UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
CENTRO DE INGENIERÍA DE SOFTWARE Y SISTEMAS

PROPUESTA METODOLÓGICA PARA EL MANTENIMIENTO DE SOFTWARE BASADO EN LA MODELACIÓN ÁGIL. CASO DE ESTUDIO RUBRICARTE

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela por el
Br. Angel Doza
Para optar al título de
Licenciado en Computación

Tutora: Profesora Nora Montaño

Caracas, 08 de abril de 2015

ACTA

Quienes suscriben, miembros del Jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por el Bachiller Angel Adolfo Doza Cruz, C.I: 19.401.924, con el título: "**Propuesta Metodológica para el Mantenimiento de Software Basado en la Modelación Ágil. Caso de Estudio Rubricarte**", con el fin de optar por el título de Licenciado en Computación, dejan constancia de lo siguiente:

Una vez leído el trabajo por cada uno de los miembros del Jurado, se fijó el día 10 de abril de 2015 a las 10 a.m., para que su autor lo defendiera en forma pública, en la Sala del Centro de Ingeniería de Software y Sistemas (ISYS) de la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela. Una vez realizada esta defensa, a través de una presentación oral de contenido y de operatividad del caso de estudio utilizado y luego de la correspondiente sección de preguntas, el Jurado decidió aprobarlo.

En fe de lo anterior, se levanta la presente Acta, en Caracas a los diez días del mes de abril del año dos mil quince.

_____

Prof. Nora Montaño

Tutor

_____ _____

Prof. Jossie Zambrano

Jurado Principal

Prof. Vanessa Miguel

Jurado Principal

RESUMEN

En los últimos años en la Ingeniería del Software se ha trabajado en tratar de resolver la dificultad asociada al desarrollo de programas libres de defectos, fácilmente comprensibles, y que sean verificables; con el fin de poder llevar a cabo el proceso de mantenimiento del software de una forma que impacte lo menos posible a los entes involucrados (gestores de empresas, responsables de centros de proceso de datos, informáticos y usuarios). A pesar de esto, en la actualidad, existen pocos métodos, herramientas y técnicas desarrollados exclusivamente para el mantenimiento de software. En este Trabajo Especial de Grado, se plantea una propuesta metodológica para el mantenimiento de software basada en la modelación ágil y la programación extrema, la cual se aplica al caso de estudio RubricArte.

Palabras claves:
Metodologías Agiles, Desarrollo Iterativo, Mantenimiento de Software, Reingeniería, Pruebas de Software, Rúbrica.

INDICE

INDICE DE FIGURAS, TABLAS Y ANEXOS

INTRODUCCION

Desde 1.968, año en que se llevó a cabo la primera conferencia organizada por la OTAN sobre desarrollo de software, se ha avanzado fuertemente en el área de la ingeniería del software para tratar de resolver el problema asociado a la crisis del software, que es un término utilizado para referirse a la dificultad asociada al desarrollo de programas libres de defectos, fácilmente comprensibles, y que sean verificables. Algunas de las causas que dan origen a esta situación son la complejidad que supone la tarea de programar y especialmente, la poca importancia dada al proceso de mantenimiento del software por parte todos los entes afectados (gestores de empresas, responsables de centros de proceso de datos, informáticos y usuarios).

Cuando en el área de la ingeniería del software se discuten las metodologías a seguir para desarrollar un sistema, las etapas del proceso suelen limitarse al análisis y especificación de requerimientos, arquitectura de desarrollo, programación, pruebas y documentación, olvidando el mantenimiento, a pesar de que es una de las más importantes dentro del ciclo de vida del software y sin duda alguna, la más extensa. Incluso la educación suele estar enfocada hacia el desarrollo de un software pero no hacia su mantenimiento, razón por la cual existen pocos métodos, herramientas y técnicas desarrollados exclusivamente para la etapa de mantenimiento.

Anteriormente existía un grupo bien extenso de técnicas que podían ser aplicadas durante la actividad de mantenimiento de software, pero estas dependían de la complejidad del software a mantener y del alcance que tuvo el desarrollo de este, lo cual hacía muy engorrosa su aplicación; es por ello que a lo largo del presente Trabajo Especial de Grado se aborda la solución que se utilizó para resolver el problema que se enmarcaba en la falta de métodos ágiles para la aplicación del proceso de mantenimiento de software.

Basado en este problema se planteó el objetivo de realizar una propuesta metodológica para llevar a cabo el mantenimiento de software, basada en metodologías de desarrollo ágil, como Programación Extrema (XP) y Modelado Ágil (MA), con el fin de poder incorporar técnicas de mantenimiento perfectivo y correctivo. Esta propuesta se ajustó a la selección de una aplicación (caso de estudio) denominada RubricArte, la cual fue construida aplicando MA con el propósito de visualizar el mantenimiento como una "acción continuada" de las iteraciones de desarrollo. En base a esto se definen los siguientes objetivos específicos:

- Realizar el levantamiento de información pertinente, necesaria o dentro del contexto del mantenimiento de software.
- Realizar una fase de exploración en la que se inspeccione el software, estudiar la posible documentación y los comentarios a lo largo del código fuente de RubricArte, con el fin de obtener el estado inicial de este.
- Determinar la técnica y/o el proceso de mantenimiento a llevar a cabo para realizar la nueva versión de RubricArte.
- Realizar una evaluación, basada en heurísticas, por parte de expertos en usabilidad, para poder conocer las recomendaciones que tienen para lograr optimizar la experiencia de los usuarios con RubricArte.
- Estudiar los resultados obtenidos de la evaluación para conocer los diferentes aspectos que deben ser modificados y presentarlos al cliente como requerimientos de la nueva versión de RubricArte.
- Aplicar la metodología de desarrollo ágil propuesta para llevar a cabo el mantenimiento a RubricArte, obteniendo como resultado una herramienta que cumpla con los estándares de calidad del software como lo son la comprensibilidad, consistencia, legibilidad, usabilidad, expansibilidad.

Además de lo anteriormente explicado se tiene como premisa lo siguiente: "quien estuvo encargado de realizar este proceso de mantenimiento, no formó parte del equipo que desarrolló el sistema"; por lo cual esta propuesta incluye

principalmente la aplicación de las técnicas de comprensión del programa y reingeniería, para luego aplicar los tipos de mantenimiento correctivo y perfectivo.

La metodología propuesta se encuentra dividida en dos (2) etapas; la primera de ellas corresponde con el levantamiento de la información (incluye a las actividades de diagnóstico de la aplicación y evaluaciones por expertos en usabilidad), mientras que la segunda consiste en la implementación del mantenimiento y esta, a su vez, se encuentra subdividida en iteraciones, cada una de las cuales empieza con la actividad de iteraciones de desarrollo, continúa con la aprobación de lo implementado a través de las pruebas unitarias y concluye al momento de realizar una liberación.

Tanto el caso de estudio en su versión inicial, como este Trabajo Especial de Grado fueron desarrollados en el Centro de Ingeniería de Software y Sistemas (ISYS) de la Escuela de Computación de la Universidad Central de Venezuela.

El presente documento se encuentra estructurado en capítulos, a continuación se muestra una síntesis del contenido de cada uno de estas:

- Capítulo 1: Este capítulo corresponde al levantamiento teórico, donde se definen los conceptos básicos y las técnicas asociadas al mantenimiento del software.
- Capítulo 2: En este capítulo se explica la metodología ágil para el desarrollo, la propuesta realizada en el Seminario asociado a este Trabajo Especial de Grado y la herramienta seleccionada como caso de estudio para llevar a cabo el mantenimiento del software basado en modelos ágiles de proceso.
- Capítulo 3: En este capítulo se exponen las actividades llevadas a cabo en cada una de las fases que forman parte de la metodología propuesta para el mantenimiento del software, así como los incrementos liberados en cada iteración.

CAPÍTULO I: CONCEPTOS Y TÉCNICAS ASOCIADOS AL MANTENIMIENTO DEL SOFTWARE (MARCO CONCEPTUAL)

El proceso de producción de software de calidad tiene asociada una complejidad, la cual se intenta abordar a través de la descomposición de este en diversas etapas y ha recibido el nombre de Ciclo de Vida del Software. Es de resaltar que desde que se publicó el primer modelo de proceso de desarrollo, el de cascada, ya se pensaba en la etapa de mantenimiento como una fase fundamental dentro del Ciclo de Vida del Software (Figura 1: Ciclo de vida del software para el modelo en cascada).

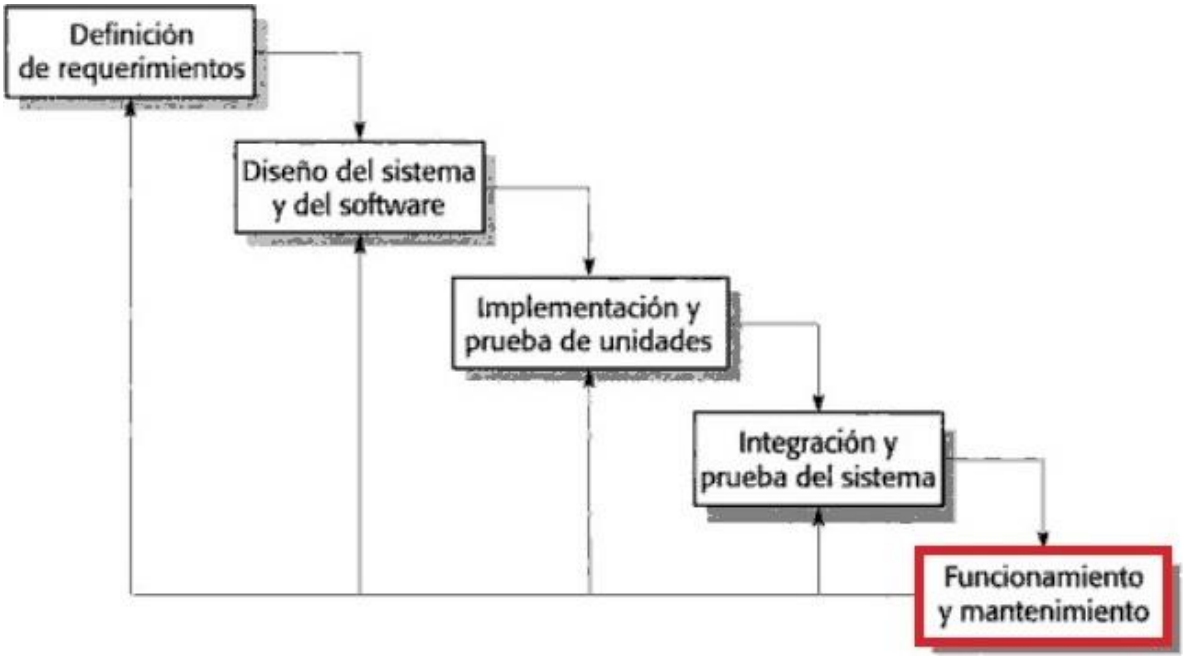


Figura 1: Ciclo de vida del software para el modelo en cascada (Sommerville, 2005)

Dado que el presente Trabajo Especial de Grado está enfocado en la etapa de mantenimiento del software, en este capítulo se explican los diferentes aspectos relacionados a esta, como lo son: la definición, los problemas asociados al alto costo que involucra, los diferentes tipos, el proceso junto con sus actividades relacionadas y los procesos de la comprensión del programa y la reingeniería.

1. Mantenimiento del Software

En la norma ISO/IEC 14764 el mantenimiento del software se define como el conjunto de actividades destinadas a proporcionar soporte económicamente rentable para un determinado producto software, las cuales son desarrolladas tanto antes de la liberación del sistema (etapa de pre-entrega) como después de esta (etapa de post-entrega). Las actividades antes de la entrega incluyen la planificación para la post-entrega, para el mantenimiento y para la determinación de la logística en las actividades de transición. Las actividades para después de la entrega incluyen la modificación del software, la capacitación y la participación de un auxiliar de operación. Además de esta definición existe la hecha en la norma IEEE para el mantenimiento del software, IEEE 1219, en la cual se conceptualiza como la modificación de un producto de software después de su liberación para corregir fallos, para mejorar su rendimiento y otros atributos, o para adaptar el producto a un entorno modificado.

Según algunos estudios se invierte por lo menos un 60% del costo total generado a lo largo del ciclo de vida del software en esta etapa (Tabla 1: Costos del mantenimiento), lo que da lugar a que la mayor parte del presupuesto total de los centros de proceso de datos se destine a mantener los sistemas existentes, por ello, se puede asegurar que el mantenimiento es la fase dominante, en tiempo y costo, del ciclo de vida (Piattini, Calvo-Manzano, Cervera, & Fernández, 2006).

Referencia	***% Mantenimiento***
Lientz y Swanson (1980)	60 %
Rock-Evans y Hales (1990)	67 %
Frazer (1992)	80 %
Pigoski (1997)	75 %
Pressman (2002)	60 %
Sommerville (2002)	65-75 %

Tabla 1: Costos del mantenimiento (Piattini, Calvo-Manzano, Cervera, & Fernández, 2006)

A pesar de la gran importancia que supone la fase de mantenimiento y del gran costo asociado a esta, la educación del ingeniero de software suele estar enfocada hacia el desarrollo de un nuevo sistema pero no hacia su conservación,

es por ello que existen pocos métodos, herramientas y técnicas desarrollados exclusivamente para el mantenimiento.

1.1. Problemas asociados al mantenimiento del software

Los altos costos asociados al mantenimiento se deben a diferentes problemas que se presentan al llegar a esta etapa (Sommerville, 2005), algunos de los cuales se explican a continuación:

- La estabilidad del equipo. Después de entregar un sistema, es normal que el equipo de desarrollo se disuelva y la gente trabaje en otros proyectos. El nuevo equipo o los responsables del mantenimiento del sistema no comprenden el software o las razones de fondo de las decisiones de su diseño y dedican demasiado esfuerzo durante el proceso de mantenimiento en comprender el sistema antes de implementar cambios en él.
- Las habilidades del personal. El personal de mantenimiento a menudo no tiene experiencia y no está familiarizado con el dominio de la aplicación. El mantenimiento tiene una pobre imagen entre los ingenieros de software, está visto como un proceso que requiere menos habilidades que el desarrollo del sistema y a menudo se le asigna al personal principiante. Además, los sistemas antiguos pueden haberse escrito en lenguajes de programación obsoletos y el personal de mantenimiento puede no tener mucha experiencia de desarrollo en estos lenguajes y debe aprenderlos para mantener el sistema.
- La edad y la estructura del programa. A medida que pasa el tiempo, la estructura de los programas tiende a degradarse con los cambios, por lo que se vuelve más difícil de comprender y modificar. Algunos sistemas han sido desarrollado sin técnicas modernas de ingeniería del software, pueden no haber sido nunca bien estructurados y quizás estén optimizados para su eficiencia en lugar de para su comprensibilidad. La documentación del sistema puede haberse perdido o no existe y los sistemas antiguos pueden no haber sido sometidos a gestión de configuraciones, por lo que a menudo

se emplea mucho tiempo en encontrar las versiones correctas de los componentes del sistema a cambiar.

Los dos primeros problemas surgen del hecho que muchas organizaciones todavía consideran el desarrollo y el mantenimiento como actividades independientes además, este último, se ve como una actividad de segunda categoría, y no es un incentivo emplear más dinero durante el desarrollo para reducir los costos de los cambios del sistema. La única solución a largo plazo a este problema es aceptar que los sistemas raramente tienen un tiempo de vida definido, pero continuarán en uso de alguna forma, durante un período de tiempo indefinido. Debería pensarse que los sistemas evolucionan a través de un proceso de desarrollo continuado.

Para solucionar el tercer problema, la degradación de la estructura del sistema, pueden aplicarse las técnicas de reingeniería del software para mejorar la estructura del sistema y su comprensibilidad, así como las transformaciones arquitectónicas pueden adaptar al sistema a un nuevo hardware y el trabajo de mantenimiento preventivo puede soportarse para mejorar el software y hacer que sea más fácil de cambiar.

Además de los problemas descritos anteriormente, se puede observar que existen otros factores que afectan directamente a estos costos (Piattini, Calvo-Manzano, Cervera, & Fernández, 2006), entre ellos se encuentran:

- La inexistencia de métodos, técnicas y herramientas que pueda proporcionar una solución global al mantenimiento. Los ciclos de vida del software y, por lo tanto, las metodologías de desarrollo que los siguen, no reflejan la importancia del mantenimiento en términos de esfuerzo y costo necesarios, ni de las actividades que hay que realizar durante esta fase. En la actualidad, prácticamente todos los métodos existentes se centran en el desarrollo de nuevos sistemas en vez de reparar o mejorar los existentes.

- Las actividades del mantenimiento se suelen realizar bajo presión de tiempo. Normalmente los programadores tienen poco tiempo para realizar las modificaciones, por lo que a veces no se actualiza la documentación y las correcciones incompletas implican nuevos esfuerzos en el futuro.
- La poca participación del usuario durante el desarrollo del sistema. Cuando los desarrolladores entregan el sistema al cliente no satisface sus necesidades, lo que da lugar a un gran esfuerzo de mantenimiento posterior.

Existen múltiples factores que afectan los costos en la fase de mantenimiento, y este sigue siendo un problema que no ha sido resuelto de manera precisa dado que existe la visión de que el desarrollo solo se limita a la implementación del sistema, olvidando las futuras modificaciones sobre este. Diversas investigaciones están enfocadas en tratar de reducir este costo y han llegado a la conclusión de que los desarrollos de calidad (aquellos que tienen de forma sistemática la documentación del programa, mantienen el código documentado, siguen estándares, etc.), poseen un costo mucho menor de mantenimiento.

Para el presente Trabajo Especial de Grado los principales inconvenientes para realizar el mantenimiento del software fueron: la estabilidad del equipo, ya que quien se encargó de llevar a cabo esta fase de mantenimiento no forma parte del equipo de desarrollo del sistema original, y la presión de tiempo para realizar las actividades del mantenimiento cumpliendo con los requerimientos de un software de calidad.

1.2. Tipos de mantenimiento del software

Dependiendo del tipo de actividad a realizar se definen cuatro tipos diferentes de mantenimiento (Piattini, Calvo-Manzano, Cervera, & Fernández, 2006):

- Mantenimiento perfectivo: Conjunto de actividades realizadas para mejorar o añadir nuevas funcionalidades requeridas por el usuario. Algunas mejoras pueden consistir en optimizar el rendimiento de un programa, aumentar la facilidad para mantener un programar ante cambios futuros, etc.
- Mantenimiento adaptativo: Conjunto de actividades que se realizan para adaptar el sistema a los cambios en su entorno tecnológico.
    - El entorno de datos: El cambio de soporte de los datos de una aplicación (paso de un sistema clásico basado en archivos a uno relacional).
    - El entorno del proceso: La migración a una nueva plataforma de explotación o implantación de un nuevo sistema operativo.
- Mantenimiento correctivo: Conjunto de actividades dedicadas a corregir fallos[1] detectados por los usuarios durante la utilización del sistema, entre los cuales se pueden considerar de:
    - Procesamiento: terminaciones anormales o salidas incorrectas del programa.
    - Rendimiento: tiempos de respuesta por debajo de los requeridos.
    - Implementación: violaciones de estándares de programación o inconsistencias del diseño.
- Mantenimiento preventivo: Conjunto de actividades que se realizan para facilitar el mantenimiento futuro del sistema. Por ejemplo, se pueden incluir sentencias que comprueban la validez de los datos de entrada o reestructurar programas para mejorar su legibilidad.

1.3. Proceso de mantenimiento del software

El proceso de mantenimiento ofrece las actividades necesarias y las entradas con sus salidas detalladas a esas actividades, y se describe en los estándares para el mantenimiento del software IEEE 1219 e ISO/IEC 14764 (Abran & Moore, 2004). En otras palabras se tiene que el proceso de

[1] Un fallo aparece en el software como consecuencia de la existencia de un defecto.

mantenimiento del software está dividido en diferentes actividades tal como se describe en las normas IEEE 1219 e ISO/IEC 14764. A pesar de la similitud entre ambas normas, para el presente Trabajo Especial de Grado solo se tomó en cuenta la norma IEEE 1219 dado que esta ofrece una visión más amplia del proceso y de las tareas involucradas en el mantenimiento del software (Figura 2: Actividades del proceso de mantenimiento para la norma IEEE 1219-98).

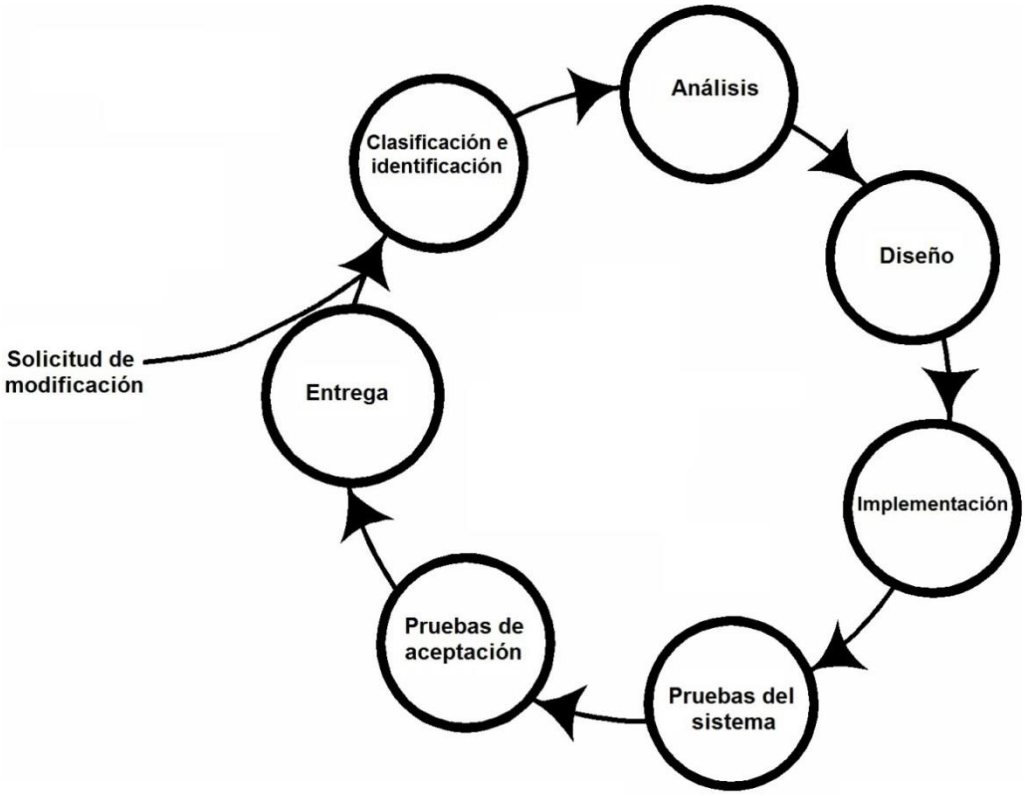


Figura 2: Actividades del proceso de mantenimiento para la norma IEEE 1219-98 (Abran & Moore, 2004)

Durante la actividad denominada “Solicitud de modificación”, el encargado de mantenimiento establece los planes y los procedimientos que deben ser ejecutados durante el proceso de mantenimiento. Las siguientes actividades se realizan de manera iterativa cuando se presenta la necesidad de modificación.

