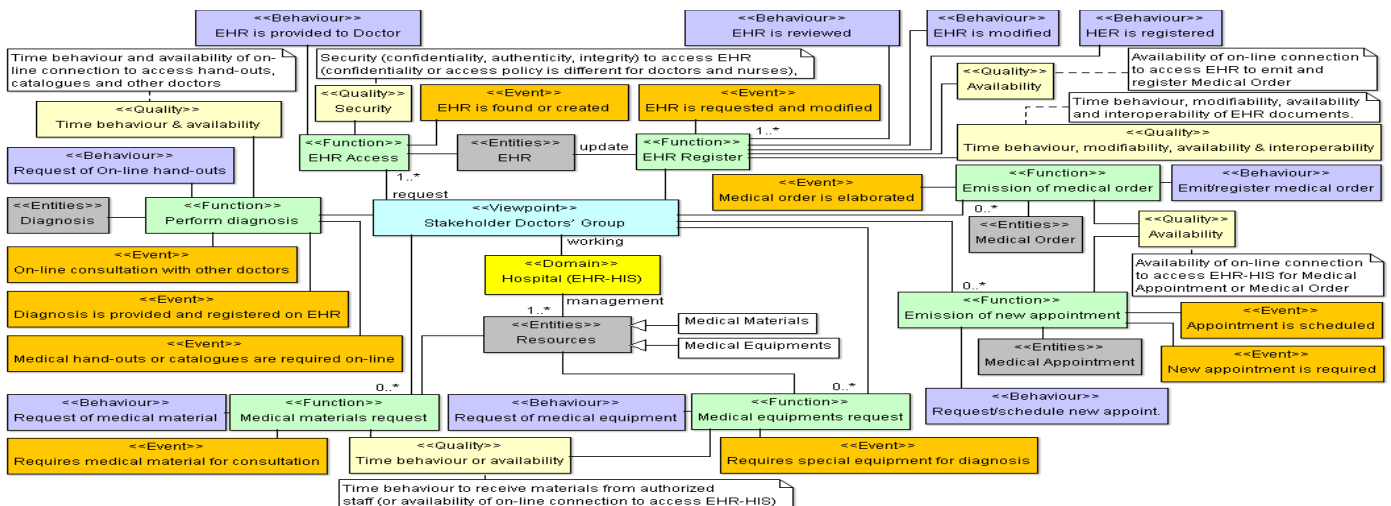


notations, widely used by the software community in this domain.

**Table VIII:** Textual Specification of Doctors Viewpoint for Appointment Services Business Process Facet Specified by Intrinsic

Domain	EHR-HIS – Healthcare institution Requiring the SPL			
Viewpoint	Stakeholders' Group: Doctors			
Appointment Services Business Process Facet				
Entities	Functions	Events	Behavior	Quality
EHR	EHR Access  EHR Register	<i>EHR is found or created</i>  <i>EHR is requested by doctor; doctor attends patient medically and EHR is modified</i>	Nurse confirms the existence of patient EHR or creates an EHR for new patient; nurse provides EHR to doctor ( <i>EHR is provided to doctor</i> )  Doctor reviews patient EHR, adds laboratory and/or examinations results, if any, to patient EHR; patient is attended medically; EHR is modified ( <i>EHR is reviewed, modified registered</i> )	<i>Security (confidentiality, authenticity, integrity) to access EHR: confidentiality or access policy is different for doctors and nurses, authenticity is needed to identify user, and integrity is required for data consistency.</i> <i>Availability of on-line connection to access EHR, time behavior to quick access to EHR, modifiability to change EHR, availability-persistency to retrieve always EHR and interoperability for EHR sharing.</i>
Diagnosis	Perform diagnosis	<i>Medical hand-outs or catalogues are required on-line; on line consultation with other doctors in different healthcare institutions or locally to perform diagnosis; diagnosis is provided and registered on EHR</i>	Doctor requests on-line hand-outs or catalogues; he can consult on-line with other doctors in different healthcare institutions or local doctors, and patient's EHR must be shared by other doctors; review laboratory and/or examinations results; doctor produces diagnosis( <i>Request of on-line hand-outs</i> )	<i>Time behavior and availability of on-line connection to access quickly hand-outs, catalogues and other doctors</i>
Resources: Medical Materials (*)	Medical materials request	Doctor <i>requires medical material</i> for consultation	Doctor requests medical materials calling authorized staff (or using the EHR-HIS system if this facility is available) ( <i>Request of medical material</i> )	<i>Physical availability of materials, time behavior to receive materials from authorized staff (or availability of on-line connection to access the EHR-HIS system for materials request, if this facility is supported)</i>
Resources: Medical Equipment (*)	Medical equipment request	Doctor <i>requires special equipment</i> for diagnosis	Doctor requests medical equipment calling authorized staff (or using the EHR-HIS system if this facility is available) ( <i>Request of medical equipment</i> )	<i>Physical availability of equipment, time behavior to receive equipment from authorized staff (or availability of on-line connection to access the HER-HIS system for equipment request if this facility is supported),</i>
Medical Appointment	Emission of new appointment	<i>New appointment is required</i> for patient, if necessary; appointment is scheduled	Doctor registers patient for a new appointment; the appointment is scheduled. ( <i>Request/schedule new appoint.</i> )	<i>Availability of on-line connection to access EHR for Medical Appointment</i>
Medical Order	Emission of medical order	<i>Medical order is elaborated</i> for present consultation	Doctor emits medical order ( <i>Emit/register medical order</i> )	<i>Availability of on-line connection to access EHR to emit and register Medical Order</i>

(\*) These services will not be considered for EHR-HIS in this work; the abridged behavior name is specified within ( ) in italics



**Figure 4:** UML Specification for Doctors' Viewpoint of the Appointment Services Business Process Facet

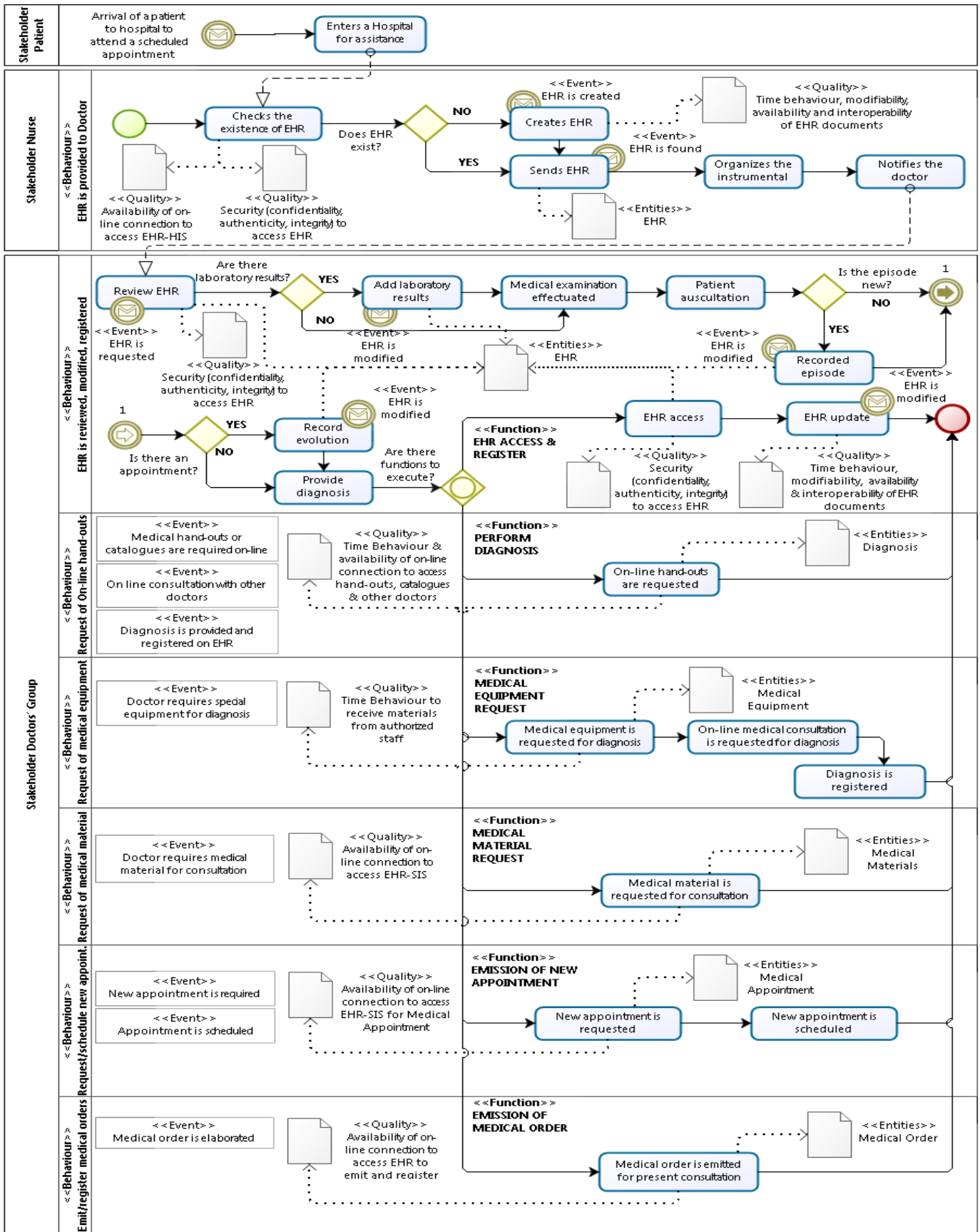
















Figure 5: Appointment Services Process from Doctors' Viewpoint Specified from the Intrinsic Facet, Expressed in BPMN

Table IX: BPMN Notation Used in the Business Process Facet Specification

BPMN Notation	Symbol	Description	Interpretation (for Figure 5)
	Task	Simple or atomic activity representing the work in an organization. They consume resources; it is not detailed further.	They can be mapped into SPL RA architectural components or services
	Exclusive gate	Control element of data workflow. A unique path is selected among several alternatives	It represents the SPL variability.
	Inclusive gate	Control element of the data workflow. One or more path (s) can be selected from several alternatives	It represents the SPL variability.
	Starting event	It initiates a process	It represents the starting or entry point of a business process.
	Starting message event	A process initiates when a message is received	It represents the events arising from an active business process.
	Ending event	It indicates the end of a workflow	It represents the end of some behavior in the business process.
	Terminal ending event	It indicates that a process ends, even if there are active workflows	It represents the end of a business process.
	Intermediate link event	It allows the connection of two process sections.	It allows a connection between two sections of the business process.
	Swimlanes (Pool)	It is a process container; it represents participant, entity or role.	It represents the stakeholders' viewpoints.
	Swimlanes (Lane)	They are pool's subdivisions; it represents participants within an organization.	It represents the different behaviors of the business process.
	Connector object: Sequence	It represents the workflow and the sequence of activities.	It represents the execution flow of the activities in a business process.
	Connector object: Message	It represents interactions among processes or pools.	It represents changes of stakeholders' viewpoints.
	Associations	They are used to relate additional information on the process.	They relate entities, activities or functions with required quality properties.
	Artefact: data object	They provide additional information on the process; they show the information required by an activity, such as input/output.	Document specifying the quality required by entities, activities or functions, in each behavior.

Other business processes can be derived from the analysis of the other facets, such as the on-line appointment services process (see Table VI) from the DDU representing the viewpoint of Directors of healthcare governmental institutions, or the construction of EHR management system (see Table VII) process from the DDU representing the viewpoint of Domain Engineers; however, they will not be considered for this study to abridge the presentation.

Table IX describes the BPMN symbols used in the *Appointment Services Process from Doctors' Viewpoint*, specified from the intrinsics facet. A glimpse on SPL variability can be inferred from the inclusive and exclusive logic gates, since they reflect alternative workflows; they imply a sequence of actions to be performed to achieve a functionality, i.e. functional variability; however, since each activity has associated its quality property, this also can imply non functional variability. This point has to be signalled, because in the domain modelling by the intrinsic descriptors, which was represented in UML (see Figure 4), variability cannot be shown.

In consequence, the use of BPMN is advantageous for our approach, because it contributes to show variability at business process level, which will be mapped later-on into the SPL RA variability model.

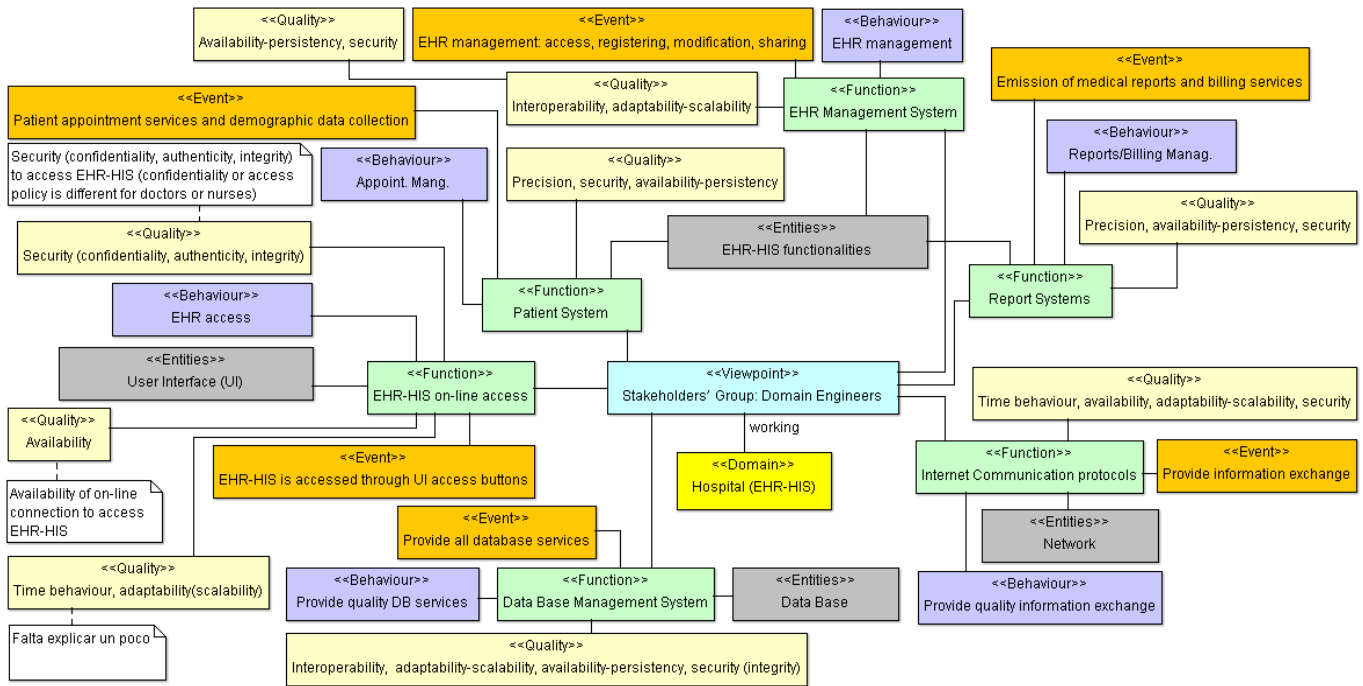
4) *Domain Modelling*: The technique used consists in the facet specifications with the intrinsics, by tables and UML diagrams, obtained in step (c) (see Tables VIII, X and XI, and Figures 4, 6, 7). Notice that we have only the Appointment Services Business Process facet, which is specified in BPMN in Figure 5, and it is an example of a Domain Model.

However, to have the complete Domain Model picture, the other business processes derived from DDU in Tables VI and VII, EHR management, from the Domain Engineer viewpoint and On-line appointment services, from the Directors of healthcare governmental institutions viewpoint respectively, are found as behaviours in the corresponding facet specification (see Tables X and XI), and they can be specified by intrinsics as new Business Process facets, as it was done for Appointment Services in Table VIII. From the business process facet analysis, see Table VIII and Figures 4 and 5, and from the BPMN specifications, we obtain information on:

- The clear identification of the involved stakeholders.
  - Possible SPL variants from a particular viewpoint, such as the security quality property (see Figure 5), which can be solved proposing later on different available technological mechanisms or services. Note that quality properties are reflected in all viewpoints specifications due to the intrinsic quality descriptor that has been introduced in the adaptation of the domain modeling approach of [20].
  - Fine-grained functionalities present in the process as "functions" that can be mapped into large-grained architectural components or services with the BPMN "Aggregation" construct.
- Quality that must be satisfied by each functionality present in the process is specified as a comment, since BPMN has no notation for quality properties.
- The entities or objects produced or manipulated from/by functionalities.

**Table X:** Textual Specification by Intrinsic of Domain Engineers' Viewpoint for the Support Technology Facet

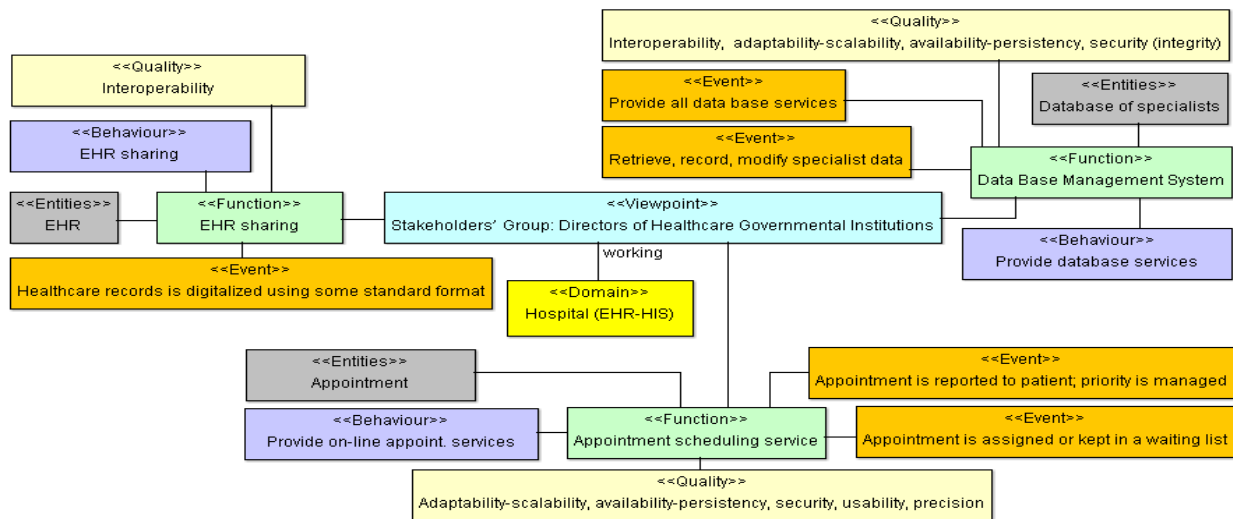
Domain	EHR-HIS – Healthcare institution requiring the SPL			
Viewpoint	Stakeholders' Group: Domain Engineers			
Support Technology Facet				
Entities	Functions	Events	Behaviour	Quality
Presentation Layer: User Interface (UI)	HIS on-line access	HIS is accessed through UI access buttons	Access to HIS main functionalities by authorized persons (EHR access)	Security (confidentiality, authenticity, integrity) to access EHR-HIS (confidentiality or access policy is different for doctors or nurses); availability-persistence of connection to access EHR-HIS; authenticity is required for user identification, and integrity refers to maintain and assure data accuracy and consistency; time behaviour response time to retrieve HER; adaptability-scalability refers to add new medical standards
Process Layer: EHR-HIS functionalities	Patient System  EHR Management System  Report Systems	Patient appointment services including demographic and general information data collection  EHR management: access, registering, modification, sharing  Emission of medical reports and billing services	Scheduling services for patient appointments (Appoint. Mang.)  EHR access, recording, modification, sharing; provide access to medical catalogues for diagnosis, to on-line consultation for diagnosis, emission of diagnosis, consultation and/or addition to EHR of new laboratory & examination results (EHR management)  Medical reports and billing management: edition, access, modification, registering (Reports/Billing Manag.)	Correctness-Precision: for computation algorithm, security, availability-persistence  Interoperability, adaptability-scalability, availability-persistence, security  Correctness-precision, availability-persistence, security
Data Layer: Data Base	Data Base Management System	Provide all database services	Allows interoperability, adaptability, persistence, integrity for database services (Provide quality DB services)	Interoperability, adaptability-scalability, availability-persistence, security (integrity)
Transmission or communication) Layer: Network	Internet Communication protocols	Provide information exchange	Allows efficient, reliable, secure information exchange; information is exchanged independently from the platform (Provide quality information exchange)	Time behaviour, availability-persistence, adaptability-scalability, security



**Figure 6:** UML Specification by Intrinsic of Domain Engineers' Viewpoint for the Support Technology Facet

**Table XI:** Textual Specification by Intrinsic of Directors of Healthcare Governmental Institutions' Viewpoint for the Rules&Regulations Facet

Domain	EHR-HIS – Healthcare institution requiring the SPL			
Viewpoint	Stakeholders' Group: Directors of Healthcare Governmental Institutions			
Rules & Regulations Facet				
Entities	Functions	Events	Behaviour	Quality
EHR	EHR sharing	Healthcare records is digitalized using some standard format	Healthcare records are shared ( <i>EHR sharing</i> )	<i>Interoperability</i>
Database of specialists	Data Base Management System	Provide all data base services; Retrieve, record, modify specialist data	Allows interoperability, adaptability, persistency, integrity for database services ( <i>Provide database services</i> )	<i>Interoperability, adaptability-scalability, availability-persistency, security (integrity)</i>
Appointment	Appointment scheduling service	Appointment is assigned or kept in a waiting list; appointment is reported to patient; priority is managed	Provide precise appointment services with specialist; handle volume of requests, provide secure access to appointment services; have available connection facility; have a friendly user interface ( <i>Provide on-line appoint. services</i> )	<i>Adaptability-scalability, availability-persistency, security, usability, precision</i>



**Figure 7:** UML Specification by Intrinsic of Directors of Healthcare Governmental Institutions' Viewpoint for the Rules&Regulations Facet

## VI. CONCLUSION

It is known that general frameworks explain the “What to do” about things, but not the “How to do” to make things work. We presents the Domain Scoping sub-phase of the PL Scoping phase within the DE lifecycle, following the SPL Reference Model ISO/IEC 26550. The “How to do” is taken from Bjørner’s domain modelling, which involves the facet notion and the stakeholders’viewpoint, focusing on business process modelling. Our main contribution is the specification of the Domain Scoping step as a systematic and repeatable process, centred on the early specification of quality properties as descriptors involved in all facets, considering their clear traceability, reflected into the BPMN representation; this issue will facilitate later on the reference architecture evolution. Notice also that the derivation of the business process specification in BPMN can be automatized from the UML representation of the facets; it is also a widely known and used notation, to bridge the gap between business processes and their implementation, for example as Web services [14]. It is claimed that the effort spent in PL Scoping will reduce the huge effort required in DRE and DD phases [21]. The Domain

Scoping step of our PLScOp is crucial to construct a detailed Domain Model that will help to delimit clearly the SPL scope and functional and non functional granularity, to reduce the effort in the subsequent DRE and DD phases, where the RA is built. Our Domain Scoping process is applied to a complete case study in the Healthcare Information System domain to illustrate our approach. A more complete specification of PLScOp, and subsequent DRE and DD phases of the DE lifecycle to construct the SPL RA, are on-going works.

## ACKNOWLEDGMENT

We wish to thank the referees for their useful and pertinent comments. Financial support to this research has been provided by the DARGRAF No. 03-8730-2013-2 project of the CDCH (Consejo de Desarrollo Científico y Humanístico), and from the Postgraduate Studies in Computer Science, Faculty of Science, Universidad Central de Venezuela.

## REFERENCES

- [1] I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen, and J. Bettin, (Editors), *Domain Engineering. Product Lines, Languages and Conceptual Models*, Springer, 2013.

- [2] E. Berard, *Essays in Object-Oriented Software Engineering*, New York: Prentice Hall, 1992.
- [3] A. Helfferich, G. Herzwurm, and S. Jesse, *Software Product Lines and Service-Oriented Architecture: A Systematic Comparison of Two Concepts*, in proceedings of the First Workshop on Service-Oriented Architectures and Software Product Lines (SOAPL 2007), Kyoto, Japan, May 2007.
- [4] P. Istoan, G. Nain, G. Perrouin, and J. Jezequel, *Dynamic Software Product Lines for Service-Based Systems*, in proceedings of the 9th IEEE International Conference on Computer and Information Technology (CIT'09), INRIA, France, October 2009.
- [5] P. Istoan, *Software Product Lines for Creating Service Oriented Applications*, Masters Internship at IRISA Rennes research institute, TRISKELL research team, Rennes, June 2009.
- [6] F. Medeiros, E. de Almeida, and S. de Lemos, *Designing a Set of Service-oriented Systems as a Software Product Line*, in proceedings of the Software Components, Architectures and Reuse (SBCARS 2010), Fourth Brazilian Symposium, IEEE, Salvador, Bahia, Brazil, September 2010.
- [7] A. Korff, *Implementing ISO 26550 Model-based*, in proceedings of the 6th edition of the International Academic-Industrial Conference in Complex Systems Design & Management (CSD&M 2015), Paris, France, November 2015.
- [8] T. Käkölä, *Standards Initiatives for Software Product Line Engineering and Management within the International Organization for Standardization*, in proceedings of the 2010 43rd Hawaii International Conference on System Sciences (HICSS 2010), Koloa, Hawaii, USA, January 2010.
- [9] K. Pohl, G. Bockle, and F. Van Der Linden, *Software Product Lines Engineering: Foundations, Principles, and Techniques*, Springer, 2005.
- [10] Q. Munir and M. Shahid, *Software Product Line: Survey of Tools*, Master's thesis, Linköping University, Department of Computer and Information Science, 2010.
- [11] ISO/IEC NP 26550, *Software and Systems Engineering – Reference Model for Software and Systems Product Lines*, ISO/IEC JTC1/SC7 WG4, 2013.
- [12] F. Losavio, O. Ordaz, and V. Esteller, *Quality-based Bottom-up Design of Reference Architecture Applied to Healthcare Integrated Information Systems*, in proceedings of the 2015 IEEE 9th International Conference in Research Challenges in Information Science (RCIS 2015), Athens, Greece, May 2015.
- [13] ISO/IEC 25010, *Systems and Software Engineering -- Systems and Software Quality Requirements and Evaluation (SQuARE) -- System and Software Quality Models*, ISO/IEC JTC1/SC7/WG6, 2011.
- [14] Object Management Group (OMG), *Business Process Model and Notation (BPMN)*, version 2.0, available in: <http://www.omg.org/spec/BPMN/2.0/>, 2011.
- [15] K. Lee, K. Kang, and J. Lee, *Concepts and Guidelines of Feature Modeling for Product Line Software Engineering*, in proceedings of the 7<sup>th</sup> International Conference on Software Reuse (ICSR-7): Methods, Techniques, and Tools, Austin, TX, USA, April 2002.
- [16] S. Apel, D. Batory, C. Kästner, and G. Saake, *Feature-Oriented Software Product Lines*, Springer, 2013.
- [17] N. Siegmund, M. Rosenmuller, M. Kuhleemann, C. Kastner, S. Apel, and G. Saake, *SPL Conqueror: Towards Optimization of Non-functional Properties in Software Product Line*, *Software Quality Journal*, vol. 20, no. 3-4, pp. 487-517, September 2012.
- [18] B. Kitchenham, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*, Version 2.3, EBSE Technical Report, Software Engineering Group, School of Computer Science and Mathematics, Keele University, UK and Department of Computer Science, University of Durham, UK, 2007.
- [19] M. de Moraes, E. de Almeida, and S. Romero, *Systematic Review on Software Product Lines Scoping*, in proceedings of 6th Experimental Software Engineering Latin American Workshop (ESELAW 2009), São Carlos, Brasil, November 2009.
- [20] D. Björner, *Software Engineering 3 Domains, Requirements, and Software Design*, Texts in Theoretical Computer Science, EATCS Series, Editors: W. Brauer G. Rozenberg A. Salomaa, Springer-Verlag Berlin Heidelberg, 2006.
- [21] I. Da Silva, P. Neto, P. O'Leary, E. De Almeida, and S. de Lemos Meira, *Software Product Line Scoping and Requirements Engineering in a Small and Medium-sized Enterprise: An Industrial Case Study*, *Journal of Systems and Software*, vol. 88, pp. 189-206, 2014.
- [22] L. Northrop and P. Clements, *A Framework for Software Product Line Practice, Version 5.0*, Product Line Practice Initiative, SEI (Software Engineering Institute), 2012.
- [23] J. Herrera, F. Losavio, and A. Matteo, *RDS de Enfoques y Técnicas para la Construcción de Arquitecturas en un Contexto de Líneas de Productos de Software*, *Revista Venezolana de Computación (ReVeCom)*, vol. 1, no. 1, pp. 17-25, Junio 2014.
- [24] J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, and J. M. DeBaud, *PuLSE: a Methodology to Develop Software Product Lines*, in proceedings of the 1999 Symposium on Software Reusability (SSR'99), ACM, Los Angeles, USA, May 1999.
- [25] F. Losavio, O. Ordaz, and I. Santos, *Proceso de Análisis del Dominio Ágil de Sistemas Integrados de Salud en un Contexto Venezolano*, *Revista Venezolana de Información, Tecnología y Conocimiento, ENL@CE*, vol. 12, no. 1, pp.101-134, Enero-Abril 2015.
- [26] J. C. Herrera, F. Losavio, and O. Ordaz, *Ingeniería del Dominio con el Estándar ISO/IEC 26550 para LPS Considerando la Faceta Calidad*, in proceedings of the Conferencia Nacional de Computación, Informática y Sistemas (CoNCISA 2015), Valencia, Venezuela, Octubre 2015.
- [27] J. M. DeBaud, *A Systematic Approach to Derive the Scope of Software Product Lines*, in proceedings of the 21st International Conference on Software Engineering (ICSE'99), Los Angeles, CA, USA, May 1999.
- [28] K. Schmid, *Customizing the PuLSE Product Line Approach to the Demands of an Organization*, in proceedings of the 7th European Workshop on Software Process Technology (EWSPT'2000), Kaprun, Austria, February 2000.
- [29] P. Kruchten, *Architectural Blueprints—The “4+1” View Model of Software Architecture*, in proceedings of the Conference on TRI-Ada'95, Anaheim, CA, USA, November 1995.
- [30] Object Management Group (OMG), *Unified Modelling Language Superstructure*, version 2.0 (formal/05-07-04), <http://www.omg.org/spec/UML/2.0>, August 2005.
- [31] M. Shaw and D. Garlan, *Software Architecture. Perspectives of an Emerging Discipline*, Prentice-Hall, 1996.
- [32] Institute of Medicine (US), *Key Capabilities of an Electronic Health Record System: Letter report*, Committee on Data Standards for Patient Safety, National Academies Press, 2003.
- [33] HIQA (Health Information and Quality Authority), *Overview of Healthcare Interoperability Standards*, 2013.
- [34] S. Samilovich, *OpenEMR – Historia Clínica Electrónica de Código Abierto y Distribución Gratuita, Apta para su Uso en el Sistema de Salud Argentina*, JAIHO CASI, 2010.
- [35] M. Allauddin, F. Azam, and M. Zia, *A Survey of Quality Assurance Frameworks for Service Oriented Systems*, *International Journal of Advancements in Technology*, vol. 2, no. 2, pp. 188-198, 2011.
- [36] W. Suryn, *Software Quality Engineering: a Practitioner's Approach*, IEEE Wiley & Sons, Inc., Hoboken, New Jersey, 2014.
- [37] H. González, *Integration of Quality Attributes in Software Product Line Development*, Master Thesis en Ingeniería del Software, Métodos Formales y Sistemas de Información (ISMFSI), 2012.
- [38] S. Wagner, *Software Product Quality Control*, Springer, 2013.
- [39] K. I. Farroñay and A. J. Trujillo, *Sistema de Registro de Atención Médica para un Centro de Salud de Nivel I-3 de Complejidad*, Doctoral dissertation, Universidad Peruana de Ciencias Aplicadas (UPC), 2013.



## QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines based on ISO/IEC 26550

### QuaDRA: Diseño orientado por la Calidad de una Arquitectura de Referencia para Líneas de Productos de Software basado en ISO/IEC 26550

Juan Carlos Herrera<sup>1</sup>, Francisca Losavio<sup>2</sup>, Oscar Ordaz<sup>3</sup>

<sup>1</sup> Universidad Bolivariana de Venezuela. Caracas, Venezuela. [jchr1982\(AT\)gmail.com](mailto:jchr1982(AT)gmail.com)

<sup>2,3</sup> Universidad Central de Venezuela. Caracas, Venezuela, [francislosavio\(AT\)gmail.com](mailto:francislosavio(AT)gmail.com); [oscarordaz55\(AT\)gmail.com](mailto:oscarordaz55(AT)gmail.com)

#### INFORMACIÓN DEL ARTÍCULO

*Artículo de Investigación*

Recibido: 17-02-2016  
Correcciones: 21-05-2016  
Aceptado: 30-05-2016

#### Keywords

Software Product Lines; Reference Architecture; Proactive Design; Extractive Design; Quality Requirements; Domain Engineering; ISO/IEC 26550; ISO/IEC 25010; QuaDRA; Integrated Healthcare Information Systems.

#### Palabras clave

Línea de Productos de Software; Arquitectura de Referencia; Diseño Proactivo; Diseño Extractivo; Requisitos de Calidad; Ingeniería del Dominio; ISO/IEC 25010; ISO/IEC 26550; QuaDRA; Sistemas Integrados de Información de Salud.

#### ABSTRACT

The goal of this work, framed in the Domain Engineering (DE), first lifecycle of Software Product Line Engineering (SPLE), is to present and apply a systematic and repeatable process, called QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines (SPL); the new ISO/IEC 26550 standard defining the SPL Reference Model is followed. SPL is a set of software-intensive or complex systems, sharing a common, managed set of features satisfying specific needs of a particular market segment or domain. These features are developed from a common set of core assets, reused in different products or software systems that form the SPL family. The Reference Architecture (RA) is the main asset shared by all SPL products; it covers the family commonality and variability and it is used as a template to produce new products in the Application Engineering (AE) second SPLE lifecycle. QuaDRA is a proactive (top-down) SPL development approach, since it follows SPLE; moreover, it is quality-oriented, because software quality issues are considered early in the PL Scoping first phase of DE, specifying quality as a domain facet intrinsic descriptor, thus providing clear traceability of quality requirements to ease RA evolution. Quality requirements, specified here by ISO/IEC 25010, are in general poorly considered in SPLE approaches, leaving them to the final RA design phase, being however main responsible of the SPL variability. ISO/IEC 26550 promotes PL Scoping (Portfolio, Domain and Asset Scoping sub-phases) as a crucial DE phase, to asses on SPL products to be constructed, risks and economical feasibility. QuaDRA combines two SPL development approaches in PL Scoping, top-down in Domain Scoping with extractive (bottom-up), profiting of refactoring techniques, in Product Portfolio Scoping, to reduce the development effort required in all subsequent DE activities. The major SPL development effort is concentrated in the DE lifecycle; our approach contributes to globally ensure RA quality and reduce the development effort. QuaDRA will be applied to a case study in the Integrated Healthcare Information Systems domain.

#### RESUMEN

El objetivo de este trabajo, enmarcado en el primer ciclo de vida de la Ingeniería del Dominio (ID), de la Ingeniería de Líneas de Productos de Software (ILPS), es presentar y aplicar un proceso sistemático y repetible, denominado *QuaDRA*, del inglés *Quality-oriented Design of Reference Architecture*, para Líneas de Productos de Software (LPS), de acuerdo al estándar ISO/IEC 26550 que define un Modelo de Referencia para la ILPS. Una LPS es un conjunto de sistemas complejos o intensivos de software, que comparten un conjunto común y organizado de características que satisfacen las necesidades de un sector específico del mercado o dominio. Estas características son desarrolladas a partir de un conjunto común de activos, que es reutilizado en diferentes productos o sistemas de software que conforman una familia SPL. La Arquitectura de Referencia (AR) es el activo principal compartido por todos los productos de la SPL; cubre las partes comunes y variables de la familia y es usada como una plantilla para producir nuevos productos en el segundo ciclo de vida de la ILPS, la Ingeniería de la aplicación (IA). QuaDRA se considera un enfoque de desarrollo proactivo (top-down) de LPS ya que sigue la ILPS; además es orientado a la calidad; ésta se toma en cuenta desde la primera fase de Alcance de la LPS en la ID, proporcionando así clara trazabilidad de los requisitos de calidad en todo el proceso, siendo éstos especificados como descriptores intrínsecos para todas las facetas del dominio, facilitando la evolución de la AR. Los requisitos de calidad, especificados aquí por ISO/IEC 25010, en general son poco considerados en los enfoques ILPS, siendo dejados para la fase final de diseño de la AR; sin embargo, son los

mayores responsables de la variabilidad de la LPS. El estándar ISO/IEC 26550 promociona la primera fase de la ID, *Alcance de la LPS (PL Scoping)*, que incluye las sub-fases Alcances del Portafolio, del Dominio y de Activos, donde los productos, los riesgos y la factibilidad económica de la LPS deben ser evaluados. QuaDRA combina dos enfoques de desarrollo de LPS en la fase de Alcance: el proactivo (top-down) en el Alcance del Dominio y el extractivo (bottom-up), con técnicas de reingeniería, durante el Alcance del Portafolio de Productos, para así reducir el gran esfuerzo de desarrollo en las subsiguientes actividades de ID. Por lo tanto, nuestro enfoque contribuye globalmente a asegurar la calidad de RA y a reducir el esfuerzo de desarrollo. QuaDRA será aplicado a un caso de estudio en el dominio de los Sistemas Integrados de Información de Salud.

