REVECOM

REVISTA VENEZOLANA DE COMPUTACIÓN

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

Sociedad Venezolana de Computación

Vol. 4, No. 2
Diciembre 2017

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. 4, No. 2
Diciembre 2017

Editorial

El Nuevo Modo de Razonar con la Nanotecnología

La nanotecnología se avizora como la tecnociencia de mayor impacto en el siglo XXI. Se espera que dé paso a una nueva revolución industrial fundamentada en la mecánica cuántica, con toda clase de nuevos artefactos antes nunca concebidos y de rendimientos económicos ilimitados: nanorobots, computación cuántica, nanoservomecanismos y otros instrumentos nanocibernéticos y nanoinformáticos. Todos a escala cuántica, valga decir entre 10 a 100 nanómetros o millardésimas de un metro.

En su ensayo "Fundamentos de la Meta-Técnica" (Editorial Monte Ávila. 1990), el filósofo venezolano Ernesto Mayz Vallenilla avizoró a la nanotecnología como un nuevo pensar, una nueva racionalidad o logos que llamó meta-técnica.

La meta-técnica como nanotecnología, con diferente pensar o logos, ofrece una novedosa interpretación de la mecánica cuántica que abre caminos a nuevos fundamentos matemáticos para la nanotecnología, como es el caso de la matemática postmoderna para conformar un nuevo paradigma del pensar para el siglo XXI, un pensar trans-racional.

Mayz sostiene que, en nuestros días, la técnica se está transformando radicalmente y la tradicional razón técnica, que llama *ratio technica*, está dando un salto, a una discontinuidad epistemológica, transmutándose en otra técnica, en una meta-técnica, que crea una supra-naturaleza o una meta-naturaleza y, en consecuencia, necesariamente, un nuevo modo del razonar, un nuevo *logos o principio del razonamiento* que Mayz llama *logos meta-técnico* y que genera un modo de pensar distinto al razonamiento natural; modo de pensar meta-natural que no es i-rracional ni a-racional sino trans-racional.

El nuevo paradigma del razonar *trans-racional*, anunciado por Mayz, es producto de la evolución de la técnica, en cuanto actividad humana eminentemente histórica y de la que depende el futuro del hombre.

La técnica como actividad se da en continua transformación y desarrollo que ha causado cambios experimentales en la ontología y la epistemología de cada época. Pero que, actualmente, en nuestro propio tiempo, se encuentra en una profunda revolución, originada en la praxis de una técnica —acuñada como meta-técnica por Mayz—, que transmuta el quehacer y pensar de la razón técnica, ingénita a la subjetividad y raigambre humana— valga decir, antropomórfica, antropocéntrica y geocéntrica—, en un quehacer y pensar trans-humanos, trans-racionales, autónomos, autárquicos e independientes del logos humano. Trans-mutación en la que se avizora la creación de una supra-naturaleza, una nueva realidad.

En nuestra reflexión "Fundamentos Matemáticos de la Meta-Técnica" (Editorial Publicia, 2014), la meta-técnica, introducida por Mayz, pone de manifiesto un componente desconocido de la realidad; valga decir, un *nuevo mundo*, una meta-naturaleza que denominamos *Mundo 4*, y constituye una ampliación del marco ontológico popperiano de la *Teoría de los Tres Mundos*, valga decir: el mundo físico, el de la conciencia y el de las ideas; y al que sólo se le tiene acceso por una nueva disciplina llamada por Mayz nootecnia.

Editorial

La nootecnia se hará posible mediante la construcción de artefactos meta-técnicos que servirán de traductores entre el logos meta-técnico de la trans-racionalidad (lenguaje trans-finito, fuera del alcance de los límites finitos cognitivos humanos) producto de la meta-técnica y la ratio technica (lenguaje finito, propio del conocimiento y logos humanos) generada por la práctica de la técnica tradicional.

Sobre los fundamentos de la nanotecnología es posible construir y programar una máquina computadora que es el ensamblaje de dos computadoras, una cuántica (formalmente hablando, una máquina de Turing cuántica), como también una computadora universal borrosa (máquina de Turing borrosa, también formalmente hablando) y cuyo acoplamiento sea capaz de satisfacer las condiciones de traductora nootécnica avizoradas por Mayz.

En síntesis, la matemática de la meta-técnica se sustenta en los siguientes dominios para servir de traductora entre la Meta-técnica y la Técnica, en nuestra opinión (ibídem):

> Meta-técnica ⇔ nanotecnología ⇔ mecánica cuántica ⇔ espacios de Hilbert ⇔ lógica computacional cuántica ⇔ lógica computacional borrosa ⇔ lógica computacional booleana ⇔ Técnica

Entonces, de una sintaxis universal en escala cuántica de la información contenida en los eventos cuánticos observados y medidos por sensores nanotecnológicos, se va transformando la información del contenido del trans-lenguaje meta-técnico, hasta un lenguaje técnico particular de la técnica bajo estudio

Mayz predice un cambio total en nuestra racionalidad, en consecuencia, en la del modo tanto de enfocar la ciencia y la tecnología como cualquier otra actividad del quehacer humano; en sus propias palabras:

> Ello significa desde ahora en adelante— tal como se verá en los tiempos por venir— una lenta pero inexorable implantación de nuevas modalidades, horizontes y límites en el despliegue de la racionalidad humana y, por supuesto, en la sintaxis de sus proyectos y gestas instituyente. En algo tan simple como esto — según pensamos — reposan los gérmenes del próximo futuro (Fundamentos de la Meta-Técnica, Monte Ávila, página 16).

Alberto Castillo Vicci
Profesor Jubilado Decanato de Ciencias y Tecnología
Universidad Centro Occidental Lisandro Alvarado

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.

En octubre de 2017, se celebraron conjuntamente la Quinta Conferencia Nacional de Informática, Computación y Sistemas (CoNCISa 2017) y la Quinta Escuela Venezolana de Informática (EVI 2017), en la Universidad Católica Andrés Bello, Ciudad Guayana, Venezuela.

La edición de este séptimo número de ReVeCom está dedicada a los mejores trabajos presentados en CoNCISa 2017. Esta edición consolida un esfuerzo grande que se ha venido haciendo en el seno de la SVC, para promover la investigación en el campo de la computación a nivel nacional, e impulsar una nueva generación académica y profesional en nuestra área de saber para el desarrollo del país.

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.

Además de selecciones de los mejores artículos de conferencias, ReVeCom también publica artículos de investigación en el campo de la computación, a través de un arbitraje por expertos del área. Por ende, se hace una invitación amplia a la comunidad informática nacional, latinoamericana e internacional, a someter sus propios trabajos para los números de ReVeCom por venir.

Directorio de la Sociedad Venezolana de Computación

Presidente:
Dr. Leonid Tineo (Universidad Simón Bolívar)

Vicepresidente:
Dra. Yudith Cardinale (Universidad Simón Bolívar)

Secretario:
Dr. Wilmer Pereira (Universidad Católica Andrés Bello)

Tesorero:
MSc. Rosseline Rodríguez (Universidad Simón Bolívar)

Coordinadora de Educación e Investigación:
Dra. Dinarle Ortega (Universidad de Carabobo)

Coordinador de Publicaciones:
Dr. Eric Gamess (Universidad Central de Venezuela)

Coordinadora de Eventos:
MSc. Nataly Carmona (Universidad Marítima del Caribe)

Edición

Comité Editorial

Miembros del Comité Editorial

Tabla de Contenido

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 2, pp. 1-11, Diciembre 2017
Selección de los Mejores Artículos de CoNCISa 2017

Hacia Aplicaciones Web con Términos Difusos

José Ángel Labbad [1], Rosseline Rodríguez [2], Leonid Tineo [2]
joseangellabbad@gmail.com, crodrig@usb.ve, leonid@usb.ve

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

Resumen: En la actualidad, casi cualquier ámbito de la vida cuenta con una aplicación Web relacionada que dispone de formularios de búsqueda. Lamentablemente, estas aplicaciones tienen problemas de rigidez propios del uso subyacente de la lógica booleana, los cuales se pueden superar gracias a la existencia de lenguajes de consulta a bases de datos, extendidos con lógica difusa. Así, los formularios de búsqueda pueden enriquecerse usando términos difusos en los controles Web. En este trabajo se hace una propuesta metodológica para abordar la migración de una aplicación Web existente hacia una nueva que haga uso de formularios Web con términos difusos, superando los problemas de rigidez de las aplicaciones tradicionales. Para ello, se realizó un estudio de opinión a programadores y usuarios Web para conocer si los programadores desean utilizar términos difusos en sus desarrollos, y si los usuarios se inclinan a realizar búsquedas en formularios Web que proveyeran de estos términos para expresar sus preferencias. Posteriormente, se diseña un mecanismo sistemático que permitiera analizar los controles Web existentes en los formularios de búsqueda de una aplicación para incorporarles términos difusos en sus opciones de entrada de datos. Se evaluaron las diferentes extensiones con lógica difusa de los SGBD que existen actualmente, en base a criterios como completitud, disponibilidad, visibilidad, calidad, soporte, portabilidad y documentación. Se seleccionó SQLfi como la que satisface la mayoría de estos criterios. Finalmente, se plantea una propuesta metodológica para la migración de un sitio Web con el fin de que sus formularios de búsqueda usen términos difusos presentes en el lenguaje natural.

Palabras Clave: Formularios Web; Términos Difusos; Controles Web; Lenguaje Natural; Migración.

Abstract: Currently, almost any area of life has a related Web application that contains search forms. Unfortunately, these applications have rigidity problems inherent to the underlying use of Boolean logic, which can be overcome thanks to the existence of database query languages extended with fuzzy logic. Thus, search forms can be enriched by using fuzzy terms in Web controls. In this work a methodological proposal is made to approach the migration of an existing Web application to a new one that makes use of Web forms with fuzzy terms, overcoming the rigidity problems of traditional applications. For this, an opinion study was made to programmers and Web users to know if programmers want to use fuzzy terms in their developments, and if users are inclined to make searches in Web forms that provide these terms to express their preferences. Subsequently, a systematic mechanism is designed to analyze the existing Web controls in the search forms of an application to incorporate fuzzy terms in their data input options. The different extensions DBMS with fuzzy logic that currently exist were evaluated, using criteria such as completeness, availability, visibility, quality, support, portability and documentation. SQLfi was selected as that one that satisfies most of these criteria. Finally, a methodological proposal for the migration of a Web site is suggested to use fuzzy terms present in the natural language in search forms.

Keywords: Web Forms; Fuzzy Terms; Web Controls; Natural Language; Migration.

I. Introducción

En la actualidad los usuarios de aplicaciones Web demandan mayor flexibilidad y adaptación a sus requisitos o preferencias. La mayoría de estas aplicaciones cuentan con formularios de búsqueda para dar acceso a los datos que ellas gestionan, los cuales en muchas oportunidades presentan rigidez en la adaptación a las demandas de los usuarios.

Para flexibilizar la gestión de datos y búsquedas a través de formularios Web [1], se ha propuesto la lógica difusa [2] como herramienta para la especificación de términos del lenguaje natural. Esta lógica [2] fue definida en base a la teoría de conjuntos difusos [3], propuesta por Zadeh, como mecanismo para dar tratamiento matemático y computacional a los términos vagos del lenguaje natural. Es por ello que se han hecho estudios dirigidos a definir lenguajes que permitan realizar consultas difusas a bases de datos relacionales [4]. Estos lenguajes son extensiones al lenguaje estándar para consultas a bases de datos, llamado SQL (*Structured Query Language*) [5], los cuales se han incorporado en los Sistemas Gestores de Bases de Datos (SGBD) más usados [6][7][8][9].

Algunas de estos SGBD extendidos están disponibles para ser utilizados libremente, sin embargo, no se observa que se haya masificado su uso a nivel de aplicaciones Web. Por otro lado, algunas de las extensiones de SQL con lógica difusa cuentan con implementaciones en distintos SGBD, sea en el código fuente o mediante una capa lógica. Sin embargo, aún no existe un número significativo de aplicaciones que aprovechen las ventajas de trabajar con lógica difusa usando un SGBD.

En este trabajo, se muestra el resultado de un estudio que valida la factibilidad de usar términos difusos en formularios Web, por parte de programadores y usuarios. Además, la principal contribución de este trabajo es una propuesta sistemática y metodológica que permite transformar los formularios de búsqueda existentes dentro de una aplicación Web operativa a una versión modificada que permita la gestión de términos lingüísticos vagos en dicho formularios usando lógica difusa.

En tal sentido el trabajo se estructura como sigue. En la Sección II se plantean los antecedentes de la investigación. En la Sección III, se exponen los conceptos que fundamentan la investigación. En la Sección IV, se muestra un estudio de opinión realizado a programadores y usuarios de aplicaciones Web sobre la posibilidad de realizar búsquedas online usando términos difusos. En la Sección V, se explica el proceso metodológico a seguir para incorporar términos difusos a formularios Web. En la Sección VI, se propone la metodología de migración de formularios Web tradicionales para que incluyan términos difusos. Finalmente, se presentan las conclusiones y trabajos futuros.

II. Antecedentes

A pesar del reciente surgimiento de tendencias NO-SQL [10], hoy en día, el lenguaje de consultas a bases de datos más utilizado a nivel mundial sigue siendo SQL [11]. Gracias a su uso masivo en diversas áreas de aplicación, la perspectiva es que SQL siga siendo ampliamente usado en el futuro.

La incorporación de lógica difusa como mecanismo para flexibilizar los lenguajes de consultas a base de datos se ha venido estudiando por varios años. Las propuestas más completas son SQLf [7] y FSQL [8]. Usando SQLf como lenguaje de consulta, en [12] se desarrollaron aplicaciones basadas en lógica difusa para expresar las preferencias de usuario: un sistema de evaluación de cursos docentes, un sistema de promoción de personal, y un sistema de compra y venta de vehículos. Estas aplicaciones fueron implementadas sobre una extensión en capa lógica del SGBD Oracle denominada SQLfi [12].

En base a esa experiencia y otras previas, en [13] se definieron elementos gramaticales y características que determinan cuando una aplicación posee requisitos difusos, es decir, aquellos que poseen términos lingüísticos vagos. En este trabajo se definen parámetros que ayudan a establecer cuándo una aplicación podría aprovechar el uso de la lógica difusa dentro de un SGBD. Por otro lado, se dan lineamientos para que los desarrolladores puedan identificar, en los requisitos escritos en lenguaje natural, los términos de la lógica difusa que luego serán usados en la definición de consultas a la base de datos.

Posteriormente, en [14] se estudió la incorporación de los beneficios de la lógica difusa en las metodologías de desarrollo de software a fin de expresar requisitos que involucran preferencias de usuario. En este trabajo, se añade al ciclo de vida de desarrollo de software, en las fases de análisis, diseño e implementación, los aspectos necesarios para que las aplicaciones usen consultas difusas sobre bases de datos.

En [15] se propuso un método formal para implementar requisitos difusos dentro de sistemas de software que soporten consultas difusas. De tal forma que, utiliza el cálculo relacional de tuplas para especificar formalmente las consultas difusas, además, provee reglas que permiten traducir una especificación formal a una consulta escrita en el lenguaje SQLf.

Por otro lado, en [16] se propone un perfil UML para representar requisitos difusos a base de datos relacionales. Este perfil hace uso de OCL (*Object Constraint Language*) extendido con condiciones difusas, a fin de completar los diferentes artefactos de la notación UML con requisitos difusos expresados formalmente, lo cual facilita el proceso de traducción a un lenguaje de consultas como SQLf. En [17] se define un algoritmo de traducción de requisitos difusos en OCL a consultas difusas en SQLf. Este algoritmo es de fácil implementación sobre un SGBD difuso, como SQLfi [12] o PostgreSQLf [6].

Los trabajos anteriores [16][17] fueron considerados en la construcción de un *framework* de persistencia [18] que permite el desarrollo de aplicaciones con consultas difusas, el cual es una extensión de Spring, *framework* muy usado actualmente en la comunidad de desarrolladores. Esta extensión puede ser incorporada dentro de ambientes de trabajo, así como, ser integrada con librerías, para el desarrollo de aplicaciones Java.

A pesar que los trabajos aquí mencionados muestran el creciente interés del uso de la lógica difusa como mecanismo para incorporar flexibilidad en diversos sistemas de software sobre bases de datos, ninguno de esos trabajos ha sido enfocado directamente a los formularios de búsqueda presentes en las aplicaciones Web, las cuales tienen mucho impacto en la actualidad. Además, la incorporación de información imprecisa e incierta en el modelo de base de datos sigue siendo un tema importante de investigación debido a que existen datos imprecisos e inciertos en muchas de las aplicaciones del mundo real [19]. Por tal razón, es razonable pensar que incorporar términos difusos en los formularios de búsqueda como funcionalidad agregada dentro de las aplicaciones Web puede resultar de impacto en el presente.

III. Marco Teórico

En esta sección se describe la teoría de los Conjuntos Difusos que es la base matemática sobre la cual se define la lógica difusa, de manera que permita especificar términos del lenguaje natural. Además, se da un panorama general de los elementos presentes en los formularios de búsqueda en la Web, conocidos como controles Web.

A. Conjuntos Difusos

El lenguaje natural que usamos a diario está lleno de imprecisión. Muy rara vez se habla de temperaturas exactas como “38 grados”, más bien se emplean frases como “está caliente”, “hace frio” o “tiene fiebre alta”. Si se opine sobre la edad de una persona es común decir que es “joven”, “adulta” o

"anciana" en lugar de un valor preciso. Al comentar sobre distancias recorridas se indica si algún lugar está "lejos" o "cerca". Para enfatizar algo se usan adverbios dentro de frases imprecisas como "es muy joven" o "está extremadamente lejos". En cuanto a cantidades, es común usar términos como "pocos", "la mayoría" o "muchos". Estos términos del lenguaje natural son mayormente adjetivos o adverbios que se caracterizan por ser impresos, vagos o difusos [13].

Una herramienta matemática que permite modelar estos términos es la Teoría de Conjuntos Difusos [3]. Un **conjunto difuso** permite membresía gradual. En los conjuntos clásicos, se dice si un elemento pertenece o no al conjunto. En los conjuntos difusos, la pertenencia de un elemento x viene dada por un grado de membresía $\mu(x)$ cuyo valor está en el intervalo [0,1]. Entonces, si el grado es cero el elemento está "completamente excluido", y si el grado es 1, el elemento está "completamente incluido". En todos los demás casos la pertenencia es gradual. Al subconjunto de los elementos que están completamente incluidos se les conoce como **núcleo**, a los que no están completamente excluidos (grado de pertenencia mayor que 0), pero tampoco son completamente incluidos se les conoce como **borde**. El **soporte** es la unión del núcleo y el borde.

Para especificar un conjunto difuso, se puede usar una notación trapezoidal de la forma (x_1, x_2, x_3, x_4), donde $x_1 \leq x_2 \leq x_3 \leq x_4$ señalan los puntos de inflexión del trapecio. También, se puede especificar un conjunto difuso con una notación por extensión $\{\mu(x_1)/x_1, \ldots, \mu(x_n)/x_n\}$, donde cada elemento x_i del dominio se acompaña de su grado de membresía $\mu(x_i)$. Estos grados son colocados según la preferencia del usuario.

B. Términos Difusos

Fundamentada en los conjuntos difusos, la lógica difusa permite expresar condiciones con términos difusos, que pueden usarse al realizar consultas o búsquedas sobre un grupo de datos. En esta lógica los valores de verdad son representados por números reales comprendidos en el intervalo [0,1]. Así el cero (0) significa completamente falso y el uno (1) significa completamente cierto, y los demás valores del intervalo representan el grado de verdad obtenido para una proposición que se está evaluando.

El operador lógico negación se interpreta como el complemento de conjuntos, la conjunción como la intersección y la disyunción como la unión. El grado de membresía de un elemento *x* en el complemento del conjunto difuso *F* se calcula como $1-\mu_F(x)$. El grado de membresía para el conjunto intersección se calcula mediante una norma triangular, que es operador binario, cerrado en [0,1], conmutativo, con elemento neutro uno (1). A cada norma triangular le corresponde una co-norma, que es el concepto dual, la cual se usa para el cálculo de la unión. Usualmente se adopta la norma triangular mínimo (*min*) y su co-norma máximo (*max*).

Las expresiones en lógica difusa se construyen utilizando términos lingüísticos, tales como predicados, modificadores, comparadores, conectores y cuantificadores [20].

Los **predicados** [13] representan adjetivos del lenguaje natural, que corresponden a los componentes atómicos de la lógica difusa, los cuales pueden ser definidos con un conjunto difuso en tres formas diferentes: una función trapezoidal, una expresión con rango en el intervalo [0,1] y por extensión indicando para cada valor del dominio del predicado su respectivo grado de verdad. Por ejemplo, los adjetivos "joven", "adulto" y "anciano" para la edad, pueden ser definidos de forma trapezoidal como se observa en la Figura 1. En este caso, el usuario especifica el predicado "joven" con el trapecio (0, 0, 20, 40), "adulto" con el trapecio (20, 40, 60, 80), y "anciano" con el trapecio (40, 60, 100, 100).

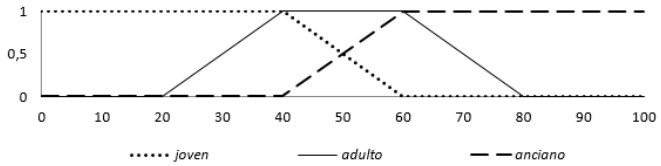

Figura 1: Definición Trapezoidal de los Términos Difusos Joven, Adulto y Anciano

Otro ejemplo, donde se usa una definición por extensión, sería el predicado "agradable", relacionado al universo de los colores = {amarillo, verde, rojo, azul, negro}, puede representarse con el conjunto difuso *agradable* = {0.4/amarillo, 1/verde, 0.1/rojo, 0.7/azul, 0.3/negro}. Un ejemplo, que usa una expresión con rango en el intervalo [0,1] es el predicado "alto" definido con la expresión *x*/250, siendo *x* un valor en el rango 0 .. 250.