Durante las actividades de “Clasificación e identificación”, “Análisis” y “Diseño” el encargado de mantenimiento analiza y documenta las solicitudes de modificación o reportes de problemas, verifica los inconvenientes, elabora las

opciones para la aplicación de la modificación y obtiene la aprobación de la opción de modificación seleccionada.

En la actividad de “Implementación” el encargado de mantenimiento desarrolla y prueba las modificaciones solicitadas al software.

Finalmente están las actividades “Pruebas del sistema”, “Pruebas de aceptación” y “Entrega”, las cuales garantizan que las modificaciones en el sistema son correctas y que se llevaron a cabo de acuerdo con las normas aprobadas mediante la metodología correcta.

1.4. Comprensión del programa

Para poder aplicar las diferentes actividades que conforman el proceso de mantenimiento del software se debe poder comprender el sistema, es por ello que en esta sección se explica este proceso fundamental (Abran & Moore, 2004).

Cuando se quiere entender un sistema hay que inspeccionar diferentes aspectos de este y para construir las perspectivas necesarias, es preciso extraer información desde los sistemas y representarla adecuadamente. La extracción de la información es importante porque es la base para construir diferentes vistas y, por otra parte, la visualización de la información es esencial para propósitos de comprensión (Berón, Uzal, Henriques, & Pereira, 2007).

El código fuente es la primera vista[2] con la que se enfrenta un programador que está tratando de comprender un sistema, la cual es muy útil ya que este suele estar familiarizado con los lenguajes de programación. Sin embargo, cuando el tamaño del sistema crece, se pierde la claridad y son necesarias otras perspectivas de este.

[2]Una vista es una representación de la información de un sistema que facilita la comprensión de un aspecto del mismo.

Un aspecto del sistema, que se encuentra en un nivel de abstracción más alto, consiste en visualizar las funciones del sistema y las relaciones existentes entre las mismas, un ejemplo de esto es el Grafo de Funciones (GF). GF es un grafo donde el conjunto de nodos está compuesto por las funciones del sistema de estudio y la relación entre ellas está dada por la comunicación de las funciones a través de sus invocaciones. Normalmente, el GF es una vista deseada por los programadores, sin embargo, al igual que el código fuente, cuando el tamaño del sistema crece no presenta una ayuda a la comprensión. Como una alternativa al GF el sistema puede ser visualizado usando los módulos que lo componen. En este caso es posible definir un grafo que muestra la relación de comunicación entre ellos, normalmente este grafo es conocido con el nombre de Grafo de Comunicaciones de Módulos (GCM). Existen experimentos que indican que el GCM presenta una vista clara del sistema aun cuando el tamaño del mismo es grande. No obstante, al presentar un mayor grado de abstracción, oculta detalles que pueden ser útiles para la comprensión.

Si se investiga a diferentes autores como Abran y Moore (2004) y Pressman (2005) se tiene que para realizar el mantenimiento a una herramienta existen una serie de metodologías para poder analizar qué módulos de ésta pueden ser reutilizados para la nueva versión, y muchas veces ayudan a comprender el software, lo cual es de gran ayuda para el caso en que quien va a realizar el mantenimiento del software no es quien lo desarrolló. En las próximas secciones se explica la metodología conocida como reingeniería que una de las técnicas aplicadas al momento de realizar el presente Trabajo Especial de Grado.

1.5. Reingeniería

A diferencia del proceso de mantenimiento donde se conserva la arquitectura del sistema, y su objetivo principal es el proceso de mejora y optimización del software desplegado, así como la prevención y corrección de errores que este pueda tener, surge el concepto de reingeniería como un proceso que conlleva a la reconstrucción parcial o total del sistema a fin de satisfacer

nuevos requerimientos organizacionales o de negocio. Existen diferentes visiones que explican qué es la reingeniería; pero para realizar este Trabajo Especial de Grado se tuvo en cuenta solamente la de Pressman (2005), según la cual constituye todo un proceso[3] para el referido mantenimiento.

Enfoque como un proceso del mantenimiento del software

En el ámbito del software, la reingeniería examina los sistemas y aplicaciones de información con la finalidad de reestructurarlos o reconstruirlos de modo que muestren mayor calidad.

La reingeniería de software comprende una serie de actividades que incluyen análisis de inventario, restructuración de documentos, ingeniería inversa, restructuración de programas y datos, e ingeniería directa (Figura 3: Actividades del modelo de proceso de la reingeniería del software). La finalidad de estas actividades es crear versiones de programas existentes que sean de mayor calidad y tengan mayor facilidad de mantenimiento (programas que serán viables ya muy avanzado el siglo XXI).

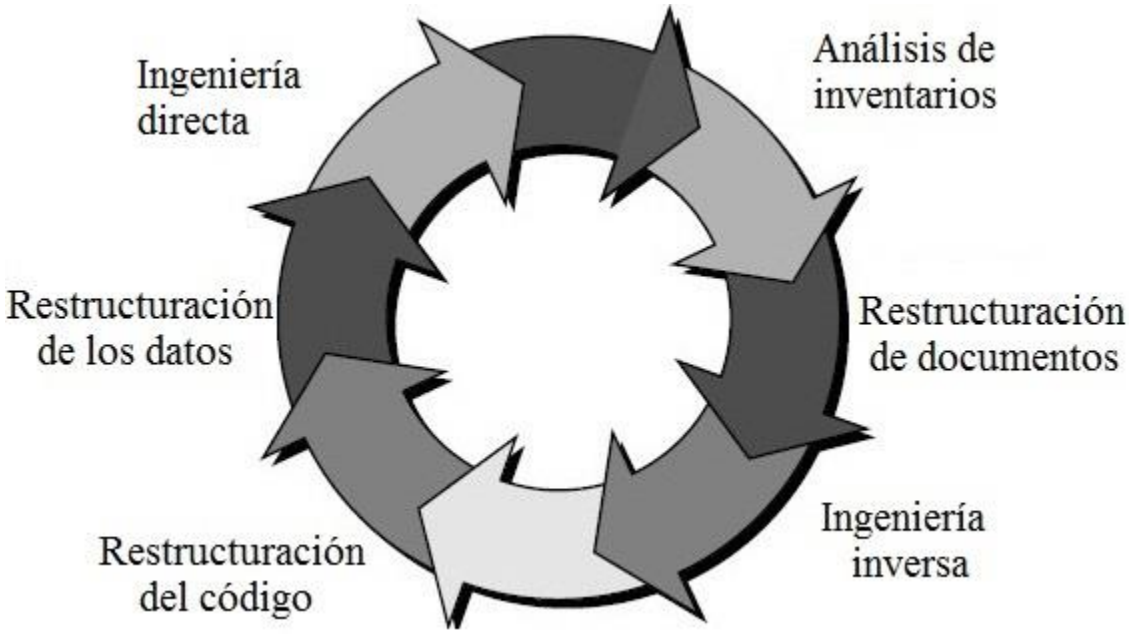


Figura 3: Actividades del modelo de proceso de la reingeniería del software (Pressman, 2005)

[3] Un proceso es un conjunto de actividades o eventos (coordinados u organizados) que se realizan o suceden (alternativa o simultáneamente) bajo ciertas circunstancias con un fin determinado.

1.5.1. Análisis de inventarios

Las organizaciones de software deberían tener un inventario de todas sus aplicaciones. El inventario tal vez no sea más que un modelo de una hoja de cálculo que contenga información que proporcione una descripción detallada (por ejemplo, tamaño, edad, importancia para el negocio) de las aplicaciones activas. Al ordenar esta información - de acuerdo con la importancia para el negocio, antigüedad, facilidad actual de mantenimiento y otros criterios localmente importantes - aparecen los candidatos para el trabajo de la reingeniería. Entonces se pueden asignar los recursos a las aplicaciones candidatas para esta labor.

Es importante señalar que el inventario deberá visitarse con regularidad. El estado de las aplicaciones (por ejemplo, la importancia respecto del negocio) puede cambiar en función del tiempo y, por ende, cambiarán las prioridades para la reingeniería.

1.5.2. Restructuración de documentos

La documentación débil es la marca de muchos sistemas heredados. ¿Pero qué se hace acerca de ello? ¿Cuáles son las opciones?

- Crear documentación consume muchísimo tiempo. Si el sistema funciona vivirá con lo que se tenga; en algunos casos este es el enfoque correcto ya que no es posible recrear documentación para cientos de sistemas. Si un software es relativamente estático está llegando al final de su vida útil, por lo que es improbable que experimente un cambio significativo.
- La documentación debe actualizarse, pero se tienen recursos limitados. Se utilizará un enfoque de “documentar cuando se toque”. Tal vez sea innecesario volver a documentar por completo la aplicación. En cambio, se documentan aquellas porciones del sistema que en la actualidad experimentan cambios. Con el tiempo evolucionará una colección de documentación útil y relevante.

- El sistema es crucial para el negocio y debe documentarse por completo. Incluso en este caso un enfoque inteligente es recortar la documentación a un mínimo esencial.

Cada una de estas opciones es viable y una organización de software debe elegir la más apropiada para su caso.

1.5.3. Ingeniería inversa

La ingeniería inversa es el proceso de analizar un programa con la finalidad de crear una representación de este en un mayor grado de abstracción que el código fuente. La ingeniería inversa es un proceso de recuperación de diseño. Las herramientas de la ingeniería inversa obtienen información del diseño de datos, arquitectónico y de procedimientos a partir de un programa existente.

1.5.4. Restructuración de código

Algunos sistemas heredados tienen una arquitectura de programa relativamente sólida, pero los módulos individuales se codificaron en una forma que dificulta comprenderlos, probarlos y mantenerlos, en tales casos se puede reestructurar el código dentro de los módulos sospechosos. Esta actividad tiende a enfocarse sobre los detalles de diseño de los módulos individuales y en las estructuras de datos locales definidos dentro de los módulos.

Llevar a cabo esta actividad requiere analizar el código fuente empleando una herramienta de restructuración. Para comenzar con el proceso se indican las violaciones de programación y el código se reestructura (lo cual puede hacerse automáticamente), seguidamente el código resultante se debe revisar y probar para garantizar que no se han introducido anormalidades y finalmente la documentación del código interno debe ser actualizada.

1.5.5. Restructuración de datos

Un programa con una estructura de datos débil será difícil de adaptar y mejorar. De hecho, en muchas aplicaciones la arquitectura de datos está más relacionada con la viabilidad a largo plazo de un programa que el código fuente.

A diferencia de la restructuración de código, que ocurre en un grado relativamente bajo de abstracción, la de datos es una actividad de reingeniería a gran escala; en la mayoría de los casos, la restructuración de datos comienza con una actividad de ingeniería inversa. La arquitectura de datos actual se analiza con minuciosidad y se definen los modelos de datos necesarios, se identifican los objetos de datos y los atributos, y después se revisa la calidad de las estructuras de datos existentes. Cuando esta es débil (por ejemplo, actualmente se implementan archivos planos, cuando un enfoque relacional simplificaría enormemente el procesamiento), los datos se someten a un proceso de reingeniería. Puesto que la arquitectura de datos tiene una fuerte influencia sobre la arquitectura del programa y los algoritmos que lo pueblan, los cambios a los datos invariablemente resultarán en cambios arquitectónicos o a nivel de código.

1.5.6. Ingeniería directa

En un mundo ideal, las aplicaciones se reconstruirían empleando un “motor de reingeniería” automatizado. El programa antiguo sería insertado en el motor, analizado, reestructurado y luego regenerado en una forma que exhibiese los mejores aspectos de la calidad del software. A corto plazo es improbable que tal “motor” aparezca, pero las empresas han introducido herramientas que mejoran un limitado subconjunto de dichas capacidades que aborden dominios de aplicación específicos (por ejemplo, las aplicaciones que se implementan mediante un sistema de bases de datos). Más importante, dichas herramientas se están volviendo cada vez más sofisticadas.

La ingeniería directa, también llamada renovación o reclamación, no solo recupera la información de diseño a partir del software existente, también utiliza

esta información para alterar o reconstituir el sistema existente con la finalidad de mejorar su calidad global. En la mayoría de los casos el software sometido a reingeniería vuelve a implementar la función del sistema existente y también añade nuevas funciones o mejoras al desempeño global.

1.6. Resumen del capítulo

El mantenimiento del software tiene dos enfoques, el primero de ellos indica que es una etapa ubicada al final del ciclo de vida del software, cuyo comienzo viene dado con la liberación del sistema y termina con la retirada de este; mientras que el segundo lo refleja como una actividad que es parte del desarrollo basado en el modelo evolutivo. Independientemente del enfoque dado al realizar el mantenimiento del software, este implicará grandes costos para las empresas, debido a que es una parte muy importante dentro del ciclo de vida del software y existen diversos problemas asociados a esta fase como lo son: la estabilidad del equipo, las habilidades del personal, la edad y la estructura del programa, la inexistencia de métodos, técnicas y herramientas que pueda proporcionar una solución global al mantenimiento, entre otros. Aunado a esto se tiene que existen diferentes tipos de mantenimiento pero, indistintamente del aplicado, se debe seguir un proceso (definido de manera similar por las normas IEEE e ISO/IEC), en el que se explican las diferentes actividades y las tareas a ejecutar en cada una de ellas.

Para realizar el mantenimiento del software que compete al presente Trabajo Especial de Grado se aplicaron las técnicas de comprensión del programa y reingeniería, para poder analizar los módulos del software que podían ser reutilizados en la nueva versión, lo cual fue de gran ayuda para llevar a cabo el mantenimiento, dado que quien realizó este proceso de mantenimiento no formó parte del equipo de desarrollo.

CAPÍTULO II: PROPUESTA METODOLÓGICA PARA EL MANTENIMIENTO BASADO EN MODELOS ÁGILES DE PROCESO

Estas metodologías existentes para el mantenimiento del software, explicadas en el capítulo anterior, por si solas son para soluciones de mayor alcance y no están basados en los modelos ágiles de proceso, razón por la cual surge entonces la pregunta: ¿Cómo llevar a cabo el mantenimiento en proyectos menos complejos, con equipos de desarrollo de pocos integrantes, donde se utilicen metodologías ágiles para el mantenimiento? Para tratar de resolver esta incógnita en este capítulo se da una introducción al proceso de desarrollo ágil y se explican las técnicas de desarrollo ágil utilizadas para realizar la propuesta metodológica para el mantenimiento de software. Además, también se realiza una introducción al caso de estudio RubricArte, para finalmente comenzar con la aplicación de las dos (2) primeras actividades definidas en la propuesta metodológica.

La ingeniería de software ágil combina una filosofía y un conjunto de directrices de desarrollo; la filosofía busca la satisfacción del cliente y la entrega temprana de software incremental, equipos de proyecto pequeños y con alta motivación, métodos informales, un mínimo de productos de trabajo de la ingeniería del software, y una simplicidad general del desarrollo; mientras que las directrices de desarrollo resaltan la entrega, por encima del análisis y del diseño (aunque estas actividades no son descartadas), y la comunicación activa y continua entre los desarrolladores y los clientes durante toda el ciclo de vida del software (Pressman, 2005).

1. Procesos ágiles

Los modelos de procesos ágiles están diseñados para producir software útil de forma rápida, generalmente son procesos iterativos en los que se entrelazan la especificación, el diseño, el desarrollo y las pruebas, en los cuales no se espera a tener la versión final del software desarrollado para pasar a su liberación, sino que

por el contrario se trabaja con una serie de incrementos que incluyen nuevas funcionalidades al sistema (Sommerville, 2005). Aunque existen diversos enfoques para el desarrollo rápido de software, estos comparten las siguientes características fundamentales:

- Las actividades de especificación, diseño e implementación son concurrentes; es por esta razón que no existe especificación detallada del sistema, la documentación general se minimiza o es generada automáticamente por el entorno de programación, y el documento de requerimientos del usuario define solamente las características más importantes del sistema.
- El sistema se desarrolla en una serie de incrementos; lo cual implica que los usuarios finales y otros interesados en el desarrollo del sistema participan en la especificación y verificación de los referidos incrementos, además que pueden proponer cambios en el software y nuevos requerimientos que deben ser contemplados e implementados en una posterior iteración.
- A menudo las interfaces de usuario se realizan utilizando un software de desarrollo interactivo, el cual permite crear rápidamente el diseño de la interfaz; dado que existen sistemas que ayudan a generar una interfaz basada en web para un navegador o una interfaz para una plataforma específica.

Uno de los principales métodos para el desarrollo ágil es la programación extrema (XP, por sus siglas en inglés), el cual es una metodología de desarrollo ágil que utiliza un enfoque orientado a objetos para lograr el objetivo de aumentar la productividad al momento del desarrollo (Pressman, 2005). En XP todos los requerimientos se expresan como escenarios llamados historias de usuario, los cuales se implementan directamente como una serie de tareas. Los programadores suelen trabajar en parejas y son quienes desarrollan las pruebas para cada tarea, antes de escribir el código, las cuales se deben ejecutar

satisfactoriamente cuando el código nuevo se integre al sistema; entre cada una de las entregas transcurre poco tiempo (Sommerville, 2005).

La programación extrema abarca un conjunto de reglas y prácticas que ocurren en el contexto de cuatro etapas del marco de trabajo: planificación, diseño, codificación y pruebas (Figura 4: Proceso de la Programación Extrema).

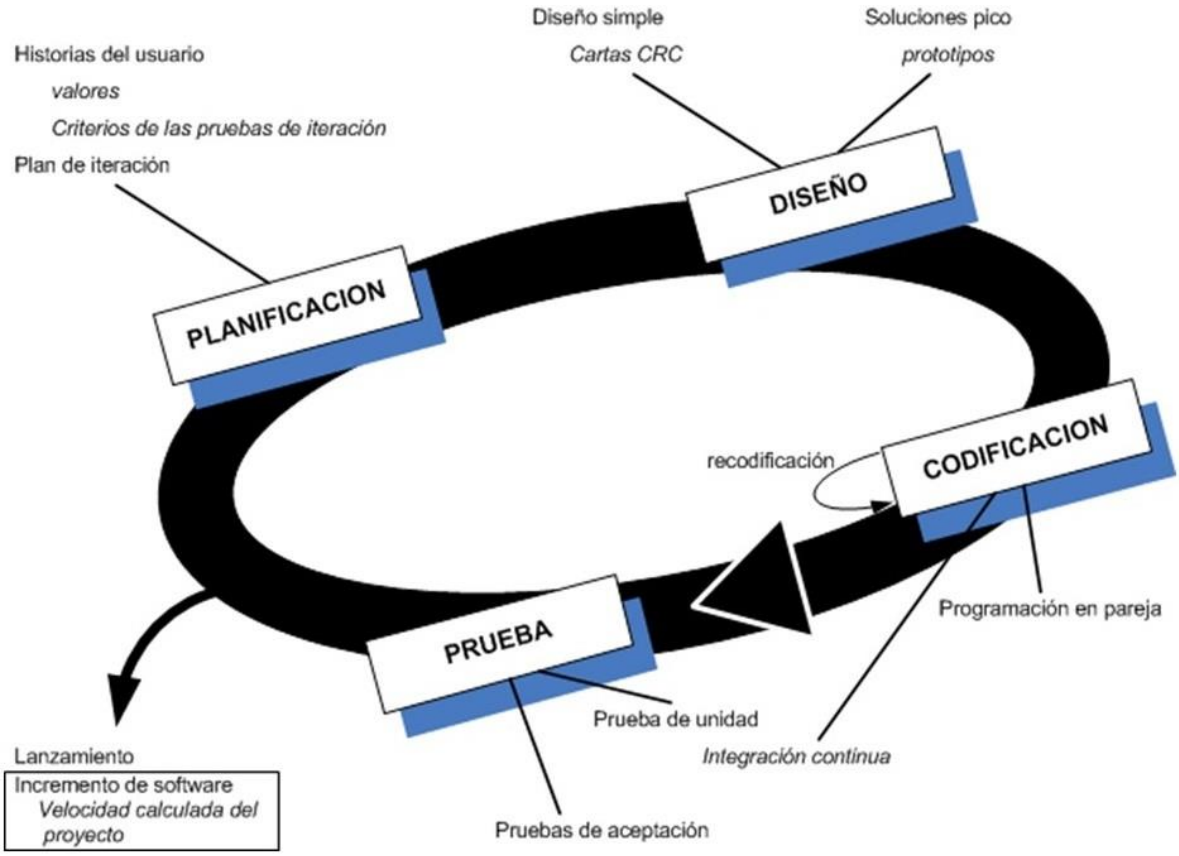


Figura 4: Proceso de la Programación Extrema (Pressman, 2005)

- Planificación: Esta actividad comienza creando una serie de historias (llamadas historias del usuario) que describen las características y funcionalidades requeridas para el software que se construirá. Cada historia es escrita por el cliente, el cual le asigna un valor (es decir, una prioridad) basándose en los valores generales del negocio respecto de la característica o la función y es colocada en una carta índice.
- Diseño: Esta fase sigue de manera rigurosa el principio de mantenerlo simple, se prefiere un diseño simple respecto de una representación más

compleja, el diseño ofrece una guía de implementación para una historia tal cual como está escrita.

- Codificación: La programación extrema recomienda que después de diseñar las historias y realizar el trabajo de diseño preliminar, el equipo no debe moverse hacia la codificación, sino que debe desarrollar una serie de pruebas de unidad que ejerciten las historias que vayan a incluirse en el lanzamiento actual (incremento de software). Una vez creada la prueba de unidad, el desarrollador conoce su alcance y puede centrarse en lo que debe implementar para poder pasarla.
- Pruebas: Las pruebas de unidad deben implementarse con un marco de trabajo que permita automatizarlas, para que puedan ejecutarse de manera fácil y repetida), que apoya una estrategia de regresión de prueba cuando el código se modifica. Una vez que el código está completo, la unidad puede probarse de inmediato, y así proporcionar una retroalimentación instantánea a los desarrolladores.