© 2016 IAI. All rights reserved

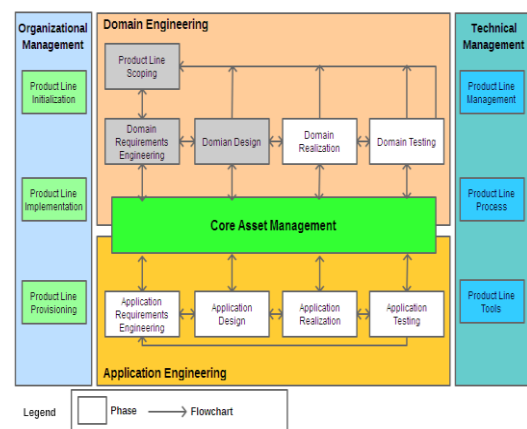
## 1. Introduction

The term domain is often used in reference to a particular knowledge area; an application domain denotes any aspect where computing can be applied [1]; *domain* is defined in [2] as the minimal set of properties describing precisely a family of problems in which a computational application or system is involved for their solution, and this is the definition adopted in this context. The problem of software development based on reusing components or a core of software elements, favoring efficient and reliable development is not new [3] and it is a complex problem not yet completely solved in academic or industrial practices; moreover, there is still a huge gap between research results and their industrial application [4, 5].

Software Product Lines (SPL), or simply Product Lines (PL) [6], is an approach that provides a way of massive personalization of individual solutions from a repository of reusable software assets, in a particular domain. It is inspired in the Fordism technique used to increase production while lowering costs in early 20th century automotive industry. The main goal of SPL development, also called SPL Engineering (SPLE) [7], is to promote reuse exploiting common elements in similar products of the SPL family in a particular domain. The main idea, but not an easy task, is to capture the essential common elements and possible variable issues to construct an evolutionary SPL. Instead of describing a single system, the SPL model describes a set or family of software systems, products or applications. Complex domain analysis must be achieved in order to specify and delimit the family that can be developed from the core of assets, with their commonality and variability [8]. Concrete products or members of the SPL family are developed according to customer requirements, following the instantiable framework provided by the SPL *Reference Architecture (RA)*, where customers should select choices among a generally huge number of feasible configurations [9, 10].

Two main lifecycles drive SPLE: *Domain Engineering (DE)* and *Application Engineering (AE)* [7, 11], see Figure 1. DE is a relatively mature discipline, born long before SPL in the broad context of Requirements Engineering, now called *Domain Requirements Engineering (DRE)* within the DE phase. We will be concerned here with the following DE phases: *Product Line Scoping*, *Domain Requirements Engineering* and *Domain Design*, outlined in grey in Figure 1. The new standard ISO/IEC 26550 defines a Reference Model for SPLE and Management [4, 5, 11], following basically the original SPLE approach of Pohl, Bockle and van der Linden [7] which is a *proactive or top-down* approach, where RA is designed from the capture and

specification of the domain knowledge, in contrast with the *extractive or bottom-up approach*, where SPL is designed from the study of a collection of existing products, which are refactored to conform the RA. We will see that in this work, both approaches are combined to reduce the whole RA development effort.



**Figure 1.** ISO/IEC 26550 Reference Model for Software and Systems Product Lines [11]

The first PL Scoping phase of DE deals with a broad capture of domain knowledge, analysing and identifying the application ambit of the SPL, including risk analysis, economical feasibility and identification of the SPL family of products to be constructed; moreover, PL Scoping becomes necessary, since the knowledge of the SPL domain must be available early to consider the complete commonality and variability modelling of the complete SPL in the next DRE phase, to facilitate this huge and basic engineering step, where the major SPL development effort is placed. RA, which is the main SPL reusable asset, is built in the Domain Design (DD) phase. In particular, in the PL Scoping phase, a bottom-up technique [12] can be used as a technique, focused to capture domain knowledge, to perform the study of existing domain products, facilitating the subsequent activities. Quality assurance should be considered early, because quality related to domain functional (FR) and non-functional requirements (NFR) is responsible in a major degree of the SPL variability; however most of the SPL development approaches leave it to the RA design phase.

According to Figure 1, *Domain Realization* concerns the specification of smaller grained components of RA, called also “SPL platform” in the literature, for example to assign technical architectural solutions or mechanisms from present technology, such as APIs, plug-ins, etc. *Domain Testing* is concerned with RA verification and validation.



The AE lifecycle [11], concerns four phases: *Application Requirements Engineering, Application Design, Application Realization and Application Testing. Core Asset Management* concerns the organization and administration of the Asset Repository, constructed in parallel during DE and AE lifecycles; it is used and enriched in AE during product derivation from RA. Organizational and Technical Management are activities that crosscut both DE and AE.

The main goal of this work is to present and apply a systematic and repeatable process called *QuaDRA: Quality-Design of Reference Architecture*. to construct an SPL RA; this process can be categorized as a proactive or top-down SPL development, since it follows SPLE; moreover, it is quality-oriented, because quality issues are considered early from the PL Scoping phase of DE. The new ISO/IEC 26550 standard guidelines will be followed. However, QuaDRA combines two SPL development tendencies: The Bjørner's proactive or top-down approach [13] in Domain Scoping, with the extractive or bottom-up approach in Product Portfolio Scoping [12], which uses a reengineering technique to perform the study of existing domain market products, supposing that in the domain are available:

- Construct or reuse an initial candidate architecture (CA) for the domain, built by a bottom-up technique [12] from existing market products; the CA artefact is the result of the study of existing products, first activity of Product Portfolio Scoping; CA will be completed (transformed) in subsequent DE phases, namely Domain Requirements Engineering (DRE) and Design (DD), to obtain the final RA.

The combination of bottom-up and top-down approaches will reduce the effort in the subsequent DRE and DD phases, where the major SPLE development effort is concentrated.

Besides this introduction and the conclusion, this paper is structured as follows: the second section is dedicated to the QuaDRA description and its context; a third section presents the QuaDRA application to the *Integrated Healthcare Information Systems (HIS)* domain case study, and finally the fourth section discusses main related works.

## 2. Domain Engineering with QuaDRA

The original guidelines provided by ISO/IEC 26550 are presented first, followed by their adaptation to the QuaDRA process.

### 2.1 Domain Engineering guidelines of ISO/IEC 26550

In what follows, the main phases proposed in ISO/IEC 26550 to achieve the RA construction are outlined:

- **Product Line Scoping (PL Scoping)**  
It defines SPL boundaries for DE, envisioning major common and variable features to all products within the SPL. Economical aspects are analysed and the commercialization of the SPL product family is planned; PL Scoping is responsible of the whole SPL management and consequent evolution: it involves 3 sub-phases:
  - *Product Portfolio Scoping*: 1. Identify products that the SPL should be developing, producing, marketing

and selling (product “roadmap”); 2. The study of common and variable feature of products should provide guidance to meet business objectives and face evolution; 3. Schedule for introducing products to the market. It is input to Domain Scoping.

- *Domain Scoping*: - identify and bound functional or organizational areas that SPL will impact to provide sufficient reuse potential to justify the SPL creation.
- *Assets Scoping*: - identify the boundaries of core assets, providing a first glance at common and variable assets. Reusable assets are identified and the cost/benefit estimated from each asset is calculated, in order to determine whether the SPL should be launched.
- **Domain Requirements Engineering (DRE)**  
It has to adhere to the specification of the SPL's high-level features provided by PL Scoping. Based on these features, it creates detailed common and variable requirements sufficient to guide subsequent domain design (RA design), realization and testing sub-phases. It involves 5 sub-phases: - Domain requirements elicitation, - Domain requirements analysis, - Domain requirements specification, - Domain requirements validation, and - Domain Requirements Management (to handle changes in requirements).
- **Domain Design (DD)**  
It draws upon the specifications to develop an SPL architecture that enables the realization of the planned commonality and variability within the SPL. The main goal is to produce the Reference Architecture (RA), defining general SPL structure and textures. RA reflects additional internal variability introduced by technical solutions besides the external variability i.e. commonality and variability in requirements. It involves the sub-phases: - RA design, - RA evaluation (quality assurance technique), and - Domain Design Management (to handle changes in the RA design).

### 2.2 The QuaDRA process

A reference model or framework, as the one presented in ISO/IEC 26550, specifies the “*What to do*”; however, the “*How to do*” must be conceived in all details, to convert the framework guidelines into systematic and repeatable processes. In what follows, the QuaDRA process adapting the ISO/IEC 26550 guidelines, is presented according to its main sub-phases: 1) PL Scoping (PLScoP), 2) Domain Requirements Engineering (DRE), 3) Domain Design (DD).

#### *QuaDRA process* **Begin QuaDRA**

##### **PL Scoping phase (PLScoP)**

*PLScoP phase*  
*begin PLScoP*

1. *Product Portfolio Scoping to determine requirements for future SPL products:*  
*Begin Product Portfolio Scoping*  
*Input: documentation on existing products;*
- 1.1 *Study of existing market products in the domain to infer about SPL products that can be constructed:*  
*- semantic similarity analysis of domain market products components is made on available documentation;*

- table CCT is constructed, showing components and connectors of considered products, see table 1;
  - apply the automatic extractive bottom-up approach in [12] to construct a Candidate Architecture (CA) from existing product configurations;
  - this bottom-up technique is applied on an undirected and connected graph structure representing products' architectures;
  - CA is also a graph obtained by the union of the graphs representing the architectural configurations of the considered products, see Figure 4; CA is represented in UML and can also be specified as an ontology;
  - construct the domain quality model (DQM), see Figure 3, w.r.t. existing products by instantiating the ISO/IEC 25010 quality model for the domain [14];
  - construct the EQM (Extended Quality Model) table (see Table 2), where CA is documented by listing common and variants components descriptions, required/provided quality properties and constraints; EQM is a different CA view;
- output: CA; DQM; EQM; CCT  
notation: CA type: graph, UML diagram, or ontology; EQM type: table; DQM type: table or graphic;; CCT type: table;
- 1.2 Elaborate marketing study for SPL products.  
input: marketing information; CA; EQM;
- perform marketing study
  - economical factors for the SPL feasibility should be also analysed.
- output: Marketing Study, Product Portfolio  
notation: Marketing Study type: document;
- 1.3 Outline commonality and variability for future SPL products.  
input: CA; EQM;
- commonality and variability for future
  - products will be specified by the Updated EQM table in step 2.4, see table 6;
  - each component of future products' configurations shall be described by an informal document;
- output: Future Products Document;  
notation: Future Products Document type: text  
end Product Portfolio Scoping;
- output: DQM; CA; EQM; Marketing Study; Product Portfolio; Future Products Document
2. Domain Scoping - adapted from Bjørner [13]  
Begin Domain Scoping
- input: DQM; EQM;  
begin
- 2.1 Stakeholders identification:  
input: visits; interviews; workshops; questionnaires (specific techniques for each one of these activities should be specified);
- Identify groups of stakeholders with similar interests in the organization requiring the SPL;
- output: list of stakeholders' groups  
notation: type: list of stakeholders' groups table;
- 2.2 Domain acquisition:  
input: List of Stakeholders' groups;
- capture and gather information from each stakeholder of interest in the domain, into declarations to build Domain Description Units (DDU) for each stakeholder viewpoint;
- output: DDU;  
notation: DDU type: table;
- 2.3 Domain analysis:  
input: DDU;
- analyse DDU, study possible inconsistencies; business processes are extracted from DDU [13], and represented as behaviours in the facet specification by intrinsic descriptors from a stakeholder viewpoint relevant to the domain; the facet specification has three views: table, UML diagram, and BPMN diagrams for processes.
  - software quality [14] is included as a new intrinsic facet descriptor.
  - it is recommended to specify first the Business Processes facet [13]. The other facets considered relevant to the domain are also specified in the same way; in our case they will be Support Technology and Rules & Regulations.
  - business processes extracted from the behaviours in the facet specification are specified in BPMN.
- output: Facet specification;  
notation: Facet specification type: table, UML diagram, BPMN diagram;
- 2.4 Domain modelling:  
input: Facet specification; CA; EQM;  
Future Products Document;
- A Domain Model is obtained from each facet specification; it is represented by BPMN diagrams:
- integrate the BPMN specification for each stakeholder viewpoint by aggregating activities of the lanes of each business process pool; recall that a BPMN "aggregation" groups activities in a process (see Table 5);
  - attach to the respective aggregation, the information on the quality properties related to each activity, as a new activity to be performed by the aggregation to accomplish the quality;
  - obtain a new BPMN aggregation diagram (BPMNAgg) for each stakeholder viewpoint considering aggregations;
  - consider aggregations as possible new functional components or sub-components in CA, connecting them to the appropriate element in the CA configuration; for example, if the new component is the Diagnostic Assistant system, it should be decided to connect it to the EHR management system or to have it as a separate system in Process Layer, and in this case the HIS User Interface should be modified in Presentation Layer; notice that syntax and semantics of CA elements should be maintained;
  - quality properties related to the aggregation should be included as variants of the new components, providing their required quality; it means to establish a new connector in CA between these components;
  - study EQM, the Future Products Document (see Step 1.3), and the Aggregations BPMNAgg, to consider if the new components found contribute to the SPL future products;
  - update CA (UCA) (in all its representations);
  - update EQM (UEQM);
- output: Domain Model: BPMNAgg; UCA, UEQM;  
notation: Domain Model type: BPMN; UEQM type: table; UCA type: same as CA;  
end Domain Scoping;
3. Asset Scoping - to determine the SPL Core Asset Proposal  
Input: CA; UCA EQM; UEQM; DQM; Domain Model BPMNAgg;  
Begin Asset Scoping
- Information for the core asset proposal will be extracted from the input, to construct a Core Asset Repository (CAR);
- end Asset Scoping;  
output: CAR;  
notation: CAR type: database;
- end PLScoP;

## Domain Requirements Engineering (DRE)

### DRE phase

#### begin DRE

1. Domain requirements analysis:  
begin  
input: UCA; UEQM;  
- UCA and UEQM contain precise information on common and variant components, including quality issues;  
  
output: UCA; UEQM  
end;
2. Domain requirements specification:  
begin  
input: UCA; UEQM;  
- UCA and UEQM contain the complete requirements specification;  
  
output: UCA, UEQM;  
end;
3. Domain requirements validation:  
begin  
input: UCA; UEQM;  
- query the UCA ontology, if present, for consistency checking, else another technique should be used;  
- update UEQM;  
end;  
  
output: UCA; UEQM;  
notation: UCA type: ontology;
4. Domain Requirements Management  
begin  
input: UCA; UEQM;  
- update UCA for any change in requirements;  
- update UEQM;  
end;  
  
output: UCA; UEQM;  
end DRE;

### Domain Design (DD)

#### DD phase

#### begin DD

1. RA design: "... Variability support is crucial for RA design, because all common and variable requirements cannot be reflected; one of the basic rules for adding variability into RA is to prioritize requirements. Common requirements to all applications should be satisfied by RA, while variable requirements cannot be because of confliction among some of them. The structure and texture of the reference architecture must support such variability ..." [11].  
  
begin  
input: UCA; UEQM;  
- RA is conformed by common components and variability points [7], constructed by grouping UCA variants performing semantically similar tasks;  
- update UEQM with the information on variation points.  
- RA is an undirected, connected graph, specified in UML; it can be specified as ontology;  
end;  
  
output: RA; UEQM;  
notation: RA type: UML diagram, ontology;
2. RA evaluation: is a quality assurance technique. "... For SPLE, an assessment of RA is crucial, at least for the SPL-related quality requirements. Only an RA that supports quality requirements sufficiently will be evolutionary..." [11].

#### Begin

#### Input: RA

- RA is compliant with quality requirements by construction; each RA component fulfils the required quality; moreover, RA satisfies the global quality requirements of the domain architectural style;  
end;
3. Domain Design Management: "... Aims at managing RA development and maintenance for the artefacts under changes. The major activities for managing domain design can be divided into three activities: - configuration and - change management for domain design and - traceability management. The relations between common and/or variable requirements and RA are not simple one-to-one mapping, therefore traceability between common/variable requirements and RA assets should be maintained at a comprehensible level..." [11].  
  
Begin  
Input: RA  
  
- Requirements traceability is solved by construction;  
end;  
  
output: RA;  
  
end DD;  
  
end QuaDRA;

Roughly, from the number of the QuaDRA activities shown in DRE and DD, we can appreciate that the required effort has been considerably reduced.

### 3. QuaDRA: Application to his domain case study

In what follows, the Integrated Healthcare Information Systems (HIS) Domain will be briefly discussed.

#### 3.1 SPL Domain: Healthcare Information Systems

HIS [16] are software intensive systems, i.e., complex integrated information systems, generally located in different and distant institutions and with mandatory (priority) NFR, such as interoperability, availability and security. Technical interoperability, the HIS crucial quality property for Electronic Health Records (EHR) management and sharing, is the ability of two or more systems or components to exchange information and to use the exchanged information; semantic interoperability refers to use a common terminology or language to communicate systems; process interoperability incorporates business processes and healthcare professionals; business rules must be standardised to ensure that health information is properly recorded, such as to guarantee that the transfer of information between systems is consistent and complete [17].

The general HIS architecture is a hybrid event-based style, Service-Oriented Architecture SOA/Layers [12, 18], see Figure 2. HIS must facilitates transparent sharing of different kinds of medical information such as EHR and laboratory and imaging results, offering also telemedicine services that can be performed on-line at remote locations, with wide support of information technology. The use of standards such as HL7, HL7 CDA, Logical Observation Identifiers Names and Codes LOINC, and Digital Communication in Medicine DICOM are mandatory for interoperability of HER, and laboratory and imaging results [17, 19, 20, 21].

Nevertheless, in actual medical practice, SPL for HIS have not yet been completely defined, developed and adopted; the lack of agreement on medical standards and psychosocial issues makes difficult the interoperability of EHR, and HIS general adoption is still difficult, even if specific laws and regulations towards these goals have been promulgated worldwide.

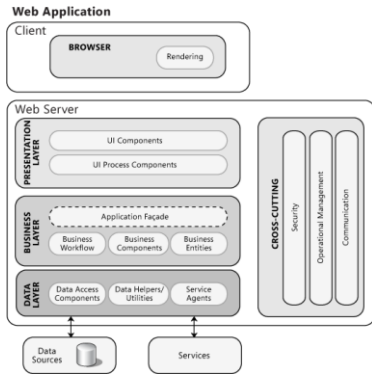


Figure 2. Hybrid architecture SOA/Layers. Source: Wikipedia

### 3.2 Software Quality Modelling in Domain Scoping

Software quality is defined in [14, 22] as the degree to which the software is capable of providing and maintaining a specified level of service. The early specification in PL Scoping of software quality will facilitate to map this quality into all subsequent DE activities; this information on domain quality will be specified as a Quality Model (DQM), reflected into the Domain Model as additional activities to be accomplished, and registered in the Asset Core Repository. As we have already pointed out, quality assurance is crucial in an industrial software production context to guarantee SPL evolution and the massive assets reuse, impacting on the quality of all products of the SPL family [9].

The ISO/IEC 25010 Quality Model standard [14] to specify software product quality is part of the SQuARE series of the International Standards Organization (ISO) [22]. This series focuses on quality, requirements and evaluation of software products. It states compatibility with other ISO standards on quality measures and process quality [23]. The central document of the SQuARE series is the well-known ISO/IEC 25010 Quality Model [14], describing a hierarchical framework where quality is decomposed into levels of characteristics, sub-characteristics, etc., until the attributes or measurable elements. The Product Quality Model will be used here, since we are in a software development context; Figure 3 shows its adaptation to specify the HIS domain quality model w.r.t. the SPL family of software products.

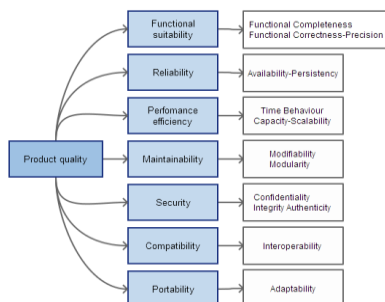


Figure 3. ISO/IEC 25010 HIS Domain Quality Model (HIS-DQM) [14], adapted in [21]

### 3.3 Case Study Scope

The majority of the twelve HIS on the market studied in [16] to establish main HIS requirements, were open-source and had been built considering the interaction episode of doctor-patient encounters; they were all focused on the EHR management as the main functionality. Our present research is based on this assumption. The HIS domain for the SPL is then restricted to its basic functionalities (EHR-HIS), namely EHR management, patient attention with patient appointment scheduling and capture of demographic data, emission of medical orders and reports, basic administrative services, such as billing for patient attention; imaging and laboratory services, hospital rooms management, nursing services, urgencies, general hospital administration, etc., were not been considered in our first studies, where the initial CA was built [12, 21]. The case study has been simplified, since only HIS elements at a high granularity level has been treated, avoiding low-level details to facilitate the illustration of our approach, following also the spirit of the PL Scoping phase.

### 3.4 Application to Product Portfolio Scoping

#### Step 1.1 Study of existing market products

An existing CA for EHR-HIS will be reused for this case study [12], see Figure 4. DQM has been presented in section 3.2, see Figure 3. Now, Table EQM will be constructed; it shows in Table 2 as a broad textual overview of CA, where components will be specified with their required quality, including priorities and architectural solutions providing these qualities.

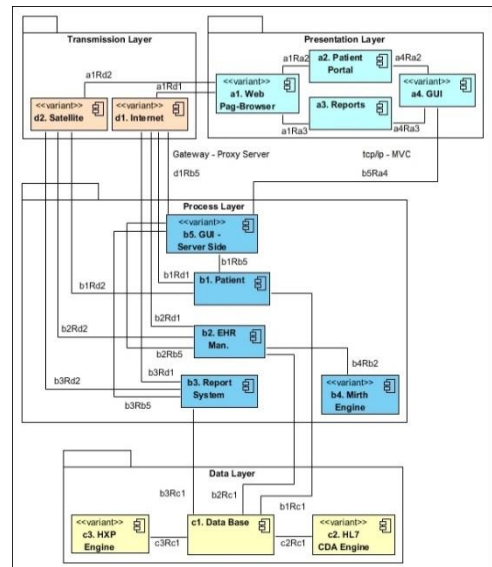


Figure 4. CA for the EHR-HIS domain [12]

The architectures of three open-source systems OpenEMR, PatientOS and Care2X were considered. Semantic similarity of components was studied in each product, on the bases of the available documentation, to determine the common core of functionality. Table 1 presents the architectures' components after the unification of component names, respecting the semantics of the functionalities as far as possible. Let us denote by  $(P_i, R_i)$ ,  $i=1, 2, 3$  valid (connected) architectural configurations of the three products considered, where  $P_i$  is the set of components or nodes and  $R_i$  is the set of edges or

connectors. Let  $CA = (P, R)$  be the initial architecture obtained automatically, where  $P = \cup P_i$  and  $R = \cup R_i, i = 1, 2, 3$ . An edge  $aRb \in R$  is variant, and will be denoted by  $\underline{aRb}$  if and only if  $\exists i, j$  such that  $a, b \in P_i \cap P_j$  and  $\exists a R_i b$  and  $\neg \exists a R_j b$ . The fact that  $a, b \in P_i$  and  $\neg \exists a R_i b$  for some  $i$  (no connector between an and b), was signalled in Table 1 by "X". Common and variant components and connectors are also shown; the "-" symbol indicates the absence of a specific connector, since one of its component is absent.

**Table 1.** CCT: Components and Connectors of each product's architectural configuration [12].

OpenEMR	PatientOS	Care2x
a. Presentation Layer	a. Present. Layer	a. Present. Layer
Components: a1. Web pages-Browser a2. Patient Portal a3. Reports -	Components: -	Components: a1
Connectors: a1Ra2 a1Ra3 a1Rd1 a1Rd2 - - -	a2 a3 a4. GUI Connectors: -	a2 a3 Connectors: a1Ra2 a1Ra3 a1Rd1 a1Rd2 -
b. Process Layer	b. Business logic	b. Process Layer

Components:	Components:	Components:
b1. Patient b2. EHR Management b3. Report System -	b1. Patient business model b2 b3 b4. Mirth (HL7 engine) b5. GUI-Server Side -	b1 b2 b3 -
Connectors: b1Rc1 b1Rd1 b1Rd2 -	Connectors: b1Rc1 X -	Connectors: b1Rc1 b1Rd1 b1Rd2 -
b2Rc1 b2Rd1 b2Rd2 -	b1Rb5 b2Rc1 X -	b2Rc1 b2Rd1 b2Rd2 -
b3Rc1 b3Rd1 b3Rd2 -	b2Rb4 b2Rb5 b3Rc1 X -	b3Rc1 b3Rd1 b3Rd2 -
-	b3Rb5 b5Rd1 -	-
c. Data Layer	c. Data Layer	c. Data Layer
Components: c1. Data Base c2. HL7 CDA Engine Connectors: c1Rc2 -	Components: c1 -	Components: c1 c3. HXP Engine Connectors: -
d. Transmission Layer	d. Transm. Layer	d. Transm. Layer
Components: d1. Internet d2. Satellite	Components: d1 -	Components: d1 d2

**Table 2.** EQM: Overview of CA architectural configuration

Common Components (CC) & <<vp>>	Variant Component	Component description	Component Quality Properties – Priority is shown in (); 1 high, 2 medium, 3 low;		Constraints
			Require (s)	Provide (s)	
	<b>a1-WebPage</b> (Browser)	- Web Pages User Interface (UI) on a Browser; it contains buttons a2 and a3 to access EHR-HIS functionalities	- Authenticity (1), - Confidentiality (1), - Integrity (1), - AvailabilityPers (1), - TimeBehavior (2), - Adaptability (2), - Capacity-Scalability (2), - Interoperability (1),  - Usability	- d1 and/or d2	<b>a1, a4</b> cannot be both present in an architectural configuration; one of them is <b>mandatory CC FR</b> . TimeBehavior, authenticity, confidentiality, integrity, are solved by internet protocols (HTTP, HTTPS) by d1 or d2; Availability and Capacity-Scalability depend on network connectivity. Adaptability depends on Web services via d1 or d2. Capacity-Scalability depends on the network capability to convey data. Availability and TimeBehaviour for a4 depend on the b5 Server Side, responsible for communications. Usability does not depend on the architecture or style, it has not been considered in this context and it is not part of the HIS-DQM. Completeness is assured by construction.
	<b>a4-GUI</b> (stand alone)	- Stand-alone UI; it contains buttons (or menus) a2 and a3 to access EHR-HIS functionalities	- Authenticity, - Confidentiality, - Integrity,  - AvailabilityPers, - TimeBehavior  - Modifiability (3), - Modularity (3),  - Usability	- d1 and/or d2  - b5 via TCP/IP, proxy, or gateway in Process Layer  - b5 via the MVC pattern in Process Layer  - not considered	
	<b>a2-Patient Portal</b>	- UI button to access EHR-HIS common functionalities b1, b2 in Process Layer	- the above quality properties are also required by a2, a3	- the above components provide also quality to a2, a3	<b>a2, a3 are mandatory CC FR</b> ; they are UI push buttons, they are actually part of a1 and a4.
	<b>a3-Reports</b>	- UI button to access b3 functionality			
	<b>b1-Patient</b>	- services for patient admission to a healthcare institution, on a scheduled appointment, attended by authorized medical staff: retrieval, storage, modification, deletion of patient personal and demographic data, laboratory/imaging results, and appointment scheduling services	- CorrectPrecision (2), - AvailabilityPers, - Authenticity, - Confidentiality, - Integrity  - Adaptability - Capacity-Scalability	- components in Process Layer - d1 or d2 or by components in Process and/or Data Layer  - components in Data Layer	<b>b1, b2, b3 are mandatory CC FR</b> - if EHR services are implemented as web services, Adaptability is solved; else Adaptability and Capacity-Scalability can be solved by c1 in Data Layer

<b>b2-EHRMang</b>	- basic EHR management services: HER retrieval, creation, storage, modification, sharing	- Interoperability,  - AvailabilityPers, - Authenticity, - Confidentiality, - Integrity	- b4 in Process Layer and/or c2 or c3 in Data Layer  - d1 or d2 or by components in Process and/or Data layer	- <b>Interoperability is mandatory HIS NFR</b> solved by b4 in Process Layer; c2 or c3 are optional solutions for Interoperability in Data Layer;
<b>b3-Report System</b>	- services for emission, retrieval, modification, and storage of documents related with medical practice (reports, orders, billing, invoices)	- Adaptability, - Capacity-Scalability  - Authenticity, - Confidentiality, - Integrity, - AvailabilityPers,  - CorrectPrecision	- components in Data Layer  - d1 or d2 or by components in Process and/or Data layer  - components in Process Layer	
<b>b4-MirthEng</b>	- translation services for EHR to HL7 standard format	- Interoperability	- market engines	<b>b4 is mandatory to solve interoperability HIS NFR</b>
<b>b5-GUI ServerSide</b>	- deserves local and remote client/server communication	- TimeBehavior  - Modifiability - Modularity	- protocols  - modules	- TCP/IP, proxy or gateway  - MVC pattern
<b>c1-DB</b>	- usual services provided by DBMS (Database Management Systems)	- Capacity-Scalability - Integrity - AvailabilityPers  - Adaptability - Interoperability	- modules, mechanisms  - market APIs - c2, c3	- <b>c1 is mandatory CC FR;</b>
<b>c2-HL7Eng</b>	- translation services to standard formats for EHR documents customization	- Interoperability	- modules, mechanisms or engines	- c2, c3 are optional; at least b4 must be present in an architectural configuration
<b>c3-HXPeng</b>	- wide area wired/wireless communication services	- TimeBehavior, - AvailabilityPers, - Adaptability, - Capacity-scalability - Authenticity, - Confidentiality, - Integrity	- hardware /software (protocols, services, etc.) wired/wireless network infrastructure	<b>d1 is mandatory CC FR</b> , d2 is optional FR; they can both be present in an architectural configuration. Adaptability depends on Web services; Availability depends on network connectivity; Capacity-Scalability depends on the network capability to convey data; HTTP/HTTPS protocols solve limited security; HTTP alone is not possible due to HIS security requirements; these protocols are not considered here separately; they are used in Process Layer to increase security levels.
<b>d1-Internet</b>				
<b>d2-Satellite</b>	- mobile communication services	- The above properties hold	- The above components hold	

### Step 1.2 Elaborate marketing study for SPL feasibility

The Marketing Study will not be explicitly performed in this work. However, the majority of the twelve systems found on the market in [16], were centred on the EHR management as main functionality, and show clear limitations, namely, they cannot be customized, poor front-ends, difficult to be installed, not good enough technology to support full telemedicine capabilities, etc. All of them also considered the physical doctor-patient encounter episode. For this research, four open-source systems currently on the market, discussed in [16], were extensively studied to produce the initial CA.

According to the study in [24], OpenEMR [19] and PatientOS [25] presented a 90% and 92% usage respectively, according to 2010 data. OpenEMR is an Electronic Medical Record (EMR) management Web-based system, offering usual EHR-HIS services for geographically distributed clients and different platforms, assuring portability and interoperability through Internet, supporting also mobile devices networks. PatientOS is the front-end for OpenEHR, following a client/server classic distributed model of 3-layered information systems; it holds an MVC-based Graphical User Interface (GUI) to

ensure maintainability and it retrieves remote services from a Web Server through the Process Layer. Other open-source systems used in recent concrete national projects were Care2x [26], very similar to OpenEMR and GNU-Health, similar to PatientOS; GNU-Health was not included into our CA because of poor architectural documentation.

In conclusion, there are no mature open-source healthcare systems available on the market, and even less HIS SPL. In consequence, the construction of a HIS SPL, is a feasible and interesting goal at research level. In industrial practice, producing adaptable HIS systems could be also an interesting challenge, doing appropriate costs-benefits studies.

### Step 1.3 Outline commonality and variability for future SPL products

Table CCT (see Table 1) has shown commonality and variability of the three existing products studied, namely OpenEMR, PatientOS, and Care2x, and Table EQM (see Table 2) has described basic EHR-HIS functionalities integrated into our CA, produced by the union of the components of the above products, with previous analysis of similarities among components.

However, new features, not considered among CA basic functionalities, can be added to conform future EHR-HIS products for the SPL to be constructed:

- An imaging system (LabImage System) to handle laboratory results and radiological images studies, linking them to the patient EHR, as needed, stored with appropriate formats to be shared by local or distant medical staff and institutions, and fast retrieval for visualization.
- A diagnostic assistant (Diagnosis) to help doctors to perform on-line diagnosis: retrieval of medical catalogues for diseases identification, protocols to be applied to particular disease, etc.
- An on-line appointment services facility to allow patients to schedule on-line their appointments, select healthcare institutions and particular specialists; on-line symptomatology consult (SymptomAssistant) to check symptomatology to select a particular specialist; retrieve, modify, remove, and store on-line personal and demographic data on patient EHR.

Possible commonality and variability for new SPL products including the above features will be outlined in next Domain Scoping phase.

### 3.5 Application to Domain Scoping

Following the steps defined in Domain Scoping, we have:

#### Step 2.1 Stakeholders Identification

Information was elicited from visits, interviews, workshops or questionnaires in different institutions related to the stakeholder's groups of interest: healthcare institutions, governmental institutions related with healthcare and software developers [21]. Different viewpoints were identified to conform the stakeholders' groups: Doctors, Domain Engineers and Directors of Healthcare governmental institutions.

#### Step 2.2 Domain acquisition

Statements or Declarations formulated by different stakeholders to illustrate their viewpoints about the EHR-HIS domain were captured. Recall that the HIS domain scope considered is restricted to the episode of doctor-patient encounters under scheduled appointments, which will be briefly called "Appointment Services" [16], and it is focused on the EHR management.

#### Step 2.3 Domain Analysis

Declarations are grouped into Domain Description Unit (DDU) [13] and are represented textually as a table (see Tables 3 A, B, and C), for each identified stakeholders' group.

- DDU specification

**Table 3 A.** DDU for EHR-HIS from the Doctors viewpoint

EHR-HIS Domain Description Unit
Viewpoint Stakeholders' Group: Doctors
Declarations
Patient is attended in hospital under scheduled appointment
If it is the patient first appointment, a new EHR must be created by nurse, else nurse retrieves patient existing EHR
Patient EHR is provided by nurse and accessed by doctor
New exams and laboratory results can be added to patient EHR by doctor, if it is the case

Diagnosis and medical orders/reports/billing for patient are produced by doctor to conclude medical attention
A new appointment is scheduled by nurse if required by Doctor
Medical equipment and material can be required by doctor to provide adequate medical attention

This DDU refers to patient scheduled appointment services involving nurses, patients and doctors present at the healthcare institution, emphasizing the EHR management, according to the scope of this case study.

**Table 3 B.** DDU for EHR-HIS from the Domain Engineers viewpoint

EHR-HIS Domain Description Unit
Viewpoint Stakeholders' Group: Domain Engineers
Declarations
EHR-HIS is supported by a hybrid architecture SOA/Layers
The EHR-HIS architectural style supports modifiability, interoperability, performance (time-behaviour); security services are provided by Internet protocols, crosscutting all layers; availability however depends on the internet connection; security on also be provided or by modules in Process Layer
Transmission Layer is managed by a Web Server that communicates all other layers
Clients access EHR-HIS by a browser in Presentation Layer, which connects to Transmission Layer via a Web Server
Medical units should have wide range internet, intranet and/or satellite access
Main EHR-HIS functionalities must be supported: patient appointment services and demographic data capture, EHR management and emission of medical orders/reports/billing.

