REVECOM

REVISTA VENEZOLANA DE COMPUTACIÓN

ML • Java • Python • C# • C++ • Scala
Perl • Haskell • Lisp • JavaScript

Sociedad Venezolana de Computación

Vol. 6, No. 2
Diciembre 2019

ISSN: 2244-7040

SVC VENEZUELA
Sociedad Venezolana de Computación

Revista
Venezolana
de Computación

REVECOM

Revista Venezolana de Computación

Sociedad Venezolana de Computación

Editores:
Eric Gamess, Wilmer Pereira, Yudith Cardinale

ISSN: 2244-7040

Vol. 6, No. 2
Diciembre 2019

Editorial

Las Tecnologías de Comunicación e Información y la Pandemia ¿Un Nuevo Hito?

Estamos pasando por una pandemia que marcará un hito en la historia, la llamada Coronavirus o COVID-19, que ha levantado muchas alertas y que pudiese desencadenar cambios importantes en el mundo. Es muy pronto para análisis y resultados estables. Esa pandemia se desarrolla en un mundo convulsionado por el cambio climático, por el calentamiento global del planeta y por guerras que han adquirido nuevas dimensiones en el siglo XXI, todo provocado por la acción del hombre, y potenciado con las Tecnologías de Comunicación e Información (TIC), un aspecto para reflexionar. Esta pandemia ha colocado en el tapete problemas que han sido eludidos: de si primero es la economía o primero es la gente, el papel del Estado, lo público y lo privado, entre otras dicotomías, o la no invencibilidad del ser humano, su vulnerabilidad.

Situándonos en esta realidad, y en tiempos de pandemia, se plantean preguntas generadoras acerca de las TIC que motiven la reflexión: ¿cuál es el papel que juega o debe jugar?, ¿qué hemos aprendido?, ¿qué se espera de ellas?, ¿qué podemos hacer como profesionales, científicos o tecnólogos?, ¿cuál es nuestro rol?, ¿cómo impactan estos cambios en los planes curriculares?

Lo primero que observamos: la gente siente temor y ha tenido que quedarse en casa (las llamadas cuarentenas) por el miedo al contagio en un contexto de un virus desconocido que ha llevado al aislamiento, y entre las actividades importantes a resolver en esta situación de confinamiento están la comunicación, el trabajo, el estudio, el entretenimiento, los servicios, y por tanto pasan a ser relevantes las herramientas tecnológicas que faciliten enfrentar esos problemas que se generan en el marco de la pandemia. Eso coloca en primer plano varias actividades que tienen que ver con el mundo informático: *el trabajo remoto* (dependiente o autónomo), *la formación a distancia* y *las compras y pagos en línea* haciendo uso de las TIC, desde la casa. Así mismo hay que evitar el endiosamiento de lo tecnológico y entender que no son la panacea para todo, por ejemplo, la educación a distancia en esta coyuntura proporciona una solución, pero nunca sustituirá *per se* a la presencial, con su riqueza en la socialización con otros y otras y de la interacción directa. Y con respecto al trabajo remoto son demasiadas las advertencias y reflexiones por hacer, pero en situaciones dadas es una opción. En todo caso no se realizará la valoración de las tecnologías que se consideren, pues todas tienen sus aspectos positivos y negativos, sus ventajas y desventajas, sus fortalezas y debilidades, sus peligros y riesgos, tan solo se señala la importancia en este contexto.

La formación a distancia, el trabajo remoto y el comercio electrónico mediante la utilización de medios digitales han adquirido gran desarrollo en los últimos tiempos, el problema que se plantea hoy es la *masificación*, el uso por millones de personas que no tienen los conocimientos informáticos para usarlas a cabalidad. ¿Cómo ayudar a esos millones de personas? Una de las responsabilidades de los especialistas en TIC es producir sistema o aplicaciones de software *usables y que generen gratas experiencias de usuario*, esto es, que le faciliten la vida a la gente cuando interactúa con ellas dejándole al usuario una experiencia satisfactoria, de esto último trata el área conocida como *Experiencia del Usuario* (User eXperience -UX-). La *usabilidad* de los sistemas informáticos es una de las cualidades relevantes del software interactivo. Un software es *usable* si es fácil de aprender, de memorizar, intuitivo, que evite los errores del usuario, que provea *feeback* a las acciones del usuario, que muestre todas las funcionalidades del sistema obteniendo resultados en forma eficiente (en el entendido que cumple sus funcionalidades). La usabilidad puede ser medida, existen técnicas y métricas desarrolladas por equipos multidisciplinarios para ello. La usabilidad se extiende a los conceptos de UX: que los usuarios sientan la aplicación agradable, atractiva, deseable, sostenible, social cuando interactúan con

Editorial

ella (se enamoran de ella). Se señalan áreas que intervienen en el proceso de construcción del software: *Ingeniería del Software*, una disciplina que trata con metodologías de desarrollo y en particular interesa las metodologías que facilitan la obtención de productos usables, en las que el usuario participa desde el inicio del proceso. Y las sub-áreas *Interfaces de Usuario* (relativo a la capa visible, en pantalla, de la aplicación), *Diseño de la Interacción* (en el contacto entre el usuario y la aplicación) y *Experiencia del Usuario* (se focaliza en el usuario). El concepto de *usabilidad*, por cierto, aplica a todo tipo de productos con el que interactúan usuarios. Desde el punto de vista curricular se debería introducir el concepto de usabilidad + UX desde el inicio de la carrera en el sentido de crear conciencia de la importancia de construir software en función de los usuarios, para que éstos puedan resolver sus tareas. Y eso implica introducir metodologías con esa orientación desde el primer curso de programación, lo cual debe ser transversal en la formación, donde competa, es una responsabilidad en el área de la ingeniería de software. A corto plazo es urgente introducir esta formación en cursos especiales.

Por su carácter y alcance mundial, esta problemática global requiere el tratamiento de grandes volúmenes de información, de diversa índole, con diferentes formatos, diferentes fuentes, heterogéneas, que crece exponencialmente y que demandan un alto poder computacional para su procesamiento, es lo que se ha llamado *Big Data*. Tan solo en las redes sociales, la información que allí se genera es gigantesca e impresionante. La combinación de *Big Data* y la *Inteligencia Artificial* es explosiva para inferir, deducir, modelar, extraer información aplicable en muchos campos. A nivel curricular es necesario fortalecer –o crear– estos cursos, este conocimiento conlleva a innovar mejores soluciones en áreas hibridas sorprendentes, entre ellas la de la salud. Estos cursos deben darse a nivel básico universitario o de postgrado. Y en un futuro cercano, crear especialidades cortas, de un año.

Los programas nacionales, masivos, deben garantizar el acceso no solo a las aplicaciones, sino a los dispositivos y a las telecomunicaciones, y deben existir políticas de estado que lo faciliten, de lo contrario sería excluida una parte importante de la población. Además, no tiene por qué ser la última tecnología sino aquella que resuelva el problema, por ejemplo, en algunos países la televisión ha sido la solución para la masificación de la educación formal, durante esta pandemia. Es interesante investigar las soluciones en época de pandemia en cada uno de los sectores de la vida de un país, la cultura, la salud, la educación, el deporte, la economía, la banca y hasta el esparcimiento y aprender de estas experiencias, lo cual se facilita si existe coordinación e integración, basadas en la solidaridad, a nivel local, regional o mundial, no solo para la definición de políticas, planificación, sino de control y seguimiento de la información a efecto de tomar las mejores decisiones.

En general, muchos de los progresos e innovaciones en las TIC: en *robótica*, *internet de las cosas, Inteligencia artificial, Big Data, Internet de las nubes, Ingeniería del software*, arquitecturas basadas en *blockchain* y sus múltiples aplicaciones, entre otras, pueden utilizarse en provecho de la humanidad o en detrimento de ella, de allí que valores como la *ética*, la *moral*, el compromiso y la responsabilidad profesional tienen un peso fundamental. Y en el rol de formadores hay que tener conciencia que la formación no solo es en conocimiento sino en valores, desarrollo de grupos de investigación con fortalecimiento del trabajo en equipo, fomentando el espíritu crítico y, en definitiva, aplicar estos avances para encauzar proyectos y soluciones para la gente y para mejorar su condición de vida y para el país, su desarrollo y soberanía. Decir soberanía en las TIC es hablar, entre otras cosas, de *software libre*, de *conocimiento libre*, necesarios para la democratización de la ciencia y la tecnología.

Dra. Nancy Zambrano Rivas
Profesora de la Escuela de Computación, Universidad Central de Venezuela

Revista Venezolana de Computación

ReVeCom (Revista Venezolana de Computación) es la primera revista venezolana arbitrada, periódica, digital, orienta a la publicación de resultados de investigación en el campo de la computación. ReVeCom fue creada por la SVC (Sociedad Venezolana de Computación) y tiene entre sus objetivos hacer conocer los trabajos de alta calidad investigativa que se realizan a nivel nacional, latinoamericano e internacional. La revista permite la divulgación de artículos con aporte original en castellano o inglés.

ReVeCom es una revista abierta para una mayor difusión de los resultados de investigación. Cuenta con una página web (http://www.svc.net.ve/revecom), donde se encuentran los trabajos publicados e información sobre la revista. La revista promueve la pluralidad de intereses, dando cabida a la divulgación de trabajos de todos los campos del conocimiento inherentes a la computación.

Este volumen de ReVeCom (Vol. 6, No. 2) corresponde a artículos de investigación en el campo de la computación que fueron seleccionados a través de un riguroso arbitraje por expertos del área. En esta oportunidad, el comité evaluador fue compuesto por:

Directorio de la Sociedad Venezolana de Computación

Presidente:
Dr. Leonid Tineo (Universidad Simón Bolívar – Venezuela)

Vicepresidente:
Dr. Eric Gamess (Jacksonville State University – USA)

Secretario:
Dr. Wilmer Pereira (Instituto Tecnológico Autónomo de México – México)

Tesorero:
Dr. David Coronado (Universidad Simón Bolívar – Venezuela)

Coordinadora de Educación e Investigación:
MSc. Mildred Luces (Universidad Nacional Experimental de la Gran Caracas – Venezuela)

Coordinadora de Publicaciones:
Dra. Yudith Cardinale (Universidad Simón Bolívar – Venezuela)

Coordinadora de Eventos:
MSc. Soraya Carrasquel (Universidad Simón Bolívar – Venezuela)

Edición

Comité Editorial

Miembros del Comité Editorial

Tabla de Contenido

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 6, No. 2, pp. 1-9, Diciembre 2019
Fecha recepción: 07/11/2019, Fecha aceptación: 28/04/2020

Competencias del Ingeniero en Informática en la Cuarta Revolución Industrial

Mildred Luces[1]
milluces@gmail.com

[1] Universidad Nacional Experimental de la Gran Caracas, Caracas, Venezuela

Resumen: La ingeniería de software evoluciona con la inteligencia artificial, la robótica, el big data, la nanotecnología, la biotecnología, la ciencia de materiales, el almacenamiento de energía y la computación cuántica, entre otros elementos, para alinearse a la Industria 4.0. Los cambios propuestos en las guías curriculares de la Association for Computing Machinery (ACM) y del Institute of Electrical and Electronics Engineers (IEEE), diseñados en la primera guía curricular CE2004 y actualizados en la CE2016, conforme a criterios científicos, sociales y profesionales; permiten orientar los planes de estudios de licenciatura o ingeniería en las instituciones universitarias acorde a las significativas exigencias actuales. Estas orientaciones coinciden con las conclusiones del Foro Iberoamericano de Ingeniería y Sociedad Digital y con distintas investigaciones en señalar que el profesional de la informática y la computación deben prepararse para desarrollar talento en competencias, actitudes y experiencias tanto digitales como de innovación, emprendimiento y creatividad. Destacan la ética, el pensamiento crítico, los idiomas, el trabajo en red, la adaptación a los cambios, la proactividad, la autogestión y la resiliencia debido al impacto que tendrán en el desarrollo económico, tecnológico y social; y de las nuevas formas de vivir, comunicarnos y relacionarnos con el mundo. Este artículo presenta un breve concepto de la Cuarta Revolución Industrial, análisis comparativo de las Guías Curriculares CE2004 y CE2016 a fin de fundamentar los cambios curriculares propuestos según las nuevas competencias, funciones y exigencias disruptivas en las que coinciden distintas investigaciones realizadas para ahondar en el papel de los profesionales del área de la computación, informática y sistemas.

Palabras Clave: Ingeniería de Software; Comparativa de Currículo; Cuarta Revolución Industrial.

Abstract: Software engineering evolves with artificial intelligence, robotics, big data, nanotechnology, biotechnology, materials science, energy storage, and quantum computing, among other elements, to align itself with Industry 4.0. The changes proposed in the curricular guides of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE), designed in the first CE2004 curricular guide and updated in CE2016, in accordance with scientific, social and professional criteria; They allow guiding the undergraduate or engineering study plans in university institutions according to the current significant requirements. These guidelines coincide with the conclusions of the Ibero-American Forum on Engineering and Digital Society and with different investigations in pointing out that the computer and computing professional should prepare to develop talent in competences, attitudes and experiences both digital and of innovation, entrepreneurship and creativity. They emphasize ethics, critical thinking, languages, networking, adaptation to change, proactivity, self-management and resilience due to the impact they will have on economic, technological and social development; and of the new ways of living, communicating and relating to the world. This article presents a brief concept of the Fourth Industrial Revolution, comparative analysis of the Curricular Guides CE2004 and CE2016 in order to base the proposed curricular changes according to the new competences, functions and disruptive requirements in which different investigations carried out to delve into the role of professionals in the area of computing, informatics and systems.

Keywords: Software Engineering; Curriculum Comparison; Fourth Industrial Revolution.

I. Introduccción

La Ingeniería en Informática y sus profesionales son considerados elementos clave en el escenario de la Cuarta Revolución Industrial, denominada la transformación digital de la industria [1]. Los países afinan estrategias para situar sus industrias en un nivel competitivo globalmente. La principal estrategia sigue siendo la formación de los profesionales que se requieren en esta área. En consonancia, según la Conferencia Regional de Educación Superior (CRES 2018), hoy más que nunca se debe "concertar la Declaración y el Plan de Acción sobre la educación superior en América Latina y el Caribe, en la perspectiva del desarrollo humano sostenible y el compromiso con sociedades más justas e igualitarias, ratificando la

responsabilidad de los Estados de garantizar la educación superior como bien público y el derecho humano y social" [2].

La Cuarta Revolución Industrial o Industria 4.0 promete cambiar el mundo tal y como lo conocemos [3], se refiere a la aplicación de nuevas tecnologías en la industria de producción, implica un nuevo modelo de organización y de control en la cadena de valor a través del ciclo de vida de los productos y a lo largo del sistema de fabricación apoyado por sensores, máquinas y sistemas informáticos. Brinda la oportunidad de incorporar a la electrónica y la informática, que vienen conviviendo en los procesos industriales, a tecnologías de la información que convergen con la sensórica y la robótica, unidos a través de la Internet que todos conocemos de información y personas, a la Internet de las Cosas (IoT) y entre todas ellas, se encuentra la Ingeniería en Informática [1][4][5].

Esta investigación estudia los cambios propuestos en la CE2016 y su adopción en planes curriculares recientes de instituciones universitarias Latinoamericanas [6][7]. Igualmente, las orientaciones de acuerdo a la demanda de nuevas habilidades en los profesionales de la Ingeniería en Informática en el contexto de la Cuarta Revolución Industrial [8]. Los resultados demuestran la compactación del núcleo o cuerpo de conocimientos, la importancia de las habilidades transversales o blandas y de la conformación de espacios formativos para la construcción de nuevas y retadoras competencias profesionales y transversales. Declaran la Ingeniería en Informática como la disciplina que sostiene la existencia de las TIC, que a su vez son el soporte de la sociedad actual y el contexto profesional en el que tienen que desenvolver los profesionales de la Industria 4.0 [9].

II. Contexto Emergente del Profesional de la Ingeniería en Informática

Las múltiples investigaciones [3][5][9][10] describen la Cuarta Revolución Industrial como la introducción masiva de robots en entornos productivos y grandes transformaciones hacia un mundo digital como resultado de estarse borrando los límites entre las esferas físicas, digitales y biológicas generando una fusión entre estos tres planos y con ello un cambio de paradigma. Describen estos procesos como la transición hacia nuevos sistemas ciberfísicos que operan en forma de redes más complejas que se construyen sobre la infraestructura de la revolución digital anterior [5]. Sin embargo, son innumerables las investigaciones dedicadas a resaltar los cambios humanísticos y sociales [3][4][8][9] debido a los aportes que prevé precisamente a las ciencias sociales, humanísticas y naturales. En consecuencia, manifiestan la necesidad no sólo de construir competencias tecnológicas y metodológicas, sino las denominadas competencias transversales que incluyen habilidades blandas o líquidas orientadas a cambios paradigmáticos en la sociedad. Refuerzan la necesidad de transformar las habilidades blandas en los profesionales de las ciencias de la computación, sin descuidar sus capacidades de programación y discernimiento de las ciencias duras [11-13].

Las investigaciones refieren que la Cuarta Revolución Industrial profundiza más en el conocimiento [4][10], en el recolectar, almacenar, procesar y recuperar datos estructurados o no estructurados a través de herramientas tecnológicas como el big data y la inteligencia artificial para la generación de sistemas cada vez más autonómicos, con la menor intervención humana incluyendo procesos de toma de decisiones [14].

Si bien los países deben garantizar una industrialización más limpia en procesos que hagan posible la sostenibilidad política, económica, ambiental y social [15], sin lugar a duda es en esta última dimensión dónde los países latinoamericanos en particular pueden comenzar a aprovechar las ventajas de la Cuarta Revolución Industrial. Según Klaus Schwab [3], Fundador del Foro Económico Mundial, la prosperidad debe ser inclusiva, es la gran oportunidad para que Latinoamérica salde sus deudas sociales y económicas. Schwab afirma que la Cuarta Revolución Industrial tiene el potencial de elevar los niveles de ingreso globales y mejorar la calidad de vida de poblaciones enteras, enfatiza la necesidad de que cada individuo se sienta capaz de vivir una vida plena y con aspiraciones; y procurar el beneficio de todos los miembros de la sociedad. Concluye que el ecosistema de tecnologías emergentes ha impactado y seguirá impactando todos los sectores, cambiará los modos de producción, la forma de comunicarnos y de relacionarnos con el mundo, enfatiza que vivimos una época de mayor promesa o potencial peligro.

Coincide con otras investigaciones [14] en que la aplicabilidad social es propicia para la innovación, la gobernabilidad, la digitalización y la producción en procesos sostenibles en distintas áreas y dimensiones de la sociedad y que requieren de una construcción colectiva, interdisciplinaria, consensuada para su aprovechamiento en mayor medida en áreas en las que los costos sean mínimos, muy especialmente en la educación, salud, registro y control de los procesos productivos de la alimentación, transporte y todos aquellos que demanden la atención del Estado por organizaciones públicas y privadas, y por supuesto del ciudadano común [16]. Para ello, trabajar en su empoderamiento, construir la igualdad de oportunidades, la democratización de los servicios, la gobernabilidad y la disminución de la corrupción como factores sociales imperantes. Procesos en los que deben intervenir equipos multidisciplinarios entre los que se encuentran los ingenieros en informática y afines.

III. Orientación de la ACM

Por su parte, la formación de los profesionales de la computación mundialmente viene siendo fundamentada por la Association for Computing Machinery (ACM) y el Institute of Electrical and Electronics Engineers (IEEE) a través de guías curriculares diseñadas conforme a criterios científicos, sociales y profesionales, identificadas como la CE2004 y la CE2016 [17], sumado a los aportes de la Association for Information Systems (AIS). La CE2016 presenta cambios significativos acorde a las exigencias actuales, ofrece un cuerpo de conocimientos identificado por áreas, a partir de las cuales las instituciones universitarias orientan sus diseños curriculares de acuerdo con sus necesidades. Contemplan: el estudio de la organización y arquitectura de las computadoras, cómo se relacionan las computadoras con los algoritmos, la programación, las bases de datos, las redes, la ingeniería de software y las comunicaciones, como núcleo articulador que debe aparecer en todo currículo, identifica las demás áreas como suplementarias [18]. Sin embargo, Vega y Aguilar [19] infieren que el currículo de ingeniería que contenga sólo unidades nucleares se considera incompleto. Una de las diferencias importantes de las guías es la compactación del núcleo

articulador de conocimientos, registrada en la CE2016, con una mayor oportunidad de flexibilizar el currículo y adicionar las denominadas competencias blandas.

De acuerdo con la ACM y la IEEE, la ingeniería en computación es una disciplina que incorpora tanto ciencia como tecnología para diseñar, construir, implementar y mantener componentes de software y hardware. Es vista como una combinación de las ciencias de la computación y la ingeniería eléctrica. En consecuencia, las universidades estadounidenses y europeas incorporan en su currículo asignaturas básicas de ingeniería eléctrica. Las universidades latinoamericanas tienen un mayor componente social y con ello buscan diferenciar la ingeniería en computación de la ingeniería en informática.

La CE2016 reconoce cinco perfiles diferenciados o muy bien definidos:

- Ciencia de la Computación (Computer Science)
- Ingeniería en Computación (Computer Engineering)
- Sistemas de Información (Information Systems)
- Ingeniería de Software (Software Engineering)
- Tecnología de la Información (Information Technology)

Los principios básicos de la ingeniería en computación refieren que:

- Es un campo amplio y en desarrollo permanente.
- Es una disciplina con su propio cuerpo de conocimiento, ethos y prácticas.
- La enseñanza de la ingeniería en computación se basa sólidamente en las teorías y principios de la computación, la matemática y la ingeniería, aplica estos principios teóricos para diseñar hardware, software, redes y equipos e instrumentos computarizados para resolver problemas. Convergen una amplia variedad de disciplinas.
- La evolución permanente de la ingeniería en computación requiere una revisión continua del currículo, en consecuencia, debe ser sensible a los cambios tecnológicos, pedagógicos y de enseñanza aprendizaje.
- Los programas de ingenierías en computación deben contener el diseño y las experiencias de laboratorios.

Además, debe identificar los cambios de habilidades y conocimientos fundamentales, la preparación para la práctica profesional, discusiones de estrategias, tácticas y recomendaciones en la implementación de sistemas informáticos que todos los graduandos deben poseer.

La ACM y el IEEE presentan cinco modelos de curriculum, donde el más cercano al perfil latinoamericano es el modelo B, el que más se diferencia es el modelo D administrado en China, a continuación, ambos modelos.

El Currículo B, mayormente utilizado en las universidades estadounidenses y europeas, contiene los elementos básicos de la ingeniería eléctrica, presente en menor grado en las universidades latinoamericanas, ver en la Tabla 1.

El Curriculum D, administrado en las universidades Chinas, presenta 43 créditos traducidos en más horas académicas que el curriculum de las universidades norteamericanas y europeas. Estos créditos son dedicados a la ciencia computacional e ingeniería, introducción a la inteligencia artificial y 12 créditos académicos dedicados a laboratorios para la ciencia computacional e ingeniería, ver en la Tabla 2.