El Modelado Ágil (AM, por sus siglas en inglés) fue propuesto por Scott Ambler no como un método ágil, sino como complemento de otras metodologías o como un enfoque de desarrollo; por ejemplo en el caso de XP, podría ayudar a los practicantes a definir mejor los procesos de modelado que falten. En el modelado ágil se presenta una colección de prácticas, guiadas por principios y valores, que puede ser aplicada por los profesionales del software en un proyecto de desarrollo, para el modelado del sistema de una manera efectiva y ágil (Ambler, 2002).

Dado que AM debe ser usado como complemento de otras metodologías, se especifica muy poco sobre métodos de desarrollo, tamaño del equipo, roles, duración de iteraciones, trabajo distribuido y criticidad, todo lo cual dependerá del método que se utilice. En general, debe cumplirse con las siguientes etapas: definición de requerimientos y del estado inicial y de la arquitectura, modelado de

la iteración, formulación preliminar de modelos, desarrollo basado en pruebas y revisiones (Figura 5: Ciclo de vida para el desarrollo basado en el Modelado Ágil).

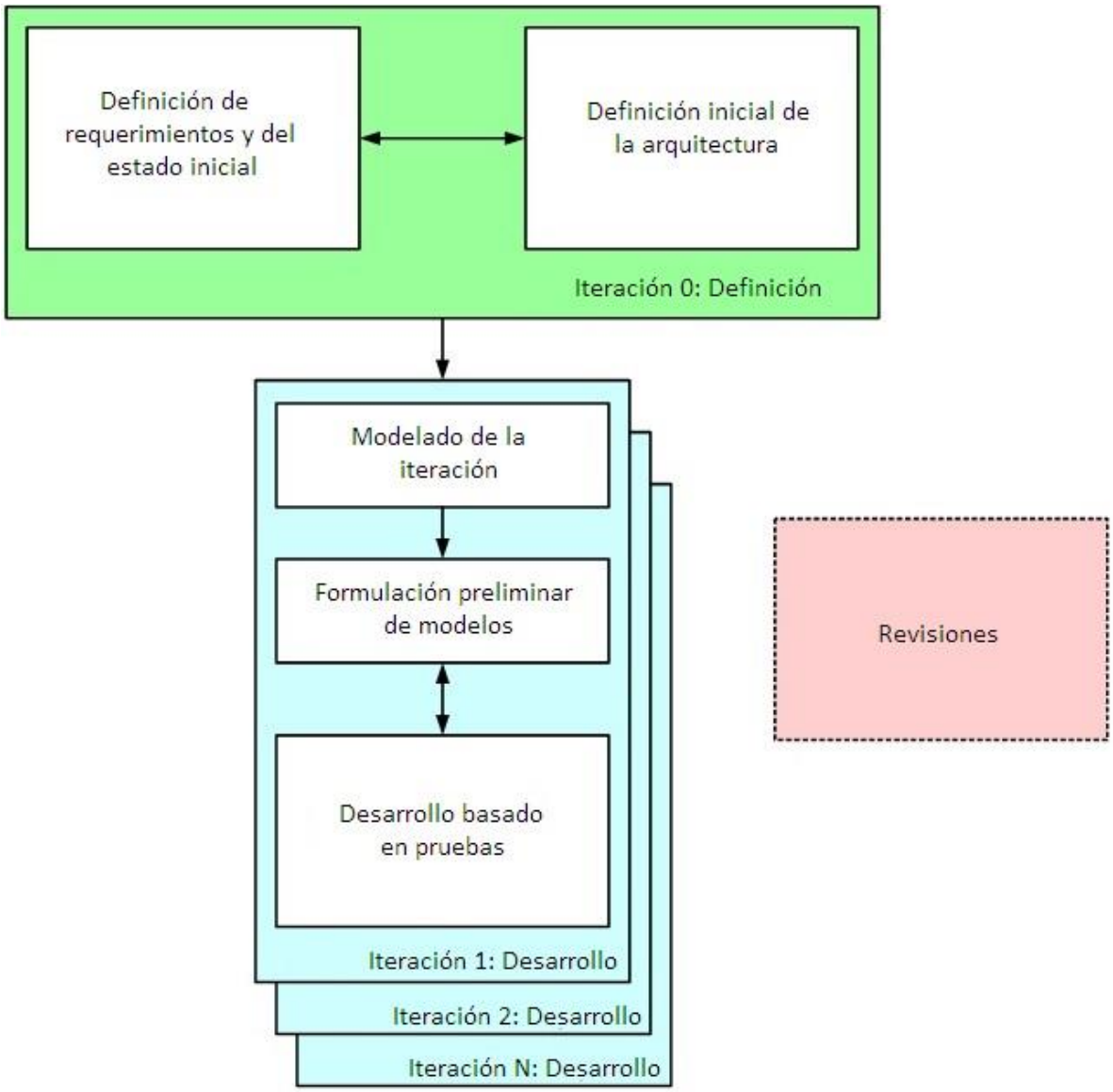


Figura 5: Ciclo de vida para el desarrollo basado en el Modelado Ágil (Ambler, 2002)

- Definición de requerimientos y del estado inicial: Para poder desarrollar un software es necesario identificar los requerimientos de alto nivel y el alcance de la liberación (lo que se piensa que el sistema debe realizar, una vez que esté concluido).El objetivo principal es conocer las necesidades de la herramienta final.
- Definición inicial de la arquitectura: Desde el punto de vista de la arquitectura es necesario crear diferentes diagramas que exploren las

técnicas de infraestructura, los modelos de dominio iniciales para explorar las diferentes entidades involucradas y sus relaciones y, opcionalmente, explorar posibles requerimientos potenciales o cambios en los requerimientos existentes que afecten la arquitectura que el sistema puede necesitar. En pasos posteriores los requerimientos y los modelos arquitectónicos iniciales tendrán que evolucionar a medida que se conoce más el proyecto, pero el objetivo principal es tratar de identificar una arquitectura que tenga una buena posibilidad de funcionar, para organizar al equipo de desarrollo (algo que es particularmente importante con equipos grandes).

- Modelado de la iteración: Al inicio de cada iteración, el equipo debe planificar el trabajo que van a realizar. Los equipos de desarrollo ágil deben implementar los requerimientos hechos por el usuario asignándoles una prioridad y enfocándose en los de mayor importancia.
- Formulación preliminar de modelos - Modelado al momento: Las sesiones de modelado suelen ser al momento, ya que un miembro del equipo del proyecto identifica un problema que desea modelar y le pedirá a uno o dos compañeros que lo ayuden con ello y, por lo general, durará entre cinco y diez minutos (no suelen tardar más de treinta minutos). El grupo se reúne alrededor de una herramienta de modelado compartida, por ejemplo una pizarra, para explorar el problema planteado hasta que sea comprendido y finalmente todos continúan con sus labores (a menudo codificando). Los programadores extremos llamarían a estas sesiones de modelado como sesiones de diseño informales o sesiones de preguntas y respuestas con el cliente.
- Desarrollo basado en pruebas: Antes de empezar con el proceso se crea una prueba rápidamente (ya sea de desarrollo o de aceptación), para luego programar solamente lo necesario para aprobarla. Seguidamente se lleva a cabo la verificación del software teniendo en cuenta la prueba establecida, y en caso de aprobarla, se repiten estos pasos hasta terminar; en caso contrario se procede a realizar los cambios necesarios en el código y a

repetir la verificación. Una vez concluido el desarrollo, se procede a la inspección y optimización del código.

- Revisiones: Opcionalmente se puede optar por realizar revisiones al modelo e incluso inspecciones de código, pero estas técnicas de garantía de calidad realmente parecen ser obsoletas para el desarrollo ágil de software cuando se trata de un equipo de proyecto con pocos integrantes. Con grupos más grandes, o en situaciones de mayor complejidad, las revisiones pueden ser de gran ayuda, puesto que proporcionan una excelente respuesta a los esfuerzos aplicados a las tecnologías de información.

2. Técnicas de prueba de software

Durante el desarrollo de un software, así como al terminarlo, es necesario realizarle una comprobación llevando a cabo el proceso de Verificación y Validación (V&V). Esta etapa comienza con verificación de los requerimientos y continúa con revisiones de diseño e inspecciones de código hasta la prueba del producto final (Sommerville, 2005).

Básicamente existen dos maneras de probar cualquier producto construido: 1) si se conoce la función específica para la que se diseñó se aplican pruebas que demuestren que cada función es plenamente operacional, mientras se buscan los errores de cada función; 2) si se conoce el funcionamiento interno se aplican pruebas para asegurarse de que “todas las piezas encajan”; es decir, que las operaciones internas se realizan teniendo en cuenta las especificaciones, y que se han probado todos los componentes internos de manera adecuada. Al primer enfoque de prueba se le denomina prueba de caja negra; mientras que al segundo, prueba de caja blanca[4] (Pressman, 2005).

La prueba de caja blanca no es una opción frente a las técnicas de caja negra. Es, en cambio, un enfoque complementario que tiene probabilidades de

[4](Los términos prueba funcional y prueba estructurada suelen usarse en lugar de prueba de caja negra y de caja blanca, respectivamente).

descubrir una clase diferente de errores de los que se descubrirían con los métodos de caja negra.

Esta clasificación de las pruebas de software es extremista y obsoleta ya que está pensada para verificar y validar el software desarrollado cuando este era de gran complejidad. En la nueva visión que se tiene sobre las pruebas, la ingeniería del sistema define el papel del software y lleva al análisis de los requisitos de este, donde se establecen el dominio de la información, la función, el comportamiento, el desempeño, las restricciones y los criterios de validación del software. Según esta nueva visión además de las pruebas de caja blanca y de caja negra existen las pruebas unitarias, las pruebas de integración y las pruebas de alto nivel (Figura 6: Tipos de prueba de software en la actualidad).

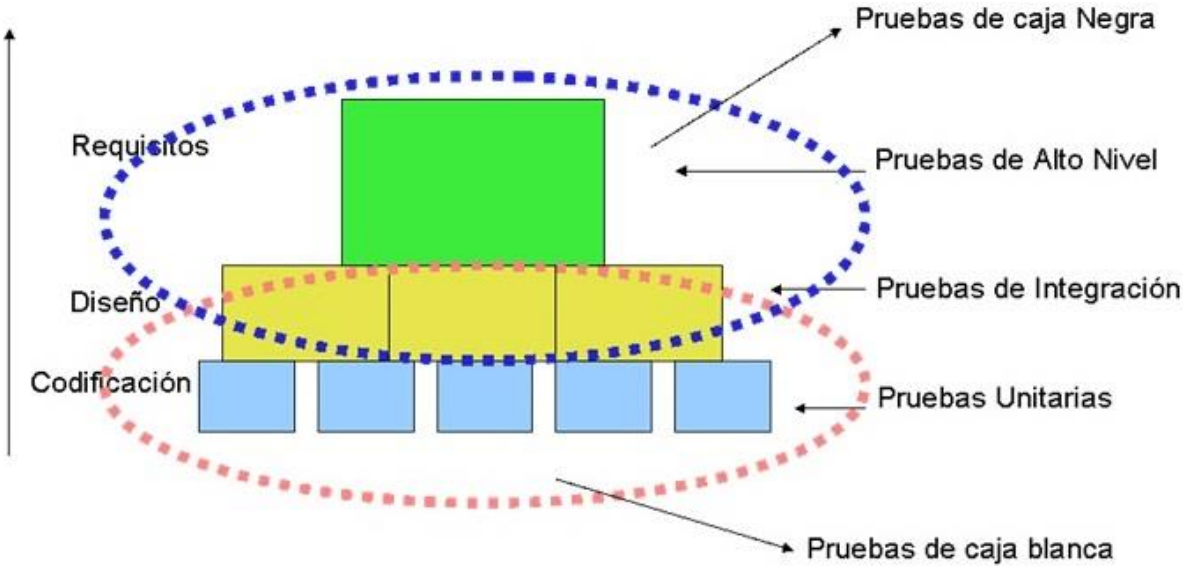


Figura 6: Tipos de prueba de software en la actualidad (Pressman, 2005)

Además de las pruebas pueden realizarse inspecciones de software las cuales son un proceso de verificación y validación estático, en el que un sistema software se revisa para encontrar errores, omisiones y anomalías. Generalmente las inspecciones se centran el código fuente, pero puede inspeccionarse cualquier representación legible del software como los requerimientos o un modelo de diseño. (Sommerville, 2005).

2.1. Pruebas de caja blanca

La prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear los métodos de prueba de caja blanca, el ingeniero del software podrá derivar casos de prueba que garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez, ejerciten los lados verdadero y falso de todas las decisiones lógicas, ejecuten todos los bucles en sus límites y dentro de sus límites operacionales, y ejerciten estructuras de datos internos para asegurar su validez. Su principal objetivo es asegurar la validez de todas las estructuras internas de los datos (Pressman, 2005).

Una de las pruebas de caja blanca más conocida es la de caminos, la cual es una estrategia de pruebas estructurales cuyo objetivo es probar cada camino de ejecución independiente en un componente o software. El punto de partida de una prueba de caminos es un grafo de flujo del programa, el cual es un modelo del esqueleto de todos los caminos en el software y consiste en nodos que representan decisiones y aristas que muestran el flujo del control. El objetivo de las pruebas de caminos es asegurar que cada camino independiente del programa se ejecuta al menos una vez. Un camino independiente del programa es aquel que recorre al menos una nueva arista en el grafo de flujo, es decir; que ejecuta una o más condiciones nuevas (Sommerville, 2005).

2.2. Pruebas unitarias

Las pruebas unitarias se concentran en cada componente del software, tal como se implementó en el código fuente, sirviendo así para asegurar que cada uno de los módulos funcione correctamente por separado. La idea es escribir casos de prueba para cada función no trivial o método en el módulo, de forma que cada caso sea independiente del resto. El objetivo principal es descubrir defectos probando los componentes de programas individualmente (Sommerville, 2005).

Estas pruebas realizadas por los desarrolladores se basan normalmente en una comprensión intuitiva de cómo los componentes deberían operar, y principalmente se utilizan para encontrar precedencia aritmética incorrecta, inicializaciones incorrectas, aritméticas incorrectas, falta de precisión, representaciones simbólicas incorrectas de expresiones y bucles inadecuados.

2.3. Pruebas de integración

Durante las pruebas de integración se construye el sistema a partir de sus componentes para comprobar que estos realmente funcionan juntos, son llamados correctamente y transfieren los datos correctos en el tiempo preciso a través de sus interfaces. Esta prueba consiste en identificar grupos de componentes que proporcionan alguna funcionalidad del sistema e integrarlos añadiendo código para hacer que estos funcionen conjuntamente. Existen dos tipos de pruebas de integración, cuando primero se desarrolla el esqueleto del sistema en su totalidad, y se le añaden los componentes se denomina integración descendente; y cuando primero se integran los componentes de infraestructura que proporcionan servicios comunes, tales como el acceso a la base de datos y redes, y a continuación se añaden los componentes funcionales se denomina integración ascendente (Sommerville, 2005).

Básicamente es la fase de prueba del software en la cual los módulos individuales de software son combinados y probados como un grupo. Su objetivo es poder asegurar el correcto funcionamiento de los componentes integrados que forman el sistema en cuestión.

2.4. Pruebas de alto nivel

Las pruebas de alto nivel o de validación empiezan tras la culminación de las pruebas de integración, cuando se han ejercitado los componentes individuales, se ha terminado de ensamblar el software como paquete y se han descubierto y corregido los errores de interfaz. La prueba se concentra en las acciones visibles para el usuario y en la salida del sistema que este puede reconocer (Pressman, 2005).

La validación del software se logra mediante una serie de pruebas que demuestran que se cumple con los requisitos. Un plan de prueba delinea la clase de pruebas que se aplicarán, y un procedimiento de prueba define los casos de prueba específicos. Para cumplir con las pruebas de alto nivel es necesario realizar:

- Revisión de configuración: Su objetivo es asegurar que todos los elementos de la configuración del software se hayan desarrollado apropiadamente, estén catalogados y tengan el detalle suficiente para reforzar la fase de soporte del ciclo de vida del software.
- Pruebas alfa y beta: Son utilizadas cuando el software desarrollado será usado por muchos clientes y no es eficiente realizar pruebas de aceptación con cada uno de ellos. Los usuarios finales son quienes aplican la prueba alfa en el lugar de trabajo del desarrollador (un entorno controlado), mientras que estos registran los errores y los problemas de uso. Las pruebas beta se aplican en el lugar de trabajo de los usuarios finales (sin la presencia del desarrollador, generando un entorno no controlado) y estos son los encargados de registrar todos los problemas que encuentre durante la prueba, los cuales informará posteriormente al desarrollador.
- Pruebas de recuperación: Es una prueba del sistema que obliga al software a fallar de varias maneras y a verificar que la recuperación se realice apropiadamente, en un tiempo determinado.
- Pruebas de seguridad: Compruebas que los mecanismos de protección integrados en el sistema realmente lo protejan de irrupciones inapropiadas.
- Pruebas de stress: En esta prueba se ejecuta el sistema de tal manera que requiera una cantidad, una frecuencia o un volumen anormal de recursos. En esencia, la persona que aplica la prueba tratará de sobrecargar el software.
- Pruebas de rendimiento: Está diseñada para probar el desempeño del software en tiempo de ejecución dentro del contexto de un sistema

integrado, a pesar de que se aplica en todos los pasos del proceso de la prueba del software.

- Pruebas de usabilidad: Estas pruebas consisten en seleccionar a un grupo de usuarios y solicitarles que lleven a cabo las tareas para las cuales fue diseñada esta herramienta, en tanto el equipo de diseño, desarrollo y otros involucrados toman nota de la interacción, particularmente de los errores y dificultades con las que se encuentren los usuarios.

2.5. Pruebas de caja negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se concentran en los requisitos funcionales del software; es decir, permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales del sistema. Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías: funciones incorrectas o faltantes, errores de interfaz, errores en estructuras de datos o en acceso a bases de datos externas, errores de comportamiento o desempeño y errores de inicialización y término.

Al aplicar técnicas de caja negra se derivan un conjunto de casos de prueba que satisfacen los siguientes criterios: 1) casos de prueba que reducen, mediante una cuenta mayor que uno, el número de casos de prueba adicionales que deben diseñarse para alcanzar una prueba razonable, y 2) casos de prueba que indican algo acerca de la presencia o ausencia de clases de errores, en lugar de un error asociado sólo con la prueba específica a la mano (Pressman, 2005).

Una de las pruebas de caja negra que se pueda realizar es la evaluación de usabilidad basada en heurísticas. Jakob Nielsen “el gurú de la usabilidad” estudió 249 problemas de usabilidad y a partir de ellos diseñó lo que denominó las “reglas generales” para identificar los posibles problemas de usabilidad (Nielsen, 1995), las cuales se definen a continuación:

- Visibilidad del estado de sistema: El sistema siempre debe mantener informado a los usuarios sobre lo que está sucediendo y proporcionarle respuesta en un tiempo razonable.
- Adecuación entre el sistema y el del mundo real: El sitio web o aplicación debe utilizar el lenguaje del usuario, con expresiones y palabras que le resulten familiares. La información debe aparecer en un orden lógico y natural.
- Libertad y control por parte del usuario: Los usuarios eligen funciones del sistema por error y necesitarán a menudo una “salida de emergencia claramente marcada” para dejar el estado indeseado sin tener que pasar por un diálogo extendido. Debe disponer también de la capacidad de deshacer o repetir una acción realizada.
- Consistencia y estándares: Los usuarios no tienen por qué saber que diferentes palabras, situaciones o acciones significan lo mismo. Es conveniente seguir convenciones.
- Prevención de error: Es importante prevenir la existencia de errores mediante un diseño adecuado. Aun así, los mensajes de error deben incluir una confirmación antes de ejecutar las acciones de corrección.
- Reconocimiento antes que recuerdo: Reducir al mínimo la carga cognitiva por parte del usuario haciendo visibles objetos, acciones, y las opciones. El usuario no debe tener que recordar la información entre distintas secciones o partes del sistema. Las instrucciones para el uso del sistema deben ser visibles o fácilmente recuperables siempre que sea necesario.
- Flexibilidad y eficacia del uso: Los aceleradores, no vistos por usuarios principiantes, pueden incrementar a menudo la interacción para el usuario experto, de tal forma que el sistema sea útil tanto para usuarios noveles como avanzados. Se debe permitir a los usuarios configurar acciones frecuentes con atajos de teclado.
- Diseño estético y minimalista: Los diálogos no deben contener información que es irrelevante o poco necesaria. Cada unidad adicional de la

información en un diálogo compite con las unidades relevantes y disminuye su visibilidad relativa.

- Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores: Los mensajes de error deben expresarse en un lenguaje común y sencillo, indicando con precisión el problema y sugiriendo las posibles alternativas o soluciones.
- Ayuda y documentación: Aunque es mejor que el sistema se puede utilizar sin la documentación, puede ser necesario proveer cierto tipo de ayuda. Dicha información debería ser fácil de buscar, estar enfocada en las tareas del usuario, con una lista de los pasos a seguir y no ser muy extensa.

3. Mantenimiento del software basado en Programación Extrema y el Modelado Ágil

Teniendo en cuenta los conceptos explicados anteriormente, la opinión de algunos autores que indican que el mantenimiento es parte del desarrollo y las características que cumplen los procesos ágiles de desarrollo, especialmente en los casos de XP y Modelado Ágil; se procede a explicar la metodología que se siguió para realizar el mantenimiento del software de manera ágil, en la cual se explican las actividades viendo este proceso de mantenimiento bajo un enfoque evolutivo (Figura 7: Actividades del mantenimiento del software ágil).

La propuesta metodológica está basada en un desarrollo iterativo incremental en el mismo sentido que XP y MA, implicando realizar un conjunto de actividades agrupadas en pequeñas iteraciones.