**Table 3 C.** DDU for EHR-HIS from the Directors of Healthcare Governmental Institutions (HGI) viewpoint

EHR-HIS Domain Description Unit
Viewpoint Stakeholders' Group: Directors of healthcare governmental institutions (HGI)
Declarations
Digitalize EHR with standard format to achieve sharing among doctors and geographically distant national and international healthcare institutions
Have a database of national and international specialists
Develop a Web platform to manage on-line appointment services

The initial domain knowledge is captured from DDUs in Table 3 A, B, and C, and described textually in Table 4, 6, and 7 by intrinsic, for each facet considered.

- Facet specification

The facets that will be specified in this work are *Business Processes, Support Technology, and Rules&Regulations*. The viewpoints considered for the three facets are: *Doctors' group* for Business Processes, *Domain Engineers' group* for Support Technology, and *Directors of healthcare governmental institutions (HGI)* for Rules&Regulations. Facets, stakeholders' groups, and viewpoints were selected on the bases of the DE development process for SPL, whose main goal is to design a Reference Architecture.

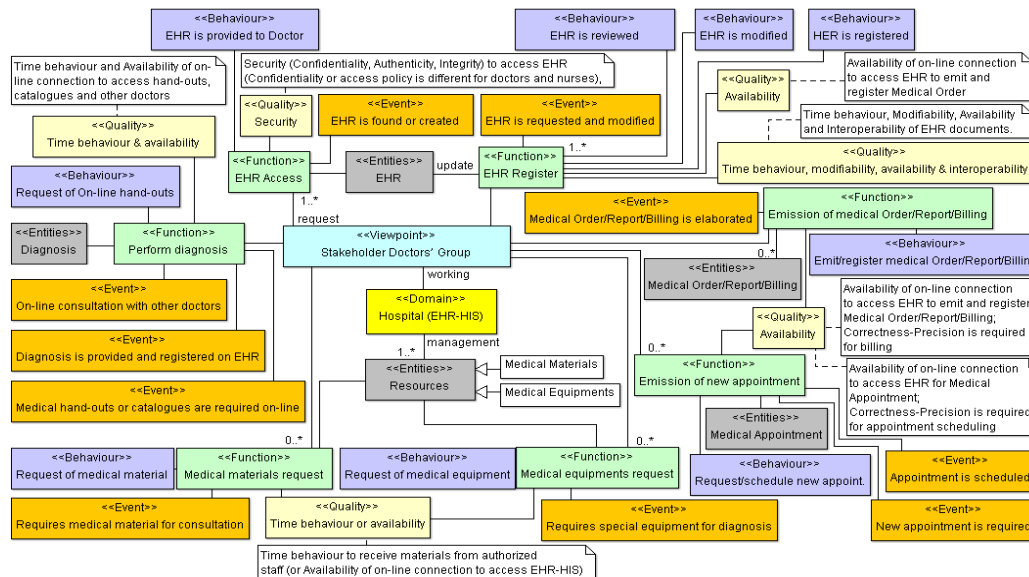
Different notations can be used to specify facets from stakeholders' viewpoints with intrinsic descriptors, each one offering different specification granularity and capabilities [13]; from each specification, more details are extracted; in this work the following notations will be used:

- Tables for informal textual specification by intrinsics, for all facets (see Tables 4, 6, and 7);
- Semi-formal UML [27] diagrams (see Figures 5, 7, and 9) for all facets;
- Semi-formal BPMN [15] diagrams for all business processes (see Figures 6, 8, 10, and 11).

**Table 4.** Specification by intrinsic of Doctors viewpoint for the Appointment Services Business Process Facet

Domain EHR-HIS				
Viewpoint Stakeholders' Group: Doctors				
Appointment Services Business Process Facet				
Entities	Functions	Events	Behaviour	Quality
EHR	EHR Access	EHR is found or created	Nurse confirms the existence of patient EHR or creates an EHR for new patient; nurse provides EHR to doctor (EHR is provided to doctor)	Security (confidentiality, authenticity, integrity) to access EHR; confidentiality or access policy is different for doctors and nurses, authenticity is needed to univocally identify user, and integrity is required for data consistency.
	EHR Register	EHR is requested by doctor; doctor attends patient medically and EHR is modified	EHR is accessed and reviewed by Doctor; he can add laboratory and/or examinations results, if any, to patient EHR; patient is attended medically; EHR is modified (EHR is reviewed, modified, registered)	Availability of on-line connection to access EHR, Time behaviour to quick access to EHR, Modifiability to change EHR, Availability-Persistency to retrieve always EHR, and Interoperability for EHR sharing.
Diagnosis	Perform diagnosis	Medical hand-outs or catalogues are required on-line; on-line consultation with other doctors in different healthcare institutions or locally to perform diagnosis; diagnosis is provided and registered on EHR	Doctor requests on-line hand-outs or catalogues; he can consult on-line with other doctors in different healthcare institutions or local doctors, and patient's EHR must be shared by other doctors (ideal situation, since psychosocial factors often avoid the achievement of this activity); review laboratory and/or examinations results; doctor produces diagnosis (Request of on-line hand-outs)	Time behaviour and Availability of on-line connection to have quick access to hand-outs, catalogues and other doctors
Resources: Medical Materials (*)	Medical request	Doctor requires medical material for consultation	Doctor requests medical materials calling authorized staff (or using the EHR-HIS system if this facility is available) (Request of medical material)	Physical Availability of materials, Time behaviour to receive materials from authorized staff (or Availability of on-line connection to access the EHR-HIS system for materials request, if this facility is supported)
Resources: Medical Equipment (*)	Medical request	Doctor requires special equipment for diagnosis	Doctor requests medical equipment calling authorized staff (or using the EHR-HIS system if this facility is available) (Request of medical equipment)	Physical Availability of equipment, Time behaviour to receive equipment from authorized staff (or Availability of on-line connection to access the HER-HIS system for equipment request if this facility is supported),
Medical Appointment	Emission of new appointment	New appointment is required for patient, if necessary; appointment is scheduled	Doctor registers patient for a new appointment; the appointment is scheduled. (Request/schedule new appoint.)	Availability of on-line connection to access EHR for Medical Appointment; correctness-precision is required for appointment scheduling
Medical Order/Report/Billing	Emission of medical Order/Report/Billing	Medical Order/Report/Billing is elaborated for present consultation	Doctor emits medical Order/Report/Billing (Emit/register medical Order/Report/Billing)	Availability of on-line connection to access EHR to emit and register Medical Order/Report/Billing; correctness-precision is required for billing

Notes: (\*) These services will not be considered for EHR-HIS in this work; the abridged behaviour name is specified within () in italics
















**Figure 5.** Doctors viewpoint of the Business Process Facet for Appointment Services (UML)

**Table 5.** BPMN notation for the Business Process specification [15]

BPMN Notation	Symbol	Description	Interpretation (Figures 5, 7, and 9)
	Task	Simple or atomic activity representing the work in an organization. They consume resources; it is not detailed further.	They can be mapped into SPL RA architectural components or services.
	Exclusive gate	Control element of the data workflow. A unique path is selected among several alternatives	It represents the SPL variability.



	Inclusive gate	Control element of the data workflow. One or more path (s) can be selected from several alternatives	It represents the SPL variability.
	Starting event	It initiates a process	It represents the starting or entry point of a business process.
	Starting message event	A process initiates when a message is received	It represents the events arising from an active business process.
	Ending event	It indicates the end of a workflow	It represents the end of some behaviour in the business process.
	Terminal ending event	It indicates that a process ends, even if there are active workflows	It represents the end of a business process.
	Intermediate link event	It allows the connection of two process sections.	It allows a connection between two sections of the business process.
	Swimlanes (Pool)	It is a process container; it represents participant, entity or role.	It represents the stakeholders' viewpoints.
	Swimlanes (Lane)	They are pool's subdivisions; it represents participants within an organization.	It represents the different behaviours of the business process.
	Connector object: Sequence	It represents the workflow and the sequence of activities.	It represents the execution flow of the activities in a business process.
	Connector object: Message	It represents interactions among processes or pools.	It represents changes of stakeholders' viewpoints.
	Associations	They are used to relate additional information on the process.	They relate entities, activities or functions with required quality properties.
	Artefact: data object	They provide additional information on the process; they show the information required by an activity, such as input/output.	Document specifying the quality required by entities, activities or functions, in each behaviour.
	Aggregation of activities included in a process	They allow to group activities of a process (sub-process)	It represents a functionality, system or subsystem; it can be mapped into a component or service of the SPL RA.
	Business Rule	Allow to specify business rules	It represents the quality required by the Aggregation, "synthetizing" those that were attached as comments.

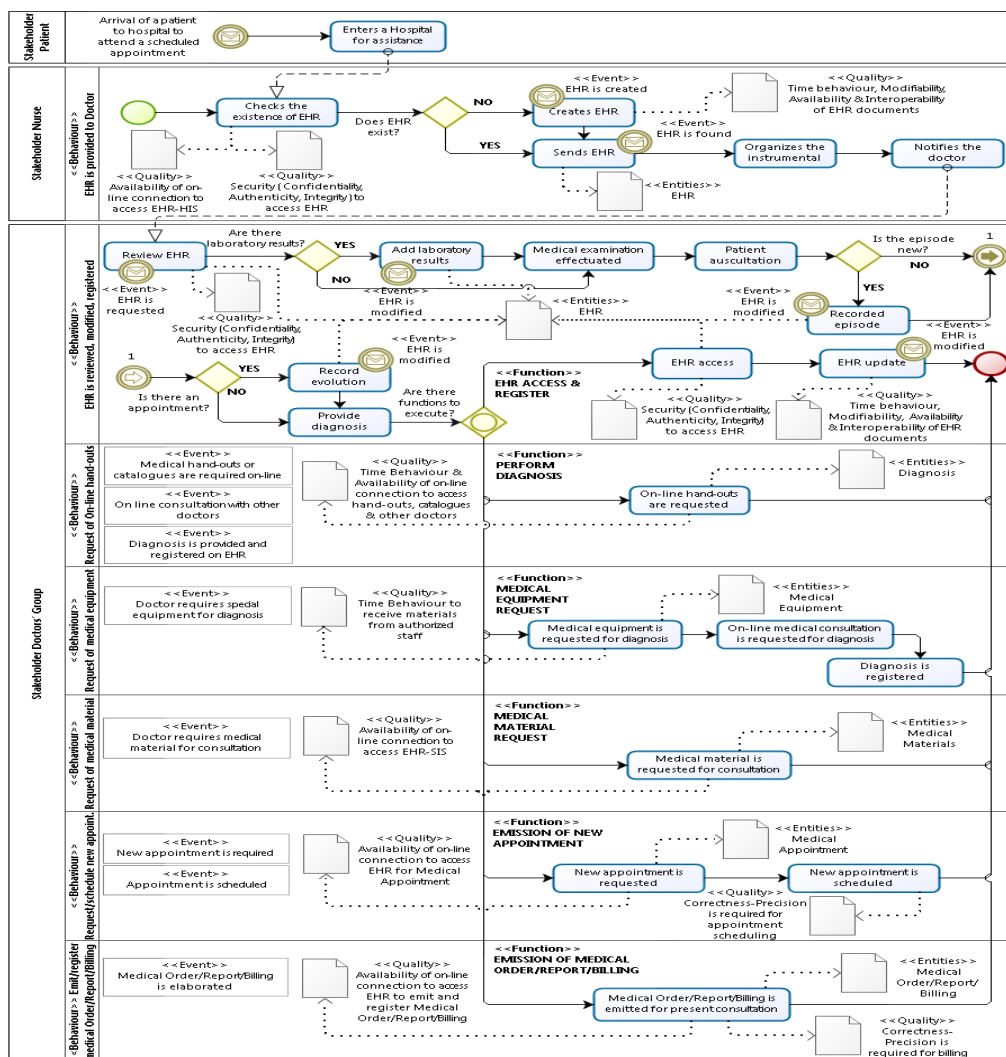


Figure 6: Business processes extracted from behaviors of Doctors viewpoint for the Business Process Facet (BPMN)



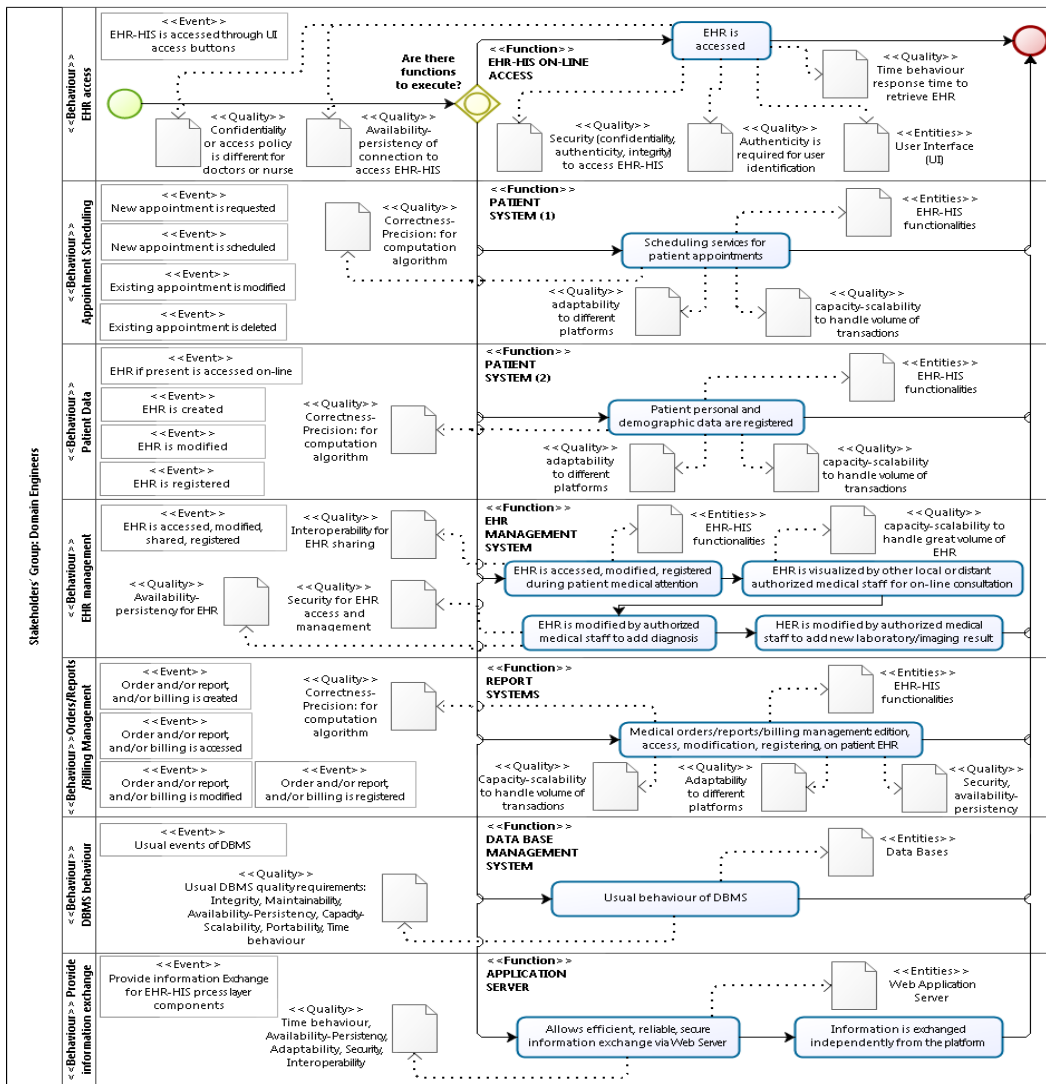


Figure 8. Business processes extracted from the Domain Engineers viewpoint for the Support Technology Facet (BPMN)

Table 7. Specification by intrinsics of Directors of HGI viewpoint for the Rules & Regulations Facet

Domain EHR-HIS				
Viewpoint Stakeholders' Group: Directors of Healthcare Governmental Institutions (HGI)				
Rules & Regulations Facet				
Entities	Functions	Events	Behaviour	Quality
EHR	EHR sharing	EHR is digitalized using some standard format	EHR records are shared (EHR sharing)	- Interoperability for EHR sharing, Security for EHR access and management, Availability-Persistence for EHR, they cannot be destroyed and they must be always available
- Specialists Data Base (SpDB), - SpInfo: Specialist Information	SpDB Management System (SpDBMS)	Provide all SpDBMS services; retrieve, register, modify, delete specialist data. SpInfo is retrieved by patient or authorized medical staff; it is registered, modified, deleted by authorized medical staff	Usual DBMS (Usual DBMS)	Usual DBMS quality requirements: Integrity, Maintainability, Availability-Persistence, Capacity-Scalability, Adaptability, Time behaviour
Symptom	On-line Symptoms Assistant	- patient list symptoms - symptoms are checked - specialist is recommended	- symptoms are listed on-line by patient and automatically checked in on-line medical catalogues - specialist is recommended on-line to patient if symptoms are found, else patient must rewrite on-line a new list of symptoms (Symptoms Management)	- Time behaviour and - Availability of on-line connection for fast responses - Usability for friendly user interface
Appointment	On-line Appointment Manager	- New appointment is requested by patient with chosen specialist - New appointment is scheduled or kept in a waiting list - is reported to patient - Existing appointment is modified by patient	- Provide precise appointment services with specialist; handle volume of requests, provide secure access to appointment services; have network connection facility; have a friendly user interface - Appointment with specialist is requested, scheduled, reported, deleted or modified on-line (On-line Appointment Management)	- Capacity-Scalability to manage volume of transactions, Availability-Persistence of on-line connection, Security, Correctness-Precision for scheduling, usability to have a friendly user interface

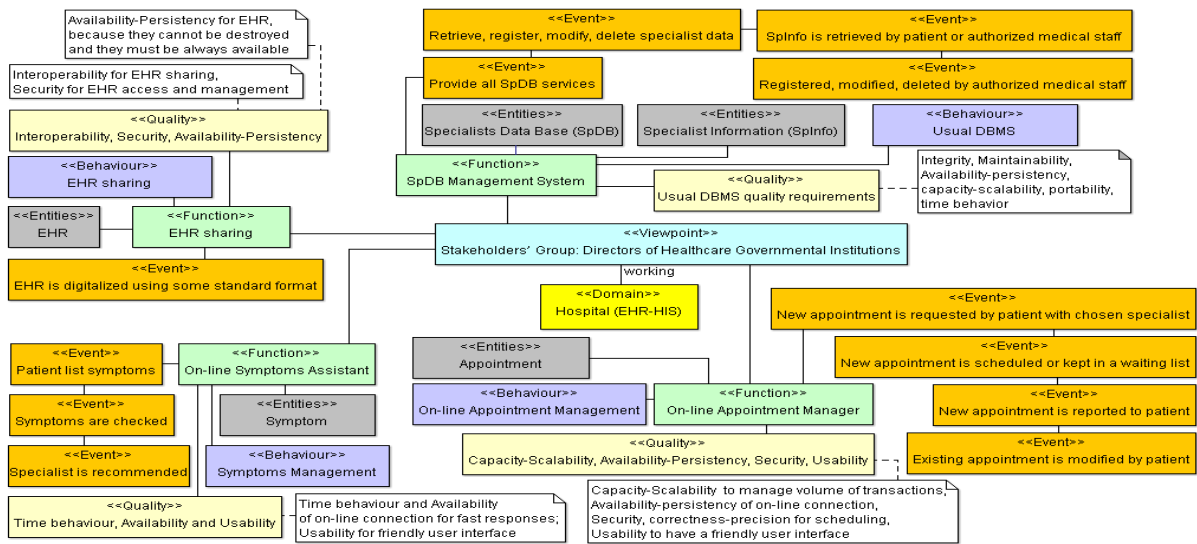


Figure 9. Directors of HGI viewpoint for the Rules & Regulations Facet (UML)

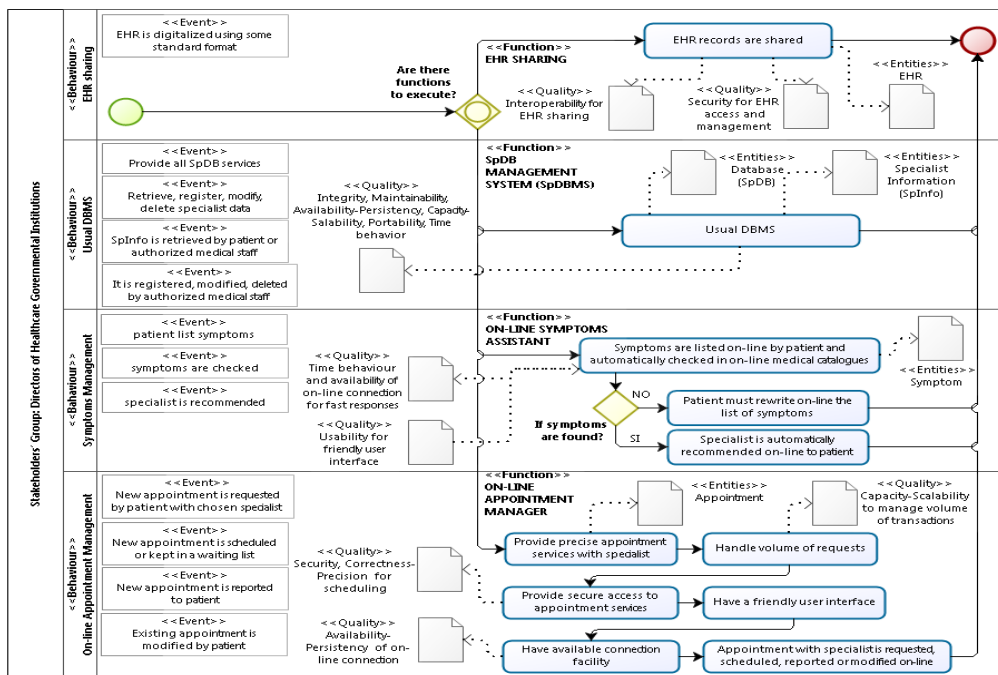


Figure 10. Business processes extracted from the Directors of HGI viewpoint for the Rules&Regulations Facet (BPMN)

### Step 2.4 Domain Modelling

The BPMN representation of the Domain Model of Figure 11 shows Aggregations of business processes identified for the three different stakeholder groups viewpoints for Business Processes, Support Technology, and Rules & Regulations facets. This information contains the potential new components to be included into CA; the updated EQM (UEQM) in Table 8, specifies these new components, selected by the domain engineer or domain expert, that will be added to CA, according to the considerations on the new products stated in step 1.3 in the Future Products Document.

Completed CA (UCA), see Figure 12, and EQM, see Table 2, are used to construct UEQM, see Table 8, where the variation points grouping new variants are also shown; to abridge the presentation; actually they will be conformed in step 3.7 of DRE. The names used in UEQM for each component, reflect semantic functional similarities found in the Aggregations of the processes studied for each

stakeholder viewpoint; recall also that non-automatic processes that do not involve interaction with digital entities are not considered, e.g. the patient entering hospital.

Modifications of EQM are:

- CC *b1-Patient* is now dedicated only to patient attention, including sub-components:
  - *b-18-AppointService*, found in all viewpoints examined;
  - *b19-SymptomAssistant* found only in the Directors of HGI viewpoint to get on-line appointment services; however, this facility can also be used by doctors for diagnosis;
  - *b20-DemographicData* also mentioned in all viewpoints;
- CC *b2-Medical Practice* now includes functionalities directly related to medical practice, with new sub-components:
  - *b12-EHRManager* for HER management, renaming former *b2-EHR Mang*;

- *b13-Diagnosis* for diagnosis assistance;
- *b14-Orders&Reports* for medical prescriptions and reports management;
- CC *b3-AdminSystem* renames former *b3-Reports* including only administrative services;

- CC *b15-LabImageSystem* is a new functionality for laboratory and imaging results management;
- CC *a6-Lab&Imaging* is a new button added to access new functionality *b15-LabImageSystem*.

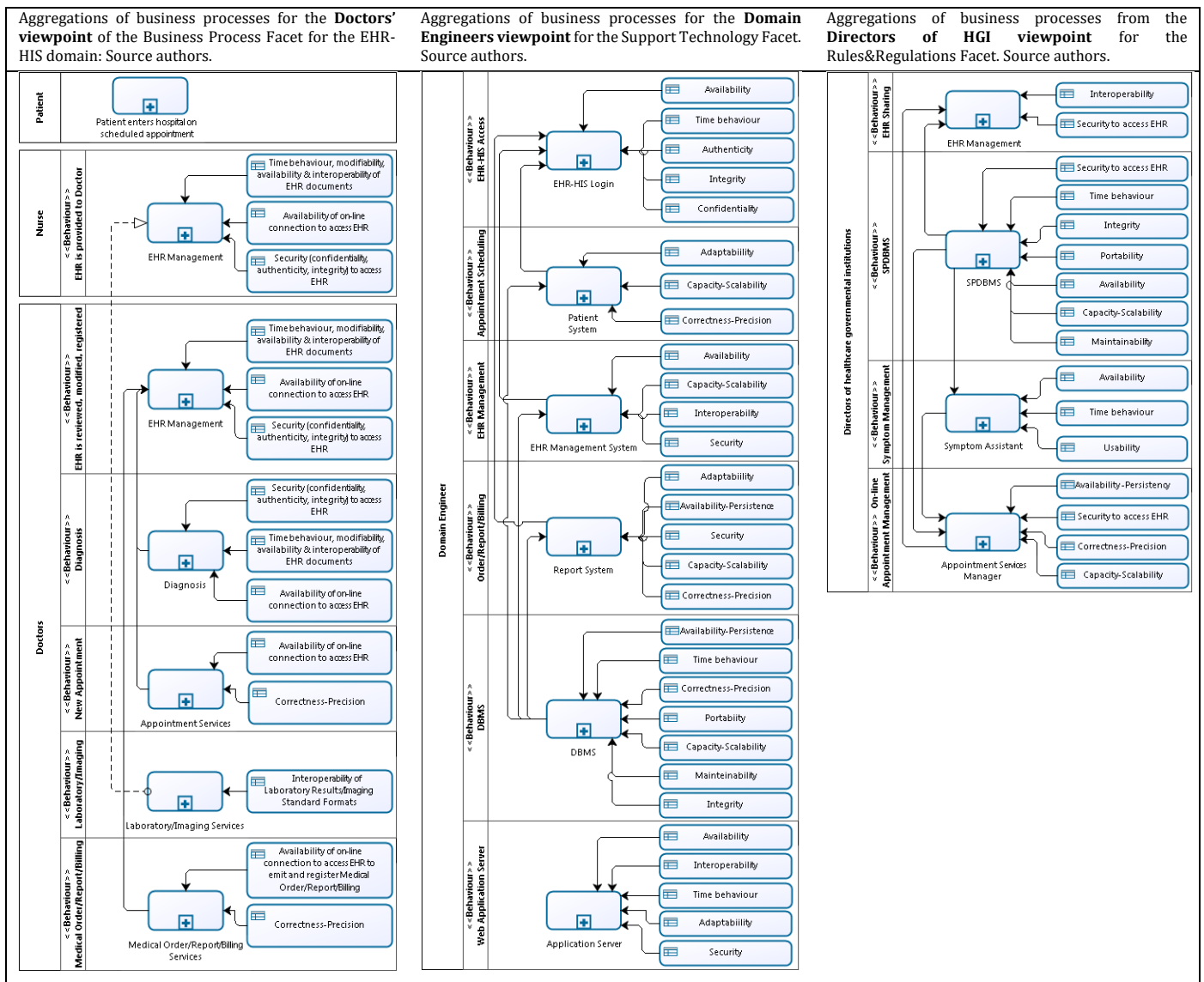


Figure 11. Aggregation of business processes for the facets studied (BPMNAgg)

Completed CA (UCA), see Figure 12, and EQM, see Table 2, are used to construct UEQM, see Table 8, where the variation points grouping new variants are also shown; to abridge the presentation; actually they will be conformed in step 3.7 of DRE. The names used in UEQM for each component, reflect semantic functional similarities found in the Aggregations of the processes studied for each stakeholder viewpoint; recall also that non-automatic processes that do not involve interaction with digital entities are not considered, e.g. the patient entering hospital.

Modifications of EQM are:

- CC *b1-Patient* is now dedicated only to patient attention, including sub-components:
  - *b18-AppointService*, found in all viewpoints examined;
  - *b19-SymptomAssistant* found only in the Directors of HGI viewpoint to get on-line appointment services;

- however, this facility can also be used by doctors for diagnosis;
- *b20-DemographicData* also mentioned in all viewpoints;
- CC *b2-Medical Practice* now includes functionalities directly related to medical practice, with new sub-components:
  - *b12-EHRManager* for HER management, renaming former *b2-EHR Mang*;
  - *b13-Diagnosis* for diagnosis assistance;
  - *b14-Orders&Reports* for medical prescriptions and reports management;
- CC *b3-AdminSystem* renames former *b3-Reports* including only administrative services;
- CC *b15-LabImageSystem* is a new functionality for laboratory and imaging results management;
- CC *a6-Lab&Imaging* is a new button added to access new functionality *b15-LabImageSystem*.

**Table 8.** Updated EQM (UEQM) Table: Overview of the completed CA (UCA); changes are outlined in grey

Common Components (CC) & Variation Points <<vp>>	Variant Component	Component description	Component Quality Properties – Priority is shown in (); 1 high, 2 medium, 3 low;		Constraints
			Require (s)	Provide (s)	
<<a5-UI>>	a1. <i>WebPage</i>	- Web Pages User Interface (UI) on a Browser	see Table 2	see Table 2	see Table 2
	a4. <i>GUI</i>	- Stand-alone UI	see Table 2	see Table 2	see Table 2
a6. <i>Lab&amp;Imaging</i>		- UI button to access b15 functionality	- the same quality properties specified in Table 2 for a2, a3 hold	- the same components specified in Table 2 for a2, a3 hold	a6 is mandatory CC FR; it is part of a1 and a4.
<b>b1-Patient</b>					
- b18- <i>AppointmentService</i>		- on-site and/or on-line appointment scheduling	- CorrectPrecision - AvailabilityPers, - Authenticity, - Confidentiality, - Integrity - Adaptability - Capacity- Scalability	- b6  - c7, c9a, c9b - b7 - c8 - c5, c6 - c4	b1 is mandatory CC FR; b18, b19 and b20 are sub-components of b1 and they are also mandatory CC  c8 is a mechanism for additional integrity; c5, c6 are alternatives for portability of commercial data bases to different platforms
- b19- <i>SymptomAssistant</i>		- assistance to symptomatology search to find medical specialist	The above properties hold for b19, b20 sub-components	The above components hold for b19, b20 sub-components	
- b20- <i>DemographicData</i>		- retrieval, storage, modification, deletion of patient personal and demographic data, laboratory/imaging			
<b>b2-MedicalPractice</b>					
- b12- <i>EHRManager</i>		- retrieval, creation, storage, modification, sharing of EHR	- Interoperability,  - AvailabilityPers, - Authenticity, - Confidentiality, - Integrity	- b4 and/or c2 or c3  - c7, c9a, c9b - b7-HTTP, b7-HTTPS, HTTPS - c8	b2 is mandatory CC FR; b12, b13 and b14 are sub-components of b2 and they are also mandatory CC - Interoperability is mandatory HIS NFR solved by b4; c2 or c3 are optional solutions for Interoperability in Data Layer;
- b13- <i>Diagnosis</i>		- retrieval of medical catalogues, protocols; symptomatology search	- AvailabilityPers, - Authenticity, - Confidentiality, - Integrity - Time behaviour	- c7, c9a, c9b - b7-HTTP, b7-HTTPS, HTTPS - c8 - protocols in Transmiss. Layer	
- b14- <i>Orders&amp;Reports</i>		- services for emission, retrieval, modification, and storage of medical orders and reports	- Authenticity, - Confidentiality, - Integrity, - AvailabilityPers, - Capacity- Scalability,	- b7-HTTP, b7-HTTPS, HTTPS - c8 - c7, c9a, c9b - c4	
<b>b3-AdminSystem</b>					
		- basic administrative services: invoices, billing, statistical reports	- Authenticity, - Confidentiality, - Integrity, - AvailabilityPers, - CorrectPrecision	- b7-HTTP, b7-HTTPS, HTTPS - c8 - c7, c9a, c9b - b6	b3 is mandatory CC FR;
<b>b-15 LabImageSystem</b>					
		- retrieval, storage, modification of laboratory and imaging radio graphical studies	- Interoperability - AvailabilityPers, - Authenticity, - Confidentiality, - Integrity, - CorrectPrecision	- b16, b17 - c7, c9a, c9b - b7-HTTP, b7-HTTPS, HTTPS - c8 - b6	b15 is mandatory CC FR
<<b21>> <i>LabIm Interoperability</i>	<b>b16-LOINC</b>	- database including a standard for identifying medical laboratory observations	- Interoperability	- <i>database (repository)</i>	Mandatory NFR for b15; b16, b17; both must be present in a configuration
	<b>b17-DICOM</b>	- mechanism for conversion to standard imaging graphic formats; conformed by a format file and a common. protocol; files are exchanged via TCP/IP	- Interoperability	- <i>Market translation engine</i>	
<<b8>> <i>CompMod</i>	<b>b6-Algor* (*) multiplicity</b>	- Algorithms for computations	- CorrectPrecision	- <i>modules</i>	Mandatory NFR for b1, b3 and b15
<<b9>> <i>SecurityMod</i>	<b>b7.Data Security</b>	Services or algorithms for user identification	- Authenticity, - Confidentiality,	- b7-HTTP - HTTPS	- Mandatory HIS NFR; alternatives combining

		(Authenticity) and policy of user access rights (Confidentiality) of medical information	- Integrity	- b7-HTTPS	security algorithms in Process Layer b7 with security protocols HTTP/HTTPS in Transmission Layer; integrity is also treated in Data layer; only one alternative holds
<<b10>> HL7InteropEng	b4. Mirth	- translation services for EHR to HL7 standard format	- Interoperability	- market translation engine	b4 is mandatory to solve interoperability HIS NFR
<<b11>> GUIServer	b5. GUI Server Side	- deserves local and remote client/server communication	- TimeBehavior - Modifiability - Modularity	- protocols - modules	- TCP/IP, proxy or gateway - MVC pattern
c1-DB - c16-EHRDB - c17-LABIMDB - c18-SPDB - c19-CATDB		- database services via usual DBMSs	The same as specified in Table 2 for c1-DB	The same as specified in Table 2 for c1-DB	c1 is mandatory CC FR, then all sub-components are also mandatory CC; in this study only c1 is considered
<<c10>> AMedStds	c4-NewMed Stds *	- addition of new standards for medical practice: formats, catalogues, medical protocols, etc.	- Capacity-Scalability	- modules	Optional NFR
<<c11>> Integrity	c8-IntegMech *	- provide data integrity to databases	- Integrity (c8)	- mechanisms	Mandatory NFR for c1-DB
<<c12>> DataAvail	c9a-Mirror c9b. MirrorReplic	- provide backups for databases	- AvailabilityPers	- mechanisms	Optional NFR for c1-DB; only one alternative holds
<<c13>> DataPers	c7-Hibernate	- provide data persistency to databases	- AvailabilityPers	- engine	Mandatory NFR for c1-DB
<<c14>> DBAPIs	c5-JDBC c6-ODBC	- Portability to different platforms for commercial databases	- Adaptability	- APIs	Optional NFR for c1-DB
<<c15>> HL7 DataModel Engine	c2-HL7Eng c3-HXPeng	- Interoperability		- engines	Optional NFR for c1-DB; at least b4 must be present

Note that c1-DB in initial CA included all databases; in the completed CA main databases used are shown as sub-components, but they will not be treated separately. Semantics of new components is provided in Table 8. In step 3.6 Application to the Asset Scoping, the Core Asset Repository (CAR) is conformed by DQM, CCT, CA, UCA, EQM, UEQM, BPMNs, and BPMNAggs, which are the basic outputs of phases I and II; however, the CAR structure has to be properly designed. Outputs for phases 3.7 Application to the Domain Requirements Engineering (DRE) and 3.8 Application to the Domain Design (DD) are UCA and RA, shown in Figures 12 and 13, respectively. RA is also added to CAR.