Tabla 1: Distribución de Créditos Académicos del Currículum B

Credit-Hours	Áreas
20	Mathematics and statics
11	Natural science (physics, Chemistry)
33	Humanities, social science, composition, and literature
25	Required computer science (excluding design Project)
11	Required electrical engineering
9	Technical electives (from computer science or engineering)
5	Culminating design Project (from computer science)
6	Free electives
120	Total Credit Hours for Computer Engineering Program

Tabla 2: Distribución de Créditos Académicos del Currículum D

Credit-Hours	Áreas
25	Mathematics
10	Natural science (physics, Chemistry)
2	Elective mathematics and natural Science
41	Humanities, social science, composition, literature, foreign language, physical education
48	Required computer science and engineering (CE)
12	Lab courses- computer science and engineering (CE)
10	Technical electives – basic and advanced (CE)
15	Capstone Project for graduation (CE)
0	Free electives
163	Total Credit Hours for Computer Engineering Program

El núcleo básico propuesto por la CE2016 está conformado por catorce áreas:

- Algoritmos computacionales
- Arquitectura y organización de computadoras
- Ingeniería de sistemas y proyectos
- Procesamiento de señales
- Sistemas incrustados
- Circuitos y electrónica
- Preparación para la práctica profesional
- Redes computacionales
- Diseño digital
- Gestión de recursos de sistemas
- Diseño de software
- Seguridad de la información

Refiere la CE2016, el plan de estudios debe enfatizar las prácticas profesionales, los aspectos legales y éticos; habilidades para la resolución de problemas y de pensamiento crítico, habilidades personales (blandas); éstas contienen: el trabajo en equipo, la empatía y la comunicación oral y escrita.

A. Sobre el Perfil Profesional, Competencias y Funciones

Según [18], el perfil del profesional en computación debe estar caracterizado por:

- Desarrollar tecnología computacional buscando el bien común de los individuos, la sociedad y las organizaciones.
- Aportar con su formación humana y sus capacidades científicas y profesionales con la solución de los problemas sociales de nuestro entorno.
- Transformar, acelerar y ampliar los límites de cualquier área del conocimiento a través de soluciones innovadoras basadas en el uso eficiente de tecnología computacional.
- Adaptarse rápidamente a los cambios tecnológicos debido a su formación basada en la investigación constante.
- Trabajar y liderar equipos multidisciplinarios que llevan a cabo proyectos de innovación tecnológica.
- Incrementar las ventajas competitivas de cualquier organización a través del uso eficiente de tecnología computacional gracias a su alta capacidad de abstracción.
- Crear empresas de base tecnológica.
- Poder seguir estudios de postgrado con exigencia internacional en áreas relacionadas.

Los empleos del futuro requieren que los jóvenes cuenten con una sólida base teórica, una buena formación práctica y estar en contacto con el mundo del hardware y el software. Según Blanco, Fontodrona y Poveda [5], ocho de cada diez jóvenes, entre 20 y 30 años encontrarán trabajo relacionados al ámbito digital que aún no existen. Los autores describen las diez profesiones más solicitadas: *Ingeniero smart factory, Chief digital officer, Experto en innovación digital, Data scientist, Experto en usabilidad, Director en contenidos digitales, Experto y gestor de riesgos digitales y Director de marketing digital*, aseguran que la demanda de estos profesionales actualmente ya es una realidad. El trabajo más cotizado en el 2016 fue el de *Growth hacker* y el *Especialista en big data*, el más buscado.

Las nuevas ocupaciones refieren a la intersección de profesionales apoyados en programas informáticos y máquinas que se conforman como arquitectos y analistas de macrodatos. Las tecnologías y la interdisciplinariedad modifican la idea de profesiones absolutas y cerradas a lo largo de una vida [20], incluso se aspira a que nuestros jóvenes tengan más de dos profesiones, sin embargo los especialistas en servicios de nubes, desarrolladores de programas informáticos y profesionales del marketing digital [15], se crearán a partir de varias funciones legales nuevas en la intersección de los programas informáticos y el derecho, por ejemplo, ingenieros y arquitectos con conocimiento legal y tecnologías, tecnólogos legales, gerentes de proyectos, gerentes de riesgos y analistas de procesos. Se abren caminos no pensados en todas las disciplinas.

La industria 4.0 se identifica con el término Internet de las Cosas (IoT, Internet of Things) e Internet de los Servicios, para ello el plan de estudios debe estar orientado a que el estudiante desarrolle competencias analíticas avanzadas (big data), simulación avanzada y modelado virtual de plantas, habilidades en la interface hombre–máquina, gestión integrada de control de calidad, de procesos y de productos.

La metodología de enseñanza debe estar enfocada a la innovación, en consecuencia, el diseño curricular debe ser flexible, en permanente actualización, proclive a la acreditación internacional. El estudio de [19] demuestra que la Cuarta Revolución Industrial gira alrededor del conocimiento, en consecuencia afectará prioritariamente a las personas, piedra angular de la gran transformación que se avecina. La universidad es considerada el recinto propicio para estudiar, comprender y generar conocimiento, ciencia y tecnología. Se considera factor nuclear para asumir y hacer denotar que los estudios universitarios no son procesos acabados, que éste debe ser un proceso para toda la vida, que si bien se estudian las bases teóricas, científicas, tecnológicas y su impacto en la sociedad, este es un proceso permanente que debe seguir desarrollándose. El curriculum debe ser atractivo, dinámico, con mayor intercambio presencial y el aprendizaje en línea y en red. En búsqueda de crear los ambientes propicios para el desarrollo, la investigación y la innovación (I+D+i), algo a lo que los países latinoamericanos están obligados, según [10], para poder salir de los ambientes de pobreza. Es oportuno instalar laboratorios informáticos dedicados a transformar el aprendizaje en aplicaciones que contribuyan al cambio de una sociedad consumidora de tecnología a productora de tecnologías que den respuesta a soluciones reales de nuestro contexto.

En ese sentido, se debe orientar la construcción de competencias digitales [15], para atraer talento humano proclive a operar bajo criterios de gestión multifuncional, en aprender a interpretar y gestionar los datos y en desarrollar desde una óptica externa habilidades de cooperación en los nuevos ecosistemas y redes de empresas emergentes. Da por hecho el surgimiento de nuevas organizaciones de bienes y servicios. Insta a desarrollar instrumentos y medidas de ciberseguridad y de protección de la información. Igualmente a definir un plan estratégico avanzado para la evolución de la organización en su transformación al mundo 4.0, esto implica el trabajo multidisciplinario, de conocimientos y habilidades del profesional de la informática con la planificación estratégica necesaria para iniciar la transformación digital en cualquier organización. Plantea los siguientes elementos que debe contener el diseño del plan estratégico:

a) Planificación del alcance y alineación con la estrategia general de la organización; cuyos objetivos se correspondan claramente con priorizar acciones de:
   - Mejora de la eficiencia operativa
   - Optimización de la cadena de valor
   - Posibilidad de crear nuevos modelos de negocio.

   Igualmente, aconseja evaluar la madurez digital actual y contrastarla con la deseada.
b) Facilitar el entorno de trabajo con mentalidad abierta al aprendizaje, al cambio y a la experimentación.
c) Definir las competencias a desarrollar tanto internas como de los proveedores.
d) Reclutar y gestionar talento, priorizando la multidisciplinariedad de los equipos y la capacidad de convertir en virtud el análisis de datos.

e) Seleccionar un conjunto de proveedores con tecnología testadas para ir construyendo el network óptimo de partners.
f) Adoptar una perspectiva de ecosistemas y desarrollar habilidades de gestión en red.
g) Iniciar con proyectos piloto, validar resultados y sistematizar los mecanismos de aprendizaje.

Por su parte, en [11-13][20][21], recopilan varios estudios relacionados con las habilidades blandas para trabajos líquidos, como nuevo paradigma laboral, donde destaca el estudio realizado por el World Economic Forum [2], en la que refieren que en menos de dos años un conjunto de habilidades necesarias para la mayoría de las ocupaciones estarán compuestas por aptitudes que hoy no son consideradas tan importantes. Describen el origen de la denominación flexible, liquido, como la capacidad de adaptarse a las situaciones que sean necesarias, asemejan como un líquido en un vaso, que en el más ligero movimiento cambia la forma del agua. La Figura 1 muestra el gráfico publicado por SpiceWorks referido a las habilidades más importantes para los profesionales de las tecnologías de la información, punteando las habilidades blandas con el conocimiento en seguridad.

Figura 1: Habilidades más Importantes para los Profesionales de las Tecnologías de la Información

La revisión permanente de las competencias no sólo de carácter técnico y metodológico, sino de las denominadas participativas y personales, descritas por el autor como competencias transversales, adiciona, generalizables y transferibles en distintos contextos y actividades [8]. Destaca la necesidad de una educación que fomente el aprender rápido y el desaprender con la misma rapidez para aprender nuevamente en la sociedad VUCA (Volatility, Uncertainty, Complexity y Ambiguity) o sociedad liquida, de grandes transformaciones. Ver el mundo desde nuevas perspectivas, con mente, corazón y voluntad abierta para conectarse con las posibilidades emergentes y hacerlas realidad. Aboga por desarrollar un nivel de consciencia de nuestras capacidades, salir de patrones cognitivos y aprender no sólo a aprender, sino a ser. A ser responsable y consciente, es decir una educación que no esté centrada en lo objetivo, sino en lo subjetivo.

Ante las investigaciones de la cultura digital, proclive a sacrificar la conversación por la conexión, al nacimiento de una época más acelerada, llena de distractores y en constante movimiento, hace un llamado a ir lento, prestar atención y pararse a reflexionar [15][20]. Debido a que la transformación, en la Cuarta Revolución Industrial, producto de innovaciones desde la biotecnología hasta la inteligencia artificial para redefinir el ser humano, se basa en varias investigaciones para describir incidencias que tendrán de manera individual y colectiva, entre ellas: la afectación de la identidad, la longevidad, la salud, la privacidad, los procesos cognitivos, la manera de relacionarnos con los demás, tiempo dedicado al ocio y al trabajo, desarrollo profesional y por supuesto el proceso formativo, especialmente para reconocer las competencias requeridas en la Revolución 4.0 para poder identificar, elegir o reconducir alternativas, personales, académicas y profesionales.

Coincide con otros autores [8][18], en la relevancia a percatarse de "que las plataformas están diseñadas estratégicamente para crear adicción", que la tecnología no es neutral y las consecuencias son obvias. Igualmente, que el impacto negativo augura aumento de la brecha social, eliminación de un número significativo de empleos, aunque se pronostica una cantidad considerable de nuevos empleos, declaran que los inconvenientes del proceso evolutivo aún están en la sombra. Describe especialmente inquietante el riesgo de profundizar las desigualdades, al aumentar la brecha entre los rendimientos del capital y los del trabajo en un mercado laboral cada vez más dual, es decir, sectores de baja cualificación con exiguos salarios y los de cualificaciones altas con remuneraciones elevadas que pudieran dar lugar al aumento de tensiones sociales.

Según estudio del Banco Interamericano de Desarrollo, las habilidades más valoradas será la capacidad para "resolver problemas complejos", en un mundo cada vez más automatizado, se valora el "pensamiento crítico", "la creatividad", de un puesto 10 en el 2015, pasa a ser la tercera habilidad más valorada. En ese mismo año la capacidad de "negociación y la flexibilidad" con un lugar destacado desciende por el avance del uso de la inteligencia artificial en el análisis de grandes volúmenes de datos y la inteligencia artificial. La "escucha activa" es sustituida por la "inteligencia emocional" como habilidad crucial [22].

B. Sobre las Tecnologías Emergentes

El desarrollo de la industria 4.0 hasta los momentos tiene presente tecnologías, conceptos y funcionalidades que ya forman parte de la cotidianidad, entre ellas:

- *API (Application Programming Interface):* Es una interface de desarrollo de aplicaciones que establece la relación entre varias aplicaciones para permitir el intercambio de datos. Se prevé se conviertan en pieza fundamental basada en Inteligencia Artificial para el establecimiento de nuevos modelos de negocio empresariales y nuevas líneas de investigación.
- *Big Data:* capacidad de coleccionar, almacenar y analizar grandes cantidades de datos para convertir los datos en información útil para facilitar la toma de decisiones, incluso en tiempo real.
- *Blockchain (Cadena de bloques):* Se inició como tecnología utilizada para la creación de monedas virtuales, sigue evolucionando hacia una plataforma alternativa a los

actuales mecanismos centralizados de transacción y mantenimiento de registros, favoreciendo la interacción de las empresas sin intermediarios en las operaciones. Es considerada una auténtica revolución para las empresas inmersas en la transformación digital.

- *Chatbot:* Sistema basado en Inteligencia Artificial, Aprendizaje Automático y Procesamiento de Lenguaje Natural que permite establecer plataformas conversacionales con robots como si estuviésemos hablando con humanos. Se prevé su utilidad para que las empresas ofrezcan al usuario una experiencia personalizada e integral.
- *Cyberseguridad:* Define los conceptos que rigen la seguridad a través de garantizar la integridad de las gestión y operatividad digital de las organizaciones, contempla habilidad en técnicas de seguridad en sistemas de telecomunicaciones.
- *Simulation:* Es la representación matemática de un fenómeno físico que nos permite el estudio tanto de procesos físicos, como de ingeniería, económicos e incluso biológicos. Capacidad de concebir, modelar, implementar, operar y optimizar productos y procesos en ambientes virtuales. Permite conocer lo que está pasando.
- *Universal System Integration:* capacidad de integración física-virtual y horizontal – vertical de todos los sistemas productivos en la fábrica digital.
- *Industrial IoT:* Permitirá reducir el tiempo de inactividad de las máquinas y conseguir una ocupación mayor. Es la habilidad de conexión industrial de Internet en tiempo real de dispositivos, plantas, oficinas y compañías para compartir información.
- *Cloud Computing:* capacidad de computación en la nube de IoT y de big data. Conforma un espacio con capacidad ilimitada para guardar información y con funcionamiento que favorece la conectividad y un modelo ágil orientado a la prestación de servicios.
- *Additive Manufacturing:* capacidad de integración de elementos físicos con elementos virtuales para crear una realidad aumentada en tiempo real en la fábrica digital.
- *Impresión 3D:* Es uno de los pilares de la transformación de la industria y con un potencial cada vez más al alza. Permite la fabricación y elaboración de prototipos cuya efectividad puede ser probada y adaptada más rápido.
- *Inteligencia Artificial:* Son sistemas que aprenden y se adaptan de forma autónoma a la empresa. Aplicada a la producción, permitirá automatizar los procesos para dotar de mayor valor a la organización.
- *Internet de las Cosas (IoT):* Todo estará conectado, comprende tecnología inteligente que conecta todas las áreas productivas, permite el seguimiento del proceso, disminuye riesgos y prevé mejoras para una producción que rozará la perfección.
- *Nube Transversal (Edge Computing):* Permite un modelo productivo no centralizado. Es un modelo basado en el procesamiento de información y recopilación para aproximarse a las fuentes de esa información.
- *Realidad Virtual (VR) y Realidad Aumentada (AR):* Comprende la combinación de experiencias reales y virtuales. Permite generar una experiencia inmersa que permita examinar escenarios de la vida real en los que simular aspectos de productividad que requieran ser analizados.
- *Wearables:* Son dispositivos electrónicos que se incorporan en alguna parte del cuerpo en forma de prenda o complemento. Son habilitadores digitales que explotan el potencial del Internet de las Cosas, convirtiendo elementos físicos en información digital para su posterior tratamiento [22-26].

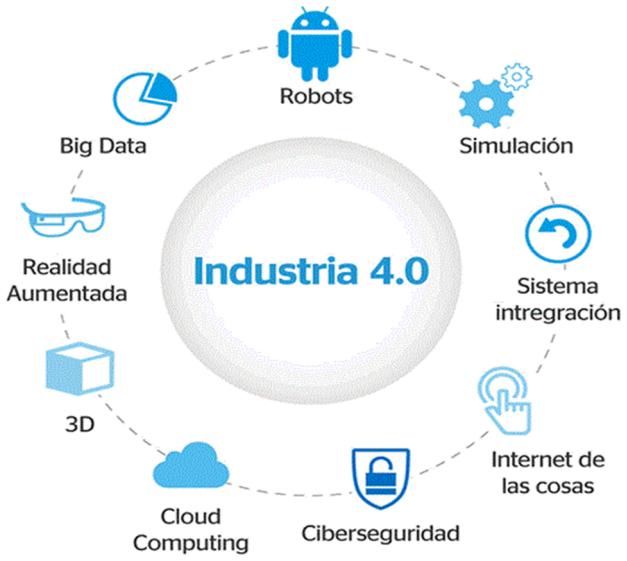

Figura 2: Pilares de la Industria 4.0

La conjunción de los pilares tecnológicos de la Industria 4.0, destaca: sistemas ciberfísicos de integración: máquinas y sistemas autónomos (robots), internet de las cosas (IoT), manufactura aditiva (impresión 3D); big data y análisis de macro datos; computación en la nube; simulación de entornos virtuales; inteligencia artificial; ciberseguridad; y realidad aumentada; esta conjugación traerá transformaciones profundas que se conocerán y conectarán en tiempo real a todos los actores sociales mediante Internet. Esto significa que la sociedad global conocerá lo que está ocurriendo con estas tecnologías en otras esferas [22]. La conjunción establecida en las organizaciones y en la vida en general, generará una enorme cantidad de datos, que gracias a los nuevos sistemas computacionales y algoritmos avanzados, pueden ser procesados y analizados minimizando el esfuerzo humano. Generando cambios significativos en la forma de operar con los datos, entre ellos: descentralizar y mejorar la toma de decisiones, instrumentar modelos predictivos en todas las áreas de las organizaciones; instrumentar sistemas de detección de fallas de equipos, de logística, entre otros. Los pilares de la Industria 4.0 son reseñados en múltiples investigaciones [4][5][10][23-26] como se presentan en la Figura 2.

Cada una de estas tecnologías contiene un potencial de transformación de sectores, vienen desarrollándose y se encuentran en un estado de madurez que le permite asociarse con otras. Todos los días seguirán sumándose tecnologías que se complementan como: drones, sensores inteligentes, controladores, plataformas electrónicas abiertas, sistemas de localización, sistemas de autoidentificación y blockchain.

Las investigaciones refieren que la Inteligencia Artificial (IA) llegó para quedarse, no es ni siquiera el mismo concepto que en sus inicios se tenía de ella. La IA en sus comienzos se pensaba en procesos programados definidos en distintos niveles de toma de decisiones, hoy el conjunto de tecnologías que acompañan la Inteligencia Artificial como el IoT, la sensórica y la robótica refieren actividades más predictivas, esto por supuesto no aleja las preocupaciones sobre todo para que el funcionamiento de la Inteligencia Artificial tenga la garantía de estar a la entera disposición de la humanidad, al respeto irrestricto de la dignidad humana y la realización universal de los derechos humanos [19].

La Conferencia Regional de Educación Superior [2] declara la importancia de las instituciones de educación superior como instituciones estratégicas para coadyuvar en la disminución y la superación de las brechas y asimetrías existentes entre los países, en especial los países Latinoamericanos y el Caribe; en la ciencia, la tecnología, la innovación, la cultura y el progreso humano para asumir la responsabilidad con el cumplimiento de los 17 Objetivos de Desarrollo Sostenible (ODS) [2], casi todos los objetivos muestran el propósito de asegurar que la inminente transformación industrial se lleve a cabo de manera sostenible [24]. Declaran el avance social, la generación de riqueza, una cultura de paz, la integración e identidad social, el afianzamiento de la identidad y la lucha contra el hambre y la pobreza. El llamado es a impulsar el desarrollo científico, tecnológico y educativo orientado a la innovación desde el sector académico articulado con las organizaciones públicas y privadas, al reconocimiento y oportunidades de atender las necesidades de Investigación, Desarrollo e Innovación (I+D+i) [2].

Las investigaciones coinciden en el cambio de la sociedad tal y como la conocemos, la transformación de la vida, de la humanidad en la cual los profesionales tienen un papel protagónico, menos individual, donde debe reconocer el escenario, la ética y la transformación en la cual participa.

Actualmente el Programa Nacional de Formación en Informática (PNFI), contempla en su propuesta 2019 [27], un núcleo básico fortalecido en las áreas de matemáticas, programación, base de datos, ingeniería de software, sistemas operativos, investigación de operaciones e introducción a la inteligencia artificial. Actualiza igualmente el área de comunicación oral y escrita y muy especialmente el área de pensamiento crítico donde se incluye el desarrollo personal, ético político, profesional, de innovación y emprendimiento. Articula las unidades curriculares a las líneas de investigación de Ingeniería de Software, Sistemas Inteligentes y Humanístico Socio Dialéctico encargada del eje de pensamiento crítico.

IV. Educación 4.0: Laboratorios de Prototipado y Construcción de Competencias Transversales

En estudio exploratorio realizado sobre las Implicaciones de la Industria 4.0 en la educación superior [28] muestra una visión general entre los conceptos de industria 4.0 y la educación 4.0, señalan que a través del diseño, desarrollo y evaluación de una secuencia didáctica aplicada, de bajo costo, bajo el enfoque de pensamiento computacional [11] y activo, puede llevarse a cabo el aprendizaje de conceptos de alto nivel, sin comprometer la calidad y la profundidad de los conceptos tratados, así como la contextualización social y profesional, especialmente dedicados a las actividades que deberán enfrentar los estudiantes como futuros profesionales.

Reseñan que el enfoque 4.0 tiene que ver con el uso de las tecnologías, en consecuencia emerge rápidamente la brecha digital, entre países, organizaciones y personas. Sin embargo, no se debe considerar que la tecnología per se fomenta o propicia escenarios basados en el enfoque 4.0. Exaltan, el ser humano juega la preponderancia en el enfoque. Para esto, sugieren replantear la práctica docente, construir un equilibrio entre las actividades docentes, de facilitador del aprendizaje en la impartición de la cátedra, la actualización profesional técnica del docente en las tecnologías emergentes y la conformación de equipos de Desarrollo, Investigación e innovación (D+I+i). De tal manera, la formación de los nuevos profesionales es producto de la construcción y valoración colectiva de conocimientos, habilidades y destrezas del equipo docente de la unidad curricular, de los ejes de formación, investigación, grupos de innovación, entre otros. En otras palabras, la conformación de equipos, con las mejores prácticas, la construcción de herramientas orientadas a la construcción de habilidades blandas o liquidas invita a los docentes a cuestionar la realidad a través de ejercicios reflexivos y metacognitivos que logre transferir a los estudiantes para que los estudiantes reconozcan lo que saben, descubran lo que no saben y propongan las acciones remediales respectivas [29]. El equipo académico a través de la metacognición debe conformar los espacios propicios para la interdisciplinariedad, el trabajo en equipo, la resolución de problemas, entre otros. Los estudiantes pueden reconocer y controlar su aprendizaje, pensar en forma crítica y reflexiva, actuar con autonomía, resolver problemas, tomar decisiones y determinar las estrategias para lograr los objetivos del aprendizaje. Construir las competencias trasversales modeladas por el equipo docente implica la vivencia de la tolerancia a la frustración, creatividad y proactividad, flexibilidad y adaptabilidad, disposición al aprendizaje y rectificación de caminos; aspectos que impulsan la inteligencia emocional en las personas [11-13].