Los **modificadores** [13] son adverbios que sirven para expresar una propiedad adicional sobre un predicado, las cuales intensifican, relajan, desplazan o invierten el concepto que representa el predicado. Se pueden definir de tres formas: a través de una potencia a la cual será elevado el valor resultante de un predicado difuso, a través de un valor de traslación que indica lo que le será sumado o restado (según sea el caso) al valor de entrada de un predicado y con una función de norma. Por ejemplo, el modificador "extremadamente" pudiera representarse con la potencia al cubo del valor de la función de membresía, y el modificador "muy" como una traslación negativa.

Los **comparadores** [13] permiten definir una comparación para dos argumentos recibidos. Los términos lingüísticos que los representan son adjetivos de grado comparativo o adverbios. Se pueden definir de dos formas: trapezoidal, si se especifica una expresión con dos variables, cuyo resultado luego de ser evaluada cae en el conjunto difuso definido por un trapecio; o por medio de una relación difusa, donde se especifica para cada par de valores su valor de verdad. Por ejemplo, se podría definir el comparador "cercano", para los valores en el intervalo [-30,30], con la expresión *x-y*; donde el resultado obtenido se evalúa en la función de membresía representada por el trapecio (-20, -1, 1, 20). Un segundo ejemplo es definir el comparador "similar" para colores, usando la relación difusa {(negro,gris)/0.5, (azul,gris)/0.7, (blanco,gris)/0.5, (azul,negro)/0.8, (azul,blanco)/0.2}. Aquí se indica que el negro es similar a gris en un grado de 0.5, mientras que el azul es similar a gris en un grado de 0.7.

Los **conectores** [13] son operadores lógicos que puede definir el usuario, para combinar dos condiciones difusas. Por ejemplo, la negación, la conjunción y la disyunción clásicas se pueden extender de forma difusa, preservando su correspondencia con los operadores de conjunto: complemento, intersección y unión. Así, el $x \Rightarrow y$ puede definirse con la expresión $max(1 - x, y)$.

Los **cuantificadores** [13] permiten describir cantidades difusas. Son representados por adverbios superlativos o cuantitativos, o también, con frases imprecisas que expresan cantidad. Pueden definirse de forma absoluta o proporcional dependiendo de los valores facilitados, usándose en ambos casos la forma trapezoidal. Los cuantificadores clásicos, existencial y universal, se mantienen como cuantificadores difusos. Por ejemplo, un cuantificador absoluto podría ser la frase "porlomenos3" representado por la función trapezoidal (3,8,∞,∞). Un cuantificador relativo podría ser "muchos" representado por el trapecio (0.75, 0.90, 1, 1).

C. Formularios Web

Para interactuar con un sitio o página Web están las interfaces conocidas como formularios Web. Tienen muchos usos, desde funciones tan simples como casillas de búsqueda, suscripciones en listas de correos, libros de visitas o encuestas, así como sistemas de comercio en línea. Los formularios reúnen información del usuario a través de controles como botones, campos de texto o menús deslizantes, pero no procesan los datos. Los controles Web más usados son (ver Figura 2):

- **Campos de entrada de texto**: sirven para introducir una sola palabra, una línea de texto, una contraseña con caracteres ocultos, direcciones de email, enlaces a recursos o sitios Web, y áreas de texto multilínea, con control deslizable.
- **Casillas de verificación**: interruptores que pueden ser activados y desactivados por el usuario, para preguntas multiselección en el caso de aceptar más de una respuesta.
- **Botón de opción**: interruptor que sólo puede activarse o desactivarse un control al mismo tiempo mientras todos los demás quedan desactivados.
- **Número**: Campos de entrada que contienen un valor numérico, pueden aplicar restricciones.
- **Rango**: para registrar valores que deben estar delimitados en un rango definido.
- **Menús desplegables**: sólo puede seleccionarse un elemento cada vez.
- **Menús deslizantes**: permite que los usuarios seleccionen más de una opción de la lista.
- **Color**: Permite al usuario seleccionar un color de la escala RGB.
- **Tiempo**: Permite indicar valores del tiempo como fecha, mes, año, semana, hora específica del día.

IV. Controles Web con Términos Difusos

Muchas aplicaciones utilizan controles Web en las interfaces con los usuarios, los cuales no fueron concebidos para usar lógica difusa desde su diseño. Por tal motivo se realizó un estudio de las posibilidades de conversión de un sistema que ya funciona con lógica precisa a uno que funcione con lógica

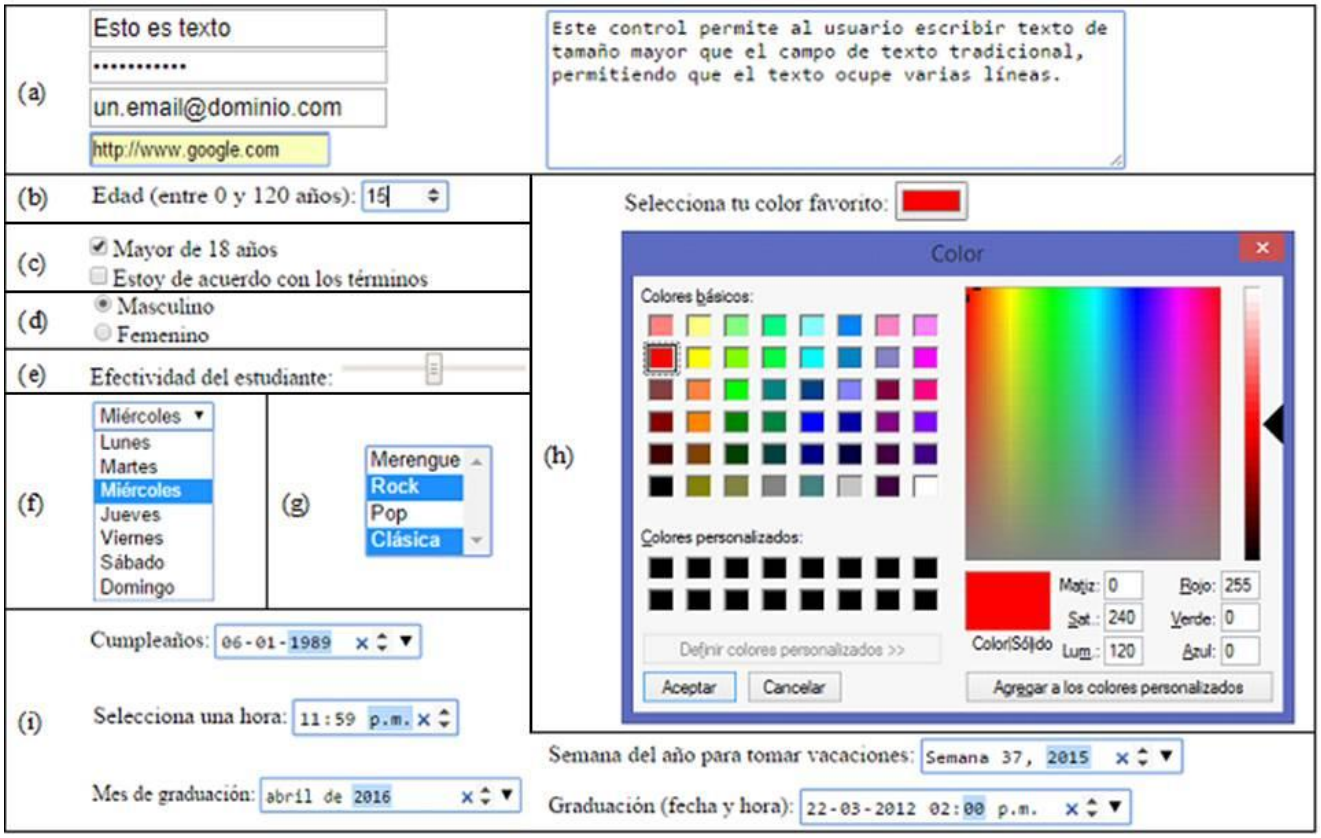

Figura 2: Tipos de Controles Web más Usados: (a) Campos de Entrada de Texto, (b) Número, (c) Casillas de Verificación, (d) Botón de Opción, (e) Rango, (f) Menú Desplegable, (g) Menú Deslizante, (h) Color, (i) Tiempo

difusa. Los detalles de este estudio y sus resultados se describen en [1].

A manera de resumen se mencionarán algunos aspectos. Primero, existen controles Web con entradas precisas que no pueden ser trasformados a difusos: campos de entrada de textos con nombres, contraseñas, dirección email o url y texto multilíneas. Además de los cuales no suelen usarse en formularios de búsqueda (i.e, contraseñas o texto multilínea).

Campos de entrada numéricos, casillas de verificación, botones de opción, menús desplegables, menús deslizantes en la mayoría de los casos pueden asociarse a predicados difusos, representables con un menú desplegable o con casillas de verificación para múltiples selecciones. Las casillas de verificación con múltiples opciones, en algunos casos, también pueden transformarse a un cuantificador difuso.

Los controles relacionados con fechas y tiempo también pueden representarse con predicados difusos (como cercano y lejano) para búsquedas difusas. La transformación de números también puede ser realizada con un comparador difuso.

Los controles tipo rango, donde se especifica un valor dentro de un dominio definido, pueden representarse con un comparador difuso especificado sobre el mismo dominio. Para los colores se puede agregar la funcionalidad de búsquedas con colores parecidos definiendo un predicado difuso que permita evaluar lo cercano que es un color de otro.

A todos los controles convertidos, se les puede aumentar la intensidad a través de modificadores difusos "muy" o "extremadamente", si aporta algo al usuario. Estos se pueden agregar con casillas de verificación o botones de opción. Además, se puede especificar en las búsquedas el grado mínimo de verdad que deben satisfacer todos los resultados presentes en la misma. A esto se le conoce como calibración. Hay algunos controles equivalentes, en esos casos las decisiones se toman por aspectos relacionados con la interfaz.

V. Estudio de Opinión

En el caso de los comercios Web, el impacto de una función de búsqueda efectiva puede ser muy importante. Los clientes que visitan estos sitios, ya saben lo que necesitan y por lo tanto van directamente a la sección de búsqueda. Esta sección muy probablemente es su primer contacto con el sitio Web, por lo que se debe asegurar que los usuarios no reciban resultados equivocados o mal clasificados, sino que realmente reciban lo que están buscando. De lo contrario, la frustración puede llevarlos a un sitio Web diferente.

En muchos casos esto ocurre cuando se hacen búsquedas precisas. Los formularios al tener controles Web que exigen entradas precisas (el valor exacto de una edad, precio, día, distancia, temperatura, etc.) puede ocasionar que las respuestas de las búsquedas sean incorrectas, incompletas o insatisfactorias. De allí surge la propuesta agregar términos difusos a los formularios Web.

Para analizar el impacto y el deseo que tendrían los programadores y usuarios Web de incluir lógica difusa en los formularios de búsqueda, se hizo una encuesta a 249 personas, de 24 países diferentes siendo el grupo más representativo el de Venezuela. Por ello, se analizaron estas 182 respuestas correspondientes a un 43% de usuarios y un 57% de programadores Web, a fin de ver si el proyecto era interesante. La información en común recabada fue edad, sexo, nivel educativo y si es programador o usuario Web. A los programadores se les consultó los SGBD que usaban y si estaban dispuestos a usar lógica difusa en sus desarrollos. A los usuarios se les consultó si al realizar búsquedas en formularios Web habían tenido respuestas negativas dándose cuenta luego que lo buscado si estaba. Además, se le consultó si estarían dispuestos a realizar búsquedas usando términos del lenguaje natural.

En los resultados recabados, un 95% de los encuestados estaban dispuestos a usar términos difusos y un 90% de los programadores estaban dispuestos a incluirlos en sus desarrollos. Ante la pregunta de si les ha ocurrido que utilizando formularios de búsqueda no encuentran lo que necesitan y posteriormente se dieron cuenta de que sí estaba, 94% respondió afirmativamente, concluyendo que los formularios presentan problemas para reflejar las preferencias de los usuarios.

En cuanto a los factores edad, sexo y nivel de educación alcanzado, se observó que no marcan una tendencia claramente diferenciada en ninguno de los casos. Por lo cual las descripciones y tendencias facilitadas en el análisis descriptivo de los datos se pueden considerar generalizadas, de forma independiente a estos factores mencionados.

Sobre la pregunta realizada a los programadores Web, referente a los SGBD preferidos para desarrollar, se observó que, aunque hay una tendencia actual de usar NO-SQL, las respuestas permiten concluir que el mercado sigue dominado el por el lenguaje SQL, con un 90% de preferencia.

VI. Selección de la Extensión con Lógica Difusa

Algunos esfuerzos se han realizado para dar mayor flexibilidad al lenguaje estándar de bases de datos SQL, incorporando elementos de datos y condiciones de consultas basados en los conjuntos difusos. FSQL y SQLf son las extensiones más completas existentes para la incorporación de conjuntos difusos en SQL. Estas dos propuestas tienen enfoques complementarios. SQLf fue propuesta por Bosc y Pivert [7], se enfoca en la extensión de las expresiones de consulta. FSQL, propuesta por Galindo [8] y otros colaboradores, se centra en la extensión de los datos. Muchos trabajos de investigación y desarrollo se han realizado a partir de estas dos propuestas. SQLf es la extensión que permite expresar la mayor cantidad de términos difusos. Varias implementaciones de estas extensiones de SQL han sido realizadas.

SQLfi [12] es un sistema de consultas difusas que recibe instrucciones en el lenguaje SQLf, las cuales pueden ser: definiciones de términos difusos (predicados, modificadores, comparadores, conectores o cuantificadores), definiciones de objetos de datos (tipos, tablas, vistas, aserciones, procedimientos almacenados o funciones), manipulación de datos (inserción, eliminación, actualización), instrucciones transaccionales (*commit* o *rollback*) o consultas difusas. Si bien SQLfi fue desarrollado sobre Oracle 9i, su diseño se hizo pensando en la portabilidad del sistema, por lo que se implementó en Java, usando el estándar de conexión JDBC, como una capa lógica que se encarga del procesamiento de las nuevas funcionalidades.

PostgreSQLf [21] es una extensión de PostgreSQL para el procesamiento de consultas difusas que usa la estrategia de acoplamiento fuerte en su implementación. El *parser* del SGBD fue extendido para el reconocimiento de términos difusos y condiciones difusas. Asimismo, el catalogo y las estructuras abstractas de representación, de manera que son reconocidos por el optimizador, el ejecutor y las funciones de comandos. El uso de acoplamiento fuerte provee mayor escalabilidad y mejor desempeño. Por otro lado, esta complejidad lo hace no portable y con un desarrollo de gran esfuerzo, lo cual ha dificultado su actualización a las últimas versiones del SGBD y completar la implementación de todos los términos difusos.

FSQL [8] posee un servidor desarrollado en Oracle para una base de datos difusa, siguiendo el modelo teórico GEFRED que permite atributos difusos, los cuales almacenan etiquetas lingüísticas, en las tablas de una base de datos. Además, este servidor provee la facilidad de realizar consultas flexibles sobre atributos precisos o difusos. También, el uso y definición de constantes, operadores, comparadores y cuantificadores difusos.

SQLf_j [22] es un sistema que permite consultas difusas a los SGBD relacionales: MySQL y PostgreSQL. Las consultas se especifican en SQLf. SQLf_j funciona como un traductor, de consultas difusas a consultas SQL estándar, las cuales se ejecutan en el SGBD relacional. SQLf_j fue escrito en Java con el *parser* de SQLf creado con yacc y flex, el cual fue reemplazado por un analizador orientado a objetos programado en SableCC.

La metodología desarrollada debía apoyarse en una de las extensiones existentes de SQL que maneje lógica difusa, se decidió establecer algunas características que permitieran evaluar tales extensiones después de realizar un proceso exhaustivo de investigación. Las características se describen a continuación y la presencia en las extensiones con lógica difusa de estas características se muestra, como un cuadro comparativo, en la Tabla I.

Tabla I: Presencia de Características Deseables en las Extensiones con Lógica Difusa de SGBD Existentes

	SGBD con lógica difusa			
Característica	**SQLfi**	**PostgreSQLf**	**FSQL**	**SQLf_j**
Completitud	Sí	Sí	No	Desconocido
Disponibilidad	Sí	Sí	Sí	Sí
Visibilidad	Baja	No visible	Alta	Alta
Calidad	Sí	No	No	No
Soporte	No	No	No	No
Portabilidad	6 SGBD	No	2 SGBD	2 SGBD
Documentación	Instalación y usuarios	Instalación y usuarios	Instalación y usuarios	Instalación
TOTAL	5	3	3	3

- **Completitud**: se refiere a si la extensión maneja todos los términos difusos definidos por Zadeh [22], así como, posee suficiente capacidad expresiva para que sea provechoso y aumente las posibilidades de realizar una migración exitosa.
- **Disponibilidad**: si la extensión está disponible en la Web para su descarga.
- **Visibilidad**: si la extensión es fácilmente ubicable en la Web para su descarga. Relacionada con la anterior, ya que el producto podría estar disponible, pero con muchas dificultades para ubicarlo.
- **Calidad**: si la extensión cuenta con documentación sobre pruebas de calidad, en términos de rendimiento y correcto funcionamiento de los términos difusos implementados en la misma. Todas las extensiones son desarrollos experimentales, por lo cual es importante saber si tienen un control de calidad que permita su utilización en un ambiente de producción.
- **Soporte**: si existen personas u organizaciones con disposición de responder correos electrónicos aclarando dudas sobre la extensión en cuestión.
- **Portabilidad**: si la extensión puede funcionar con múltiples SGBD, y no está atada a uno específico.
- **Documentación**: si la extensión cuenta con documentación para los desarrolladores de software y a nivel de infraestructura para su instalación.

La Tabla I muestra que la extensión SQL que cumple con la mayor cantidad de criterios deseados es SQLf, por tal razón se selecciona como la extensión a ser utilizada en la metodología.

VII. Proyecto de Migración

En esta sección de describe en detalle la propuesta metodológica denominada Proyecto de Migración, cuyo objetivo es transformar un sitio Web con formularios de búsqueda que utilicen lógica clásica para que gestione términos difusos. Se explican los roles a considerar durante el proceso de migración, las diferentes etapas, la documentación y algunos aspectos relacionados a la calidad y viabilidad de la migración. La Figura 3 resume los diferentes componentes del proyecto de migración que se describen a continuación.

A. Roles

En los proyectos de tecnologías de información, particularmente en proyectos de desarrollo o actualización de software, existen diferentes roles a lo largo de la planificación y ejecución de los mismos. En ocasiones, estos roles son cubiertos por una misma persona o grupo de trabajo.

Según Pressman [23], todo proyecto de software requiere de varios participantes que deben organizarse por roles. Tomando como base la propuesta de Pressman [23] se definen los roles para la planificación y ejecución del proyecto de migración de formularios Web.

- **Gerente del proyecto**: encargado de gestionar la planificación del proyecto en términos de tiempo, ejecución, costo y resultados esperados.
- **Especialista en infraestructura TI**: experto en instalaciones de SGBD, configuraciones de sistemas operativos y conocimientos en redes de computadoras.
- **Programador SQL**: persona conocedora de las consultas utilizadas en los formularios de búsqueda a migrar, capacitada en programación en el lenguaje SQL, dispuesta a entrenarse en la extensión SQLfi.
- **Programador web**: personal capacitado en programación, en el lenguaje utilizado en el desarrollo funcional del sitio Web a migrar.

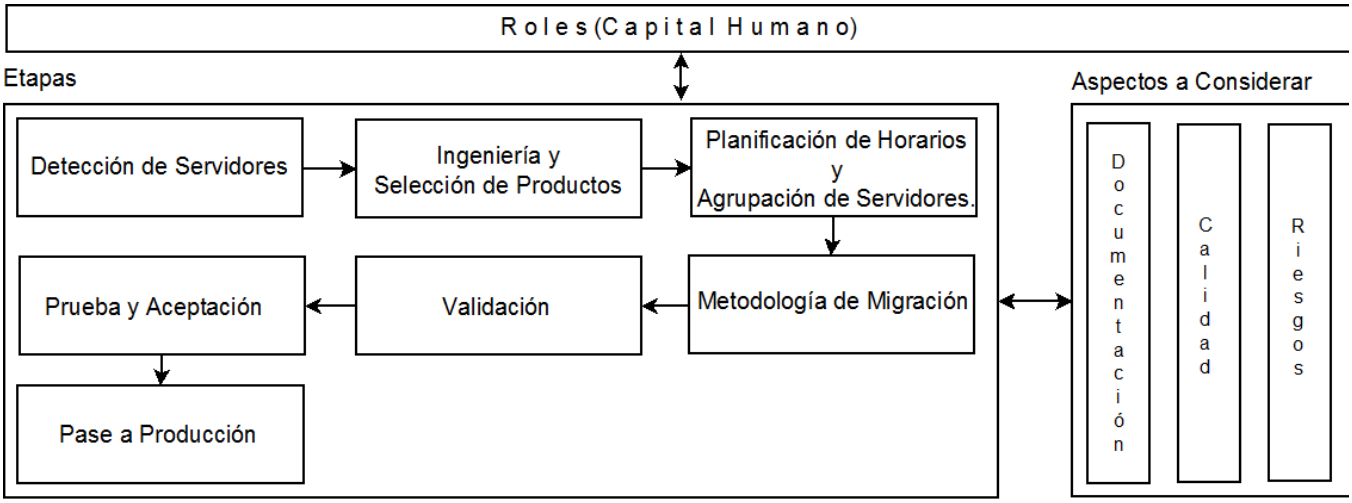

Figura 3: Esquema Resumen del Proyecto de Migración

- **Probador de software/encargado de la calidad**: persona con habilidades para la realización de pruebas y detección de posibles fallas o bugs en los desarrollos de software, específicamente en sitios Web.
- **Diseñador web**: encargado(s) del aspecto visual del sitio Web, preferiblemente quien participó en el diseño inicial.
- **Clientes**: conocedores expertos del modelo de negocio y la funcionalidad del sitio Web. Son los que dan las indicaciones de los cambios a realizar en el sitio Web para incorporarle lógica difusa.
- **Usuarios finales**: grupo de personas que utilizarán el sitio Web una vez migrado, quienes participan en las pruebas de migración. Sólo deben saber navegar en internet y utilizar un formulario de búsqueda. Se sugiere que este grupo sea lo más numeroso posible, y con diversidad en términos de localización, edad, sexo, nivel de conocimiento, entre otros factores.