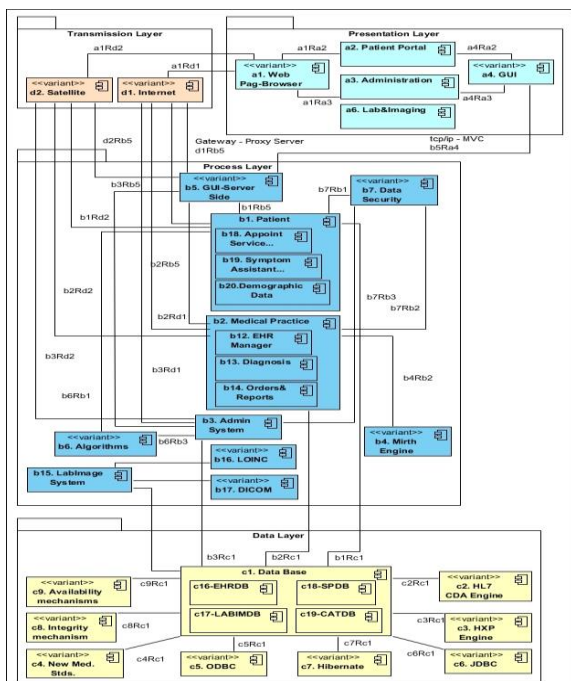


Figure 12. Completed CA (UCA)

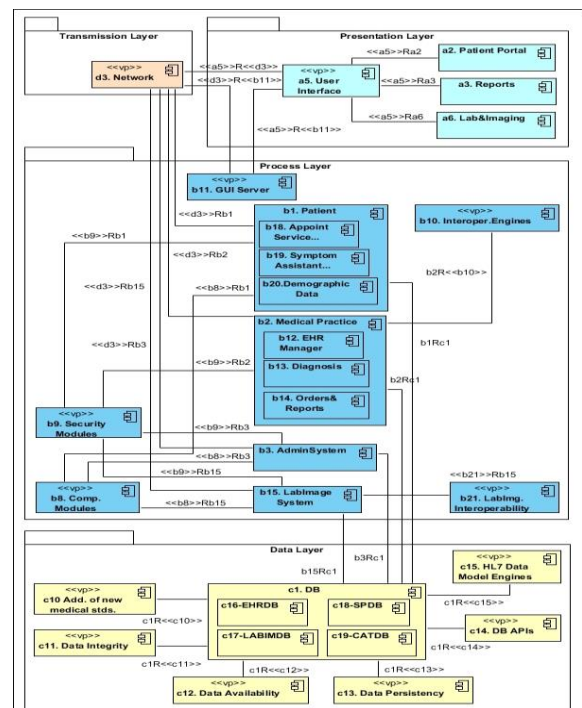


Figure 13. RA

#### 4. Related works

A proactive process for DE is proposed in [28] for Service-Oriented Applications domain; it includes business modelling and enhances NFR traceability; it is based on a set of UML conceptual models and profiles [15, 27] and it uses Bjørner's domain modelling approach [13], since it starts from specifying business processes behaviours to capture domain knowledge; early quality requirements issues are also considered. Our Domain Scoping sub-phase supports traceability from quality required by functions or activities involved in business processes, to functional RA components.

The SPL Reference Model framework is discussed in [4, 5, 11]; it provides guidelines on “*What to do*”, but not on “*How to do*”, even if it presents lists of existing techniques that can be applied, how these should be tailored or adapted is not specified. Our work complements the framework, offering techniques on how to perform specific activities in Domain Scoping, including quality requirements specification [29] in Bjørner’s domain modelling [13], issue not considered early in these works.

A systematic literature review [30] on best practices, challenges and limitations of the SPL Scoping phase points out that it is essential for SPL success. It is claimed that the majority of the approaches reviewed do not have a clear understanding of the relation between scoping and DRE phases. In this sense, our proposition to use Bjørner’s domain modelling will fill this gap.

A framework is described in [31] for SPL developments including the SPL scoping phase; a specific method based on PuLSE [32], Pulse-Eco, is presented. A bottom-up process is used. We are focusing on a top-down approach in the spirit of SPLE; however, the study of existing products is considered for the Portfolio Scoping, [11]; in this step we reuse an existing candidate architecture automatically built by a bottom-up approach [12], combined with the top-down approach in Domain Scoping, thus reducing considerably the effort in the DRE and DD phases.

## 5. Conclusion

The basic idea of our approach, which is not new in architectural design for single systems, is to consider RA main functionalities, requiring quality properties to be functionally suitable, and including variant components providing potential “implementations” to satisfy this quality by available market components. This is facilitated by the bottom-up process proposed to automatically construct the initial CA architecture, respecting the domain architectural style properties and the core functionality. To bridge the gap found in ISO/IEC 26550 about the late treatment of quality issues, quality is introduced early as an intrinsic descriptor into facet specifications in Domain Scoping, and then mapped into business processes, which are agglomerated to conform new architectural components for the future SPL products. The huge effort required in DRE and DD phases is thus reduced by the combination of bottom-up and top-down approaches, being this a major contribution. It is clear that the domain engineer, architect, or expert play a major role in the EQM and UEQM specifications (see Tables 2 and 8); it is clear that a deep knowledge on available technology will result in a more flexible and evolutionary RA. The recommended RA ontological representation is based on EQM-UEQM, where main domain knowledge is gathered. The RA ontology will be necessary for consistency checking of requirement constraints and for future SPL concrete products derivation in the AE lifecycle.

On the other hand, recall that software quality is not considered early in either [4, 5, 7, 11], being left to Domain Evaluation, in DD; moreover, in [7] it is left completely to the domain engineer or architect expertise; our contribution is to have included software quality explicitly

into Domain Scoping, profiting of the facet intrinsic specification, which applies to all domain facets; the accomplishment of quality issues plays a major role for the whole SPL success [11]. Besides, when software components are developed according to some quality requirement, and the software product is built from these components, the quality of the whole product cannot be assured; however, we consider global product quality to a certain extent, assuring that the quality properties of the domain architectural style are fulfilled; but this is still an open research issue.

The RA ontology for the EHR-HIS domain is being developed, with semiautomatic support tools involving interaction with the domain engineer. In the near future the QuaDRA process will be applied to Service-Oriented SPL, considering also the EHR-HIS domain.

## Acknowledgments

This work has received partial support from national projects and institutions: DARGRAF PG 03-8730-2013-2 project of CDCH (Consejo de Desarrollo Científico y Humanístico), and Postgraduate Studies of Computer Science, Faculty of Science, both of Venezuela Central University.

## References

- [1] Reinhartz, I. et al. (2013). (Eds.), [Domain Engineering. Product Lines, Languages and Conceptual Models](#). Berlin: Springer.
- [2] Berard, E. (1992). [Essays in Object-Oriented Software Engineering](#). Nueva York: Prentice Hall.
- [3] Medeiros, F., de Almeida, E. & de Lemos, S. (2010). [Designing a set of service-oriented systems as a software product line](#). Proceedings Fourth Brazilian Symposium on Software Components, Architectures and Reuse (pp. 70-79). Salvador, Bahia, Brazil.
- [4] Käkölä, T. (2010). [Standards initiatives for software product line engineering and management within the international organization for standardization](#). Proceedings 43rd Hawaii International Conference on System Sciences (pp. 1-10). Manoa, Hawaii.
- [5] Korff, A. (2014). [Implementing ISO 26550: 2013 model-based](#). Proceedings Fifth International Conference on Complex Systems Design & Management (pp. 325-326). Paris, France.
- [6] Clements, P. & Northrop, L. (2001). [Software product lines: practices and patterns](#). USA: Addison Wesley.
- [7] Pohl, K., Bockle, G. & Van Der Linden, F. (2005). [Software Product Line Engineering: Foundations, Principles, and Techniques](#). London: Springer.
- [8] Munir, Q. & Shahid, M. (2010). [Software Product Line: Survey of Tools](#). Master’s thesis. Linköping University, Department of Computer and Information Science.
- [9] Siegmund, N. et al. (2012). [SPL-Conqueror: Toward Optimization of Non-Functional Properties in SPL](#). Software Quality Journal.
- [10] Apel, S. et al. (2013). [Feature-Oriented Software Product Lines](#). New York: Springer.
- [11] ISO/IEC. (2013). [ISO/IEC NP 26550: Software and Systems Engineering – Reference Model for Software and Systems Product Lines](#). ISO/IEC JTC1/SC7 WG4.
- [12] Losavio, F., Ordaz, O. & Esteller, V. (2015). [Refactoring-Based Design of Reference Architecture](#). Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS) 5(1), pp. 32-48.



- [13] Bjørner, D. (2006). [Software Engineering 3 Domains, Requirements, and Software Design](#). Texts in Theoretical Computer Science. Berlin: Springer.
- [14] ISO/IEC (2011). [ISO/IEC 25010: Systems and software engineering -- Systems and software Quality Requirements and Evaluation \(SQuaRE\) -- System and software quality models](#). ISO/IEC JTC1/SC7/WG6, Draft.
- [15] Object Management Group (OMG). (2011). [Business Process Model and Notation \(BPMN\)](#). Online [May 2015].
- [16] Morrison, C., Iosif, A. & Danka, M. (2010). [Report on existing open-source electronic medical records](#). Technical Report No. 768, UCM-CL-TR-768. University of Cambridge.
- [17] Institute of Medicine (US). (2003). [Key capabilities of an electronic health record system](#): Letter report. Committee on Data Standards for Patient Safety. National Academies Press.
- [18] Shaw, M. & Garlan, D. (1996). [Software architecture: perspectives on an emerging discipline](#). USA: Prentice Hall.
- [19] Pazos, P. (2010). [Taller de OpenEHR: Un Estándar para crear HCEs](#). Congreso Argentino de Informática y Salud. Caba, Argentina.
- [20] Farroñay, K. & Trujillo, A. (2013). [Sistema de registro de atención médica para un centro de salud de nivel I-3 de complejidad](#). Disertación Doctoral, Universidad Peruana de Ciencias Aplicadas.
- [21] Losavio, F., Ordaz, O. & Santos, I. (2015). [Proceso de análisis del dominio ágil de sistemas integrados de salud en un contexto venezolano](#), ENL@CE 12 (1), pp. 101-134.
- [22] ISO/IEC (2005). [ISO/IEC 25030: Systems and software engineering -- Systems and software Quality Requirements and Evaluation \(SQuaRE\) -- System and software quality models](#). ISO/IEC JTC1/SC7/WG6, Draft.
- [23] Wagner, S. (2013). [Software Product Quality Control](#). Boston: Springer.
- [24] Samilovich, S. (2010). [OpenEMR – Historia Clínica Electrónica de código abierto y distribución gratuita, apta para su uso en el sistema de salud argentino](#). Congreso Argentino de Informática y Salud. Caba, Argentina.
- [25] Caulton, G. (2013). [PatientOS – an Open Source \(GPL\) Health Care Information System](#). Online [Jun 2015].
- [26] COSGov Vietnam (2005). [Care2X](#). International Conference and Exposition. Hanoi, Vietnam.
- [27] Object Management Group (OMG). (2005). [Unified Modelling Language Superstructure](#). Version 2.0.
- [28] Mohabbati, B. et al. (2013). [Combining service-orientation and software product line engineering: A systematic mapping study](#). Information and Software Technology 55(11), pp. 1845-1859.
- [29] Herrera, J., Losavio, F. & Ordaz, O. (2015). [Ingeniería del Dominio con el Estándar ISO/IEC 26550 para Líneas de Productos de Software considerando la faceta calidad](#). Memorias Conferencia Nacional de Computación, Informática y Sistemas (pp. 107-118). Valencia, Venezuela.
- [30] de Moraes, M., de Almeida, E. & Romero, S. (2009). [A systematic review on software product lines scoping](#). Proceedings 6th Experimental Software Engineering Latin American Workshop (pp. 63-72). São Carlos, Brazil.
- [31] Northrop, L. & Clements, P. (2012). [A Framework for Software Product Line Practice](#). Software Engineering Institute. Online [Jun 2015].
- [32] Bayer, J. et al. (1999). [PuLSE: A methodology to develop software product lines](#). Proceedings 1999 Symposium on Software reusability (pp. 122-131). Los Angeles, USA.

**2016 2nd International Conference on  
Humanity and Social Science  
(ICHSS2016)**

**August 28-29, 2016 in Phuket, Thailand**

**2016 2nd International Conference on Humanity and Social Science**

DEStech Publications, Inc.  
439 North Duke Street  
Lancaster, Pennsylvania 17602 U.S.A.

Copyright © 2016 by DEStech Publications, Inc.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Manufactured in the U.S.A.

Main entry under title:  
2016 2nd International Conference on Humanity and Social Science (ICHSS 2016)

ISBN: 978-1-60595-388-5

Permission for publication outside of this Conference Proceeding  
must be given by the Publisher.

## Product Lines Scoping Using ISO/IEC 26550 Reference Model Considering Software Quality

Juan C. HERRERA<sup>1</sup>, Francisca LOSAVIO<sup>2</sup>, Oscar ORDAZ<sup>2,\*</sup>  
and Jørgen BØEGH<sup>3</sup>

<sup>1</sup> PFG Universidad Bolivariana De Venezuela, Caracas, Venezuela

<sup>2</sup> Most Laboratory, Escuela De Computación, UCV, Caracas, Venezuela

<sup>3</sup> Lemvig Gymnasium, Lemvig, Denmark

\*Corresponding author

**Keywords:** Domain engineering, Software product lines (SPL), SPL scoping, ISO/IEC 26550, Quality model, ISO/IEC 25010, Business processes, Integrated health care information systems.

**Abstract.** Software Product Line (SPL) is a family of similar products or systems, sharing common and reusable elements or assets. This work is framed within the Domain Engineering (DE) lifecycle of SPL Engineering (SPLE), where major efforts are to construct the SPL Reference Architecture (RA) and Asset Repository. The main goal of this paper is to introduce techniques for the Domain Scoping sub-phase of Product Line Scoping (PLScop), first DE phase, defined in ISO/IEC 26550 (2013). Three approaches are compared, two SPLE approaches for DE of Pohl et al. (2005) and the standard, with classic Bjørner's (2006) DE. Software quality consideration is crucial for the success of SPL, and we specify it as a new Bjørner's intrinsic facet. The PLScop process for Domain Scoping is outlined and illustrated with a case study on the Integrated Healthcare Information Systems domain.

### Introduction

*Software Product Lines (SPL)* is an approach that provides massive personalization of individual solutions from a repository of reusable software assets, in a particular domain [1, 5]. A domain is defined as the minimal set of properties describing precisely a family of problems in which a computational application or system is involved for their solution [2]. The main goal of *SPL Engineering (SPLE)* [5], is to reuse common elements in similar products or systems of the SPL family. Complex domain analysis must be achieved in order to specify and delimit the SPL family, developed from core assets. Products of the SPL family can be developed according to customer requirements, by instantiating the *SPL Reference Architecture (RA)* or framework where feasible choices are selected among a generally huge number of configurations. Two main lifecycles drive SPLE: *Domain Engineering (DE)*, where RA and Asset Repository are constructed, and *Application Engineering (AE)*, where the RA variability model is instantiated to derive new products [5]. In this work, DE and its first phase, Product Line Scoping (PLScop) in the new standard ISO/IEC 26550 Reference Model for SPLE and Management [3, 4, 6] will be considered. PLScop, following a top-down approach, deals with a broad capture of domain knowledge, to delimit the SPL, including risk analysis, economical feasibility and identification of the SPL family of products to be constructed. PLScop becomes necessary, since the knowledge on the SPL domain should be captured early, to reduce the major effort concentrated on the basic phases of *Domain Requirements Engineering (DRE)* and *Domain Design (DD)*, where RA is built. Quality assurance should also be taken into account early, because quality related to domain functional (FR) and non-functional requirements (NFR) is responsible in a major degree of the SPL variability; however most of the approaches leave it to DD. Our idea is to consider software components, often common components that respond to precise quality goals, provided by other often variant components. Traceability of FR and NFR is achieved by construction guaranteeing the RA evolution, which is a priority SPL requirement.

The main goal of this paper is to introduce specific techniques for the *Domain Scoping*, sub-phase of PLScop. On one hand, we establish a comparison of three DE approaches, based on their terminology and definitions [5, 6, 7]. Frameworks generally state “What to do” but not “How to do”; we propose the integration of Bjørner’s DE stage [7], as a technique which includes *business processes (BP)* as the main domain knowledge source; we profit of his *facet* notion, defined as a finite set of generic forms of describing a domain from different stakeholders’ perspectives (viewpoints), namely, *BP, Support Technology, Management & Organization, Rules & Regulations, Human Behavior, and Intrinsic*s. Facets and views notions are not new in Software Engineering [9]; in [7], each facet represents a view of the domain; the union of these views conforms a complete domain specification; the special “*intrinsic*s” facet contains descriptors or attributes (*entity, function, event, behavior*) necessary to describe all other facets. On the other hand, we include *quality*, specified by the standard ISO/IEC 25010 quality model [10], as a new descriptor of the intrinsic facet (see Table 1), to specify early in PLScop the domain quality properties related to FR and NFR of future SPL products. *Software quality* is defined as the degree to which software is capable of providing and maintaining a specified level of service [10]. Software quality issues in general, are not treated properly in SPL, since usual approaches focus more on products’ features directly perceived by the user [11], which is not the case of quality properties that emerge as “implicit functionalities” during the SPL development process; nevertheless, quality requirements are responsible of most of the SPL variability. Moreover, RA to be evolutionary, must guarantee the overall domain quality [6] and, to a certain extent, the quality of the derived concrete products members of the SPL family. A process, called PLScop, is proposed in this paper; it is illustrated by a brief case study on the *Integrated Healthcare Information Systems (HIS)* domain. Our adaptation of PLScop from [6] can be of practical use, fulfilling also illustrative academic purposes on its “modus operandi”. Software quality issues will be emphasized, to guarantee that RA holds by construction the expected quality of an evolutionary architecture.

Table 1. Intrinsic to describe all domain facets.

Intrinsic descriptors	Description
Entities	Phenomena and concepts of the domain
Functions	Operations (actions) performed on the entities
Events	Imply changes in entities by function invocations, i.e., actions in the domain
Behavior	Sequences of actions and events affecting the domain entities
Quality	Quality is specified in [10] in terms of characteristics or properties. They are specified in BP as quality goals to be accomplished by FR (functions, activities or tasks)

Besides this introduction and the conclusion, this work is structured as follows: a second section presenting the integration of Bjørner’s approach into Domain Scoping; a third section outlines PLScop; finally, the fourth section is dedicated to the HIS case study.

### Approaches for SPLE: Pohl Et Al., ISO/IEC 26550, Bjørner

Three DE approaches or frameworks, selected for their maturity and relevance, are presented in what follows [5, 6, 7]. To establish a comparison, a correspondence is made between the approaches to clarify the terminology used, see Table 2; the approach of Pohl et al. [5] is imbedded into the new standard [6], and in [3, 4] this is confirmed; Product Management [5] and PL Scoping [6] have very similar goals. Bjørner’s approach [7] is not focused on SPL but on single systems, but his DE stage is general and his domain model concerns a family of systems.

Table 2. Correspondence of Pohl et. al., ISO/IEC 26550 and Bjørner DE approaches.

SPLE - Pohl et al. [5]	ISO/IEC 26550 - SPL Reference Model [6]	Software Engineering - Bjørner [7]
<p><i>Domain Engineering Process (DE):</i> 5 key sub-processes, to anticipate changes to take into account SPL evolution; they are performed in the order established by the company or organization requiring the SPL.</p> <ul style="list-style-type: none"> <li><i>Product Management:</i> it deals with economic aspects and market strategy; its main concern is the management of the SPL product portfolio. Input are company goals and output is the product “roadmap” that determines major common and variable feature of the future SPL products, as well as a schedule for product planned release. <i>Product Portfolio Management</i> is a dynamic decision process, where the list of active new products and/or projects is constantly updated and revised. Major activities involved are: <ul style="list-style-type: none"> <li><i>Product Portfolio Scoping:</i> it defines the products to be developed by the SPL and their key features: existing products in the domain can also be studied;</li> <li><i>Domain Scoping:</i> it defines the boundaries of the SPL domain</li> <li><i>Assets Scoping:</i> it identifies particular components to be reused or developed for reuse</li> </ul> </li> </ul> <p><b>Note:</b> only Product Management will be considered in this work</p>	<p><i>Domain Engineering Lifecycle (DE):</i> 5 phases, performed in the order established by the organization requiring the SPL, as in [5].</p> <ul style="list-style-type: none"> <li><i>Product Line Scoping (PLScop):</i> it defines SPL boundaries for DE, envisioning major common and variable features to all products within the SPL. Existing products can also be considered. Economical aspects are analysed and the commercialization of the SPL product family is planned; PLScop is responsible of the whole SPL management: it involves 3 sub-phases: <ul style="list-style-type: none"> <li><i>Product Portfolio Scoping:</i> 1. Identify products that the SPL should be developing, producing, marketing and selling (product “roadmap”); 2. The study of common and variable feature of products should provide guidance to meet business objectives and face evolution; 3. Schedule for introducing products to the market. It is input to Domain Scoping. Some proposed techniques are: Problem Solving Method; Documents survey; Feature Modelling; Group Decision Process.</li> <li><i>Domain Scoping:</i> identify and bound functional or organizational areas that SPL will impact to provide sufficient reuse potential to justify the SPL creation. Some proposed techniques are: Task Force Team (knowledge engineering); Problem Solving Methods focused on technical aspects; Functional Decomposition Method; Review; Group Decision Methods.</li> <li><i>Assets Scoping:</i> identify the boundaries of core assets, providing a first glance at common and variable assets. Some proposed techniques are: Task Force Team (knowledge engineering); Problem Solving Methods; Data Mining; Effort estimation for existing products, Cost and Risk estimation/evaluation; Return On Investment (ROI) analysis; Group Decision Methods.</li> </ul> </li> </ul> <p>Output of PLScop: <i>asset proposal</i>; it includes major assets (functional areas and high-level common and variable features of all products) that will be included in SPL with their quantified costs and benefits estimation results. <b>Note:</b> only Product Line Scoping will be considered in this work</p>	<p><i>Domain Engineering or Domain Development (DE):</i> 3 phases; <i>domain</i> is defined as any areas of interest and <i>application domain</i> is a domain where computing is applied, the <i>Domain Model</i> is the main output; it describes not just a specific instance of a domain, but a family of systems; it includes 6 stages: <ol style="list-style-type: none"> <li><i>Stakeholders identification:</i> identifies and classify <i>all relevant</i> people (stakeholders) in different work teams within an organization; they are identified in the DRE phase in [6].</li> <li><i>Domain acquisition:</i> to capture, describe and classify information on the domain from stakeholders viewpoints; many <i>domain description units (DDU)</i> are the output.</li> <li><i>Domain analysis:</i> to analyse DDU to study conflicts, inconsistencies and incompleteness during domain acquisition; <i>facets</i> are used for this analysis.</li> <li><i>Domain Modelling:</i> the <i>domain model</i>, i.e., the meaning of the domain description, must capture: “intrinsic” or facet common to all facets, business processes, management and organisation, rules &amp; regulations, support technologies, and human behaviour. the Domain Model is the output; we introduce the new <i>Quality Intrinsic Facet Descriptor</i> to specify domain quality; a Domain Quality Model (DQM) [8], adapted from [10] specifies this quality in our work.</li> <li><i>Domain Verification/Validation:</i> the domain model formal specification can be verified with automatic tools, pr validated against relevant stakeholders of different workgroups.</li> <li><i>Domain Theory:</i> a formal theory built on a mathematical Domain Model specification. <b>Note:</b> only stages 1, 2, and 3 DE stages are considered in this work.</li> </ol> </p>
<ul style="list-style-type: none"> <li><i>Domain Requirements Engineering:</i> it encompasses all activities for eliciting and documenting common and variable requirements of the SPL. Input: the product roadmap. Output: domain requirements models and the SPL variability model</li> </ul>	<ul style="list-style-type: none"> <li><i>Domain Requirements Engineering (DRE):</i> it has to adhere to the specification of the SPL’s high-level features provided by PLScop. Based on these features, it creates detailed common and variable requirements sufficient to guide subsequent domain design (RA design), realization and testing phases.</li> </ul>	<ul style="list-style-type: none"> <li><i>Requirements Engineering:</i> It supports the domain organization operations identified in the previous stage; it specifies the software requirements (requirements prescriptions) <b>Note:</b> It focuses on single systems, not on SPL.</li> </ul>
<ul style="list-style-type: none"> <li><i>Domain Design:</i> the domain architecture or Reference Architecture (RA), which is a generic abstract architecture for the SPL, is designed. The architect determines how requirements, including variability are reflected into RA. It refers to “traditional” design activities: <i>Building the Architecture; Complexity reduction; Modelling; Simulating/prototyping; Validating; Quality assurance:</i> <b>Note:</b> This crucial aspect is very poorly treated and left to the architect criteria.</li> </ul>	<ul style="list-style-type: none"> <li><i>Domain Design (DD):</i> It draws upon the specifications to develop an SPL architecture that enables the realization of the planned commonality and variability within the SPL. The main goal is to produce the Reference Architecture (RA), defining general SPL structure and textures. <i>RA Evaluation</i>, which describes a quality assurance technique. For SPLE, an assessment of RA is crucial for the SPL-related quality requirements. Only an RA that supports quality requirements sufficiently will be evolutionary. <b>Note:</b> Quality considerations are practically missing in [6], and in <i>Domain Design Management</i> traceability must be assured.</li> </ul>	<ul style="list-style-type: none"> <li><i>Software Design</i> the process of turning all prescribed software requirements and their documentation into executable code with appropriate hardware.</li> <li><i>Software architecture design:</i> it builds a first specification of software indicating how to handle requirements in terms of components and their interconnections, without detailing the components, producing all appropriate documentation. <b>Note:</b> It focuses on single systems, not on SPL.</li> </ul>

Product Line Scoping steps in [6], and DE (steps 1 to 4) in [7], will be considered in this work. Main conclusions on this comparison are the following: - The first four steps of Bjørner’s DE stage [7] fit nicely into the SPL approach [6], to have a systematic way to define PL Scoping; - the approach of Pohl et al. [5] treats very lightly quality issues in SPL development, leaving them at DD level, where the architect is given complete responsibility of constructing RA assuring its global quality and

integrating the variability model; - the ISO/IEC 26550 [6] standard introduces explicitly the sub-phase *RA Evaluation* to consider quality, however quite late in DD; nevertheless, quality assurance is claimed in [6] to be crucial to guarantee RA evolution, a priority quality requirements. This fact confirms our approach of including quality issues early in PLScoP.

## **PLScoP: Adapting PL Scoping of ISO/IEC 26550**

Begin PLScoP

1. Product Portfolio Scoping: Input: Documentation of existing products;

1.1 Construct the Domain Quality Model (DQM) [8] by customizing [10] to the domain;

1.2 Study domain Market Products (they are assumed to exist, and have semantically similar main functionalities), to infer about future SPL products; an extractive bottom-up process, based on reengineering techniques, produces automatically a Candidate Architecture (CA) conformed by the union of the existing products' architectures, represented by undirected and connected graphs. Note: The bottom-up process has been defined in [8], and will not be detailed here; CA is a connected graph that will be the "core" to define future SPL products.

1.3 Construct The Extended Quality Model (EQM) table, gathering information on components and quality properties required/provided by these components, using DQM, showing also architectural solutions or technological mechanisms solving or "implementing" these quality properties; EQM is another CA view;

1.4 Perform a Marketing Study should also be included for the feasibility of future SPL products.

- The Products Roadmap artefact includes the study of existing products, CA in UML(<http://www.omg.org/spec/UML/2.0/>, 2005)and/or its representation by an ontology, EQM Table and an informal document specifying the future SPL products;

Output: CA type: UML and/or ontology; Future Products Document type: document; DQM type: table; marketing study type: document; EQM type: table;

2. Domain Scoping (adapted from [7]): Input: Domain informal documentation provided by the organization requiring the SPL; DQM; CA;

2.1 Stakeholders identification: Input: visits, interviews, workshops, questionnaires (techniques for each activity should be specified).

- Identify groups of stakeholders with similar interests in the organization requiring the SPL.

Output: list of stakeholders' groups type text.

2.2 Domain acquisition: Input: List of Stakeholders' groups

- Capture and gather information from the stakeholders' groups w.r.t. the SPL domain, into declarations to build informal Domain Description Units (DDU).

Output: DDU type: table

2.3 Domain analysis: Input: DDU

- Analyse DDUs, study inconsistencies, and study first the domain business processes (BP) facet, as recommended in [7]; BP are represented as behaviours in the facet specification by intrinsic descriptors from a stakeholder viewpoint relevant to the domain; the facet specification has three representations: table, UML diagram, and BPMN(<http://www.omg.org/spec/BPMN/2.0/>, 2011) diagram for BP.

- Quality is included as a new intrinsic facet descriptor;

- Other facets relevant to the domain can be specified in the same way;

IOOutput: Facet specification type: table, UML and BPMN diagrams;

2.4 Domain Modelling: Input: Facets specification;

- The domain model is obtained from the BPMN facet specifications; this specification is low-grained and should be agglomerated using the Aggregate BPMN mechanism, to extract other reusable components that can conform future SPL products;

- The components obtained by Aggregation, should be analysed w.r.t. the Future Products Document;

- Update CA (UCA): the Domain Model is integrated to CA by connecting the new components found;
  - The quality required by new components is considered in the BPMN specification of the Domain Model;
  - Update EQM (UEQM) table with the information of the new products added to CA;  
Output: Domain Model type: BPMN; UCA type: UML or ontology; UEQM type: table;
  - 3. Asset Scoping: Input: DQM, Marketing Study, UEQM, Domain Model; UCA;  
Information on core assets will be extracted from the outputs of the previous steps to conform the Core Asset Repository (CAR) as the Asset Proposal;  
Output: CAR type: repository;
- end PLScoP;

### Applying the Main Domain Scoping Steps of PLScoP for the HIS Domain

HIS [12] are complex integrated information systems, generally located in different and distant institutions and with mandatory NFR. Their general architectural style is a hybrid event-based style SOA /Layers, [8, 13]. HIS must facilitate transparent sharing of medical information such as *Electronic Health Records (EHR)*, offering also on-line services at remote locations, with wide support of information technology. Standards such as HL7 are mandatory to achieve EHR interoperability [12]. The HIS domain, called *EHR-HIS*, is restricted here to basic HIS functionalities,. The Quality Intrinsic Facet descriptor was shown in Table 1. The domain model specification allows to integrate into PLScoP a technique of quality assurance, reducing effort in next DRE and DD phases. The ISO/IEC 25010 standard Quality Model [10] specifies software product quality. This model is customized for the EHR-HIS domain as follows [8]: *Functional suitability (completeness, correctness), Reliability (availability), Performance efficiency (time behaviour, capacity), Maintainability (modifiability, modularity), Security (authenticity, confidentiality, integrity), Compatibility (interoperability), Portability (adaptability)*. Priority and mandatory quality requirements for EHR-HIS are Interoperability, Security and Availability. Domain Model integrates a set of models obtained from the facets' descriptions; viewpoints are the declarations of the stakeholders' groups, and are registered as *Domain Description Units (DDU)*.

- *Domain acquisition. Technique used:* statements or declarations formulated by stakeholders. grouped into the DDU and represented textually in Table 3.

Table 3. DDU for EHR-HIS Domain from the Doctors' Viewpoint.

EHR-HIS Description Unit	
Viewpoint	Stakeholders' Group: Doctors
<i>Declarations</i>	
Patient is attended at hospital under scheduled appointment	
If it is the patient first appointment, a new EHR must be created by nurse, else nurse retrieves patient existing EHR	
Patient EHR is accessed by doctor	
New exams and laboratory results can be added to patient EHR by doctor, if it is the case	
Diagnosis and medical orders for patient are produced by doctor to conclude medical attention	
A new appointment is scheduled by nurse if required by Doctor	

- *Domain analysis. Technique used:* facets are specified from DDU in BPMN for BP.



Table 4. Appointment Services BP Facet from the Doctors' viewpoint—Example.

Domain	EHR-HIS – Healthcare institution requiring the SPL			
Viewpoint	Stakeholders' Group: Doctors			
BP Facet	Appointment Services Process			
Entities	Functions	Events	Behaviour	Quality
Medical Appointment	Emission of new appointment	<i>New appointment is required, if necessary; appointment is scheduled</i>	Doctor registers patient for new appointment; appointment is scheduled.	<i>Availability of on-line connection to access EHR-HIS for Medical Appointment</i>
Medical Order	Emission of medic. order	<i>Medical order is elaborated for present consultation</i>	Doctor emits medical order	<i>Availability of on-line connection to access EHR-HIS</i>
EHR	EHR Access  EHR Register	<i>EHR is found or created</i>  <i>EHR is requested by doctor; doctor attends patient medically and EHR is modified</i>	Nurse confirms the existence of patient EHR or creates an EHR for new patient; nurse provides EHR to doctor Doctor reviews patient EHR, adds laboratory and/or examinations/imaging results, if any, to patient EHR; patient is attended medically; EHR is modified.	<i>Security (confidentiality, authenticity, integrity) to access HER, Availability of on-line connection to access EHR-HIS, time behaviour, modifiability, availability and interoperability of EHR documents.</i>
Diagnosis	Perform diagnosis	<i>Medical hand-outs are required on-line; on-line consultation with doctors in different or local institutions to perform diagnosis, registered on EHR</i>	Doctor requests on-line hand-outs, consults on-line with other local doctors in different or local institutions; patient's EHR must be shared; laboratory and/or examinations /imaging results are reviewed; diagnosis is produced	<i>Time behaviour and availability of on-line connection to access hand-outs, catalogues and other doctors</i>

### Conclusion and Future Work

Three domain engineering approaches [5. 6. 7] have been considered for PLScOP, specifying techniques and artifacts. Software product quality [10] was incorporated early into the domain model. Finally, PLScOP has been outlined and main activities of Domain Scoping have been shown. The development of DRE and DD phases are on-going work and support tools are under design.

### References

- [1] I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen and J. Bettin, (Editors), Domain Engineering. Product Lines, Languages and Conceptual Models, Springer, 2013.
- [2] E. Berard, Essays in OO Software Engineering, Prentice Hall, Nueva York, 1992.
- [3] T. Käkölä, Standards initiatives for SPL engineering and management within the international organization for standardization. In System Sciences (HICSS), 43rd Hawaii International Conference, IEEE (2010) 1-10.
- [4] A. Korff, Implementing ISO 26550: 2013 model-based. Complex Systems Design & Management, 346 (2015).
- [5] K. Pohl, G. Bockle and F. Van Der Linden, SPL E: Foundations, Principles, and Techniques, Springer, 2005.

- [6] ISO/IEC NP 26550: Software and Systems Engineering—Reference Model for Software and Systems Product Lines. ISO/IEC JTC1/SC7 WG4, 2013.
- [7] D. Bjørner, Software Engineering 3 Domains, Requirements, and Software Design. Texts in Theoretical Computer Science. EATCS Series. Editors: W. Brauer G. Rozenberg A. Salomaa, Springer-Verlag, Berlin Heidelberg, 2006.
- [8] F. Losavio, O. Ordaz and Esteller V. Quality-Based Bottom-up Design of Reference Architecture applied to Healthcare Integrated Information Systems, RCIS (2015) 76-81 IEEE, Athens, Greece.
- [9] P. Kruchten, Architectural Blueprints-The “4+1” View Model of Software Architecture. Tutorial, Tri-Ada, 95, (1995) 540-555.
- [10] ISO/IEC 25010: SQuaRE, Quality model, 2011.
- [11] K. Lee, K. Kang and J. Lee, Concepts and Guidelines of Feature Modeling for Product Line, Software Engineering, 7th. Int. Conf. on Software Reuse: Methods, Techniques, and Tools, (2002) 62-77.
- [12] Institute of Medicine (US), Key capabilities of an electronic health record system: Letter report. Committee on Data Standards for Patient Safety, National Academies Press, 2003.
- [13] M. Shaw, D. Garlan, Software Architecture. Perspectives of an emerging discipline, Prentice-Hall, 1996.