El proceso educativo debe acelerar una adaptación a los ritmos de evolución de la cuarta revolución industrial en el sector industrial, de lo contrario se puede propiciar una ruptura, un desfase, en la relación industria–academia generando un conflicto y cuestionamiento con respecto a las funciones fundamentales de la universidad, este proceso sólo es posible conformando los equipos académicos, de Desarrollo, Investigación e Innovación institucional en cada institución universitaria.

La integración de tecnologías, metodologías, herramientas, enfoques y procesos propiciará nuevos enfoques de trabajos, puntos de vistas, interacción en las sociedades digitalizadas y sobre todo del aprendizaje [13], entre ellos el pensamiento computacional [28] para resolver problemas, diseñar sistemas y comprender el comportamiento humano haciendo uso de los conceptos fundamentales de la informática. El pensamiento computacional es descrito como el desarrollo sistemático de habilidades del pensamiento crítico, el pensamiento lateral en la resolución de problemas utilizando la capacidad de abstracción y las técnicas de resolución de problemas utilizados por los científicos e ingenieros de la computación y que pueden ser enseñados y aplicados en otras disciplinas. Este enfoque permite construir una nueva dinámica que debe enfrentarse a través de la incorporación de laboratorios de aprendizaje como estrategia formativa para operacionalizar el desarrollo de prototipos de sistemas informáticos, donde los estudiantes

desarrollen en equipos de trabajo, desde los ejercicios más sencillos, rutinas o módulos de procesos de distinto nivel hasta sistemas automatizados. Este enfoque permitirá dinamizar, discutir y desarrollar el cumplimiento de distintas etapas de la enseñanza aprendizaje en las distintas fases de la ingeniería de desarrollo de software en prototipos base. Este proceso demanda el diseño de guías de laboratorios con distintos ejemplos de procesos, módulos, rutinas hasta llegar a completar un sistema informático desarrollado en horas de laboratorio teórico práctico. Modelar en su práctica las competencias transversales del equipo docente en la conformación de equipos de trabajo, empatía, solidaridad, creatividad, innovación, desprendimiento y resiliencia ante los desafiantes retos por venir.

Es oportuno igualmente asociar a las estrategias de desarrollo de soluciones informáticas, procesos de investigación e innovación, dada la oportunidad de reconocer los cambios en la introducción de nuevas estrategias formativas. La innovación surge, según Rave [10], de las fronteras de la interdisciplinariedad, las orientaciones sobre las cuales surge la Cuarta Revolución Industrial, ya reconoce la necesidad del trabajo en equipo interdisciplinario, transformador, veloz, dinámico y enfocado.

Implica igualmente la conformación de equipos de docentes multidisciplinarios, capaces de analizar, construir y adoptar las competencias transversales, las habilidades blandas o liquidas que los nuevos profesionales necesitan aprender. La conformación de consejos científicos, de investigación y desarrollo, en permanente trabajo y revisión de la producción alcanzada.

V. Conclusiones y Trabajos Futuros

Las ciencias de la computación, informática y sistemas se desarrollan exponencialmente, en consecuencia, las instituciones universitarias que forman los profesionales que demanda la sociedad tienen la responsabilidad de garantizar la calidad académica, a fin de detectar la actualización de los conocimientos, habilidades y destrezas que se pretenden desarrollar en los estudiantes que requiere la sociedad actual y futura.

La identificación del núcleo formativo debe estar acompañado de la identificación de las áreas para formar las competencias trasversales, habilidades interpersonales, cognitivas y emocionales, destacan: el trabajo en equipo, la resolución de problemas complejos y la toma de decisiones, el pensamiento crítico y reflexivo, la innovación, la investigación y el emprendimiento.

La actualización curricular permanente debe contener la actualización del personal docente dedicada a la gestión del proceso formativo, así como el manejo y aplicación de las competencias transversales.

Las competencias profesionales y técnicas están fuertemente acompañadas de competencias transversales, habilidades blandas o líquidas, en consecuencia el proceso formativo debe contribuir en su construcción en el estudiante y en los docentes.

La adecuación y fortalecimiento del currículo demanda a la actualización del personal docente dedicado a la gestión del proceso formativo y a la construcción de guías de laboratorio para el desarrollo de sistemas informáticos donde se cumplan las fases de la ingeniería de software y los estudiantes puedan desarrollar sistemas conformándose equipos de desarrollo de cualquier organización y pongan en práctica las competencias técnicas y blandas.

La construcción de competencias digitales incorpora la planificación estratégica, la transdisciplinariedad y el monitoreo permanente interno y externo de las organizaciones.

La construcción de guías de laboratorio para el desarrollo de sistemas informáticos por parte de los estudiantes permitirá avanzar en el cumplimiento de las distintas fases de la ingeniería de desarrollo de software permitirá adoptar la estrategia de horas de laboratorio que contempla el currículo que se gestiona en China y en otras regiones. De vital importancia el seguimiento de su implementación.

Como trabajo futuro, la revisión permanente de las implicaciones de un modelo curricular en constante evaluación y fortalecimiento demanda investigaciones y propuestas con valor científico que permita la toma de decisiones más acertada, la orientación de investigaciones e innovaciones desde las instituciones universitarias hacia su contexto.

Referencias

[1] J. L. del Val Román. Conferencia de Directores y Decanos de Ingeniería Informática CODDIInforme (2016). *Industria 4.0. la Transformación Digital de la Industria.* Facultad de Ingeniería de la Universidad de Deusto, 2016.

[2] UNESCO. *El Papel Estratégico de la Educación Superior en el Desarrollo Sostenible de América Latina y el Caribe*, 2018.

[3] K. Schwab, *La Cuarta Revolución Industrial.* Foro Económico Mundial, Octubre 2016.

[4] Tendencias 21. *Tendencias Informáticas: Los Ingenieros Informáticos, Pieza Clave de la Industria 4.0.* http://www.tendencias21.net.

[5] R. Blanco, J. Fontodrona, y C. Poveda. *La Industria 4.0: El Estado de la Cuestión*. Revista Dialnet, no. 406, pp. 151-164, Fundación Dialnet, 2017.

[6] Universidad Pontificia Bolivariana. Escuela de Ingeniería. Ingeniería de Sistemas e Informática. *Programa Educativo del Programa (PEP). Formación Integral para la Transformación Social y Humana,* Medellin, Colombia. 2017.

[7] Universidad Católica San Pablo. *Plan Curricular 2016 de la Escuela Profesional de Ciencia de la Computación*. http://cs.ucsp.edu.pe.

[8] E. Samanes y C. Martínez. *Revolución 4.0. Competencias, Educación y Orientación*. Revista Digital en Docencia Universitaria, vol. 12, no. 2, pp. 4-34, 2018.

[9] F. J. García. *Los Estudios de Ingeniería en Informática*. Área de Ciencia de la Computación e Inteligencia Artificial. Departamento de Informática y Automática. Universidad de Salamanca, 2018.

[10] R. Rave. *La Cuarta Revolución Industrial en Latinoamérica.* http//www.cnnespanol.cnn.com.

[11] J. M. Wing. *Computational Thinking*. Communications of the ACM, vol. 49, no. 3, pp. 33-35, March 2006.

[12] C. Chaves. *Habilidades del Siglo XXI*. Revista Conexiones. pp. 12-18, 2016.

[13] F. Vera. *Infusión de Habilidades Blandas en el Curriculo de la Educación Superior: Clave para el Desarrollo de Capital Humano Avanzado*. Revista Akademeia, vol. 15, no. 1, pp. 53-73, 2016.

[14] UNESCO. *Foro Políticas Públicas para la Innovación 4.0. Ciencia, Tecnología Cultura e Innovación en la Cuarta Revolución Industrial*. vol. 47, no. 5, pp. 6. Oficina de México. Agosto 2017.

[15] C. Roing. *Industria 4.0: La Cuarta (Re)Evolución Industrial.* Harvard Deusto Business review, no. 266, pp. 64-70. https://dialnet.unirioja.es/ servlet/ articulo?codigo-5909151.

[16] IndustriALL Global Union. *El Desafío de la Industria 4.0 y la Exigencia de Nuevas Respuestas.* Oficina de América Latina y el Caribe, Montevideo, Uruguay, 2017.

[17] Association for Computing Machine (ACM), Institute of Electrical and Electronics Engineers (IEEE), *Computer Engineering Curricula 2016 (CE2016)*, *Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*, December 2016.

[18] E. Cuadros. *Plan Curricular 2016 de la Escuela Profesional de Ciencia de la Computación.* Escuela Profesional de Ciencias de la Computación. Universidad Católica San Pablo. Sociedad Peruana de Computación, 2016.

[19] J. de la Vega e I. Aguilar. *Comparativa de la Guía Curricular ACM/IEEE CE2004 y ACM/IEEE CE2016.* Revista Iberoamericana de las Ciencias Computacionales e Informática, vol. 7, no. 13, 2018.

[20] D. Susskind y R. Susskind. *The Future of the Professions*. *How Technology will Transfor the Work of Human Experts*. Oxford University, 2018.

[21] S. Guerra. *Una Revisión Panorámica al Entrenamiento de las Habilidades Blandas en Estudiantes Universitarios*. Revista Scielo, vol. 23. Corporación Universitaria. Bogotá, Colombia, 2019.

[22] Grupo Garatu. *IT Solutions*. https://grupogaratu.com/que-es-y-que-aporta-la-industria-4-0.

[23] SERESCO.S.A. *15 Conceptos Básicos para Entender la Industria 4.0.* http://www.industria4.es/empresa/entender-la-industria-4-0.

[24] A. Basco, G. Belitz, D. Coatz y P. Gamero: *Industria 4.0. Fabricando el Futuro*. Banco Interamericano de Desarrollo. Buenos Aires, Argentina, 2018.

[25] F. Yañez. *La Meta es la Industria 4.0. Descubre la Tecnología que Hace Posible la Nueva Revolución Industrial.* Independently published, 2017.

[26] F. Yañez. *Las 20 Tecnologías Clave de La Industria 4.0: El Camino Hacia la Fábrica del Futuro*. Independently published, 2019.

[27] MPPEU. *Programa Nacional de Formación en Informática (PNFI)*. Propuesta actualización curricular 2019.

[28] F. Blanco, J. M. Castro, R. A. Gayoso y W. Santana. *Las Claves de la Cuarta Revolución Industrial. Cómo Afectará a los Negocios y a las Persona.* Barcelona, Madrid, 2019.

[29] D. Sánchez. *Industria y Educación 4.0 en México: Un Estudio Exploratorio*. Revista Innovación Educativa, vol. 19, no. 81, pp. 39-63. Tema: Implicaciones de la Industria 4.0 en la Educación Superior. Educación. Secretaria de Educación Publica. México, 2019.

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 6, No. 2, pp. 10-20, Diciembre 2019
Fecha recepción: 29/11/2019, Fecha aceptación: 10/04/2020

Precondición Más Débil de Algoritmos de Punto Fijo

Federico Flaviani[1]
fflaviani@usb.ve

[1] Departamento de Computación, Universidad Simón Bolívar, Caracas, Venezuela

Resumen: Dijkstra definió el transformador de predicados wp (*weakest precondition*), y en dicha definición establece para la instrucción de iteración Do un predicado $H_k(\text{Post})$ que describe los estados iniciales que hacen que el ciclo itere a lo sumo k veces, satisfaciendo la postcondición Post. En trabajos recientes se han determinado técnicas para calcular explícitamente $H_k(\text{Post})$, lo cual representa una alternativa a la regla de Hoare del invariante y una forma de calcular precondiciones más débiles de Do. No existen muchas técnicas prácticas, que permitan calcular explícitamente la precondicion más débil wp, por lo tanto historicamente ha sido más versatil usar las reglas de Hoare para obtener precondiciones, con lo cual se sacrifica la generalidad de las aserciones. En este trabajo se muestra como hacer este cálculo explícito de wp para una familia de algoritmos de punto fijo. Esta familia de algoritmos, junto con sus precondiciones más débiles, representan un framework para construir algoritmos de punto fijo correctos.

Palabras Clave: Corrección Formal de Programas; Precondición más Débil; Derivación de Algoritmos.

Abstract: Dijkstra defined the predicate transformer called wp (weakest precondition), and that definition establishes the predicate $H_k(\text{Post})$ for the iteration statement DO, that describes the initial states that forcing the loop to iterate at most k times, with a final state that satisfy the post-condition Post. In recent works, some techniques have been developed to calculate explicitly $H_k(\text{Post})$, which represents an alternative to invariants of Hoare's rule, and a way to calculate weakest precondition of DO. There are not many practical techniques to explicitly calculate the weakest precondition wp, therefore historically it has been more versatile to use Hoare's rules to obtain preconditions, thereby sacrificing the generality of assertions. This work shows how to do this explicit calculation of wp for a family of fixed point algorithms. This family of algorithms, along with their weakest preconditions, represent a framework for building correct fixed point algorithms.

Keywords: Formal Program Verification; Weakest Precondition; Derivation of Algorithms.

I. Introducción

En este trabajo se realiza un análisis de una familia de algoritmos de punto fijo para calcular un conjunto. Dada una postcondición que enuncia que cierta variable contiene el conjunto deseado y un algoritmo de punto fijo, se estudia la precondición más débil y la condición de terminación de los ciclos de algoritmos de punto fijo, usando las técnicas de [1][2].

Para unificar la presentación de los algoritmos en este trabajo, todos los algoritmos serán escritos en pseudolenguaje *GCL* (*Guarded Command Language*) [3], que es un pseudolenguaje definido por Dijkstra, donde se pueden escribir algoritmos no determinísticos. GCL admite una lógica de Hoare [4] y fórmulas para precondiciones más débiles, relativamente simples, que facilitan la actividad de corrección de un programa.

Si Pre y Post son predicados para ser usados como precondición y postcondición de una instrucción S, entonces la lógica de Hoare [4] se basa en la noción de tripletas $\{\text{Pre}\}S\{\text{Post}\}$, que se entienden como un predicado que es verdadero si y sólo si la instrucción S manda los estados del programa que satisfacen Pre a estados que satisfacen Post.

En este trabajo se supone que ninguna expresión tiene valores indefinidos, sino más bien, indeterminados. Por ejemplo si $\{R_i\}_{i=a}^{n}$ es una familia de conjuntos, entonces si i es de tipo entero, se supone que la expresión R_i está definida para todo i entero, solo que para $i < a \vee i > n$ no se conoce el valor de R_i, pero sí existe. De esta forma un predicado de la forma $P(R_i)$ tiene siempre un valor de verdad, solo que para $i < a \vee i > n$ no se conoce. Al modificar el predicado anterior por $a \leq i \leq n \wedge P(R_i)$, se tiene que siempre se puede calcular el valor de verdad, ya que para los valores de i que no se conoce el valor de verdad de $P(R_i)$, sí se conoce que $a \leq i \leq n$ es falso y por lo tanto todo el predicado sería falso.

La abreviación $\text{domain}(Exp_1, \ldots, Exp_n)$ [5] referencia a

un predicado que indica los valores de las variables de las expresiones $Exp_1, \ldots, Exp_n$, en que el valor de todas ellas están determinadas. Por ejemplo $\mathrm{domain}(R_i) \equiv a \leq i \leq n$. Al igual que en el párrafo anterior, un predicado cualquiera $P(Exp_1, \ldots, Exp_n)$ siempre puede ser modificado por $\mathrm{domain}(Exp_1, \ldots, Exp_n) \wedge P(Exp_1, \ldots, Exp_n)$, para que su valor de verdad siempre se pueda calcular, incluso cuando el valor de las expresiones esté indeterminado. En el presente trabajo se sigue la filosofía de usar siempre predicados de este tipo.

Salvo en la Sección XI, las nomenclaturas $\Rightarrow$ y $B \rightarrow S$ denotan la implicación y una guardia con condición B e instrucción S respectivamente. Solamente en la Sección XI, estas nomenclaturas denotarán la relación de derivación y una producción de una gramática libre de contexto respectivamente.

En este trabajo si se abrevian a S_i con $0 \leq i \leq n$ como instrucciones, entonces se denota a $S_i; S_j$ como la secuenciación de instrucciones, se denota IF como la instrucción de selección:

if $\mathtt{B_0} \rightarrow$
 | $\mathtt{S_0}$
[ ] $\mathtt{B_1} \rightarrow$
 | $\mathtt{S_1}$
 | $\vdots$
[ ] $\mathtt{B_n} \rightarrow$
 | $\mathtt{S_n}$
fi

y la abreviación DO como la instrucción de iteración con múltiples guardias:

do $\mathtt{B_0} \rightarrow$
 | $\mathtt{S_0}$
[ ] $\mathtt{B_1} \rightarrow$
 | $\mathtt{S_1}$
 | $\vdots$
[ ] $\mathtt{B_n} \rightarrow$
 | $\mathtt{S_n}$
od

Se denotará como Do a la instrucción de iteración con una sola guardia **do** $B_0 \rightarrow S_0$ **od**.

Se denota SKIP a la instrucción que no cambia el estado de ejecución, se denota $\overline{y}$ y $\overline{Exp}$ como listas de variables y expresiones respectivamente, y $[\overline{y} := \overline{Exp}]$ como el operador que sustituye en paralelo, las variables $\overline{y}$ por las expresiones $\overline{Exp}$ en predicados.

Para hacer derivaciones lógicas sobre tripletas de Hoare, se definen para el lenguaje *GCL*, las siguientes reglas de inferencia, en donde $B_0, \ldots, B_n$ son expresiones booleanas del lenguaje *GCL* y *DG* es una abreviación de $\mathrm{domain}(B_0, \ldots, B_n)$:

$$\{A\}\mathrm{SKIP}\{A\}$$

$$\{\mathrm{domain}(\overline{\mathrm{Exp}}) \wedge B[\overline{y} := \overline{\mathrm{Exp}}]\}\overline{y} := \overline{\mathrm{Exp}}\{B\}$$

$$\frac{\{A\}S_0\{B\} \qquad \{B\}S_1\{C\}}{\{A\}S_0; S_1\{C\}}$$

$$\frac{\begin{array}{ccc} A \Rightarrow DG & \{A \wedge B_0\} & \ldots \{A \wedge B_n\} \\ \wedge (B_0 \vee \cdots \vee B_n) & S_0 & S_n \\ & \{B\} \ldots & \{B\} \end{array}}{\{A\} \text{ if } B_0 \rightarrow S_0[\,]\cdots[\,]B_n \rightarrow S_n \text{ fi } \{B\}}$$

$$\frac{A \Rightarrow A' \qquad \{A'\}S_0\{B'\} \qquad B' \Rightarrow B}{\{A\}S_0\{B\}}$$

$$\frac{\begin{array}{cc} & \{\mathrm{Inv} \wedge B_0 \wedge 0 \leq Exp = Z\} \\ \mathrm{Inv} \Rightarrow \mathrm{domain}(B_0) & S_0 \\ & \{\mathrm{Inv} \wedge 0 \leq Exp < Z\} \end{array}}{\{\mathrm{Inv}\} \text{ do } B_0 \rightarrow S_0 \text{ od } \{\mathrm{Inv} \wedge \neg B_0\}}$$

El predicado Inv de la última regla de inferencia se conoce como invariante, y éste se usa para verificar la correctitud de una Tripleta de Hoare que involucre un ciclo.

Dijkstra adicionalmente en [3] definió el transformador de predicados wp para el lenguaje *GCL*, que es una función que recibe una instrucción y una postcondición sintácticamente hablando y devuelve la precondición más débil de la instrucción y la postcondición. Si BB es una abreviación del predicado $B_0 \vee \cdots \vee B_n$, entonces las reglas que definen wp son las siguientes:

- $\mathrm{wp}(\mathrm{SKIP}, \mathrm{Post}) := \mathrm{Post}$
- $\mathrm{wp}(y_{i_1}, \ldots, y_{i_k} := \mathrm{Exp}_1, \ldots, \mathrm{Exp}_k, \mathrm{Post}) :=$
  $\mathrm{domain}(\mathrm{Exp}_1, \ldots, \mathrm{Exp}_k) \wedge$
  $\mathrm{Post}[y_{i_1}, \ldots, y_{i_k} := \mathrm{Exp}_1, \ldots, \mathrm{Exp}_k]$
- $\mathrm{wp}(S_0; S_1, \mathrm{Post}) := \mathrm{wp}(S_0, \mathrm{wp}(S_1, \mathrm{Post}))$
- $\mathrm{wp}(\mathrm{IF}, \mathrm{Post}) := \mathrm{domain}(B_0, \ldots, B_n) \wedge$
  $(B_0 \vee \cdots \vee B_n) \wedge (B_0 \Rightarrow \mathrm{wp}(S_0, \mathrm{Post})) \wedge \ldots$
  $\wedge (B_n \Rightarrow \mathrm{wp}(S_n, \mathrm{Post}))$
- $\mathrm{wp}(DO, \mathrm{Post}) := (\exists k | k \geq 0 : H_k(\mathrm{Post}))$ en donde $H_k(\mathrm{Post})$ es un predicado que satisface las ecuaciones

$$H_0(\mathrm{Post}) \equiv \mathrm{domain}(BB) \wedge \neg(BB) \wedge \mathrm{Post}$$
$$H_k(\mathrm{Post}) \equiv H_0(\mathrm{Post}) \vee \mathrm{wp}(\mathrm{IF}, H_{k-1}(\mathrm{Post}))$$

  para $k \geq 1$

El predicado $H_k(\mathrm{Post})$ en la definición de $\mathrm{wp}(DO, \mathrm{Post})$ anterior, describe el conjunto de estados que hacen que el ciclo DO itere a lo sumo k veces satisfaciendo la postcondición Post al final de la ejecución.

Trabajar con $H_k(\mathrm{Post})$ como precondición de un ciclo, tiene la ventaja de que la expresión k determina el número máximo de iteraciones que el ciclo puede realizar, por lo tanto se puede estimar la complejidad en tiempo de ejecución. Por otro lado intentar usar $H_k(\mathrm{Post})$ en lugar de cualquier otro invariante puede ser conveniente ya que en los trabajos [6][1] se ha venido planteando, que en algunos casos es muy fácil calcular $H_k(\mathrm{Post})$ formalmente, sin necesidad de conjeturar un invariante y usar las reglas de Hoare para validar.

Por otro lado Morgan en [7], muestra una forma de abstraer partes del código de un algoritmo y aún así, seguir calculando la precondición más débil del algoritmo dado una postcondición. Estas abstracciones de código reciben el nombre de "instrucción de especificación", ya que son manejadas dentro del algoritmo como si fueran instrucciones, aunque su código aún no se conoce. Por medio de instrucciones de especificación es que en este trabajo se hacen las abstracciones necesarias,

para describir la familia de algoritmos de punto fijo, que es objeto de estudio.