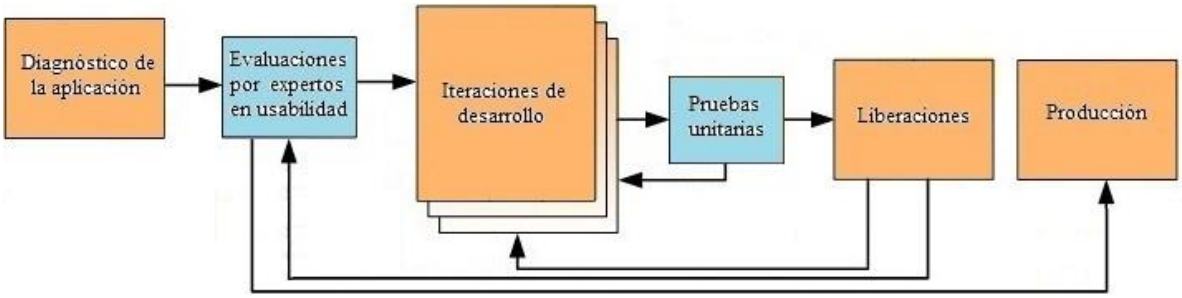


Figura 7: Actividades del mantenimiento del software ágil (Hussain, y otros, 2012)

Las actividades asociadas a la metodología ágil propuesta para llevar a cabo el mantenimiento del software son las siguientes:

- Diagnóstico de la aplicación: Esta es la actividad de inicio, en ella se realiza una inspección del software a través de la técnica de comprensión del programa, explicada en la sección 1.4 del presente Trabajo Especial de Grado, para poder conocer la situación actual y lograr estudiar tanto la posible documentación como la estructura completa, incluyendo los comentarios a lo largo del código del sistema. Una vez finalizada esta fase deben conocerse tanto la factibilidad de recuperar algunos módulos que comprenden a la versión actual del software, como la técnica y/o proceso a seguir para realizar el mantenimiento.
- Evaluaciones por expertos en usabilidad: En esta actividad se realizan pruebas de caja negra, utilizando herramientas de evaluación basadas en heurísticas con expertos de ingeniería del software. Se realizaron dos (2) evaluaciones de este tipo, la primera de ellas tuvo el objetivo de evaluar el estado inicial del sistema; mientras que el propósito de la segunda fue calcular el avance, luego de haber aplicado la propuesta metodológica.
- Iteraciones de desarrollo: En esta actividad se procede a realizar el levantamiento de los requerimientos con el cliente para posteriormente estudiar cada uno de estos, para así poder implementarlos modificando el código del sistema y actualizando la documentación actual.
- Pruebas unitarias: En esta actividad el encargado de llevar a cabo el proceso de mantenimiento del software realiza pruebas de caja blanca basadas principalmente en la comprensión del programa y la reingeniería del software; estas pruebas se realizan para verificar la estructura y el funcionamiento del requerimiento desarrollado en esta iteración.
- Liberaciones: Esta actividad corresponde a entregas de software que se realizan luego de aprobarse las pruebas unitarias y la posterior reunión de certificación que se realiza conjuntamente con el cliente, para que este pueda verificar los avances en el mantenimiento de la herramienta. Al finalizar esta actividad se obtiene un incremento en la versión del software,

es decir, se va optimizando la calidad de la herramienta a costa de la complejidad que implica el desarrollo asociado.

- Producción: Esta es la última actividad y corresponde con la entrega final de la herramienta, una vez que se haya concluido con el proceso de mantenimiento del software y se satisfagan todos los requerimientos del usuario. Para llegar a esta actividad de la metodología es necesario que el software pase por rigurosas pruebas que garanticen el correcto funcionamiento, la calidad del mismo y el grado de usabilidad.

4. Explicación del Caso de Estudio RubricArte y su Contexto Asociado

Las rúbricas son instrumentos de medición cualitativa, en los cuales se establecen criterios y estándares por niveles, mediante la disposición de escalas, que permiten determinar la calidad de la ejecución de los estudiantes en unas tareas específicas. Estas facilitan la calificación del desempeño del estudiante en las áreas del currículo (asignaturas o temas) que son complejas, imprecisas y subjetivas, realizándolo a través de un conjunto de criterios graduados, que permiten valorar el aprendizaje, los conocimientos y/o competencias logradas por el estudiante (García, 2012).

Por lo general se diseña de manera que el estudiante pueda ser evaluado en forma “objetiva y consistente”. Al mismo tiempo permite al profesor especificar claramente qué espera del estudiante y cuáles son los criterios con los que se va a calificar un objetivo que se ha establecido previamente: un trabajo, una presentación, un reporte escrito y todo aquello que esté de acuerdo con el tipo de actividad que desarrolle con los alumnos (García, 2012).

4.1. Sobre la Herramienta

Fue desarrollada por la Lic. María Gracia Fernández para su Trabajo Especial de Grado a nivel de pregrado en el Centro de Ingeniería de Software y Sistemas (ISYS) de la Escuela de Computación de la Universidad Central de

Venezuela, y llevó por nombre RUBRICARTE: Una Herramienta para Evaluación de Aprendizajes a través de Rúbricas. El alcance estaba dado por el desarrollo del repositorio y los servicios de gestión, aun cuando se presentan los primeros prototipos funcionales de la evaluación de aprendizajes, se considera este producto como una versión Beta.

RubricArte es una herramienta que permite, de forma sencilla, la creación y administración de rúbricas, además de llevar un registro de evaluaciones realizadas o bien por el docente, o por los propios alumnos. El proceso de desarrollo de esta aplicación fue llevado a cabo a través del Modelado Ágil, un método que por su flexibilidad permite mejorar las especificaciones del usuario, sin necesidad de hacer cambios radicales sobre todo el proyecto (García, 2012).

RubricArte es una herramienta colaborativa que facilita el desarrollo, uso y divulgación de rúbricas, donde se promueve el libre acceso a éstas a través de un repositorio abierto, permitiendo la evaluación, co-evaluación y autoevaluación de grupos de estudio (docente y estudiantes) que comparten la misma visión del proceso de enseñanza-aprendizaje.

RubricArte permite a los docentes crear un proceso de evaluación a través de la gestión de rúbricas, la cual se hace necesaria debido a que facilita su proceso de creación, uso y reutilización, tanto para personas expertas como para aquellas con poca o ninguna experiencia. En particular, presenta las ventajas adicionales de ser totalmente gratuita, en español y permitir la reutilización total o parcial de las rúbricas; ya sean previamente elaboradas, propias o de otros usuarios. Al mismo tiempo que permite al docente realizar la evaluación de los estudiantes utilizando las creadas, y a ellos autoevaluarse y evaluar a sus compañeros. Es precisamente el aspecto del ambiente colaborativo que presenta la mayores fallas desde su implementación.

En resumen, esta herramienta provee dos estructuras de datos con protocolos de acceso bien definidos: el repositorio de rúbricas y el espacio para la evaluación. El proceso de construcción de una rúbrica requiere operaciones para crear matrices de contenidos permitiendo que las dimensiones sean variables, las rúbricas son compartidas en una comunidad pero el propietario (quién la creó) sólo puede eliminarla del repositorio. La creación de una rúbrica puede ser partiendo desde una matriz vacía o modificando una ya existente, que se guardará con un nuevo nombre.

El espacio para realizar la evaluación en forma colaborativa requiere definir una estructura que permita almacenar las evaluaciones realizadas en base a una rúbrica determinada, mantener controles sobre el proceso de evaluación y la sincronización de las evaluaciones. Otra funcionalidad es la visualización de las evaluaciones de forma gráfica.

RubricArte fue seleccionada como caso de estudio para el presente Trabajo Especial de Grado dado que durante su desarrollo se utilizó Modelado Ágil y en su posterior puesta en marcha se detectaron varios problemas asociados, como lo son la estabilidad y las habilidades del equipo y la poca participación con el usuario; haciendo que la complejidad asociada a la fase de mantenimiento del software creciera en cierta medida.

4.2. Sobre el Desarrollo

Como arquitectura de la aplicación se seleccionó el modelo MVC (Model/View/Controller) en su versión de tres capas: la capa de datos, la vista del usuario y la lógica de negocio respectivamente (Figura 8: Arquitectura de capas de la aplicación RubricArte).

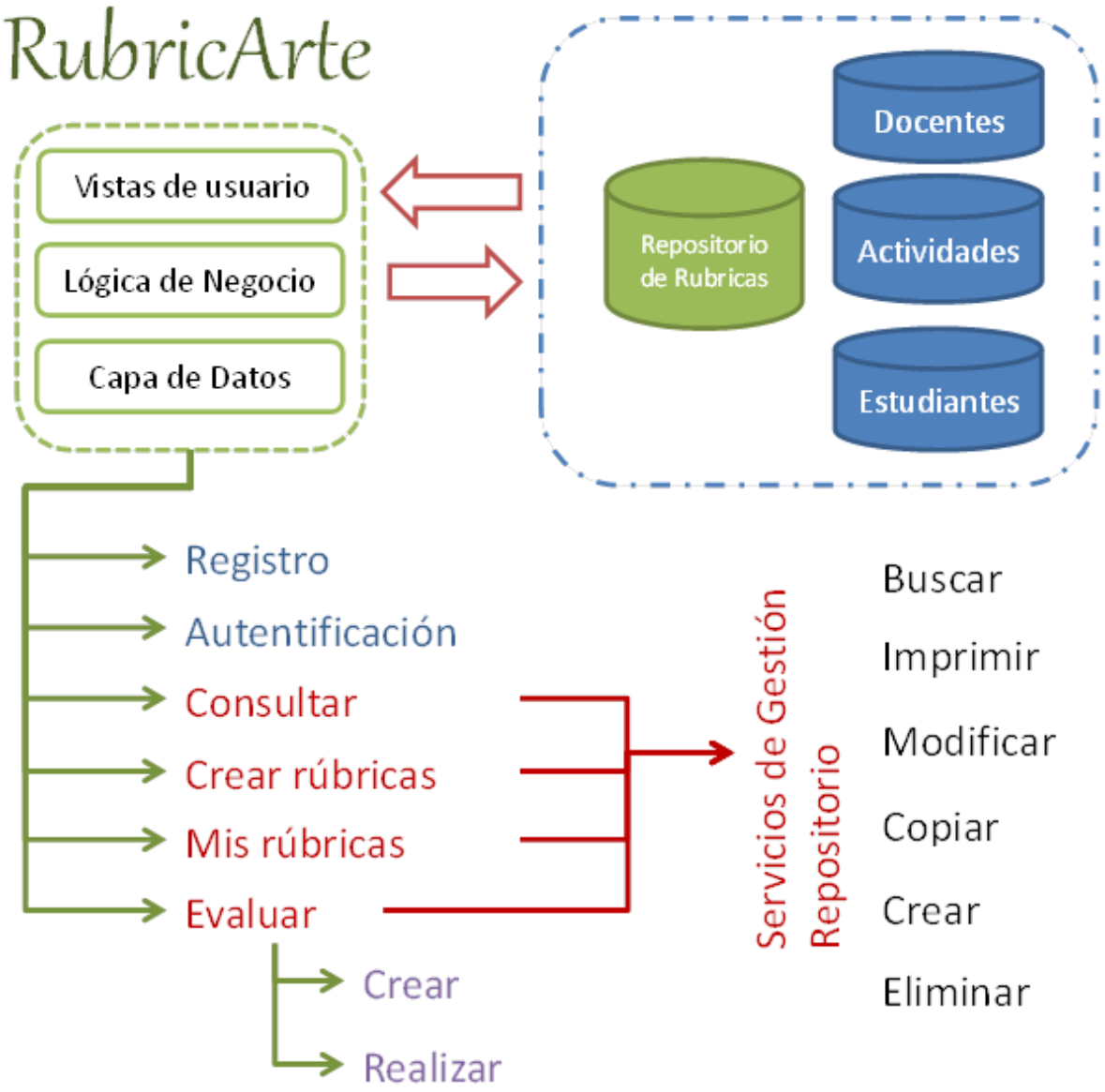


Figura 8: Arquitectura de capas de la aplicación RubricArte (García, 2012)

Luego de aplicar esta arquitectura se diseñaron diferentes funcionalidades en la aplicación para la gestión de las rúbricas con sus escenarios de uso asociados, entre las principales tenemos (Tabla 2: Funcionalidades de gestión de rúbricas con los escenarios de uso asociado):

Funcionalidad	***Escenarios de Uso Asociados***	***Descripción***
Consultar Rúbricas	Listar todas las rúbricas → Seleccionar → Imprimir	Se navega a través del listado de todas las rúbricas para proceder a seleccionar una para su visualización, opcionalmente se puede proceder a la impresión de la rúbrica.

	Buscar rúbrica → Seleccionar → Imprimir	Se provee de un conjunto de parámetros (Id, nombre, autor, etiquetas) para la visualización de una o más rúbricas.
Crear Rúbricas	Crear una rúbrica a partir de una plantilla	Se despliega un ambiente matricial para la captura de información.
	Crear una rúbrica a partir de otra ya existente	Se despliega un ambiente matricial con la información de la rúbrica origen, pero al terminar se debe guardar la nueva con un nombre diferente.
Mis Rúbricas	Modificación de la rúbrica editando cada uno de sus campos	Sólo pueden modificarse las rúbricas en el espacio privado del usuario.
	Establecer el nivel de privacidad de la rúbrica	Se proveen dos niveles: Públicas y privadas.
	Eliminación de Rúbricas	La eliminación sólo se permite por parte del docente que creó la rúbrica.

Tabla 2: Funcionalidades de gestión de rúbricas con los escenarios de uso asociado (García, 2012)

4.3. Sobre la Investigación

Rubricarte es producto de software resultado del proyecto de investigación "Modelo de Enseñanza Colaborativa Basado en la Web 2.0 Para el Fortalecimiento de la Enseñanza de la Ciencia y la Tecnología", financiado por el Fondo Nacional de Ciencia, Tecnología e Innovación (FONACIT) del Ministerio del Poder Popular para la Educación Universitaria, Ciencia y Tecnología de la República Bolivariana de Venezuela. Entre algunos objetivos que persigue el proyecto se encuentra:

- Desarrollar un modelo educativo colaborativo basado en la web2.0 y las redes sociales que permita la creación de ambientes y comunidades virtuales de aprendizaje en ciencia y tecnología.
- Diseñar soluciones tecnológicas "open source" que permitan desarrollar las comunidades virtuales de aprendizaje y el almacenaje de los proyectos formulados por estas comunidades.

Rubricarte representa una de las soluciones tecnológicas que apoyan el modelo educativo producto de esta investigación.

Los requerimientos considerados en el proyecto son:

I. Plataforma social de interacción dotada de herramientas de la Web 2.0: foros, blog, fotos, muro, chat, video-conferencia, entre otras.
II. Herramientas de comunicación asíncronas para la generación de conocimiento tipo tormentas de ideas.
III. Estrategias de aprendizaje constructivista: basadas en proyecto, problemas o estudio de casos.
IV. Repositorios de conocimientos basados en la teoría de aprendizaje generativo: objeto de aprendizaje, experiencias de aprendizaje y rúbricas. Se requiere para la gestión de conocimiento mecanismos inteligentes que faciliten las búsquedas y relacionar el conocimiento entre repositorios.

Los participantes del proyecto son:

Coordinadores - Investigadores	***Correo Electrónico***	***Institución***
Vanessa Miguel (Coordinadora)	vanessa.miguel@ucv.ve	Universidad Central de Venezuela
Nora Montaño (Coordinadora Subproyecto)	nora.montano@ciens.ucv.ve	Universidad Central de Venezuela
Mariano Fernández (Coordinador Subproyecto)	mariano.fernandez@ucv.ve	Universidad Central de Venezuela
Feliciana A Victoria Lucci (Investigadora)	vlucci@usb.ve victorialucci@gmail.com	Universidad Simón Bolívar
José Miguel Flores (Investigador)	jmiguelfs79@gmail.com	Universidad Central de Venezuela
Keybell Díaz (Investigadora)	kdiazg@gmail.com	Universidad Central de Venezuela
Isis Landaeta (Investigadora)	isislanda@gmail.com	Universidad Central de Venezuela

Tabla 3: Coordinadores - Investigadores del Proyecto

5. Aplicación de las Dos (2) Primeras Actividades de la Metodología Ágil Propuesta

Para poder llevar a cabo el mantenimiento del software se requirió investigar la documentación, la estructura y los comentarios a lo largo del código del sistema, así como también obtener el estado inicial de este; por estas razones es que se procedió a cumplir con las dos (2) primeras actividades descritas en la propuesta realizada.

5.1. Diagnóstico Inicial de la Aplicación:

Esta primera actividad del modelo propuesto fue llevada a cabo realizando una inspección del software aplicando la técnica de comprensión del programa explicada en la sección 1.4, con lo cual se pudo observar que el código fuente no se encontraba comentado internamente e incluso la documentación general del software era muy poca, por lo que esta actividad requirió de una mayor cantidad de tiempo que el planificado al momento de realizar la propuesta metodológica.

Además de lo anteriormente expuesto, se pudo evidenciar que al momento de desarrollar la herramienta RubricArte no se pensó en realizar un código que fuera fácilmente entendible, consistente y legible para quien se encargara posteriormente de la fase de mantenimiento, demostrando que se presentaron algunos de los problemas descritos en la sección 1.1 (las actividades del desarrollo se suelen realizar bajo presión de tiempo, la poca participación del usuario durante el desarrollo del sistema y por la falta de interés en el tema).

5.2. Evaluaciones por expertos en usabilidad:

Luego de realizar el diagnóstico inicial de la aplicación se procedió a concluir con el proceso de levantamiento de información y a llevar a cabo el reconocimiento del estado inicial del sistema; razón por la cual un grupo de expertos en usabilidad conformado por diez (10) estudiantes del Postgrado de Ciencia de la Computación, cursantes de la asignatura Interacción Humano Computador, con conocimientos generales en el desarrollo de aplicaciones de calidad bajo los principios de la Ingeniería del Software y familiarizados con las técnicas de evaluación de usabilidad basadas en heurísticas, realizaron las siguientes tareas:

- Navegaron a través del sistema para tratar de conocer los diferentes aspectos y funcionalidades disponibles en este, durante un tiempo determinado.

- Respondieron una encuesta basada en las heurísticas de Nielsen, (Anexo I: Evaluación Heurística Realizada a RubricArte). Entre los principales comentarios obtenidos se encuentran los siguientes:
    - Al iniciar sesión no existe ninguna opción que permita al usuario recuperar la clave. En caso de que el usuario olvide la contraseña no podrá entrar más, a menos que solicite el cambio al administrador, pero no veo en la página ninguna forma de contactarlo.
    - La opción de "evaluar" que se encuentra en la página principal no tiene un enlace, como lo tiene las otras opciones.
    - Puedo hacer la evaluación de una sola rúbrica: Actividad de evaluación heurística a RubricArte, #90- Evaluación heurística realizada a RubricArte.
    - ¿Existe la posibilidad de colocar una descripción opcional a cada rúbrica? (así como hiciste en el documento que nos entregaste). Una de las cosas que veo más confuso de este tipo de evaluación es que a veces varía la interpretación de una persona a otra dependiendo de qué también redactado esté el título de la rúbrica. Quizás si se incluye una breve descripción como hiciste en el papel que nos entregaste, los usuarios podrían hacer una mejor evaluación, consultando esta descripción antes de evaluar.
    - Debes revisar algunos detalles con respecto al lenguaje de la base de datos o el html, pues hay algunos caracteres que no se leen bien en las rúbricas que ya están cargadas.
    - No pude probar la opción de creación de rúbricas, y no sé si verificas la información para evitar que se repitan las rúbricas. Lo comento porque viendo las que están registradas veo tres rúbricas relacionadas con la evaluación de tesis. ¿Esos fue así a propósito? ¿Eso no podría generar confusión después entre los usuarios? ¿Es irrelevante para el objetivo del sistema?
    - En general me parece que la aplicación es fácil de usar. El diseño es simple, lo cual permite al usuario navegar con facilidad sin mucha

carga cognitiva. El lenguaje es claro, aunque en algunas partes es un poco extenso. Emite los mensajes de error que corresponden, lo cual permite al usuario entender por qué puede o no hacer ciertas tareas, sin embargo, se podrían evitar algunos de esos mensajes si por ejemplo se quita la opción de crear rúbricas si el usuario no ha iniciado sesión todavía.

  - Si se guarda la rúbrica, luego se hace clic en regresar y nuevamente en guardar se crea una nueva rúbrica.
  - Los botones de agregar fila deberían estar en cada fila y los de agregar columna en cada columna.

- Respondieron un cuestionario con el fin de evaluar la satisfacción obtenida al momento de interactuar con la herramienta RubricArte (Anexo IV: Cuestionario de Satisfacción de RubricArte). Las preguntas realizadas, junto con los resultados obtenidos del cuestionario se muestran agrupadas por características, a continuación:
  - Aspectos de Usabilidad (Tabla 4: Puntuación Cuestionario Inicial Aspectos de Usabilidad):

Número de Pregunta	***Pregunta***	***Puntuación Obtenida (1 - 5)***	***Desv. Estándar***
1	¿El sistema indica qué está pasando en un tiempo razonable?	4,4	0,97
2	¿El sistema está escrito en un lenguaje de forma clara y concisa?	4,4	0,97
3	¿El sistema deja del lado del usuario la navegación y el control?	4,6	0,52
4	¿El sistema es consistente y sigue los estándares?	4,5	0,71
5	¿El sistema previene los errores?	4	1,05
6	¿Las herramientas que provee el sistema son fáciles de reconocer o deben ser recordadas? (Calificación mínima: Recordar / Calificación máxima: Reconocer)	4,4	0,70
7	¿El sistema es flexible y da un uso eficiente de la navegación?	4,5	0,71
8	¿El sistema tiene un diseño práctico y simple?	4,9	0,32
9	¿El sistema le ofrece ayudas ante situaciones complejas?	4	1,05
10	¿El sistema posee compatibilidad con respecto a sistemas operativos, navegadores web, etc.?	4	0,94

Tabla 4: Puntuación Cuestionario Inicial Aspectos de Usabilidad (n = 10)

En este aspecto resaltó positivamente el diseño práctico y simple, teniendo en cuenta los diferentes errores en la navegación y la escasa ayuda.

- Aspectos Generales (Tabla 5: Puntuación Cuestionario Inicial Aspectos Generales):

Número de Pregunta	*Pregunta*	*Puntuación Obtenida (1 - 5)*	*Desv. Estándar*
11	¿Es eficiente?	4,6	0,70
12	¿Es intuitivo?	4,3	0,82
13	¿Mantiene la consistencia (en funcionamiento y apariencia)?	4,6	0,70
14	¿Facilita que el usuario se sienta cómodo y con el control del sitio?	4,6	0,52
15	¿Tiene una URL correcta, clara y fácil de recordar, así como las URL internas?	4,1	0,88
16	¿Las URL son claras y permanentes?	4,1	0,99

Tabla 5: Puntuación Cuestionario Inicial Aspectos Generales (n = 10)

En este tema resaltaron la eficiencia, la consistencia y la satisfacción producida por el sistema, pero teniendo en cuenta los problemas que se presentan con las URL; lo cual pareció contradictorio.