# Diseño de Arquitecturas de Referencia para Líneas de Productos de Software Orientadas a Servicios: Revisión Documental Sistemática

Juan C. Herrera<sup>1</sup>, Francisca Losavio<sup>2</sup>, Oscar Ordaz<sup>3</sup>  
jchr1982@gmail.com, francisosavio@gmail.com, oscarordaz55@gmail.com

<sup>1</sup> PFG Informática para la Gestión Social, Universidad Bolivariana de Venezuela, Caracas, Venezuela

<sup>2</sup> Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela

<sup>3</sup> Escuela de Matemáticas, Universidad Central de Venezuela, Caracas, Venezuela

---

**Resumen:** Una Revisión Documental Sistemática (RDS) permite extraer conocimiento sobre un tema de investigación; esto se hace mediante preguntas adecuadas, a partir del gran volumen de información disponible en general en la literatura y en particular en internet. El objetivo del trabajo es realizar una RDS en el tema de Líneas de Productos de Software Orientadas a Servicios (LPSOS), el cual integra los enfoques de Líneas de Productos de Software (LPS) y de Arquitecturas Orientadas a Servicios (SOA). La intención es que a partir de la RDS se identifiquen métodos (fases, etapas, actividades, artefactos y roles) para el diseño del artefacto principal de las LPS, la Arquitectura de Referencia (AR) siguiendo un modelo de arquitectura empresarial SOA y conformada por Servicios Web. Los resultados de la RDS servirán de guía para definir un proceso sistemático para el diseño de una AR en el contexto LPSOS. Esta arquitectura contemplará el tratamiento de requisitos de calidad en las etapas tempranas del desarrollo de LPSOS para garantizar su evolución, y el modelado de la variabilidad para la derivación de productos concretos, aspectos esenciales en el contexto de producción industrial de software.

**Palabras Clave:** Líneas de Productos de Software; Arquitectura Orientada a Servicios (SOA); Líneas de Productos de Software Orientadas a Servicios (LPSOS); Revisión Documental Sistemática (RDS); Arquitectura de Referencia; Servicios Web.

**Abstract:** A Systematic Review (SR) allows to extract knowledge about a research topic; this is done by formulating adequate questions from the large volume of information available in the literature in general and particularly searching on the Internet. The goal of the work is to perform an SR on the subject of Service Oriented Software Product Lines (SOSPL), which integrates the approaches of Software Product Lines (SPL) and Service Oriented Architectures (SOA). The intention is to use SR methods (phases, stages, activities, artifacts and roles) for the design the Reference Architecture (RA), the main artifact of LPS, following the SOA enterprise architecture model conformed by Web Services. The results of the SR will provide guidance to define a systematic process to design a RA in the SOSPL context. This architecture will cover the treatment of quality requirements at early stages of SOSPL development to ensure its evolution, and the variability modeling for the derivation of concrete products, which are essential issues in the context of software industrial production.

**Keywords:** Software Product Lines; Service Oriented Architecture (SOA); Service Oriented Software Product Lines (SOSPL); Systematic Review (SR); Reference Architecture (RA); Web Services.

---

## I. INTRODUCCIÓN

El desarrollo de sistemas de software de gran complejidad y de gran escala representa un desafío para los investigadores y desarrolladores en el área de la *Ingeniería de Software (IS)*; surge entonces la tendencia al desarrollo basado en componentes reutilizables, para configurar nuevos sistemas con rapidez y de más bajo costo [1][2][3].

El enfoque de *Líneas de Productos de Software (LPS)* permite la construcción de soluciones individuales basadas en un repositorio de componentes de software reutilizables. El hecho de proporcionar soluciones individuales responde a diversas necesidades en el software con respecto a la funcionalidad, la tecnología subyacente y las propiedades no funcionales, como por ejemplo rendimiento y espacio de memoria [4]. Las LPS prometen distintos beneficios [4][5], de los cuales los más importantes son: a) adaptabilidad a las necesidades del cliente,

b) reducción de costos, c) mejora de la calidad global, d) evolución, y e) disminución del tiempo de comercialización. Este enfoque representa un gran reto para los investigadores y desarrolladores, principalmente por la tarea de construir artefactos (*activos* o del inglés *assets*) suficientemente genéricos para poder ser reutilizados en la configuración de nuevos sistemas acordes con las necesidades de los clientes [2][3]. Para el abordaje adecuado de este enfoque, la *Ingeniería de Líneas de Productos de Software (ILPS)* ofrece métodos y técnicas eficaces para un desarrollo de software basado en la reutilización, con el fin de: a) apoyar arquitecturas de software configurables o instanciables y b) permitir la personalización masiva de los sistemas de software [6][7]. Ha sido reconocida como un enfoque exitoso para la gestión de la variabilidad y de la reutilización. La ILPS consta de dos ciclos de vida principales: *Ingeniería del Dominio (ID)* e *Ingeniería de la Aplicación (IA)* [8]. La ID abarca el análisis y la identificación del ámbito de aplicación de la LPS, que incluye la captura de todo el conocimiento sobre el dominio de interés a través del modelado de partes comunes (del inglés “commonality”) y de partes variables o puntos de variación (del inglés “variation points”) que están presentes en la *Arquitectura de Referencia (AR)*, la cual es una arquitectura genérica basada en un esquema instanciable que dirige la derivación de productos concretos de la familia LPS.

Por otra parte, la *Arquitectura Orientada a Servicios* (del inglés *Service Oriented Architecture (SOA)*) [9][10][11] es un modelo de arquitectura empresarial distribuida, centrado en la comunicación en redes; su objetivo central es la integración y desarrollo rápido de sistemas empresariales en una organización o dominio específico. Un *servicio* se define como una unidad discreta de una funcionalidad del negocio, que está disponible a través de una *interfaz estándar* del servicio, mediante la cual el servicio se comunica con otros servicios [12]; el servicio puede ser visto como un componente de software reutilizable. Una implementación muy conocida de servicios son los Servicios Web, en el contexto de la WWW [9]. SOA especifica lineamientos para la construcción independiente de servicios alineados al negocio que pueden ser combinados a partir de procesos de negocio de alto nivel y trasladados a soluciones computacionales en el contexto empresarial. El valor real de SOA no es solo la provisión adecuada de servicios, sino más bien cuando los servicios reutilizables se combinan para “implementar” procesos de negocios ágiles y flexibles. SOA ha demostrado su rentabilidad en el desarrollo de sistemas de software interoperables, reutilizables, adaptables y de rápido desarrollo, por lo cual existen importantes ventajas mutuas en la convergencia de SOA y LPS [11][13][14], dando origen al enfoque de *Líneas de Productos de Software Orientadas a Servicios (LPSOS)*. La integración de los enfoques LPS y SOA requiere que las organizaciones de desarrollo de LPS apliquen la gestión de la variabilidad respecto a los servicios que son los componentes esenciales de SOA, y que los desarrolladores de sistemas orientados a servicios apliquen técnicas que usan los desarrolladores de LPS. Sin embargo esta tarea no es trivial; la necesidad de variación en cuanto a servicios, puede ser impulsada por las condiciones del mercado, oportunidades del negocio, las nuevas tecnologías y otros factores relacionados con la empresa. Esta necesidad puede expresarse en forma de

objetivos de negocio establecidos por las organizaciones que desarrollan LPSOS [13]. Ahora bien, en un contexto de LPSOS, particularmente para la construcción de una AR, deben manejarse tres problemas principales: (1) la representación de la AR conformada por componentes y conectores [15], en donde los componentes son servicios y los conectores son mensajes entre servicios, incluyendo el traslado de servicios a componentes arquitecturales; (2) la gestión de las propiedades de calidad requeridas por los componentes que representan funcionalidades para asegurar la completitud y evolución de la AR; (3) la gestión de la variabilidad funcional y no funcional. Las soluciones a estos problemas determinarán los pasos a seguir para definir un método de diseño de una AR basada en servicios en un contexto LPS.

En los estudios anteriormente referenciados se han reflejado los problemas mencionados, aún no completamente resueltos, para algunos de los cuales este trabajo pretende proporcionar una guía para su solución. El objetivo de esta investigación, se centra principalmente en una Revisión Documental Sistemática de la literatura (RDS) [16] de trabajos recientes sobre diseño de una AR basada en servicios y configurada como una arquitectura de software [15] para LPS y no como una arquitectura empresarial, para identificar métodos sistemáticos de diseño que respondan a los tres problemas identificados anteriormente y que además contemplen:

- artefactos de entrada y de salida,
- enfoque de diseño utilizado para la AR,
- roles de quienes participan en las actividades del modelo de proceso del método.

Una RDS [16] permite extraer el conocimiento sobre el estado del arte en un tema específico de investigación; este conocimiento es extraído mediante preguntas adecuadas, a partir de un gran volumen de información disponible. En consecuencia una RDS que aborde el tema de métodos de desarrollo de LPSOS, es indispensable para determinar los alcances y limitaciones de las propuestas encontradas. Para alcanzar este objetivo, nuestra RDS se apoya en un marco de evaluación conformado por la integración de los aspectos de interés de tres propuestas de evaluación [17][18][19], que abordan las siguientes temáticas: LPS, SOA y AR-SOA. Este marco de evaluación posteriormente será utilizado para comparar los “mejores” métodos encontrados por la RDS, es decir aquellos que satisfacen más del 50% del total de las 33 propiedades descritas en las Tablas VII, VIII, IX.

Este trabajo está estructurado de la siguiente manera, además de esta introducción: la Sección II presenta algunos aspectos a ser considerados para realizar la RDS; en la Sección III se define el marco integrado de evaluación; la Sección IV presenta el análisis de la RDS realizada; la Sección V discute los resultados obtenidos. Finalmente, la Sección VI presenta las conclusiones y las perspectivas.

## II. ASPECTOS A SER CONSIDERADOS EN LA RDS

El enfoque LPSOS requiere considerar los siguientes elementos: la *Gestión de Procesos de Negocios (GPN)* o del inglés *Business Process Management (BPM)*, la gestión de la variabilidad funcional y no funcional, la representación de una

Arquitectura de Referencia Orientada a Servicios y la gestión de la calidad. Estos aspectos son considerados prioritarios en una RDS de dos Santos Rocha y Fantinato [7] que presenta enfoques para la aplicación conjunta de LPS y GPN, incluyendo el soporte de SOA para GPN, y en la cual se concluye que la combinación LPS, BPM y SOA es factible y beneficiosa para la construcción de un proceso sistemático integrado para elaborar una LPSOS. A continuación se describen brevemente cada uno de estos aspectos.

#### A. Gestión de Procesos de Negocios

GPN en [20] define un *proceso de negocio* como un conjunto de actividades que se realizan en coordinación con un entorno organizacional y técnico. Estas actividades se desempeñan de manera conjunta para realizar los objetivos del negocio. Cada proceso de negocio es generado por una sola organización, pero puede interactuar con los procesos de negocios generados por otras organizaciones. Las partes fundamentales de GPN son su ciclo de vida y el lenguaje de modelado de procesos de negocio del inglés *Business Process Management Language (BPML)* [20]. Existen varios BPMLs concebidos en la academia y la industria que se utilizan para representar el flujo de trabajo y los artefactos y/o producidos utilizados durante la realización de las actividades en el ciclo de vida. Algunos de estos son el Diagrama de Actividades (UML 2.0) y la Notación de Procesos de Negocios [21], los cuales fueron evaluados por un framework en [22]. Esta evaluación determinó que el *Business Process Modeling Notation (BPMN)* es uno de los BPMLs más utilizado en la fase de diseño de un ciclo de vida GPN. Representa los procesos como actividades centradas en el control de flujo. El lenguaje de modelado debe proporcionar elementos notacionales y mecanismos de apoyo a aspectos tales como la especificación de las actividades, el flujo de secuencia, el flujo de datos y los eventos. En cambio, no soporta especificaciones de aspectos o requisitos no funcionales. Por otra parte, la ejecución de los procesos de negocio, además de la selección de una plataforma de ejecución, como por ejemplo SOA, puede requerir de otros lenguajes. Por ejemplo el Lenguaje *Business Process Execution Language (BPEL)* es utilizado para la ejecución de los procesos de negocio a través de la composición de servicios por orquestación [20]. No obstante, la OMG<sup>1</sup> con la definición de la semántica de cada elemento de la notación en BPMN 2.0, permitió la posibilidad de ejecutar los modelos sin necesidad de algún otro lenguaje de apoyo. Por ejemplo la herramienta Activiti<sup>2</sup> que implementa BPMN. En la actualidad, la mayoría de los enfoques de GPN enfatizan el desarrollo de un proceso de negocio, y a partir de este mismo se derivan muchas variantes, que están especializadas según diversas necesidades de la organización. Los procesos de negocio pueden tener piezas comunes para un grupo (es decir, la familia) de diferentes casos de aplicación, implicando cierta variabilidad en la selección de las actividades de un proceso de negocio. Este aspecto podría ser utilizado para LPS, donde una funcionalidad común puede ser desarrollada como un activo reutilizable para la creación de nuevas variantes de los procesos de negocio en un dominio [13][20].

<sup>1</sup> Object Management Group

<sup>2</sup> <http://activiti.org>

#### B. Gestión de la Variabilidad

La *Gestión de la Variabilidad* (del inglés *Variation Management*) [13] se aplica tanto a SOA como a LPS; los puntos de variación pueden ser implementados ya sea por un único servicio (donde una interfaz de servicios puede ofrecer parametrización por *QoS (Quality of Service)* o algún otro mecanismo) o a través de servicios similares que traten cada variación. Mohabbati [11], indica que la variabilidad es gestionada teniendo varios servicios que tienen la misma funcionalidad, pero difieren en su QoS.

#### C. Arquitecturas de Referencia en LPS y SOA

Otro aspecto importante corresponde a la construcción de la AR, y en particular a las *Arquitecturas de Referencia Orientadas a Servicios (AROS)* que apoyen la reutilización sistemática de los activos principales para la configuración de productos de software concretos. En la RDS [23] se indica que existen diferencias entre las *Arquitecturas de Líneas de Productos (ALP)* y las AR: ALP tienden a ser menos abstractas que las AR, pero más abstractas que las arquitecturas de software concretas, es decir, las ALP son un tipo de AR [11][23]. Con respecto a nuestra RDS, AR y ALP son consideradas similares. Las AROS se han propuesto como un tipo de AR; lo que debe estar claro es que las AROS en el contexto SOA son un modelo de arquitecturas empresariales para la integración de aplicaciones en la empresa, no son arquitecturas de software “reales” en el sentido de Garlan y Shaw [15]. En cambio en el contexto LPS, AROS son arquitecturas de software genéricas e instanciables a partir de un modelo de variabilidad, para generar sistemas concretos orientados a servicios de la familia LPS. Una AROS-LPS será entonces para nosotros una AR en un contexto de LPSOS. Varias AROS bajo SOA han sido propuestas para sistemas de gobierno, entornos de trabajo colaborativo, o sistemas de atención de la salud; generalmente no son AROS “puras” solo conformadas por servicios, sino que incluyen componentes que no son servicios. Una RDS realizada sobre las AROS bajo LPS en [7], indica que faltan directrices sobre cómo definir las AROS en particular como definir las AROS que soporten un alto grado de variabilidad [11].

#### D. Gestión de la Calidad en Servicios

En los actuales momentos no se concibe un sistema de software que no tenga la calidad necesaria que satisfaga las expectativas de los usuarios finales. Más aun, las propiedades de calidad para una LPSOS son fundamentales para garantizar su completitud y carácter evolutivo. Por ello, es vital abordar las características de calidad de estos sistemas desde una etapa temprana en el ciclo de vida del desarrollo de software. Particularmente esto se debe hacer en el ciclo de vida de la ID en el proceso de una *ILPS Orientada a Servicios (ILPSOS)*. Los servicios exponen propiedades de calidad de bajo nivel a las que responden en tiempo de ejecución, para intercomunicar con otros servicios; los valores de los QoS se ofrecen a través de parámetros. En general, lo que ocurre con las propiedades de calidad de alto nivel a las cuales también deben responder los servicios compuestos, cuando representan componentes funcionales en el caso de AROS, no se trata en detalles. Los *Servicios Web* (en inglés *Web Services (WS)*) son tipos de servicios que pueden ser descubiertos, especificados y accedidos utilizando XML y protocolos Web estándar. El

elemento central de un WS está en la definición estándar de su interfaz en WSDL<sup>3</sup>, que contiene entre otras informaciones, los QoS. La descripción de un WS tiene dos partes: - una definición abstracta que describe el servicio como un componente, con su interfaz y operaciones; - una definición que describe los enlaces concretos de las operaciones hacia puntos de acceso, que contienen una descripción de los datos, una dirección física y la información del protocolo de comunicación [24]. En un entorno de una aplicación que utiliza WS, se distinguen tres funciones: intermediario, proveedor y consumidor de servicios. El proveedor registra un servicio en el intermediario o “broker”, por ejemplo en un repositorio UDDI<sup>4</sup>. El consumidor descubre un servicio gestionado por el intermediario y llama al servicio del proveedor. Todas las partes utilizan protocolos tipo SOAP<sup>5</sup>, un protocolo estándar basado en XML para el manejo de respuestas y solicitudes de WS [24]. Está claro que un WS, como componente de una aplicación, también debe responder a propiedades de calidad de alto nivel, las cuales no necesariamente son observables en ejecución, por ejemplo persistencia de la información y políticas de confidencialidad.

### III. MARCOS PARA LA EVALUACIÓN DE MÉTODOS PARA EL DESARROLLO DE AR-LPS, SOA, AR-SOA

#### A. Framework para la Evaluación de los Métodos

El marco de evaluación o *framework*, constituido por las Tablas I, II y III, es utilizado como herramienta de análisis. La Tabla I muestra los criterios para evaluar métodos de diseño AR-LPS. La Tabla II muestra criterios para evaluar métodos de desarrollo de sistemas orientados a servicios bajo SOA. La Tabla III presenta criterios para evaluar métodos de desarrollo de AR-SOA. El objetivo de este framework de evaluación no es la comparación de métodos, sino proporcionar un panorama de métodos actuales para tomar decisiones en cuanto a un método general para el diseño de una arquitectura AROS-LPS.

AROS contempla tres estrategias de desarrollo:

- *bottom-up o ascendente (reactivo o extractivo)*: derivar los servicios a partir de componentes existentes, es decir de aplicaciones heredadas o de una AR ya realizada, como por ejemplo la identificada a partir de un proceso “bottom-up” que considera la refactorización de productos existentes del mercado.
- *top-down o descendente o proactivo*: derivar los servicios a partir de la especificación del modelo del dominio expresado mediante procesos de negocios.
- *middle-out o meet-in-the-middle o híbrido*: partir de la especificación de los procesos de negocio como orquestaciones de servicios y relacionarlos con los componentes de una AR ya existente.

Para la RDS de este trabajo, se buscan los métodos o enfoques existentes (estado del arte) que describan de forma sistemática las etapas, actividades, roles y artefactos necesarios para realizar una AR en un contexto LPSOS (AROS-LPS),

utilizando cualquiera de los tres frameworks mencionados [17][18][19].

La aplicación del marco de evaluación proporcionará la base para la definición de categorías detalladas de elementos o aspectos presentes en un criterio. Mediante un conjunto de preguntas, este estudio intenta abordar el alcance de los métodos para identificar las diferencias y similitudes que estos presentan. Como no existe un marco específico para la evaluación de enfoques LPSOS, se van a combinar los frameworks de las Tablas I, II y III, en un marco único que incluya todos los criterios de evaluación. A continuación se describen en detalles los criterios:

#### Criterios del marco de evaluación para métodos de desarrollo de AR-LPS [17]:

Hay cuatro criterios esenciales para la evaluación de estos métodos:

- *Contexto del método*: situación del problema, objetivo específico, ciclo de vida abordado.
- *Usuario del método*: personas que solucionan el problema, que rol desempeña, guía proporcionada para aplicar el método.
- *Componentes del método o proceso para la resolución del problema*: 1) modelos subyacentes, 2) lenguaje, 3) definición de los pasos, su secuencia, 4) artefactos, 5) vistas arquitecturales, como el modelo “4+1 vistas” de Kruchten [25]), 6) variabilidad (nivel de abstracción donde la variabilidad es manifiesta: a nivel de proceso de negocio, a nivel de los componentes), 7) herramientas que facilitan la realización del método.
- *Gestión de la calidad (nuevo criterio)*: ¿Cómo el método soporta la especificación de las propiedades de calidad y/o requisitos no funcionales?; el aseguramiento de la calidad del producto es una actividad crucial para el éxito de la industria del software, pero es más importante cuando se trata del desarrollo de la LPS, dado que la reutilización masiva de activos de software hace que sus atributos de calidad (o propiedades medibles de un artefacto de software) impacten en la calidad de los productos de la LPS [26].
- *Validación del método*: casos de estudios donde se compruebe la efectividad del método, el estilo arquitectural elegido satisface las expectativas de los usuarios finales.

#### Criterios del marco de evaluación para métodos de desarrollo de sistemas bajo el modelo SOA [18]:

Hay cinco criterios esenciales para la evaluación de un método para desarrollo de sistemas o aplicaciones aisladas bajo el modelo SOA de arquitectura empresarial, que no involucran AR sino una arquitectura concreta de un producto:

- *Concepto de servicio*: para analizar si los conceptos de servicio, servicios de negocios y software, son soportados; un servicio de negocio es un conjunto específico de acciones que son llevadas a cabo por una organización, un servicio de software expone las funcionalidades de la

<sup>3</sup> Web Services Definition Language

<sup>4</sup> Universal Description, Discovery and Integration

<sup>5</sup> Simple Object Access Protocol

aplicación que pueden ser reutilizados y compuestos basados en las necesidades u objetivos del negocio, por lo tanto un servicio de software soporta la ejecución de un servicio de negocio.

- *Estrategia de realización o desarrollo*: top-down, bottom-up, middle-out o meet-in-the-middle que combina ambos; esta última estrategia es la más recomendada para implementar SOA.
- *Cobertura del ciclo de vida* o fases principales para el desarrollo bajo SOA; según SOMA [27] son: identificación, especificación y realización.
- *Grado de granularidad del análisis*: el nivel de detalle utilizado para describir el método, que va desde un nivel bastante descriptivo, con gran cantidad de detalles, hasta un nivel más abstracto, para describir un proceso más general, flexible y adaptable.
- *Accesibilidad y validez*: Indica que el método debe proveer documentación accesible y poder ser validado.

### Criterios del marco de evaluación para métodos de desarrollo de AR-SOA [19]:

Las arquitecturas de software [15] juegan un papel importante en la determinación de la calidad del sistema, ya que forman la columna vertebral de cualquier sistema intensivo de software exitoso; el estilo arquitectónico garantiza por sí solo el cumplimiento de propiedades de calidad globales del dominio. En el contexto LPS, las AR reúnen el conocimiento del dominio y son los instrumentos que proveen esquemas reutilizables. Entre los principales beneficios de AR, se encuentran una mejor comprensión del dominio, el establecimiento de un vocabulario común, la reutilización de activos, la consistencia en la representación del sistema, y un menor tiempo de comercialización. SOA, como modelo de *arquitectura de referencia empresarial (AR-SOA)* ha tomado la atención en los últimos años. Facilita la integración mediante la interoperabilidad de servicios, la escalabilidad y la reutilización, ya que los sistemas basados en SOA son independientes respecto al lenguaje de programación y a la plataforma de ejecución utilizada. El estilo arquitectónico de sistemas de software empresariales orientados a servicios “puros”, que no involucran otro tipo de sistemas no orientados a servicios, es basado en eventos [15], siguiendo un modelo cliente-servidor para la distribución y la comunicación en redes.

El marco de evaluación AR-SOA surge de responder la pregunta: ¿Cuáles características de SOA han sido consideradas durante el diseño y desarrollo de la AR-SOA? Se presentan tres criterios:

- *Publicación del servicio*: La AR-SOA debe permitir la publicación de la descripción de los servicios directamente a los consumidores, a través de intermediarios, por ejemplo UDDI.
- *Calidad del servicio (QoS)*: La AR-SOA debe permitir utilizar mecanismos para capturar, controlar, registrar y señalar el incumplimiento de requisitos no funcionales

descritos en los acuerdos de servicio por su interfaz. Estos acuerdos especifican las características de calidad que proporciona el servicio en ejecución, como el tiempo de respuesta, rendimiento, disponibilidad, entre otros.

- *Composición de los servicios*: es el proceso de construir un nuevo servicio a partir de servicios existentes [9]. Todos los servicios se comunican por paso de mensajes, a través de protocolos. La AR-SOA debe permitir la composición de los servicios: a través de los procesos de negocio, mediante una orquestación coordinada de servicios que enfatiza el paso de mensajes, o mediante la coreografía, más orientada hacia los aspectos funcionales que el servicio debe proporcionar.

### B. Criterios y Elementos del Marco Único de Evaluación basado en los Tres Framework AR-LPS, SOA y AR-SOA

El objetivo del *Marco Único de Evaluación (MUE)* no es solo proporcionar una visión general de los actuales métodos de ingeniería para la LPSOS, sino además indagar cómo los métodos difieren en cuanto a criterios de diseño. Esto es especificado en la sección del informe de la RDS, donde se muestran los resultados de nuestra evaluación. La aplicación de MUE proporciona la base para la definición detallada de los elementos presentes en los criterios. Mediante preguntas, este marco intenta abordar por ejemplo, madurez, sentido práctico y alcance de los métodos para encontrar las similitudes y diferencias, además, de cómo se identifican los servicios a partir del modelo de procesos de negocio, y cuáles son los componentes que constituirán una AROS-LPS y que implementarán los servicios, incluyendo la relación entre estos componentes y el modelo de variabilidad.

A continuación, las Tablas I, II, III describen criterios y elementos que conforman el MUE, con sus respectivas preguntas de análisis. Estas preguntas ayudan a la captura de información relevante para cada uno de los criterios de evaluación. El MUE se deriva integrando todos los criterios de evaluación y será utilizado en la RDS para efectuar el análisis de los artículos seleccionados como *estudios primarios*: son aquellas investigaciones relevantes que quedan para el análisis, luego de haberse aplicado los criterios RDS de inclusión, exclusión y calidad:

**Tabla I:** Criterios y Elementos de Evaluación para Métodos de Desarrollo de AR-LPS, Adaptado de [17]

Criterios	Elementos	Preguntas
Dominio de la LPS	Ámbito	¿Cuál es/son el/los dominio(s) de la LPS en la que se enfoca el método? Ejemplo: dominio de los sistemas de información.
Contexto del método	Objetivo específico	¿Cuál es el objetivo específico del método?
	Fase(s) de línea de productos	¿Qué fases de la LPS cubre el método? Por ejemplo, el método abarca solamente las fases de análisis y diseño en el ciclo de vida de la ID.
	Entradas del método	¿Cuál es (son) la (s) entrada (s) del método?
	Salidas del método	¿Cuáles son la (s) salida (s) del método?
Usuario del	Grupo	¿Quiénes son los actores que

método	destinatario	intervienen en el método?
	Motivación	¿Cuáles son los beneficios que percibe el usuario al utilizar el método?
	Habilidades necesarias	¿Qué habilidades necesita el usuario para cumplir las tareas requeridas por el método?
Componentes del método	Orientación	¿Cómo el método guía al usuario mientras se aplica el método?
	Estructura del método	¿Cuáles son los pasos, fases y/o actividades del proceso de diseño que se utilizan para llevar a cabo un objetivo específico del método?
	Artefactos	¿Cuáles son los artefactos creados y gestionados por el método?
	Vistas arquitecturales	¿Cuáles son las vistas arquitecturales que se consideran en el método?
	Lenguaje/notación	¿El método define un lenguaje o notación para representar los modelos, diagramas y otros artefactos que produce?
	Variabilidad	¿Cómo el método soporta la expresión de la variabilidad funcional y no funcional?
	Herramientas	¿Cuáles son las herramientas y/o técnicas que apoyan el método?
Gestión de la calidad en el método	Propiedades de calidad	¿Cómo el método soporta la especificación de las propiedades de calidad y/o requisitos no funcionales?
Validación del método	Madurez	¿Se ha validado el método en casos de estudios prácticos industriales?
	Calidad de la arquitectura	¿Cómo el método valida la calidad del diseño de la AR-LPS?

**Tabla II:** Criterios y Elementos de Evaluación para Métodos de Desarrollo de Sistemas bajo el Modelo SOA, Adaptado de [18]

Criterios	Elementos	Preguntas
Concepto de servicio	Servicios de negocios	¿Cuál es el concepto de servicio soportado?
	Servicios de software o WS	
	Ambos	
Estrategia de realización o desarrollo	Top-down	¿Cuál es el tipo de estrategia utilizada para el desarrollo bajo SOA?
	Bottom-up	
	Middle-out	
Cobertura del ciclo de vida	Identificación	¿Cuál es el grado de cobertura con relación al ciclo de vida SOA?
	Especificación	
	Realización	
Grado de granularidad del análisis	Bajo	¿Cuál es el grado de granularidad del nivel de abstracción del método?
	Moderado	
	Alto	
Accesibilidad y validez	Documentación no accesible	¿La documentación del método es apropiada y sirve de guía para su uso en la práctica?
	Parcialmente documentado	
	Bien documentado (Caso de estudio)	

**Tabla III:** Criterios y Elementos de Evaluación para Métodos de Desarrollo de AR-SOA, Adaptado de [19]

Criterios	Elementos	Preguntas
Publicación del servicio	Directamente a los consumidores	¿La AR-SOA permite la publicación de la descripción de los servicios?
	A través de intermediarios	
Composición de los servicios	Orquestación de servicios	¿La AR-SOA permite que el desarrollo de software pueda ser construido por medio de la composición de servicios?
	Coreografía de servicios	

Calidad del Servicio	Cumplimiento de los requisitos no funcionales de alto nivel	¿La AR-SOA permite el uso de mecanismos para validar que se cumplan los requisitos no funcionales de alto nivel requeridos por la funcionalidad que un servicio desempeña?
	Cumplimiento de los requisitos no funcionales de bajo nivel (QoS)	¿La AR-SOA permite el uso de mecanismos para validar que se satisfagan los requisitos no funcionales de bajo nivel (QoS) descritos en los acuerdos de su interfaz?

#### IV. REVISIÓN DOCUMENTAL SISTEMÁTICA (RDS)

Kitchenham [16] indica que la gestión del proceso de revisión sistemática se fundamenta en tres fases principales: Planificación, Realización e Informe de la revisión. En la Tabla IV se describen las fases, las etapas de cada fase y los artefactos generados en el proceso RDS.

**Tabla IV:** Proceso de Revisión Documental Sistemática

Fases	Etapas	Artefactos
Planificación de la revisión	Identificación de la necesidad de una revisión.	Protocolo
	Especificación de la(s) interrogante(s) de investigación.	
	Desarrollo del protocolo de revisión.	
Realización de la revisión	Identificación de estudios relacionados.	Estudios aceptados
	Selección de estudios primarios.	
	Evaluación de la calidad de los estudios.	
	Extracción de los datos.	
	Síntesis de los datos.	Modelo de datos (checklist)
Informe de la revisión	Especificación de mecanismos de difusión.	Framework de evaluación
	Presentación de resultados	

##### A. Planificación de la Revisión

Esta fase define los objetivos de la investigación y el protocolo en que la revisión se ejecutará.

1) *Identificación de la Necesidad:* Revisar los procesos de desarrollo actuales con el fin de proporcionar una guía para el diseño de una AR para LPSOS, en nuestro caso. Además, de identificar problemas pendientes por resolver y posibles áreas para la mejora de los procesos de desarrollo.

2) *Desarrollo del Protocolo:* El protocolo es el plan o conjunto de pasos a seguir en un estudio, constituido por las preguntas de investigación, las estrategias de búsqueda, los criterios y procesos de selección, los criterios de evaluación de la calidad, el modelo y proceso de extracción de datos, y el plan de análisis y síntesis de los datos.

Las siguientes *Preguntas de Investigación (PI)* fueron consideradas en inglés; aquí las colocamos en castellano:

- P11. ¿Cuáles métodos que siguen enfoques “bottom-up” y/o “top-down” existen (estado del arte) para pasar de (conversión) una arquitectura basada en componentes a una arquitectura basada en WS o como llevar a cabo esta correspondencia?



- PI2. ¿cómo se expresan los requisitos de calidad de las funcionalidades respecto a los QoS de los WS?

Con relación a la estrategia de búsqueda, las fuentes de búsqueda seleccionadas para encontrar los estudios primarios son los siguientes: Google Academic y ResearchGate.

Se identificaron las palabras claves de búsqueda (descriptores): Software Product Line, Service-Oriented Architecture, Methods and Tools Design, Reference Architecture, Web Services, Business Process, Non-Functional Properties, Quality of Service; combinándolas con operadores lógicos AND, OR; para capturar los estudios primarios.

Los *Criterios de Inclusión (CI)* utilizados para incluir los estudios relevantes en esta RDS son:

- CI1: Estudios publicados entre 2008 y 2015
- CI2: Que aborde la arquitectura para el desarrollo de Líneas de Productos de Software para Familias de Productos Orientados a Servicios utilizando SOA.
- CI3: Que en su contenido incluya al menos el ciclo de vida de la ID para LPSOS.
- CI4: Que en su contenido describa un método de desarrollo para LPSOS.

Los *Criterios de Exclusión (CE)* son:

- CE1: Que no aborde la integración de LPS y SOA como alternativa para el desarrollo de productos de software.
- CE2: Que esté escrito en un lenguaje diferente al inglés o castellano y que no esté disponible en formato PDF; o que tenga como contenido solo una presentación en un congreso o workshop realizado por los autores.
- CE3: Si existen trabajos repetidos, se elige uno de ellos.

En la selección inicial de las publicaciones entre los resultados de búsqueda, se considera la lectura de los títulos, palabras claves y resúmenes de las publicaciones encontradas. Las publicaciones serán almacenadas en carpetas independientes de acuerdo a la expresión de búsqueda utilizada, y de esta forma facilitar la ubicación de los estudios para realizar el análisis con mayor fluidez y comodidad.

Con respecto a la extracción de datos y método de síntesis, se prevé la construcción de tablas para la obtención de datos relacionados con las preguntas de investigación.

Para estandarizar la forma en que la información estará representada, se definió un formato (modelo de datos) para recopilación de los datos en los estudios seleccionados. Las Tablas VII, VIII, IX muestran la descripción de las propiedades de extracción o “checklist” que serán utilizadas a lo largo de la RDS. Las Tablas X y XI muestran los hallazgos encontrados de la extracción de los datos relevantes en los estudios.

El modelo de datos será utilizado en la fase de Realización de la revisión, para seleccionar los mejores estudios, y luego en la fase de Informe de la revisión, estos estudios serán analizados a través del MUE que se describe en la Sección III.

### B. Realización de la Revisión

En esta fase se sigue el protocolo previamente establecido. Como resultado de la búsqueda, 331 estudios fueron localizados, de los cuales solo 21 fueron aceptados inicialmente para su análisis a través de tres “checklists”, las Tablas VII, VIII y IX respectivamente. La Tabla V a continuación, muestra un resumen de los resultados indicando la cantidad de estudios localizados, aceptados y rechazados.

**Tabla V:** Relación de Estudios Localizados y Aceptados según la Fuente de Datos

Fuentes de datos	Artículos		
	Aceptados	Rechazados	Localizados
Google Academic	12	283	295
ResearchGate	9	27	36
<b>Total</b>	<b>21</b>	<b>309</b>	<b>331</b>

De los estudios localizados se descartaron aquellos artículos cuyos contenidos eran idénticos, aun cuando sus identificadores eran diferentes. También, se identificaron estudios similares, que abordaban el mismo proceso y en los cuales participaban los mismos autores, aunque el título del estudio era distinto; para estos casos, se tomó la decisión de elegir el estudio más completo y actualizado. La Tabla VI muestra los estudios primarios aceptados según el tipo de evento y año de publicación.

**Tabla VI:** Relación de Estudios Primarios Relativos a Procesos de Desarrollo para LPSOS según Tipo y Año de Publicación

Año	Tipo de evento			Total
	Conferencia	Revista	Tesis	
2008	2	1		3
2009	2			2
2010	2	2	1	5
2011	5	1		6
2012	1			1
2013			1	1
2014	1	1		2
2015		1		1
<b>Total</b>	<b>13</b>	<b>6</b>	<b>2</b>	<b>21</b>

Los 21 estudios primarios describen un proceso para el desarrollo de una LPSOS. A continuación, un análisis más detallado se llevó a cabo en cada uno de ellos. Este análisis se realizó mediante la aplicación de dos “checklists” (Tablas X y XI) respecto a las propiedades que nos interesan evidenciar en los estudios (Tablas VII, VIII y IX). Esto permite caracterizarlos con relación a las propiedades presentes en sus respectivos contenidos. Se utiliza la nomenclatura (++), (+), (-), que significa que la calidad de la documentación fue buena, regular con un valor en ambos casos de 1 punto c/u o que carece de esa propiedad con un valor de 0 punto, respectivamente. Este aspecto también influyó en la decisión sobre cuales estudios fueron seleccionados para ser evaluados por el MUE.