Adicionalmente, todos los roles deben contar con habilidades de comunicación efectiva y asertiva, así como de trabajo en equipo. A lo largo de todo el proceso será necesario el intercambio de ideas, así como la generación de propuestas.

B. Etapas

En base a la propuesta de Martínez [24], la cual afirma que un proceso de migración de sistemas incluye seis componentes principales, se describen a continuación cada una de etapas con las adaptaciones correspondientes a la propuesta metodológica presentada en este trabajo.

1) Detección de Servidores: Corresponde a determinar los requisitos de diseño relacionados a los productos o las soluciones que se utilizarán en el proyecto de migración. Para el caso de migración a formularios Web con lógica difusa, el requisito a considerar es que el servidor de SQLfi utilizado sea compatible con la arquitectura actual del sitio Web.

2) Ingeniería y Selección de Productos: Se identifica y establece toda la información básica para determinar los sistemas que se deben migrar. Para el caso de migración a formularios Web con lógica difusa se proponen dos actividades concretas:

- **Entrenamiento**: Los roles profesionales, el gerente de proyecto y los clientes deben comprender el funcionamiento y las capacidades de los términos difusos provistos por SQLfi. En el caso de los clientes, deben recibir capacitación en lo que son términos difusos y lo que son capaces de hacer, para ayudar a proponer los cambios a realizar. El programador SQL y programador Web, requieren el mismo conocimiento que el experto del negocio y adicionalmente requieren entrenamiento en el lenguaje SQLf.
- **Selección de controles a migrar**: Los clientes deben analizan uno a uno los controles existentes en los formularios de búsqueda, considerando todas las propuestas y tomando la decisión de migrar o no de forma individual. En algunos casos puede resultar que existe más de una migración viable para un determinado control. Es en este momento cuando el diseñador Web evalúa ambas propuestas, y opta por la que sea más amigable para los usuarios finales y más acorde con el diseño del sitio Web en cuestión.

Una vez que se han diseñado los controles Web difusos, es necesario organizar o coordinar un proyecto de migración de formularios con términos precisos a formularios con términos difusos.

3) Planificación de Horarios y Agrupación de Servidores: Se determina el estado futuro de cada servidor según su configuración actual y se agrupan por grupo de coherencia. Algunos servidores tienen interdependencias que requieren que se los migre de forma conjunta para eliminar la interrupción de servicios esenciales para los usuarios finales. Una vez que se hayan establecido los grupos de coherencia se hacen los cambios por olas de migraciones. Estos grupos sientan las bases para un plan de proyectos real, conformar el cronograma y los hitos de la migración.

4) Metodología de Migración: Corresponde a los procesos que conforman la propuesta de este trabajo de investigación. Las actividades propuestas son:

- **Instalación e integración**: El especialista de infraestructura TI instala y deja operativa la extensión SQLfi sobre el SGBD utilizado por el sitio Web a migrar.

Esto se realiza en un ambiente de pruebas lo más similar posible al ambiente de producción. Luego, el programador de la aplicación Web redirecciona todas las consultas de los formularios de búsqueda hacia SQLfi. Posteriormente, el probador de software verifica que los formularios sigan operando de la forma habitual y sin ninguna incidencia.

- **Definición de términos difusos**: El cliente, junto al programador SQL y programador Web definen los nuevos términos difusos usando SQLfi. Se hacen consultas de prueba sobre este SGBD para verificar que todos los términos están funcionando según lo esperado y acorde a su definición.
- **Definición de consultas SQL**: El programador SQL precisa las consultas apropiadas para las nuevas búsquedas a realizar, en todas sus combinaciones, a fin que el sitio Web sea modificado por el programador Web. La meta es que los formularios de búsqueda envíen las consultas para ser ejecutadas por SQLfi.

5) Validación: Conjunto de tareas que aseguran que el software que se construye sigue los requerimientos del cliente. En esta etapa se realizan pruebas con el objeto de descubrir y prevenir problemas futuros. También, se establecen planes sobre mitigación de riesgos, que consideren los siguientes aspectos:

- Cuando las consultas no funcionen como se espera, no se hacen suficientes pruebas o el ambiente de producción y de prueba no son completamente equivalentes. Se recomienda estar preparado para regresar a la versión anterior del sitio Web.
- Cuando los usuarios no comprendan el funcionamiento del nuevo sistema de búsqueda, debido a un mal diseño de los términos difusos, o de los nuevos controles Web, o no se hicieron suficientes pruebas con múltiples usuarios externos al proyecto. Se sugiere regresar a la versión anterior del sitio Web.

6) Prueba y Aceptación: Se asegura que todos los sistemas funcionen en su totalidad antes de entrar en producción. Para ello, se realizan pruebas de aceptación y un plan de recuperación. En esta etapa el probador de software se encarga de revisar exhaustivamente que el sitio Web no presenta errores en los formularios de búsqueda. También se valida el nivel semántico, es decir, si los resultados son los esperados y que se excluye cualquier otro. Luego de estas verificaciones, se integra a los usuarios finales en el proceso de prueba, así como, en conocer su opinión sobre los cambios realizados. Si éstos no resultan adecuados y agradables, se regresa a la etapa de selección de controles a migrar a fin de tomar nuevas decisiones en base a la experiencia reportada por los usuarios.

7) Pase a Producción: Una vez culminadas todas las etapas anteriores se deben realizar los cambios en el ambiente de producción, y validar su funcionamiento como se hizo en el ambiente de pruebas. Algunas recomendaciones a considerar durante este procedimiento:

- Realizar los cambios en un horario donde el sitio Web tenga el menor tráfico posible.
- Informar a los usuarios del cambio a realizar mediante avisos en el sitio Web.
- Durante el proceso debe estar disponible todo el equipo de trabajo del proyecto para canalizar cualquier inconveniente y solventarlo en la forma más rápida posible.

Desde el punto de vista técnico, se sugiere realizar los cambios en el siguiente orden:

a. Instalar la extensión SQLfi del SGBD usado por el sitio Web en el ambiente de producción.

b. Probar el sitio Web original sobre la extensión, a fin de verificar que los formularios de búsqueda continúan funcionando. Esto es posible pues SQLfi tiene la capacidad de ejecutar consultas clásicas, por lo cual el sitio Web debería operar con total normalidad. Si hay algún problema, revisar el paso anterior.

c. Si todo funciona apropiadamente, se procede a migrar los formularios de búsqueda con los nuevos términos difusos, y se indica a los usuarios el cambio realizado.

En la Tabla II, se resumen las etapas del proyecto de migración y los actores que participan en cada una ellas.

C. Documentación

Durante el proyecto de migración se deben elaborar los siguientes informes a fin de documentar todo el proceso realizado:

- **Cambios en infraestructura de TI**: el especialista en infraestructura de TI reporta todos los cambios a nivel de puertos, las rutas utilizadas para nuevas instalaciones, y cualquier otro cambio pertinente que haya aplicado.
- **Cambios en la arquitectura de la aplicación**: se detalla la arquitectura anterior, los cambios a realizar, y la nueva arquitectura del sitio Web producto de la incorporación de consultas sobre la extensión SQLfi.
- **Términos difusos**: se debe reportar todos los términos difusos definidos, su significado, sus valores, su concepción original y motivación. Además, a nivel técnico, se debe especificar los nombres y dominios que les fueron asignados dentro de SQLfi.
- **Pruebas de calidad**: se deben describir en detalle todas las pruebas realizadas durante la etapa seis, Pruebas y Validación, así como los resultados obtenidos.

Se recomienda que cada documento sea elaborado una vez terminada la etapa correspondiente al mismo. Luego que el sitio Web esté en el ambiente de producción, con resultados satisfactorios, se debe revisar toda la documentación nuevamente para verificar que cualquier cambio ocurrido ha sido reportado apropiadamente. Finalmente, se debe realizar cualquier documentación adicional que aporte valor al proyecto.

D. Calidad

Se propone como enfoque para comprobar la calidad del proyecto comisionar a un grupo de personas que se encarguen de revisar el software, su documentación y la supervisión de los procesos utilizados en el desarrollo. Este equipo verifica que se sigan los estándares seleccionados para el proyecto en

Tabla II: Etapas del Proyecto de Migración y los Actores que Participan en cada Una

Rol \ Actividad	Detección de Servidores	Ingeniería y Selección de Productos	Planificación de Horarios	Metodología de Migración	Validación	Pruebas y Aceptación	Pase a Producción
Gerente del Proyecto	✓	✓	✓	✓	✓	✓	✓
Especialista en Infraestructura TI	✓		✓	✓			
Programador SQL	✓	✓		✓			✓
Programador Web	✓	✓		✓			✓
Probador de Software					✓	✓	✓
Diseñador Web		✓				✓	
Clientes		✓		✓		✓	
Usuarios Finales						✓	

los tres aspectos mencionados. Cualquier estándar que no haya sido cumplido debe ser atendido por el gerente del proyecto.

Además, se sugiere realizar las siguientes verificaciones de calidad:

- A nivel de infraestructura de TI, todos los cambios indicados en la documentación coinciden con el ambiente final de ejecución del sitio Web.
- Los términos difusos escogidos tienen una sola interpretación en lenguaje natural.
- Los usuarios finales seleccionados para las pruebas semánticas y de usabilidad representan apropiadamente todo el universo de usuarios del sitio Web.
- Se realizaron todas las pruebas propuestas con rigurosidad, sin omitir ningún caso.
- La documentación de los términos difusos coincide con lo que está operando en el ambiente de producción.

Estas verificaciones deben realizarse durante las etapas del proyecto que lo ameriten, buscando siempre que el resultado sea lo más satisfactorio posible. Por otro lado, no se excluye cualquier validación adicional que se desee realizar.

E. Viabilidad de la Migración

Para determinar si un sitio Web es factible de ser migrado a trabajar con lógica difusa en sus formularios de búsqueda, se debe contar con algunos requisitos:

- Disponer de capital humano suficiente para cubrir todos los roles descritos.
- Llevar a cabo las dos primeras etapas del proyecto de migración a fin de determinar si al menos un control Web es seleccionado para ser migrado.
- El equipo de trabajo está dispuesto a asumir los riesgos asociados al proyecto.

Se deben considerar todos los riesgos posibles, y planificar para cada uno de ellos una estrategia de mitigación y una de contingencia.

VIII. Conclusiones y Trabajos Futuros

En casi cualquier ámbito de la vida se pueden conseguir aplicaciones Web, las cuales usualmente están dotadas de formularios que permiten la realización de búsquedas para satisfacer requerimientos de los usuarios. De manera que resulta de gran interés el tratar con este tipo de formularios, considerando su extensión usando términos lingüísticos vagos.

En este artículo se han relacionado algunos de los varios trabajos previos que se han hecho en cuanto a la incorporación de lógica difusa en SQL y la consecuente extensión de los SGBD. Como resultado de esos trabajos se cuenta con SGBD que proveen mayor expresividad para búsquedas basadas en preferencias del usuario, los cuales están disponibles para ser utilizadas libremente, pero no se ha masificado su uso en aplicaciones Web. También se han mostrado los pocos esfuerzos previos en el uso metodológico de la incorporación de SQL extendido difuso. Estos trabajos relacionados no habían tenido en cuenta la extensión de un formulario de búsqueda Web existente para agregarle términos difusos.

Se expuso en este artículo un marco conceptual resumido de la investigación realizada que involucra la teoría de conjuntos difusos, los términos lingüísticos y los controles web. La teoría de conjuntos difusos es una herramienta para el tratamiento matemático y computacional de términos vagos del lenguaje natural, la cual da soporte a una lógica gradual, conocida como lógica difusa. Los términos en esa lógica son palabras del lenguaje natural, la cuales se clasifican en: predicados, modificadores, comparadores, conectores y cuantificadores. Por otro lado, los formularios Web permiten el uso de diferentes tipos de controles: campos de texto, casillas de verificación, botón de opción, números, rangos, ingreso de fechas, menú desplegable, menú deslizante y selector de colores.

En un trabajo previo se ha reportado la incorporación de términos difusos en estos controles, a fin de dar mayor flexibilidad a los formularios Web de búsqueda. De tal forma que se provee de un mecanismo sistemático, que permite extender formularios de búsquedas existentes, que no fueron

concebidos originalmente con lógica difusa, para que se facilite la expresión de preferencias de usuarios al involucrar términos difusos. La mayoría de los controles Web existentes puede incorporar términos difusos, por lo que se dispone de una guía que puede ser usada por cualquier programador Web que quiera flexibilizar las búsquedas en una aplicación Web existente, a fin de lograr mayor satisfacción del usuario.

A los fines de validar la aceptación que tendría el desarrollo y uso de aplicaciones Web incluyan controles Web con términos difusos en sus formularios de búsqueda, se realizó un estudio de opinión. Con base a este estudio, se puede concluir que los programadores Web tienen el deseo de utilizar herramientas de lógica difusa. Por otro lado, la encuesta deja ver que actualmente los usuarios de aplicaciones Web demandan mayor flexibilidad y adaptación a sus requisitos o preferencias. Más aún, en muchas oportunidades los usuarios quedan insatisfechos con los formularios de búsqueda presentes en aplicaciones Web debido a su rigidez. De manera que el estudio muestra tanto que los usuarios están deseando este tipo términos en sus formularios de búsqueda Web, así como los programadores están dispuestos a incorporarlos en sus desarrollos. El soporte para estos desarrollos serían los SGBD basados en SQL extendido mediante la aplicación de la teoría de conjuntos difusos. La encuesta muestra que, a pesar de la nueva tecnología NO-SQL, los programadores prefieren actualmente el uso de SGBD basados en SQL.

Antes de proponer el proceso de migración, se analizaron las diferentes extensiones del lenguaje SQL que se implementan sobre los diferentes SGBD existentes, las cuales usan conjuntos difusos en los elementos de datos y las condiciones de consulta. De esta forma, se incorpora la lógica difusa en los SGBD como mecanismo para flexibilizar las consultas y expresar preferencias de usuario. Se estudiaron cuatro extensiones distintas: SQLfi, PostgreSQL, FSQL y SQLf_j. Además, se propusieron ciertas características que permitían evaluar la más conveniente para el proyecto de migración. Como resultado de esta investigación, se concluyó que SQLfi es la extensión con lógica difusa que posee mayor cantidad de características deseables: completitud, disponibilidad, visibilidad, calidad, soporte, portabilidad y documentación. Por tal motivo, se seleccionó SQLfi para ser utilizada en la propuesta aquí presentada.

El principal aporte de este trabajo es una propuesta metodológica que permite realizar la migración de sitios Web existentes para que sus formularios de búsqueda incorporen términos difusos. Se definen los roles de los participantes en el Proyecto de Migración, estos son: Gerente del proyecto, Especialista en infraestructura TI, Programador SQL, Programador Web, Probador de software o encargado de la calidad, Diseñador Web, Clientes, Usuarios Finales. Además, se describen las etapas por las que debe pasar este proyecto: Detección de servidores, Ingeniería y selección de productos, Planificación de horarios y agrupación de servidores, Metodología de migración, Validación, Prueba y aceptación, Pase a producción. Estas etapas son similares a otros proyectos de migración, aunque requieren aspectos distintivos al objetivo del trabajo aquí presentado. En particular, en la etapa de la Metodología de migración es necesario instalar e integrar la extensión SQLfi al SGBD usado por el sitio Web a migrar, establecer las definiciones de los términos difusos y especificar las nuevas consultas que gestionarán estos términos. Finalmente, se dan sugerencias para garantizar la calidad y operatividad del sitio Web migrado, así como, recomendaciones para que el proyecto sea viable.

Como trabajos futuros se espera realizar experiencias prácticas de migración de aplicaciones Web existentes usando la propuesta metodológica aquí presentada. Así mismo, se procederá con el análisis estadístico de estas experiencias a fin de observar los beneficios obtenidos de los sitios Web migrados. Sería interesante evaluar el esfuerzo que debe realizar un programador al tratar de incorporar términos difusos durante la migración de un sitio Web. La expectativa de los autores es que este esfuerzo no sea significativo.

Por otro lado, se espera hacer nuevas versiones de la metodología que abarquen aplicaciones de escritorio y aplicaciones móviles.

AGRADECIMIENTOS

Agradecemos Aquél que nos da fe y valor para emprender proyectos hacia lo desconocido: “Por la fe Abraham, siendo llamado, obedeció para salir al lugar que había de recibir como herencia; y salió sin saber a dónde iba” (Hebreos 11:8).

REFERENCIAS

[1] J. A. Labbad, R. Rodríguez, y L. Tineo, *Formularios Web con Lógica Difusa*, unpublished.

[2] L. A. Zadeh, *Fuzzy Logic - A Personal Perspective*, Fuzzy Sets and Systems, vol. 281, no. C, pp. 4-20, December 2015.

[3] L. A. Zadeh, *Fuzzy Sets*. Information Control, vol. 8, no. 3, pp. 338-353, June 1965.

[4] O. Pivert and P. Bosc, *Fuzzy Preference Queries to Relational Databases*, Imperial College Press, Febrary 2012.

[5] ISO/IEC. *Information Technology — Database Languages — SQL — Part 2: Foundation (SQL/Foundation)*, ISO/IEC 9075-2:2011 (en), 2011.

[6] A. Aguilera, L. Borjas, R. Rodríguez, and L. Tineo, *Experiences on Fuzzy DBMS: Implementation and Use*, in proceedings of the XXXIX Latin American Computing Conference (CLEI 2013), Naiguatá, Venezuela, October 2013, pp. 478-485.

[7] P. Bosc and O. Pivert, *SQLf: A Relational Database Language for Fuzzy Querying*, IEEE Transactions on Fuzzy Systems, vol. 3, no. 1, pp. 1-17, Febrary 1995.

[8] J. Galindo, *FSQL (Fuzzy SQL) A Fuzzy Query Language*. Universidad de Málaga, Málaga, España, 2008, http://www.lcc.uma.es/~ppgg/FSQL.

[9] S. Carrasquel, A. Gyomrey, S. Moreau, R. Rodríguez, B. Stornelli, C.Timaury y L. Tineo, *Extensión de MariaDB para Ordenamiento y Agrupamiento Difuso*, Novática. Revista de la Asociación de Técnicos de Informática, no. 229, pp. 92-97, julio-septiembre 2014.

[10] E. Lai, *No To SQL? Anti-database Movement Gains Steam*. Computerworld, 2009, http://www.computerworld.com/s/article/9135086/No_to_SQL_Anti_database_movement_gains_steam.html.

[11] MetalByte, *¿Cuáles son los Lenguajes de Programación más Demandados en la Empresa?*, muylinux, 2013. http://www.muylinux.com/2013/04/18/cuales-son-los-lenguajes-de-programacion-mas-demandados-en-la-empresa.

[12] M. Goncalves y L. Tineo, *SQLfi y Sus Aplicaciones*. Revista Avances en Sistemas e Informática, vol. 5, no. 2, pp. 33-40, Mayo 2008.

[13] R. Rodríguez y L. Tineo, *Elementos Gramaticales y Características que Determinan Aplicaciones con Requerientos Difusos*, Revista Tekhné, vol. 12, pp. 50-64, Enero 2009.

[14] M. Goncalves, R. Rodríguez y L. Tineo, *Incorporando Consultas Difusas en el Desarrollo de Software*, Revista Avances en Sistemas e Informática, vol. 6, no. 2, pp. 87-101, Noviembre 2009.

[15] M. Goncalves, R. Rodríguez y L. Tineo, *Formal Method to Implement Fuzzy Requirements*, DYNA, Revista de la Facultad de Minas, vol. 173, no. 2, pp. 15-24, Enero 2012.

[16] R. Rodríguez y M. Goncalves, *Perfil UML para el Modelado Visual de Requisitos Difusos*, Enl@ce: Revista Venezolana de Información, Tecnología y Conocimiento, vol. 6, no. 3, pp. 29-46, septiembre-diciembre 2009.

[17] R. Rodríguez y M. Goncalves, *Implementación de Requisitos Difusos en Sistemas Orientados a Datos Utilizando el Lenguaje OCL y Lógica Difusa*, Enl@ce: Revista Venezolana de Información, Tecnología y Conocimiento, vol. 8, no. 1, pp. 31-54, Enero-Abril 2011.

[18] A. Aguilera, M. Goncalves, and R. Rodríguez, *Framework for Fuzzy Application Development*, in proceedings of the XXXVIII Latin American Computing Conference (CLEI 2012), Medellín, Colombia, pp. 478-485, October 2012.

[19] L. Yan and Z.M. Ma, *Modeling Fuzzy Information in Fuzzy Extended Entity-Relationship Model and Fuzzy Relational Databases*, Journal of Intelligent & Fuzzy Systems, vol. 27, no. 4, pp. 1881–1896, July 2014.

[20] L. A. Zadeh, *PRUF – A Meaning Representation Language for Natural Languages*. International Journal of Man-Machine Studies, vol. 10, no. 4, pp. 395-460, January 1978.

[21] A. Aguilera, J. Cadenas, and L. Tineo, *Fuzzy Querying Capability at Core of a RDBMS*, Advanced Database Query Systems: Techniques, Applications and Technologies, IGI Global. New York, USA, pp. 160-184, March 2011.

[22] P. Kalinowski, *SQLF_J*, Poznan University of Technology, Polonia. 2006. http://calypso.cs.put.poznan.pl/~sqlf_j/en/ index.php?.

[23] R. Pressman, *Ingeniería del Software: Un Enfoque Práctico*. México D.F., McGraw Hill, 2010.

[24] H. Martínez, *Ocho Pasos para el Éxito en Migraciones*. The GMB Journal. 2013. https://es.scribd.com/document/246408796/Pasos-Para-El-Exito-en-Migraciones-GBM-Journal.