- Identidad e Información (Tabla 6: Puntuación Cuestionario Inicial Identidad e Información):

Número de Pregunta	*Pregunta*	*Puntuación Obtenida (1 - 5)*	*Desv. Estándar*
17	¿La página de inicio muestra la naturaleza del negocio y se identifica bien el logotipo o marca?	4,2	0,79
18	¿Aparecen elementos de la imagen corporativa en todas las páginas?	4,4	0,52
19	¿Aparece el logo corporativo en un lugar importante dentro de la página?	4,3	0,82
20	¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?	3,3	1,34
21	En artículos, noticias, informes, etc. ¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?	3,6	1,43

Tabla 6: Puntuación Cuestionario Inicial Identidad e Información (n = 10)

En este aspecto resaltaron positivamente la consistencia corporativa del sistema, mientras que no sintieron confiabilidad con respecto al almacenamiento de los datos personales.

- Navegación (Tabla 7: Puntuación Cuestionario Inicial Navegación):

Número de Pregunta	*Pregunta*	*Puntuación Obtenida (1 - 5)*	*Desv. Estándar*
22	¿Aparece la navegación en un lugar prominente, donde se vea fácilmente?	4,4	0,84
23	¿Existen enlaces rotos o que no conducen a ningún sitio?	3,7	1,25
24	¿Tiene un mapa del sitio o un buscador para quienes quieren acceder directamente a los contenidos sin tener que navegar?	4	1,33
25	¿Se mantiene una navegación consistente y coherente a lo largo del site?	4,2	1,03
26	¿Existen elementos que permitan al usuario saber exactamente dónde se encuentra dentro del site y cómo volver atrás (breadcrumbs)?	3,9	1,10
27	¿Indican los enlaces claramente hacia dónde apuntan? ¿Está claro lo que el usuario encontrará detrás de cada uno?	4,3	1,16
28	¿Los enlaces son fácilmente reconocibles como tales? ¿Su caracterización indica su estado (visitados, activos, etc.)?	4,1	1,37
29	¿Es predecible la respuesta del sistema antes de visitar un enlace?	4	1,15

Tabla 7: Puntuación Cuestionario Inicial Navegación (n = 10)

En este aspecto existió discrepancia dado que puntuaron altamente a la dirección apuntada por los enlaces y su consistencia, contra la negativa de que existen enlaces rotos.

- Búsqueda (Tabla 8: Puntuación Cuestionario Inicial Búsqueda):

Número de Pregunta	*Pregunta*	*Puntuación Obtenida (1 - 5)*	*Desv. Estándar*
30	¿Se encuentra fácilmente accesible?	4,3	0,82
31	¿Es fácilmente reconocible como tal?	4,3	1,06
32	¿Permite la búsqueda avanzada?	3,9	1,10
33	¿Muestra los resultados de la búsqueda de forma comprensible?	4,1	0,88
34	¿La caja de texto es lo suficientemente ancha?	3,8	0,92
35	¿Asiste al usuario en caso de no poder ofrecer resultados?	3,5	1,18

Tabla 8: Puntuación Cuestionario Inicial Búsqueda (n = 10)

Con respecto a este punto, la máxima calificación la obtuvo la facilidad asociada a la búsqueda, mientras que la mínima fue la ayuda ofrecida al usuario cuando no se obtienen resultados.

- Contenidos (Tabla 9: Puntuación Cuestionario Inicial Contenidos):

Número de Pregunta	*Pregunta*	*Puntuación Obtenida (1 - 5)*	*Desv. Estándar*
36	¿Es coherente el contenido con el contexto de la página o site?	4,4	0,70
37	¿La redacción es corta y precisa?	4,6	0,84
38	¿Existen referencias cruzadas entre textos que están relacionados?	3,8	1,32

Tabla 9: Puntuación Cuestionario Inicial Contenidos (n = 10)

En este aspecto la máxima calificación la precisión de la redacción del sistema, mientras que la mínima fue para las referencias entre textos relacionados.

- Tecnología (Tabla 10: Puntuación Cuestionario Inicial Tecnología):

Número de Pregunta	*Pregunta*	*Puntuación Obtenida (1 - 5)*	*Desv. Estándar*
39	¿La tecnología utilizada en el site es compatible con el software y hardware de los usuarios objetivos? (No tendrán que descargar elementos como plugins para poder usarlo)	4,3	0,95
40	Si es importante utilizar recursos técnicos que requieran la descarga de plugins, ¿se le informa al usuario de esta situación y se le explica la importancia de hacerlo?	3,9	1,45

Tabla 10: Puntuación Cuestionario Inicial Tecnología (n = 10)

En este aspecto resaltó la calificación mínima, dado que no es necesario software alguno.

- Interfaz (Tabla 11: Puntuación Cuestionario Inicial Interfaz):

Número de Pregunta	*Pregunta*	*Puntuación Obtenida (1 - 5)*	*Desv. Estándar*
41	¿Tiene una interfaz amigable, con colores que concuerden con los objetivos y propósitos?	4,3	1,34
42	¿Hay espacios blancos (libres) entre el contenido, para descansar	4,3	0,95

	la vista?		
43	¿Se ve el sitio exactamente igual en diferentes navegadores?	3,6	1,51

Tabla 11: Puntuación Cuestionario Inicial Interfaz (n = 10)

En esta característica se esperaba tener una mejor calificación en la compatibilidad de los navegadores, dada la alta puntuación obtenida anteriormente en la compatibilidad de software y hardware.

- Feedback (Tabla 12: Puntuación Cuestionario Inicial Feedback):

Número de Pregunta	***Pregunta***	***Puntuación Obtenida (1 - 5)***	***Desv. Estándar***
44	¿Existen respuestas frente a interacciones del usuario? (ej., se le informa que se ha recibido satisfactoriamente un formulario enviado)	3,7	1,57
45	¿Puede ponerse en contacto para hacer sugerencias o comentarios?	3,8	1,62

Tabla 12: Puntuación Cuestionario Inicial Feedback (n = 10)

Esta característica reflejó de mejor forma el estado inicial del sistema, dados los diferentes inconvenientes que este presentó.

Con la ayuda de las tablas anteriores se obtuvo como conclusión general del cuestionario aplicado a este grupo de expertos en usabilidad una calificación general del sistema fue de 4.15 puntos sobre 5 (con una desviación estándar de 0.34 puntos), la cual pareció estar sesgada con respecto al resultado esperado, dados los errores presentes en este.

Luego de esta tarea se estudiaron los resultados obtenidos de la encuesta y los del cuestionario, realizando posteriormente un estudio comparativo entre ellos y se pudo observar que existía un alto contraste entre los resultados del primer estudio contra la valoración promedio resultante del segundo, a pesar de que ambos fueron realizados sobre la misma versión del sistema. Posteriormente se definió una lista de posibles modificaciones a realizar sobre RubricArte dadas las sugerencias, comentarios, dudas y críticas realizadas al sistema por parte del grupo de expertos en usabilidad (Tabla 13: Clasificación de Errores y Sugerencias).

Error	*Clasificación*
Los estudiantes deben autenticarse a través de Oxwall Software.	Crítico
Notificar al usuario de la cantidad máxima de caracteres permitidos cuando se está registrando.	Crítico
Verificar la cantidad de caracteres que se almacenan en la base de datos para la contraseña cuando se registra un usuario.	Crítico
Verificar el correo cuando se crea un estudiante en una actividad.	Crítico
Verificar la unicidad de los correos electrónicos en la base de datos.	Crítico
Verificar el tipo de usuario al almacenar en la base de datos.	Crítico
Asignar varias actividades a un solo estudiante.	Crítico
Crear una opción para que el usuario pueda recuperar su clave.	Crítico
Verificar el proceso de evaluación, co-evaluación y auto-evaluación	Mediano
Verificar el proceso de copiar rúbricas, así como su opción de cancelar.	Mediano
Verificar la página de resultados de la búsqueda de rúbricas.	Mediano
Revisar máxima cantidad de caracteres por campo en la base de datos para la opción de crear una rúbrica.	Mediano
Eliminar la opción de regresar al momento de crear una rúbrica.	Mediano
Generar consistencia en el guardado de las rúbricas cuando se crean y cuando se modifican.	Mediano
Arreglar el problema con los caracteres especiales al momento de mostrar y buscar las rúbricas.	Mediano
Mostrar una opción de contacto con el administrador.	Mediano
Verificar el proceso de muestra de rúbricas privadas en la búsqueda general.	Trivial
Acortar los links de las páginas en lo posible para aportar mayor usabilidad y claridad al usuario.	Trivial
Errores en general de ortografía y redacción.	Trivial
Acomodar el campo estado de rúbrica en la base de datos, que dice 0 = incompleta y 1 = completa, cuando se refiere a 0 = privada y 1 = pública.	Trivial
Verificar el proceso de guardado de las etiquetas al momento de editar una rúbrica creada.	Trivial
No mostrar opciones no disponibles en un momento dado, como crear rúbrica, evaluar, etc.	Trivial
Generar consistencia entre iniciar sesión y entrar al sistema.	Trivial
Indicar concretamente al usuario su error cometido y, de ser posible, mostrarlo en la misma página	Trivial
Llenar la columna 1 de la rúbrica base para que sirva de guía a los usuarios al momento de crear una rúbrica.	Mejora
Arreglar la opción “copiar rúbrica”, ya que no es clara.	Mejora

Error	*Clasificación*
Arreglar la visibilidad de las opciones “buscar” y “ver todas” en la sección de buscar rúbricas.	Mejora
El número del ID tiende a confundir al usuario.	Mejora
Mover los botones de agregar fila/columna con su renglón asociado.	Mejora
Crear una opción que permita volver al menú de rúbricas al momento de eliminar una rúbrica.	Mejora

Tabla 13: Clasificación de Errores y Sugerencias

Al momento de concluir con las actividades descritas anteriormente se decidió que a pesar de los resultados obtenidos, algunos módulos que comprendían al software podían ser reutilizados para la nueva versión con la ayuda del proceso de reingeniería aplicado en la metodología ágil propuesta; en ese momento se ratificó la elección de este proceso como algo fundamental para realizar la propuesta de la metodología ágil para el mantenimiento del software.

CAPÍTULO III: MARCO DE DESARROLLO

Una vez que se llevó a cabo el proceso de levantamiento de la información para la nueva versión de la herramienta RubricArte, se procedió a aplicar las actividades restantes de la metodología ágil propuesta; las cuales son descritas en este capítulo, junto con las iteraciones necesarias que fueron realizadas en cada una de estas actividades.

1. Aplicación de las actividades restantes de la metodología ágil propuesta

Según la metodología ágil propuesta al concluir con la fase de análisis, la cual comprende a las actividades de comprensión del programa y la evaluación por expertos en usabilidad, se procede a la fase de implementación del mantenimiento, la cual se dividió en varias iteraciones.

1.1. Iteración de desarrollo 1:

Teniendo en cuenta los resultados del proceso de levantamiento de la información, estos fueron llevados a una reunión que se tuvo junto con el cliente, en la cual se logró la definición del alcance de la nueva versión a liberar del referido sistema. De esta forma se procedió a clasificar los requerimientos que fueron aprobados por el cliente, según la modificación a la estructura y/o a la arquitectura del sistema (Tabla 14: Clasificación de Errores y Requerimientos).

Descripción	***Clasificación***	***Tipo Mant.***	***Módulo***
Crear una opción para que el usuario pueda recuperar su correo y/o clave.	Influyente	Adaptativo	Nuevo
Crear un módulo de administración del sistema.	Influyente	Adaptativo	Nuevo
Crear listas para el manejo de los estudiantes de un profesor para las actividades.	Influyente	Adaptativo	Nuevo
Importar listas desde un archivo .csv	Influyente	Adaptativo	Nuevo
Exportar listas a un archivo .csv	Influyente	Adaptativo	Nuevo
Crear listas al momento de crear una actividad.	Influyente	Adaptativo	Nuevo

Descripción	***Clasificación***	***Tipo Mant.***	***Módulo***
Generar reportes para el profesor, en base a un estudiante y una actividad, donde se visualice gráficamente la auto-evaluación, la co-evaluación y la evaluación.	Influyente	Adaptativo	Nuevo
Verificar la cantidad de caracteres que se almacenan en la base de datos para la contraseña cuando se registra un usuario.	Influyente	Correctivo	Registro
Modificar el registro de la evaluación del profesor, ya que hay inconsistencia en el almacenamiento.	Influyente	Correctivo	Actividad
Verificar el correo cuando se crea un estudiante en una actividad.	Influyente	Correctivo	Actividad
Verificar la unicidad de los correos electrónicos en la base de datos.	Influyente	Correctivo	Actividad
Verificar el tipo de usuario al almacenar en la base de datos.	Influyente	Correctivo	Registro
Asignar varias actividades a un solo estudiante.	Influyente	Correctivo	Actividad
Notificar al usuario de la cantidad máxima de caracteres permitidos cuando se está registrando.	Influyente	Correctivo	Registro
Modificar el proceso de edición y muestra de rúbrica.	Influyente	Correctivo	Actividad
Crear una sección de opciones donde el usuario pueda corregir sus datos.	Media	Adaptativo	Nuevo
Verificar la búsqueda de rúbricas cuando un profesor está autenticado.	Media	Correctivo	Búsqueda
Verificar el proceso de evaluación, co-evaluación y auto-evaluación.	Media	Correctivo	Actividad
Verificar el proceso de copiar rúbricas, así como su opción de cancelar.	Media	Correctivo	Rúbricas
Generar consistencia en el guardado de las rúbricas cuando se crean y cuando se modifican.	Media	Correctivo	Rúbricas
Mostrar sugerencias al momento de llenar el formulario de búsqueda.	Media	Correctivo	Búsqueda
Arreglar la opción de ver rúbrica luego de proceder a buscarla.	Media	Correctivo	Búsqueda
Mostrar una opción de contacto con el administrador.	Media	Adaptativo	Nuevo

Descripción	*Clasificación*	*Tipo Mant.*	*Módulo*
Arreglar el problema con los caracteres especiales al momento de mostrar y buscar las rúbricas.	Insignificante	Correctivo	Búsqueda
Revisar máxima cantidad de caracteres por campo en la base de datos para la opción de crear una rúbrica.	Insignificante	Correctivo	Rúbricas
Eliminar la opción de regresar al momento de crear una rúbrica.	Insignificante	Correctivo	Rúbricas
Verificar el proceso de muestra de rúbricas privadas en la búsqueda general.	Insignificante	Correctivo	Búsqueda
Acortar los links de las páginas en lo posible, para aportar mayor usabilidad y claridad al usuario.	Insignificante	Correctivo	General
Eliminar los errores en general de ortografía y redacción.	Insignificante	Correctivo	General
Cambiar el campo estado de rúbrica en la base de datos, que dice 0 = incompleta y 1 = completa, cuando se refiere a 0 = privada y 1 = pública.	Insignificante	Correctivo	General
Verificar el proceso de guardado de las etiquetas al momento de editar una rúbrica creada.	Insignificante	Correctivo	Rúbricas
No mostrar opciones no disponibles en un momento dado, como crear rúbrica, evaluar, etc.	Insignificante	Correctivo	General
Generar consistencia entre iniciar sesión y entrar al sistema.	Insignificante	Correctivo	General
Indicar concretamente al usuario su error cometido y, de ser posible, mostrarlo en la misma página	Insignificante	Correctivo	General
Llenar la columna 1 de la rúbrica base para que sirva de guía a los usuarios al momento de crear una rúbrica.	Mejora	Correctivo	Rúbricas
Arreglar la opción “copiar rúbrica”, ya que no es clara.	Mejora	Correctivo	Rúbricas
Arreglar la visibilidad de las opciones “buscar” y “ver todas” en la sección de buscar rúbricas.	Mejora	Correctivo	Búsqueda
Cambiar el formato de visualización del ID ya que tiende a confundir al usuario.	Mejora	Correctivo	General

Descripción	*Clasificación*	*Tipo Mant.*	*Módulo*
Mejoras a nivel de la interfaz en la página de inicio.	Mejora	Adaptativo	Nuevo
Mover los botones de agregar fila/columna con su renglón asociado.	Mejora	Correctivo	Rúbricas
Crear una opción que permita volver al menú de rúbricas al momento de eliminar una rúbrica.	Mejora	Correctivo	Rúbricas

Tabla 14: Clasificación de Errores y Requerimientos

Posteriormente se llevaron a cabo los cambios necesarios para atender aquellos requerimientos que fueron clasificados como “Mejora” e “Insignificante”, es decir; aquellos que no conllevaban una amplia modificación a la estructura o a la arquitectura del sistema.

1.2. Pruebas Unitarias y Liberación

En esta fase se procedió a realizar las pruebas de caja blanca necesarias para verificar el correcto funcionamiento de los cambios realizados sobre el sistema; para luego llevar a cabo una reunión con el cliente en la que se pudo validar, a través de pruebas de caja negra, cada una de las modificaciones hechas.

A continuación se muestran las evidencias de los principales requerimientos atendidos en esta iteración (Figura 9: Nueva página principal de RubricArte, Figura 10: Nueva Página Creación de Rúbricas, Figura 11: Nueva URL RubricArte y Figura 12: Nueva Codificación RubricArte):

Figura 9: Nueva página principal de RubricArte

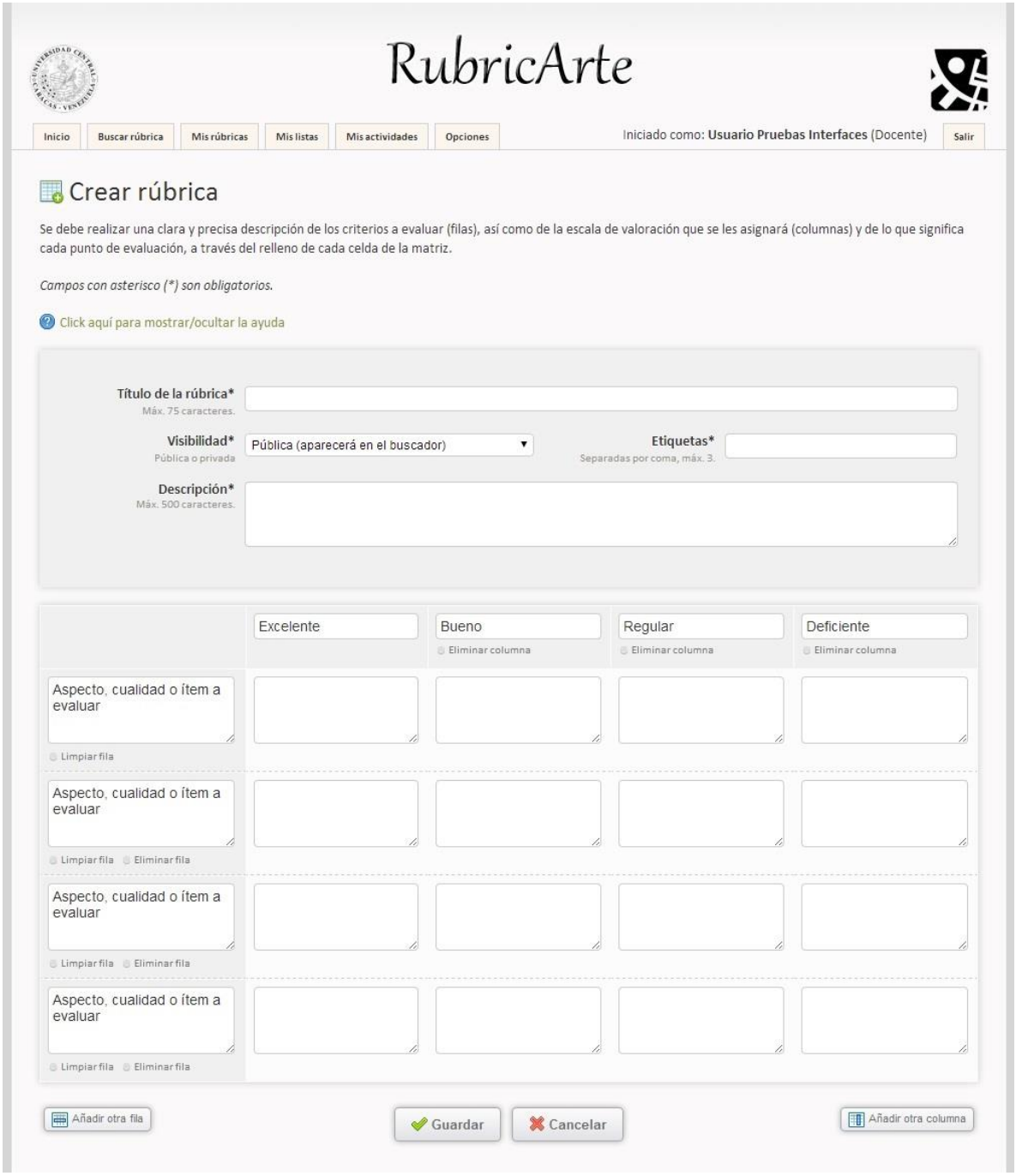


Figura 10: Nueva Página Creación de Rúbricas

Figura 11: Nueva URL RubricArte

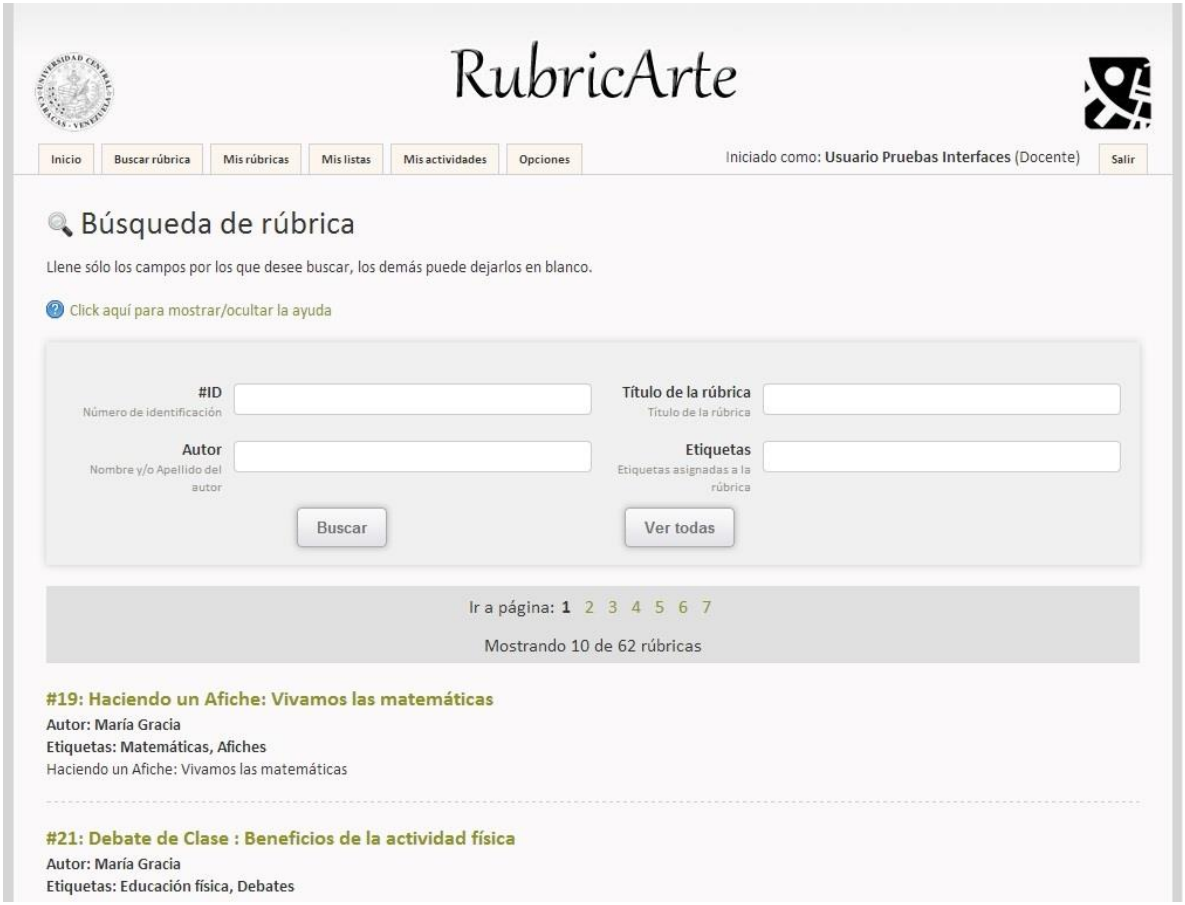


Figura 12: Nueva Codificación RubricArte

Una vez que fueron verificados los cambios realizados, se procedió a liberar una versión nueva, tal como lo indica la metodología propuesta al finalizar cada iteración.