**Tabla VII:** Descripción de las Propiedades: Métodos para AR-LPS

Propiedad	Descripción
<b>ID</b>	<b>Ciclo de Vida de Ingeniería del Dominio (ID)</b>
An	Fase de Análisis
Di	Fase de Diseño
Re	Fase de Realización
<b>IA</b>	<b>Ciclo de Vida de Ingeniería de la Aplicación (IA)</b>
<b>Estrategia Desarrollo</b>	<b>Estrategía utilizada para abordar el método [28]:</b>
Td	Top-down (proactivo)
Bu	Bottom-up (extractivo/reactivo)
Hi	Middle out o Meet-in-the-middle (híbrido)
<b>Método</b>	<b>El método describe un proceso sistemático:</b>
Act	Actividades a seguir
E/S	Entradas y salidas
Art	Artefactos producidos/utilizados
Rol	Roles de participación en el proceso
Arq	Vistas arquitecturales
CasoEst	Caso de estudio que valida el proceso
<b>ModVariab</b>	<b>Modelado de la variabilidad funcional y no funcional (calidad) [29]</b>
Car	En el modelo de características [46]
Arq	En el modelo arquitectural
PN	En el modelo de procesos de negocio
WS	En servicios: un servicio proporciona variantes mediante la parametrización; o varios servicios ofrecen la misma funcionalidad, pero con diferentes QoS (implementaciones)
<b>Estd</b>	Uso de un Modelo de Calidad estándar para especificar las propiedades de calidad [30]: ISO-9126, ISO-25010, McCall, Boehm, FUR PS, Dromey, etc.
<b>EspArq</b>	<b>El método considera la especificación de la AR</b>
<b>Tech</b>	<b>Enfoques y Tecnologías: SCA<sup>6</sup>; MDA<sup>7</sup>; MDE<sup>8</sup>; SOC<sup>9</sup></b>
<b>Ont</b>	<b>Ontologías:</b> expresan el conocimiento del dominio; expresan la AR como medio para relacionar el dominio del problema con el dominio de la solución con una terminología común.
<b>Herr</b>	<b>Herramientas:</b> grado en que el método es automatizado; si una herramienta es propuesta o implementada.

**Tabla VIII:** Descripción de las Propiedades: Métodos para SOA

Propiedad	Descripción
<b>GPN</b>	<b>incluye el ciclo de vida de GPN [20]:</b>
NotacPN	Notación (BPMN, etc.) para el modelado de los procesos de negocios
LengEj	Utiliza el lenguaje BPEL para implementar (ejecutar) los procesos de negocios en la etapa de implementación del ciclo de vida de GPN
<b>EstArq</b>	<b>El estilo (modelo) arquitectural es SOA</b>
<b>WS</b>	<b>WS como tecnología para realizar SOA</b>
<b>Métodos de conversión</b>	<b>Método para identificar, especificar y realizar los servicios bajo un enfoque SOA</b>
Car/S	Actividades de como traducir del Modelo de Características a Servicios (“top-down”)
PN/S	Actividades de como traducir de un Modelo de Procesos de Negocios a Servicios (“top-down”)
WS/Arq	Actividades de como traducir los Servicios a Componentes arquitecturales (“top-down”)
Arq/WS	Actividades de como traducir una AR de software (componentes, conectores) a una AR cuyo componentes son Servicios (“bottom-up”)
<b>Métodos de desarrollo</b>	<b>Métodos de desarrollo SOA para las fases de Análisis y Diseño [28]:</b> IBM RUP/SOMA; SOAF; SOUP; SOMA; Erl methodology; Papazoglou methodology
<b>ModRef/ModConcept</b>	<b>Modelos de Referencia/Modelos conceptuales SOA [31]:</b> SOA-RM <sup>10</sup> ; SOA-RFA <sup>11</sup> ; SOA Ontology <sup>12</sup> ; SOMF <sup>13</sup> ; PIM4SOA <sup>14</sup> ; SoaMI <sup>15</sup>

<sup>6</sup> Service Component Architecture

<sup>7</sup> Model Driven Architecture

<sup>8</sup> Model Driven Engineering

<sup>9</sup> Service Oriented Computing

<sup>10</sup> OASIS Reference Model for SOA

<sup>11</sup> OASIS Architecture Foundation for SOA

<sup>12</sup> Open Group SOA Ontology

<sup>13</sup> Service-oriented Modeling Framework

<sup>14</sup> Platform-independent Model for SOA

<sup>15</sup> SOA Modeling Language

**Tabla IX:** Descripción de las Propiedades: Métodos para AR-SOA

Propiedad	Descripción
<b>Pub</b>	Permite la publicación de la descripción de los servicios directamente a los consumidores, a través de intermediarios
<b>Comp</b>	Permite que el software pueda ser construido por medio de la composición de los servicios
<b>RNF</b>	Cumplimiento de los requisitos no funcionales de alto nivel y/o de bajo nivel

**Tabla X:** Checklist de las Propiedades para Métodos AR-LPS Relevantes Identificadas en los Estudios Primarios sobre LPSOS

Estudios Primarios	AR-LPS																				Total	
	ID			IA	Estrategia Desarrollo			Método					ModVariab				Estd	Esp Arq	Tech	Ont		Herr
	An	Di	Re		Td	Bu	Hi	Act	E/S	Art	Rol	Arq	Caso Est	Car	Arq	PN	WS					
ZGCAL [32]	++	++	++	-	++	-	-	++	++	++	-	++	++	++	-	-	-	-	-	-	++	11
GB [24]	++	+	+	-	-	++	-	++	++	-	-	++	++	-	-	-	-	-	-	-	-	8
Istoan [2]	++	++	++	++	++	-	-	++	++	++	-	-	++	++	-	++	-	-	-	MDE <sup>16</sup>	-	12
RMALAAG [29]	+	-	++	-	++	-	-	++	++	++	-	-	-	-	-	-	++	-	-	-	-	7
CK [13]	++	-	-	-	-	++	-	++	-	-	-	-	++	-	-	-	++	-	-	-	-	5
GE [33]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
BGFF [34]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
EMA [35]	++	+	-	-	++	-	-	++	++	++	-	++	++	++	-	-	-	-	++	-	-	10
Med [3]	++	++	++	-	++	-	-	++	++	++	++	++	++	++	-	-	-	-	++	-	-	12
BMKARBH [36]	++	++	++	++	++	-	-	++	++	++	++	-	++	++	-	-	-	-	-	-	++	13
DRHU [37]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
PJ [38]	++	++	++	++	++	-	++	++	++	++	++	++	++	++	-	-	-	-	-	-	++	13
SS [39]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
AG [40]	++	++	++	++	++	-	-	++	++	++	++	++	++	++	-	++	++	-	-	-	-	14
BCCMV [41]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
MAGL [6]	++	++	++	++	++	-	-	++	++	++	-	++	++	++	-	++	++	-	++	-	-	14
LMN [42]	++	++	++	++	++	-	-	++	++	++	-	++	++	++	-	++	-	-	-	-	-	12
Galster [43]	++	+	-	-	-	++	-	++	++	++	++	-	++	-	-	-	-	-	-	-	-	8
AMKGBH [20]	++	++	++	++	++	-	-	++	++	++	++	++	++	++	-	++	++	-	++	MDE	-	17
NBG [44]	++	-	-	-	++	-	-	++	-	-	-	-	-	-	-	-	-	-	-	-	-	3
ALHG [45]	++	++	++	++	-	++	-	++	++	++	-	++	++	++	-	++	-	++	-	-	-	14
<b>21</b>	<b>21</b>	<b>13</b>	<b>12</b>	<b>8</b>	<b>16</b>	<b>5</b>	<b>0</b>	<b>21</b>	<b>14</b>	<b>13</b>	<b>6</b>	<b>9</b>	<b>14</b>	<b>11</b>	<b>2</b>	<b>5</b>	<b>6</b>	<b>0</b>	<b>5</b>	<b>2</b>	<b>1</b>	<b>4</b>

**Tabla XI:** Checklist de las Propiedades para Métodos bajo SOA y AR-SOA Relevantes Identificadas en los Estudios Primarios sobre LPSOS

Estudios Primarios	SOA										AR-SOA			Total
	GPN		EstArq	WS	Métodos de conversión a Servicios				Métodos de Desarrollo	ModRef/ModConcept	Pub	Comp	RNF	
	NotacPN	LengEj			Car/S	PN/S	WS/Arq	Arq/WS						
ZGCAL [32]	++	++	++	++	++	++	++	++	-	-	-	-	8	
GB [24]	-	-	++	-	++	-	-	-	-	-	-	-	2	
Istoan [2]	++	-	++	-	++	++	-	-	-	-	-	++	5	
RMALAAG [29]	-	-	++	-	-	-	-	-	-	-	-	-	1	
CK [13]	-	-	++	-	-	-	-	-	-	-	-	-	1	
GE [33]	-	-	++	-	-	-	-	-	-	-	-	-	1	
BGFF [34]	-	-	++	-	-	-	-	-	-	-	-	-	1	
EMA [35]	++	-	++	-	-	-	-	-	-	-	-	-	2	
Med [3]	-	-	++	++	++	-	-	-	-	RUP/SOMA	-	++	7	
BMKARBH [36]	-	-	++	++	++	-	-	-	-	-	-	++	4	
DRHU [37]	-	-	++	-	-	-	-	-	-	-	-	-	1	
PJ [38]	-	-	++	++	++	-	++	-	-	-	-	++	5	
SS [39]	-	-	++	-	-	-	-	-	-	-	-	-	1	
AG [40]	++	-	++	-	++	++	-	-	-	-	SoaML <sup>17</sup>	++	6	
BCCMV [41]	-	-	++	-	-	-	-	-	-	-	-	-	1	
MAGL [6]	++	-	++	-	-	++	-	-	-	-	-	++	5	
LMN [42]	++	-	++	-	++	++	++	-	-	-	-	++	6	
Galster [43]	-	-	++	-	-	-	-	-	-	-	-	-	1	
AMKGBH [20]	++	-	++	-	++	++	-	-	-	-	-	++	5	
NBG [44]	-	-	++	-	-	-	-	-	-	-	-	-	1	
ALHG [45]	-	-	++	++	++	++	++	-	-	-	-	++	6	
<b>21</b>	<b>7</b>	<b>1</b>	<b>21</b>	<b>5</b>	<b>10</b>	<b>7</b>	<b>4</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>10</b>	<b>2</b>	

<sup>16</sup> Model Driven Engineering

<sup>17</sup> <http://www.omg.org/spec/SoaML>

Luego de la captura de los datos, reflejados a través de las propiedades identificadas en las Tablas X, XI, aquellos estudios que obtuvieron el mayor número de propiedades consideradas, fueron seleccionados para ser evaluados utilizando las categorías, elementos y preguntas del MUE descrito en la sección III (Tablas I, II, III).

V. DISCUSIÓN

A continuación se presenta el informe de los resultados analíticos de la RDS, considerando cada pregunta de investigación y analizados a través de la aplicación del MUE, expresado a través de sus criterios y elementos de análisis. La Tabla XII identifica los 10 estudios más representativos relacionados con procesos de desarrollo de LPSOS, de los 21 estudios primarios inicialmente aceptados y que luego de ser analizados, son los que satisfacen el mayor número de propiedades encontradas en los métodos, este valor es obtenido de la sumatoria de los totales de las Tablas X, XI. MUE se aplicó a estos 10 estudios, que tratan explícitamente procesos de desarrollo para LPSOS.

**Tabla XII:** Estudios más Representativos entre los Primarios para LPSOS

Estudios	Propiedades
ZGCAL [32]	19
Istoan [2]	17
Med [3]	19
BMKARGBH [36]	17
PJ [38]	18
AG [40]	20
MAGL [6]	19
LMN [42]	18
AMKGBH [20]	22
ALHG [45]	20

A. Con Respecto a Métodos para AR-LPS (ver Tabla I)

1) *Criterio Dominio:* Los estudios tratan los dominios: gobierno electrónico o “e-government”, comercio electrónico (e-business), y reservaciones (e-travel).

2) *Criterio Contexto del Método:* objetivos, fases, entradas, salidas

*Objetivo:* Se destacan dos propuestas para el desarrollo de LPSOS [2][20] que utilizan el paradigma de la ingeniería dirigida por modelos, en inglés *Model Driven Engineering (MDE)*. La principal diferencia entre ambos, es que aplican el enfoque de forma diferente: [20] lo aplica para desarrollar el ciclo de vida de ID para LPS completo; y [2] lo utiliza en la etapa final del ciclo de vida de la IA, para la generación de los productos de la línea.

*Fases:* Todas las propuestas incluyen el ciclo de vida de la ID para LPS.

*Entradas:* Todas las propuestas parten de que el dominio del problema es conocido.

*Salidas:* Modelo de características [4][20][32][38][42][46], modelo de variabilidad [8], modelo de procesos de negocio [6][7][20][40][41], modelo de servicios [2], modelo de componentes [3][42] y AR [11][19][20][45].

3) *Criterio Usuarios del Método:* grupo, motivación habilidades, orientación

*Grupo:* La mayoría de las propuestas señalan como participantes o actores a ingenieros del dominio, arquitectos e ingenieros de software y analistas SOA. Los roles más

*representativos* son: *Business Analyst (Analista de Negocio)* que es el responsable de verificar que los candidatos a orquestación de servicios representen de forma precisa la lógica del negocio; *Domain Architect (Arquitecto del Dominio)* para la identificación de los componentes que proveen la implementación de las operaciones expuestas por los servicios; *SOA Architect (Arquitecto SOA)* para la identificación de los servicios candidatos; *Domain Designer (Diseñador del Domino)* and *Service Designer (Diseñador de Servicios)*. Para [3] los más importantes son: Analista de Negocio, el Arquitecto SOA y el Arquitecto del Dominio.

*Orientación:* Todas las propuestas indican al usuario como realizar las actividades en cada una de las fases del método.

*Motivación y habilidades:* Ninguna de las propuestas describe las habilidades requeridas por los usuarios de los métodos para poder realizar de forma efectiva las actividades presentadas.

4) *Criterio Componentes del Método:* estructura, artefactos, vistas, lenguajes, variabilidad y herramientas

*Estructura:* En cuanto a la *definición de los pasos y su secuencia*, existe consenso en determinar el alcance de la LPS mediante el análisis de sus características expresado a través del modelo de características, esto es desde el punto de vista estructural, y desde el punto de vista de su comportamiento, se realiza un análisis de los procesos de negocios utilizando por lo general BPMN para expresarlos, ambos conceptos son muy utilizados en la especificación de la variabilidad en la LPSOS.

En [20], para lograr la integración LPS-MDE-SOA, se comienza por la definición de los procesos de negocio y por la especificación del conjunto de actividades coordinadas para el cumplimiento de los objetivos organizacionales. Luego, los elementos de software necesarios para lograr estos objetivos (los servicios) son identificados, y para cada servicio, se identifican y desarrollan el conjunto de interfaces realizando la especificación del servicio, así como las correspondientes implementaciones de servicio.

*Artefactos:* el modelo de características [46] es el más utilizado, seguido del modelo de procesos de negocios [6][40], el modelo de variabilidad [32][38][42], y la AR [45].

*Variabilidad:* [3] indica que la gestión de la variabilidad es una actividad clave en ILPSOS, en la cual los servicios proporcionan la flexibilidad de las LPS y de sus productos, diferenciándolos en el momento de establecer el enlace de los servicios: en tiempo de diseño (estáticamente) y en tiempo en ejecución (dinámicamente). La variabilidad en la mayoría de los estudios analizados se realiza en el modelado de características, en algunos estudios ésta es realizada a través de los procesos de negocios [20], y en un solo estudio [45] es realizada a nivel de los servicios y de los componentes, expresados mediante una AR.

También en [20] se indica que la mayoría de los enfoques de gestión de procesos de negocio enfatizan el desarrollo de un proceso de negocio, y a partir del mismo, se derivan muchas variantes, especializadas según diversas necesidades de la organización. Los procesos de negocio pueden tener algunas piezas comunes para un grupo de diferentes casos de aplicación, donde implica cierta variabilidad en la selección de las actividades de un proceso de negocio. Una funcionalidad común puede ser desarrollada como un activo reutilizable para la creación de nuevas variantes de los procesos de negocio en

un dominio. Además, los puntos de variación, derivados de las propiedades no funcionales y tecnológicas de las actividades, proporcionan la funcionalidad del negocio con diferentes propiedades no funcionales y tecnológicas. Tal variabilidad facilita la configuración flexible del grupo de procesos de negocio dirigida por la calidad [20].

*Vistas:* Con relación a las *vistas arquitecturales*, la especificación de la arquitectura es la actividad, en la cual la arquitectura es documentada utilizando diferentes vistas y notaciones para representar los intereses de los diferentes actores involucrados. Existen cuatro estudios [3][32][40][42] que proponen la utilización de vistas arquitecturales similar al enfoque del 4+1 vistas [25].

En [40] se especifican 5 vistas: Vista de Características, Vista de Contrato de Servicio, Vista de Procesos de Negocio, Vista de Interface de Servicios y la Vista de Coordinación de Servicios. En [3] las 4 vistas arquitecturales consideradas son: Vista de Capas; Vista de Integración; Vista de Componente y Vista de Interacción. En [32] se especifican 3 vistas arquitecturales: Vista de Coordinación, Vista de Aplicación, Vista de Datos. En [42] se propone un estilo arquitectural heterogéneo, con tres niveles de descomposición asociados a tres estilos arquitecturales representados mediante 3 vistas: Vista Arquitectural de Servicios, Vista Arquitectural de Comunicación entre Procesos, Vista Arquitectural (en el ADL<sup>18</sup> C2). Solo los estudios [3][42] consideran la Vista de Componentes, donde existe una correspondencia entre las interfaces de los servicios y los componentes que las implementan.

*Lenguajes:* Los lenguajes utilizados por el método para el modelado arquitectural, [3][32][40][42] utilizan UML para expresar algunas de las vistas arquitecturales, especialmente [45] la utiliza para especificar la AR; para modelar los procesos de negocios, en [20][36] utilizan BPMN y en [30][40] el diagrama de actividades se expresa en UML.

*Herramientas:* En cuanto a las herramientas utilizadas solo cuatro trabajos mencionan alguna [20][32][36][38], que abordan parcialmente alguna parte del método propuesto; en [20] se utiliza FeatureMapper<sup>19</sup> para especificar el modelo de características y relacionar las características con los artefactos de software que las implementan; en [36] utilizan un enfoque basado en ontologías, donde utilizan el modelo de características [46] para la generación automática de composiciones de servicios ejecutables, para ello usan un lenguaje semántico de WS y utilizan un framework que soporta la composición de WS a nivel semántico, denominado *Web Service Modelling Ontology (WSMO)*, donde las composiciones de servicios son codificadas en el lenguaje WSMML, este framework tiene una herramienta que ejecuta estas composiciones, denomina *Web Service Modelling eXecution environment (WSMX)*<sup>20</sup>, que permite generar y ejecutar la orquestación y coreografía de los servicios. También utilizan *ATLAS*<sup>21</sup> *Transformation Language (ATL)* para traducir del modelo de características al WSMML. En [32]

*Feature Plug-in*<sup>22</sup> y [38] *FeatureIDE*<sup>23</sup> son utilizadas para especificar el modelo de características.

5) *Criterio Gestión de la Calidad en el Método: propiedades de calidad*

*Propiedades de calidad:* La calidad en general es muy poco abordada en la LPSOS y tratada tarde en la etapa de diseño, solo dos estudios [3][6] trataron este aspecto de forma más precisa. En [6] se indica que en la fase de diseño del dominio se debe incluir la especificación de las propiedades no funcionales, debido a que los NFRs están entrelazados y relacionados con los requisitos funcionales, y que puede haber varios servicios que proporcionan la misma funcionalidad, pero que difieren de la implementación de su QoS. Para ello, las características en el modelo de características tipo FODA [46] son etiquetadas con los rangos de calidad que serán soportados por la arquitectura de la línea de productos, ayudando al ingeniero de servicios y de software a evaluar el impacto de las características variantes seleccionadas acorde a las características de calidad que los servicios proveen. Por su parte [3] presenta una técnica para identificar servicios a partir de los atributos de calidad, analizando escenarios de atributos de calidad para identificar los servicios que ayuden al cumplimiento de los atributos de calidad arquitectural, para ello, dispone de un conjunto de patrones de diseño SOA que permiten satisfacer algunos atributos de calidad. Ninguna propuesta utiliza un estándar para especificar la calidad.

6) *Criterio Validación del Método: madurez, calidad de la arquitectura*

*Madurez:* Todas las propuestas analizadas presentaban un caso de estudio, donde se evaluaba su factibilidad.

*Calidad de la arquitectura:* Solo dos propuestas evalúan la calidad arquitectural: [6] lo hace desde el punto de vista de los usuarios finales del método, mediante la evaluación del desempeño y escalabilidad del método; [3] presenta métricas para medir la cohesión y el acoplamiento entre los componentes a nivel arquitectural.

## B. Con Respecto a Métodos para SOA (ver Tabla II)

### 1) Criterio Concepto de Servicio: negocio, WS

En términos generales la mayoría de los métodos abordan ambas definiciones: servicios de negocios y WS, primero identifican cuales son los servicios necesarios para dar soporte a las funcionalidades del negocio y luego identifican cuales WS son requeridos.

### 2) Criterio Estrategia de realización o desarrollo: top-down, bottom-up, middle-out

“Top-down” fue la más utilizada, con un 76,2% del total de estudios analizados, seguido del “bottom-up” con un 23,8%; en ninguno de los métodos se abordó un enfoque híbrido.

### 3) Criterio Cobertura del ciclo de vida: Identificación, Especialización, Realización

Solo un estudio [45] abarcó en su análisis el ciclo de vida completo SOA, el resto de los estudios solo se enfocan en la presentación de propuestas para la fase de identificación de los servicios.

<sup>18</sup> Architecture Description Language

<sup>19</sup> <http://featuremapper.org>

<sup>20</sup> <http://www.wsmx.org>

<sup>21</sup> <https://eclipse.org/atl>

<sup>22</sup> <http://gsd.uwaterloo.ca/fmp>

<sup>23</sup> [http://www.witi.cs.uni-magdeburg.de/iti\\_db/research/featureide](http://www.witi.cs.uni-magdeburg.de/iti_db/research/featureide)

#### 4) Criterio Grado de granularidad del análisis: Bajo, Moderado, Alto

El nivel de abstracción en general es alto, la identificación de los servicios es la actividad más realizada en los estudios [3][20][36][38][40], la especificación es muy poco abordada y la realización casi inexistente. Sin embargo, dos estudios ameritan mencionarse: en [20], en el paradigma SOA, las actividades en un grupo de procesos de negocio pueden ser implementadas por servicios o delegadas a los usuarios. Cada actividad, puede tener uno o más servicios que pueden realizarla. Los servicios pueden ser comunes y reutilizados entre diversos procesos de negocios. Diferentes servicios pueden proporcionar funcionalidad del negocio para soportar distintos requisitos tecnológicos, como la comunicación a través de diversos medios de transporte. Además, pueden proporcionar funcionalidad del negocio con diferentes propiedades no funcionales, como seguridad y rendimiento, expuestos por diferentes implementaciones de interfaces de servicios y composición de servicios. En [45], la variabilidad de una LPS es realizada a nivel de los servicios y componentes, expresados en una AR. Ninguno de los trabajos revisados aborda tener una AR y luego adaptarla a servicios.

#### 5) Criterio Accesibilidad y validez: Documentación no accesible, parcialmente accesible, bien accesible

Todos los métodos analizados presentan una buena documentación y presentan un caso de estudio, donde se detallaba las actividades a seguir y se evaluaba su validez.

### C. Con Respecto a Métodos para AR-SOA (ver Tabla III)

#### 1) Criterio Publicación del servicio: directamente a consumidores, por intermediarios

Este criterio no fue abordado en los métodos analizados.

#### 2) Criterio Composición de los servicios: orquestación, coreografía

La mayoría de las propuestas presentan como alternativa para la composición de los servicios la orquestación, implementada a través de los procesos de negocios, quienes coordinan el llamado a los servicios.

#### 3) Criterio Calidad del Servicio: cumplimiento de requisitos no funcionales de alto nivel y de bajo nivel

Ninguno de los métodos evalúa la calidad de servicio a un alto nivel de abstracción en las arquitecturas presentadas, como por ejemplo desempeño, escalabilidad, interoperabilidad.

## VI. CONCLUSIONES

La principal contribución de este trabajo fue realizar una RDS para presentar una perspectiva detallada sobre las fases, actividades, artefactos y roles que participan en un proceso de desarrollo para LPSOS; en la literatura no se encontraron RDS similares. Aunque los beneficios de la orientación a servicios son frecuentes en la literatura, la revisión, análisis y evaluación de los 21 métodos presentados en esta RDS ha demostrado que falta un enfoque integral que incluya en las fases de un ciclo de vida ID de LPSOS, la consideración de las propiedades de calidad, para la identificación, especificación y realización de los servicios como componentes arquitecturales representados mediante una AROS-LPS, que muestre la clara trazabilidad entre los componentes de servicios funcionales y no funcionales. La gestión de calidad en los métodos fue muy poco abordado, y ningún estándar para especificar las

propiedades de calidad fue considerado. Por su parte, la gestión de la variabilidad es clave en un contexto: SOA y LPS. En ambos casos, los puntos de variación deben ser implementados ya sea por un único servicio (donde una interfaz de servicios puede ofrecer parametrización) o a través de servicios similares que traten cada variación; estas variaciones pueden estar influenciadas por las tecnologías seleccionadas para satisfacer propiedades de calidad requeridas para alcanzar los objetivos empresariales, representados como componentes de servicios que resuelven funcionalidades. Los métodos que aplican la gestión de procesos de negocio enfatizan el desarrollo de un proceso de negocio, y a partir de él, se derivan las variantes, especializadas según las necesidades de la organización. Los procesos de negocio pueden tener algunas partes comunes para un grupo de diferentes casos de aplicación, donde implica cierta variabilidad en la selección de las actividades de un proceso de negocio. La RDS presentada provee a la comunidad de desarrolladores de LPSOS la visión general de métodos actuales para tomar decisiones en cuanto a la definición de un método general para el diseño de una AROS-LPS. La principal limitación del estudio realizado es que la RDS no es una comparación para determinar quién de los métodos de desarrollo es el mejor, la intención es de describir como los autores abordan el proceso de desarrollo de una AROS-LPS. Una perspectiva a ser considerada sería evaluar el alcance de la AROS-LPS respecto a la calidad y número de productos que se pueden derivar a partir de ella, para una familia de LPSOS determinada.

## AGRADECIMIENTOS

Al CDCH-UCV (Consejo de Desarrollo Científico y Humanístico), proyecto DARGRAF PG 03-8730-2013-2.

## REFERENCIAS

- [1] P. Istoan, G. Nain, G. Perrouin, and J. Jezequel, *Dynamic Software Product Lines for Service-Based Systems*, 9th IEEE International Conference on Computer and Information Technology (CIT'09), Xiamen, China, October 2009.
- [2] P. Istoan, J. M. Jézéquel, and G. Perrouin, *Software Product Lines for Creating Service-Oriented Applications*, Ph.D. thesis, Irista Rennes Research Institute, Rennes, France, June 2009.
- [3] F. M. Medeiros, *SOPLE-DE: an Approach to Design Service-Oriented Product Line Architectures*, Master. thesis, Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife, Brasil, 2010.
- [4] S. Apel, D. Batory, C. Kästner, and G. Saake, *Feature-Oriented Software Product Lines*, Springer, 2013.
- [5] Q. Munir and M. Shahid, *Software Product Line: Survey of Tools*, Master. thesis, Department of Computer and Information Science, Linköping University, Linköping, Sweden, 2010.
- [6] B. Mohabbati, M. Asadi, D. Gašević, and J. Lee, *Software Product Line Engineering to Develop Variant-Rich Web Services*, In *Web Services Foundations*, Springer, New York, 2014.
- [7] R. Dos Santos and M. Fantinato, *The Use of Software Product Lines for Business Process Management: A Systematic Literature Review*, *Information and Software Technology* vol. 55, no. 8, pp. 1355-1373, 2013.
- [8] K. Pohl, G. Bockle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*, Springer, 2005.
- [9] World Wide Web Consortium, *Web Services Architecture Requirements*, W3C Working Group Note, February 2004.
- [10] J. Nicolai, *SOA in Practice: the Art of Distributed System Design*, O'Reilly: Sebastopol, CA, 2007.
- [11] M. Galster, P. Avgeriou, and D. Tofan, *Constraints for the Design of Variability-Intensive Service-Oriented Reference Architectures – An*

- Industrial Case Study*, Information and Software Technology, vol. 55, no. 2, pp. 428-441, 2013.
- [12] M. Rosen, B. Lublinsky, K. Smith, and M. Balcer, *Applied SOA: Service-Oriented Architecture and Design Strategies*, Wiley & Sons, 2008.
- [13] S. Cohen and R. Krut, *Managing Variation in Services in a Software Product Line Context* (No. CMU/SEI-2010-TN-007), Carnegie-Mellon University, Software Engineering Institute, Pittsburgh, PA, USA, 2010.
- [14] G. Kotonya, J. Lee, and D. Robinson, *A Consumer-Centred Approach for Service-Oriented Product Line Development*, In proceedings of Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA 2009), pp. 211-220, Cambridge, UK, September 2009.
- [15] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.
- [16] B. Kitchenham, *Guidelines for Performing Systematic Literature Reviews in Software Engineering, Version 2.3*, EBSE Technical Report, Software Engineering Group, School of Computer Science and Mathematics, Keele University, UK and Department of Computer Science, University of Durham, UK, 2007.
- [17] M. Matinlassi, *Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, Kobra and QADA*, In Proceedings of the 26th International Conference on Software Engineering (ICSE'04), pp. 127-136, Edinburgh, UK, May 2004.
- [18] T. Kohlborn, A. Korhous, T. Chan, and M. Rosemann, *Identification and Analysis of Business and Software Services - a Consolidated Approach*, Services Computing, IEEE Transactions, vol. 2, no. 1, pp. 50-64, 2009.
- [19] L. B. R. de Oliveira, K. R. Felizardo, D. Feitosa, and E. Y. Nakagawa, *Reference Models and Reference Architectures Based on Service-Oriented Architecture: a Systematic Review*, In Software Architecture, pp. 360-367, Springer Berlin Heidelberg, 2010.
- [20] M. Asadi, B. Mohabbati, N. Kaviani, D. Gašević, M. Bošković, and M. Hatala, *Model-driven Development of Families of Service-Oriented Architectures*, In Proceedings the 1th International Workshop on Feature-Oriented Software Development (FOSD'09), pp. 95-102, Denver, Colorado, USA, October 2009.
- [21] Object Management Group (OMG). *Business Process Model and Notation (BPMN), Version 2.0*, 2011.
- [22] B. List and B. Korherr, *An Evaluation of Conceptual Business Process Modelling Languages*, In Proceedings of the 2006 ACM symposium on Applied computing (SAC'06), pp. 1532-1539, Dijon, France, April 2006.
- [23] J. Herrera, F. Losavio, and A. Matteo, *RDS de Enfoques y Técnicas para la Construcción de Arquitecturas en un Contexto de Líneas de Productos de Software*, Revista Venezolana de Ciencias de la Computación ReVeCom, vol. 1, no. 1, pp. 17-25, Junio 2014.
- [24] S. Günther and T. Berger, *Service-Oriented Product Lines: Towards a Development Process and Feature Management Model for Web Services*, In SPLC, vol. 2, pp. 131-136, October 2008.
- [25] P. Kruchten, *Architectural blueprints—The “4+1” View Model of Software Architecture*, In proceedings of the Conference on TRI-Ada'95, Anaheim, CA, USA, November 1995.
- [26] H. J. González, *Integration of Quality Attributes in Software Product Line Development*, Tesina de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información (ISMFSI), 2012.
- [27] A. Arsanjani, *Service-Oriented Modeling and Architecture: How to Identify, Specify, and Realize Services for your SOA*, IBM Software Group, November 2004.
- [28] S. Svanidzaitė, *A Comparison of SOA Methodologies Analysis & Design Phases*, In Tenth International Baltic Conference on Databases and Information Systems (Baltic DB & IS 2012), Vilnius, Lithuania, July 2012.
- [29] H. G. B. Ribeiro, S. R. de Lemos Meira, E. S. de Almeida, D. Lucredio, A. Alvaro, V. Alves, and V. Garcia, *An Assessment on Technologies for Implementing Core Assets in Service-Oriented Product Lines*, In 4th Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS'10), pp. 90-99, Bahia, Brazil, September 2010.
- [30] J. P. Miguel, D. Mauricio, and G. Rodríguez, *A Review of Software Quality Models for the Evaluation of Software Products*, International Journal of Software Engineering & Applications (IJSEA), vol. 5, no. 6, November 2014.
- [31] M. Mohammadi and M. Mukhtar, *A Review of SOA Modeling Approaches for Enterprise Information Systems*, Procedia Technology, vol. 11, pp. 794-800, 2013.
- [32] F. Zaupa, I. M. de Souza Gimenes, D. D. Cowan, P. S. Alencar, and C. J. P. de Lucena, *A Service-Oriented Process to Develop Web Applications*, Journal UCS, vol. 14, no. 8, pp. 1368-1387, 2008.
- [33] M. Galster and A. Eberlein, *Identifying Potential Core Assets in Service-Based Systems to Support the Transition to Service-Oriented Product Lines*, In 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems In Engineering of Computer Based Systems (ECBS), pp. 179-186, Las Vegas, Nevada, USA, April 2011.
- [34] M. H. ter Beek, S. Gnesi, A. Fantechi, and J. L. Fiadeiro, *Variability and Rigour in Service Computing Engineering*, In 35th Annual IEEE Software Engineering Workshop (SEW), pp. 122-127, Limerick, Ireland, June 2011.
- [35] A. Ezenwoke, S. Misra, and M. O. Adigun, *An Approach for E-Commerce On-Demand Service-Oriented Product Line Development*, Acta Polytechnica Hungarica, vol. 10, no. 2, pp. 69-87, 2013.
- [36] M. Bošković, D. Gašević, B. Mohabbati, M. Asadi, M. Hatala, N. Kaviani, J. Rusk, and E. Bagheri, *Developing Families of Software Services: a Semantic Web Approach*, Journal of Research & Practice in Information Technology, vol. 43, no. 3, 2011.
- [37] N. C. Das, S. Ripon, O. Hossain, and M. S. Uddin, *Requirement Analysis of Product Line Based Semantic Web Services*, Lecture Notes on Software Engineering, vol. 2, no. 3, p. 210, 2014.
- [38] C. Parra and D. Joya, *SPLIT: an Automated Approach for Enterprise Product Line Adoption Through SOA*, Journal of Internet Services and Information Security (JISIS), vol. 5, no. 1, pp. 29-52, 2015.
- [39] H. Serajzadeh and F. Shams, *An Approach for Discovering Services for a Service-Oriented Product Line*, Global Journal on Technology, vol. 1, 2012.
- [40] M. Abu-Matar and H. Gomaa, *Feature Based Variability for Service Oriented Architectures*, In WICSA'11 Proceedings of the 2011 Ninth Working IEEE/IFIP Conference on Software Architecture, pp. 302-309, Boulder, Colorado, USA, June 2011.
- [41] N. Boffoli, D. Caivano, D. Castelluccia, F. M. Maggi, and G. Visaggio, *Business Process Lines to Develop Service-Oriented Architectures Through the Software Product Lines Paradigm*, In SPLC, vol. 2, pp. 143-147, 2008.
- [42] J. Lee, D. Muthig, and M. Naab, *A Feature-Oriented Approach for Developing Reusable Product Line Assets of Service-Based Systems*, Journal of Systems and Software, vol. 83, no. 7, pp. 1123-1136, 2010.
- [43] M. Galster, *Describing Variability in Service-Oriented Software Product Lines*, In Proceedings of the 4th European Conference on Software Architecture (ECSA'10), pp. 344-350, Copenhagen, Denmark, August 2010.
- [44] M. Njima, M. H. Ter Beek, and S. Gnesi, *Product Line Architectures for SOA*, In Proceedings of the 11th Conference on Software Engineering Research and Practice (SERP'11), Las Vegas, Nevada, USA, July 2011.
- [45] I. Achour, L. Labeled, R. Helali, and H. B. Ghazela, *A Service Oriented Product Line Architecture for E-Government*, The 2011 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE'11), Las Vegas, Nevada, USA, July 2011.
- [46] K. Lee, K. Kang, and J. Lee, *Concepts and Guidelines of Feature Modeling for Product Line Software Engineering*, Lecture Notes in Computer Science, vol. 2319, pp. 62-77, 2002.