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 2, pp. 12-20, Diciembre 2017
Selección de los Mejores Artículos de CoNCISa 2017

EAW: Evaluador de Criterios de Accesibilidad Web para Pautas Relacionadas con Discapacidad Visual y Discapacidad Motora

Yusneyi Carballo Barrera, María Gabriela Acosta Vásquez, Ronald Aguilera González
yusneyi.carballo@ciens.ucv.ve, maria.gabriela.acosta.v@gmail.com, ronald.aguilera@gmail.com

Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela

Resumen: La accesibilidad Web indica la posibilidad de acceso a contenidos, servicios o productos disponibles en la WWW por parte de las personas, independientemente de su condición, discapacidad o contexto de uso. El Consorcio World Wide Web mediante la *Web Accessibility Initiative* (WAI) ha promovido el desarrollo de documentos con pautas, técnicas y recursos que ayudan a garantizar un grado de accesibilidad mayor, independientemente de la presencia de una discapacidad visual, auditiva, motriz, del lenguaje o cognitiva en el usuario, del hardware que posea o de su ubicación geográfica. Son ejemplos de estas pautas el *Web Content Accessibility Guidelines* (WCAG) y el *Evaluating Websites for Accessibility*. Garantizar la accesibilidad Web redunda en construir una sociedad donde la comunicación mediada por tecnologías no amplíe brechas. En esta línea, surge el interés en desarrollar recursos para apoyar la evaluación de accesibilidad Web en las investigaciones del CENEAC UCV en el área de Tecnologías Educativas, creándose la Herramienta para Verificación de Criterios de Accesibilidad en Sitios Web (HEVAC) y posteriormente el Evaluador de Criterios de Accesibilidad Web (EAW). Este artículo describe las mejoras incorporadas en EAW con el objetivo de verificar en sitios Web pautas relacionadas con la discapacidad visual y la discapacidad motriz, a partir del análisis de sentencias HTML y CSS, con el posterior despliegue de los resultados de la verificación, el grado de accesibilidad del recurso, los errores encontrados, el código dónde se presentan y recomendaciones para su corrección.

Palabras Clave: Accesibilidad; Discapacidad Visual y Motora; Pautas de Accesibilidad Web; W3C WAI; W3C WCAG; HEVAC; EAW.

Abstract: Web accessibility indicates the possibility of access to content, services or products available on the WWW by people, regardless of their condition, disability or context of use. The World Wide Web Consortium through the Web Accessibility Initiative (WAI) has promoted the development of documents with guidelines, techniques and resources that help guarantee a greater degree of accessibility, regardless of the presence of a visual, auditory, motor, language disability or cognitive in the user, of the hardware that he possesses or of his geographical location. Examples of these guidelines are the Web Content Accessibility Guidelines (WCAG) and the Evaluating Websites for Accessibility. Web accessibility guarantee results in building a society where mediated communication technologies not widen gaps. In this line, there is interest in developing resources to support the evaluation of Web accessibility in the CENEAC UCV research in the area of Educational Technologies, creating the Tool for Verification of Accessibility Criteria in Websites (HEVAC, Herramienta para Verificación de Criterios de Accesibilidad en Sitios Web) and subsequently the Evaluator of Web Accessibility Criteria (EAW, Evaluador de Criterios de Accesibilidad Web). This article describes the improvements incorporated in EAW with the aim of verifying in Web sites guidelines related to visual disability and motor disability, from the analysis of HTML and CSS sentences, with the subsequent deployment of the results of the verification, the degree of accessibility of the resource, the errors found, the code where they are presented and recommendations for their correction.

Keywords: Accessibility; Visual and Motor Disability; Web Accessibility Guidelines; W3C WAI; W3C WCAG; HEVAC; EAW.

I. Introducción

La accesibilidad de los sitios Web está relacionada con la capacidad de acceso por parte de los usuarios, independientemente de las limitaciones que pueda presentar el individuo o por limitaciones que se deriven del contexto de uso o características técnicas del equipo. La accesibilidad Web procura un diseño que permite a las personas navegar, entender, percibir, interactuar y aprovechar contenidos, recursos y servicios, beneficiando no solo a las personas que

presentan una discapacidad, sino también a los adultos mayores y personas que por condiciones temporales ven mermadas sus habilidades y capacidades para el uso de la tecnología [1].

La Organización Mundial de la Salud define **discapacidad** como una deficiencia, carencia o limitación en la capacidad de realizar una actividad en la misma forma o grado que se considera normal para un ser humano, incluyendo las restricciones en la participación y las limitaciones en la ejecución de actividades, en las aptitudes o en las conductas que se esperan de las personas o del cuerpo en conjunto [2].

Valdéz define la **discapacidad visual** como la alteración del sistema visual o la deficiencia en la estructura o funcionamiento de los órganos visuales, razones que ocasionan dificultad en el desarrollo normal de las actividades cotidianas que requieran el uso de la visión [3], bien sea por ceguera total, visión reducida o baja visión. La deficiencia visual es aquella visión menor de 20/400, es decir, se presenta en una persona que requiere estar a 20 pies (aproximadamente 6m) del punto observado, en comparación a la necesidad de estar a 400 pies (aproximadamente 122m) de una persona con visión normal, considerando siempre el mejor ojo y con la mejor corrección. Se considera que existe ceguera parcial cuando la visión es menor de 20/200 (cercanía de 6m, en lugar de 61m) en el mejor ojo y con la mejor corrección, o cuando independientemente de que su visión sea mejor, tiene un campo visual inferior a 20º [2].

La **discapacidad motora** o **discapacidad motriz** se puede definir como el impedimento físico o dificultad para trasladarse, controlar o mover algún miembro superior o inferior del cuerpo, debido a que no lograron desarrollarse normalmente o sufrieron algún traumatismo. Entre las principales causas de alteraciones en el sistema motriz se encuentran enfermedades como la parálisis cerebral (incluyendo la cuadriplejia y hemiplejia), distrofia muscular, esclerosis múltiple, espina bífida, artritis y la enfermedad de Parkinson [2][4].

Para desarrollar código Web accesible el Consorcio World Wide Web (W3C) publicó en el año 1999 a través de la *Web Accessibility Initiative* 1 (WAI, Iniciativa de Accesibilidad a la Web) las guías o lineamientos *Web Content Accessibility Guidelines 1.0* o WCAG 1.0 [5]. En el año 2008 se publicaron las WCAG 2.0, la última versión hasta el momento [6]. También conocidas como **Pautas de Accesibilidad del Contenido en la Web**, fueron definidas para que los desarrolladores tengan a disposición una serie de criterios, técnicas y recursos que los ayuden a crear un diseño accesible y a evaluar el nivel de accesibilidad de sitios y contenidos Web.

Mientras que la WCAG 1.0 está conformada por catorce (14) pautas centradas en técnicas, la WCAG 2.0 se centra en cuatro (4) principios de nivel superior, alrededor de los cuales se organizan pautas específicas que explican cómo hacer accesibles los contenidos. Estos principios conforman el conjunto de propiedades *POUR: Perceivable, Operable, Understandable, Robust* (Perceptible, Operable, Comprensible y Robusto). Al estar centradas en principios, y no en pautas técnicas, los lineamientos de la WCAG 2.0 se mantienen vigentes aún con los cambios en las tecnologías y lenguajes de desarrollo, pasando a ser un estándar técnico internacional en el año 2012 [7][8][9].

La WAI también ha definido criterios relacionados con la evaluación de la accesibilidad, recopilando en el documento *Evaluating Websites for Accessibility* (Evaluación de Accesibilidad de Sitios Web) lineamientos y técnicas para evaluar en forma rápida algunos de los problemas de accesibilidad e indicando procedimientos generales para verificar el cumplimiento de las pautas [10].

Verificar los principios WCAG en un sitio Web se torna una tarea tediosa, pudiendo ser mucho más compleja en función de la cantidad de enlaces y recursos que lo conforman. Por ello, se recomienda incorporar las pautas desde las etapas tempranas del diseño y a lo largo del desarrollo de las aplicaciones, contenidos o recursos. Sin embargo, es un hecho común encontrar aplicaciones y contenidos Web que no cumplen con las pautas de accesibilidad, lo cual motiva la investigación en el área y el desarrollo de aplicaciones para verificarlas, detectar errores y suministrar recomendaciones que ayuden a mejorar la creación de recursos que puedan ser usados por todos.

Con este objetivo fueron desarrolladas las herramientas que se describen en este artículo, el cual ha sido estructurado en las secciones de Motivación y Antecedentes, Desarrollo de la Herramienta EAW, Uso de la Herramienta EAW, Resultados, Conclusiones y Referencias.

II. Motivación y Antecedentes

En la actualidad se cuenta con aplicaciones y sitios Web que en teoría pueden ser usados por todas las personas, pero ¿realmente todos pueden acceder y hacer uso de los contenidos disponibles en la Web? La respuesta a esta pregunta es no, especialmente para las personas con discapacidad o limitación.

Aunque algunos usuarios tengan la mejor disposición para navegar por la red y posean el hardware o software necesario, los estudios realizados por Hassan y Martín [11] sobre evaluación de accesibilidad en sitios web indican, que hay barreras que se deben superar asociadas a la usabilidad, limitaciones presenten incluso en los sitios web de instituciones gubernamentales, como destacan Olalere y Lazar [12]. A manera de ejemplo, Jackson-Sanborn, Odess-Harnish y Warren reportan que al evaluar 100 páginas principales o *index* de organismos federales sólo el 60% era accesible en 2002 [13]; mientras que Loiacono, McCoy y Chin reportan que al evaluar 417 sitios Web federales y de contratistas federales, sólo el 23% cumplían con los lineamientos de accesibilidad de la Sección 508 de la Ley de Rehabilitación o "Acta de los Americanos con Discapacidad" [14][15].

Estas barreras se incrementan además para las personas con discapacidad visual o discapacidad motriz, incluso por la necesidad del uso de un dispositivo tan común como el ratón para dirigir el acceso a la información en la pantalla y la navegabilidad. Cobra importancia entonces la investigación en torno a la creación de herramientas que faciliten la verificación de pautas de accesibilidad e indiquen a los encargados del desarrollo, prueba o certificación de calidad, cuáles son las omisiones o errores presentes en el código de los sitios Web.

En relación a las herramientas para la evaluación de pautas de accesibilidad Web (WAET, *Web Accessibility Evaluation Tools*), han evolucionado en su objetivo y alcance, desde listas

de verificación de pautas, hasta herramientas que en la actualidad verifican en forma automatizada las páginas, detectando problemas de accesibilidad según necesidades de diferentes usuarios (diseñadores, desarrolladores, evaluadores, instancias de certificación, etc.) [9][14].

No todas las herramientas pueden realizar una evaluación automatizada del sitio Web completo o de todo su contenido, en algunos casos sólo pueden evaluar una página a la vez, otras se enfocan en elementos específicos en la página (p.e., menús, colores, textos, metadata de imágenes, tablas) cuyo mal diseño puede limitar el uso del recurso para las personas con discapacidad. En su mayoría, verifican criterios asociados a una de las dos normas mundialmente aceptadas, bien sea las pautas de la WCAG 2.0 para determinar la accesibilidad de la página según los niveles de conformidad y el cumplimiento de los cuatro principios POUR, o trabajan verificando los dieciséis estándares de la norma Sección 508.

En el marco de esta investigación se realizó una revisión detallada de las pautas WCAG, en sus dos versiones. También un análisis comparativo de las herramientas WAVE (*Web Accesibility Evaluation Tool*, WebAIM), TAW (Test de Accesibilidad Web, Fundación CTIC) y HERA (Hojas de Estilo para la Revisión de la Accesibilidad, Fundación SIDAR) [17][18][19].

La **Herramienta para Verificación de Criterios de Accesibilidad en Sitios Web** (HEVAC) [20] se enfocó principalmente en la evaluación de pautas definidas en la WCAG 2.0 que apoyan la accesibilidad de personas con discapacidad visual, aunque algunas de estas pautas también apoyan la accesibilidad de contenidos para personas con discapacidad auditiva y motora. Específicamente se incorporaron en los algoritmos de comprobación 25 técnicas de la WCAG 2.0 asociadas a éxitos y fallos comunes generados con la tecnología HTML, algunas de las cuales se identifican a continuación por su nombre y nivel de conformidad, organizadas según el principio que involucran:

- Principio: Perceptible

  Nivel A: Campo de texto sin nombre, Enlace sin aviso de nueva página, Campo de imagen sin texto alternativo, Botón de formulario sin texto, Página con elemento en movimiento, Página con elemento de parpadeo, Página con refrescamiento automático, Enlace sin destino de referencia, Enlace de imagen sin descripción, Imagen con texto alternativo vacío. Nivel AA: Selección de formulario sin opciones.

- Principio: Operable

  Nivel A: Elemento con accesibilidad vía ratón únicamente. Nivel AA: Campo de texto sin orden de tabulación. Nivel AAA: Área de imagen sin título, Conjunto de enlaces sin indexación tabulada.

- Principio: Comprensible

  Nivel A: Botón de formulario sin valor, Campo del formulario sin etiqueta referenciada, Formulario sin botón de envío, Página sin indicador de lenguaje, Elementos de marcos sin título, Campo del formulario sin etiqueta referenciada. Nivel AA: Tabla sin texto de resumen, Tabla sin demarcación de título, Tabla sin celdas de cabecera, Imagen sin enlace hacia su descripción, Celda de cabecera sin objetivo.

- Principio: Robusto

  Nivel AA: Objeto o *plug-in* sin elemento de sustitución.

El **Evaluador de Criterios de Accesibilidad Web (EAW)** [4] también evalúa considerando los principios POUR y pautas que pueden apoyar la accesibilidad de personas con discapacidad visual incluidas en HEVAC, adicionando pautas para ayudar a personas con discapacidad motora, las cuales se identifican en el punto III.

Inicialmente se identifica el recurso y se indica la cantidad de problemas detectados (pautas que siempre deberían cumplirse pero presentan problemas), las advertencias (pautas recomendadas u opcionales para mejorar la accesibilidad) y el grado de accesibilidad del recurso o grado de accesibilidad de la página Web.

En un segundo nivel de detalle, más orientado a diseñadores y desarrolladores de aplicaciones, se puede consultar los resultados detallados de la verificación, con la identificación de la página Web, histórico de evaluaciones, detalles de la comprobación, descripción de los problemas encontrados, recomendaciones para su corrección y la identificación de las líneas de código en donde se encuentran.

En un tercer nivel, se puede consultar el detalle de cada uno de los errores detectados, clasificados según el principio de accesibilidad (navegabilidad, comprensibilidad, robustez), la incidencia o cantidad de apariciones en la página Web, y un resumen, con el total de elementos identificados en el recurso y el total de elementos evaluados, esta diferencia debido a que la herramienta HERA de la Fundación SIDAR solo verifica elementos en código HTML.

En la Tabla I puede verse un cuadro con las características observadas en las herramientas de accesibilidad Web analizadas y su comparación con HEVAC [4].

Tabla I: Comparación de las Herramientas para Evaluación de Accesibilidad TAW, HEVAC y HERA

Característica	TAW	HEVAC	HERA
Sistemas operativos soportados	Windows, Mac OS, Linux, AIX, Solaris, HP-UX	Cualquier que ejecute un navegador	Cualquier que ejecute un navegador
Lenguaje de Programación	Java	Java	PHP
Estándar verificado	WCAG 1.0 WCAG 2.0	WCAG 2.0	WCAG 1.0
Trabaja en modo *Offline* (versión de escritorio)	Sí aplica	Sí aplica	Sí aplica
Trabaja en modo *Online*	Sí aplica	Sí aplica	Sí aplica
Idiomas	Español, inglés, gallego y catalán	Español	Español, alemán, portugués, catalán, inglés, francés, gallego, italiano, danés, rumano y serbio

Evaluación de múltiples páginas en un mismo ciclo de revisión	Sí lo permite	No lo permite	No lo permite
Reporte de resultados	Sí lo permite, los presenta en versión HTML	Sí lo permite, los presenta en versión HTML	Sí lo permite, disponibles para la descarga en formatos XHTML, RDF y PDF
Almacenamiento de resultados en base de datos	No aplica	No se dispone de información	Sí, por 7 días
Licencia	Propietario	*Open Source*	*Open Source*

La Figura 1 muestra un ejemplo del reporte generado por HEVAC para la evaluación de una página Web.

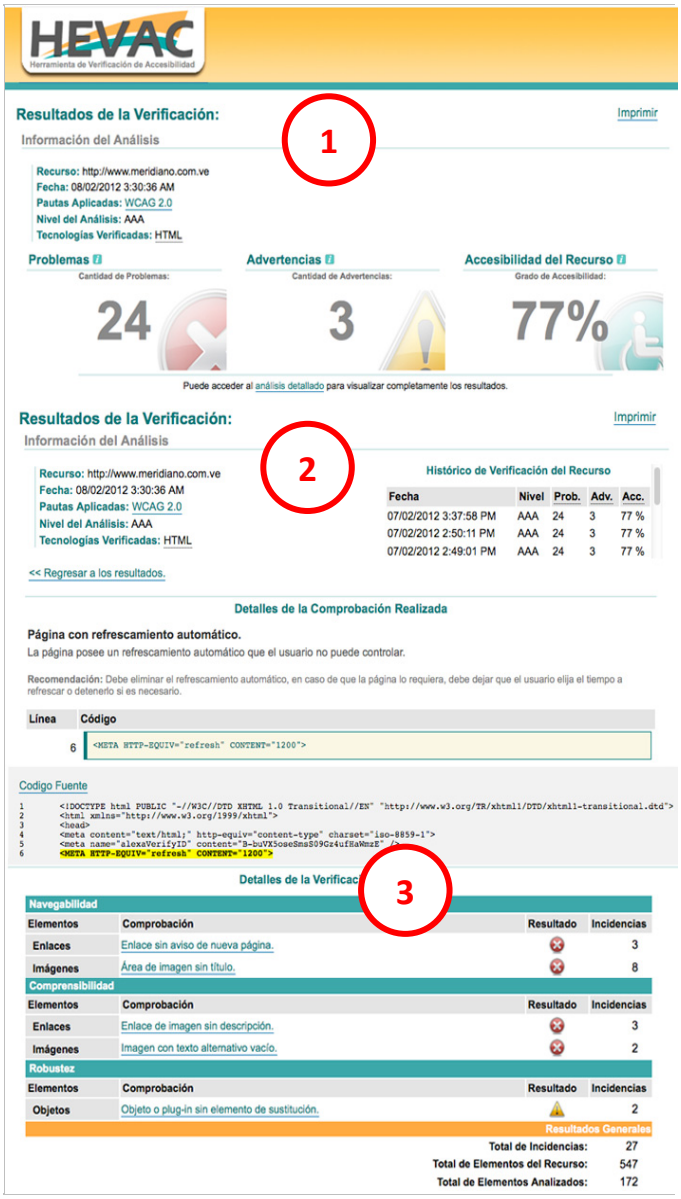

Figura 1: Reporte de Resultados de la Verificación de una Página Web Utilizando HEVAC

III. Desarrollo de la Herramienta EAW

Considerando la conveniencia de ampliar el alcance de HEVAC y de continuar con las investigaciones en el área de la accesibilidad Web, nos enfocamos en diseñar y desarrollar una nueva aplicación que incluyera funcionalidades no disponibles en HEVAC, creándose así el Evaluador de Criterios de Accesibilidad Web, EAW.

A. Objetivo y Alcance

El objetivo principal de EAW es indicar a diseñadores y desarrolladores Web el nivel de accesibilidad de los recursos evaluados, orientarlos en la solución de errores y ayudar a quienes asisten a los usuarios que presentan algún tipo de discapacidad visual o motriz a seleccionar las páginas Web con la mejor accesibilidad y usabilidad. Entre las mejoras incorporadas en la herramienta destacan:

- Considera criterios pertenecientes a pautas relacionadas con discapacidad visual y motriz que pueden ser verificados en forma automática, a fin de apoyar los cuatro principios del estándar WCAG 2.0: crear contenidos perceptibles, operables, comprensibles y robustos.
- Evalúa código HTML y CSS.
- Almacena un historial de resultados para posteriores consultas, con la opción de actualizar una revisión ya existente o eliminarla. Esto facilita mantener un historial de páginas Web evaluadas.
- Los usuarios pueden agregar un comentario a cada evaluación, lo cual permite responder consultas relacionadas con la accesibilidad del recurso.
- Evalúa múltiples páginas Web en una sola revisión, partiendo del URL suministrado y sus enlaces de primer nivel.

A continuación se indican algunas de las pautas que fueron incorporadas a EAW para ampliar la evaluación de criterios de accesibilidad, especialmente relacionadas con el apoyo a personas con discapacidad motora:

- Principio: Perceptible

  Pauta 1.3: Adaptabilidad, creación de contenido flexible que pueda presentarse de diversas maneras, sin perder parte de la información, ni estructura, al tener que adaptarse a otras modalidades y tecnologías.

- Principio: Operable

  Pauta 2.1: Acceso por medio del teclado, permitiendo que toda funcionalidad pueda ser operable a través del mismo.

  Pauta 2.2: Tiempo suficiente, considerando un tiempo promedio prudencial para poder transmitir la información de manera efectiva, sean textos, audios o videos, y permitir interactuar con la aplicación.

- Principio: Comprensible

  Pauta 3.2: Desarrollo de páginas Web que aparezcan y se manejen de manera predecible, incluyendo un adecuado manejo de foco, de la entrada de datos, una navegación consistente y cambios de petición solo a solicitud del usuario.

  Pauta 3.3: Ayuda a los usuarios para evitar y corregir los errores.

- Principio: Robusto

  Pauta 4.1: Compatibilidad, maximizándola para los agentes de usuario actuales y futuros, incluyendo los productos de apoyo y la tiflotecnología.

B. Arquitectura de la Aplicación y Proceso de Verificación

La interacción con la herramienta se realiza a través de un sitio Web desarrollado con páginas dinámicas en PHP, el cual, a partir de datos de entrada proporcionados por el usuario puede realizar una nueva evaluación de los criterios de accesibilidad o mostrar una evaluación previa guardada en el historial (ver Figura 2 [4]).

Figura 2: Página Principal de EAW

La aplicación fue desarrollada bajo arquitectura Modelo-Vista-Controlador. Consta de diez componentes principales, conformados por tres vistas (página de inicio o vista *index*, página de nueva evaluación o vista *create* y página de resultados o vista *show*), un controlador, cinco modelos (*webpage, child, comment, tag* y *css*), además de una base de datos MySQL, como puede observarse en el diagrama de componentes de la Figura 3.