A. Contribución

Dada la existencia de una sucesión de conjuntos $\{R_i\}_{i=a}^{n}$, se define una familia de algoritmos de punto fijo, que computan el punto fijo de la sucesión de conjuntos, la familia se describe por medio de instrucciones de especificación de Morgan [7]. Dada una postcondición que indica que en cierta variable R, se encuentra almacenado el punto fijo de la sucesión (Es decir que $R = R_i = R_{i-1}$), entonces se demuestra que todos estos algoritmos con la postcondición descrita anteriormente, tienen como precondición más débil a $(\exists \epsilon | a \leq \epsilon \leq n-1 : R_\epsilon = R_{\epsilon+1})$. Esta demostración se hace conjeturando un predicado y demostrando que ese predicado es un $H_k(Post)$.

Adicionalmente se muestra que la técnica para conjeturar $H_k(Post)$ definida en [1] aplica para este caso, mostrando cómo fue el procedimiento para conjeturar el predicado que se usó para calcular la precondición más débil.

Durante los cálculos del transformador de predicados wp, se usan las propiedades algebraicas descritas en [2], mostrando cómo el uso de estas propiedades facilita el proceso, dando así una pequeña contribución, en aras de construir un cálculo práctico para precondiciones más débiles.

La familia de algoritmos definida, junto con sus abstracciones de código y precondición más débil representan un framework para construir algoritmos de punto fijo correctos. Como ejemplo del uso de este framework, se instancia la familia en el algoritmo de búsqueda de símbolos anulables, para hacer limpieza de gramáticas libre de contexto. Esto se hace refinando las instrucciones de especificación y demostrando que la precondición más débil del algoritmo para este caso es una tautología.

B. Trabajos Relacionados

Originalmente Dijkstra en [3] definió el transformador de predicados wp con las reglas recursivas anteriores, salvo que no usó el predicado domain. Posteriormente en [5], buscando que las aserciones de *GCL* sean totalmente evaluables, es decir que no existan estados donde el valor de verdad de la aserciones sea indefinido, se definió por primera vez el predicado domain, pero solo en la definición de wp de la asignación. Luego en trabajos recientes como [8][9], se usa domain también en la regla de definición de wp para el IF. Más adelante n [10] se hace una demostración basada en semántica denotacional, que justifica que para que las aserciones de *GCL* sean totalmente evaluables, se debe usar domain también en la regla del wp del DO al definir $H_0(\mathrm{Post})$. De modo que la definición de wp que se usa y se presentó en este trabajo es la de [10].

Morgan [7] define el concepto de refinamiento, instrucción de especificación y cómo calcular wp de esta instrucción bajo ciertas condiciones sintácticas sobre las variables de especificación de las aserciones. Posteriormente en [11] se revisita la definición de la instrucción de especificación desde un punto de vista denotacional, interpretándola como una relación de estados del programa y con eso se estableció una fórmula general para calcular wp de estas instrucciones. Otro trabajo donde se estudia a la instrucción de especificación y el concepto de refinamiento como relaciones de estados del programa, es [12].

En [13] se muestra una técnica semántica llamada relaciones invariantes, para calcular invariantes en el lenguaje de la teoría de relaciones y wp de un ciclo. Análogamente, pero con técnicas sintácticas, en [6] y posteriormente en [1], se muestran teoremas que sirven para calcular $H_k(\mathrm{Post})$ explícitamente y por lo tanto para calcular wp de Do con postcondición Post. Posteriormente en [2], se demuestra propiedades algebraicas del transformador sintáctico wp usando semántica denotacional, estas propiedades pueden facilitar los cálculos de wp. Estas propiedades fueron usadas para demostrar los teoremas de [1], pero hasta el momento no existen trabajos que las usen para hacer cálculos en concreto y mostrando en qué medida representan un aporte a la praxis de corrección. Este trabajo es el primero en que estas propiedades son usadas para realizar cálculos concretos.

II. Preliminares

En esta sección enunciamos las definiciones y teoremas más recientes, según la bibliografía de este trabajo, relacionados al cálculo de wp y de $H_k(Post)$ de un ciclo. Las demostraciones ausentes de los teoremas de esta sección se pueden conseguir en [1][2].

Lema 1. $wp(S, P \wedge Q) \equiv wp(S, P) \wedge wp(S, Q)$

Notación. *Para abreviar* $\mathrm{wp}(S, \mathrm{true})$ *se usará la notación* $\mathrm{support}(S)$*, donde* S *es una instrucción.*

Así tenemos que $\mathrm{support}(S)$ define el conjunto de estados iniciales más grande en el cual la intrucción S no aborta. En el siguiente lema se muestra un caso, en donde es útil usar $\mathrm{support}$ para calcular wp.

Lema 2. *Sea* P *un predicado y* S *una instrucción que no modifica los valores de las variables de* P*, entonces:*

$$\mathrm{wp}(S, P) \equiv \mathrm{support}(S) \wedge P.$$

Lema 3. *Sea* S *una instrucción, entonces:*

$$\mathrm{support}(S; i := i + 1) \equiv \mathrm{support}(S)$$

Lema 4. *Sea* S *una instrucción (determinística o no) tal que se comporta de forma determinística sobre las variables del predicado* $P \vee Q$*, entonces:*

$$wp(S, P \vee Q) \equiv wp(S, P) \vee wp(S, Q)$$

Lema 5. $wp(S, P) \Rightarrow support(S)$

Lema 6. *Sean* P *y* Q *predicados y* S *una instrucción que no modifica los valores de las variables de* P*, entonces:*

$$wp(S, P \wedge Q) \equiv P \wedge wp(S, Q)$$

y:

$$wp(S, P \vee Q) \equiv support(S) \wedge (P \vee wp(S, Q))$$

Lema 7. *Sean P y Q predicados y S una instrucción que se comporta deterministicamente sobre los valores de las variables de P, entonces:*

$$wp(S, P \Rightarrow Q) \equiv support(S) \wedge (wp(S, P) \Rightarrow wp(S, Q))$$

Lema 8. *Sea P un predicado, S una instrucción que se comporta deterministicamente sobre los valores de las variables de P, y ϵ una variable no declarada en el programa, entonces:*

$$wp(S, (\exists \epsilon | : P)) \equiv (\exists \epsilon | : wp(S, P))$$

Definición 1. *Sea K una expresión y Do una instrucción de la forma:*

do B $\rightarrow$
| S
od
{Post}

Sean además k', ϵ, ϵ' variables no declaradas en el programa (es decir no ocurren en Do) y no ocurren en Post*, donde S es una instrucción (determinística o no determinística). Se definen:*

1. *El predicado $domBG$ como un predicado que satisface:*
   - *En $domBG$ ocurren sólo ϵ' y las variables del programa*
   - $domBG[\epsilon' := 0] \equiv \text{domain}(B)$
   - $domBG \equiv \text{wp}(S, domBG[\epsilon' := \epsilon' - 1])$ *suponiendo que* $0 < \epsilon' \leq K \wedge \text{domain}(B) \wedge B \wedge \text{support}(S)$.
2. *El predicado NBG como un predicado que satisface:*
   - *En NBG ocurren sólo ϵ y las variables del programa*
   - $NBG[\epsilon := 0] \equiv \neg B$
   - $NBG \equiv \text{wp}(S, NBG[\epsilon := \epsilon - 1])$ *suponiedo que* $0 < \epsilon \leq K \wedge \text{domain}(B) \wedge B \wedge \text{support}(S)$.
3. *El predicado $T_{k'}$ como:*

$$(\exists \epsilon | 0 \leq \epsilon \leq k' : (\forall \epsilon' | 0 \leq \epsilon' \leq \epsilon : domBG) \wedge NBG).$$

4. *El predicado $TI_{k'}$ como: $T_{k'} \wedge B$*

En el siguiente teorema se establecen las condiciones suficientes para determinar si un predicado es un $H_{k'}(\text{Post})$.

Teorema 1. *Sean K una expresión, Do una instrucción como en la definición 1 y k', ϵ, ϵ' variables como en la Definición 1. Entonces, si S actúa de forma determinística sobre las variables de $domBG$ y NBG, se satisface lo siguiente:*
Si existe un predicado inv *tal que:*

1. $\text{domain}(B) \wedge \neg B \wedge \text{Post} \equiv \text{domain}(B) \wedge \neg B \wedge \text{inv}$.
2. $(\forall k' | 1 \leq k' \leq K : TI_{k'} \Rightarrow (\text{wp}(S, \text{inv}) \equiv \text{inv}))$

Entonces:

$$H_{k'}(\text{Post}) \equiv T_{k'} \wedge \text{inv}$$

para todo k' tal que $0 \leq k' \leq K$.

El predicado en 1 del teorema anterior se le llamará *Obligación de Prueba de Terminación* y al predicado en 2, *Obligación de Prueba de Iteración*.

Corolario 1. *Si un predicado inv cumple con la obligación de prueba de terminación y de iteración usando k' sin ninguna cota superior K que lo restrinja, entonces definiendo T_∞ como:*

$$(\exists \epsilon | 0 \leq \epsilon : (\forall \epsilon' | 0 \leq \epsilon' \leq \epsilon : domBG) \wedge NBG)$$

se tiene que:

$$wp(Do, Post) \equiv T_\infty \wedge inv.$$

Por otro lado si $H_{k'}(\text{Post})$ es de la forma $T_{k'} \wedge inv$ para $k' \leq K + 1$ y $H_{K+1}(\text{Post}) \equiv H_K(\text{Post})$, entonces:

$$wp(Do, Post) \equiv H_K(\text{Post})$$

Teorema 2. *Si B es la guardia del ciclo Do en la definición 1 y $\text{domain}(B) \equiv$ true, entonces $domBG \equiv$ true.*

Proof: Se demuestra que definiendo $domBG$ como true, se satisfacen las reglas recursivas de la definición 1.

$domBG[\epsilon' := 0] \equiv \text{true}[\epsilon' := 0] \equiv \text{true} \equiv \text{domain}(B)$

Se supone que $0 < \epsilon' \leq K \wedge \text{domain}(B) \wedge B \wedge \text{support}(S)$ y se demuestra lo siguiente:

$domBG \equiv wp(S, domBG[\epsilon' := \epsilon' - 1]) \equiv wp(S, \text{true}[\epsilon' := \epsilon' - 1]) \equiv wp(S, \text{true}) \equiv \cancelto{\text{true}}{\text{support}(S)}$ ■

Teorema 3. *Sean $K_1, \ldots, K_n$ expresiones en donde solo ocurren variables del programa que no son modificadas por S y tales que $0 \leq K_1 < \ldots < K_n < K$. Sean $P_0, \ldots, P_n$ predicados en donde solo ocurren ϵ y variables del programa en las que S actúa de forma determinística. Suponiendo que $\epsilon \leq K$, $\text{domain}(B)$, B, $\text{support}(S)$, $P_0[\epsilon := 0] \equiv \neg B$, $wp(S, P_{i-1}[\epsilon := K_i]) \equiv P_i$ y $wp(S, P_i[\epsilon := \epsilon - 1]) \equiv P_i$ si $i = 0 \wedge 1 \leq \epsilon \leq K_1$ o $i \neq 0 \wedge i \neq n \wedge K_i + 1 < \epsilon \leq K_{i+1}$ o $i = n \wedge K_n + 1 < \epsilon$, entonces:*

$$NBG$$

$$\equiv$$

$$(0 \leq \epsilon \leq K_1 \wedge P_0) \vee .. \vee (K_i < \epsilon \leq K_{i+1} \wedge P_i) \vee .. \vee (K_n < \epsilon \wedge P_n)$$

para $0 \leq \epsilon \leq K$

Proof: En la fórmula NBG del enunciado solo ocurre ϵ y las variables del programa. A continuación se demuestra que NBG satisface las condiciones de la definición 1.

$NBG[\epsilon := 0]$
$\equiv$
$\cancelto{true}{(0 \leq 0 \leq K_1 \wedge P_0[\epsilon := 0])} \vee .. \cancelto{false}{(K_i \leq 0 \leq K_{i+1} \wedge P_i[\epsilon := 0])}$
$\vee \cdots \vee \cancelto{false}{(K_n < 0 \wedge P_n[\epsilon := 0])}$
$\equiv$
$P_0[\epsilon := 0]$
$\equiv$
$\neg B$

Para $\epsilon > 0$ se tiene:

$wp(S, NBG[\epsilon := \epsilon - 1])$
$\equiv$
$wp(S, (0 \leq \epsilon - 1 \leq K_1 \wedge P_0[\epsilon := \epsilon - 1])$
$\vee (\bigvee i | 1 \leq i < n : K_i < \epsilon - 1 \leq K_{i+1} \wedge P_i[\epsilon := \epsilon - 1])$
$\vee (K_n < \epsilon - 1 \wedge P_n[\epsilon := \epsilon - 1]))$

$$
\begin{array}{l}
\equiv \\
wp(S, (1 \leq \epsilon \leq K_1 + 1 \wedge P_0[\epsilon := \epsilon - 1]) \\
\vee(\bigvee i | 1 \leq i < n : K_i + 1 < \epsilon \leq K_{i+1} + 1 \wedge P_i[\epsilon := \epsilon - 1]) \\
\vee(K_n + 1 < \epsilon \wedge P_n[\epsilon := \epsilon - 1])) \\
\equiv \\
wp(S, (1 \leq \epsilon \leq K_1 \wedge P_0[\epsilon := \epsilon - 1]) \\
\vee(\epsilon = K_1 + 1 \wedge P_0[\epsilon := K_1]) \\
\vee(\bigvee i | 1 \leq i < n : (K_i + 1 < \epsilon \leq K_{i+1} \wedge P_i[\epsilon := \epsilon - 1]) \\
\vee(\epsilon = K_{i+1} + 1 \wedge P_i[\epsilon := K_{i+1}])) \\
\vee(K_n + 1 < \epsilon \wedge P_n[\epsilon := \epsilon - 1])) \\
\equiv \\
wp(S, (1 \leq \epsilon \leq K_1 \wedge P_0[\epsilon := \epsilon - 1]) \\
\vee(\epsilon = K_1 + 1 \wedge P_0[\epsilon := K_1]) \\
\vee(\bigvee i | 1 \leq i < n : K_i + 1 < \epsilon \leq K_{i+1} \wedge P_i[\epsilon := \epsilon - 1]) \\
\vee(\bigvee i | 1 \leq i < n : \epsilon = K_{i+1} + 1 \wedge P_i[\epsilon := K_{i+1}]) \\
\vee(K_n + 1 < \epsilon \wedge P_n[\epsilon := \epsilon - 1])) \\
\equiv \quad \langle \text{ Lema 4 y 6 e hipótesis } \mathrm{support}(S) \rangle \\
(1 \leq \epsilon \leq K_1 \wedge wp(S, P_0[\epsilon := \epsilon - 1])) \\
\vee(\epsilon = K_1 + 1 \wedge wp(S, P_0[\epsilon := K_1])) \\
\vee(\bigvee i | 1 \leq i < n : K_i + 1 < \epsilon \leq K_{i+1} \wedge \\
\qquad wp(S, P_i[\epsilon := \epsilon - 1])) \\
\vee(\bigvee i | 1 \leq i < n : \epsilon = K_{i+1} + 1 \wedge wp(S, P_i[\epsilon := K_{i+1}])) \\
\vee(K_n + 1 < \epsilon \wedge wp(S, P_n[\epsilon := \epsilon - 1])) \\
\equiv \\
(1 \leq \epsilon \leq K_1 \wedge P_0) \vee (\epsilon = K_1 + 1 \wedge P_1) \\
\vee(\bigvee i | 1 \leq i < n : K_i + 1 < \epsilon \leq K_{i+1} \wedge P_i) \\
\vee(\bigvee i | 1 \leq i < n : \epsilon = K_{i+1} + 1 \wedge P_{i+1}) \\
\vee(K_n + 1 < \epsilon \wedge P_n) \\
\equiv \quad \left\langle \begin{array}{c} \text{Si } n > 1 \text{ se usa separación de rango,} \\ \text{si } n = 1 \text{ se usa idenpotencia del} \vee \text{con } K_2 = \infty \\ \text{y rango vacío} \end{array} \right\rangle \\
(1 \leq \epsilon \leq K_1 \wedge P_0) \\
\vee(\epsilon = K_1 + 1 \wedge P_1) \vee (K_1 + 1 < \epsilon \leq K_2 \wedge P_1) \\
\vee(\bigvee i | 2 \leq i < n : K_i + 1 < \epsilon \leq K_{i+1} \wedge P_i) \\
\vee(\bigvee i | 1 \leq i < n - 1 : \epsilon = K_{i+1} + 1 \wedge P_{i+1}) \\
\vee(\epsilon = K_n + 1 \wedge P_n) \vee (K_n + 1 < \epsilon \wedge P_n) \\
\equiv \\
(1 \leq \epsilon \leq K_1 \wedge P_0) \\
\vee((\epsilon = K_1 + 1 \vee K_1 + 1 < \epsilon \leq K_2) \wedge P_1) \\
\vee(\bigvee i | 2 \leq i < n : K_i + 1 < \epsilon \leq K_{i+1} \wedge P_i) \\
\vee(\bigvee i | 1 \leq i < n - 1 : \epsilon = K_{i+1} + 1 \wedge P_{i+1}) \\
\vee((\epsilon = K_n + 1 \vee K_n + 1 < \epsilon) \wedge P_n) \\
\equiv \\
(1 \leq \epsilon \leq K_1 \wedge P_0) \\
\vee(K_1 + 1 \leq \epsilon \leq K_2 \wedge P_1) \\
\vee(\bigvee i | 2 \leq i < n : K_i + 1 < \epsilon \leq K_{i+1} \wedge P_i) \\
\vee(\bigvee i | 2 \leq i < n : \epsilon = K_i + 1 \wedge P_i) \\
\vee(K_n + 1 \leq \epsilon \wedge P_n) \\
\equiv \\
(1 \leq \epsilon \leq K_1 \wedge P_0) \\
\vee(K_1 + 1 \leq \epsilon \leq K_2 \wedge P_1) \\
\vee(\bigvee i | 2 \leq i < n : (\epsilon = K_i + 1 \wedge P_i) \vee \\
\qquad (K_i + 1 < \epsilon \leq K_{i+1} \wedge P_i)) \\
\vee(K_n + 1 \leq \epsilon \wedge P_n) \\
\equiv \\
(1 \leq \epsilon \leq K_1 \wedge P_0) \\
\vee(K_1 < \epsilon \leq K_2 \wedge P_1) \\
\vee(\bigvee i | 2 \leq i < n : K_i < \epsilon \leq K_{i+1} \wedge P_i) \\
\vee(K_n < \epsilon \wedge P_n) \\
\equiv \\
((\cancel{false} \vee 1 \overset{\epsilon := 0}{\leq} \epsilon \leq K_1) \wedge P_0) \\
\vee(\bigvee i | 1 \leq i < n : K_i < \epsilon \leq K_{i+1} \wedge P_i) \\
\vee(K_n < \epsilon \wedge P_n) \\
\equiv \\
(0 \leq \epsilon \leq K_1 \wedge P_0) \vee \cdots \vee (K_n < \epsilon \wedge P_n) \\
\equiv \\
NBG
\end{array}
$$

■

III. Instrucción de Especificación

Según [10], dado un algoritmo en que se declaran identificadores de tipos $T_1, \cdots, T_n$ (en ese orden), se considera un estado de la ejecución, a un elemento en el conjunto $\{abort\} \cup \prod_{i=1}^{n} T_i$, en donde $abort$ es un elemento especial para indicar una ejecución abortada. Adicionalmente una instrucción (determinística o no) se considera como un conjunto de pares ordenados de estados de ejecución, los cuales representan entradas y salidas de una ejecución de la instrucción.

Con los conceptos del párrafo anterior se define lo siguiente:

Definición 2 (Instrucción de Especificación)**.** *Sea A un algoritmo en donde $\overline{x}$ y $\overline{y}$ son listas de identificadores de tipos $\overline{T}$ y $\overline{T'}$ respectivamente, tales que $\overline{x}, \overline{y}$ es la lista de todos los identificadores declarados en A, y aquellos identificadores que se declararon constantes, ocurren en $\overline{x}$. Si $\overline{Z}$ es una lista de variables, de tipos $\overline{T''}$, no declaradas en A y* Pdef, Qdef *son predicados que dependen de $\overline{x}, \overline{y}, \overline{Z}$, entonces la nomenclatura $\overline{y} : [\mathrm{Pdef}(\overline{x}, \overline{y}, \overline{Z}), \mathrm{Qdef}(\overline{x}, \overline{y}, \overline{Z})]$ se considera como una instrucción, que se define como el siguiente conjunto de pares ordenados de estados de ejecución:*

$$R' \cup (\mathrm{Dom}(R')^c \times \{abort\})$$

donde $\mathrm{Dom}(R')^c = ((\overline{T} \times \overline{T'}) \cup \{abort\}) \setminus \mathrm{Dom}(R')$ *y*

$$
\begin{array}{rl}
R' := & \bigcup_{\overline{Z} \in \overline{T''}} R_{\overline{Z}} \\
R_{\overline{Z}} := & \{\langle (\overline{x}, \overline{y}), (\overline{x'}, \overline{y'}) \rangle \in \mathrm{Dom}_{\overline{Z}} \times \mathrm{Rgo}_{\overline{Z}} \,|\overline{x} = \overline{x'}\} \\
\mathrm{Dom}_{\overline{Z}} := & \{(\overline{x}, \overline{y}) \in \overline{T} \times \overline{T'} |\, \mathrm{Pdef}(\overline{x}, \overline{y}, \overline{Z})\} \\
\mathrm{Rgo}_{\overline{Z}} := & \{(\overline{x}, \overline{y}) \in \overline{T} \times \overline{T'} |\, \mathrm{Qdef}(\overline{x}, \overline{y}, \overline{Z})\}
\end{array}
$$

En otras palabras la instrucción de especificación $\overline{y} : [\mathrm{Pdef}(\overline{x}, \overline{y}, \overline{Z}), \mathrm{Qdef}(\overline{x}, \overline{y}, \overline{Z})]$, se considera como una abstracción de código. El código que se abstrae corresponde a una instrucción, que se comporta de forma que, cada estado de ejecución inicial que satisface $\mathrm{Pdef}(\overline{x}, \overline{y}, \overline{Z})$ (para valores fijos de $\overline{Z}$), es transformado a un estado que satisface $\mathrm{Qdef}(\overline{x}, \overline{y}, \overline{Z})$ (para los mismos valores de $\overline{Z}$), pero con la restricción de que los valores de los identificadores $\overline{x}$ no cambian en la transformación.