## ***Web-services reference architecture for software product lines: A quality-driven approach***

### **Arquitectura de referencia de servicios web para líneas de productos de software: Un enfoque dirigido por la calidad**

**Juan Carlos Herrera<sup>1</sup>, Francisca Losavio<sup>2</sup>, Oscar Ordaz<sup>3</sup>**

<sup>1</sup>Universidad Bolivariana de Venezuela. Caracas, Venezuela. [jchr1982\(AT\)gmail.com](mailto:jchr1982(AT)gmail.com)

<sup>2</sup>Universidad Central de Venezuela. Caracas, Venezuela. [francislosavio\(AT\)gmail.com](mailto:francislosavio(AT)gmail.com)

<sup>3</sup>Universidad Central de Venezuela. Caracas, Venezuela. [oscarordaz55\(AT\)gmail.com](mailto:oscarordaz55(AT)gmail.com)

Artículo de Investigación

*Recibido:* 22-09-2016

*Revisado:* 08-11-2016

*Aceptado:* 19-11-2016

#### **Abstract**

*Software Products Lines (SPL) and Service-Oriented Architecture (SOA) are two approaches for software development used in industrial practice favouring reuse of existing assets and capabilities, rather than redevelop new systems. The main goal of this work is to exploit benefits from both approaches and integrate them into a unique architectural design process WSRA-SPL (Web-Services Reference Architecture for SPL), centred on software quality assurance in early steps of the software development lifecycle.*

*Keywords: Software Product Lines; Service-oriented Architecture; Web Services Reference Architecture; Software Product Quality; Healthcare Integrated Information Systems.*

#### **Resumen**

Las líneas de Productos de Software (LPS) y las Arquitecturas Orientadas a Servicios, (del inglés SOA), son dos enfoques para el desarrollo de software que favorecen la reutilización, en lugar de re-desarrollar nuevos sistemas. El objetivo principal del trabajo es explotar los beneficios de ambos enfoques e integrarlos en un único proceso de diseño arquitectural para SPL, denominado WSRA-SPL, (del inglés Web-Services Reference Architecture for SPL), centrado en asegurar la calidad del software en etapas tempranas del ciclo de vida del desarrollo del software.

Palabras clave: Líneas de Productos de Software; Arquitectura Orientada a Servicios; Arquitectura de Referencia de Servicios Web; Calidad del Producto de Software; Sistemas de Información Integrados de Salud.

© 2016. IAI All rights reserved



## 1. Introduction

*Software products lines (SPL)* and *Service-Oriented Architecture (SOA)* are approaches for software development used in industrial practice favouring reuse of existing assets and capabilities, rather than redevelop to construct new systems. The main goal of this work is to exploit benefits from both approaches and integrate them into a unique architectural design process, *Web-Services Reference Architecture for SPL (WSRA-SPL)*, centered on early software quality assurance.

The term SOA emerged in [1] as the approach of building loosely coupled distributed systems with minimal shared understanding among system components, and was defined by W3C [2]; it is a way of designing systems composed of services or *service-oriented systems (SOS)*, invoked in a standard way [30]; SOA is a *reference model* of a business enterprise architecture, where business functionality or application logic is made available as shared, reusable services on a network [9]. We will speak indistinctly of *services* or *Web services (WS)*, even if implementations may be different, following a loosely coupled, synchronous or asynchronous, message-based communication model. SOA's main goal is to guarantee transparent, flexible and dynamic interaction of provided/consumed services, over multiple interconnected domains. A *Domain* denotes any aspect where computing can be applied [35].

On the other hand, *SPL* is a set of software-intensive systems sharing a common, managed set of features that satisfy specific needs of a particular market segment or mission, developed from a common set of core assets in a prescribed way [4]. Reusable features or assets are imbedded into a software architecture [19] called *Reference Architecture (RA)*, which is an instantiable schema, where reusability is achieved combining the assets according to the schema, to derive different products of the SPL family [4, 5]. The SPL development model or *SPL Engineering (SPLE)* [6, 7], considers two main lifecycles, *Domain Engineering (DE)* where RA is constructed, and *Application Engineering (AE)* where concrete product configurations are derived from RA.

While SPL aims at identifying variability and commonalities in a product family developed by one producer alone, SOA aims at modularize and build independent services developed by various companies that, when combined, form complex SOS [8, 9]. Services variability management must be handled [3, 8, 31, 33] to integrate SOA and SPL. While goals are similar in both approaches, they lead to different variation needs; both types of systems must meet quality attributes that may vary in SPL or SOA.

A *Web-Service Reference Architecture (WSRA)* for SPL is conformed by architectural components (the services) and connectors (the WS messaging exchange mechanisms). WSRA is built over the SOA model supporting *maintainability (modifiability-extensibility, reusability), reliability, security, scalability, and consistency-correctness*. *Interoperability* is provided by services standard interfaces. However, we have to distinguish between the *SOA Reference Model (Enterprise Architecture)*, and the *SPL*

*Reference Architecture*, where built-in placeholders, constituting the variability model [6], should be present to allow the SPL RA instantiation.

This paper, besides the Introduction and the Conclusion, is structured as follows: the second section describes the work context; the third section discusses related works on the research topic; the fourth section outlines main activities of three WSRA design methods; the fifth section presents the WSRA-SPL process; the sixth section treats the application of WSRA-SPL to the HIS domain, and finally the seventh section discusses the main results obtained from this research.

## 2. Context

### 2.1 Robust Programming, Web Services, and QoS

According to the quality-driven SPLE approach [10], each high-level architectural component solving a system functionality or functional requirement (FR), has associated required quality properties or non-functional requirements (NFR), to work conveniently, i.e., to be functionally suitable [11]. The component implemented by a service must satisfy the required high-level quality goals. For a robust programming style [12, 13], quality must be taken into account early, as a built-in capability of the SPL RA. On the other hand, in WS programming, QoS (Quality of Service) represent low-level NFR concerning the service behaviour, related mostly with performance and security, expressed as quality attributes or measurable elements [11, 15]. QoS are specified in the standard interface of the service, separated from its implementation, written in XML or WSDL (Web Service Description Language) [2], to achieve service interoperability. A QoS is defined as a set of qualities related to the collective behaviours of one or more objects. Services can implement complex functional components (composite services), but they must also fulfil specific quality goals, as any other functionality, besides their QoS.

The following major QoS requirements from ISO/IEC 13236 [15], are mapped to the standard terminology of ISO/IEC 25010 [11], specifying high-level software quality properties:

*Performance efficiency (time-behaviour* – measures: *time units, capacity-scalability* – measures: *throughput, ratio, rate: e.g. bit/sec, instructions/sec*), *Functional suitability (correctness-accuracy-integrity-coherence-consistency, related to transmission errors, e.g. resilience or recognition, delivery, recovery, etc.* – measures: *probability*), *Security (confidentiality-protection-data protection, authenticity-access control* – measures: *probability, value or level*), *Reliability (availability-channel-connection-processing* – measures: *MTBF, MTTR, range 0-1 over a time period, e.g. 0.99 over 30 days*).

*WS interoperability* is achieved by the standard WSDL interface, as it was mentioned; it is not a QoS, but a WS high-level quality property. QoS must not be confused with the high-level quality characteristics [11] that the service must accomplish as a software component. Some works have been done towards the modification of the WSDL service interface [16, 17], to include high-level quality

properties that the service will provide; however, this solution attempts against service interoperability, breaking the standard interface design of the service [2].

## 2.2 Web Services Architecture (WSA) and SOA model

A *software architecture* is classically defined by a configuration of architectural elements, components, connectors, and holding a specific behaviour, to achieve a desired set of properties [19]; several views are proposed to represent an architectural configuration, including this classic logic view [18]. An architecture based on WS, called *Web Services Architecture (WSA)*, is expressed in [2] by multiple views or models, such as messages, services, resources, and policy. However, a logic view of WSA considering services as architectural components and their connections (by messages, protocols, etc.) is still missing, and composability could be facilitated using this logic view [3]. The SOA reference model (Figures 1, 2) shows Services and Components Layers; services from Services Layer are mapped to software components, but how this is done and how they are organized is not detailed in the literature.

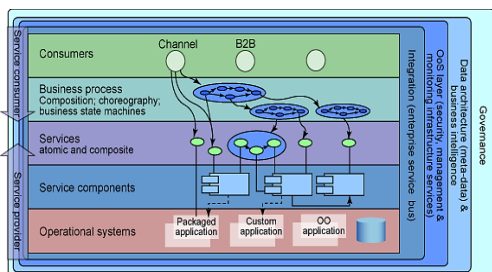


Figure 1. Layers of the SOA Reference Model; source [22]

SOA follows an event-based architectural style [19], establishing distribution and communication through a network using a client-server model; applications are decomposed into managed networks of interoperating services, which can be implemented in a variety of technologies, being WS the most popular one; using SOA does not mean to have a “full” WSA [34].

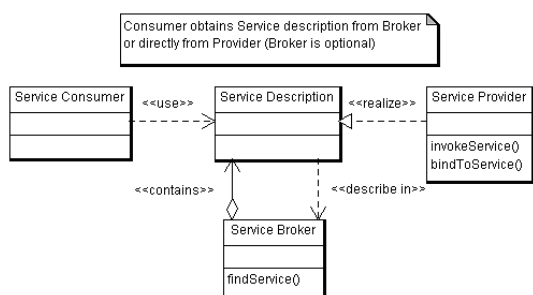


Figure 2. SOA Services Layer using ArgoUML [21]

## 2.3 The HIS-Domain

HIS are software intensive systems [27], located in different and distant institutions, with mandatory NFR *interoperability, availability and security. Electronic Health Records (EHR)* management and sharing is a basic functionality, considering transparent information exchange between two or more systems [32]. The use of standards, such as HL7 [34] for EHR, and LOINC, DICOM for imaging, is mandatory [27, 34] to have interoperability. The HIS architecture holds the hybrid style shown in

Figure 3 [10, 19]: Event-based/Layers style, following a Client-Server model for distribution.

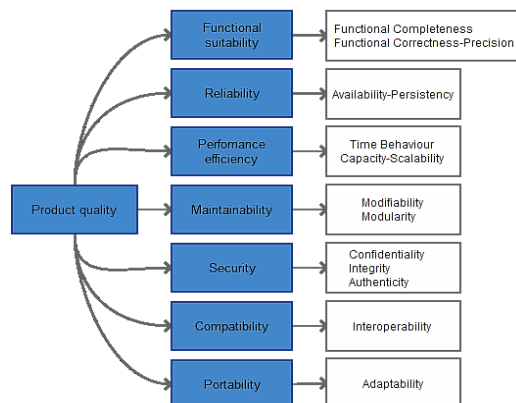


Figure 3. HIS hybrid Event-based/Layers/Client-Server Style (Wikipedia)

Patient-doctor’s episodes with general medical practice and EHR management, billing, and laboratory/imaging services are HIS mail functionalities treated here; 3 open source widely used market products were studied: Open EMS, PatientOS, and Care2x [10, 27] to extract an initial architecture.

## 2.4 HIS Domain Quality Model (HIS-DQM)

The Quality Model of ISO/IEC 25010 [11] specifies eight high-level quality properties of a software product. Its adaptation to the HIS domain (HIS-DQM) is shown in Figure 4. It represents the global quality of each product or system in the HIS SPL family.

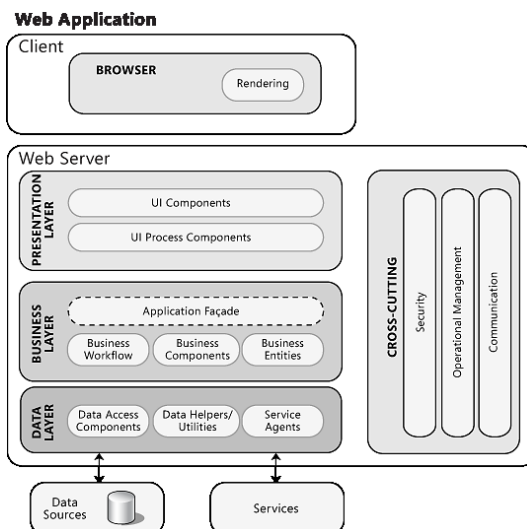


Figure 4. HIS Quality Model (HIS-DQM) [14]

## 3. Related works

Works relevant to our research are discussed in this section. Tables 1, 2, and 3 are organized by main research topics found in the literature: - SOA, WS and QoS generalities, - Services composition and variability management in SOA/SPL, and finally - Reference Architecture Design Process for SOA/SPL; the authors’ names initials and year of the publication are shown in brackets for legibility purposes to distinguish each work, with the number of the reference that can be found in the Reference section at the end of this paper.

**Table 1: SOA, WS, and QoS generalities**

Reference	Description of research topic	Discussion
<i>Topic: Generalities on SOA, WS, and QoS</i>		
[2] [W3C] [16] [RS 09]	WS are based on a collection of standards and protocols to make processing requests to remote systems by speaking a common, non-proprietary language, using common transport protocols (http, SMTP, etc.); they provide functional and business operations. Due to the dynamic and unpredictable nature of the Web, providing the acceptable QoS is really a challenging task.	We cited the work of [17] on this topic, considering that an attempt to include high-level quality characteristics in the WSDL standard interface specification violates service interoperability.
[15] ISO/IEC 13236 [ISO/IEC 98]	It is an ISO/IEC standard QoS framework, structured as a collection of concepts and their relationships. It is intended to assist in the designing and specifying SOS based on IT. QoS are grouped according to quality characteristics, and general metrics are provided: time-related, coherence, integrity, safety, security, reliability, precedence.	This framework will be used to decide on the selection of architectural solutions in the AE lifecycle, to derive SPL products from the WSRA, which is not discussed in this work; however, a correspondence with high-level and low-level quality terminology in ISO/IEC 25010 and ISO/IEC 13236, respectively, is established.

**Table 2: Services composition, variability management in SOA/SPL**

Reference	Description of research topic	Discussion
<i>Topic: Services composition, variability management in SOA&amp;SPL</i>		
[3] [Mil 06]	It describes SOA; the topic on services composition (orchestration and choreography) is discussed; <i>orchestration</i> uses traditional programming techniques and methods to 'program' services into different paths of execution using flow control commands and has been addressed by BPEL. <i>Choreography</i> , on the other hand, is a process of building a new service out of existing ones, for which we want to guarantee how it will behave with respect to FR (what it will deliver) and NFR (how and under which conditions it will deliver). However, since there is no support for NFR, any serious attempt to compose Web Services automatically based only on WSDL properties is very difficult, if not impossible. The variability problem in SOA is also discussed: knowing QoS properties for a given service, it is possible to manage different compositions, however, a definition of WS-Architecture is missing; <i>composability</i> could be achieved as a property of the architecture.	Choreography is the service composition we are using in this work, to achieve a logical view of the WSRA architectural configuration: architectural components are composite (or simple) services and connectors are the usual messaging exchange between services.
[8] [CL 15]	Differences between SOA and SPL are discussed; variability is also touched: business variability refers to the different reusable functionalities that are tightly coupled with specific business (Feature Model). Framework variability refers to the different functionalities that support the business features. Platform variability refers to the requirements of the platform where the software products are executed. The goal of this level is to allow architects to select business and functional features, and validate whether such features are supported by a specific platform; on these basis, an architecture metamodel is defined, where a component is a single unit of composition and is linked to one feature in the variability model. It also offers a set of Services and requires a set of References. Both Services and References are connection points between components. A connection point also defines a Contract, which represents the interface, provided or required, depending on the type of connection point.	We use the variation point notion [6], which groups a set of services performing similar tasks as variants; the feature model [29] is not used, since our WSRA has imbedded the variability model, and optional and mandatory issues are considered in the EQM table (see Table 6).
[31] [CK 10]	It examines existing SOA and SPL approaches for variation management, focusing on their possible combination, with two objectives: 1) for SOS (Software Oriented Systems) development, to present an approach for managing variation by identifying and designing services explicitly targeted to multiple SOS, 2) for SPL systems, to present an approach for managing variation where services are mechanisms for variation within an SPL or to expand the SPL scope. Mutual benefits of combining systematic reuse approaches from SPL development with flexible approaches for implementing business processes within SOA. Under a combined SOA-SPL approach, developers build core assets, including services, and construct systems through the systematic reuse of these core assets in a predefined way. SPL approaches exploit commonality across products, applying planned variation among their core assets. SOA provides a flexible mechanism for dealing with variation through services that are not bound to a specific product. In addition, an SPL may be composed of SOS, if services are seen as a basic element of composition and provide support for variation among members of the product family. General differences between SOA and SPL are outlined. They state that SOA can be used as a variation mechanism for SPL; service may have built-in variation points that are accessible through parameterized service calls, or the service registry may identify variations among related service components that may include both the newly packaged capabilities and existing services from the enterprise. Their approach is applied to the HIS domain, focusing on health information exchange. The Service Migration and Reuse Technique (SMART) [30] is used in [31], but details are not provided.	This report is focusing on the specific goal of our paper. Variability management is a key issue in SPL to design a RA; however, it is not so for SOA, which is an enterprise architecture. Actually, we use services variability at a high architectural abstraction level, depending on which architectural solution (mechanism) they can provide; low-level QoS variability can be used to determine the best available mechanism in terms of the values of the parameters involved in the service standard WSDL interface providing the mechanism.
[33] [GZ 14]	Besides the claimed benefits of SOA and SPL development approaches, they have limitations w.r.t. platform dependencies for SPL and variability management for SOA. The work presents an SDR (Systematic Literature Review) on the subject of service variability management, considering 9810 works in the period 2008-2010; 10 works were selected for discussion; however no outstanding results were provided to specify in detail a systematic process to handle services variability, to be integrated into SPL RA design. Moreover, their analysis is poor, techniques used are not detailed. A combination of SOA and SPL is proposed, where variability is handled categorizing services, and using logic languages; no details are provided.	In our case, we handle services variability in the WSRA variability model; <<vp>> group variants, which are simple or composite services solving specific quality requirements demanded by functional components which are considered composite services.

**Table 3:** Reference Architecture Design Process for SOA/SPL

Reference	Description of research topic	Discussion
<i>Topic: SOA/SPL approaches for a Reference Architecture design</i>		
[14] [HLO 16]	A systematic and repeatable process, called QuaDRA: Quality-oriented Design of Reference Architecture for SPL is presented; the new ISO/IEC 26550 [23] standard defining the SPLE Reference Model is followed. This process enhances software quality assurance to guarantee an evolutionary SPL. It starts with the SPL Scoping initial phase, to reduce effort in the subsequent steps of the Domain Engineering lifecycle	Our WSRA-SPL process is based on an adaptation of QuaDRA, combined with SOMA, to develop a service-oriented RA for SPL, called WSRA, which is the main goal of this work
[34] [OMG-HL7 08]	SOA is an architectural style (as is client/server, hub-and-spoke, etc.) that may be realised in different implementation forms. SOA as an enterprise architecture, must not be confused with the technology platform, which is a technological layer within the architecture. Just because something is implemented using Web Services (WS) does not make it a SOA, and does not itself guarantee interoperability, nor do Web Services: top-down and bottom-up SOA development approaches are discussed; a case study on HIS is presented, detailing steps to achieve a HIS SOA; however, the SPL approach is not discussed.	Our goal is to construct a service-oriented RA for SPL, called WSRA; we consider terms services and Web Services alike in this context.
[20] [MR 12]	It combines SOMA [21] and RUP [18] to design a non-SPL enterprise WSRA for HIS; it is well detailed. Business Processes (BP) and goals (BG) are the design starting points. WS are connected by SOAP (Simple Object Access Protocol) via RPC over HTTP protocols [38]. Activities in BP are aggregated and translated to executable services workflows in BPEL by Intalio. It is a nice and detailed work, but the document legibility is poor because lots of tables are shown and it is difficult to follow the complete process.	The elicitation and specification of NFR as BG it is not clear and high-level quality properties are not specified in BP; they are only treated as QoS by the architect to select the provider component; ATAM (Attribute Trade-off Analysis Method) is used to decide on architectural choices, thus mixing low-level with high-level technics, being in our opinion, a main limitation of this work.
[21] [Ars 04]	SOMA (Service-Oriented Modelling and Architecture) is a methodology to develop SOS; it follows top-down, middle-out and bottom-up approach. Main phases and activities are outlined graphically in the figure below taken from [21] and in Table 1.	We are using SOMA which is for SOS development, in combination with the SPL QuaDRA development process, to construct WSRA in the DE lifecycle of SPLE.

From the literature analysis, three main problems were identified to treat the integration of SPL and SOA software development approaches: 1) the representation of the SPL RA with services, 2) the satisfaction of quality properties required by functionalities to behave properly, and 3) the variability management. Our present work aims to provide a solution to these problems.

#### 4. Design of web service architectures

Table 4 presents a correspondence of three main RA design approaches involving WS development; the complete WSRLA-SPL process will be specified in Section 5, and a quick glance at the process can be appreciated in Figure 5. SOMA input is from business componentization and analysis [24, 25]. A business component is described by the business services it offers. A business service is based on grouping business functionalities by workflows, tasks, activities, often including implicit and explicit rules.

#### 5. WSRA-SPL: web services reference architecture design process for SPL

The WSRA-SPL process is general and can be applied to any domain, see Figure 5 for its SPEM representation.

##### 5.1 Preliminary assumptions

They are used to improve the legibility of the WSRA-SPL process specification presented in Section 5.2 and are outlined in what follows.

We will use the classical software architecture definition [19], components and connectors; components are represented as services and their connections will be the communication between them, i.e. SOAP messages

exchanged via HTTP, HTTPS, SMTP, FTP, etc. through the network. The service, viewed as a component, is an abstract, logical view of systems, modules, programs, databases, etc., defined in terms of what it does, typically carrying out a business level operation or functionality, specified from business processes. Hence a business model should be available.

Execution Language). What is important is that services act together in a composition as a single application [22], and we suppose it to be correct.

In the SPL RA defined in [11, 14], components originating from implicit functionalities providing solutions or low-level mechanisms to satisfy a high-level quality property required by a component solving a system functionality or functional component, are designed as simple services; functional components instead will be designed as composite services; a composite service is then conformed in turn by composite or by simple services. Traceability between composite services and their required quality properties, represented by simple services, is given by the choreography of simple and composite services involved in the functional component design.

The composition of the services from business processes workflows (business processes orchestration), is assumed to be correct, and it is a responsibility of the service bus middleware and/or automatic translation tools, such as Intalio, etc. [20] into executable languages, such as BPEL (Business Process Execution Language).

**Table 4:** Correspondence with SOMA [21], Architectural Design: RUP & SOMA [20], and WSRA-SPL: adapted QuaDRA [14]

	SOMA (SOA) [21]	RUP & SOMA (SOA) [20]	WSRA-SPL: QuaDRA (SPL) [14]
Phases	Main Tasks	Main Activities	Main phases and activities
Identification (of Candidate Services, Business Goals, Components, Business Processes flow)	Domain decomposition (top-down approach)	Analyse business processes (BP) and identify business goals (BG)	<ul style="list-style-type: none"> <li>- <i>PLScop phase</i>- Study of market products; note that this step is performed first in the PL Scoping phase;</li> <li>- construct initial Candidate Architecture (CA); - identify BG from main CA components;</li> <li>- associate BG to Candidates Services (CS);</li> <li>- outline future SPL products as additional BG;</li> <li>- associate these BG to CS;</li> <li>- construct the Domain Quality Model (DQM);</li> <li>- construct the Extended Quality Model Table (EQM), which will constitute the Services Portfolio;</li> <li>- construct the Products Portfolio;</li> </ul>
	Goal-Service Modelling (middle-out approach)	<ul style="list-style-type: none"> <li>- Associate services with functional and non-functional business goals (BG)</li> <li>- Determine CS</li> </ul>	<ul style="list-style-type: none"> <li>- <i>PLScop phase - Domain Modelling</i>:</li> <li>- obtain BP in BPMN, including NFR attached to BP activities (not explicit specified in the BPMN syntax);</li> <li>- identify BG from BP; functional BG are domain main functionalities (extracted from CA); non-functional BG are domain priority quality properties (extracted from DQM);</li> <li>- CS are associated to these BG;</li> </ul>
	Analysis of existing systems (bottom-up approach)	<ul style="list-style-type: none"> <li>- Study existing systems to extract CS</li> <li>- Relate CS with entities in BP</li> <li>- Group entities according to service; each service is related to FR and NFR</li> <li>- Construct Service Portfolio with CS, which is updated if new services are found</li> </ul>	<ul style="list-style-type: none"> <li>- <i>PLScop phase - Domain Modelling (cont.)</i>:</li> <li>- aggregate BP:</li> <li>- establish new BG from BP Aggregations;</li> </ul>
Specification (of Services, Components, Flows)	Services specification	Services composition, identification of FR, NFR and QoS for services in the Portfolio	<ul style="list-style-type: none"> <li>- <i>PLScop phase - Domain Modelling (cont.)</i>:</li> <li>- <i>Service composition</i>:</li> <li>- perform services functional and non-functional design (verify FR and NFR for each service composition from EQM);</li> <li>- associate composite services to BG in the BP Aggregations;</li> <li>- check CS already identified from CA against the new CS, to avoid redundancy;</li> <li>- determine if these new CS are really composite services;</li> <li>- update EQM (UEQM: Service Portfolio);</li> <li>- update CA (UCA)</li> </ul>
	Subsystems analysis	Associate services to architectural components	<ul style="list-style-type: none"> <li>- <i>DRE (Domain Requirements Engineering) phase</i> - Associate services to architectural components:</li> <li>- composite services from the Aggregations and the future SPL services, are main WSRA components, conforming UCA;</li> <li>- establish connections among UCA components to specify traceability from composite to simple services by checking UEQM for consistency;</li> </ul>
	Components specification	Specify architectural components	<ul style="list-style-type: none"> <li>- <i>DD (Domain Design) phase - Variability modelling</i>:</li> <li>- determine the variability model by grouping composite services performing similar tasks into variation points;</li> <li>- Construct WSRA;</li> </ul>
Realization (Decisions)	Realization decisions	Selection of specific available market services or specific services development	It will not be considered here; mechanisms identified during the study of existing products and others required in UCA will be available in the assets repository.

Service composition, called also choreography or orchestration [2] is the process of building a new service out of existing ones, focusing more on the service functionality (choreography) or emphasizing more the messaging exchange (orchestration). The threshold between these two ways of composing and how and when they are used is not quite clear in the literature [2]. In our case, we want to guarantee how the service will behave with respect to functional properties (what it will deliver) and non-functional properties (how and under which conditions it will deliver) [3]; hence from the architectural perspective, we are not concerned with message exchanges, being a responsibility of the service provider and service bus, but instead, in the properties of a

component and the composite services conforming it and their relationships which is a choreography.

Nevertheless, knowing QoS offers a possibility to manage different services composition choices (variability management). For example, in case of a simple service S, whose main purpose (or implicit functionality) is to satisfy data availability-persistency, the choice between replication or mirror with replications API mechanisms, could be done by evaluating Time-behaviour or Capacity attributes, which are specified as QoS parameters in the WSDL interface of S. In the SPL RA case, S would be a variation point [6] for availability, containing the mentioned APIs as variants.

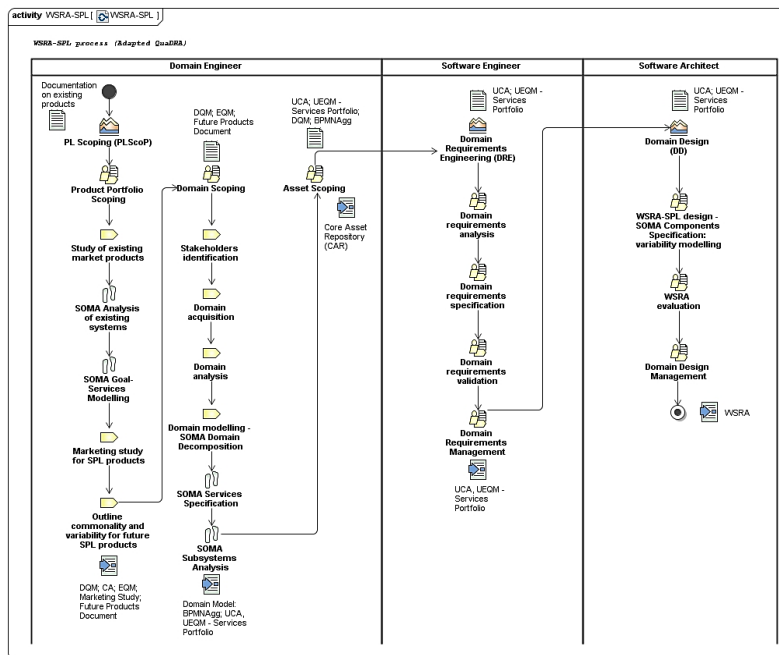


Figure 5. WSRA-SPL process as a glance in SPEM

## 5.2 WSRA-SPL process (Adapted QuaDRA)

The modifications w.r.t. QuaDRA [14] have been outlined in grey in the WSRA-SPL process, presented below in a pseudo-formal notation; WSRA-SPL was shown graphically in Figure 5:

WSRA-SPL

Begin WSRA-SPL

1. PL Scoping phase (PLScoP)

PLScoP phase

begin PLScoP

1. Product Portfolio Scoping - to determine requirements for future SPL WS-based products:

Begin Product Portfolio Scoping

input: documentation on existing products or systems;

1.1 Study of existing market products

1.1.1 SOMA Analysis of existing systems in the domain to infer about SPL WS-based products that can be constructed:

- semantic similarity analysis of products components is made on available documentation;
  - table CCT [10] (Table 5) is constructed;
  - a Candidate Architecture (CA) is automatically built from the product architectural configurations by applying an *automatic extractive bottom-up approach* [10] - this bottom-up technique is applied on an undirected and connected graph structure representing products' architectures;
  - CA is also a graph obtained by the union of the graphs representing the products' architectures (Figure 6); CA is represented in UML 2.0 (<http://www.omg.org/spec/UML/2.0/>) and can also be specified as an ontology;
- 1.1.2 SOMA Goal-Services Modelling:

- CA components represent main functional BG; these components are associated to CS; however, some non-functional BG could also be present in CA, depending on the features extracted from the products studied; they will also be associated to CS;
- construct the domain quality model (DQM) to identify non-functional BG (Figure 4), w.r.t. existing products, by instantiating the ISO/IEC 25010 quality model for the domain [11];
- identify CS: construct the EQM table (Extended Quality Model), see Table 6, from CA and DQM; CA is documented in EQM by listing common and variant components as CS with scenarios of required/provided quality properties and constraints;
- EQM contains this documentation, and it is a different CA view; it conforms the Services Portfolio;

output: CCT; CA; DQM; EQM (Services Portfolio);

notation: CA type: graph, UML diagram or ontology; EQM type: table; DQM type:

table or graphic; CCT type: table;

1.2 Elaborate marketing study for SPL products;

input: marketing information; CA; EQM;

- perform marketing study
- economic factors for the SPL feasibility should be also analysed.

output: Marketing Study

notation: Marketing Study type: document;

1.3 Outline commonality and variability for future SPL products.

input: CA; EQM;

- commonality and variability for future SPL products will be specified by the
- Updated EQM (UEQM) table in step 2.4, see table 6;
- each component of future products' configurations will be described by an

informal document;

output: Future Products Document;

notation: Future Products Document type: text

output: DQM; CA; EQM; Marketing Study; Future Products Document;

end Product Portfolio Scoping;

2. Domain Scoping - adapted from Bjørner's DE [26]

Begin Domain Scoping

input: DQM; EQM; Future Products Document;

begin

2.1 Stakeholders identification:

input: visits; interviews; workshops; questionnaires (specific techniques for each one of these activities should be specified);

- Identify groups of stakeholders with similar interests in the organization requiring the SPL;

output: list of stakeholders' groups

notation: type: list of stakeholders' groups table;

2.2 Domain acquisition:

input: List of Stakeholders' groups;

- capture and gather information from each stakeholder of interest in the domain, into *declarations* to build Domain Description Units (DDU) for each stakeholder viewpoint;

output: DDU;

notation: DDU type: table;

2.3 Domain analysis:

input: DDU;

- analyse DDU, study possible inconsistencies; BP are extracted from DDU [26], and represented as behaviours in the facet specification by intrinsic descriptors from a

stakeholder viewpoint relevant to the domain; the facet specification has three representations: table, UML diagram to represent the table conceptual model, and BPMN (<http://www.omg.org/spec/BPMN/2.0/>) diagrams to represent BP. Behaviours are used to identify BGs; notice that from CA in *Step 1.1*, main functional and eventually some non-functional BG have been already identified and they can be verified against the BPMN BP; additional BG can also appear from this more complete domain specification;

- software quality [11] is included as a new intrinsic facet descriptor.
- it is recommended in [26] to specify first the BP facet. Other facets considered relevant to the domain are also specified in the same way, namely, *Support Technology* and *Rules & Regulations*.

*output:* Facet specification;  
*notation:* Facet specification type: table, UML diagram, BPMN diagram;

## 2.4 Domain modelling - SOMA Domain Decomposition

### 2.4.1 SOMA Services Specification - identification of additional BG;

*input:* Facet specification; CA; EQM; Future Products Document;

A partial Domain Model is obtained from each facet specification; it is represented by BPMN BP:

- integrate the BPMN BP for each stakeholder viewpoint by aggregating activities of the lanes of each BP pool; recall that a BPMN "Aggregation" groups BP activities; these aggregations will be mapped to composite services, and they represent functional BG;

- attach to the respective aggregation, the information on all quality properties for each activity, as a new activity to be performed by the aggregation to accomplish its overall quality; this information corresponds to the composite services high-level quality properties; they represent non-functional BG; notice that they are not QoS, which are considered in the composite service choreography, and that the composition is supposed to be correct;

- obtain a new BPMN aggregation diagram (BPMNAgg) for each stakeholder viewpoint considering aggregations; these are the composite services with their required quality properties;

### 2.4.2 SOMA Subsystems Analysis - in our terms: consider aggregations, update CA, update EQM, map aggregations to composite services

- consider aggregations (composite services) as possible CA new functional components or sub-components; provide communication with the appropriate service in the CA configuration; for example, if the new composite service is the *Diagnostic Assistant system*, it should be decided to connect it to the *EHR Management System* composite service or to have it as a separate composite service, and in this case the *HIS User Interface* should be modified; notice that syntax and semantics of CA elements should be maintained; architectural connections provide/require between two composite services means that they should interact by messaging exchanges, as the usual WS behaviour.

- simple services for solving quality properties related to the aggregation (service composition) should be included as variants of the new components providing their required quality; it means to establish a new connection (communication) between these simple services in CA;

- study EQM, the Future Products Document (see *Step 1.3*), and the BPMNAgg aggregations, to consider if the new composite services found contribute to the SPL future products;

- update CA (UCA) (in all its representations);
- update the EQM (UEQM - Updated Services Portfolio);

*output:* Domain Model: BPMNAgg; UCA, UEQM;  
*notation:* Domain Model type: BPMN; UEQM type: table; UCA type: graph, UML diagram and/or ontology;

*end Domain Scoping;*

### 3. Asset Scoping - to determine the SPL Core Asset Proposal

*Input:* UCA; UEQM; DQM; BPMNAgg;

*Begin Asset Scoping*

- Information for the core asset proposal will be extracted from the input to construct a *Core Asset Repository (CAR)*;

*end Asset Scoping;*

*output:* CAR;  
*notation:* CAR type: database;  
*end PLScop;*

## II. Domain Requirements Engineering (DRE)

*DRE phase*

*begin DRE*

### 1. Domain Requirements Analysis:

*begin DR Analysis*

*input:* UCA; UEQM;

- UCA and UEQM contain precise information on common and variant composite or simple service, including quality issues;

*output:* UCA; UEQM;

*end;*

### 2. Domain Requirements Specification:

*begin DR Specification*

*input:* UCA; UEQM;

- UCA and UEQM contain the complete WSRA requirements specification;

*output:* UCA, UEQM;

*end;*

### 3. Domain Requirements Validation:

*begin DR Validation*

*input:* UCA; UEQM;

- query the UCA ontology, if present, for consistency checking, else another technique should be used;

- update UEQM;

*output:* UCA; UEQM;

*notation:* UCA type: ontology;

*end;*

### 4. Domain Requirements Management

*begin DR Management*

*input:* UCA; UEQM;

- update UCA for any change in requirements;

- update UEQM; step 3 should be applied again to validate changes;

*output:* UCA; UEQM;

*end;*

*end DRE;*

## III. Domain Design (DD)

*DD Phase*

*begin DD*

### 1. WSRA-SPL design - SOMA Components Specification: SPL variability modelling

*begin*

*input:* UCA; UEQM;

- WSRA is conformed by common composite services and variability points [6], constructed by grouping UCA variants (composite or simple services) performing semantically similar tasks;

- update UEQM with the information on variation points.