1.3. Iteración de desarrollo 2:

En esta segunda iteración se llevaron a cabo los cambios necesarios para atender aquellos errores y requerimientos que fueron clasificados como “Media”, es decir; aquellos que conllevaban cierta modificación a la estructura y a la arquitectura del sistema; con esta liberación se pretende obtener un avance en la funcionalidad y en la mejora en calidad respecto a la anterior versión.

Los cambios que se realizaron en esta iteración fueron con la finalidad de obtener mejoras en cuanto a la retroalimentación y usabilidad del sistema.

1.4. Pruebas Unitarias y Liberación

En esta fase se realizaron las pruebas necesarias para certificar los cambios realizados sobre el sistema, para luego proceder a realizar una reunión con el cliente en la que se pudieron validar cada una de las modificaciones hechas. Seguidamente se liberó la nueva versión de RubricArte, en la cual se incluyeron los requerimientos desarrollados en la fase anterior.

A continuación se muestran las evidencias de los principales requerimientos atendidos en esta iteración (Figura 13: Contacto Administrador, Figura 14: Opciones de Usuario, Figura 15: Crear Actividad, Figura 16: Opción Ver Rúbrica Funcionando y Figura 17: Sugerencia de Búsqueda):

Figura 13: Contacto Administrador

Figura 14: Opciones de Usuario

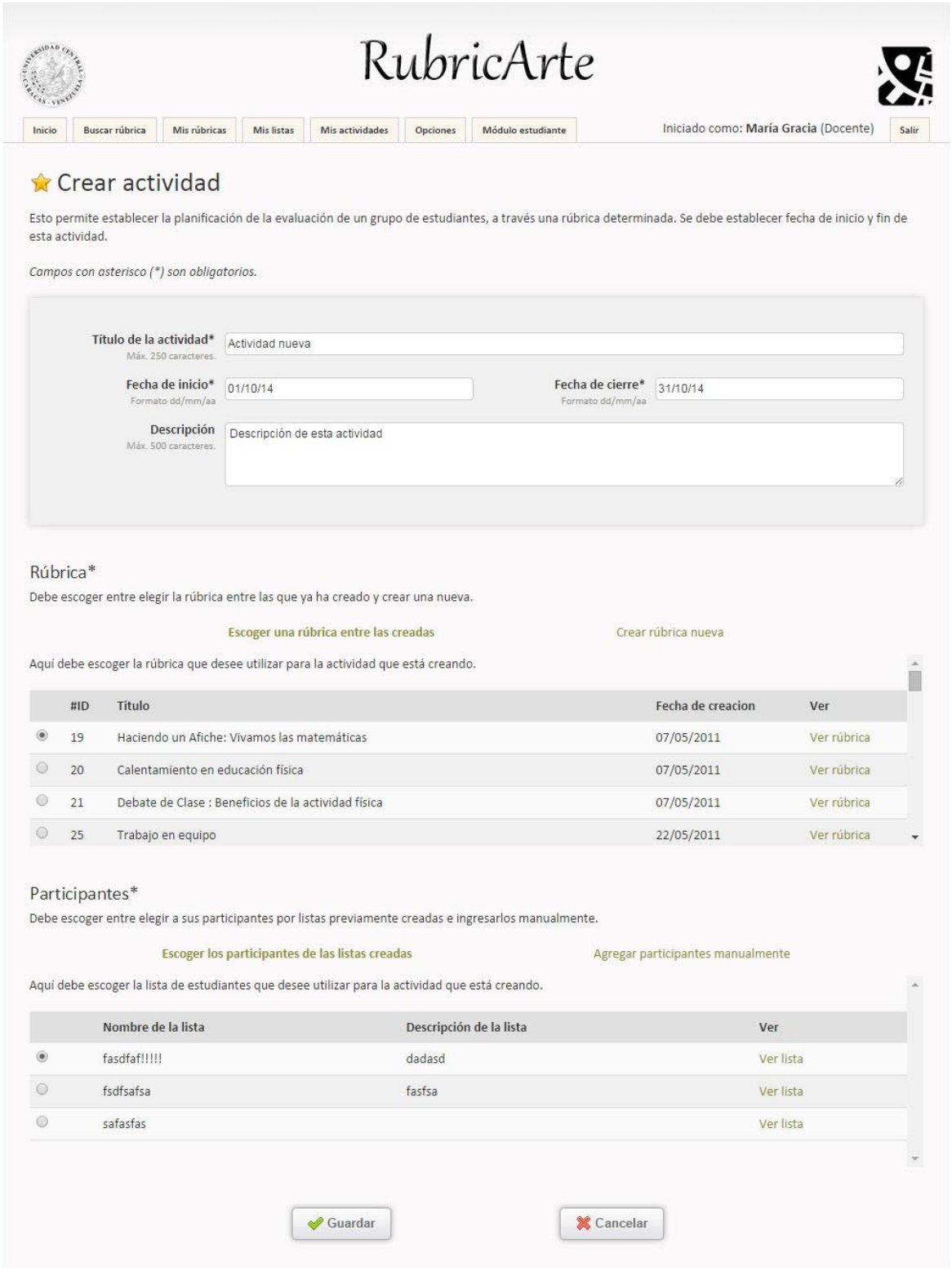


Figura 15: Crear Actividad

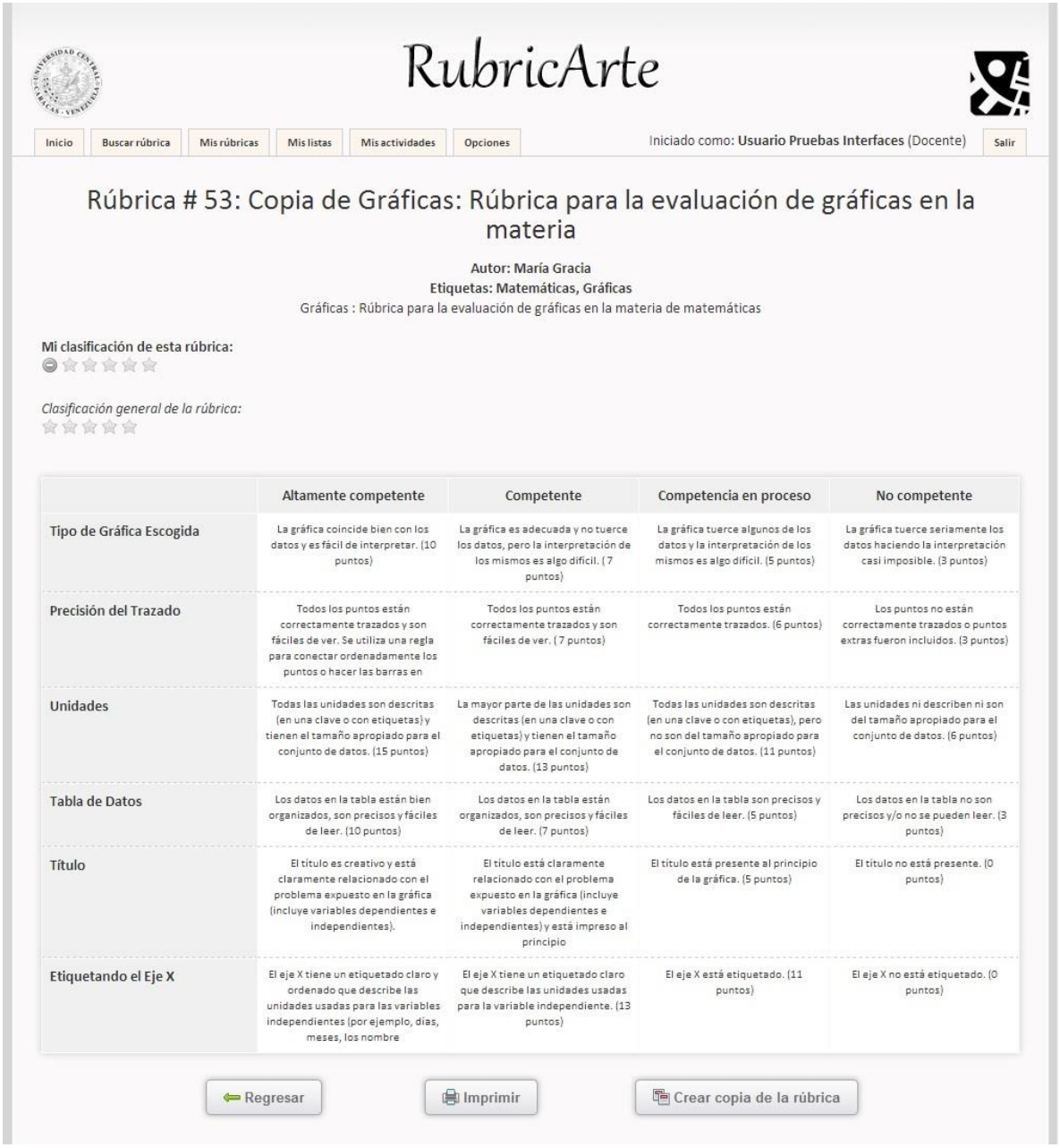


Figura 16: Opción Ver Rúbrica Funcionando

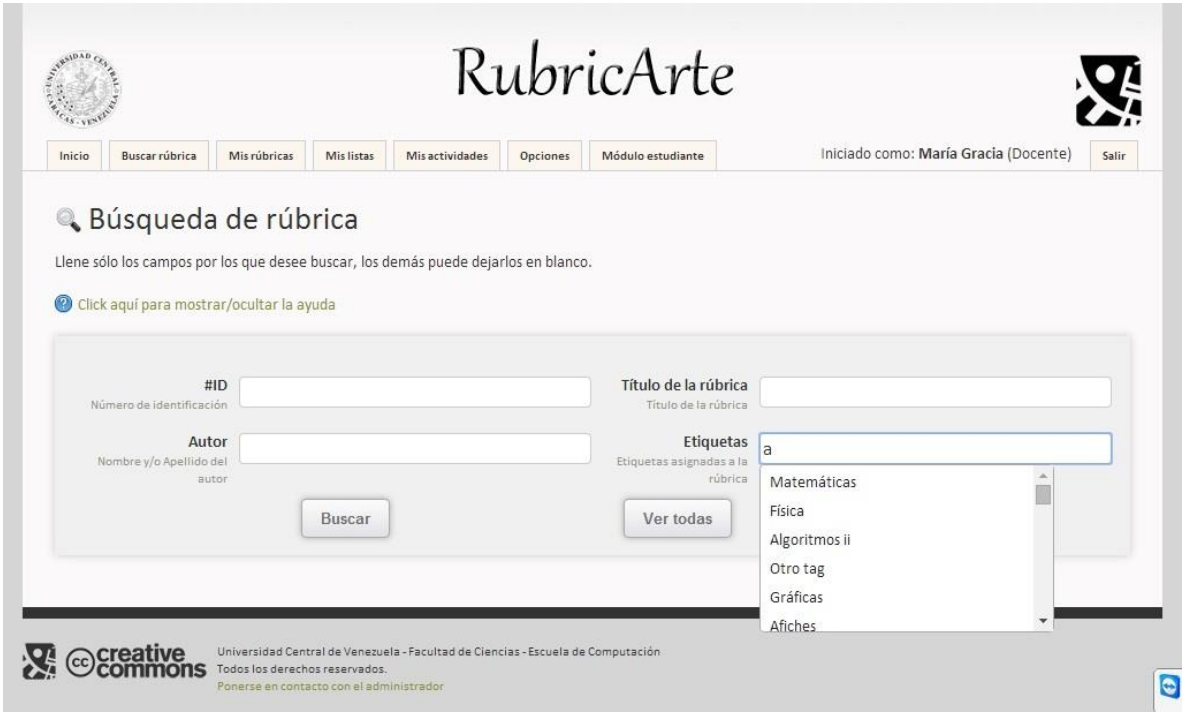


Figura 17: Sugerencia de Búsqueda

1.5. Iteración de desarrollo 3:

En esta tercera iteración fue atendida la primera mitad de aquellos errores y requerimientos que fueron clasificados como “Influyente”, es decir; aquellos que conllevaban una amplia modificación a la estructura y/o a la arquitectura del sistema.

Los desarrollos realizados en esta fase se hicieron con el fin de ofrecer más ayudas al usuario al momento de la interacción, así como nuevas funcionalidades.

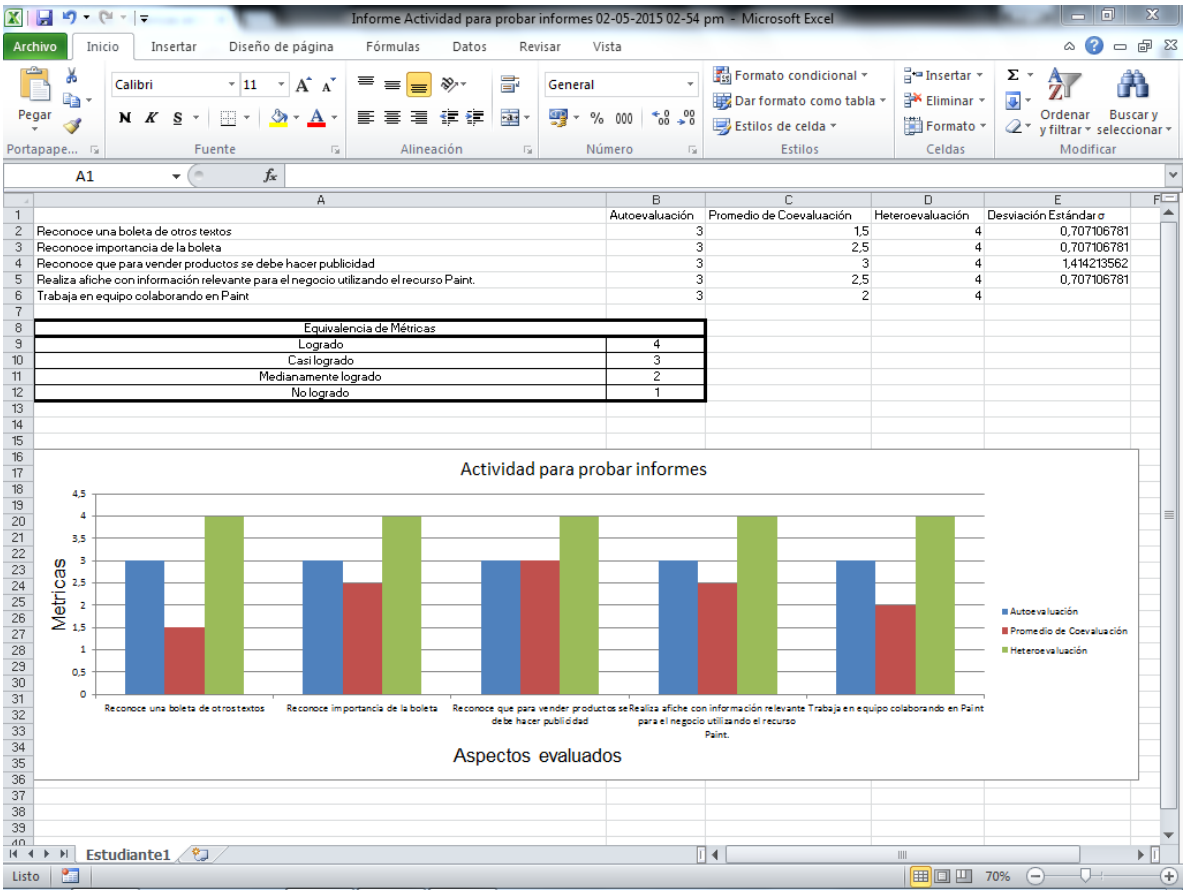
1.6. Pruebas Unitarias y Liberación

Una vez que se realizaron las pruebas de cada uno de los desarrollos, se hizo una reunión con el cliente para exponerlos y se procedió a realizar la liberación de la nueva versión.

A continuación se muestran las evidencias de los principales requerimientos atendidos en esta iteración (Figura 18: Varias Actividades, Figura 19: Ver Actividad, Figura 20: Reporte Generado y Figura 21: Ver Reporte):

Figura 18: Varias Actividades

Figura 19: Ver Actividad



	A	B	C	D	E
1		Autoevaluación	Promedio de Coevaluación	Heteroevaluación	Desviación Estándar σ
2	Reconoce una boleta de otros textos	3	1,5	4	0,707106781
3	Reconoce importancia de la boleta	3	2,5	4	0,707106781
4	Reconoce que para vender productos se debe hacer publicidad	3	3	4	1,414213562
5	Realiza afiche con información relevante para el negocio utilizando el recurso Paint.	3	2,5	4	0,707106781
6	Trabaja en equipo colaborando en Paint	3	2	4	

Equivalencia de Métricas	
Logrado	4
Casi logrado	3
Medianamente logrado	2
No logrado	1

Figura 20: Reporte Generado

Figura 21: Ver Reporte

1.7. Iteración de desarrollo 4:

En esta cuarta iteración se llevaron a cabo los cambios necesarios para atender la segunda mitad de aquellos errores y requerimientos que fueron clasificados como “Influyente”.

1.8. Pruebas Unitarias y Liberación

Una vez que se probaron cada uno de los requerimientos desarrollados en esta iteración, y estos fueron aceptados por el cliente en la respectiva reunión, se procedió llevar a cabo la liberación de la nueva versión.