Se puede calcular wp de una instrucción de especificación dado una postcondición de forma general, pero para este trabajo nos interesa solo un caso particular, que es cuando en los predicados no hay variables de especificación $\overline{Z}$. Para este caso se tiene el siguiente teorema de Caroll Morgan [7]:

Teorema 4. *Sea un algoritmo donde $\overline{x}, \overline{y}$ es la lista de identificadores declarados en donde aquellos declarados como constantes ocurren en la lista $\overline{x}$. Si* $\mathrm{Pdef}(\overline{x}, \overline{y})$, $\mathrm{Qdef}(\overline{x}, \overline{y})$ *y* $\mathrm{Post}(\overline{x}, \overline{y})$ *son predicados sin variables de especificación (sólo ocurren los identificadores declarados en el programa), tales que para todo $\overline{x}, \overline{y}$ en el que se satisface* $\mathrm{Pdef}(\overline{x}, \overline{y})$, *existe $\overline{y'}$ tal que* $\mathrm{Qdef}(\overline{x}, \overline{y'})$, *entonces, una precondición más débil para la instrucción de especificación $\overline{y} : [\mathrm{Pdef}(\overline{x}, \overline{y}), \mathrm{Qdef}(\overline{x}, \overline{y})]$ y* $\mathrm{Post}(\overline{x}, \overline{y})$ *es:*

$$\mathrm{Pdef}(\overline{x}, \overline{y}) \wedge (\forall \overline{y} |\, \mathrm{Qdef}(\overline{x}, \overline{y}) : \mathrm{Post}(\overline{x}, \overline{y}))$$

IV. Esquema de Algoritmos de Punto Fijo

Dado una familia de conjuntos $\{R_i\}_{i=a}^{n}$ (donde $n \geq 0$ y puede ser $n = \infty$) se define el esquema de algoritmos de punto fijo a la familia de algoritmos resultantes de refinar la instrucción de especificación del siguiente algoritmo:

```
|[
Var i, Prev, R : Int, Set, Set;

i, R := a, R_a;
do
  | Prev := R;
  | R : [a ≤ i < n ∧ Prev = R_i, R = R_{i+1}];
  | i := i + 1
while Prev ≠ R →
]|
```

Observación 1. *Las instrucciones:*

$$R : [a \leq i < n \wedge Prev = R_i, R = R_{i+1}]$$

y:

$$R : [a \leq i < n \wedge Prev = R_i, a \leq i < n \wedge R = R_{i+1}]$$

son las mismas ya que las variables a, i y n no son modificadas por dicha instrucción. Siguiendo la filosofía de que no deben existir estados en que las aserciones queden indeterminadas, se debería usar la segunda versión de la instrucción, pero al ser ambas iguales, se escogió la primera porque facilita los cálculos.

Usando los teoremas anteriores se calculará la precondición más débil del algoritmo anterior para la postcondición $\mathrm{Post} : a < i \leq n \wedge R = R_i = R_{i-1}$, para esto se descompone el do-while en una instrucción de iteración Do usual de la siguiente forma.

```
|[
Var i, Prev, R : Int, Set, Set;

i, R := a, R_a;
Prev := R;
R : [a ≤ i < n ∧ Prev = R_i, R = R_{i+1}];
i := i + 1;
do Prev ≠ R →
  | Prev := R;
  | R : [a ≤ i < n ∧ Prev = R_i, R = R_{i+1}];
  | i := i + 1
od
]|
```

De ahora en adelante se abreviará con S, a las tres instrucciones dentro del bloque del Do anterior.

V. Cálculo de $domBG$ y $\mathrm{support}(S)$

Como $\mathrm{domain}(B) \equiv \mathrm{domain}(Prev \neq R) \equiv$ true, entonces por el teorema 2 se tiene que:

$$domBG \equiv \text{true}.$$

Por otro lado para calcular $\mathrm{support}(S)$ se calcula $wp(S, \text{true})$ como a continuación.

$$wp\left(\begin{array}{c} Prev := R; \\ R : [a \leq i < n \wedge Prev = R_i, R = R_{i+1}]; \quad , \quad \text{true} \\ i := i + 1 \end{array}\right)$$

$\equiv$ ⟨definición de wp⟩

$$wp\left(\begin{array}{c} Prev := R; \\ R : \left[\begin{array}{c} a \leq i < n \\ \wedge \\ Prev = R_i \end{array}\right. , \left. R = R_{i+1} \right], wp(i := i + 1, \text{true}) \end{array}\right)$$

$\equiv$ ⟨definición de wp⟩

$$wp(Prev := R, wp(R : \left[\begin{array}{c} a \leq i < n \\ \wedge \\ Prev = R_i \end{array}\right. , \left. R = R_{i+1} \right], \text{true}))$$

$\equiv$ ⟨teorema 4⟩

$$wp(Prev := R, \begin{array}{c} a \leq i < n \wedge Prev = R_i \wedge \\ (\forall R | R = R_{i+1} : \text{true}) \end{array}^{true})$$

$\equiv$

$$a \leq i < n \wedge R = R_i$$

Con lo que:

$$\mathrm{support}(S) \equiv a \leq i < n \wedge R = R_i$$

VI. Cálculo de NBG

Para calcular NBG se estudian diferentes casos para diferentes valores de ϵ.

Para $\epsilon = 0$

$NBG \equiv Prev = R$ por definición

Para $\epsilon \geq 1$

Para calcular NBG se usará el Teorema 3, para esto se debe suponer como hipótesis $0 < \epsilon \leq K \wedge \mathrm{domain}(B) \wedge B \wedge \mathrm{support}(S)$. El valor de K se determinará más adelante. A continuación se hacen cálculos de $wp(S, .)$ para determinar una fórmula NBG para diferentes valores de ϵ fijos.

Para $\epsilon = 1$

$$wp\left(\begin{array}{c} Prev := R; \\ R : \left[\begin{array}{c} a \leq i < n \\ \wedge \\ Prev = R_i \end{array}\right. , \left. R = R_{i+1} \right]; \quad , \quad Prev = R \\ i := i + 1 \end{array}\right)$$

$\equiv$ ⟨definición de wp⟩

$$wp\left(\begin{array}{c} Prev := R; \\ R : \left[\begin{array}{c} a \leq i < n \\ \wedge \\ Prev = R_i \end{array}\right. , \left. R = R_{i+1} \right], wp(i := i + 1, \begin{array}{c} Prev \\ = \\ R \end{array}) \end{array}\right)$$

$\equiv$ ⟨definición de wp⟩

$$wp(Prev{:=}R, wp(R{:}\begin{bmatrix} a \le i < n \\ \wedge \\ Prev = R_i \end{bmatrix}, \begin{matrix} Prev \\ = \\ R \end{matrix}))$$
$\equiv$ ⟨teorema 4⟩
$$wp\left(Prev := R, \begin{matrix} a \le i < n \wedge Prev = R_i \wedge \\ (\forall R | R = R_{i+1} : Prev = R) \end{matrix}\right)$$
$\equiv$ ⟨regla de un punto⟩
$$wp(Prev := R, a \le i < n \wedge Prev = R_i \wedge Prev = R_{i+1})$$
$\equiv$ ⟨definición de wp⟩
$$a \le i < n \wedge R = R_i \wedge R = R_{i+1}$$
$\equiv$ ⟨hipótesis $\mathrm{support}(S)$⟩
$$a \le i < n \wedge R = R_i \wedge R_i = R_{i+1} \quad \text{true}$$
$\equiv$ ⟨$\epsilon = 1$⟩
$$R_{i+\epsilon-1} = R_{i+\epsilon}$$

Como n es una cota superior para los índices de la familia de conjuntos $\{R_i\}_{i=a}^{n}$, entonces $i + \epsilon \le n$, con lo que $\epsilon \le n - i$ (inecuación válida incluso si $n = \infty$), con lo que se deduce que el valor de K de la definición 1 es $n - i$ y por lo tanto hay que suponer en todo momento que $0 < \epsilon \le n - i$.

A continuación se demuestra que para todo $\epsilon > 1$, se cumple que $wp(S, (R_{i+\epsilon-1} = R_{i+\epsilon})[\epsilon := \epsilon - 1]) \equiv R_{i+\epsilon-1} = R_{i+\epsilon}$

Para $\epsilon > 1$

$$wp\left(\begin{matrix} Prev := R; \\ R : \begin{bmatrix} a \le i < n \\ \wedge \\ Prev = R_i \end{bmatrix}; \ , R_{i+\epsilon-2} = R_{i+\epsilon-1} \\ i := i + 1 \end{matrix}\right)$$
$\equiv$ ⟨definición de wp⟩
$$wp\left(\begin{matrix} Prev := R; \\ R : \begin{bmatrix} a \le i < n \\ \wedge \\ Prev = R_i \end{bmatrix}, wp(\begin{matrix} i & R_{i+\epsilon-2} \\ := & = \\ i+1 & R_{i+\epsilon-1} \end{matrix}) \end{matrix}\right)$$
$\equiv$ ⟨definición de wp⟩
$$wp(Prev := R; R : \begin{bmatrix} a \le i < n \\ \wedge \\ Prev = R_i \end{bmatrix}, \begin{matrix} R_{i+\epsilon-1} \\ = \\ R_{i+\epsilon} \end{matrix})$$
$\equiv$ ⟨Lema 2 y 3 e hipótesis $\mathrm{support}(S)$⟩
$$a \le i < n \wedge R = R_i \wedge R_{i+\epsilon-1} = R_{i+\epsilon} \quad \text{true}$$

Según el Teorema 3, tomando $n = 1$, $K_1 = K_n = 0$, P_0 y P_1 como $Prev = R$ y $R_{i+\epsilon-1} = R_{i+\epsilon}$ respectivamente, se tiene que:

$$NBG \equiv (\epsilon = 0 \wedge Prev = R) \vee (0 < \epsilon \wedge R_{i+\epsilon-1} = R_{i+\epsilon})$$

VII. Condición de Terminación de Algoritmos de Punto Fijo

Según la definición 1 se tiene que:

$$T_{k'} \equiv (\exists \epsilon | 0 \le \epsilon \le k' : (\epsilon = 0 \wedge Prev = R) \vee (0 < \epsilon \wedge R_{i+\epsilon-1} = R_{i+\epsilon}))$$

Lo cual es equivalente a:

$$Prev = R \vee (\exists \epsilon | 1 \le \epsilon \le k' : R_{i+\epsilon-1} = R_{i+\epsilon})$$

ya que por hipótesis $0 \le k'$.

VIII. Cálculo de $H_{k'}(Post)$

En esta sección será de utilidad demostrar el siguiente lema.

Lema 9. *Sea S las tres instrucciones del bloque interno del Do del algoritmo al final de la Sección IV, entonces:*

1. $wp(S, R = R_i) \equiv \mathrm{support}(S)$
2. *Suponiendo* $\mathrm{support}(S)$ *como hipótesis, se tiene que:*
   - $wp(S, Prev = R) \equiv R_i = R_{i+1}$
   - $wp(S, R_{i+\epsilon-2} = R_{i+\epsilon-1}) \equiv R_{i+\epsilon-1} = R_{i+\epsilon}$ *para cualquier* ϵ

Proof: La demostración de la parte 2 se encuentra en los cálculos de wp de la Sección VI, los cálculos hechos no dependían del valor de ϵ, sino solo de la hipótesis $\mathrm{support}(S)$. La parte 1 se demuestra a continuación.

$$wp\left(\begin{matrix} Prev := R; \\ R : \begin{bmatrix} a \le i < n \\ \wedge \\ Prev = R_i \end{bmatrix}; \quad , \quad R = R_i \\ i := i + 1 \end{matrix}\right)$$
$\equiv$
$$wp\left(\begin{matrix} Prev := R; \\ R : \begin{bmatrix} a \le i < n \\ \wedge \\ Prev = R_i \end{bmatrix}, wp(\begin{matrix} i & R \\ := & = \\ i+1 & R_i \end{matrix}) \end{matrix}\right)$$
$\equiv$
$$wp(Prev{:=}R, wp(R : \begin{bmatrix} a \le i < n \\ \wedge \\ Prev = R_i \end{bmatrix}, \begin{matrix} R \\ = \\ R_{i+1} \end{matrix}))$$
$\equiv$ ⟨teorema 4⟩
$$wp\left(Prev := R, \begin{matrix} a \le i < n \wedge Prev = R_i \wedge \\ (\forall R | R = R_{i+1} : R = R_{i+1}) \end{matrix}\right) \quad \text{true}$$
$\equiv$
$$a \le i < n \wedge R = R_i$$
$\equiv$
$$\mathrm{support}(S)$$
■

Observación 2. *La segunda parte del Lema anterior puede ser usada si dentro de una fórmula, se encuentra en conjunción* $\mathrm{support}(S)$ *junto con* $wp(S, Prev = R)$ *o* $wp(S, R_{i+\epsilon-2} = R_{i+\epsilon-1})$.

A continuación usando el teorema 1, se demostraran las obligaciones de prueba de iteración y de terminación tomando inv como:

$$R = R_i \wedge (Prev = R \Rightarrow a < i \le n \wedge R_i = R_{i-1}) \wedge (Prev \ne R \Rightarrow a \le i < n)$$

Para la obligación de prueba de terminación se tiene:

$$\mathrm{domain}(B) \wedge \neg B \wedge inv$$
$\equiv$
$$\mathrm{domain}(B) \wedge Prev = R \wedge R = R_i \wedge (Prev = R \Rightarrow a < i \le n \wedge R_i = R_{i-1}) \wedge (Prev \ne R \Rightarrow a \le i < n)$$
$\equiv$
$$\mathrm{domain}(B) \wedge Prev = R \wedge R = R_i \wedge a < i \le n \wedge R_i = R_{i-1}$$
$\equiv$
$$\mathrm{domain}(B) \wedge \neg B \wedge a < i \le n \wedge R = R_i = R_{i-1} \quad \text{Post}$$

Para la obligación de prueba de iteración se debe suponer $T_{k'} \wedge Prev \neq R$ con $1 \leq k' \leq n-i$ y hacer lo siguiente:

$wp(S, inv)$
$\equiv$
$wp\left(S, \begin{array}{c}(Prev = R \Rightarrow a < i \leq n \wedge R_i = R_{i-1}) \wedge \\ (Prev \neq R \Rightarrow a \leq i < n) \wedge R = R_i\end{array}\right)$
$\equiv \quad \langle Lema\ 1 \rangle$
$wp(S, Prev = R \Rightarrow a < i \leq n \wedge R_i = R_{i-1}) \wedge$
$wp(S, Prev \neq R \Rightarrow a \leq i < n) \wedge wp(S, R = R_i)$
$\equiv \quad \langle Lema\ 9 \rangle$
$wp(S, Prev = R \Rightarrow a < i \leq n \wedge R_i = R_{i-1}) \wedge$
$wp(S, i < a \vee i \geq n \Rightarrow Prev = R) \wedge \text{support}(S)$
$\equiv \quad \langle Lema\ 7\ y\ 1 \rangle$
$\left(wp(S, Prev = R) \Rightarrow \begin{array}{c} wp(S, a < i \leq n) \wedge \\ wp(S, R_i = R_{i-1})\end{array}\right) \wedge$
$(wp(S, i < a \vee i \geq n) \Rightarrow wp(S, Prev = R)) \wedge \text{support}(S)$
$\equiv \quad \langle Lema\ 9\ y\ \text{support}(S) \Rightarrow a < i+1 \leq n \rangle$
$(R_i = R_{i+1} \Rightarrow a \leq i+1 \leq n \wedge R_{i+1} = R_i) \wedge$ (true)
$(i+1 < a \vee i+1 \geq n \Rightarrow R_i = R_{i+1}) \wedge \text{support}(S)$ ($i+1 < a \vee i+1 \geq n$: false)
$\equiv \quad \left\langle \begin{array}{c} \text{support}(S) \Rightarrow (i+1 \geq n \equiv i = n-1)\ y \\ 1 \leq k' \leq n-i \wedge T_{k'} \wedge Prev \neq R \wedge i = n-1 \\ \Rightarrow \\ R_i = R_{i+1} \end{array}\right\rangle$
$(R_i = R_{i+1} \Rightarrow R_{i+1} = R_i) \wedge (i = n-1 \Rightarrow R_i = R_{i+1}) \wedge$ (true) (true)
$\text{support}(S)$
$\equiv$
$\text{support}(S) \wedge (false \Rightarrow a < i \leq n \wedge R_i = R_{i-1})$
$\equiv \quad \langle hipótesis\ Prev \neq R \rangle$
$a \leq i < n \wedge R = R_i \wedge (Prev = R \Rightarrow a < i \leq n \wedge R_i = R_{i-1})$
$\equiv \quad \langle hipótesis\ Prev \neq R \rangle$
$R = R_i \wedge (Prev = R \Rightarrow a < i \leq n \wedge R_i = R_{i-1})$
$\wedge\ (Prev \neq R \Rightarrow a \leq i < n)$
$\equiv$
inv

Observación 3. *Note que en el cálculo anterior, la sugerencia* $\text{support}(S) \Rightarrow (i+1 \geq n \equiv i = n-1)$ *es cierta si* $n = \infty$*, ya que en este caso* $i+1 \geq n \equiv false$ *y* $i = n-1 \equiv false$*. Por la misma razón es cierto que* $1 \leq k' \leq n-i \wedge T_{k'} \wedge Prev \neq R \wedge i = n-1 \Rightarrow R_i = R_{i+1}$ *cuando* $n = \infty$*.*

Con esto queda demostrado que $T_{k'} \wedge inv$ con $0 \leq k' \leq n-i$ es un $H_{k'}(\text{Post})$.

IX. Cálculo de la Precondición Más Débil del Ciclo

En esta sección se demuestra que $H_{n-i}(\text{Post})$ es la precondición más débil del ciclo. Como se indicó en la Sección VI, si $n = \infty$ entonces $K = \infty$ y por Corolario 1 la precondición más débil del ciclo es $T_\infty \wedge inv$. Pero como $n - i = \infty$, entonces $H_{n-i}(\text{Post}) \equiv T_\infty \wedge inv$.

Para el caso cuando $n \neq \infty$, se muestra que si se define inv' como $T_{n-i} \wedge inv$, entonces inv' satisface la obligación de prueba de terminación y la obligación de prueba de iteración para todo k' sin cota superior K. Haciendo esto, según el Teorema 1, se tiene que $H_{k'}(\text{Post}) \equiv T_{k'} \wedge inv'$ para todo k', lo cual a su vez es equivalente a $T_{k'} \wedge T_{n-i} \wedge inv$. Como $T_{n-i} \Rightarrow T_{k'}$ para todo $k' \geq n-i$, entonces $H_{n-i+1}(\text{Post}) \equiv H_{n-i}(\text{Post})$ y según el Corolario 1, $H_{n-i}(\text{Post})$ sería la precondición más débil del ciclo.

La obligación de prueba de terminación para inv' se demuestra de la siguiente forma:

$\text{domain}(B) \wedge \neg B \wedge inv'$
$\equiv$
$\text{domain}(B) \wedge \neg B \wedge T_{n-i} \wedge inv$
$\equiv \quad \langle \text{absorción} \rangle$
$\text{domain}(B) \wedge \neg B \wedge inv$
$\equiv \quad \langle \text{obligación de prueba de terminación de } inv \rangle$
$\text{domain}(B) \wedge \neg B \wedge \text{Post}$

Se hace notar que:

$$T_{n-i} \equiv Prev = R \vee (\exists \epsilon | 1 \leq \epsilon \leq n-i : R_{i+\epsilon-1} = R_{i+\epsilon})$$

y a su vez, esto último es equivalente a (incluso si $n = \infty$):

$$Prev = R \vee (\exists \epsilon | : i < \epsilon \leq n \wedge R_{\epsilon-1} = R_\epsilon)$$

Con esto último se demuestra la obligación de prueba de iteración para inv' asumiendo $T_{k'} \wedge Prev \neq R$ para $k' \geq 1$ y haciendo lo siguiente:

$wp(S, inv')$
$\equiv$
$wp(S, T_{n-i}(\text{Post}) \wedge inv)$
$\equiv \quad \langle \text{Lema } 1 \rangle$
$wp(S, T_{n-i}(\text{Post})) \wedge wp(S, inv)$
$\equiv \quad \langle \text{Lema } 4,\ 8 \text{ y } 6 \rangle$
$(wp(S, Prev = R)$
$\vee (\exists \epsilon | : wp(S, i < \epsilon \leq n) \wedge R_{\epsilon-1} = R_\epsilon)) \wedge wp(S, inv)$
$\equiv \quad \langle \text{Lema } 5 \rangle$
$(wp(S, Prev = R)$
$\vee (\exists \epsilon | : wp(S, i < \epsilon \leq n) \wedge R_{\epsilon-1} = R_\epsilon)) \wedge wp(S, inv) \wedge$
$\text{support}(S)$
$\equiv \quad \langle \text{Lema } 9,\ 2\ y\ 3 \rangle$
$(R_i = R_{i+1}$
$\vee (\exists \epsilon | i+1 < \epsilon \leq n : R_{\epsilon-1} = R_\epsilon)) \wedge wp(S, inv) \wedge$
$\text{support}(S)$
$\equiv \quad \langle \text{Lema 5 y separación de rango} \rangle$
$(\exists \epsilon | i < \epsilon \leq n : R_{\epsilon-1} = R_\epsilon) \wedge wp(S, inv)$
$\equiv \quad \langle \text{hipótesis } Prev \neq R \rangle$
$(Prev = R \vee (\exists \epsilon | i < \epsilon \leq n : R_{\epsilon-1} = R_\epsilon)) \wedge wp(S, inv)$ (T_{n-i})
$\equiv \quad \langle \text{obligación de prueba de iteración de } inv \rangle$
$T_{n-i} \wedge inv$
$\equiv$
inv'

X. Cálculo de la Precondición Más Débil del Algoritmo

Según lo demostrado hasta el momento el algoritmo planteado tiene las siguientes aserciones.