- WSRA is an undirected, connected graph, specified in UML 2.0 and/or as an ontology;

*output:* WSRA; UEQM;

*notation:* WSRA type: UML diagram and/or ontology;

*end;*

### 2. WSRA evaluation: is a quality assurance technique

*Begin*

*Input:* WSRA

- WSRA is compliant with high-level quality requirements by construction; each WSRA-SPL component (composite or simple service) fulfils the required quality; moreover, WSRA satisfies the global quality requirements of the domain architectural style;

*end;*

### 3. Domain Design Management

*Begin*

*Input:* WSRA

- Requirements traceability is solved by construction;

*output:* WSRA;

*end;*

*end DD;*

*end WSRA-SPL;*

## 6. Application of the WSRA-SPL design process to his domain

WSRA-SPL will be applied step-by step in what follows:

### I. PL Scoping phase (PLScOP)

1. Product Portfolio Scoping - to determine requirements for future SPL products:

#### 1.1 Study of existing market products

##### 1.1.1 SOMA Analysis of existing systems

- CA (Figure 6) and CCT (Table 5) are constructed; CA common components (main functionalities) (CC), are: a1, a2, a3, d1; variant components that are not present in all products are: a4, b4, b5, c2, c3, d2. They are all potential CS and conform EQM.

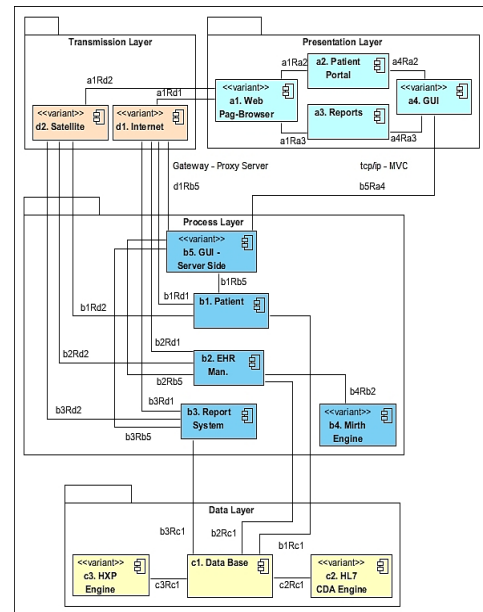
**Table 5:** CCT: Components & Connectors of Market Products Architectures [14]

OpenEMR	PatientOS	Care2x
<i>a. Presentation Layer</i>	<i>a. Present. Layer</i>	<i>a. Present. Layer</i>
<i>Components:</i>	<i>Components:</i>	<i>Components:</i>
a1. Web pages-Browser	a2	a1
a2. Patient Portal	a3	a2
a3. Reports	a4. GUI	a3
<i>Connectors:</i>	<i>Connectors:</i>	<i>Connectors:</i>
a1Ra2	-	a1Ra2
a1Ra3	-	a1Ra3
a1Rd1	-	a1Rd1
a1Rd2	-	a1Rd2
-	a4Ra2	-
-	a4Ra3	-
-	a4Rb5	-
<i>b. Process Layer</i>	<i>b. Business logic</i>	<i>b. Process Layer</i>
<i>Components:</i>	<i>Components:</i>	<i>Components:</i>
b1. Patient	b1. Patient business model	b1
b2. EHR Management	b2	b2
b3. Report System	b3	b3
-	b4. Mirth (HL7 engine)	-
-	b5. GUI-Server Side	-
<i>Connectors:</i>	<i>Connectors:</i>	<i>Connectors:</i>
b1Rc1	b1Rc1	b1Rc1
b1Rd1	X	b1Rd1
b1Rd2	-	b1Rd2
-	b1Rb5	-
b2Rc1	b2Rc1	b2Rc1
b2Rd1	X	b2Rd1
b2Rd2	-	b2Rd2
-	b2Rb4	-
-	b2Rb5	-
b3Rc1	b3Rc1	b3Rc1
b3Rd1	X	b3Rd1
b3Rd2	-	b3Rd2
-	b3Rb5	-
-	b5Rd1	-
<i>c. Data Layer</i>	<i>c. Data Layer</i>	<i>c. Data Layer</i>
<i>Components:</i>	<i>Components:</i>	<i>Components:</i>
c1. Data Base	c1	c1
c2. HL7 CDA Engine	-	-
-	-	c3. HXP Engine
<i>Connectors:</i>	<i>Connectors:</i>	<i>Connectors:</i>
c1Rc2	-	-
-	-	c1Rc3
<i>d. Transmission Layer</i>	<i>d. Transmission Layer</i>	<i>d. Transmission Layer</i>
<i>Components:</i>	<i>Components:</i>	<i>Components:</i>
d1. Internet	d1	d1
d2. Satellite	-	d2

#### 1.1.2 SOMA Goal-Services Modelling: to construct initial Service Portfolio

- DQM (Figure 4) and EQM (Table 6) are constructed: EQM(Service Portfolio) is obtained from DQM, CA and CCT (Table 5); it will be updated (UEQM) later-on in Step 2.2.4.1; it is elaborated in parallel with Steps 1.1.2

and 1.1.3; <<vp>> [6] are also placed in Table 6; however they will be specified in Step 1.2.2.4, but are all shown in Table 6 to abridge the presentation.



**Figure 6.** CA [14]

#### - Candidate Services are identified

CS are identified from main functionalities; we consider the *logical view* of a service [3] as a functionality, and a service as such must satisfy its own high-level quality properties.

New services added to CA are outlined in grey in Table 6; <<vp>> [6] are sets of simple or composite services that are the variants.

#### 1.2 Elaborate marketing study for SPL

The market study is considered within the analysis of the market products, which are supposed to exist in the domain (Table 5), to guarantee the SPL feasibility.

#### 1.3 Outline commonality and variability for future SPL products.

New features for new products in the HIS SPL family are specified. Table 5 shows commonality and variability of the three products studied, and Table 6 describes EHR-HIS functionalities in CA to conform the *Future Products Document*:

- *imaging system (LabImage System)*: imaging laboratory results.
- *diagnostic assistant (Diagnosis)*: on-line diagnosis.
- *on-line appointment services*: schedule on-line patient's appointments, select healthcare institutions and specialists.

#### 2. Domain Scoping

Steps 2.1, 2.2, and 2.3 are detailed in [14]; they define the Domain Model represented by BPMN BP.

#### 2.4 Domain modelling - SOMA Domain Decomposition

##### 2.4.1 SOMA Services Specification - identification of additional BG

Activities in BG are identified (Figure 7) with their required quality properties; BG are functions in BP behaviours:



**Table 6: EQM Table – Services Portfolio**

CA Common Components (CC) & Variation Points <<vp>>	Service	Description	Service High-Level Quality Properties – Priority in (); 1 high, 2 medium, 3 low		Constraints
			Require (s)	Provide (s)	
<i>a5-Portal</i>	<i>Composite service: a5</i>	Portal service: provides access to EHR-HIS functionalities	- Authenticity (1), - Confidentiality (1), - Integrity (1), - Availability-Pers (1), - TimeBehavior (2), - Adaptability (2), - Capacity-Scalability (2), - Interoperability (1), - Usability	- d1 and/or d2       - Web pages' design	Mandatory CC FR       It is not an architectural property, it will not be treated here
<i>b1-Patient</i>	<i>Composite Service composed by services: b18, b19, b20</i>				b1 is mandatory CC FR; b18, b19 and b20 are composed services of b1, and they are mandatory CC services
<i>- b18-AppointService</i>	<i>Composite service: b18</i>	- on-site and/or on-line appointment scheduling	- CorrectPrecision  - Availability-Pers,	- b6  - c7, c9a, c9b	Modules for computation algorithms data base mechanisms for persistency combined security mechanisms
<i>- b19-SymptomAssistant</i>	<i>Composite service: b19</i>	- assistance to symptomatology search to find medical specialist	- Authenticity, - Confidentiality, - Integrity - Adaptability - Capacity-Scalability The above properties hold for b19, b20 services	- b7-HTTP, b7-HTTPS, HTTPS - c8 - c5, c6 - c4 The above services hold for b19, b20	c8 is a simple service or mechanism to provide additional integrity; c5, c6 are alternative simple services for portability of commercial data bases to different platforms; c4 is a simple service to add new medical standards
<i>- b20-DemographicData</i>	<i>Composite service: b20</i>	- retrieval, storage, modification, deletion of patient personal and demographic data, laboratory/imaging			
<i>b2-MedicalPractice</i>	<i>Composite Services composed by services: b12, b13, b14</i>				b2 is mandatory CC FR; b12, b13 and b14 are also composite services of b2, and they are also mandatory CC;
<i>- b12-EHRManager</i>	<i>Composite service: b12</i>	- retrieval, creation, storage, modification, sharing of EHR	- Interoperability, - Availability-Pers, - Authenticity, - Confidentiality, - Integrity - AvailabilityPers, - Authenticity, - Confidentiality,	- b4 and/or c2 or c3 - c7, c9a, c9b - b7-HTTP, b7-HTTPS, HTTPS - c8 - c7, c9a, c9b - b7-HTTP, b7-HTTPS, HTTPS - c8	b12 is mandatory HIS NFR solved by b4; c2 or c3 are optional solutions for Interoperability
<i>- b13-Diagnosis</i>	<i>Composite service: b13</i>	- retrieval of medical catalogues, protocols; symptomatology search	- Integrity - Time behaviour	- protocols in Transmission Layer - b7-HTTP, b7-HTTPS, HTTPS - c8	
<i>- b14-Orders&amp;Reports</i>	<i>Composite service: b14</i>	- services for emission, retrieval, modification, and storage of medical orders and reports	- Authenticity, - Confidentiality, - Integrity, - Availability-Pers, - Capacity-Scalability	- c7, c9a, c9b - c4	
<i>b3-AdminSystem</i>	<i>Composite service: b3</i>	- basic administrative services: invoices, billing, statistical reports	- Authenticity, - Confidentiality,  - Integrity, - AvailabilityPers, - CorrectPrecision	- b7-HTTP, b7-HTTPS, HTTPS - c8 - c7, c9a, c9b - b6	b3 is mandatory CC FR;
<i>b15 LabImageSystem</i>	<i>Composite service: b15</i>	- retrieval, storage, modification of laboratory and imaging radio graphical studies	- Interoperability - AvailabilityPers, - Authenticity, - Confidentiality, - Integrity, - CorrectPrecision	- b16, b17 - c7, c9a, c9b - b7-HTTP, b7-HTTPS, HTTPS - c8 - b6	b15 is mandatory CC FR
<i>&lt;&lt;b21&gt;&gt;LabIm Interoperability</i>		- database including a standard for identifying medical laboratory observations	- Interoperability		
	<i>Set of simple services: b16-LOINC</i>	- mechanism for conversion to standard imaging graphic formats;		- database (repository)	Interoperability is mandatory NFR for b15; both b16, b17 must be present in a configuration
	<i>b17-DICOM</i>	conformed by a format file and a common. protocol; files are exchanged via TCP/IP	- Interoperability	- Market translation engine	

<<b8>>CompMod	Set of simple services: b6- Algor* (*) multiplicity	- Algorithms for computations	- CorrectPrecision	- modules	CorrectPrecision is mandatory NFR for b1, b3 and b15
<<b9>>SecurityMod	Set of simple services: b7. Data Security - b7-HTTP - HTTPS - b7-HTTPS	- Algorithms for user identification (Authenticity) and policy of user access rights (Confidentiality) of medical information	- Authenticity, - Confidentiality, - Integrity	- b7-HTTP - HTTPS - b7-HTTPS	- security is mandatory HIS NFR; alternatives for security algorithms in b7 combined with security protocols HTTP /HTTPS; integrity is also treated in data base; only one alternative holds
<<b10>>HL7InteropEng	Set of simple services: b4. Mirth	- translation services for EHR to HL7 standard format	- Interoperability	- market translation engine	b4 is mandatory to solve interoperability HIS NFR
c1-DB - c16-EHRDB - c17-LABIMDB - c18-SPDB - c19-CATDB	Set of composite composite services: c16, c17, c18, c19	- database services via usual DBMSs	- Capacity-Scalability - Integrity - AvailabilityPers - Adaptability - Interoperability	- modules, mechanisms	c1 is mandatory CC FR, then all sub-components are also mandatory CC; in this study, only c1 is considered
<<c10>>AMedStds	Set of simple services: c4. NewMedStds *	- addition of new standards for medical practice: formats, catalogues, medical protocols, etc.	- Capacity-Scalability	- modules	Optional NFR
<<c11>>DataInteg	Set of simple services: c8. IntegMech *	- provide data integrity for databases	- Integrity (c8)	- mechanisms	Mandatory NFR for c1-DB
<<c12>>DataAvail	Set of simple services: c9. Availability Mechanisms - c9a. Mirror - c9b. MirrorReplic	- provide backups for databases	- AvailabilityPers	- mechanisms	Optional NFR for c1-DB; only one alternative holds
<<c13>>DataPers	Set of simple service: c7- Hibernate	- provide data persistency to databases	- AvailabilityPers	- engine	Mandatory NFR for c1-DB
<<c14>> DBAPIs	Set of simple services: c5. JDBC c6. ODBC	- Portability to different platforms for commercial databases	- Adaptability	- APIs	Optional NFR for c1-DB
<<c15>> HL7Data ModelEng	Set of simple services: - c2. HL7Eng - c3. HXPEng	- provide data interoperability	- Interoperability	- engines	Optional NFR; at least b4 must be present in an architectural configuration

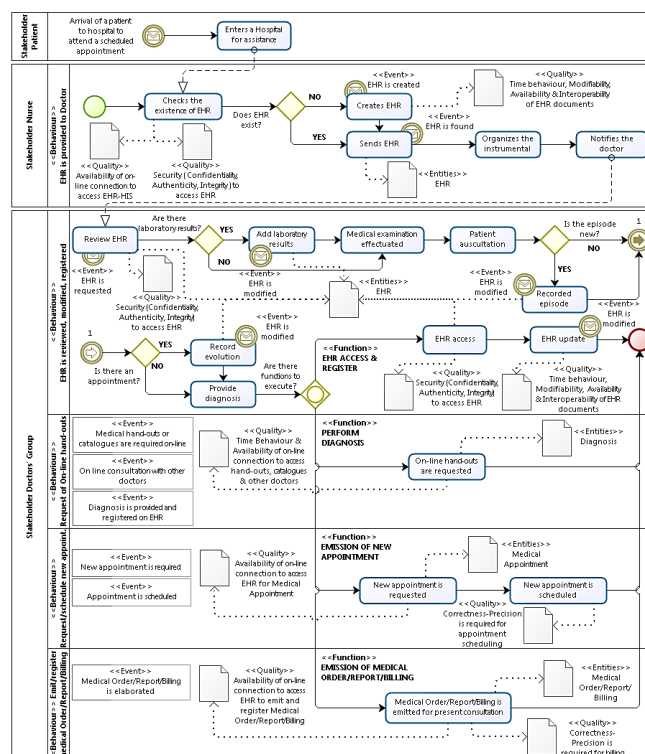


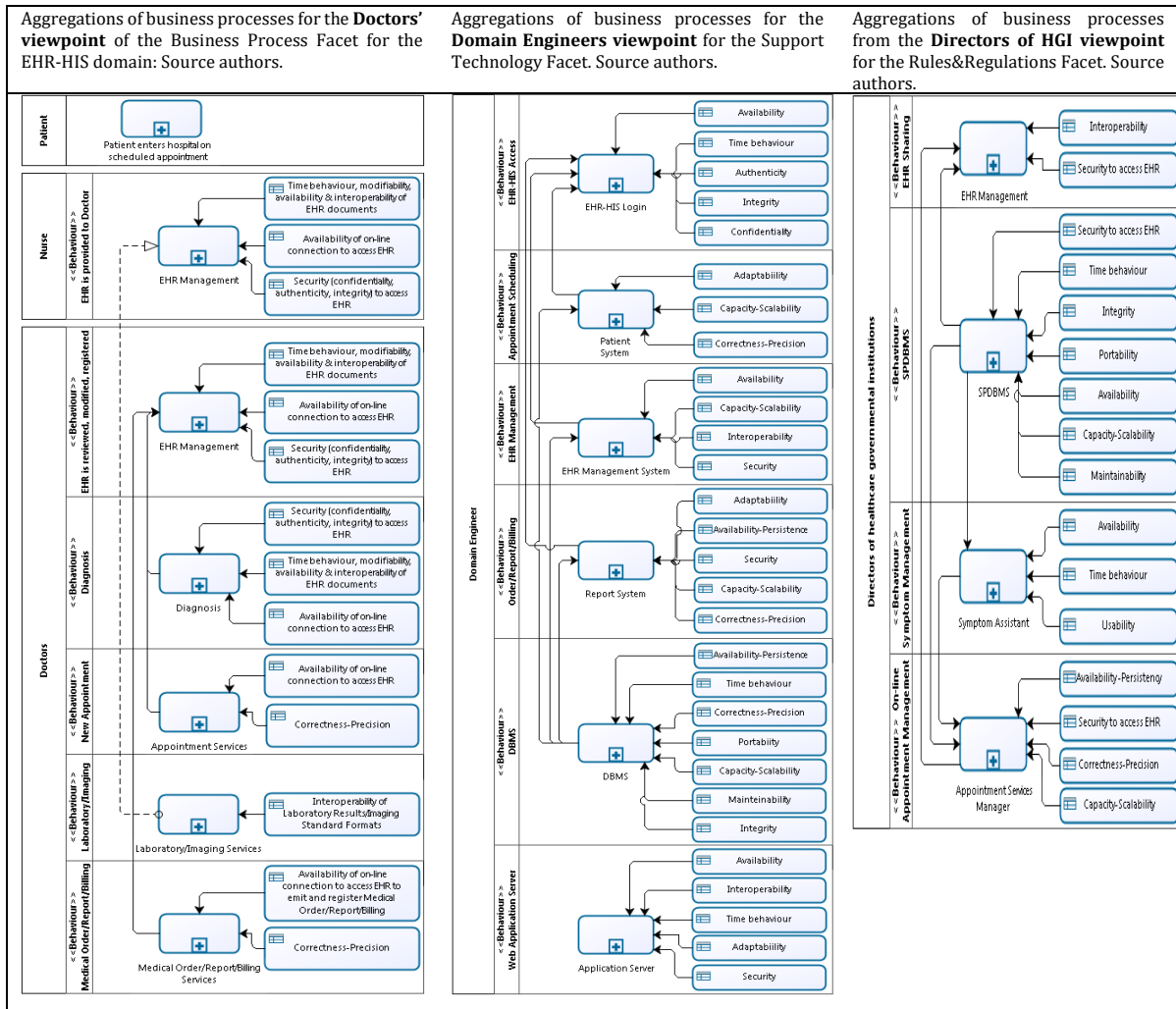
Figure 7. BPMN BP extracted from Behaviours of Doctors Viewpoint

*EHR ACCESS&REGISTER, PERFORM DIAGNOSIS, EMISSION OF NEW APPOINTMENT, EMISSION OF MEDICAL/ORDER/REPORT/BILLING;* these functions are CS from the Doctors viewpoint. *Other stakeholder's viewpoints and facets are studies to obtain additional BG: PATIENT SYSTEM, EHR MANAGEMENT SYSTEM, REPORT SYSTEM, DBMS, APPLICATION SERVER, EHR SHARING, SpDBMS, ON-LINE SYMPTOMS ASSISTANT, ON-LINE APPOINTMENT MANAGEMENT. CS are those satisfying these BG.*

*2.4.2 SOMA Subsystems Analysis – in our terms: aggregation, update CA, update EQM (Service Portfolio), map aggregations to composite services*

Activities and functions from other stakeholder's views and facets are aggregated in BPMNagg to have a unique Domain Model (Figure 8) with aggregated activities (FR) and their quality properties (NFR), represented by BG, and mapped into composite services, which will be architectural components of UCA (Figure 9); EQM is also updated in UEQM (Service Portfolio).

*3. Asset Scoping - to determine the SPL Core Asset Proposal*  
The Core Asset proposal is conformed by CA, UCA, EQM, UEQM, DQM, BPMNagg.



**Figure 8.** Domain Model with all stakeholder's viewpoints and facets specifications

The Future Products Document (Step I.1.3), and the BPMNagg are studied to consider if new composite services contribute to the SPL future products; changes of CA are mapped into new composite services (outlined in grey in Table 6).

## II. Domain Requirements Engineering (DRE) phase

Steps II.1, II.2, II.3 and II.4 for requirements analysis, specification, validation, and management have been considered in UCA and UEQM; if UCA is represented by an ontology [28], reasoning tools can be used to validate requirements consistency by formulating correct consistency rules according to optional and mandatory

constraints in EQM (Table 6). The feature oriented model (FODA) approach [29] is not used here as such; however optional and mandatory options are considered in the WSRA variability model (Table 6).

## III. Domain Design (DD) phase

### 1. WSRA-SPL design - SOMA Components Specification: variability modelling

WSRA is conformed by CC (composite services) and by <<vp>> (sets of simple or composite services) that group UCA variants (simple or composite services) performing semantically similar tasks; UEQM is updated (Table 6).

- WSRAs are an undirected, connected graph, constructed from UEQM (Figure 10); it can also be specified as an ontology [28].

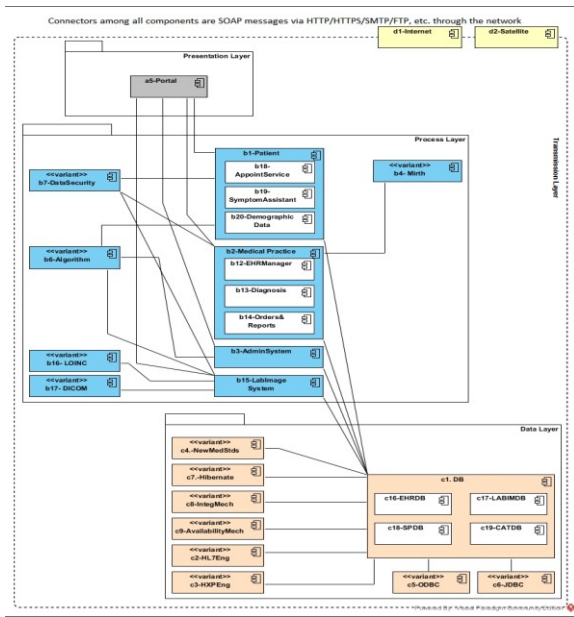


Figure 9. Updated CA (UCA)

2. WSRAs evaluation: is a quality assurance technique.

WSRA is compliant with quality requirements by construction; each WSRA component fulfils the required quality; moreover, WSRA satisfies quality requirements of the domain architectural style.

3. Domain Design Management:

Requirements traceability is solved by construction, to guarantee the WSRA evolution; all high-level quality properties required by each composite service in the architecture are specified and solved by simple services.

7. Analysis of results

As it had been pointed out in this paper, SPL and SOA are approaches for software development used in industrial practice favouring reuse of existing assets and capabilities, rather than redevelop to construct new systems. This work exploited benefits from both approaches integrating them into a unique service-oriented SPL architectural design process called WSRAs-SPL, specified in Section 5, which is our main result. In the literature discussed in Section 3, a complete process to design a RA considering WS was not found. Only the SOMA process, discussed in Section 4, was found, but it builds an architecture for a single system, and not for a family of systems, as it is the SPL case. This was the main motivation to specify our WSRAs-SPL process.

In a previous work, the QuaDRA [14] process was defined to build an SPL RA without considering explicitly WS as RA components, hence QuaDRA was adapted in our present work including the SOMA method to conform the WSRAs-SPL design process. The classical software architecture definition [19] was used; components are represented as services and their connections are the communication between them, i.e. SOAP messages exchanged via HTTP, HTTPS, SMTP, FTP, etc.

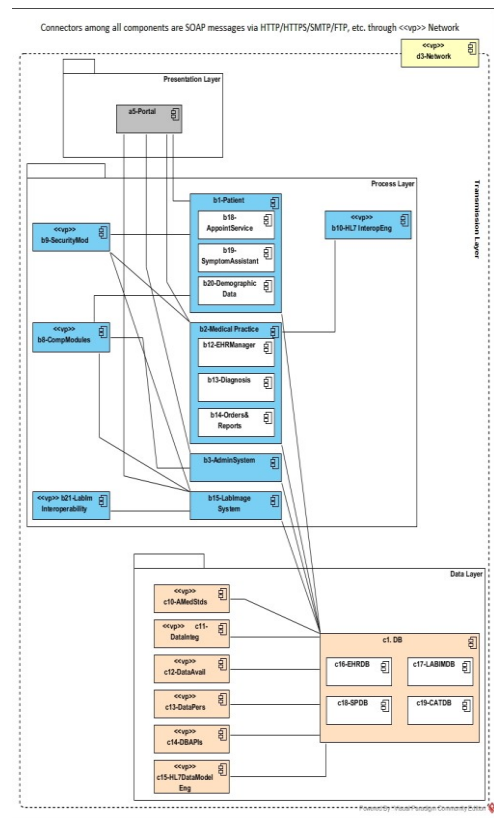


Figure 10. WSRAs

WS are considered in WSRAs-SPL as components conforming the abstract, logical view of a system, modules, programs, databases, etc., carrying out a business operation or functionality, specified by business processes. Hence a business model is considered in the first phase of WSRAs-SPL, the PL Scoping (PLScoP). Another important consideration included in WSRAs-SPL was that the orchestration of WS from business processes workflows is assumed to be correct, being this a responsibility of the service bus middleware and/or automatic translation tools; what is important for WSRAs-SPL is that WS act together in a composition as a single application [22], and we suppose it to be correct. Hence, from the architectural perspective, we are not concerned with message exchanges, but in their choreography representing properties of a component and the relationships between the services conforming it. With respect to include explicitly quality issues, as in the architectural configuration produced by QuaDRA, simple services are also considered architectural components originating from implicit functionalities providing solutions or low-level mechanisms to satisfy high-level quality properties required by more complex components solving system functionalities, designed as composite services.

In consequence, the three main problems found to combine SPL and SOA approaches into a unique reference architecture design process, were solved in this research: the representation of the SPL RA with services, represented in the standard UML notation; the satisfaction of the quality properties required by functionalities to behave properly, obtained by construction at the early steps of the process. Finally, the variability management required in SPL, which is obtained by conforming WSRAs with common composite services and variability points

[6], or set of services containing variants (simple or composite services), were constructed by grouping the UCA variants (composite or simple services) performing semantically similar tasks.

## 8. Conclusions

The SPL development is in general a complex process, since the construction of an evolutionary generic architecture to build concrete software products and a repository of reusable assets are involved. In this work, the SOA and SPL approaches were integrated into the new WSRA-SPL process to design a RA composed by WS, following the Domain Engineering lifecycle outlined in the ISO/IEC 26550 standard reference model for the SPL development [7]. However, since this reference model was defined generally for any kind of software systems' family, service oriented SPL were not explicitly considered. We decided to define the complete WSRL-SPL development process, including the SOMA development approach to design service-based applications [21], adapting it to the SPL context, to complement this limitation of the ISO/IEC 26550 model. The SPL Domain Engineering lifecycle was already considered in the QuaDRA process [14] which added also software quality issues, that had not been explicitly specified in the standard, and the connected graph structure representing the RA configuration has helped to automatize some of the WSRA-SPL steps. In conclusion, the main contribution of this research was the definition and application of the WSRA-SPL process, which integrates two main software development processes, WS driven SOMA and SPL driven QuaDRA.

Another important contribution of this work, imbedded in the specification of tWSRA-SPL, was to provide solutions to the three main problems identified in this research on the integration of the SOA and SPL software development approaches, namely, the representation of the SPL RA with services, the satisfaction of quality properties required by functionalities to behave properly, and the SPL variability management. They were solved as follows in order to achieve our goal: - the 1<sup>st</sup> problem of representing WSRA as a "real" software architecture (components and connectors) [19] was solved considering the service choreography as a set of WSRA components, connected by usual service messaging exchanges. - the 2<sup>nd</sup> problem was solved considering high-level quality characteristics required by functional components or composite services, as simple or composite services (mechanisms, API, engines or toolkits) providers of these qualities; - the 3<sup>rd</sup> problem was solved considering the variation points imbedded by construction into the WSRA architecture, taking into account tasks' similarities; QoS, which are low-level abstraction values, can be also used for variants' selection in the Application Engineering lifecycle, where new products of the SPL family are derived; however the QoS issue was outside the scope of our paper, since WSRA-SPL concerns the Domain Engineering lifecycle.

The WSRL-SPL process has been specified as a reusable tool, precising artefacts and technics, and it has been applied to the HIS domain, which considers families of complex software systems, and where complete SPL have

not yet been defined. The automatic steps in the WSRA-SPL process are the construction of the initial CA and the WSRA generation from UCA; however, we consider necessary to build software tools to support the WSRA-SPL semiautomatic steps, being this an on-going work.

## Acknowledgment

This work has received partial support from national projects and institutions: DARGRAF PG 03-8730-2013-2 project of CDCH (Consejo de Desarrollo Científico y Humanístico), and the Postgraduate Studies of Computer Science, Faculty of Science, both of the Universidad Central de Venezuela.

## References

- [1] Bultan, T., Fu, X. & Hull, R. (2003). [Conversation Specification: A New Approach to Design and Analysis of E-Service Composition](#). In Proceedings of 12th International World Wide Web Conference on (pp. 403–410). Budapest, Hungary.
- [2] W3C Working Group. (2004). [Web Services Architecture Note 11](#), Feb. 2004.
- [3] Milanovic, N. (2001). [Contract-based Web Service Composition Framework](#), Doctoral Dissertation, Humboldt University, Berlin.
- [4] Clements, P. & Northrop, L. (2001). [Software product lines: practices and patterns](#). Readings: Addison Wesley.
- [5] Nakagawa, E., Antonio, P. & Becker, M. (2011). [Reference Architecture and Product Line Architecture: a subtle but critical difference](#), ECSA 2011, LNCS 6903, pp. 207-211, Springer-Verlag, Berlin.
- [6] Pohl, K., Bockle, G. & Van Der Linden, F. (2005). [Software Product Line Engineering: Foundations, Principles, and Techniques](#). Springer.
- [7] ISO/IEC. (2013). [ISO/IEC NP 26550: Software and Systems Engineering – Reference Model for Software and Systems Product Lines](#). ISO/IEC JTC1/SC7 WG4.
- [8] Carlos, P. & Diego, J. (2015). [SPLIT: An Automated Approach for Enterprise Product Line Adoption Through SOA](#), Journal of Internet Services and Information Security (JISIS), (5) 1 pp. 29-52, Feb.
- [9] Helferich, A. et al. (2007). [Software Product Lines, Service-Oriented Architecture and Frameworks: Worlds Apart or Ideal Partners?](#) Draheim and Weber (Eds), TEAA 2006, Berlin Heidelberg, LNCS 4473, pp. 187-201.
- [10] Losavio, F., Ordaz, O. & Esteller, V. (2015). [Refactoring-Based Design of Reference Architecture](#), Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS) 5(1), pp. 32-48.
- [11] ISO/IEC. (2011). [ISO/IEC 25010: Systems and software engineering -- Systems and software Quality Requirements and Evaluation \(SQuaRE\)](#) -- System and software quality models. ISO/IEC JTC1/SC7/WG6, Draft.
- [12] Department of Defense of the U.S.A. (1983). [ADA Programming Language ANSI/MIL-STD 1815-A](#), USA. January 1983.
- [13] Brosgol, B. (2012). [ADA 2012: The joy of contracts, Electronic Design](#).
- [14] Herrera, J. C., Losavio, F. & Ordaz, O. (2016). [QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines based on ISO/IEC 26550](#). Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS) 6(1), pp. 20-38.
- [15] ISO/IEC. (1998). [ISO/IEC 13236: Information Technology – Quality of service: Framework](#), ISO/IEC JTC 1/SC 6.

- [16] Rajendran, T. & Balasubramanie, P. (2009). [Analysis on the Study of QoS-Aware Web Services Discovery](#). Journal of Computing, Vol. 1, No. 1, Dec.
- [17] Losavio, F., Matteo, A. & Rahamut, R. (2008). [Characterization of Web Services domain](#), 6th Latin American and Caribbean Conference for Engineering and Technology, Tegucigalpa, Honduras June 4 – 6.
- [18] Kruchten, P. (2000). [Architectural Blueprints—The Rational Unified Process An Introduction](#). 2nd Edition.
- [19] Shaw, M. & Garlan, D. (1996). [Software architecture: perspectives on an emerging discipline](#). Prentice Hall.
- [20] Martínez, N. & Román, A. (2012). [Diseño de una arquitectura orientada a servicios para un establecimiento de salud de nivel de complejidad 1-3](#). Trabajo de grado., Universidad Peruana de Ciencias Aplicada, Lima.
- [21] Arsanjani, A. (2004). [Service-oriented modelling and architecture How to identify, specify, and realize services for your SOA](#). IBM Software Group. November.
- [22] IBM. (2009). [IBM Service-oriented Architecture \(SOA\) Solutions](#). IBM SOA Foundation.
- [23] Herrera, J., Losavio, F. & Ordaz, O. (2016). [Product Line Scoping for Healthcare Information Systems Using the ISO/IEC 26550 Reference Model](#), 4to Simposio SCTC 2016 (pp. 33-46), Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela, 9-11 Mayo.
- [24] Zhang, L., Zhang, J. & Cai, H. (2008). [Service-oriented Computing](#), ICSOC 2008 Workshops, Springer, Tsinghua University Press, Beijing.
- [25] Tian, C. et al. (2006). [Business Componentization: A Guidance to Application Service Design](#), IFIP, Springer, Vol. 205, Research and Practical Issues of Enterprise Information Systems, pp. 97-107.
- [26] Bjørner, D. (2006). [Software Engineering 3 Domains, Requirements, and Software Design. Texts in Theoretical Computer Science](#). EATCS Series. Editors: W. Brauer G. Rozenberg A. Salomaa. Springer-Verlag Berlin Heidelberg.
- [27] Morrison, C., Iosif, A. & Danka, M. (2010). [Report on existing open-source electronic medical records](#), University of Cambridge, Computer Laboratory, Technical Report No. 768, UCM-CL-TR-768.
- [28] Losavio, F. & Ordaz, O. (2016). [Reference Architecture Representation by an Ontology for Healthcare Information Systems Software Product Line](#), 4to Simposio SCTC 2016, (pp. 20-32). Escuela de Computación, Universidad Central de Venezuela, 9-11 Mayo.
- [29] Lee, K., Kang, K. & Lee, J. (2002). [Concepts and Guidelines of Feature Modelling for Product Line Software Engineering](#), Proceedings of the 7th. International Conference on Software Reuse: Methods, Techniques, and Tools, pp. 62-77.
- [30] Lewis, G., Morris, E., Smith, D. & Simanta, S. (2008). [SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment](#). CMU/SEI-2008-TN-008. Carnegie Mellon University.
- [31] Cohen, S. & Krut, R. (2010). [Managing Variation in Services in a SPL Context](#), Technical Note. CMU/SEI-2010-TN-007. Carnegie Mellon University.
- [32] Losavio, F., Ordaz, O. & Santos, I. (2015). [Proceso de análisis del dominio ágil de sistemas integrados de salud en un contexto venezolano](#), ENL@CE, Vol. 12, No.1, pp.101-134.
- [33] Gómez, J. & Zapata, C. (2014). [SPL como complemento para el desarrollo de soluciones orientadas a servicios \(SOA\): Una revisión de la literatura](#). Trabajo de grado. Universidad de Medellín, Colombia.
- [34] OMG. (2008). [OMG - HL7. Practical Guide to SOA in Healthcare](#), Healthcare Services Specification Project, Health Level Seven, OMG.
- [35] Berard, E. (1992). [Essays in Object-Oriented Software Engineering](#). Nueva York: Prentice Hall.
- [36] Enríques, H. & Evelin, G. (2007). [Capítulo 3-SOA \(Arquitectura Orientadas a Servicios\)](#).