A continuación se muestran las evidencias de los principales requerimientos atendidos en esta iteración (Figura 22: Recuperar Correo, Figura 23: Recuperar Datos, Figura 24: Importar Lista, Figura 25: Exportar Lista, Figura 26: Crear Listas, Figura 27: Crear Lista con Autocompletar en Crear Actividad y Figura 28: Módulo Administrador):

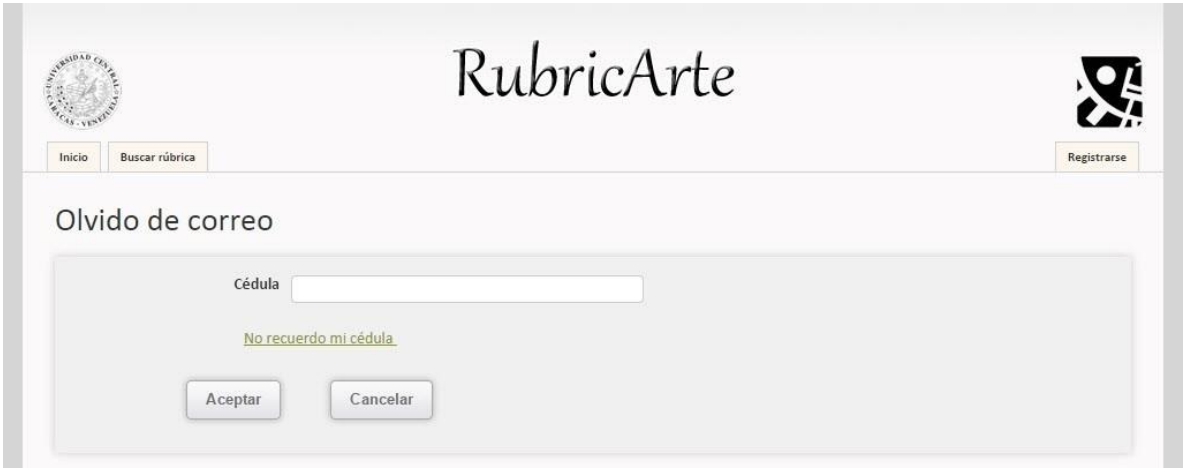


Figura 22: Recuperar Correo

Figura 23: Recuperar Datos

Figura 24: Importar Lista

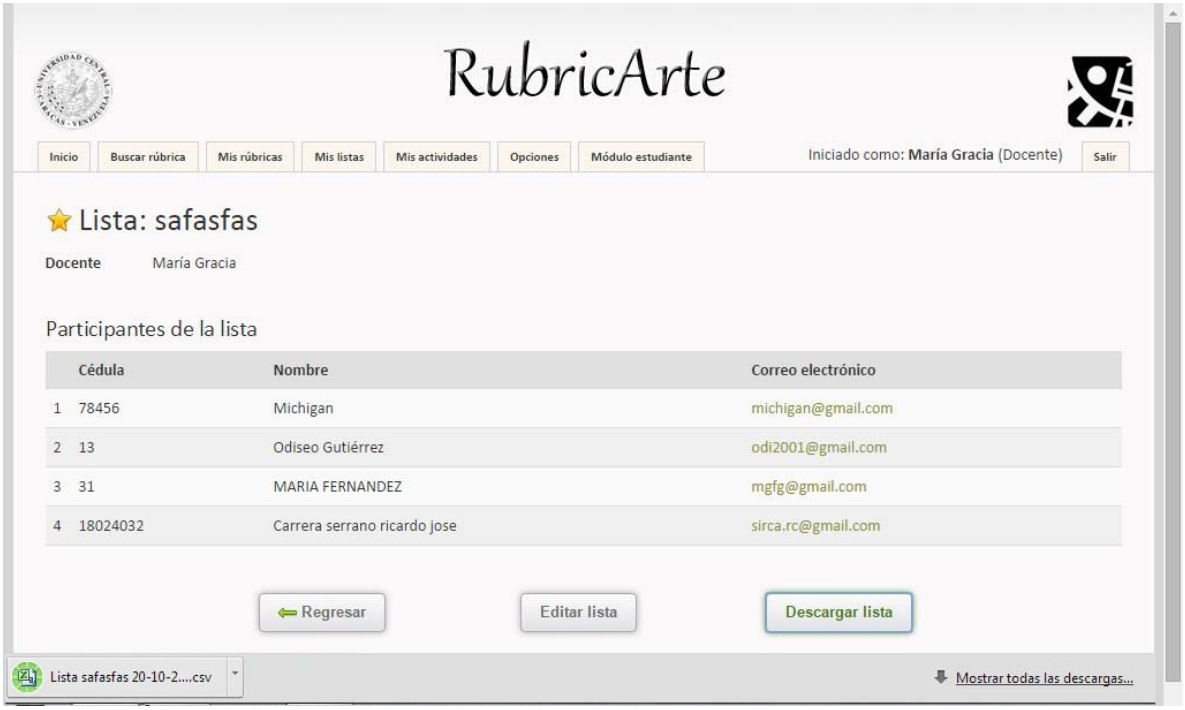


Figura 25: Exportar Lista

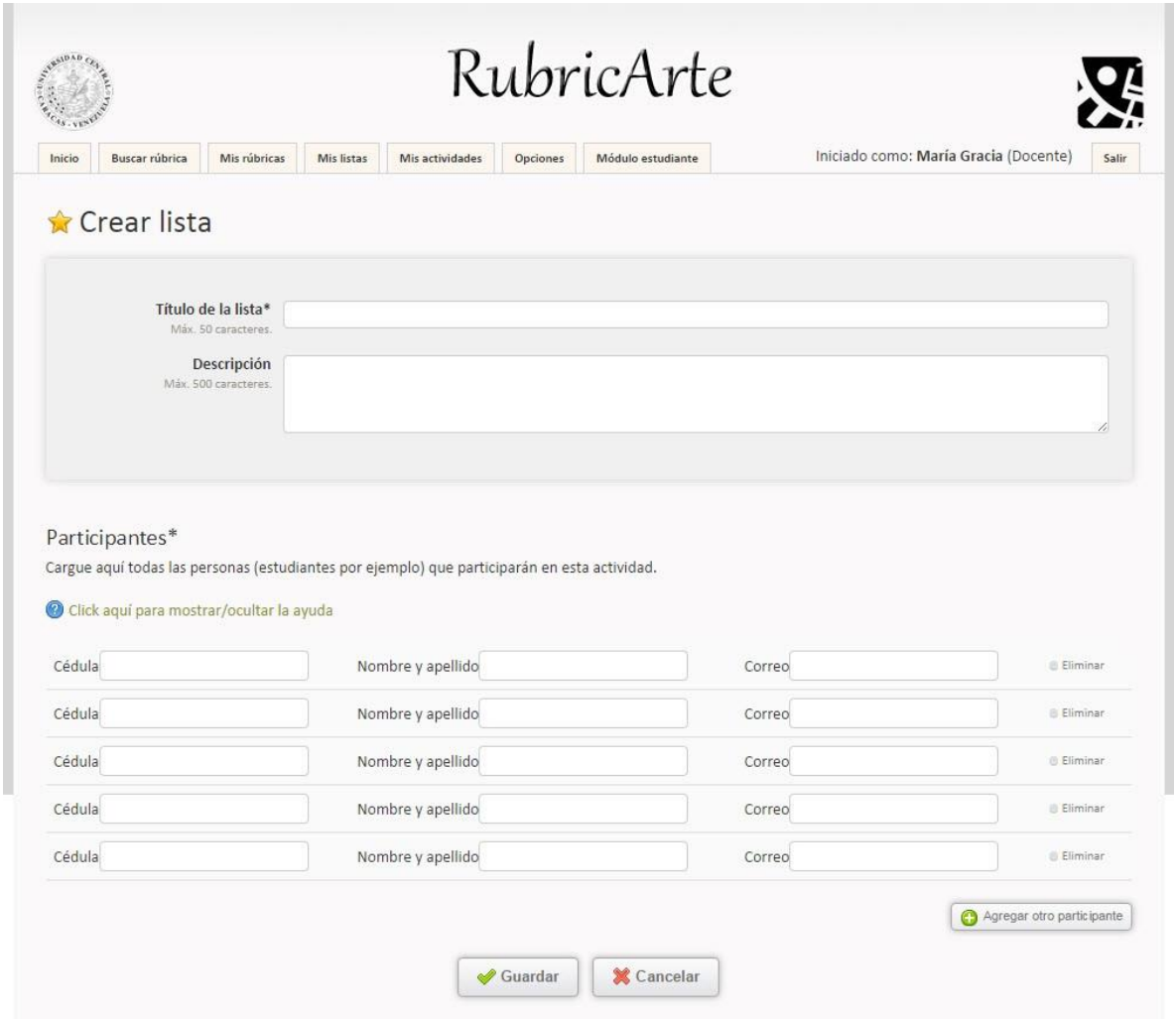


Figura 26: Crear Listas

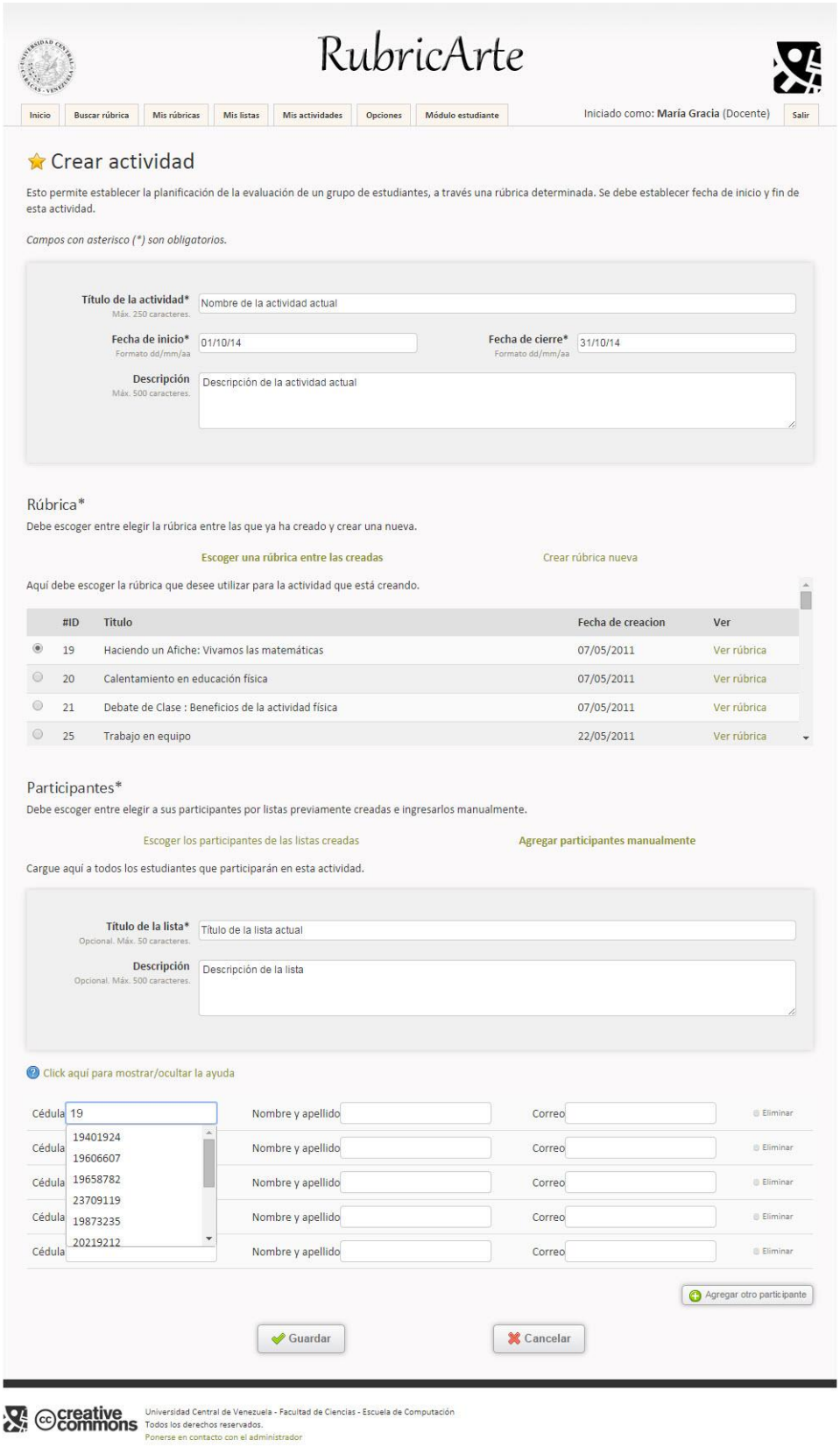


Figura 27: Crear Lista con Autocompletar en Crear Actividad

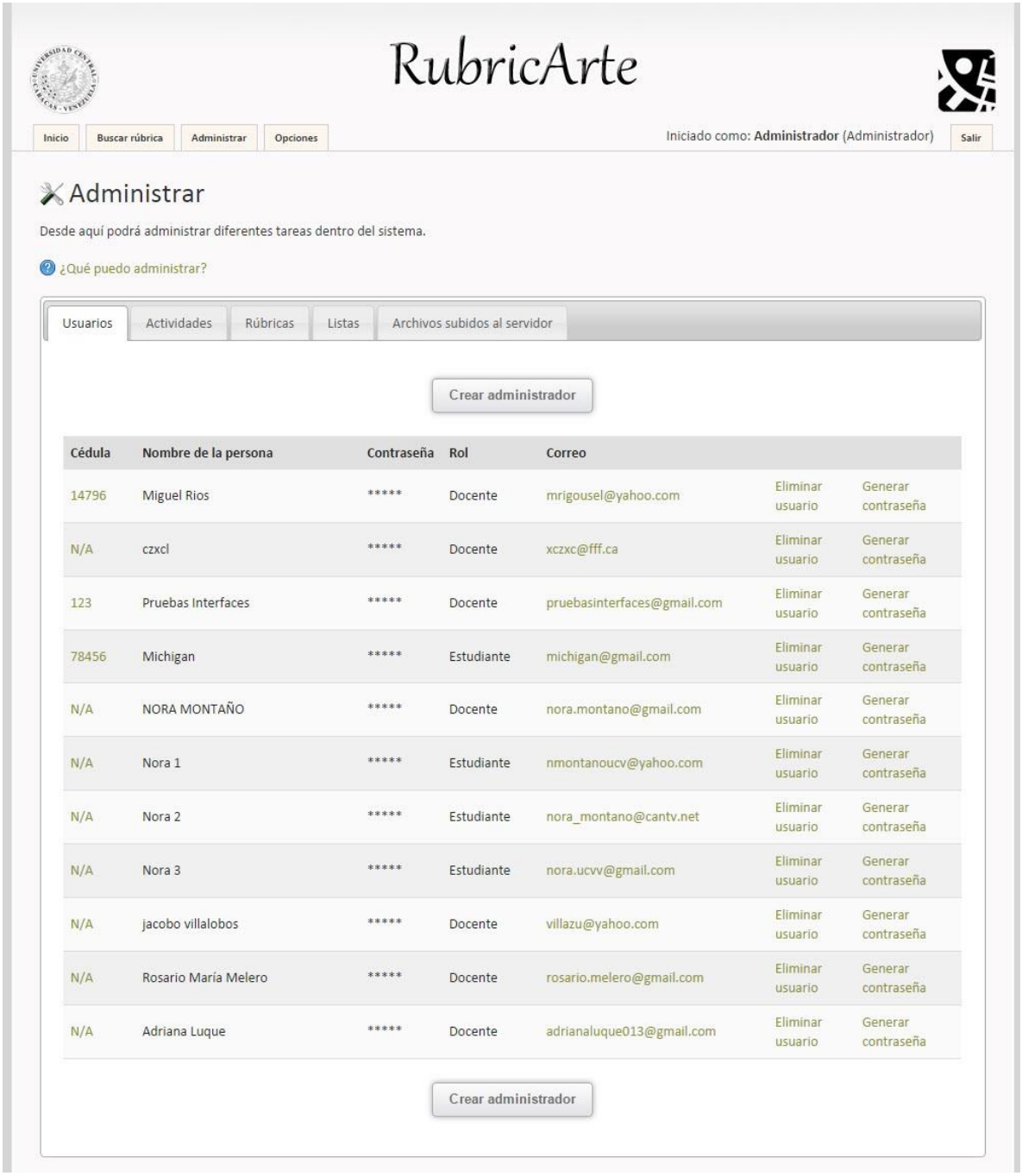


Figura 28: Módulo Administrador

1.9. Evaluaciones por expertos en usabilidad:

Una vez que se desarrollaron cada uno de los requerimientos hechos por el usuario y se aplicaron los tipos de mantenimiento correctivo y adaptativo, se procedió a realizarle el mismo cuestionario que se trabajó inicialmente a un grupo de expertos en usabilidad, pero esta vez se encontraba conformado por nueve (9)

empleados de Indra Sistemas, quienes también tenían conocimientos generales en el desarrollo de aplicaciones de calidad bajo los principios de la Ingeniería del Software y estaban familiarizados con las técnicas de evaluación de usabilidad basadas en heurísticas; con el fin de evaluar la satisfacción obtenida al momento de interactuar con la versión final de RubricArte (Anexo IV: Cuestionario de Satisfacción de RubricArte). Las preguntas realizadas, junto con los resultados obtenidos del cuestionario se muestran agrupadas por características, a continuación:

- Aspectos de Usabilidad (Tabla 15: Puntuación Cuestionario Final Aspectos de Usabilidad):

Número de Pregunta	***Pregunta***	***Puntuación Obtenida (1 - 5)***	***Desv. Estándar***
1	¿El sistema indica qué está pasando en un tiempo razonable?	4,8	0,42
2	¿El sistema está escrito en un lenguaje de forma clara y concisa?	4,8	0,42
3	¿El sistema deja del lado del usuario la navegación y el control?	4,8	0,42
4	¿El sistema es consistente y sigue los estándares?	4,6	0,52
5	¿El sistema previene los errores?	4,7	0,67
6	¿Las herramientas que provee el sistema son fáciles de reconocer o deben ser recordadas? (Calificación mínima: Recordar / Calificación máxima: Reconocer)	4,7	0,48
7	¿El sistema es flexible y da un uso eficiente de la navegación?	4,6	0,52
8	¿El sistema tiene un diseño práctico y simple?	4,8	0,42
9	¿El sistema le ofrece ayudas ante situaciones complejas?	4,5	0,53
10	¿El sistema posee compatibilidad con respecto a sistemas operativos, navegadores web, etc.?	4,4	0,97

Tabla 15: Puntuación Cuestionario Final Aspectos de Usabilidad (n = 9)

En esta característica se pudo observar una paridad con los diversos objetos evaluados, con excepción de la compatibilidad visual en diversos ambientes.

- Aspectos Generales (Tabla 16: Puntuación Cuestionario Final Aspectos Generales):

Número de Pregunta	***Pregunta***	***Puntuación Obtenida (1 - 5)***	***Desv. Estándar***
11	¿Es eficiente?	4,8	0,42
12	¿Es intuitivo?	4,6	0,70

13	¿Mantiene la consistencia (en funcionamiento y apariencia)?	4,7	0,67
14	¿Facilita que el usuario se sienta cómodo y con el control del sitio?	4,7	0,48
15	¿Tiene una URL correcta, clara y fácil de recordar, así como las URL internas?	4,7	0,67
16	¿Las URL son claras y permanentes?	4,6	0,70

Tabla 16: Puntuación Cuestionario Final Aspectos Generales (n = 9)

En este aspecto también se mantuvo en promedio una alta calificación del sistema, mostrándose cónsono con la mejora esperada, especialmente con las URL.

- Identidad e Información (Tabla 17: Puntuación Cuestionario Final Identidad e Información):

Número de Pregunta	***Pregunta***	***Puntuación Obtenida (1 - 5)***	***Desv. Estándar***
17	¿La página de inicio muestra la naturaleza del negocio y se identifica bien el logotipo o marca?	4,6	0,52
18	¿Aparecen elementos de la imagen corporativa en todas las páginas?	4,7	0,48
19	¿Aparece el logo corporativo en un lugar importante dentro de la página?	4,7	0,48
20	¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?	4,5	0,71
21	En artículos, noticias, informes, etc. ¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?	4,6	0,70

Tabla 17: Puntuación Cuestionario Final Identidad e Información (n = 9)

Para esta característica se obtuvo una buena calificación, la cual estuvo bastante pareja.

- Navegación (Tabla 18: Puntuación Cuestionario Final Navegación):

Número de Pregunta	***Pregunta***	***Puntuación Obtenida (1 - 5)***	***Desv. Estándar***
22	¿Aparece la navegación en un lugar prominente, donde se vea fácilmente?	4,3	0,82
23	¿Existen enlaces rotos o que no conducen a ningún sitio?	4,5	0,85

24	¿Tiene un mapa del sitio o un buscador para quienes quieren acceder directamente a los contenidos sin tener que navegar?	4,5	0,85
25	¿Se mantiene una navegación consistente y coherente a lo largo del site?	4,4	0,84
26	¿Existen elementos que permitan al usuario saber exactamente dónde se encuentra dentro del site y cómo volver atrás (breadcrumbs)?	4,3	0,82
27	¿Indican los enlaces claramente hacia dónde apuntan? ¿Está claro lo que el usuario encontrará detrás de cada uno?	4,3	0,82
28	¿Los enlaces son fácilmente reconocibles como tales? ¿Su caracterización indica su estado (visitados, activos, etc.)?	4,5	0,85
29	¿Es predecible la respuesta del sistema antes de visitar un enlace?	4,7	0,67

Tabla 18: Puntuación Cuestionario Final Navegación (n = 9)

En este punto se tuvo una buena calificación en cuanto a lo predecible que es el sistema, mientras que la orientación dada al usuario es relativamente baja.

- Búsqueda (Tabla 19: Puntuación Cuestionario Final Búsqueda):

Número de Pregunta	***Pregunta***	***Puntuación Obtenida (1 - 5)***	***Desv. Estándar***
30	¿Se encuentra fácilmente accesible?	4,8	0,42
31	¿Es fácilmente reconocible como tal?	4,7	0,48
32	¿Permite la búsqueda avanzada?	4,5	0,71
33	¿Muestra los resultados de la búsqueda de forma comprensible?	4,9	0,32
34	¿La caja de texto es lo suficientemente ancha?	4,5	0,71
35	¿Asiste al usuario en caso de no poder ofrecer resultados?	4,4	0,84

Tabla 19: Puntuación Cuestionario Final Búsqueda (n = 9)

En este punto se obtuvo una alta calificación con la comprensibilidad de la información solicitada, mientras que la asistencia al usuario fue mediana.

- Contenidos (Tabla 20: Puntuación Cuestionario Final Contenidos):

Número de Pregunta	***Pregunta***	***Puntuación Obtenida (1 - 5)***	***Desv. Estándar***
36	¿Es coherente el contenido con el contexto de la página o site?	4,9	0,32
37	¿La redacción es corta y precisa?	4,7	0,67
38	¿Existen referencias cruzadas entre textos que están relacionados?	4,8	0,42

Tabla 20: Puntuación Cuestionario Final Contenidos (n = 9)

En este aspecto, se obtuvo una alta calificación para la calidad de la data esperada por los usuarios.

- Tecnología (Tabla 21: Puntuación Cuestionario Final Tecnología):

Número de Pregunta	*Pregunta*	*Puntuación Obtenida (1 - 5)*	*Desv. Estándar*
39	¿La tecnología utilizada en el site es compatible con el software y hardware de los usuarios objetivos? (No tendrán que descargar elementos como plugins para poder usarlo)	4,7	0,48
40	Si es importante utilizar recursos técnicos que requieran la descarga de plugins, ¿se le informa al usuario de esta situación y se le explica la importancia de hacerlo?	4,8	0,42

Tabla 21: Puntuación Cuestionario Final Tecnología (n = 9)

En este punto se observó una alta compatibilidad a nivel de software y hardware, así como de los insumos necesarios para utilizar el sistema.

- Interfaz (Tabla 22: Puntuación Cuestionario Final Interfaz):

Número de Pregunta	*Pregunta*	*Puntuación Obtenida (1 - 5)*	*Desv. Estándar*
41	¿Tiene una interfaz amigable, con colores que concuerden con los objetivos y propósitos?	4,6	0,70
42	¿Hay espacios blancos (libres) entre el contenido, para descansar la vista?	4,7	0,48
43	¿Se ve el sitio exactamente igual en diferentes navegadores?	4,2	0,92

Tabla 22: Puntuación Cuestionario Final Interfaz (n = 9)

Para esta característica se reflejó la aceptación dada a las nuevas actividades disponibles en el sistema. También se observa un sesgo en la compatibilidad de navegadores, dada la alta calificación de la compatibilidad de software y hardware.

- Feedback (Tabla 23: Puntuación Cuestionario Final Feedback):

Número de Pregunta	***Pregunta***	***Puntuación Obtenida (1 - 5)***	***Desv. Estándar***
44	¿Existen respuestas frente a interacciones del usuario? (ej., se le informa que se ha recibido satisfactoriamente un formulario enviado)	4,6	0,70
45	¿Puede ponerse en contacto para hacer sugerencias o comentarios?	4,9	0,32

Tabla 23: Puntuación Cuestionario Final Feedback (n = 9)

Este aspecto demostró el agrado de los usuarios al momento de interactuar con el sistema.