$|[$

```
Var i, Prev, R : Int, Set, Set;

i, R := a, Ra;
{wp(S, T_{n-i} ∧ inv)}
Prev := R;
R : [a ≤ i < n ∧ Prev = Ri, R = Ri+1];
i := i + 1;
{H_{n-i}(Post) : T_{n-i} ∧ inv}
do Prev ≠ R →
   Prev := R;
   R : [a ≤ i < n ∧ Prev = Ri, R = Ri+1];
   i := i + 1
od
{Post : a < i ≤ n ∧ R = R_i = R_{i-1}}
]|
```

Por lo tanto el próximo paso consiste en calcular $wp(S, T_{n-i} \wedge inv)$

$$
\begin{aligned}
& wp(S, T_{n-i} \wedge inv) \\
\equiv & \quad \left\langle wp(S, inv') \equiv \begin{array}{c} (\exists \epsilon | i < \epsilon \leq n : R_{\epsilon-1} = R_\epsilon) \\ \wedge \\ wp(S, inv) \end{array} \right\rangle \\
& (\exists \epsilon | i < \epsilon \leq n : R_{\epsilon-1} = R_\epsilon) \wedge wp(S, inv) \\
\equiv & \quad \left\langle \begin{array}{c} 5 \text{ primeros pasos del cálculo de } wp(S, inv) \\ \text{de la Sección } VIII \end{array} \right\rangle \\
& (\exists \epsilon | i < \epsilon \leq n : R_{\epsilon-1} = R_\epsilon) \wedge \\
& (i + 1 \geq n \Rightarrow R_i = R_{i+1}) \wedge \text{support}(S) \\
\equiv & \quad \left\langle \begin{array}{c} \text{support}(S) \Rightarrow (i + 1 \geq n \equiv i = n - 1) \\ i = n - 1 \wedge (\exists \epsilon | i < \epsilon \leq n : R_{\epsilon-1} = R_\epsilon) \\ \Rightarrow \\ R_i = R_{i+1} \end{array} \right\rangle \\
& (\exists \epsilon | i < \epsilon \leq n : R_{\epsilon-1} = R_\epsilon) \wedge \\
& (i = n - 1 \Rightarrow \overset{\text{true}}{\cancel{R_i = R_{i+1}}}) \wedge \text{support}(S) \\
\equiv & \\
& (\exists \epsilon | i \leq \epsilon \leq n - 1 : R_\epsilon = R_{\epsilon+1}) \wedge a \leq i \wedge \overset{\text{true}}{\cancel{i < n}} \wedge R = R_i
\end{aligned}
$$

De esta forma se tiene que el predicado anterior, es correcto colocarlo como una aserción en el algoritmo, de la siguiente manera:

```
|[
Var i, Prev, R : Int, Set, Set;

i, R := a, Ra;
{(∃ε | i ≤ ε ≤ n − 1 : R_ε = R_{ε+1}) ∧ a ≤ i ∧ R = R_i}
Prev := R;
R : [a ≤ i < n ∧ Prev = Ri, R = Ri+1];
i := i + 1;
do Prev ≠ R →
   Prev := R;
   R : [a ≤ i < n ∧ Prev = Ri, R = Ri+1];
   i := i + 1
od
{Post : a < i ≤ n ∧ R = R_i = R_{i-1}}
]|
```

Que es equivalente a escribir:

```
|[
Var i, Prev, R : Int, Set, Set;

i, R := a, Ra;
{(∃ε | i ≤ ε ≤ n − 1 : R_ε = R_{ε+1}) ∧ a ≤ i ∧ R = R_i}
do
   Prev := R;
   R : [a ≤ i < n ∧ Prev = Ri, R = Ri+1];
   i := i + 1
while Prev ≠ R →
{Post : a < i ≤ n ∧ R = R_i = R_{i-1}}
]|
```

Calculando la precondición más débil se tiene:

```
|[
Var i, Prev, R : Int, Set, Set;

{Pre : (∃ε | a ≤ ε ≤ n − 1 : R_ε = R_{ε+1}) ∧ a ≤ a ∧ R_a = R_a}   (a ≤ a → true; R_a = R_a → true)
i, R := a, Ra;
do
   Prev := R;
   R : [a ≤ i < n ∧ Prev = Ri, R = Ri+1];
   i := i + 1
while Prev ≠ R →
{Post : a < i ≤ n ∧ R = R_i = R_{i-1}}
]|
```

Con esto se concluye que la precondición más débil del algoritmo y Post es $(\exists \epsilon | a \leq \epsilon \leq n - 1 : R_\epsilon = R_{\epsilon+1})$.

XI. Ejemplos

La tripleta de Hoare encontrada al final de la sección anterior, induce un marco de trabajo para conseguir algoritmos de punto fijo concretos. Para construir un algoritmo de punto fijo, basta con definir una familia de conjuntos $\{R_i\}_{i=a}^{n}$, de tal forma que la precondición Pre sea una tautología, y refinar la instrucción de especificación en un trozo de código St que satisfaga $\{a \leq i < n \wedge Prev = R_i\} St \{R = R_{i-1}\}$.

En esta sección se construye el algoritmo de búsqueda de símbolos anulables de una gramática libre de contexto. Una gramática G libre de contexto es una tupla $G = (V, \Sigma, P, S)$ donde V es un conjunto de símbolos no terminales, Σ es un conjunto de símbolos terminales, $P \subseteq V \times (\Sigma \cup V)^*$ y $S \in V$ (dado un conjunto de símbolos Γ, se denota Γ^* y Γ^+ a los conjuntos de las frases construidas con los símbolos de Γ de longitud ≥ 0 y longitud > 0 respectivamente). Los pares $\langle A, w \rangle \in P$ se le llaman producciones, y se suelen denotar como $A \to w$.

Definición 3. *Dado una gramática libre de contexto G, se define la relación de derivación $\Rightarrow \subseteq (\Sigma \cup V)^+ \times (\Sigma \cup V)^*$ de forma que $t \Rightarrow w$ si y sólo si existen $A \in V$ y $t_1, t_2, w_1 \in (\Sigma \cup V)^*$ tales que $t = t_1 A t_2$, $w = t_1 w_1 t_2$ y $A \to w_1 \in P$. Por otro lado se define la relación de derivación en paralelo $\Rightarrow_p \subseteq (\Sigma \cup V)^+ \times (\Sigma \cup V)^*$ de forma que $t \Rightarrow_p w$ si y sólo si t es de la forma $t_0 A_1 t_1 A_2 \ldots t_{k-1} A_k t_k$ con $A_i \in V$, $t_i \in (\Sigma \cup V)^*$ y w es de la forma $t_1 w_1 t_2 w_2 \ldots t_k w_k$ en donde $A_i \Rightarrow w_i$.*

Definición 4. *Dada una relación binaria R, la notación $tR^* w$ y $tR^{\leq i} w$ significan $(\exists m | m \in \mathbb{N} : tR^m w)$ y $(\exists m | m \leq i :$*

$tR^m w$) *respectivamente, en donde* R^m *es la composición de* R *consigo misma* m *veces.*

El conjunto de los símbolos anulables de una gramática es $NULL := \{A \in V | A \Rightarrow^* \lambda\}$, donde λ es la frase vacía. Para diseñar un algoritmo de punto fijo primeramente se define una familia de conjuntos de la siguiente forma:

Definición 5. *Dada una gramática libre de contexto* $G = (\Sigma, V, P, S)$*, se define para* $i \geq 1$

$$NULL_i := \{A \in V | A \Rightarrow_p^{\leq i} \lambda\}.$$

El siguiente teorema justifica el uso de un algoritmo de punto fijo para conseguir el conjunto de símbolos anulables.

Teorema 5. *Si existe* $\epsilon \geq 1$ *tal que* $NULL_\epsilon = NULL_{\epsilon+1}$*, entonces* $NULL = NULL_\epsilon$

Por esta razón para calcular $NULL$ basta con encontrar el punto fijo de $\{NULL_i\}_{i=1}^{\infty}$.

Para instanciar la familia de algoritmos en un algoritmo que compute el punto fijo de la familia $\{NULL_i\}_{i=1}$, se debe demostrar que Pre es una tautología.

Teorema 6.

$$(\exists \epsilon | 1 \leq \epsilon : NULL_\epsilon = NULL_{\epsilon+1})$$

Proof: Por la definición de $\Rightarrow^{\leq i}$ se tiene que si $A \Rightarrow^{\leq i} \lambda$ entonces $A \Rightarrow^{\leq i+1} \lambda$, por lo tanto $NULL_1 \subseteq NULL_2 \subseteq \cdots \subseteq NULL_i$. Si el enunciado del teorema no fuera cierto, entonces se tendría para todo i que:

$$NULL_1 \subset NULL_2 \subset \cdots \subset NULL_i,$$

con lo que aumentando i se pudiera conseguir un $NULL_i$ de cardinalidad arbitrariamente grande, pero esto es imposible ya que $NULL_i \subseteq V$ para todo i y por lo tanto $|NULL_i| \leq |V|$, con lo que $|NULL_i|$ está acotado por ser V finito. ■

A continuación se refinará la instrucción de especificación:

$$R : [1 \leq i \wedge Prev = NULL_i, R = NULL_{i+1}] \qquad (*)$$

del esquema de algoritmos, en una instrucción concreta.

Teorema 7. *La siguiente definición recursiva es equivalente a la defición 5:*

- $NULL_1 := \{A \in V | A \to \lambda \in P\}$
- $NULL_{i+1} = \{A \in V | A \to w \in P \wedge w \in NULL_i^*\}$
  $\cup\, NULL_i$ *para* $i \geq 1$

Proof: Para demostrar el primer item se hace lo siguiente:

$NULL_1 = \{A \in V | A \Rightarrow_p^{\leq 1} \lambda\} =$
$\{A \in V | (\exists m | m \leq 1 : A \Rightarrow_p^m \lambda)\} = \{A \in V | A \to \lambda\}$ en donde la última igualdad es cierta debido a que es siempre falso que $A \to^0 \lambda$ si $A \in V$.

Para demostrar el segundo item se usa doble contención. La demostración en el sentido $\subseteq$ es la siguiente:

Si $A \in NULL_{i+1}$ entonces por definición $A \Rightarrow_p^{\leq i+1} \lambda$ y por lo tanto existe $m \leq i+1$ tal que $A \Rightarrow_p^{\leq m} \lambda$. Si $m \leq i$ entonces $A \in NULL_i$ con lo que se cumple la contención. Por otro lado si $m = i+1$ entonces $A \Rightarrow_p w \Rightarrow_p^i \lambda$ donde w debe ser de la forma $A_1 \ldots A_n$ y $A_k \Rightarrow_p^{\leq i} \lambda$, con lo que $A_k \in NULL_i$ y por lo tanto $w \in NULL_i^*$, de modo que $A \in \{A \in V | A \to w \in P \wedge w \in NULL_i^*\}$.

Para demostrar el sentido $\supseteq$ se hace lo siguiente:

Si $A \in \{A \in V | A \to w \in P \wedge w \in NULL_i^*\}$, entonces $A \Rightarrow_p w$ donde w es de la forma $A_1 \ldots A_n$ con $A_k \in V$, en donde $(\exists m'_k | m'_k \leq i : A_k \Rightarrow_p^{m'_k} \lambda)$. Si se toma m' como el máximo de los m'_k, se tiene que $(\exists m' | m' \leq i : w = A_1 \ldots A_k \Rightarrow_p^{m'} \lambda)$ y por lo tanto $A \Rightarrow_p w \Rightarrow_p^{\leq i} \lambda$, con lo cual $A \in NULL_{i+1}$. Por otro lado como en la demostración de 6 se determinó que $NULL_i \subseteq NULL_{i+1}$, entonces si $A \in NULL_i$, la demostración es inmediata. ■

Denotando $\mathtt{R} := \{\mathtt{A} \in \mathtt{V} | \mathtt{A} \to \mathtt{w} \in \mathtt{P} \wedge \mathtt{w} \in \mathtt{Prev}^*\} \cup \mathtt{Prev}$; como St, se tiene que:

$1 \leq i \wedge Prev = NULL_i$
$\Rightarrow \quad \langle$Teorema 7$\rangle$
$NULL_{i+1} = \{A \in V | A \to w \in P \wedge w \in Prev^*\} \cup Prev$
$\equiv$
$wp(St, 1 \leq i \wedge R = NULL_{i+1})$

de modo que la tripleta:

$\{1 \leq i \wedge Prev = NULL_i\}$
$\quad \mathtt{R} := \{\mathtt{A} \in \mathtt{V} | \mathtt{A} \to \mathtt{w} \in \mathtt{P} \wedge \mathtt{w} \in \mathtt{Prev}^*\} \cup \mathtt{Prev};$
$\{1 \leq i \wedge R = NULL_{i+1}\},$

es cierta y St es una realización de la especificación $(*)$.
Por último para demostrar el teorema 5 es necesario demostrar los siguiente lemas.

Lema 10. *Si* $NULL_\epsilon = NULL_{\epsilon+1}$ *con* $\epsilon \geq 1$*, entonces* $NULL_\epsilon = NULL_{\epsilon+n}$ *para todo* $n \geq 1$.

Proof: Por inducción natural sobre n, en donde el caso base $NULL_\epsilon = NULL_{\epsilon+1}$ está dado por hipótesis. Asumiendo $NULL_\epsilon = NULL_{\epsilon+n}$ como hipótesis inductiva (H.I.) se tiene que $NULL_{\epsilon+n+1} = NULL_{\epsilon+n} \cup \{A \in V | A \to w \in P \wedge w \in NULL_{\epsilon+n}^*\} \overset{H.I.}{=} NULL_\epsilon \cup \{A \in V | A \to w \in P \wedge w \in NULL_\epsilon^*\} = NULL_{\epsilon+1} = NULL_\epsilon$ ■

Lema 11. *Si para todo* $m' > \epsilon$ *se tiene que* $NULL_\epsilon = NULL_{m'}$ *entonces:*

$$(\exists m' | : A \Rightarrow_p^{m'} \lambda) \equiv A \Rightarrow_p^{\leq \epsilon} \lambda$$

Proof: Por doble implicación.

Si $A \Rightarrow_p^{\leq \epsilon} \lambda$ entonces por definición:
$(\exists m' | m' \leq \epsilon : A \Rightarrow_p^{m'} \lambda)$ y por lo tanto $(\exists m' | : A \Rightarrow_p^{m'} \lambda)$.

Por otro lado si existe m' tal que $A \Rightarrow_p^{m'} \lambda$ y $m' \leq \epsilon$, entonces por definición $A \Rightarrow_p^{\leq \epsilon} \lambda$, pero si $m' > \epsilon$ entonces se tendría que $A \in NULL_{m'} = NULL_\epsilon$ y por lo tanto $A \Rightarrow_p^{\leq \epsilon} \lambda$. ■

Ahora se puede demostrar el teorema 5.

Proof: Sea $A \in V$ entonces:

$$A \in NULL \overset{\text{definición de } NULL}{\equiv} A \Rightarrow^* \lambda$$
$$\overset{\text{definición de } \Rightarrow^*}{\equiv} (\exists m | : A \Rightarrow^m \lambda) \equiv (\exists m' | : A \Rightarrow_p^{m'} \lambda)$$
$$\overset{\text{Lema 11}}{\equiv} A \Rightarrow_p^{\leq \epsilon} \lambda \overset{\text{definición de } NULL_\epsilon}{\equiv} A \in NULL_\epsilon \quad \blacksquare$$

Con esto se concluye que el algoritmo de punto fijo para calcular símbolos anulables es:

$||[\texttt{Var i}, \texttt{Prev}, \texttt{R} : \texttt{Int}, \texttt{Set}, \texttt{Set};$

$\{Pre : true\}$
$\texttt{i}, \texttt{R} := 1, \{\texttt{A} \in \texttt{V} | \texttt{A} \rightarrow \lambda \in \texttt{P}\};$
do
  $\texttt{Prev} := \texttt{R};$
  $\texttt{R} := \{\texttt{A} \in \texttt{V} | \texttt{A} \rightarrow \texttt{w} \in \texttt{P} \wedge \texttt{w} \in \texttt{Prev}^*\} \cup \texttt{Prev};$
  $\texttt{i} := \texttt{i} + 1$
while $\texttt{Prev} \neq \texttt{R} \rightarrow$
$\{\text{Post} : 1 < i \wedge R{=}NULL_i{=}NULL_{i-1} \overset{\text{Teo 5}}{=} NULL\}$
$]||$

XII. Conclusiones

La técnica que se usa en este trabajo es efectiva para encontrar la precondición más débil de un algoritmo. Por otro lado en [1][6], existen ejemplos en que esta técnica, además de ser efectiva, también es más eficiente para corregir un algoritmo, que si se usara la lógica de Hoare. Sin embargo, en vista de la cantidad de cálculos que se tuvieron que hacer para corregir los algoritmos de punto fijo, se puede conjeturar, que era más eficiente hacer la corrección de la familia de algoritmos planteados, usando la lógica de Hoare.

Aunque para el tipo de algoritmos visto en este trabajo, la técnica propuesta resulte más complicada que las clásicas, aún existe una ventaja, la de poder demostrar que una aserción es la precondición más débil de todo el algoritmo. Esto es algo que no se puede garantizar con las reglas de Hoare.

Adicionalmente si se tiene la precondición más débil WP, se puede validar si cualquier otro predicado Pre es válido para ser usado como una precondición del algoritmo, verificando $Pre \Rightarrow WP$. De esta forma otra justificación de los cálculos realizados, es que la familia de algoritmos definida, junto con la familia de preconciones más débiles calculada, puede usarse como framework para construir algoritmos de punto fijo corretos. Esto es porque para corregir el algoritmo Alg, con la precondición Pre (y la postcondición $Post$ usada en el trabajo), basta con verificar que Alg es una instancia de la familia y que Pre implica la precondición más débil que se calculó aquí para Alg.

Referencias

[1] F. Flaviani, *Calculation of Invariants Assertions*, Electronic Notes in Theoretical Computer Science, vol. 339, pp. 63–83, 2018.

[2] F. Flaviani, *Propiedades Algebraicas y Decidibilidad del Transformador de Predicados wp Sobre la Teoría de Conjuntos*, Revista Venezolana de Computación (ReVeCom), vol. 4, no. 2, pp. 46–58, Diciembre 2017.

[3] E. W. Dijkstra, *Guarded Commands, Nondeterminacy and Formal Derivation of Programs*, Commun. ACM, vol. 18, pp. 453–457, August 1975.

[4] C. A. R. Hoare, *An Axiomatic Basis for Computer Programming*, Commun. ACM, vol. 12, pp. 576–580, October 1969.

[5] D. Gries, *The Science of Programming*, Springer New York, 1981.

[6] F. Flaviani, *Cálculo de Precondiciones más Débiles*, Revista Venezolana de Computación (ReVeCom), vol. 3, no. 2, pp. 68–80, Diciembre 2016.

[7] C. Morgan, *The Specification Statement*, ACM Trans. Program. Lang. Syst., vol. 10, pp. 403–419, July 1988.

[8] M. Todorova and D. A. Orozova, *The Predicate Transformer and its Application in Introduction to Programming Courses*, Burgas Free University Yearbook, vol. 32, no. 1, pp. 194–207, 2015.

[9] G. O'Regan, *Giants of Computing: A Compendium of Select, Pivotal Pioneers*, Springer London, 2013.

[10] F. Flaviani, *Inference of the Definition of the Predicate Transformer wp with Occurrences of the Predicate domain Based on Denotational Semantics of GCL on ZF Set Theory*, in proceedings of the XLII Latin American Computing Conference, Sao Paulo, Brazil, Octubre 2018.

[11] F. Flaviani, *Modelo Relacional de la Teoría Axiomática del Lenguaje GCL de Dijkstra*, en las memorias de la Conferencia Nacional de Computación, Informática y Sistemas (CoNCISa 2015), Valencia, Venezuela, Noviembre 2015, pp. 153-164.

[12] L. Jilani, O. Mraihi, A. Louhichi, W. Ghardallou, K. Bsaies, and A. Mili, *Invariant Functions and Invariant Relations: An Alternative to Invariant Assertions*, Journal of Symbolic Computation, vol. 48, pp. 1–36, January 2013.

[13] O. Mraihi, W. Ghardallou, A. Louhichi, L. L. Jilani, K. Bsaies, and A. Mili, *Computing Preconditions and Postconditions of While Loops*, in proceedings of the 8th International Colloquium on Theoretical Aspects of Computing, Johannesburg, South Africa, pp. 173–193, Springer, September 2011.

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 6, No. 2, pp. 21-29, Diciembre 2019
Fecha recepción: 14/12/2019, Fecha aceptación: 10/05/2020

Extendiendo ZoneMinder para su Uso Masivo con una Refactorización de las Interfaces de Administración y la Integración de un Módulo de Reconocimiento Facial

Ilich Rondon[1], Kenny Jimenez[1], Dedaniel Urribarri[1], Eric Gamess[2]
ilichrondon@gmail.com, kenny.jimenez05@gmail.com, dedanielu@gmail.com, egamess@jsu.edu

[1] Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela
[2] MCIS Department, Jacksonville State University, Jacksonville, AL, USA

Resumen: A través de los años, los requerimientos de monitoreo y control de sitios de importancia en empresas, instituciones, e inclusive hogares, se han incrementado. La solución inicial fue la implementación de CCTV (Circuitos Cerrados de TeleVisión) basados en cámaras analógicas y restringida a usuarios pudientes. Además de los altos costos, esta solución carecía de sensores de movimientos. Frente a estas limitaciones, surgió una alternativa conocida como video vigilancia IP que usa la red de datos preexistente. Debido al auge de la video vigilancia IP, se hace necesario un software libre, con capacidad de administración centralizada para cámaras IP, con pocos requerimientos de cómputo, y una interfaz sencilla y amigable que permita visualizar las proyecciones y configurar de forma remota los parámetros de monitoreo. En este trabajo, se desarrolló un NVMS (Network Video Management System) basado en software abierto, con reconocimiento facial, que pueda ser usado por usuarios con pocos conocimientos en el área, como una solución de monitoreo de espacios pequeños o medianos. Para validar la poca carga del sistema propuesto, se hicieron pruebas de validaciones en varios escenarios.

Palabras Clave: Video Vigilancia; NVMSs; ZoneMinder; Cámaras IP; Reconocimiento Facial; Código Abierto; Administración Centralizada.

Abstract: Over the years, the requirements to monitor and control important sites in companies, institutions, and even homes, have increased. The initial solution was the implementation of CCTV (Closed-Circuit TeleVision) based on analog cameras and restricted to wealthy users. In addition to the high costs, this solution lacked motion sensors. Given these limitations, an alternative emerged which is known as "IP video-surveillance" that uses the pre-existing data network. Due to the increase in IP video-surveillance, a free software is needed, with centralized administration capacity for IP cameras, with low computational requirements, and a simple and friendly interface that allows viewing projections and remotely configuring monitoring parameters. In this work, a NVMS (Network Video Management System) based on free software was developed, with facial recognition, that can be used by users with little knowledge in the area, as a monitoring solution for small or medium spaces. To validate the low load of the proposed system, validation tests were performed in various scenarios.

Keywords: Video-Surveillance; NVMSs; ZoneMinder; IP Cameras; Facial Recognition; Open Source Code; Centralized Management.

I. Introducción

La video vigilancia [1-5] se originó como una solución a la necesidad de mantener constantemente vigilado un lugar y se popularizó con los primeros Circuitos Cerrados de Televisión (CCTV) [2] con cámaras analógicas. En sus inicios, la video vigilancia era destinada únicamente a ambientes empresariales y gubernamentales. Sin embargo, con el creciente desarrollo tecnológico de las últimas décadas, la significativa bajada de los precios de las cámaras IP y la facilidad de crear aplicaciones de software, el uso de la video vigilancia se ha extrapolado a cualquier lugar que necesite una solución eficiente de seguridad, inclusive lugares como el hogar.

Con el pasar de los años, la video vigilancia IP ha ido involucrando y agregando un conjunto de herramientas y tecnologías que han mejorado el control de seguridad que se puede tener de un lugar a través de las cámaras IP. Tecnologías tales como la detección de movimiento por medio de sensores, o incluso la agregación de algoritmos para el reconocimiento facial han proporcionado a los sistemas de administración de

cámaras IP grandes avances en materia de monitoreo y seguridad.

Debido a la gran demanda que existe actualmente sobre los softwares de administración de cámaras IP en el ámbito de la seguridad, y la capacidad que estos poseen para poder aplicar avanzadas tecnologías como lo es el reconocimiento facial con el análisis de las imágenes y videos recolectados, es necesario que estos posean interfaces de administración que sean fáciles e intuitivas de utilizar. El objetivo final es que todos los entes interesados en aplicar este tipo de sistemas de seguridad se vean respaldados al momento de implementarlos, debido a que no van a necesitar de personal especializado, y se van a beneficiar de las importantes funcionalidades de monitoreo y seguridad que estos ofrecen en el área de la video vigilancia.