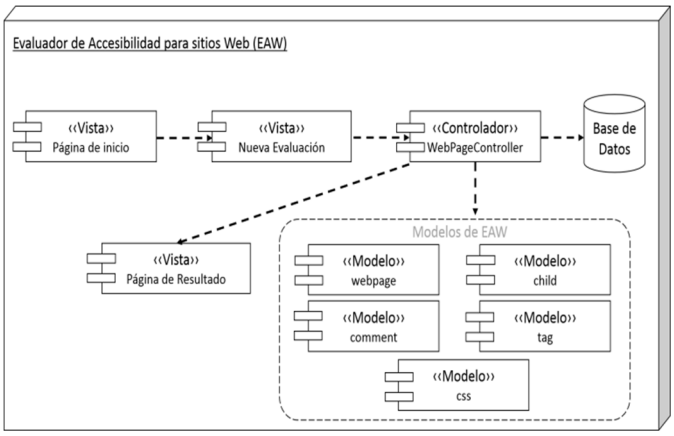

Figura 3: Diagrama de Componentes de EAW

Si el usuario elige la opción de realizar una nueva evaluación se le solicita el URL de la página Web, el nombre con el cual será identificada en el historial y el tipo de evaluación. Se puede elegir entre verificar por principios de accesibilidad (perceptible, comprensible, operable, robusto) o por niveles de conformidad (A-AA-AAA), el botón con signo de interrogación remite al usuario a una ayuda donde se le orienta sobre los principios de accesibilidad y niveles de conformidad, como puede observarse en la Figura 4.

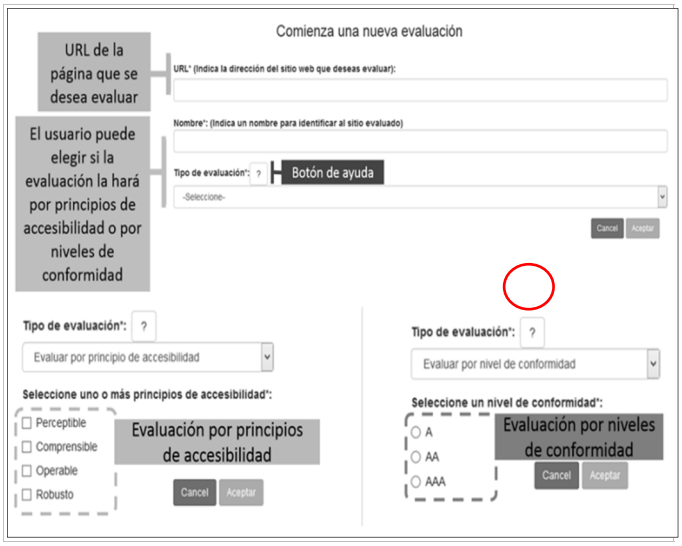

Figura 4: Datos y Opciones para la Evaluación de una Página Web en EAW

El proceso de evaluación tiene una secuencia de eventos que inicia con el despliegue de la vista *index*, la cual muestra el historial de evaluaciones y la opción "Nueva evaluación". Si el usuario elige evaluar un nuevo recurso, el enrutador recibe la petición, se comunica con el componente *WebPageController* (controlador) y se despliega al usuario la interfaz para indicar los datos de entrada del recurso (página Web) mediante la vista *create.*

Al enviar los datos, el método *store* los verifica, incluyendo que se suministre un URL válido, para lo cual se utiliza el método *validateURL* y el modelo *Webpage*, adicionalmente se verifica e informa si existen evaluaciones previas del recurso. Si no se ha registrado previamente una evaluación del recurso en el historial, se procede a crear una nueva entrada en la base de datos, utilizando los modelos *webpage, child* y *comment.*

El controlador suministra el URL verificado y bien formado al paquete cURL [21] a fin de crear un objeto *DOMDocument* de la página Web, considerando el URL suministrado como la raíz del árbol de enlaces [22][23], esto permitirá acceder al código de la página, principalmente su HTML y CSS. El controlador realiza peticiones a los modelos *tag* y *css* para ejecutar las evaluaciones de los criterios de accesibilidad en los elementos codificados en estos lenguajes de etiquetado según pautas específicas del WCAG 2.0.

Finalmente cada modelo suministra al controlador los resultados que son desplegados al usuario por la vista *show*. En la Figura 5 se observa el diagrama de secuencia para la evaluación de un nuevo recurso por principio de accesibilidad.

El reporte de resultados de EAW se realiza con un despliegue progresivo de información, iniciando con la identificación de la página Web evaluada, una captura de pantalla de la página y sus enlaces de primer nivel, los botones de opciones, la accesibilidad global del recurso y la sección de comentarios.

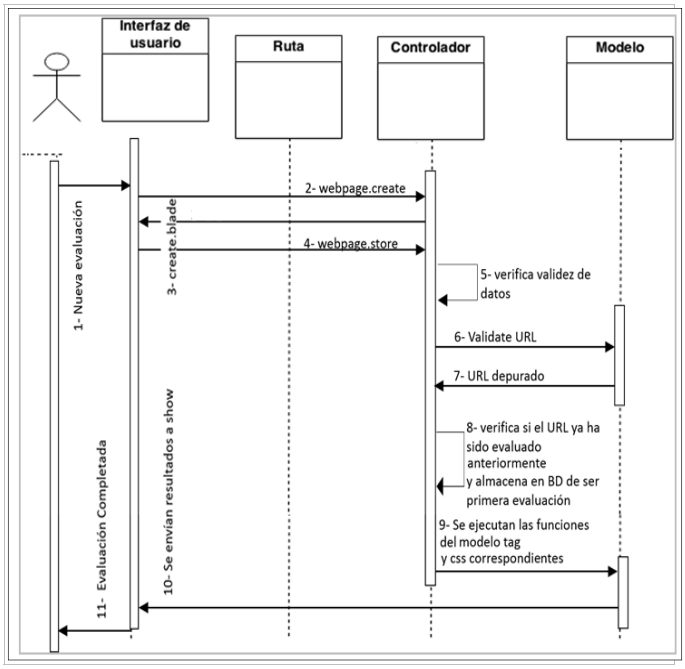

Figura 5: Diagrama de Secuencia para la Evaluación de un Recurso en EAW

En un segundo nivel de detalle, se presentan los resultados de cada elemento evaluado en la página agrupados por etiqueta HTML y una tabla con detalles más específicos como el número de línea en donde fue detectado el error, los atributos involucrados, el principio y si se aprueba o no el criterio de accesibilidad, como puede observarse en la Figura 6.

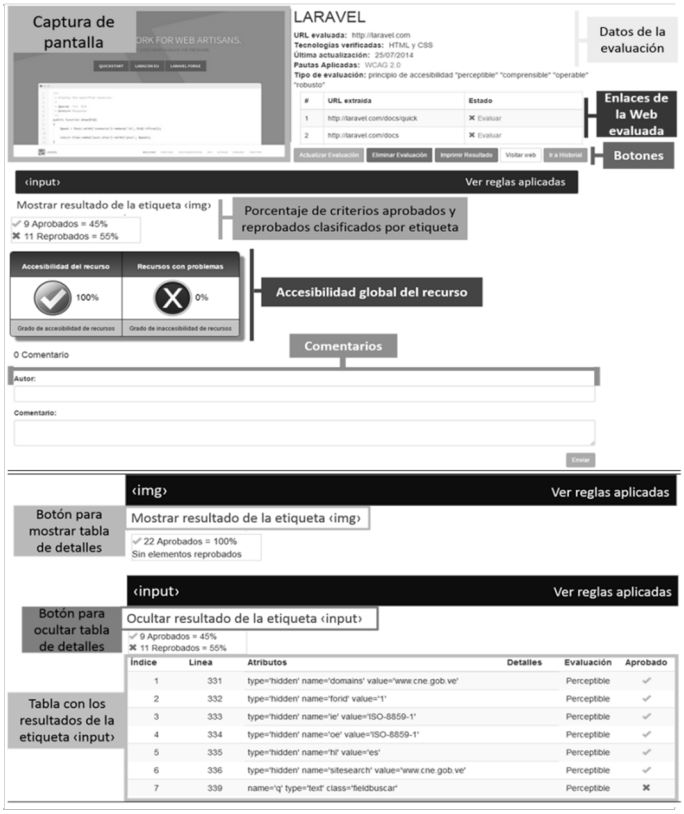

Figura 6: Reporte de Resultados de la Evaluación de un Recurso en EAW

C. Método de Desarrollo y Tecnologías

Se utilizó una metodología de desarrollo ad-hoc, incorporando principios de Programación Extrema (XP) y utilizando algunos artefactos UML para la documentación en las fases de planeación y diseño, entre otros, bocetos de interfaces y diagramas de casos de uso (en lugar de historias de usuario); diagrama de componentes, diagramas de secuencia, diagrama entidad-relación, diagrama de navegación de la aplicación.

Para crear la herramienta se utilizaron las tecnologías Laravel como *framework* de desarrollo, PHP 5.4.3, HTML5, JavaScript, Bootstrap para el diseño de una interfaz adaptativa con CSS3, MySQL como manejador de bases de datos, utilizando un servidor Web Apache. Para la gestión de la información de los URLs, comprobar la existencia del URL a evaluar, explorar el contenido de la página Web, obtener su árbol de enlaces y copiar el contenido del URL, se utilizó el paquete cURL de la librería libcurl. Compatible con PHP, cURL permite la conexión y comunicación con diversos servidores y protocolos, entre otros, http, https, ftp, gopher, telnet, dict, file y ldap [4][21].

En la Tabla II se muestra una comparativa entre las herramientas EAW, TAW y HEVAC, pudiéndose visualizar de manera más concreta características comunes y diferencias [4].

En el proceso de desarrollo se realizaron pruebas unitarias a las funcionalidades. Una vez desarrollada EAW, se verificó la usabilidad de la aplicación con la participación de diez (10) usuarios con distinto nivel de experiencia en uso de aplicaciones Web y con perfiles de desarrollador de aplicaciones, docente y público general.

Tabla II: Comparación de las Herramientas para Evaluación de Accesibilidad TAW, HERA y EAW

Características	TAW	HEVAC	EAW
Pautas	WCAG 1.0, WCAG 2.0, MobileOK	WCAG 2.0	WCAG 2.0
Tecnologías soportadas	HTML, CSS, JavaScript (parcialmente)	HTML	HTML, CSS
Técnicas HTML	Completa	Parcial, criterios de discapacidad visual	Parcial, criterios de discapacidad visual y motriz
Selección del nivel de conformidad (A-AA-AAA)	Sí	Sí	Sí
Selección de principios de accesibilidad	No	Sí	Sí
Clasificación de comprobaciones	Por numeración de criterios de éxito	Según técnicas propias de la herramienta	Por principios de accesibilidad y niveles de conformidad
Se indican las comprobaciones no realizadas	Sí	No	Parcialmente, se indica el número de errores suministrado por el servicio de validación de marcado de la W3C [24]

Agrupación de resultados por principios	Sí	Sí	Sí
Consejos de desarrollo	No (suministra enlace a la página de la técnica en el sitio WCAG)	Sí	Sí
Se indican el grado de accesibilidad	No	Sí	Sí
Pre-visualización de la página evaluada	Sí	No	Sí
Se destaca el problema en el código fuente	Sí	Sí	Sí
Exportación e impresión de resultados en PDF	No	Sí	Sí
Registro histórico de verificaciones	No	Sí	Sí
Verificación de tecnología HTML	Sí	Sí	Sí
Verificación de tecnología CSS	Sí	No	Sí
Incorporación de comentarios del usuario	No	No	Sí
Revisión de las páginas hijas del URL	Sí	No	Sí

Luego de utilizar EAW los usuarios respondieron un cuestionario con doce (12) preguntas, siete (7) obligatorias de selección simple y cinco (5) opcionales de respuesta abierta. De esta prueba se obtuvieron sugerencias para mejorar los textos de ayuda que explican los criterios de accesibilidad verificados y funcionalidades, el despliegue de información del historial de páginas Web, la búsqueda de las páginas registradas, la explicación de los resultados detallados de los errores, entre otras. Ante la solicitud de evaluar la funcionalidad general de la aplicación en una escala de 1 (mínimo) a 10 (máximo), el 70% de los usuarios la calificaron con el máximo puntaje de 10, el 20% la calificó con 9 y el 10% con 8. En términos globales las opiniones fueron favorables, destacando que la herramienta es intuitiva en su uso, además de útil para las personas con discapacidad, para quienes los apoyan en el uso de aplicaciones Web y para los desarrolladores.

Finalmente, una instancia de EAW fue sometida a evaluación para identificar y corregir problemas de accesibilidad, lo cual se describe en la próxima sección.

IV. Uso de la Herramienta EAW

Como se ha mencionado anteriormente, la herramienta EAW extendió el alcance de la verificación para incluir criterios relacionados con la accesibilidad de usuarios con discapacidad motriz, complementando los asociados a la discapacidad visual incluidos en investigaciones previas.

Estas dos categorías son de interés por ser comunes en la población, no sólo como condiciones de nacimiento o congénitas, sino como consecuencia de accidentes, enfermedades o por el avance en la edad. Incluyen deficiencias físicas, limitaciones de la actividad y restricciones de la participación que también afectan a usuarios de aplicaciones, servicios y productos Web. La OMS estimó que para el 2011 más de 1000 millones de personas vivían en todo el mundo con alguna forma de discapacidad; para 2014 había aproximadamente 285 millones de personas con discapacidad visual, de las cuales 39 millones eras ciegas y 246 millones presentaban baja visión [2]. Tomando en cuenta que la población está envejeciendo, que el riesgo de discapacidad es superior entre los adultos mayores y también que aumentan las enfermedades crónicas (diabetes, cardiovasculares, cáncer y trastornos de la salud mental), la OMS, indica que la prevalencia de la discapacidad seguirá aumentando en años futuros [25].

Estas estadísticas deben llamar a concientizarnos sobre la importancia de crear recursos que cumplan con los principios de accesibilidad para garantizar un diseño universal, entendido como el diseño de productos, servicios o entornos para la mayor cantidad de usuarios posible, sin que tengan que ser adaptados o rediseñados. Estos principios generales del diseño, son aplicables en la arquitectura, la ingeniería y también en la informática [26].

EAW se utilizó para evaluar sitios Web correspondientes a instituciones públicas y privadas, en áreas que consideramos de importancia para el acceso a servicios, a contenidos y el desarrollo de la ciudadanía digital, sitios web en donde se debería garantizar la accesibilidad a la información y a los medios electrónicos de manera segura y comprensible [27]. Ejemplos de los resultados de estas evaluaciones se presentan en la Tabla III, mostrando el porcentaje de accesibilidad, de recursos con problemas y el error de mayor incidencia en un conjunto de sitios Web relacionados con los sectores educación universitaria, instituciones gubernamentales y banca.

Tabla III: Uso de EAW para Evaluación de Sitios Web de Instituciones en el Sector Educativo, Salud y Gobierno (2014)

Característica / Sitio Web	Porcentaje de accesibilidad del sitio Web	Porcentaje de recursos con problemas	Error con mayor incidencia
Escuela de Computación UCV	93,27%	6,73%	a
Escuela Biología UCV	84,78%	15,22%	b
UCAB, Caracas	99,71%	0,29%	b
USB	91,48%	8,52%	a
Consejo Nacional Electoral	85,11%	14,89%	c
Gobernación de Miranda	85,11%	14,89%	c
Seniat	51,75%	48,25%	c
Banco de Venezuela	91,99%	8,01%	a
Banesco	83,05%	16,95%	a
Banco Mercantil	97,05%	2,95%	b

Los errores de mayor incidencia fueron:

a. Uso del mismo valor para el atributo name en la etiqueta <*a*>, es decir, varios enlaces tienen el mismo nombre y esto ocasiona ambigüedad al momento de orientar al usuario en la navegación de la página Web.

b. La etiqueta <*img*> debe tener el atributo alt, lo cual se traduce en que no se incluyó texto alternativo para señalar la existencia de la imagen y describirla. Si la imagen no puede ser mostrada por el navegador o la página está siendo usada por una persona con discapacidad visual, el atributo alt permitiría indicar la presencia de la imagen, así como describírsela, en la traducción de texto a voz del contenido de la página con el apoyo de alguna herramienta tiflotecnológica.

c. En las tablas debe utilizarse las etiquetas <*tr*> y <*th*>, esto implica que las tablas deben presentar una estructura correcta, completa en sus etiquetas y atributos, y con delimitación de filas, celdas de título, celdas de datos y título que identifique a la tabla.

Otro error detectado con frecuencia fue la falta del atributo tabindex en los campos de los formularios, lo cual dificulta establecer el orden en que se recorren los campos utilizando el teclado, opción utilizada tanto por las personas con discapacidad motora, como con discapacidad visual.

Analizando los resultados de EAW para esta muestra de diez (10) sitios Web, sólo el 50% presenta un porcentaje de accesibilidad superior al 90%. Debe destacar que los sitios con el menor porcentaje de accesibilidad se relacionan con servicios bancarios y tributarios, en este caso, el sitio web de Banesco (83,05%) y del Seniat (51,75%).

El 20% (2 de 10) presenta menos del 5% de errores en los criterios de accesibilidad verificados, el 40% presenta entre 5% y 15% de criterios con problemas, y el 30% tiene más de 15% de criterios con errores de accesibilidad. En este indicador también destaca el sitio web del Seniat con 48,25% de recursos con problemas de accesibilidad.

En términos generales, las dificultades que comúnmente afectan el acceso de las personas con limitación o discapacidad motora están relacionadas con el uso del ratón, el teclado y las pantallas táctiles como dispositivos para orientar la navegación y la selección de opciones en el sitio Web.

En el caso del uso del ratón puede que los usuarios no tengan la precisión o coordinación necesaria para ejecutar los movimientos; en el caso del teclado necesitan tener la fuerza y precisión para teclear, y en las pantallas táctiles la precisión, presión, control del tiempo y del movimiento para escribir o seleccionar las opciones. Algunas personas mayores y personas afectadas por la artritis u otras inflamaciones en las extremidades superiores, codos, manos, presentan dolores en las articulaciones que pueden causar fatiga y limitar el tiempo de empleo del ratón o del teclado.

En el caso de las personas con discapacidad visual, las limitantes más comunes se relacionan con no poder tener acceso a toda la información incluida en la página Web, no poder modificar el tamaño en que se presentan los contenidos o el uso de colores que no tienen buen contraste.

En las páginas Web se incorporan estructuras mal diseñadas, en especial tablas, capas, marcos, imágenes, animaciones, videos, botones e hipervínculos. También es común que no se incluyan identificadores únicos, metadatos, descripciones y textos alternativos. Estos errores son muy comunes a pesar de la existencia de pautas de accesibilidad Web que todo diseñador y programador debería conocer y emplear.

Es importante realizar una correcta programación de estos elementos, en cuanto a etiquetas y atributos HTML o CSS que los conforman, pero también incluir información descriptiva, de manera que usuarios con discapacidad visual puedan valerse de otros canales de percepción para acceder a los contenidos. Estos usuarios pueden recurrir a técnicas como el cambio de tamaño de las fuentes, combinaciones de colores con un alto contraste o apoyarse en el uso de software o dispositivos de hardware, por ejemplo, el uso de magnificadores de pantalla, ampliadores de imagen, sintetizadores de voz, grabadoras de sonido e incluso salidas en Braille, entre otras herramientas tiflotecnológicas.

Otra evaluación importante que se realizó en el marco de esta investigación se aplicó sobre la misma herramienta EAW. Debido a que también es una aplicación Web, era lógico y necesario verificar su grado de accesibilidad. Para ello se realizó una evaluación de pautas según los principios de accesibilidad (perceptible, comprensible, operable, robusto) sobre una versión espejo. Se detectaron algunos errores que fueron corregidos y se repitió la verificación hasta obtener como resultado un 100% de accesibilidad del recurso y un 0% de elementos con problemas, sobre 167 elementos evaluados que incluyeron etiquetas de títulos, imágenes, tablas, párrafos, hipervínculos, campos y nombres de campos. También se realizó la evaluación de EAW mediante el *W3C Markup Validation Service*, identificándose errores asociados a pautas que no están incluidas entre las verificaciones propias de la herramienta.

V. Resultados

El resultado principal es el desarrollo del Evaluador de Accesibilidad Web, herramienta que permite la verificación del nivel de accesibilidad de una página Web e indica los errores encontrados en los recursos evaluados en ella. Esta herramienta amplía el alcance de investigaciones previas del CENEAC, específicamente de la aplicación HEVAC, al incluir criterios relacionados con la discapacidad motriz y la validación de código en estilos CSS, además de los criterios relacionados con la discapacidad visual y la verificación de HTML. EAW es una aplicación Web, disponible en línea, de libre acceso, que no requiere un proceso de instalación.

En un nivel más detallado, EAW identifica y destaca en su reporte de resultados las sentencias con errores, indica la naturaleza del problema según el criterio incumplido clasificando por etiqueta y por principio de accesibilidad, suministra recomendaciones para corregir cada error y permite socializar las evaluaciones mediante los comentarios que se pueden agregar a cada evaluación realizada.

Para cada URL suministrado, se obtiene una evaluación de la página Web a la cual enlaza y de las páginas que conforman el primer nivel de hipervínculos disponibles. Estas evaluaciones están a disposición del usuario en un historial de páginas revisadas, con la posibilidad de eliminar un resultado o actualizar una evaluación sin necesidad de volver a registrarla.

VI. Conclusiones

En este estudio se utilizan los criterios para el desarrollo Web accesible y para la evaluación de accesibilidad desarrollados

por la W3C a través de la *Web Accessibility Initiative* (WAI), en especial las pautas del estándar WCAG 2.0. Sin embargo, otros lineamientos de importancia para un correcto diseño, usabilidad y accesibilidad de recursos informáticos se compilan en los estándares de la ISO, AENOR, y ANSI [28][29], entre otros, en la Sección 508, la Norma Requisitos de accesibilidad para contenidos en la Web (UNE 139803:2012) y en la *Accessible Rich Internet Applications* (WAI-ARIA).

El beneficio de herramientas como HEVAC y EAW tiene un alcance más amplio que el entorno de los desarrolladores de aplicaciones y sitios Web. Pueden ser usadas por personas encargadas de seleccionar, recomendar o evaluar recursos con fines educativos, informativos, acceso a gobierno electrónico, gestión en línea, servicios públicos y a plataformas en línea, en donde debe garantizarse que todas las personas puedan realizar trámites, consultar información o utilizar productos y servicios.

Indiscutiblemente la información proporcionada por medio del Internet debe estar a disposición de la mayor cantidad de personas, más aún cuando vivimos en una sociedad que debe promover la inclusión. Es por ello que no se puede descuidar o ignorar las necesidades de los usuarios que presentan alguna limitación o discapacidad.