Con la ayuda de las tablas anteriores se pudo observar que la calificación general de la versión final del sistema fue de 4.62 puntos sobre 5 (con una desviación estándar de 0.17 puntos) lo cual demostró la conveniencia de haber aplicado el proceso de mantenimiento.

Tras esta actividad se procedió a estudiar los resultados obtenidos del cuestionario, realizando posteriormente un estudio comparativo entre los obtenidos en esta instancia y iniciales, con lo que se pudo observar que se incrementó la valoración promedio realizada sobre el sistema y se obtuvo que esta última tuvo un mayor grado de usabilidad y dejó un mayor grado de satisfacción al usuario al momento de la navegación; según los cuestionarios aplicados este avance fue de 9,4% (teniendo en cuenta la hipótesis que indica que el primer grupo de expertos en usabilidad pareció haber calificado el sistema de manera errónea, dadas las repetidas discrepancias encontradas al momento de analizar los resultados).

Además de este crecimiento que fue arrojado al estudiar los resultados obtenidos, resaltaron las nuevas funcionalidades en el sistema y el perfeccionamiento en las ya existentes. Aunado a esto se observó que pudieron cumplirse todos los requerimientos realizados junto con el cliente durante la fase de levantamiento de información, demostrando que la propuesta metodológica fue exitosa para el caso de estudio y el contexto seleccionados.

1.10. Producción:

Con la culminación de la última actividad de la propuesta metodológica se pudo observar que luego de cuatro (4) iteraciones se logró cumplir con cada uno de los objetivos descritos y el alcance esperado, dado que se pudo desarrollar completamente cada uno de los requerimientos solicitados por el usuario para la versión final del sistema en las actividades iniciales de esta propuesta metodológica (Tabla 14: Clasificación de Errores y Requerimientos), razón por la cual en la última liberación se procedió a realizar el pase a producción del software, procediendo a cargar los archivos fuentes y demás insumos necesarios en el servidor de SAPDRO, el cual fue escogido para sacar el máximo provecho de la versión final del caso de estudio.

CONCLUSIONES

Una vez cumplidos todos y cada uno de los objetivos propuestos en este Trabajo Especial de Grado, el cual incluye la aplicación de la propuesta metodológica para el mantenimiento ágil del software en el caso de estudio RubricArte, se obtuvieron una serie de conclusiones, las cuales se detallan a continuación:

- Se pudo demostrar que existe una manera de llevar a cabo el mantenimiento en proyectos cuya complejidad es relativamente baja, con equipos de desarrollo de pocos integrantes, donde fueron utilizadas metodologías ágiles para el mantenimiento, todo esto teniendo en cuenta que el encargado de realizar esta tarea, no formó parte del equipo que desarrolló el sistema inicial.
- La metodología ágil para el mantenimiento que fue propuesta en el Seminario asociado al presente Trabajo Especial de Grado fue implementada satisfactoriamente, cumpliéndose con cada una de las actividades asociadas a esta.
- Entre las diferentes actividades que conformaron la propuesta metodológica se destaca la Comprensión del Programa, dado que es una actividad crucial para poder empezar con el proceso de mantenimiento del software.
- La propuesta metodológica estuvo basada principalmente tanto en la aplicación de las técnicas de comprensión del programa y reingeniería, como en los tipos de mantenimiento correctivo y perfectivo; además de las técnicas de desarrollo ágil XP y MA.
- Al estudiar los resultados iniciales, obtenidos en la encuesta y el cuestionario, se concluyó que estos pudieron verse afectados por la no fiabilidad del grupo de expertos en usabilidad, dado que existió una gran cantidad de inconformidades con el sistema y que, sin embargo, este obtuvo un alto puntaje de valoración.

- Con respecto al caso de estudio, la herramienta RubricArte, se logró concluir con el proceso de mantenimiento del software y se cumplieron con todos los requerimientos del usuario.

RECOMENDACIONES

Dado que la metodología propuesta para el mantenimiento ágil del software fue implementada exitosamente en el caso de estudio RubricArte, se obtuvieron una serie de recomendaciones, las cuales se detallan a continuación:

- La propuesta metodológica debería implementarse nuevamente en diversos desarrollos de características similares, para poder llegar a una conclusión general en base a su factibilidad, dado que con una sola aplicación no puede realizarse una generalización acerca de esta, a pesar de que los resultados al implementarla hayan sido muy buenos.
- Al momento de desarrollar los sistemas debe pensarse en la fase de mantenimiento del software como una actividad fundamental dentro del ciclo de vida del software, para lograr tanto desarrollos de calidad como de fácil aplicación de la etapa de mantenimiento.
- Con respecto al caso de estudio, la herramienta de evaluación en línea RubricArte, se propone en implementar una interfaz que permita realizar una vinculación entre los usuarios de RubricArte y los de la Red Colaborativa de Postgrado http://kuainasi.ciens.ucv.ve/red_educativa/index dado que esta última es utilizada como “Sitio de integración social y de aprendizaje”, así como se indica en su página principal, por los diferentes estudiantes del Postgrado de Ciencia de la Computación.

REFERENCIAS

Abran, A., & Moore, J. W. (2004). *Guide to the Software Engineering Body of Knowledge.* California: Institute of Electrical and Electronics Engineers.

Ambler, S. W. (2002). *Agile Modeling (AM)*. Recuperado el 20 de Marzo de 2013, de Effective Practices for Modeling and Documentation: http://www.agilemodeling.com/

Berón, M., Uzal, R., Henriques, P., & Pereira, M. J. (2007). Comprensión de programas por inspección visual y animación. *IX Workshop de Investigadores en Ciencias de la Computación*, (págs. 355-359). Chubut - Argentina.

Doza, A. (2013). *Nueva Metodología Ágil para el Mantenimiento del Software Basada en las Técnicas de Desarrollo Ágil XP y MA Aplicado Sobre el Caso de Estudio Rubricarte.* Caracas.

García, M. G. (2012). RUBRICARTE: Una Herramienta para Evaluación de Aprendizajes a través de Rúbricas. *Trabajo Especial de Grado*. Universidad Central de Venezuela, Caracas, Venezuela.

Hussain, Z., Lechner, M., Milchrahm, H., Shahzad, S., Slany, W., Umgeher, M., y otros. (2012). Practical Usability in XP Software Development Processes. *The Fifth International Conference on Advances in Computer-Human Interactions* (pág. 10). Nawabshah, Pakistan: Center for Usability Research & Engineering.

Nielsen, J. (1 de Enero de 1995). *10 Usability Heuristics.* Recuperado el 15 de Enero de 2013, de Nielsen Norman Group - Evidence-Based User Experience Research, Training, and Consulting: http://www.nngroup.com/articles/ten-usability-heuristics/

Piattini, M. G., Calvo-Manzano, J. A., Cervera, J., & Fernández, L. (2006). *Análisis y diseño de Aplicaciones Informáticas de Gestión.* México D.F.: Alfaomega.

Pressman, R. S. (2005). *Ingeniería del Software. Un Enfoque Práctico.* McGraw-Hill.

Sommerville, I. (2005). *Ingenieria del Software.* Madrid: Pearson Educación.

Anexo I: Evaluación Heurística Realizada a RubricArte

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Evaluación Heurística Realizada a RubricArte

INSTRUCCIONES

Muchas gracias por su cooperación en este proceso de evaluación heurística realizada a la herramienta RubricArte, el cual tiene como finalidad poder conocer la opinión que tienen diferentes posibles usuarios sobre esta, para así poder realizar un mejor mantenimiento al software descrito. Pasos a seguir:

a) Realizar un recorrido por la herramienta web RubricArte, localizada en la URL http://eucalipto.ciens.ucv.ve/ra/index.php con el fin de poder ambientarse a ella.
b) Los diferentes lineamientos que deben regir a RubricArte son:

H1. *Visibilidad del estado del sistema*: El sistema siempre debe mantener informado a los usuarios sobre lo que está sucediendo, las diferentes opciones que se le presentan en un momento dado y proporcionarle respuesta en un tiempo razonable.

H2. *Adecuación entre el sistema y el del mundo real*: El sitio web debe utilizar el lenguaje del usuario, con expresiones y palabras que le resulten familiares, la información debe aparecer en un orden lógico y natural y cualquier término que pueda ser desconocido para el usuario debe definirse.

H3. *Libertad y control por parte del usuario*: Los usuarios eligen funciones del sistema por error y necesitarán a menudo una “salida de emergencia claramente marcada” para dejar el estado indeseado sin tener que pasar por un diálogo extendido. Debe disponer también de la capacidad de deshacer o repetir una acción realizada.

H4. *Consistencia y estándares*: Los usuarios no tienen por qué saber que diferentes palabras, situaciones o acciones significan lo mismo. Es conveniente seguir convenciones.

H5. *Prevención de error*: Es importante prevenir la existencia de errores mediante un diseño adecuado. Los mensajes de error deben incluir una confirmación antes de ejecutar las acciones de corrección.

H6. *Reconocimiento antes que recuerdo*: Reducir al mínimo la carga cognitiva por parte del usuario haciendo visibles objetos, acciones, y las opciones. El usuario no debe tener que recordar la información entre distintas secciones o partes del sistema. Las instrucciones para el uso del sistema deben ser visibles o fácilmente recuperables siempre que sea necesario.

H7. *Flexibilidad y eficacia del uso*: Los aceleradores pueden incrementar la interacción para el usuario experto, de tal forma que el sistema sea útil tanto para usuarios noveles como avanzados.

H8. *Diseño estético y minimalista*: Los diálogos no deben contener información que es irrelevante o poco necesaria. Cada unidad adicional de la información en un diálogo compite con las unidades relevantes y disminuye su visibilidad relativa.

H9. *Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores*: Los mensajes de error deben expresarse en un lenguaje común y sencillo, indicando con precisión el problema y sugiriendo las posibles alternativas o soluciones.

H10. *Ayuda y documentación*: Aunque es mejor que el sistema se puede utilizar sin la documentación, puede ser necesario proveer cierto tipo de ayuda. Dicha información debería ser fácil de buscar, estar enfocada en las tareas del usuario, con una lista de los pasos a seguir y no ser muy extensa.

H11. *Compatibilidad*: Debe proveerse compatibilidad en cuanto a los diferentes ambientes sobre los cuales puede ser utilizado el sistema.

H12. *Sensación de seguridad y ambientación*: Para poder proveer al usuario cierta sensación de seguridad y que este conozca el ambiente de desarrollo del sistema, deben aparecer elementos de la imagen corporativa en todas las páginas, en un lugar importante de estas.

Tareas a realizar:

1. Registrarse
2. Consultar rúbricas
3. Construir rúbricas
4. Participar en una evaluación con rúbrica

Para la tarea de construcción utilizarán la Rúbrica Anexo II: Gerencia de proyectos de software y la Rúbrica Anexo III: El trabajo de investigación.
Gracias por su colaboración.

Anexo II: Gerencia de proyectos de software

RubricArte

Rúbrica #110: Gerencia de proyectos de software

Autor: Antonio dos Santos

Etiquetas: Proyectos, Gerencia, Software

	Superior a lo esperado	**Esperado**	**Debe mejorar**	**Deficiente**
Definición del proyecto	Existe una correcta definición del objetivo general. Los objetivos específicos cuadran perfectamente con el objetivo principal. Los indicadores de logro están perfectamente identificados y son consistentes con el objetivo.	Existe una correcta definición del objetivo general. Los objetivos específicos sumados, llevan al objetivo principal. Se identifican los indicadores de logro.	El objetivo general está especificado. Los objetivos específicos no están completos y no garantizan el logro del objetivo general	El objetivo general del proyecto no está definido o no está claro. Los objetivos específicos no cuadran con el objetivo general.
Administración de requerimientos	Los requerimientos funcionales y no funcionales están identificados y correctamente documentados. Se han documentado los atributos de los requerimientos y se	Todos los requerimientos principales han sido identificados y documentados. Los requerimientos no funcionales han sido categóricamente	Existen requerimientos importantes que no han sido identificados, documentados o mal redactados. Los requerimientos	Los requerimientos no están suficientemente inventariados. Los requerimientos no están claros, son ambiguos o contradictorios. Se omiten varios

	Superior a lo esperado	Esperado	Debe mejorar	Deficiente
	han establecido las matrices de trazabilidad.	establecidos.	no funcionales son incipientes.	requerimientos.
Definición del alcance	El alcance del proyecto está definido, documentado y reflejado en el plan general. Existe un plan de aceptación coherente para confirmar su entrega al cliente. Se prevén mecanismos para administrar el alcance	El alcance del proyecto está claramente definido y documentado. Existe un plan de aceptación del producto.	El alcance del proyecto no cuadra con los planes presentados. Existen ambigüedades e indefiniciones.	El alcance del proyecto no se ha definido en modo alguno o es incoherente o contradictorio
Planificación del proyecto	El plan del proyecto está completo y en plena coordinación con los objetivos del proyecto y el alcance definido. Contiene o referencia perfectamente los planes subordinados. Define claramente los entregables y fechas. Contiene suficientes hitos y puntos de control.	El plan del proyecto refleja correctamente el ciclo de vida, contiene lo necesario para lograr el alcance comprometido. Contiene o referencia los planes subordinados	El plan no contempla todas las actividades o planes subordinados. No establece claramente los entregables y plazos.	El plan del proyecto no es coherente con los objetivos, alcance, proceso de desarrollo, estimación. No incluye los planes subordinados o es incoherente con éstos.

Anexo III: El trabajo de investigación

RubricArte

Rúbrica #169: El trabajo de investigación

Autor: María Gracia

Etiquetas: Evaluación, Postgrado

	(1) Principiante: El trabajo de investigación tiene un nivel de principiante	(2) En desarrollo: El trabajo de investigación contiene indicios de calidad pero requiere mejoras	(3) Competente: El trabajo de investigación es aceptable	(4) Ejemplar: El trabajo de investigación es ejemplar
Información contenida	El trabajo carece de datos y/o los datos no son correctos ni pertinentes.	Provee información básica, parte de la cuál es incorrecta y/o no es pertinente basado en el trabajo mínimo de investigación.	Provee información parcialmente completa, correcta y pertinente como resultado de un trabajo adecuado de investigación.	Provee información completa, correcta y pertinente apoyada claramente por investigaciones extensivas y cuidadosas.
Calidad de pensamiento y comunicación	Demuestra poca comprensión del tema. No se expresa bien las ideas ni las apoya con ejemplos, razones, detalles ni explicaciones. Ni interpreta ni analiza las materiales.	Demuestra comprensión limitada del tema sin mucha reflexión ni análisis profundo... No se expresa bien las ideas y faltan ejemplos, razones, detalles y explicaciones. Examina el	Demuestra una comprensión general del tema. Por lo general, expresa las ideas claramente por medio de ejemplos, razones, detalles y explicaciones. Examina el problema desde más de un punto de	Demuestra comprensión profunda y conocimiento del problema bajo investigación por un análisis esmerado y por reflexión. Apoya las ideas completamente y las expresa claramente por el uso de

	(1) Principiante: El trabajo de investigación tiene un nivel de principiante	(2) En desarrollo: El trabajo de investigación contiene indicios de calidad pero requiere mejoras	(3) Competente: El trabajo de investigación es aceptable	(4) Ejemplar: El trabajo de investigación es ejemplar
		problema de una sola perspectiva.	vista.	ejemplos, razones, detalles y explicaciones. Examina el problema desde tres puntos de vista o más.
Organización, ortografía y vocabulario	Las secciones escritas carecen de mecanismos de organización, a saber, de la división de párrafos, secciones, capítulos y transiciones. Repleto de errores gramaticales, de puntuación, deletreo y del uso de letras mayúsculas. Falta una sección bibliográfica.	Tiene lenguaje copiado de otras fuentes de información. Emplea los mecanismos de organización escrita tales como la división de párrafos, secciones, capítulos y transiciones de una forma incompleta y equivocada. Repleto de errores gramaticales, de puntuación, deletreo y del uso de letras mayúsculas. La sección bibliográfica contiene un número insuficiente de fuentes primarias y	El trabajo está escrito en las propias palabras del estudiante. Se ven algunos problemas con la división de párrafos, secciones, capítulos y transiciones. Hay varios errores de gramática, puntuación, deletreo y del uso de letras mayúsculas. La sección bibliográfica identifica un número adecuado de fuentes primarias y secundarias de	El trabajo está escrito en las propias palabras del estudiante con un vocabulario apropiado. Utiliza los mecanismos de organización escrita tales como la división de párrafos, secciones, capítulos y transiciones de una forma efectiva. El trabajo contiene muy pocos errores de gramática, puntuación, deletreo y del uso de letras mayúsculas. La sección bibliográfica identifica una variedad de fuentes primarias y secundarias de

	(1) Principiante: El trabajo de investigación tiene un nivel de principiante	(2) En desarrollo: El trabajo de investigación contiene indicios de calidad pero requiere mejoras	(3) Competente: El trabajo de investigación es aceptable	(4) Ejemplar: El trabajo de investigación es ejemplar
		secundarias de información.	información.	información.
Ayudas visuales	Las ayudas visuales no facilitan la comprensión del contenido y el mensaje central a la audiencia.	Las ayudas visuales no tienen una relación directa del trabajo o prestan poco apoyo al trabajo. Las gráficas, tablas, cuadros, diagramas, imágenes o modelos no son ni claros ni pertinentes.	El trabajo es apoyado por el uso de ayudas visuales. Existen algunos errores gráficas o de diseño (ej. una imagen confunde al lector por falta de una explicación escrita)	Las ayudas visuales hechas con lujo de detalle dan poder al trabajo escrito. Las gráficas, tablas, cuadros, diagramas, imágenes y modelos son útiles y claramente nombrados.

Anexo IV: Cuestionario de Satisfacción de RubricArte

Muchas gracias por su cooperación en este proceso de evaluación heurística realizada a la herramienta RubricArte, el cual tiene como finalidad poder conocer la opinión que tienen diferentes posibles usuarios sobre esta, para así poder realizar un mejor mantenimiento al software descrito. A continuación se le realizan diferentes preguntas sobre RubricArte, tenga en cuenta los lineamientos que deben regirla, explicados en la actividad anterior. Califique en una escala del 1 al 5, en la que 1 y 5 corresponden a la mínima y máxima puntuación respectivamente.

Nombre completo *

E-mail *

1. ¿El sistema indica qué está pasando en un tiempo razonable? *

2. ¿El sistema está escrito en un lenguaje de forma clara y concisa? *

3. ¿El sistema deja del lado del usuario la navegación y el control? *

4. ¿El sistema es consistente y sigue los estándares? *

5. ¿El sistema previene los errores? *

6. ¿Las herramientas que provee el sistema son fáciles de reconocer o deben ser recordadas? (Calificación mínima: Recordar / Calificación máxima: Reconocer) *

7. ¿El sistema es flexible y da un uso eficiente de la navegación? *

8. ¿El sistema tiene un diseño práctico y simple? *

9. ¿El sistema le ofrece ayudas ante situaciones complejas? *

10. ¿El sistema posee compatibilidad con respecto a sistemas operativos, navegadores web, etc.? *

Sugerencia o comentario

Aspectos generales

11. ¿Es eficiente? *

12. ¿Es intuitivo? *

13. ¿Mantiene la consistencia (en funcionamiento y apariencia)? *

14. ¿Facilita que el usuario se sienta cómodo y con el control del sitio? *

15. ¿Tiene una URL correcta, clara y fácil de recordar, así como las URL internas?*

16. ¿Las URL son claras y permanentes? *

Sugerencia o comentario

Identidad e información

17. ¿La página de inicio muestra la naturaleza del negocio y se identifica bien el logotipo o marca? *

18. ¿Aparecen elementos de la imagen corporativa en todas las páginas? *

19. ¿Aparece el logo corporativo en un lugar importante dentro de la página? *

20. ¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web? *

21. En artículos, noticias, informes, etc. ¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento? *

Sugerencia o comentario

Navegación

22. ¿Aparece la navegación en un lugar prominente, donde se vea fácilmente? *

23. ¿Existen enlaces rotos o que no conducen a ningún sitio? *

24. ¿Tiene un mapa del sitio o un buscador para quienes quieren acceder directamente a los contenidos sin tener que navegar? *

25. ¿Se mantiene una navegación consistente y coherente a lo largo del site? *

26. ¿Existen elementos que permitan al usuario saber exactamente dónde se encuentra dentro del site y cómo volver atrás (breadcrumbs)? *

27. ¿Indican los enlaces claramente hacia dónde apuntan? ¿Está claro lo que el usuario encontrará detrás de cada uno? *

28. ¿Los enlaces son fácilmente reconocibles como tales? ¿Su caracterización indica su estado (visitados, activos, etc.)? *

29. ¿Es predecible la respuesta del sistema antes de visitar un enlace? *

Sugerencia o comentario

Búsqueda

30. ¿Se encuentra fácilmente accesible? *

31. ¿Es fácilmente reconocible como tal? *

32. ¿Permite la búsqueda avanzada? *

33. ¿Muestra los resultados de la búsqueda de forma comprensible? *

34. ¿La caja de texto es lo suficientemente ancha? *

35. ¿Asiste al usuario en caso de no poder ofrecer resultados? *

Sugerencia o comentario

Contenidos

36. ¿Es coherente el contenido con el contexto de la página o site? *

37. ¿La redacción es corta y precisa? *

38. ¿Existen referencias cruzadas entre textos que están relacionados? *

Sugerencia o comentario

Tecnología

39. ¿La tecnología utilizada en el site es compatible con el software y hardware de los usuarios objetivos? (No tendrán que descargar elementos como plugins para poder usarlo) *

40. Si es importante utilizar recursos técnicos que requieran la descarga de plugins, ¿se le informa al usuario de esta situación y se le explica la importancia de hacerlo?*

Sugerencia o comentario

Interfaz

41. ¿Tiene una interfaz amigable, con colores que concuerden con los objetivos y propósitos? *

42. ¿Hay espacios blancos (libres) entre el contenido, para descansar la vista? *

43. ¿Se ve el sitio exactamente igual en diferentes navegadores? *

Sugerencia o comentario

Feedback

44. ¿Existen respuestas frente a interacciones del usuario? (ej., se le informa que se ha recibido satisfactoriamente un formulario enviado) *

45. ¿Puede ponerse en contacto para hacer sugerencias o comentarios? *

Sugerencia o comentario