En este trabajo, se implementó un sistema de video vigilancia basado en softwares de código abierto como lo son ZoneMinder [6][7] y el framework de Python llamado Django [8-10]. La idea principal es el uso de ZoneMinder como un sistema para configurar y explotar cámaras IP, con una refactorización de sus interfaces de administración para que la gestión del sistema y de los sensores sea al alcance de todos, además de la integración de un módulo de reconocimiento facial. Para validar los pocos recursos requeridos por el sistema desarrollo y su adecuado rendimiento para entidades pequeñas o medianas, se hicieron pruebas de desempeño para las cuales se reportan (1) el uso de la memoria RAM, (2) la carga promedia, (3) el uso del CPU y (4) el tiempo de respuesta, en diferentes escenarios de pruebas. Nuestras pruebas indican que el sistema propuesto en este trabajo es adecuado para dichas entidades.

El resto del presente documento está organizado como especificado a continuación. Los trabajos relacionados son revisados en la Sección II. La Sección III presenta la arquitectura y el diseño de la solución. Se discuten las pruebas de estrés y el análisis de los resultados obtenidos en la Sección IV. Finalmente, se presenta un balance del trabajo en la Sección V a través de las conclusiones y se dan direcciones para trabajos futuros en la Sección VI.

II. Trabajos Relacionados

Existen muchos proyectos comerciales para la televigilancia basada en IP, como por ejemplo Xeoma [11] de Felenasoft, Kerberos [12] de Verstraeten, XProtect [13] de Milestone Systems, EyeLine [14] de NCH Software, ContaCam [15] de Contaware.com, iVideon [16] de Mobile Video Solutions Inc., Blue Iris [17] de Blue Iris Software, Evo S [18] de Luxriot, webcamXP [19] de Moonware Studios, SmartViewer [20] de Hanwha Techwin Europe, y Camcloud VMS [21] de Camcloud.

Xeoma [11] es un software comercial de video vigilancia que permite la administración de decenas de cámaras desde un único servidor. El programa es multiplataforma y funciona en Windows, Linux, MacOS, y dispositivos Android e iOS, con acceso remoto completo y vista desde cualquier dispositivo móvil.

Xeoma ofrece auto-detección y soporte para casi cualquier cámara IP, como las basadas en ONVIF [22] (Open Network Video Interface Forum, un foro abierto industrial que provee y promueve interfaces estándares para una interoperabilidad efectiva de productos de seguridad basados en IP), y las webcams USB (hasta 99,9% de las cámaras del mercado son soportadas). Con su ayuda, en tan sólo unos segundos, un computador y una cámara se convierten en un sistema de vigilancia. También posee un detector de movimiento con funciones avanzadas de falsas alarmas, evitando así la activación de la lógica cuando no es necesario. Igualmente, incluye notificaciones programadas (SMS, correo electrónico, sonido de alarma, entre otras). A diferencia de otros productos en el mercado, Xeoma también ofrece el monitoreo del audio obtenido a través de las cámaras.

Kerberos [12] es un software de video vigilancia que tiene una versión de código abierto (Kerberos.io) y una versión comercial (Kerberos Enterprise). La versión libre de costo es más limitada, por ejemplo, funciona bien cuando el número de cámaras es bajo. La versión comercial, Kerberos Enterprise, escala bien con docenas o centenas de cámaras y es de alta disponibilidad. Kerberos Enterprise está basada en Kubernetes [23][24]. En el caso de Kerberos.io, existen varias posibilidades como un Docker o la instalación en SBCs (Single Board Computer) como Raspberry Pi para un mejor soporte de las aplicaciones IoT (Internet de las Cosas).

Kerberos.io está provisto de una interfaz limpia y ordenada, y tiene soporte para una gran cantidad de cámaras. Puede usar cámaras USB, IP o Raspberry Pi (v1.3 y v2.1). Cuando utiliza cámaras Raspberry Pi, se beneficia de su codificación de hardware que permite grabar videos a 30 FPS en un Raspbery Pi Zero. Cuando se desea utilizar una cámara IP, se recomienda usar la conexión RTSP [25-27] (Real Time Streaming Protocol), si está disponible. Se debe tener en cuenta que es posible que las cámaras IP económicas no funcionen correctamente. En forma general, las cámaras USB que necesitan controladores especiales no funcionan.

Ciertos proyectos son totalmente basados en soluciones orientadas a la nube, como Camcloud VMS [21]. Con Camcloud VMS, lo único que el cliente necesita comprar al nivel de hardware son las cámaras IP. El software de administración corre en la nube y autodetecta las cámaras para una configuración sencilla. La nube almacena todas las grabaciones críticas hasta 90 días, y los usuarios son capaces de acceder y manejar sus registros desde el sitio web de Camcloud o desde una aplicación móvil. La empresa también ofrece una opción adicional para almacenamiento local para mayor redundancia, de tal forma que, si la conexión entre las cámaras y el Internet se pierde, todo se graba localmente y puede ser revisado a posteriori.

El precio de las soluciones comerciales depende del número de cámaras a conectar, de las opciones escogidas (e.g., detección de movimiento, reconocimiento facial) y del almacenamiento requerido en la nube. Muchos de estos productos tienen versiones gratis, pero muy limitadas. Entre las limitaciones más comunes de las versiones de evaluación, se pueden citar: (1) el número de cámaras que se puede conectar, (2) el almacenamiento en la red, (3) el reconocimiento facial, y (4) la disponibilidad del código fuente. A no ser de código abierto, los usuarios no tienen la posibilidad de agregar o modificar funcionalidades, a gusto.

Si existen centenas de soluciones comerciales, a la fecha, el mundo del código abierto está mucho más limitado, y los grandes nombres incluyen iSpy [28] de iSpyConnect.com, Bluecherry [29] de Bluecherry, Shinobi [30] de Moe Alam,

MotionEyes [31] de Calin Crisan, y ZoneMinder [6][7] soportado por una gran comunidad.

iSpy [28] es un sistema de video vigilancia de código abierto desarrollado en C# que funciona solo en PC con Windows (versiones de 32 y 64 bits). Se sabe que funciona bien en Windows 7/8/10, pero pudiese funcionar en otras variantes de Windows que soporten el .NET Framework v4.5, o superior. iSpy es compatible con la gran mayoría de cámaras web y cámaras IP. El uso más común de iSpy es la seguridad en empresas pequeñas, pero el monitoreo del hogar, la vigilancia del vecindario y el control de los niños son también características valiosas. Con iSpy, los usuarios pueden definir áreas específicas del video que se deben monitorear para ver si hay movimiento y establecer un valor umbral para la cantidad de movimiento que desencadenaría la grabación automática. No hay limitaciones en cuanto al número de sensores (micrófonos o cámaras) que se pueden conectar al software. iSpy se puede descargar y usar de forma gratuita localmente, pero acceder a las cámaras de forma remota, las alertas por SMS, Twitter o emails requieren una suscripción paga.

Bluecherry [29] es una aplicación de código abierto de video vigilancia para Linux (Debian, Ubuntu y CentOS) que soporta cámaras IP. El software hace un bajo uso de memoria, es compatible con ONVIF [22] y toma ventaja del GPU para la detección de movimiento. Esto significa que un servidor de baja gama con un GPU soporta múltiples flujos provenientes de cámaras, con un bajo uso del CPU. Para la configuración y el monitoreo en tiempo real, Bluecherry se puede acceder tanto a través de una interfaz web como a través de un cliente de código abierto soportado en múltiples sistemas operativos (Linux, Windows y MacOS X). El servidor está principalmente escrito en C/C++, con algunos scripts desarrollados en PHP. El cliente está programado en C++ y hace uso de la librería Qt [32].

Shinobi [30] es un servidor de video vigilancia de código abierto escrito en Node.js. Shinobi se basa en FFmpeg [33] (una solución multiplataforma para grabar, convertir y transmitir audios y videos), MariaDB y Node.js, y utiliza masivamente JavaScript, un poco de Python, y algo de Bourne Shell. El servidor es multiplataforma (BSD, Linux, MacOS, Windows) y compatible con la arquitectura ARM, además que también se puede utilizar con una imagen en Docker. Shinobi permite recuperar secuencias de audio y video de cámaras a través de HTTP, RTP/RTSP y ONVIF [22], se admite HTTPS, pero solo con certificados X.509 válidos. En términos de funcionalidad, Shinobi está en algún lugar entre ZoneMinder (utilizable en un entorno profesional y antiguo) y Kerberos.io.

MotionEyeOS [31] es una distribución de Linux que convierte un SBC (Single Board Computer) en un sistema de video vigilancia. El sistema operativo se basa en Buildroot [34-36] y utiliza Motion como backend y MotionEye como frontend.

- Buildroot es un conjunto de Makefiles y parches que simplifica y automatiza el proceso de construcción de un entorno Linux completo y de arranque para un sistema embebido. Utiliza la compilación cruzada para permitir la construcción de múltiples plataformas objetivo (target platforms) en un solo sistema de desarrollo basado en Linux. Buildroot puede construir automáticamente la cadena de herramientas de compilación cruzada requerida, crear un sistema de archivos raíz, compilar una imagen del kernel de Linux y generar un cargador de arranque para el sistema embebido, o puede realizar cualquier combinación independiente de estos pasos. Por ejemplo, una cadena de herramientas de compilación cruzada ya instalada se puede usar de forma independiente, mientras que Buildroot solo crea el sistema de archivos raíz.
- Motion es un programa altamente configurable que monitorea las señales de video de muchos tipos de cámaras. Puede ser utilizado para crear videos o guardar fotos de las actividades capturadas, grabar desde múltiples cámaras IP, ver transmisión en vivo de cámaras, ejecutar scripts cuando ocurren actividades, registrar actividad en múltiples tipos de bases de datos, y dar soporte completo de TLS (HTTPS) con autenticación para control web y transmisiones. Motion es compatible con muchos tipos de dispositivos y protocolos tales como cámaras de red a través de RTSP, RTMP y HTTP, cámaras web V4L2, tarjetas de captura de video, y archivos de películas existentes, entre otros.
- MotionEye es un frontend web para el demonio Motion, escrito y desarrollado bajo el lenguaje Python.

ZoneMinder [6][7] es un conjunto integrado de aplicaciones de código abierto desarrolladas en C++, Perl y PHP que provee una solución completa de video vigilancia, permitiendo la captura, el análisis, la grabación y el monitoreo a través de cámaras de seguridad, conectadas a un sistema Linux o FreeBSD. Fue diseñado para ser ejecutado en distribuciones que soporten la interfaz V4L (Video For Linux) y puede trabajar con una gran variedad de cámaras que incluyen tanto las cámaras USB y FireWire, como las cámaras IP.

ZoneMinder es una opción popular para la video vigilancia de casas como de negocios. Trabaja simultáneamente con varias cámaras. La grabación puede empezar cuando la aplicación detecta cambios entre frames, y el usuario puede definir zonas de las vistas que deben ser ignoradas. ZoneMinder es seguramente el proyecto de código abierto más avanzado y utilizado en la comunidad. Sin embargo, es bien conocido que la configuración de las cámaras no es sencilla, y la documentación de ZoneMinder a este nivel deja mucho que desear.

Eusebio y Mello [37] presentaron una solución para teléfonos inteligentes Android, llamada DroidMinder, que consiste en una aplicación cuyo objetivo es actuar como agente cliente para un sistema de vigilancia ZoneMinder. Es de recordar que si ZoneMinder ofrece un agente propio con una interfaz web simplificada, este no fue pensado para las nuevas tecnologías de teléfonos móviles, motivando este desarrollo.

En ZoneMinder, la información de las cámaras, los usuarios y las alarmas se almacenan en una base de datos relacional MySQL. El cliente predeterminado de ZoneMinder consiste en una página en PHP que accede directamente a esta base de datos, disfrutando de la facilidad de estar ejecutándose en la misma máquina donde se encuentra la base de datos. El objetivo de DroidMinder es permitir a los usuarios monitorear sus cámaras de vigilancia a través de Internet, estando esta aplicación desacoplada físicamente de la máquina donde está en ejecución ZoneMinder. El grupo de investigadores exploraron dos opciones para el diseño de DroidMinder:

- Desarrollar un agente cliente que realice consultas directas a la base de datos MySQL.
- Desarrollar un agente cliente que accede a la interfaz web de ZoneMinder y obtenga los datos multimedia a partir de los flujos de video de las cámaras de vigilancia.

Los autores de DroidMinder optaron por el segundo enfoque por el hecho de ser más transparente para el usuario, pues bastaría proporcionar al DroidMinder la dirección IP y puerto donde se está ejecutando ZoneMinder, para así obtener acceso a las cámaras de vigilancia conectadas a la red. De esta forma, no hay necesidad de realizar modificaciones en la base de datos MySQL, o incluso crear nuevas reglas en el firewall.

En el laboratorio de Investigación Imagelab de la Universidad de Modena y Reggio Emilia, Italia [38], se llevó a cabo un proyecto llamado ViSERaS (Video Surveillance in Emilia Romagna as a Service) entre los años 2010 y 2011. El objetivo del proyecto fue desarrollar un sistema integrado de administración y análisis de videos como un servicio, donde las entidades públicas (en particular en los pequeños municipios de la Región de Emilia-Romagna) podrían disponer de la gestión, almacenamiento y análisis de un gran número de cámaras, a través de un SaaS, aprovechando el ancho de banda proporcionado por Lepida SpA.

Uno de los requisitos más importantes del proyecto era emplear, especialmente para el sistema de Video Management, soluciones de código abierto. Entre las muchas soluciones posibles hoy en día, el laboratorio de investigación eligió ZoneMinder porque exhibe varias características interesantes, tales como:

- Admite varios tipos de cámaras, incluida PTZ.
- Está basado en herramientas de programación estándares como C ++, Perl y PHP.
- Usa un manejador de bases de datos de código abierto (MySQL) y de alto rendimiento.
- Admite videos en vivo en MPEG y JPEG de varias partes e imágenes fijas.

Es de aclarar que, si el reconocimiento facial es una opción bastante común en los softwares comerciales, los desarrollos de dominio público se limitan a la detección de movimientos. Además, en muchos softwares de código abierto, la configuración se debe hacer a través de archivos de configuración y reiniciando los demonios asociados, lo que hace su administración más compleja que los software comerciales. Estas razones motivaron este desarrollo e investigación.

III. Arquitectura y Diseño de la Solución

En la Figura 1, se puede observar las fases establecidas para llevar acabo la solución propuesta. También se muestra, de forma genérica, cómo es el flujo básico del sistema, específicamente él del módulo de reconocimiento facial. A continuación, se especifican cada una de las fases involucradas en la Figura 1.

A. Refactorización de las Interfaces Web para la Administración de ZoneMinder

Parte de la propuesta consiste en la refactorización de las interfaces para el sistema de administración de cámaras IP ZoneMinder. ZoneMinder fue la opción más favorable según la investigación llevada a cabo (ver Sección II) para la refactorización de las interfaces web y la integración de un módulo de reconocimiento facial. Es de aclarar que además de ser de código abierto, ZoneMinder dispone de una buena documentación al nivel de desarrollo, que permite entender y trabajar con el código fuente, facilitando así su modificación para adaptarlo a las necesidades específicas.

La restructuración de las interfaces fue enfocada al mejoramiento de la experiencia de administración de las cámaras a través del sistema propuesto. Además de una mejor experiencia para la administración, el sistema desarrollado permite al usuario tener un excelente control de todas las funcionalidades, inclusive el ingreso desde cualquier tipo de dispositivo (teléfono inteligente, tablet, laptop, o computador).

B. API RESTful para el Reconocimiento Facial

El núcleo principal de la segunda fase fue en el diseño y la implementación de un API RESTful relacionado con el reconocimiento facial, basado en un framework de Python llamado Django [8-10]. El API desarrollado ofrece funciones que incluyen (1) el registro o enrolamiento de usuarios, (2) la carga de imágenes del rostro de los usuarios con diferentes

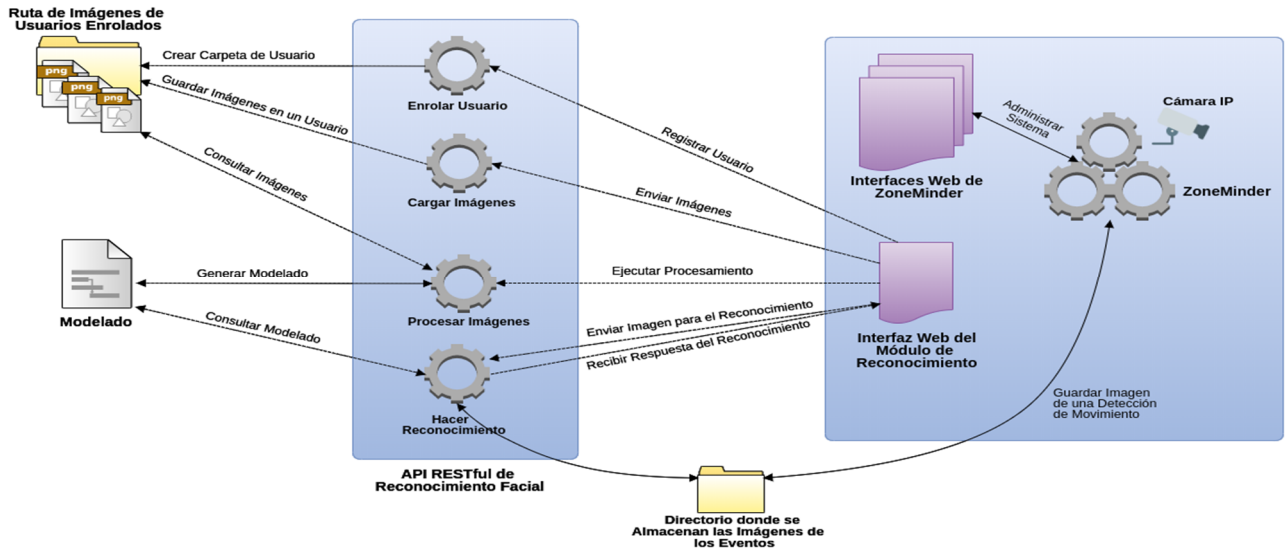

Figura 1: Arquitectura de la Solución

ángulos, (3) la ejecución del procesamiento de imágenes para generar la base de conocimiento para el reconocimiento de los usuarios, y (4) la realización de peticiones de reconocimiento de un usuario, dado una imagen.

C. *Módulo de Reconocimiento Facial*

Este módulo fue integrado como una interfaz web en el sistema de administración de ZoneMinder. La fase de reconocimiento facial, o segunda fase, está constituida por el API y el módulo de reconocimiento facial. A través de este módulo, el usuario tendrá control de todas las funcionalidades presentes en el API.

IV. Pruebas de Rendimiento y Analisis de Resultados

Las pruebas se realizaron estresando directamente el API RESTful sin necesidad de utilizar el módulo de reconocimiento facial que fue integrado en ZoneMinder. Para llevar a cabo las pruebas de rendimiento y de estrés, se utilizó JMeter [39-42], un proyecto de Apache que puede ser utilizado como una herramienta de prueba de cargas para analizar y medir el desempeño de diversos servicios, con énfasis en aplicaciones web. La herramienta de JMeter con la cual se ejecutaron las pruebas corrió en una máquina real (i.e., no en un ambiente virtualizado), fuera del ambiente donde se desplegó la solución. Esta máquina contaba con 16 GB de RAM, un disco duro de 500 GB, un procesador Intel i5-3210M @ 2.50 GHz con 4 núcleos y Ubuntu 18.04 como sistema operativo.

Para implementar el sistema a probar, se utilizó un cluster de tres servidores donde se instaló el hipervisor Proxmox VE [43-46], con las siguientes características agregadas entre los tres servidores:

- 80 núcleos de procesamiento
- 500 GB de memoria RAM
- 5 TB de disco duro

Proxmox VE [46] es una plataforma de código abierto para virtualización que integra el hipervisor KVM [47][48], los contenedores LXC [49], el almacenamiento definido por software (Software-Defined Storage), y funcionalidades de redes en un único sistema, con alta disponibilidad y sencillo de administrar a través de interfaces web. Sobre este marco de virtualización, se creó una máquina virtual con CentOS 7.8 como sistema operativo, donde fue desplegada la solución. La máquina virtual fue configurada con 8 núcleos de procesamiento, 16 GB de RAM y un disco duro de 20 GB.

Se realizaron dos sets de pruebas de estrés para evaluar diferentes aspectos del sistema propuesto cuando se encuentra sometido a una fuerte carga. Para las pruebas, se disponía de un conjunto de imágenes almacenadas en el sistema, obtenidas desde una cámara IP conectada a ZoneMinder, que correspondían a usuarios conocidos. En el primer set de pruebas realizadas, se mantuvo fijo el número de imágenes y se varió el número de peticiones al servicio, como es especificado en la Tabla I. Es importante mencionar que el Rampup de JMeter para esta y todas las demás pruebas fue de 1 segundo. Esto significa que todas las peticiones se dispararan de forma concurrente durante el primer segundo del experimento. Un número de 20 peticiones en un segundo se acerca a la máxima carga que podría tener un hogar. Por otro lado, 160 peticiones en un segundo corresponderían a la máxima carga que se debería observar en una entidad mediana.

Tabla I: Datos Asociados al Set de Pruebas #1

Peticiones	20	40	60	80	100	120	140	160
Imágenes	50	50	50	50	50	50	50	50

Durante la ejecución de las pruebas, se recolectó un conjunto de datos sobre el uso de los recursos del ambiente, empleando la versión de código abierto de Grafana [50], una plataforma para el monitoreo y la observación de experimentos. Con estos datos, se realizaron una serie de gráficas que muestran el comportamiento y el uso de los recursos en la máquina virtual durante la ejecución de las pruebas (uso de memoria RAM, carga promedio, uso del CPU y tiempo de respuesta).

En la Figura 2, se puede observar el uso de la memoria RAM en la máquina virtual para las pruebas del primer set. Por cada experimento, se obtuvieron los valores mínimo, promedio y máximo del uso de la memoria RAM. Se puede observar que, con un número de 50 imágenes y un conjunto variable de peticiones al servicio, el punto más alto de consumo de memoria RAM fue de 11.7 GB. Eso es, un computador mediano con 16 GB debería poder soportar sin problema los requerimiento de memoria RAM.