El proceso de verificación que se realizó con HEVAC y con EAW dejó entrever que un gran número de páginas Web de instituciones de educación superior, gobierno y banca se enfocan en entregar un diseño de calidad, en algunos casos con excelente presentación de contenido gráfico y textual, pero descuidan los aspectos de accesibilidad, dificultando su navegación y uso, especialmente por parte de personas con discapacidad visual o motora.

Es fundamental que se haga del diseño accesible una norma, garantizando el acceso a la información y servicios, no agravando la brecha digital o infoexclusión. En palabras de Tim Berners-Lee, creador de la World Wide Web, el poder de la Web está en su universalidad, siendo un aspecto esencial garantizar el acceso de todos, independientemente de su condición o discapacidad (Figura 7).

> *"The power of the Web is in its **universality**. **Access** by everyone regardless of disability is an essential aspect".*
>
> *Sir Tim Berners-Lee.*

Figura 7: El Poder de la Web Reside en su Universalidad

REFERENCIAS

[1] WAI, *Web Accessibility Initiative,* Consorcio World Wide Web (W3C), http://www.w3c.es/Traducciones/es/WAI/intro/accessibility.

[2] OMS, *Discapacidades*, Organización Mundial de la Salud, http://www.who.int/topics/disabilities/es

[3] L. Valdez, *Discapacidad Visual*, Departamento de Educación Especial, Dirección Provincial de Educación del Guayas, Ecuador, 2011.

[4] M. Acosta, *Evaluador de Criterios de Accesibilidad Web para Pautas Relacionadas con Discapacidad Visual y Discapacidad Motora*, Trabajo Especial de Grado, Licenciatura en Computación, Universidad Central de Venezuela. 2014.

[5] W3C-WAI, *Introducción a la Accesibilidad Web*, http://www.w3c.es/Traducciones/es/WAI/intro/accessibility

[6] WCAG 1.0, *Web Content Accessibility Guidelines 1.0*, W3C-WAI, https://www.w3.org/WAI/intro/wcag10docs

[7] WCAG 2.0, *Web Content Accessibility Guidelines 2.0*, W3C-WAI, https://www.w3.org/TR/WCAG20

[8] S. Luján Mora, *Accesibilidad Web*, Universidad de Alicante, http://accesibilidadweb.dlsi.ua.es/?menu=pautas-accesibilidad-contenido-web

[9] WebAIM, *Constructing a POUR Website, Putting People at the Center of the Process*, http://webaim.org/articles/pour

[10] *Evaluating Accessibility*, W3C-WAI, https://www.w3.org/WAI/eval/Overview.html

[11] Y. Hassan y F. Martín, *Qué es la Accesibilidad Web*, No Solo Usabilidad, ISSN 1886-8592, 2013, http://www.nosolousabilidad.com/articulos/accesibilidad.htm

[12] A. Olalere and J. Lazar, *Accessibility of U.S. Federal Government Home Pages: Section 508 Compliance and Site Accessibility Statements,* Government Information Quarterly, vol. 28, no. 3, pp. 303-309, 2011.

[13] E. Jackson-Sanborn, K. Odess-Harnish, and N. Warren, *Website Accessibility: A Study of Six Genres*, Library Hi Tech, vol. 20, no. 3, pp. 308-317, 2002.

[14] E. Loiacono, S. McCoy, and W. Chin, *Federal Website Accessibility for People with Disabilities*, Information Technology Professional, vol. 7, no. 1, pp. 27-31, 2005.

[15] *Government-wide Section 508 Accessibility Program (GSA)*, https://www.section508.gov

[16] WebAIM, *Accessibility Evaluation Tools*, http://webaim.org/articles/tools

[17] WebAIM, *Web Accessibility Evaluation Tool*, http://wave.webaim.org

[18] TAW, *Test de Accesibilidad Web*, Fundación CTIC, España, http://www.tawdis.net

[19] HERA, *Hojas de Estilo para la Revisión de la Accesibilidad*, Fundación Sidar, España, http://www.sidar.org/hera

[20] R. Aguilera, *Desarrollo de una Herramienta para Verificación de Criterios de Accesibilidad en Sitios Web*, Trabajo Especial de Grado, Licenciatura en Computación, Universidad Central de Venezuela, 2012.

[21] PHP.net, *Biblioteca URL Cliente cURL*, The PHP Group, http://php.net/manual/es/book.curl.php

[22] W3Schools, *The HTML DOM Document Object*, https://www.w3schools.com/jsref/dom_obj_document.asp

[23] PHP.net, *Clase DOMDocument*, The PHP Group, http://php.net/manual/es/class.domdocument.php

[24] W3C, *Markup Validation Service*, https://validator.w3.org

[25] OMS, *Resumen Informe Mundial sobre la Discapacidad*, Organización Mundial de la Salud, 2011.

[26] NCSU, *Universal Design Resources*, Center for Universal Design, College of Design, North Carolina State University, https://www.ncsu.edu/ncsu/design/cud/about_ud/about_ud.htm

[27] AGESIC, *Gobierno en Red*, Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento, Uruguay Digital, https://www.agesic.gub.uy/innovaportal/v/168/1/agesic/principios-y-lineamientos-.html?idPadre=26

[28] C. Varela, A. Miñán, J. Hilera, F. Restrepo, H. Amado, M. Córdova y A. Villaverde, *Estándares y Legislación sobre Accesibilidad Web*, en las memorias del IV Congreso Internacional ATICA, Loja, Ecuador, pp. 46-53, 2012.

[29] *Normas de Accesibilidad*, Portal de Administración Electrónica, Ministerio de Hacienda y Función Pública, Secretaría General de Administración Digital, Gobierno de España, http://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Accesibilidad/pae_normativa/pae_eInclusion_Normas_Accesibilidad.html

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 2, pp. 21-33, Diciembre 2017
Selección de los Mejores Artículos de CoNCISa 2017

Lineamientos para el Despliegue de Redes SDN/OpenFlow

Gustavo Pereira [1], Eric Gamess [2]
gustavo.pereira.salas@gmail.com, egamess@jsu.edu

[1] Escuela de Computacion, Universidad Central de Venezuela, Caracas, Venezuela
[2] Department of Mathematical, Computing, and Information Sciences, Jacksonville State University, Jacksonville, AL, USA

Resumen: SDN (Software Defined Networking) es una arquitectura de red emergente que separa el plano de control del plano de datos de los dispositivos de la red y coloca el plano de control en uno o varios servidores de control capaz(ces) de gestionar las reglas de reenvío de tráfico de todos los dispositivos de comunicación bajo su(s) dominio(s). Este cambio de paradigma ofrece grandes beneficios en comparación con los métodos de red tradicionales debido a que el aprovisionamiento de políticas y servicios se efectúa en una sola entidad central, simplificando la administración, reduciendo los gastos de operación, acelerando el ciclo de innovación de nuevas tecnologías y ofreciendo la posibilidad de programar el comportamiento de la red. Esta investigación describe los principales componentes de una arquitectura SDN incluyendo componentes de hardware, software y protocolos y las consideraciones de diseño para el despliegue de redes SDN en el ámbito de campus empresariales. Las redes de campus empresariales están delimitadas a un conjunto de edificios o pisos de una edificación interconectados mediante redes Ethernet.

Palabras Clave: SDN; OpenFlow; Controladores SDN; Modelos de Despliegue SDN; Lineamientos Técnicos y Estratégicos; Fases para el Despliegue de Red SDN.

Abstract: SDN (Software Defined Networking) is an emerging network architecture that separates the control plane from the data plane of the network devices and places the control plane in one or several control servers that manage(s) the rules for traffic forwarding of all the communication devices under its/their domain(s). This new paradigm offers great benefits compared to traditional network methods, because the provision of policies and services is carried out in a single central entity, hence simplifying administration, reducing operating expenses, accelerating the innovation cycle of new technologies and offering the possibility of programming the behavior of the network. This research describes the main components of an SDN architecture including hardware components, software and protocols, and the design considerations for the deployment of SDN networks in business campuses. Business campus networks are limited to a set of buildings or floors of a building, interconnected by Ethernet networks.

Keywords: SDN; OpenFlow; SDN Controllers; SDN Deployment Models; Technical and Strategic Guidelines; Phases to Deploy SDN Networks.

I. Introducción

Un entorno de campus empresarial está conformado por un conjunto de edificios interconectados a través de medios de transmisión guiados y no guiados que enlazan los dispositivos de comunicación de la red y permiten el transporte del tráfico de los usuarios. En un campus empresarial tradicional el plano de control se encuentra distribuido en todos los dispositivos de la red y requiere la coordinación de los mismos para decidir cómo tratar los paquetes que ingresan a sus puertos. Esta naturaleza distribuida del plano de control obliga a los administradores de la red a configurar todos los dispositivos de red cada vez que se incorpora una nueva aplicación o política a la red, ralentizando el aprovisionamiento y la escalabilidad de servicios en ambientes con gran cantidad de dispositivos. Esta limitación ha impulsado a la comunidad científica a buscar nuevos enfoques y arquitecturas de redes alternativas que mejoren la agilidad, la flexibilidad y la escalabilidad en las redes de campus empresariales.

En un entorno SDN (Software Defined Networking) [1] uno o varios controladores se encargan de operar y gestionar el tráfico de la red desde un nodo lógico central. El controlador SDN instruye a los dispositivos SDN las reglas que se deben aplicar a los paquetes que circulan en la red en base a las políticas de una organización.

El despliegue de redes SDN requiere la aplicación de las mejores prácticas y principios de diseño que satisfagan los requerimientos de diseño de las organizaciones. Las mejores prácticas resultan de experiencias propias y de una investigación documental que abarca las siguientes categorías: (1) artículos académicos que describen la arquitectura de red SDN y el protocolo de red OpenFlow, (2) documentación de

implementaciones SDN en campus empresariales y (3) evaluaciones presentadas en artículos y notas técnicas.

II. Trabajos Relacionados

Los estudios relacionados son de carácter general y se encuentran algunos casos de estudio de implementaciones SDN en campus empresariales. La Universidad de Stanford [2][3] efectuó en el año 2010 una migración de parte de su red a una red SDN OpenFlow. El estudio de la migración a SDN en la Universidad de Stanford se divide en dos secciones: la primera sección muestra la red de arranque, las premisas de diseño y las herramientas de monitoreo utilizadas durante la migración. La segunda sección describe las fases de migración en la red de producción de los edificios considerados en la implementación.

Skorupa y Fabbi [4] describen los modelos de despliegue SDN existentes dependiendo de las características de la red de una organización.

La selección del tipo de controladores SDN y los dispositivos SDN para un despliegue se obtiene a partir de la revisión de las especificaciones de los productos abiertos y de carácter comercial disponibles por sus desarrolladores.

Los aspectos generales de diseño se encuentran en publicaciones y estudios [5][6] y textos de referencia [7][8] incluyendo principios de escalabilidad, alta disponibilidad, gestión y seguridad.

III. SDN

SDN [9] es un nuevo enfoque en la programación de redes que consiste en la capacidad de inicializar, controlar, cambiar y gestionar el comportamiento de reenvío del tráfico de una red mediante APIs abiertas. En una red SDN se separan los planos de control y datos de los dispositivos de red y se desplaza el plano de control a una unidad central llamada controlador SDN. Un controlador SDN se encarga de definir y comunicar las reglas de reenvío de tráfico a los dispositivos SDN y abstraer la infraestructura de red y su topología a las aplicaciones. Bajo este modelo las aplicaciones consideran a la red como un solo switch lógico central que provee servicios de conectividad a los usuarios y a las aplicaciones. Al separar los planos de datos y control, los switches de la red se convierten en dispositivos de reenvío simples y la lógica de control se implementa en un sistema operativo de red centralizado o NOS (Network Operating System). Con este nuevo enfoque se obtienen grandes beneficios: Aprovisionamiento de políticas simple, agilidad para reconfiguración e implementación de nuevos servicios y aceleración en la innovación de las redes [10].

A. *Arquitectura de Red SDN*

Una red SDN está conformada generalmente por tres capas: Capa de Aplicación, Capa de Plano de Control y Capa de Plano de Datos (ver Figura 1 tomada de [11]).

Capa de Aplicación: Está conformada por las aplicaciones del negocio y por las aplicaciones de servicios de red.

Capa del Plano de Control: Contiene los controladores SDN encargados de gobernar y dirigir la manera en que se transportan los datos en los dispositivos SDN. Se encuentra a cargo de todas las funciones complejas de enrutamiento, manejo de políticas, monitoreo y chequeos de seguridad de la red.

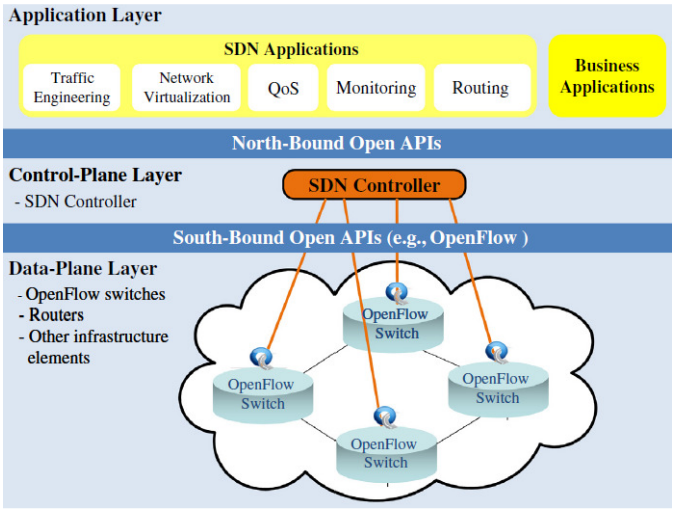

Figura 1: Representación Lógica de una Arquitectura de Red SDN

Capa del Plano de Datos: Está conformada por dispositivos SDN físicos y virtuales encargados de transportar datos en base a instrucciones recibidas por los controladores SDN de la red.

En una red SDN los controladores se comunican con las aplicaciones externas mediante APIs (Application Program Interface) Northbound abiertas y con los dispositivos SDN mediante APIs Southbound abiertas incluyendo el protocolo OpenFlow.

Una red SDN también puede ser vista como una composición de capas y sistemas (ver Figura 2 tomada de [12]). Algunas capas se encuentran presentes en todos los despliegues SDN incluyendo: controladores, infraestructura de red y APIs Southbound, mientras que otras son opcionales incluyendo APIs Northbound, aplicaciones, hipervisores y virtualización de redes.

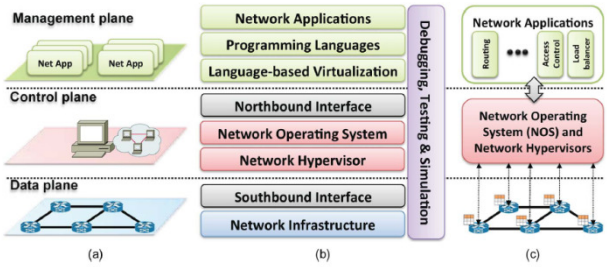

Figura 2: Arquitectura SDN basada en (a) Planos, (b) Capas y (c) Sistemas

Comenzando de abajo hacia arriba, se identifican los siguientes sistemas, planos y capas:

- Infraestructura de Red: Está conformada por los dispositivos SDN físicos o virtuales que manipulan y reenvían paquetes en base a reglas definidas por un controlador. Un dispositivo de red contiene un plano de datos (Forwarding Plane) encargado de transportar y manipular campos de cabecera de paquetes y un plano operacional (Operational Plane) encargado de efectuar tareas administrativas relacionadas con su funcionamiento.

Los dispositivos SDN pueden ser switches, routers o elementos de reenvío físicos o virtuales que soportan planos de datos y Soutbound SDN, como OpenFlow [2].

- APIs Southbounds: Constituyen el API de comunicación que facilita la comunicación entre controladores y dispositivos SDN. Estas APIs pueden ser abiertas o propietarias. El controlador SDN cuenta con las siguientes opciones de APIs Soutbound abiertas: OpenFlow [2], OVSDB [13] (Open vSwitch Database) y ForCES [14] (Forwarding and Control Element Separation). También existen las siguientes opciones de plugins: BGP [15] (Border Gateway Protocol), SNMP [16] (Simple Network Management Protocol) y NETCONF [17] (Network Configuration Protocol) entre otros. OpenFlow es el API Southbound estándar de la industria de las redes para entornos SDN. A través de OpenFlow un controlador puede crear, actualizar, modificar y eliminar entradas en las tablas de flujos de los dispositivos SDN y obtener información de estadísticas de estos.
- Plano de Datos (Forwarding Plane): El plano de datos se encuentra en los dispositivos de red y se encarga de la manipulación y del reenvío de los paquetes e incluye, pero no está limitado a filtros, medidores, marcadores, y clasificadores.
- Hipervisor de Red: En un ambiente de virtualización de servidores, un hipervisor es una plataforma de software que permite correr varias VMs sobre un mismo dispositivo de cómputo. Un hipervisor cuenta con switches virtuales, o vSwitches, que permiten la comunicación entre VMs. En una red SDN basada en vSwitches, un controlador le comunica a los vSwitches las reglas de reenvío de tráfico asociadas a la comunicación entre las VMs. Los vSwitches disponen de mecanismos de túneles para comunicarse con VMs hospedadas en otros hipervisores externos.
- Controlador SDN o NOS (Network Operating System): Es un software que corre en un servidor de red para gestionar las reglas de reenvío de tráfico de los dispositivos SDN. Un controlador SDN posee en su núcleo abstracciones, servicios esenciales y APIs comunes para comunicarse con el resto de los elementos de la arquitectura (ver Figura 3 tomada de [12]). Entre los principales servicios de red ofrecidos se encuentran: (1) gestión de la topología, encargado de descubrir la topología de la red, (2) gestión de estadísticas, encargado de recopilar información de contadores del tráfico de la red, (3) gestión de notificaciones, encargado de gestionar la comunicación del plano de control con los elementos de la red, (4) gestión de dispositivos, encargado de configurar y gestionar los elementos de la infraestructura de red, (5) reenvío de caminos más cortos, encargado de seleccionar los mejores caminos hacia los destinos y (6) mecanismos de seguridad, encargado de proveer mecanismos de protección a la red.
- Plano de Control: Es responsable de gestionar y dirigir el reenvío de tráfico de la red desde un nodo central.
- APIs Northbound: Presentan una interfaz común para el desarrollo de aplicaciones y permiten la interacción entre las aplicaciones externas y el controlador SDN. Actualmente los controladores de red ofrecen una variedad de APIs Northbound, como APIs RESTful [18][19], sistemas de archivos y lenguajes de programación.

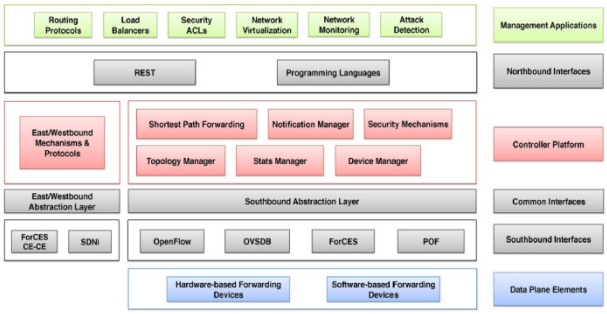

Figura 3: Componentes de un Controlador SDN

- APIs Eastbound/Westbound: Se utilizan en esquemas de controladores SDN distribuidos para permitir la comunicación entre los controladores y compartir información de alcanzabilidad y control. Entre las funciones de estas interfaces se encuentran: (1) importar o exportar datos entre controladores, (2) proveer algoritmos para modelos de consistencia de datos y (3) proveer capacidades de monitoreo y notificaciones. Para que exista compatibilidad e interoperabilidad entre diferentes controladores, se requiere disponer de APIs Eastbound/ Westbound estandarizadas, como SDNi [20], PCEP [21] y BGP [15].
- Virtualización de Red (Slicing): Consiste en la capacidad de compartir el plano de datos del hardware de red a múltiples redes lógicas, cada una con su propio direccionamiento y su propio mecanismo de reenvío. En una red SDN se puede virtualizar la capa de hardware de red en slices y asignar a cada slice recursos y espacios de direcciones. Un slice se define como una instancia de una red virtual, y dos redes virtuales distintas sobre el mismo hardware físico se conocen como slices. La virtualización de una red SDN se puede implementar siguiendo un esquema de virtualización mediante proxys o siguiendo un esquema de virtualización basado en lenguajes. La virtualización de redes SDN mediante proxys se logra con la colocación de un controlador SDN especial entre los controladores y los dispositivos SDN de la red. El proxy actúa como un multiplexor de tráfico y como un gestor de la asignación de recursos a slices independientes. En la actualidad se encuentran los siguientes virtualizadores de red basados en proxies: FlowVisor [22], OpenVirteX [23], y AutoSlice [24]. FlowVisor [22] es un controlador OpenFlow de propósito especial que actúa como un proxy transparente entre switches OpenFlow y controladores OpenFlow ofreciendo una capa de virtualización de red basada en OpenFlow. En FlowVisor los slices pueden ser definidos por una combinación de puertos de switches Capa 1, direcciones Ethernet origen/destino Capa 2, direcciones IP origen/destino de Capa 3 y puertos TCP/UDP origen/destino o códigos/tipos ICMP Capa 4. En la Figura 4 se muestra una topología de red conformada por tres controladores OpenFlow, un controlador FlowVisor y seis switches OpenFlow. FlowVisor recibe todos los comandos OpenFlow de los controladores hacia los switches OpenFlow y las

respuestas y notificaciones de estadísticas de los switches hacia los controladores respectivos garantizando el aislamiento de tráfico entre cada slice definido. En el ejemplo se puede observar como cada controlador tiene una vista particular y diferente de la misma red física. A través de FlowVisor un grupo de investigadores puede crear sus propias instancias de red lógicas corriendo sus propios protocolos de enrutamiento y ejecutarlas sobre una red real en paralelo con una red de producción y mantener el aislamiento y las velocidades de reenvío del hardware existente. La virtualización basada en lenguajes es el uso de lenguajes de propósito especial para virtualización incluyendo Pyretic [30] y Splendid [25].

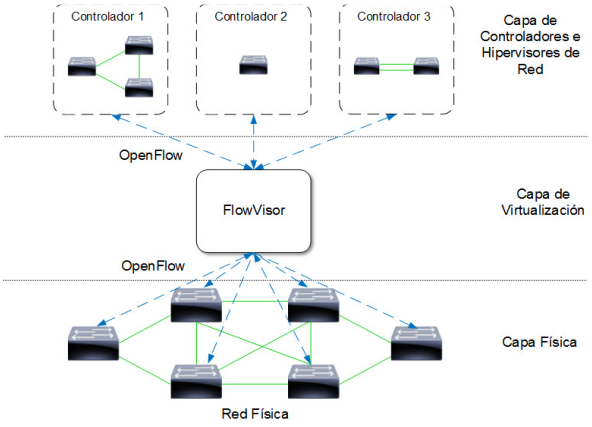

Figura 4: Topología de Red OpenFlow con FlowVisor

- Lenguajes de Programación: Se encuentran los lenguajes que pueden interactuar con el controlador SDN. Entre los más utilizados se encuentran: Python y Java y lenguajes de programación específicos para SDN, como Procera [26], FML [27] (Flow Based Management Language), Frenetic [28], NetCore [29] y Pyretic [30].
- Aplicaciones: Constituyen las aplicaciones del negocio de las organizaciones y aplicaciones externas de propósito especial como balanceadores de carga, sistemas de monitoreo y aplicaciones de QoS.
- Plano de Gestión: El plano de gestión es responsable de las funciones de monitoreo, configuración y mantenimiento de los dispositivos de red. En SDN la gestión de la red se implementa mediante una jerarquía de capas de protocolos, modelos y datos (ver Figura 5 tomada de [32]). Los protocolos de gestión utilizan un lenguaje de modelo de datos como YANG [33] (Yet Another Next Generation) para configurar y obtener datos del estado de los dispositivos. Entre los protocolos de gestión de red SDN se encuentran NETCONF [17] , RESTCONF [34], gRPC [35] y OF-CONFIG [36] (OpenFlow Management and Configuration Protocol).

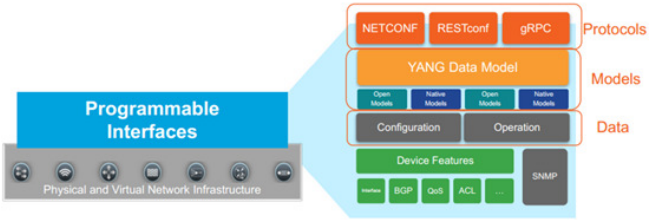

Figura 5: Arquitectura de Gestión basada en Modelos

- NETCONF es un protocolo de configuración de red estandarizado por el IETF en el RFC 6241 [17] que provee mecanismos simples para instalar, manipular, y eliminar la configuración de dispositivos de red. Sigue un paradigma solicitud-respuesta RPC (Remote Procedure Call) con codificación XML y transporte seguro SSH (Secure Shell). NETCONF define almacenes de datos que contienen la configuración de los dispositivos de red y operaciones CRUD (Create, Read, Update, Delete) que permiten recuperar, configurar, copiar y eliminar almacenes de datos. RESTCONF es un protocolo de configuración de red basado en HTTP estandarizado por el IETF en el RFC 8040 [34]. A través de RESTCONF las aplicaciones Web pueden acceder y modificar la configuración de un dispositivo de red siguiendo una arquitectura cliente/servidor basada en RPC con clientes y servidores RESTCONF. gRPC [35] es un framework para llamadas a procedimientos remotos RPC de código abierto y de alto desempeño liberado por la empresa Google para construir sistemas distribuidos masivos. OF-CONFIG [36] es un protocolo de gestión desarrollado por la ONF (Open Networking Foundation) para gestionar switches OpenFlow físicos y virtuales. Utiliza NETCONF para la administración, XML para la codificación y SSH para el transporte.

B. Modelos de Despliegue SDN

En la práctica se utilizan tres modelos de despliegue SDN [4]: (1) Modelo SDN basado en Dispositivos, (2) Modelo SDN Overlay, y (3) Modelo SDN Híbrido. El Modelo SDN basado en Dispositivos [4] se refiere a una red de switches físicos SDN que operan solo bajo las instrucciones de un controlador SDN. Se implementa con rapidez en despliegues nuevos como un nuevo complejo de oficinas dentro de un campus. En la Figura 6 se muestra un despliegue SDN con 6 switches dirigidos por las instrucciones de un controlador SDN. El modelo SDN Overlay [4] está basado en la superposición de redes sobre una infraestructura de red física subyacente. En una red SDN Overlay, los nodos finales SDN son dispositivos virtuales que forman parte de hipervisores en un ambiente de virtualización de servidores.

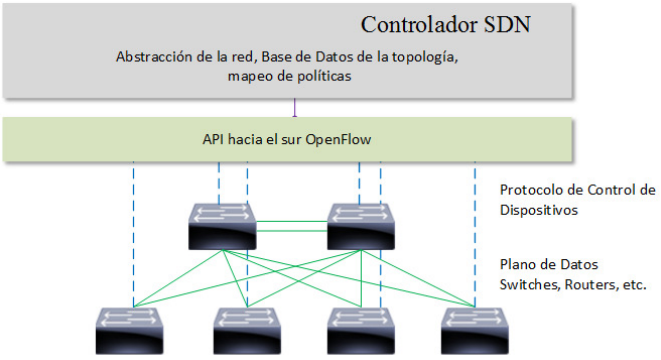

Figura 6: Ejemplo del Modelo SDN basado en Dispositivos

En este escenario el controlador controla el reenvío del tráfico de los switches lógicos que se encuentran definidos en los hipervisores y no altera la red física actual ni el plano de control distribuido de la red subyacente. Para crear la red virtual, los nodos SDN lógicos establecen túneles overlay entre sí a través de un protocolo de túneles: (1) VXLAN [37]

(Virtual Extensible LAN), (2) NVGRE [38] (Network Virtualization using Generic Routing Encapsulation), o (3) STT [39] (Stateless Transport Tunneling). Los túneles overlay usualmente terminan en los switches virtuales dentro de los hipervisores o en dispositivos físicos que actúan como gateways hacia la red existente. En la Figura 7 se muestra un despliegue SDN Overlay conformado por un controlador SDN, tres vSwitches hospedados en tres hipervisores y una red tradicional subyacente de cinco switches.

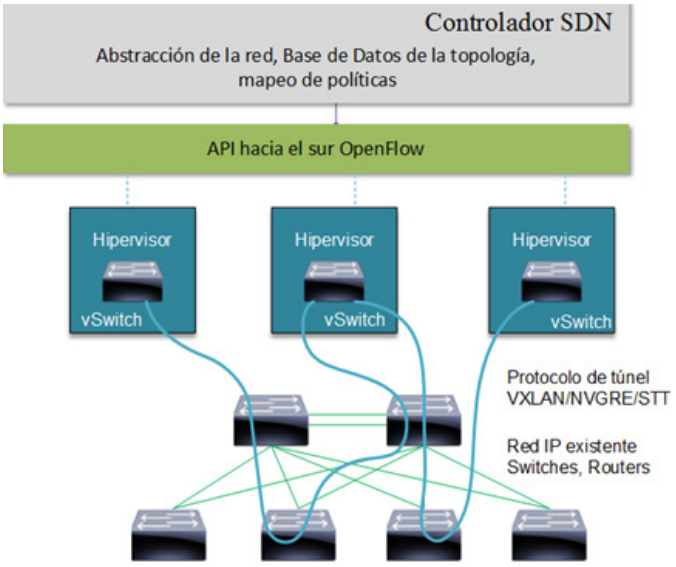

Figura 7: Modelo SDN Overlay

La red virtual consiste de switches lógicos interconectados por enlaces virtuales punto-a-punto. El controlador SDN se apoya en el protocolo hacia el sur OpenFlow para aprovisionar las reglas de reenvío a las tablas de flujos de los vSwitches presentes en los hipervisores de las VMs. El modelo se utiliza en escenarios donde se quiere hacer una implementación rápida de una solución SDN montada sobre una red IP existente. La limitante del modelo es la pérdida de visibilidad del tráfico en la red subyacente que restringe la aplicación de servicios diferenciados e ingeniería de tráfico y la complejidad de gestionar y operar ambas redes.

C. Modelo SDN Híbrido

El modelo SDN Híbrido está basado en la convivencia de tecnologías de redes tradicionales con tecnologías de red SDN en un mismo entorno. En este caso un gateway SDN corre tanto el modelo SDN Overlay, como el modelo SDN Basado en Dispositivos para esquemas híbridos de modelos SDN basado en dispositivos con modelos SDN Overlay. También existe la opción de disponer de un gateway SDN que corre protocolos legados tradicionales y protocolos SDN/OpenFlow. El gateway en un esquema híbrido se comunica con el controlador SDN OpenFlow y con los dispositivos de red Ethernet tradicionales y corre bajo ambos esquemas de red y protocolos.

IV. OpenFlow

OpenFlow [40] es la primera interfaz estandarizada diseñada específicamente para SDN, brindando alto desempeño, y control de tráfico granular en dispositivos de múltiples fabricantes. OpenFlow es un protocolo SDN abierto que se utiliza para controlar el plano de datos de los dispositivos SDN desde un nodo central. Constituye la primera interfaz de comunicación estándar definida entre los planos de datos y control de una arquitectura SDN y permite el acceso directo y la manipulación del plano de datos de routers y switches, tanto físicos, como virtuales. El protocolo se implementa en ambos lados de la interfaz entre los dispositivos de la infraestructura de red y el software de control SDN. Utiliza el concepto de flujos para identificar el tráfico de la red basado en reglas estáticas predefinidas o en reglas programadas dinámicamente por el software de control SDN.

A. Arquitectura OpenFlow

La arquitectura de red OpenFlow consiste de tres conceptos básicos: (1) la red está soportada por switches con capacidad OpenFlow que conforman el plano de datos, (2) el plano de control consiste de uno o más controladores OpenFlow que definen y publican las reglas de reenvío al plano de datos de los switches OpenFlow y (3) el controlador y los switches OpenFlow se comunican entre sí mediante un canal de control seguro (ver Figura 8 tomada de [40]). En una arquitectura OpenFlow, el reenvío de los datos se efectúa en los switches de la red, y las decisiones de reenvío se hacen en un programa de software de un controlador externo implementado en un servidor que se comunica con los switches a través del protocolo OpenFlow. A través del protocolo OpenFlow, un controlador puede añadir, actualizar, y eliminar entradas de flujos en las tablas de flujos, de manera proactiva o de manera reactiva en respuesta a la llegada de paquetes, modificando el comportamiento de reenvío del plano de datos de los switches.

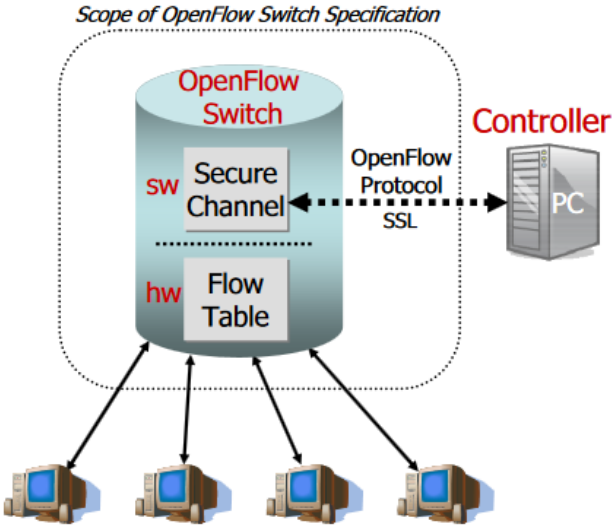

Figura 8: Arquitectura de Red OpenFlow

Dentro de un dispositivo OpenFlow, cuando un paquete ingresa a alguno de sus puertos se inicia un proceso de consulta en la tabla de flujos para encontrar una entrada de flujos que concuerde con el paquete. Las entradas de flujos son evaluadas en orden de prioridad y se utiliza la primera concordancia en la tabla. Si existe una concordancia, se ejecutan las acciones indicadas en la entrada de flujo y se actualizan los contadores de paquetes para la entrada de flujos correspondiente. En caso de no existir concordancia en alguna entrada de la tabla de flujos, el paquete se envía al controlador sobre el canal OpenFlow o se descarta el paquete.

Entre las instrucciones asociadas con cada entrada de flujo se encuentran: (1) reenviar el paquete a un puerto, (2) modificar los campos de cabecera del paquete y (3) descartar el paquete. Las entradas de flujos pudieran indicar enviar un paquete a un puerto físico, o a un puerto virtual definido por el switch, o a un puerto virtual reservado por la especificación OpenFlow. Los puertos virtuales reservados pudieran especificar acciones de reenvío genéricas como reenvío a un controlador, hacer una inundación de puertos, o reenviar los paquetes usando métodos no-OpenFlow, como el procesamiento de un switch Ethernet tradicional. Los puertos virtuales definidos por el switch permiten especificar grupos de agregación de enlaces, túneles o interfaces de loopback.

B. *Ejemplo de Tabla de Flujos OpenFlow*

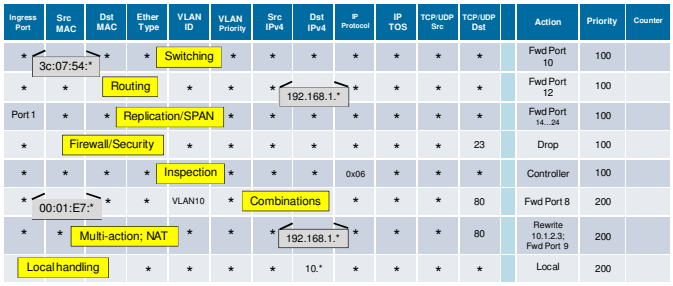

Figura 9: Ejemplo de Tabla de Flujos OpenFlow

En la Figura 9 tomada de [41] se presenta un ejemplo de una tabla de flujos de un switch OpenFlow v1.0.

En este ejemplo, se presentan las siguientes reglas para todos los paquetes que ingresan al switch:

- Switching: Todos los paquetes cuya dirección MAC origen sea 3c:07:54:* son enviados al puerto 10.
- Routing: Todos los paquetes cuya dirección IP destino sea 192.168.1.* son enviados al puerto 12.
- Replicación/SPAN: Todos los paquetes que ingresen por el puerto 1 son replicados en los puertos que van desde el puerto 14 hasta el puerto 24.
- Firewall/Seguridad: Todos los paquetes con número de puerto TCP/UDP destino 23 son descartados.
- Inspección: Todos los paquetes con protocolo IP 0x06, que corresponden a segmentos TCP, son reenviados al controlador para análisis ulterior.
- Combinaciones: Todos los paquetes cuya dirección MAC origen sea 00:01:E7:* y que pertenezcan a la VLAN 10 y que tengan puerto TCP/UDP destino 80 son enviados al puerto 8.
- Multi-acción/NAT: Todos los paquetes cuya dirección IP destino sea 192.168.1.* y con puerto TCP/UDP destino 80 tendrán nueva dirección IP origen 10.1.2.3 y son enviados al puerto 9.
- Manejo local: Todos los paquetes cuya dirección IP destino sea 10.*.*.* son manejados localmente.

C. *Tipos de Switches OpenFlow*

Existen dos categorías de switches OpenFlow: switches Solo-OpenFlow (OpenFlow-only), cuyo procesamiento de paquetes está basado únicamente en el protocolo OpenFlow y switches OpenFlow-híbrido (OpenFlow-hybrid), que soportan tanto la operación OpenFlow como la operación de switching Ethernet tradicional.

D. *Protocolo OpenFlow*

El protocolo OpenFlow describe el intercambio de mensajes que tiene lugar entre un controlador OpenFlow y un switch OpenFlow. Generalmente el protocolo es implementado sobre SSL (Secure Socket Layer) o TLS (Transport Layer Security), para proveer un canal OpenFlow seguro o sobre un canal TCP. Existen tres tipos de mensajes entre el controlador y los switches OpenFlow: (1) Controller to Switch, son mensajes enviados por el controlador a un switch para solicitar notificaciones de capacidades, estados de los puertos y estadísticas de paquetes, o para modificar el estado del reenvío del switch, como añadir, eliminar o modificar entradas en las tablas de flujos, (2) Asynchronous, son mensajes enviados de manera asíncrona por los switches al controlador para denotar la llegada de un nuevo paquete, notificar el cambio de estado en el switch, o informar la ocurrencia de algún error y (3) Symmetric, son mensajes enviados en cualquier dirección sin una solicitud expresa, como los mensajes Hello que notifican la disponibilidad de un dispositivo y los mensajes Echo Request y Echo Reply que pueden ser utilizados para medir la latencia en el canal de comunicación entre el controlador y el switch.

E. *Especificaciones OpenFlow*

OpenFlow ha evolucionado desde su aparición a finales de 2007 hasta la actualidad. En la siguiente sección se describen las características resaltantes de cada versión mayor:

OpenFlow 1.0.0 [42]: Especificación donde cada switch tiene una sola tabla de flujos. La tabla de flujos consta de tres grupos de campos: Header Fields, Counters y Actions. El grupo Header Fields contiene 12 campos: Ingress Port o puerto de ingreso del paquete, Ethernet Source Address o dirección Ethernet origen del paquete, Ethernet Destination Address o dirección Ethernet destino del paquete, Ethernet Type que indica cual es el protocolo encapsulado en el payload del frame Ethernet, VLAN ID para la VLAN del paquete, VLAN priority CoS que es un campo de prioridad de 3 bits, IPv4 Source Address o dirección IPv4 origen del paquete, IPv4 Destination Address o dirección IPv4 destino del paquete, IPv4 Protocol Number o número de protocolo IPv4, IP ToS o campo Type of Service de IPv4, TCP/UDP Source o número de puerto TCP/UDP origen o tipo ICMP (Internet Control Message Protocol), TCP/UDP Destination o número de puerto TCP/UDP destino o tipo ICMP.

OpenFlow 1.1.0 [43]: Modifica la nomenclatura de los campos a Match Fields, Counters e Instrucctions. Amplía los campos de cabecera incluyendo etiquetas para el soporte de MPLS. Incorpora múltiples tablas de flujos e implementa un mecanismo pipeline OpenFlow, en el cual un paquete atraviesa varias tablas de flujos en serie y recibe un procesamiento particular en cada tabla. Incluye una tabla de grupos para brindar funcionalidad común a un conjunto de paquetes. Permite disminuir el campo TTL (Time-To-Live) en la cabecera IP (ver Figura 10).

OpenFlow 1.2.0 [44]: Incluye campos para el soporte de IPv6. Soporta el análisis de campos de paquetes adicionales a través de una estructura TLV (Type-Length-Value), referenciada como OXM (OpenFlow Extensible Match). Soporta canales redundantes desde un switch a múltiples controladores.

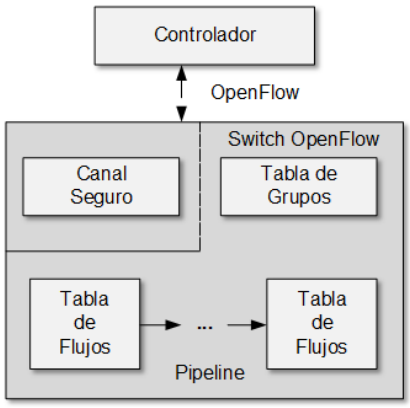

Figura 10: Componentes de un Switch OpenFlow 1.1.0

OpenFlow 1.3.0 [45]: Introduce nuevas funcionalidades para el soporte de funciones administrativas OAM (Operations Administration and Management) e incorpora la tabla especial de medidores al plano de datos del switch que recopilan estadísticas particulares como la tasa de bytes o paquetes de un flujo particular. Otra mejora importante es el soporte extendido de múltiples controladores auxiliares, permitiendo implementar mecanismos de balanceo de cargas hacia los controladores.

OpenFlow 1.4.0 [46]: Agrega estructuras TLV para puertos, tablas y colas. Incluye el soporte de puertos ópticos. Adicionalmente, los controladores soportan el envío de mensajes en lotes.

OpenFlow 1.5.0 [47]: Introduce el mecanismo de Tablas de Egreso permitiendo que el procesamiento se realice en el contexto de puertos de salida.

V. CONTROLADORES SDN

En esta sección se analizan 3 controladores de código abierto disponibles en la industria: OpenDaylight [48], Floodligth [49] y Ryu [50]. Los criterios de selección de los controladores utilizados fueron: (1) tiempo en el mercado mayor a 3 años, (2) foco activo en su desarrollo, (3) soporte del API OpenFlow 1.3 o superior y (4) disponibilidad de documentación y soporte en línea con FAQs (Frequently Asked Questions).

A. OpenDaylight

La Tabla I resume las características principales del controlador OpenDaylight.

Tabla I: Caracteristicas de OpenDaylight

Concepto
Controlador de código abierto que sigue una arquitectura modular e incluye su propia máquina virtual Java, lo cual le permite ser desplegado en cualquier plataforma de hardware y software con soporte Java.
Software
Última versión y fecha de liberación: Sexta versión denominada Carbon, mayo, 2017. Lenguaje de Núcleo: Está basado en un framework modular basado en Java 1.7 y soporta el framework de programación OSGI (Open Specifications Group Initiative). APIs Northbound: Soporta REST bidireccional. Servicios y Orquestación de Aplicaciones de Red: Incluye aplicaciones del negocio y aplicaciones de control, aprovisionamiento y gestión de la red, aplicaciones de nube, aplicaciones de centros de datos, funciones de red y servicios de virtualización. Plataforma de Controladores: Incluye los siguientes módulos: gestión de topología (Topology Manager), gestión de estadísticas (Statistics Manager), gestión de switches (Switch Manager), gestión de reglas de reenvío (Forwarding Rules Manager) y rastreador de hosts (Host Tracker). Plugins de protocolos y APIs Southbound: OpenFlow 1.0, OpenFlow 1.3, OF-CONFIG, NETCONF, LISP, OVSDB, BGP, CAPWAP y otros. Soporte de Comunicación C2C (Controlador-a-Controlador): Incluye la aplicación ODL-SDNi para proveer el establecimiento de la interfaz Este-Oeste (SDNi communication) entre múltiples controladores OpenDaylight.
Usabilidad
Documentación de instalación y documentación operativa disponible en la wiki del proyecto.
Implementación
El controlador OpenDaylight corre sobre una máquina virtual Java. Para mejores resultados, se recomienda una distribución Linux y Java 1.7.
Funcionalidades
Soporta OpenFlow 1.0 y 1.3. Soporta el framework OSGi y Java como lenguaje de desarrollo y soporta REST bidireccional empleando JSON y API DOM empleando XML. Incluye aplicaciones: OpenDOVE y VTN para implementar virtualización de red overlay. Integración con OpenStack a través de un API Neutron.
Licenciamiento
Licencia: Eclipse Public License (EPL-1.0).
Impacto Potencial
Popularidad: Alta. Tiene fuerte soporte de la industria y de la comunidad de código abierto.