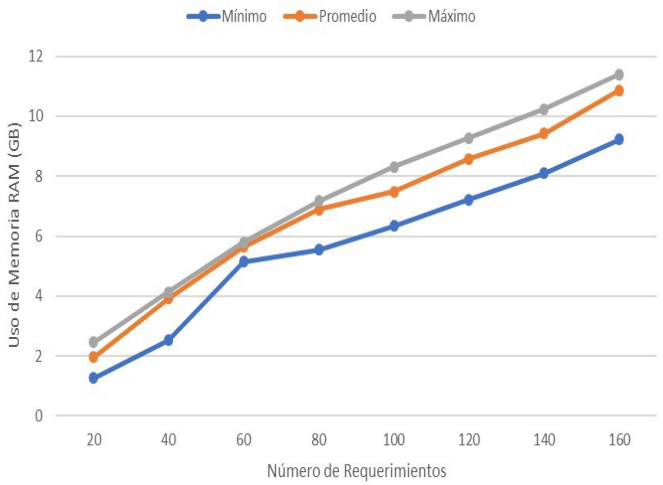

Figura 2: Uso de Memoria RAM para el Set de Pruebas #1

La Figura 3 muestra la "carga promedio" del sistema durante la ejecución de las peticiones, que corresponde al número promedio de procesos a ejecutarse en un tiempo dado, mientras que se ejecutaban peticiones. Se puede observar que la mayor carga registrada durante el primer set de pruebas fue de 56. Como la máquina virtual usada tenía 8 núcleos de procesamiento, esto quiere decir que un momento dado, hubo 48 procesos pendientes por ejecutarse en el sistema. En otras palabras, en el peor periodo de peticiones, se generó un encolamiento de hasta 48 procesos. Sin embargo, la curva promedio de la "carga promedio" revela que cuando se hicieron 160 peticiones, el número promedio de procesos fue de 34.

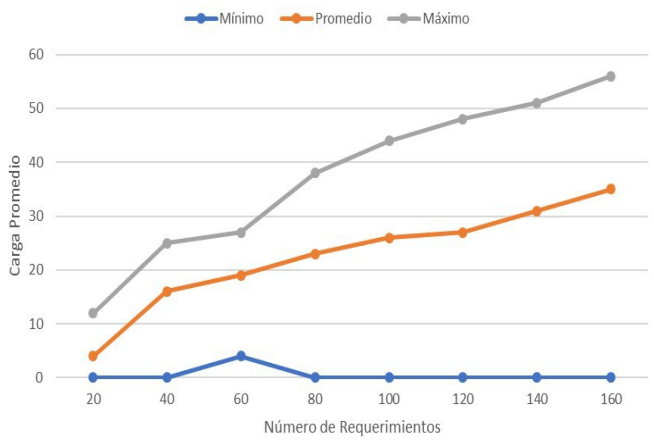

Figura 3: Carga Promedio para el Set de Pruebas #1

En cuanto al uso del CPU en la Figura 4, podemos observar que a medida que aumenta el número de peticiones, también aumenta el uso del CPU. Con la curva promedio, se puede observar que después de 50 peticiones, el CPU tiene un uso de por lo menos 80%, que se podría considerarse como la carga máxima a la cual se debería someter esta máquina virtual.

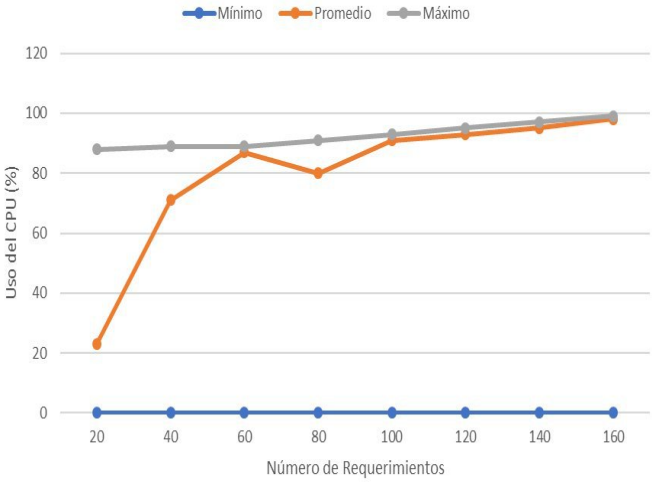

Figura 4: Porcentaje del Uso del CPU para el Set de Pruebas #1

La Figura 5 representa el tiempo de respuesta total para que el sistema pudiese procesar las peticiones. Se puede observar que el tiempo total aumenta en promedio entre 60 y 80 segundos, al incrementar el número de peticiones en 20. Eso es, tenemos un tiempo de procesamiento para el reconocimiento facial del orden de 3 a 4 segundos, cuando el número de imágenes en el sistema es fijo e igual a 50, y el número de peticiones simultáneas es significativo (entre 20 y 160 peticiones). Es importante mencionar que una petición de reconocimiento aislada o con un sistema sin carga, cuando se tienen 50 imágenes en el sistema, se tarda aproximadamente 2 segundos. Obtener un tiempo de respuesta del orden de 3 o 4 segundos cuando el sistema esté estresado (entre 20 y 160 peticiones), es una muestra que escala bien, dentro de este rango de peticiones en paralelo.

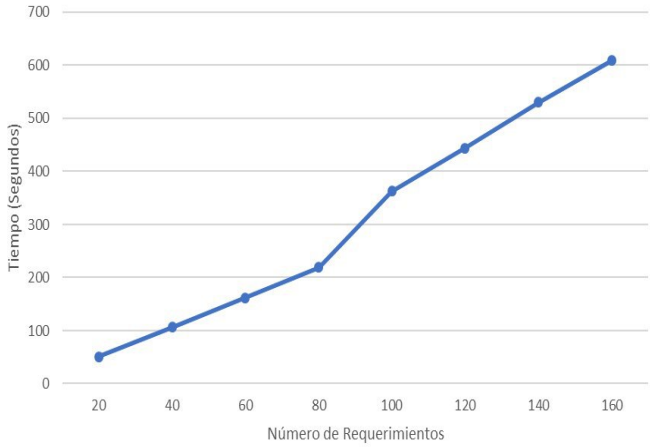

Figura 5: Tiempo de Respuesta Total para Procesar las Peticiones del Set de Pruebas #1

En el segundo set de pruebas realizadas, se mantuvo fijo el número de peticiones y se varió el número de imágenes a procesar, como se puede ver en la Tabla II.

Tabla II: Datos Asociados al Set de Pruebas #2

Peticiones	40	40	40	40	40	40	40	40
Imágenes	20	40	60	80	100	120	140	160

En la Figura 6, se observa el uso de la memoria RAM en la máquina virtual para cada experimento del segundo set de pruebas. Se puede ver que las tres curvas (mínimo, promedio y máximo de uso de memoria RAM) son muy similares entre sí, y el aumento del consumo de memoria es lento. Eso es, el número de imágenes en la base de datos afecta ligeramente el uso de la memoria RAM. Sin embargo, en comparación con el set de pruebas #1 (ver Figura 2), el uso de la memoria RAM es menor en este caso.

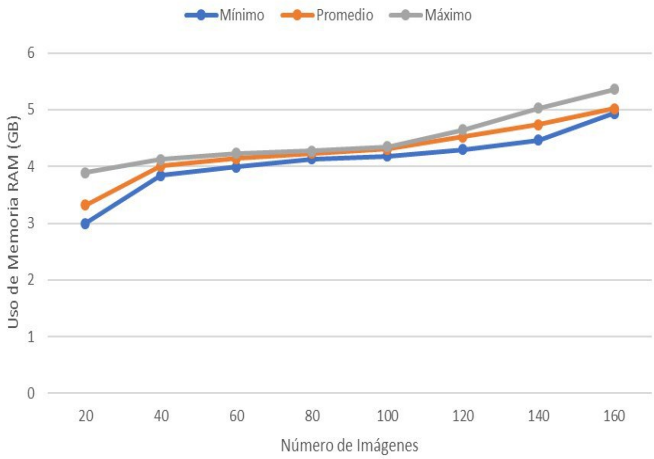

Figura 6: Uso de Memoria RAM para el Set de Pruebas #2

En la Figura 7, se observa la carga promedio de la máquina virtual con este nuevo set de pruebas. Se evidencia que la mayor carga registrada fue en la última prueba (160 imágenes) con una carga de 47 procesos para una máquina virtual con 8 núcleos de procesamiento, lo que implica que en el peor de los casos, hubo 39 procesos encoladas en un momento dado.

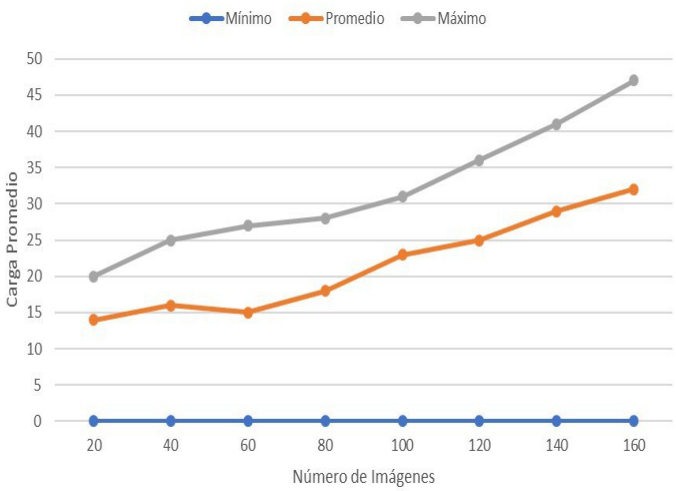

Figura 7: Carga Promedio para el Set de Pruebas #2

En cuanto al uso del CPU, al igual que en el set de prueba anterior (ver Figura 4), en la Figura 8, se evidencia que a medida que aumenta el número de imágenes, también aumenta el uso del CPU. Con la curva promedio, se puede observar que después de 120 imágenes, el CPU tiene un uso de por lo menos 80%, que se podría considerarse como la carga máxima a la cual se debería someter esta máquina virtual.

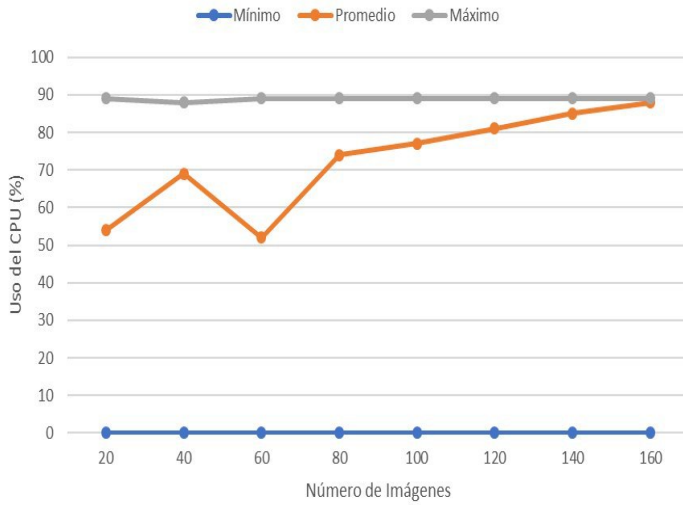

Figura 8: Porcentaje del Uso del CPU para el Set de Pruebas #2

El tiempo de respuesta total para atender todas las peticiones del segundo set de pruebas se muestra en la Figura 9. El menor resultado se obtiene para 20 imágenes y corresponde a un tiempo de respuesta promedio de 50/40 = 1.25 segundos. El mayor resultado se obtiene para 160 imágenes y corresponde a un tiempo de respuesta promedio de 430/40 = 10.75 segundos. Es importante mencionar que, para el caso de 160 imágenes almacenadas en el sistema, un reconocimiento facial aislado se tarda aproximadamente 4.50 segundos. Eso es, el sistema propuesto sigue escalando bien hasta un cierto punto.

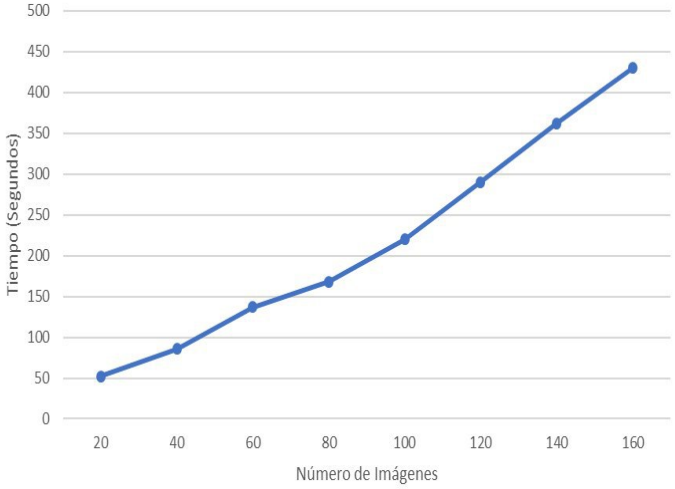

Figura 9: Tiempo de Repuesta las Peticiones del Set de Pruebas #2

V. Conclusiones

Con el constante crecimiento que han tenido las tecnologías involucradas en los sistemas de video vigilancia y el gran interés de llevar estos sistemas hasta los hogares, la comunidad se ha visto en la necesidad de desarrollar softwares de administración de sistemas de seguridad con cámaras IP que sean más adaptados a las necesidades de los usuarios finales. Por ejemplo, hoy en día, se necesitan soluciones de bajo costos con sistemas de administración que provean interfaces fáciles de utilizar, o que incluyan tecnologías como el reconocimiento facial para hacer más preciso el proceso de resguardo y vigilancia. Por lo antes dicho, en este trabajo de investigación, se evaluaron varios sistemas de administración de cámaras IP de código abierto: iSpy, Bluecherry, Shinobi, MotionEyes y ZoneMinder. Como resultado, fue elegido el software de administración de cámaras IP conocido como ZoneMinder [6][7] por ser uno de los más utilizados, y por su amplia documentación para desarrolladores que permitió la integración de otras tecnologías como lo es el reconocimiento facial.

Se desarrolló un API RESTful en lenguaje Python utilizando el framework Django [8-10] que permite a través de un conjunto de servicios y recursos, realizar todo el proceso de reconocimiento facial, desde llevar a cabo el registro de un usuario, hasta realizar un reconocimiento facial basado en una imagen dada. Este servicio está relacionado con un módulo que fue desarrollado directamente en el sistema de administración de cámaras IP de ZoneMinder, donde a través de un conjunto de interfaces web se pueden utilizar todos los recursos desarrollados en el API.

De igual modo, se realizaron un conjunto de refactorizaciones en las interfaces web del software de administración de cámaras IP de ZoneMinder, con la premisa de proporcionarle a los usuarios finales una mejor experiencia de uso a la hora de utilizar el sistema.

Con la intención de comprobar y validar la funcionalidad y el rendimiento de la solución, se realizaron un conjunto de pruebas. En la prueba de funcionalidad se comprobó el correcto comportamiento del módulo de reconocimiento facial y se verificó que las modificaciones realizadas a las interfaces no afectan el correcto funcionamiento del sistema. En las pruebas de rendimiento, con la herramienta JMeter [39-42], se realizaron

dos sets de pruebas al servicio de reconocimiento facial y se evaluó el uso de los recursos. Basado en estas pruebas de estrés, se puede concluir que el sistema sería óptimo para hogares y entidades pequeñas, con el uso de un computador con capacidades medianas, y sin GPU (Graphic Processing Unit). Para un sitio de tamaño mediano, se recomendaría contar con un GPU para el procesamiento digital de las imágenes, ya que utilizando únicamente el CPU, se llega a los límites del sistema en cuanto a uso de los recursos y al tiempo de respuesta de las peticiones de reconocimiento facial.

VI. Trabajos Futuros

Como posibles trabajos futuros, los autores están interesados en:

- Realizar una aplicación móvil que se comunique con el servidor de ZoneMinder y pueda proveer las mismas funcionalidades, pero desde el dispositivo móvil.
- Desarrollar un módulo de reconocimiento facial en el sistema de ZoneMinder bajo el lenguaje PHP, de tal forma que este sea propio del servicio, sin necesidad de contar con un API externa.
- Extender el módulo de reconocimiento facial para que también pueda realizar el reconocimiento de placas en automóviles.

Referencias

[1] J. Romanowich, D. Chin, and T. Lento, *Smart Video Security Handbook: A Practical Guide for Catching Intruders Before They Act*, Video Valley Press, 1 edition, October 2015.

[2] F. Nilsson, *Intelligent Network Video: Understanding Modern Video Surveillance Systems*, CRC Press, 2 edition, December 2016.

[3] N. Dey, A. Ashour, and S. Acharjee, *Applied Video Processing in Surveillance and Monitoring Systems (Advances in Multimedia and Interactive Technologies)*, IGI Global, 1st edition, October 2016.

[4] M. H. Kolekar, *Intelligent Video Surveillance Systems: An Algorithmic Approach*, Chapman and Hall/CRC, 1st edition, July 2018.

[5] E. Robins, *Recent Advances in Video Surveillance*, Clanrye International, February 2015.

[6] ZoneMinder, A Full-featured, Open Source, State-of-the-art Video Surveillance Software System, https://www.zoneminder.com.

[7] A. A. Deshmukh, A. D. Mihovska, and R. Prasad, *ZoneMinder as Software as a Service and Load Balancing of Video Surveillance Requests*, in procceedings of the 2012 ATSMA Networking and Electronic Commerce Research Conference (NAEC 2012), Riva del Garda, Italy, October 2012.

[8] A. Mele, *Django 3 by Example: Build Powerful and Reliable Python Web Applications from Scratch*, Packt Publishing, March 2020.

[9] A. Bendoraitis and J. Kronika, *Django 3 Web Development Cookbook: Actionable Solutions to Common Problems in Python Web Development*, Packt Publishing, 4th edition, March 2020.

[10] G. C. Hillar, *Django RESTful Web Services: The Easiest Way to Build Python RESTful APIs and Web Services with Django*, Packt Publishing, January 2018.

[11] Xeoma, A Video Surveillance Bestseller, https://felenasoft.com/xeoma/en.

[12] Kerberos.io, Video Surveillance Software for Everyone, http://www.kerberos.io.

[13] Milestone Systems, *XProtect*, https://www.milestonesys.com/solutions/platform/video-management-software.

[14] NCH Software, *EyeLine Video Surveillance Software*, https://www.nchsoftware.com/surveillance.

[15] ContaCam - *Video Surveillance Software*, https://www.contaware.com/contacam.html.

[16] Mobile Video Solutions Inc., *Ivideon: Smart Video Surveillance*, https://www.ivideon.com.

[17] Blue Iris Software, *Blue Iris*, https://blueirissoftware.com.

[18] Luxriot, *Evo S*, https://www.luxriot.com/product/video_surveillance/luxriot-evo-s.

[19] Moonware Studios, *webcamXP*, http://www.webcamxp.com/home.aspx.

[20] Hanwha Techwin Europe, *SmartViewer*, https://www.hanwha-security.eu/business-security-products/smart-viewer.

[21] Camcloud, *Cloud Video Surveillance Made Simple*, https://www.camcloud.com.

[22] ONVIF: Open Network Video Interface Forum, https://www.onvif.org.

[23] B. Burns, J. Beda, and K. Hightower, *Kubernetes: Up and Running: Dive into the Future of Infrastructure*, O'Reilly Media, 2nd edition, October 2019.

[24] M. Karslioglu, *Kubernetes - A Complete DevOps Cookbook: Build and Manage your Applications, Orchestrate Containers, and Deploy Cloud-Native Services*, Packt Publishing, March 2020.

[25] M. D'Oliveiro, *The Streaming Media Guide: How to Successfully Integrate Streaming Media into your Communications Strategy*, 1st edition, Routledge, June 2019.

[26] H. Schulzrinne, A. Rao, and R. Lanphier, *Real Time Streaming Protocol (RTSP)*, RFC 2326, April 1998.

[27] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling, *Real-Time Streaming Protocol Version 2.0*, RFC 7826, December 2016.

[28] iSpy, *iSpy User Guide*, https://www.ispyconnect.com/userguide.aspx.

[29] Bluecherry, *Linux Video Surveillance Software*, https://www.bluecherrydvr.com.

[30] Shinobi, *The Open Source CCTV Solution*, https://moeiscool.github.io/Shinobi.

[31] MotionEyeOS, *A Video Surveillance OS for Single-Board Computers,* https://github.com/ccrisan/motioneyeos/wiki.

[32] G. Lazar and R. Penea, *Mastering Qt 5: Create Stunning Cross-platform Applications using C++ with Qt Widgets and QML with Qt Quick*, Packt Publishing, 2nd edition, August 2018.

[33] J. Riselvato, *FFmpeg For Beginners: Edit Audio and Video Like a Pro for Youtube and Social Media*, Independently published, April 2020.

[34] Buildroot, *Making Embedded Linux Easy*, https://buildroot.org.

[35] C. Simmonds, *Mastering Embedded Linux Programming: Unleash the Full Potential of Embedded Linux*, 2nd edition, Packt Publishing, June 2017.

[36] D. Manchón Vizuete, *Instant Buildroot*, Packt Publishing, September 2013.

[37] M. Eusebio y E. Mello, *DroidMinder – Monitoramento de Câmeras de Vigilância Através de um Telefone Celular Android*, Instituto Federal de Santa Catarina (IFSC), São Jose, Brazil, 2010.

[38] R. Cucchiara, A. Prati y R. Besan, *Designing Video Surveillance Systems as Services*, Imagelab Lab, Information Engineering Department of University of Modena and Reggio Emilia, Italia, 2010.

[39] G. Blokdyk, *Apache JMeter: A Complete Guide*, 5STARCooks, July 2018.

[40] A. Rodrigues, B. Demion, P. Mouawad, *Master Apache JMeter - From Load Testing to DevOps: Master Performance Testing with JMeter*, Packt Publishing, August 2019.

[41] B. Erinle, *Performance Testing with JMeter 3: Enhance the Performance of your Web Application*, Packt Publishing, 3rd edition, July 2017.

[42] K. Rungta, *Learn JMeter in 1 Day: Definitive Guide to Learn JMeter for Beginners*, Independently published, May 2017.

[43] W. Ahmed, *Mastering Proxmox: Build Virtualized Environments using the Proxmox VE Hypervisor*, Packt Publishing, 3rd edition, November 2017.

[44] R. Goldman, *Learning Proxmox VE*, Packt Publishing, March 2016.

[45] W. Ahmed, *Proxmox Cookbook*, Packt Publishing, August 2015.

[46] Proxmox VE, An Open-Source Virtualization Platform: Compute, Network and Storage in a Single Solution, https://proxmox.com/en/proxmox-ve.

[47] S. Hyderkhan, *Performance of TCP in a KVM Virtualized Environment*, Lambert Academic Publishing (LAP), January 2019.

[48] Y.-J. Huang, H.-H. Wu, Y.-C. Chung, and W.-C. Hsu, *Building a KVM-based Hypervisor for a Heterogeneous System Architecture Compliant System*, in proceedings of the 12th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, Atlanta Georgia, USA, April 2016.

[49] S. Kumaran, *Practical LXC and LXD: Linux Containers for Virtualization and Orchestration*, Apress, 1st edition, September 2017.

[50] Grafana: The Open Observability Platform, https://www.grafana.com.

Índice de Autores

REVECOM

Sociedad Venezolana de Computación

La Sociedad Venezolana de Computación está comprometida con el impulso de una nueva generación académica y profesional en nuestra área de saber para el desarrollo del país.

Los conceptos y puntos de vista expresados en los trabajos publicados en este libro representan las opiniones personales de los autores y no reflejan el juicio de los editores o de la Sociedad Venezolana de Computación.

ISSN: 2244-7040

9 772244 704006

www.svc.net.ve/revecom