B. Floodlight

La Tabla II resume las características principales del controlador Floodlight.

Tabla II: Caracteristicas de Floodlight

Concepto
Controlador OpenFlow extensible basado en Java con licencia Apache desarrollado por una comunidad abierta de programadores y patrocinado por Big Switch Networks y un conjunto de aplicaciones construidas en el tope del controlador.
Software
Última versión y fecha de liberación: Floodlight v1.2, febrero. 2016. Lenguaje de Núcleo: Java 1.7. APIs Northbound: REST con formato de datos JSON. Aplicaciones: Circuit Pusher, utiliza APIs REST Floodlight para crear un circuito bidireccional entre 2 puntos terminales IP. OpenStack Quantum Plugin, permite que el controlador Floodlight corra como una red de backend para el sistema operativo de nubes OpenStack utilizando el plugin Neutron. Virtual Network Filter, módulo para virtualizar redes basadas en direcciones MAC capa 2. Learning Switch, switch de aprendizaje capa 2 común. Firewall, implementa reglas ACL a switches OpenFlow. Hub, aplicación que siempre inunda un paquete de ingreso a todos los puertos activos de un switch. Load Balancer, balancea cargas de flujos TCP, UDP y ping. Servicios/Funciones del Core: Inventario de dispositivos (hosts). Administrador de la topología. Módulo para insertar flujos y grupos en la red OpenFlow (Static Entry Pusher). Monitoreo del desempeño del controlador. Capacidad de reenvío Dijkstra que permite interconectar islas de switches OpenFlow con switches no-OpenFlow. APIs Southbound: Solamente OpenFlow desde la v1.0 hasta la v1.4. Soporte de Comunicación C2C: Floodlight no provee soporte para ningún tipo de comunicación C2C, por lo cual no soporta alta disponibilidad, ni interfaces este-oeste, ni controladores jerárquicos.
Usabilidad
Documentación de la instalación y documentación operativa disponible.
Implementación
Se recomienda una distribución Linux Ubuntu 14.0.4 TLS Trusty Tahr o Ubuntu 16.04.1 LTS Xenial Xerus. Para instalar Floodlight en Linux se requiere un cliente Git, Python y la herramienta Apache Ant.
Funcionalidades
Soporta OpenFlow v1.0 a v1.4. Se integra con sistemas de gestión de nube a través del módulo Virtual Network Filter (VNF), el cual provee un API REST para integrarse con Quantum/OpenStack. Se integra con aplicaciones externas a través de APIs REST.
Licenciamiento
Licencia: Apache 2.0.
Impacto Potencial
Uso Comercial: Representa el núcleo del controlador comercial Big Switch Network Controller.

C. Ryu

La Tabla III resume las características principales del controlador Ryu.

Tabla III: Caracteristicas de Ryu

Concepto
Ryu es un framework SDN basado en componentes con APIs bien definidas que facilitan la creación de aplicaciones de control y gestión de redes.
Software
Última versión y fecha de liberación: Ryu v4.8, noviembre. 2016. Lenguaje de Núcleo: Python 2.6+, greenlets, pseudo-multihilo cooperativo (utilizando un sólo núcleo). Interfaces/Lenguaje de Aplicaciones: Python 2.6+ para las aplicaciones del controlador y REST para las aplicaciones externas. Arquitectura de Software: El framework SDN Ryu está conformado por un conjunto de componentes, librerías e interfaces para el intercambio de información. La capa de aplicaciones contempla las siguientes aplicaciones de red: switch capa 2, firewall, IDS (Snort), abstracciones de túneles GRE, VRRP (Virtual Router Redundancy Protocol) y los servicios de descubrimiento de la topología y manejo de estadísticas entre otros. La capa de control incluye el soporte de protocolos hacia el sur y los siguientes componentes principales: manejador de eventos, analizador de mensajes OpenFlow, gestor de memoria, gestor de aplicaciones, servicios de infraestructura y un conjunto de librerías que incluyen NETCONF, sFlow y NetFlow. Soporte de Comunicación C2C: Es capaz de soportar servicios de alta disponibilidad a través del componente Zookeeper.
Usabilidad
Documentación de la instalación y documentación operativa disponible.
Implementación
Sistema Operativo Soportado: Distribución GNU/Linux reciente. Dependencias en Tiempo de Ejecución: Paquetes Python (python-eventlet, python-routes, python-webob y python-paramiko).
Funcionalidades
Versión OpenFlow Soportada: 1.0, 1.2, 1.3, 1.4, 1.5 y extensiones Nicira. APIs Southbound: NETCONF, OF-CONFIG, SNMP, OVSDB y OpenFlow. Integración con Sistemas de Gestión de Nube: Incluye el plugin OpenStack. Utiliza APIs REST. Servicios/Funciones del Core: Incluye analizadores y generadores de paquetes de diferentes protocolos de red. Incluye un plugin para integrarse al controlador de redes de nube OpenStack Neutron que soporta túneles overlay basados en GRE y configuraciones de VLANs.
Licenciamiento
Licencia: Apache 2.0.
Impacto Potencial
Popularidad: Es utilizado principalmente en campos de investigación debido a que es fácil de aprender y fácil de expandir. Uso comercial: Ha tenido poco uso comercial debido en parte a que varias evaluaciones de desempeño [51][52] han reportado lentitud del controlador en comparación con otros controladores.

La selección de un controlador para un despliegue SDN depende de factores como el tamaño de la red, el costo de adquisición, el desempeño de la red requerido por las aplicaciones, la confiabilidad, la facilidad de gestión, el nivel de escalabilidad y las APIs soportadas. Para ambientes de escala grande y mediana se puede optar por un conjunto de controladores OpenDaylight en alta disponibilidad en base a su robustez, escalabilidad y amplio soporte. Para ambientes de escala pequeña se puede considerar una implementación basada en un controlador Floodlight y para entornos de experimentación y desarrollo de nuevas tecnologías se pueden utilizar controladores Ryu.

En la industria existen alternativas de controladores SDN/OpenFlow comerciales incluyendo: (1) Big Network Controller [53], basado en el controlador Floodlight, (2) Brocade SDN Controller [54], distribución comercial de OpenDaylight por Brocade, (3) Cisco Open SDN Controller [55], distribución comercial de OpenDaylight por Cisco Systems, (4) Aruba VAN SDN Controller [56] y (5) NEC ProgrammableFlow PF6800 [57]. La selección del controlador comercial dependerá de factores como el costo de capital, el servicio técnico, la documentación, el apoyo durante la migración o implementación de la solución, la facilidad de la gestión, los APIs soportados para interactuar con el controlador y las aplicaciones embebidas en las propuestas de cada fabricante.

VI. Fases para el Despliegue de SDN

En esta sección se describen las fases para desplegar redes SDN basado en las mejores prácticas y recomendaciones recopiladas de los casos de estudio de SDN descritos por el Grupo de Trabajo para Migración hacia SDN de la ONF, específicamente considerando las pautas relacionadas con la migración hacia SDN en el campus de la Universidad de Stanford [2][3].

La Universidad de Stanford migró dos edificios de su campus a SDN en el año 2010 [2]. La meta fue promulgar la tecnología SDN para facilitar la experimentación en redes de producción, comprender y verificar la tecnología SDN y contribuir con el desarrollo del protocolo OpenFlow. La implementación estuvo basada en un despliegue SDN Híbrido con switches OpenFlow híbridos de múltiples fabricantes con asignación de una VLAN OpenFlow y una VLAN legada. También se consideraron Access Points con switches de software de referencia OpenFlow basados en Linux. El plano de control se hizo con controladores abiertos disponibles para la fecha y se virtualizó la red con FlowVisor en slices para tráfico de datos de experimentación y slices para tráfico de datos de producción. Para garantizar el éxito de la migración se implementó una infraestructura de monitoreo que recopiló estadísticas del plano de control antes, durante y después de la migración.

En general un despliegue SDN consta de tres fases importantes: (1) Preparación, (2) Implementación y (3) Validación.

A. Preparación

La fase de preparación contempla (1) el análisis de los requerimientos del negocio, (2) el diseño de alto nivel de la solución, (3) la preparación de las pruebas de concepto, (4) el diseño de bajo nivel de la solución y (4) la planificación de la implementación.

Análisis de Requerimientos: Esta etapa consiste en la identificación de las metas y objetivos del negocio que se deben alcanzar con la nueva red y en el reconocimiento del estado actual de la red. Particularmente se debe recopilar y analizar la siguiente información: (1) metas del negocio, (2) caracterización de las aplicaciones del negocio y los servicios de red incluyendo tipo de aplicación, volumen, prioridades, requerimientos de ancho de banda y reglas de acceso, (3) topología de red incluyendo usuarios, sistemas, servidores, equipos de red, tipo y velocidad de enlaces de conexión, (4) identificación de segmentos de red incluyendo definiciones de VLANs, direccionamiento IP y protocolos de red, (5) identificación de políticas del negocio, políticas de seguridad, políticas de calidad de servicio de la red y normas de cumplimiento como regulaciones y (6) identificación de los sistemas de gestión de la red.

La evaluación de la red se puede hacer levantando la información con informes del cliente y/o corriendo herramientas de monitoreo y análisis de tráfico que permitan identificar métricas de utilización de la red identificando las aplicaciones, usuarios y servicios con mayor uso y volumen de solicitudes. Esta etapa también contempla la identificación de riesgos, cuellos de botella de la red o problemas de desempeño.

Diseño de Alto Nivel: Muestra un diagrama de alto nivel de la solución identificando los componentes de la arquitectura de red SDN de la solución incluyendo controladores, dispositivos de red, aplicaciones y sistemas de gestión. La selección de los controladores y los dispositivos de red se realizan en base a los resultados del análisis de los requerimientos y considera aspectos como costos, facilidad de uso, desempeño, soporte y APIs soportadas. Se debe seleccionar el tipo de switch OpenFlow solo-OpenFlow/Híbrido o virtual, dependiendo del modelo de despliegue de la solución seleccionado.

Pruebas de Concepto: Consiste en la preparación de un banco de pruebas para determinar si la solución a implementar satisface las metas del negocio. La implementación del banco de pruebas se puede realizar mediante la utilización de herramientas de simulación SDN de código libre de prototipado rápido incluyendo las herramientas Mininet [58], VT-Mininet [59] y OFNet [60] o herramientas comerciales incluyendo EstiNet [61]. La herramienta OFNet permite hacer emulaciones de red gráficas y provee capacidades de depuración visual, monitoreo de desempeño y capacidades de generación de tráfico sintético que facilitan la depuración de la solución y la optimización del diseño de la solución.

Diseño de Bajo Nivel: Consiste en la definición de las reglas de control de tráfico que serán implementadas en el controlador SDN/OpenFlow en base a los requerimientos y las políticas del negocio.

Planificación: Se establece la línea de tiempo de las fases de ejecución del despliegue SDN y los entregables e informes de cada fase.

B. Implementación

La fase de implementación consiste en la migración de la red legada, los usuarios y servicios a la red SDN/OpenFlow de la solución. El esquema de migración dependerá del análisis de requerimientos donde se determinará el modelo de despliegue SDN. En el caso de una red nueva totalmente OpenFlow, se implementa el modelo SDN basado en dispositivos. En el caso de escenarios de migración de red legada a SDN/OpenFlow, se implementa un modelo SDN Híbrido.

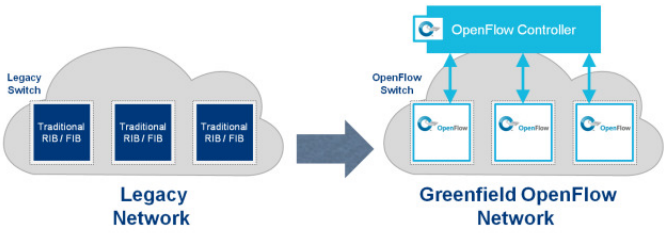

Figura 11: Despliegue Greenfield

Despliegue Greenfield: La ONF denota al modelo SDN basado en Dispositivos con el nombre de despliegue Greenfield (ver Figura 11). En este caso la red existente es actualizada para convertirse en OpenFlow y el control de los dispositivos de la red es reemplazado con un esquema de control de tráfico a través del controlador SDN. Los dispositivos SDN pueden ser dispositivos que aceptan un plugin a los cuales se le instala un agente OpenFlow o switches OpenFlow nativos. El despliegue de esta red consiste en la conexión de red entre el controlador y los dispositivos de red. A partir de este momento el controlador se encargará de hacer un descubrimiento de la red y publicará las reglas de las tablas de flujos a los dispositivos de red. Posteriormente se definen las reglas de filtrado y conectividad específicas en base a las políticas de seguridad, calidad de servicio y políticas de red recopiladas durante la fase de preparación.

Despliegue Mixto (Mixed): La ONF denota el modelo SDN Híbrido con el nombre Despliegue Mixto. Este enfoque considera que se desplegarán nuevos dispositivos OpenFlow los cuales coexistirán con los dispositivos legados que operan bajo protocolos tradicionales distribuidos. En este caso el controlador SDN/OpenFlow y los dispositivos tradicionales intercambiaran información de enrutamiento entre sí a través de protocolos de red tradicionales.

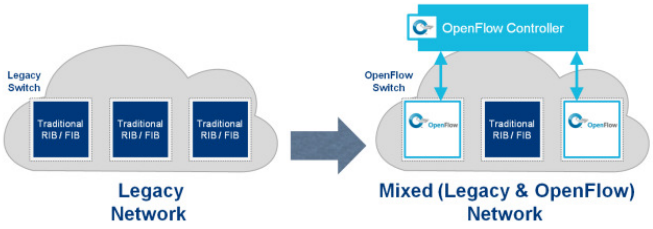

Figura 12: Despliegue Mixto

Enfoque de Migración: El enfoque de migración recomendado por la ONF para este escenario consiste en mover usuarios individuales y VLANs individuales a la red de control OpenFlow para gestionar el riesgo asociado en desplegar la nueva tecnología. Se consideran 4 fases fundamentales:

- Añadir soporte OpenFlow al hardware: Se hace una actualización del firmware a los dispositivos de red para soportar OpenFlow.
- Verificar el soporte OpenFlow en los dispositivos de red: Se verifica la funcionalidad OpenFlow incorporando una VLAN experimental gestionada por el controlador SDN.
- Migración de usuarios a la nueva red: Se crea una nueva red no-OpenFlow y se migran los usuarios a esta nueva red antes de utilizar OpenFlow para el tráfico de producción. Esta tarea involucra las siguientes actividades:
  - Añadir una nueva subred de producción.
  - Añadir/Mover usuarios gradualmente a la nueva subred.
  - Verificar la alcanzabilidad dentro de la nueva subred.
- Habilitar OpenFlow en la nueva subred: Una vez verificado que la nueva subred es funcional, se habilita el control OpenFlow a la subred configurando la regla correspondiente en el controlador.

C. Validación

La validación es una fase fundamental en el proceso de despliegue de la red ya que permite verificar el cumplimiento de los objetivos del diseño y de las metas del negocio. Para

efectuar la validación, el despliegue debe contar con una infraestructura de monitoreo que permita recopilar, analizar y verificar el correcto funcionamiento y la alcanzabilidad, desempeño y estabilidad de la nueva red.

Infraestructura de Monitoreo: La infraestructura de monitoreo está conformada por un conjunto de herramientas de captura, almacenamiento y análisis de tráfico y por encuestas de satisfacción al cliente. La infraestructura de monitoreo recopila información en 2 planos:

- Plano de Control: Se recopila información a nivel de flujos basado en mensajes de control entrantes *packet-in* y *flow-exp*. Otras estadísticas importantes a recopilar son la tasa de arribo de flujos *flow_arrival_rate* y los flujos activos *active_flows*.
- Plano de Datos: Se pueden implementar nodos de monitoreo dedicados a la recolección de estadísticas ejecutando comandos ping y wget entre sí que permitan recopilar información sobre la utilización de CPU del switch, el tiempo de establecimiento de flujos, RTT, retardos wget y tasa de pérdidas de paquetes.

Entre las herramientas de monitoreo convencionales se encuentran ping [62], tcpdump[1] y wget[2]. También es conveniente disponer de herramientas más completas con interfaces gráficas y soporte de reportes como NetFlow [63], sFlow [64], JFlow [65] o herramientas de monitoreo especializadas que verifiquen el comportamiento de la red y la correctitud del funcionamiento OpenFlow. Algunas herramientas especiales disponibles son:

PayLess [66]: Es un framework de monitoreo basado en consultas para redes SDN que provee un API RESTful flexible para recopilar estadísticas a diferentes niveles de agregación, como flujos, paquetes y puertos. PayLess ejecuta la recopilación de información con alta precisión en tiempo real sin incurrir en un alto overhead de red.

OpenTM [67]: Es una arquitectura de monitoreo que lleva la traza de los flujos activos en una red OpenFlow. Adicionalmente obtiene la información de enrutamiento de la aplicación de enrutamiento del controlador y sondea periódicamente los contadores de cantidad de bytes y número de paquetes de los flujos activos en los switches a lo largo del camino de datos. Para reducir la sobrecarga en la red se pueden sondear de maneara aleatoria un subconjunto de los switches seleccionados cuidadosamente para no afectar la precisión de las estadísticas recopiladas por la herramienta.

FlowSense [68]: Es una arquitectura de monitoreo que permite estimar el desempeño de una red OpenFlow a un bajo costo. Utiliza un método pasivo que captura y analiza el intercambio de los mensajes de control entre los switches y el controlador de una red OpenFlow asociados a cambios en el tráfico de la red, como ocurre con los mensajes *PacketIn* y *FlowRemoved*. FlowSense utiliza los mensajes *PacketIn* que notifican la llegada de un nuevo flujo y los mensajes *FlowRemoved* que notifican la expiración de un flujo, para estimar la utilización de un enlace por un flujo.

[1] http://www.tcpdump.org
[2] https://www.gnu.org/software/wget

VII. Lineamientos Generales

El éxito de una implementación SDN se encuentra alineado a los siguientes lineamientos generales:

A. Factibilidad Técnica y Comercial

La red SDN/OpenFlow de la solución debe satisfacer los requerimientos del negocio y el siguiente conjunto mínimo de objetivos generales para proveer factibilidad técnica:

- La disponibilidad de la red debe ser superior a un objetivo definido. Por ejemplo, en la red de la Universidad de Stanford, la disponibilidad esperada era superior a 99.9%.
- Debe existir un mecanismo de reversión a la red legada en caso de fallas durante la migración.
- El desempeño de la red debe estar cercano o ser superior al desempeño de la red legada.
- La experiencia de los usuarios no debe ser afectada durante la migración. La red objetivo debe ser programable mediante APIs abiertos y extensibles.
- La red objetivo debe ser utilizable y soportar actualizaciones automatizadas de software con mínima interrupción de servicio y debe tener mecanismos de reversión.
- La red objetivo debe ser interoperable con dispositivos de red de diferentes fabricantes.
- La red objetivo debe ser administrable con software, herramientas y simuladores disponibles.
- La red de arranque puede requerir la preparación y transformación a un estado intermedio seguro desde el cual se pueda proceder a una migración total de la solución.
- La red final debe ser validada con respecto a los requerimientos y expectativas del negocio.

Los costos de la solución deben respetar el presupuesto asignado al proyecto para satisfacer la factibilidad comercial.

B. Modelo de Despliegue

La selección del modelo de despliegue es un factor determinante en los costos y la gestión de riesgo de la solución. El modelo SDN basado en Dispositivos debe implementarse en despliegues Greenfield, mientras que los modelos SDN Overlay o SDN Híbridos deben implementarse en despliegues Mixtos.

C. Selección de los Componentes de Hardware y Software

La selección de los componentes de la solución dependerá de los costos y los requerimientos de criticidad de la solución. El tipo y modelo de controladores se determina en base a las cargas de tráfico actual, al crecimiento esperado de la red y a las funcionalidades soportadas. Para la selección de los controladores se toman en cuenta los siguientes criterios: (1) desempeño, (2) escalabilidad, (3) alta disponibilidad, (4) modularidad, (5) flexibilidad, (6) interoperabilidad, (7) soporte de delegación del control, (8) portabilidad de aplicaciones y (9) licencias.

D. Fases

Para el despliegue de una red SDN en un campus empresarial, se deben cumplir una serie de fases y un conjunto de consideraciones y lineamientos técnicos y estratégicos que

garanticen el éxito de la implementación. En un despliegue SDN se deben cumplir las siguientes fases de ejecución: (1) Preparación, (2) Implementación y (3) Validación.

E. Validación

El éxito de un despliegue SDN depende del cumplimiento de la solución con las expectativas del negocio. Se deben implementar pruebas de validación en los planos de control y datos durante la migración para verificar que se cumplen los objetivos del negocio.

F. Disponibilidad

En una red SDN/OpenFlow, el controlador es el único responsable del funcionamiento de la red lo cual ocasiona la condición de un solo punto de falla, en la cual al fallar el controlador o al fallar su comunicación ocasiona un fallo general en la red. Para evitar esta condición se pueden implementar los siguientes mecanismos:

Redundancia de controladores: La plataforma de controladores debe ser redundante para continuar el funcionamiento de la configuración de políticas en caso de que falle el controlador principal a cargo de la red.

Datos de controladores en espejo: La base de datos utilizada por el controlador principal para efectuar la toma de decisiones debe ser replicada y estar disponible para su uso por otros controladores de la red, de tal forma, que los controladores puedan configurar los dispositivos de red de una manera consistente.

Redundancia de caminos hacia los controladores: Se deben establecer enlaces y caminos redundantes hacia los controladores para asegurar que exista al menos un camino de comunicación disponible entre un switch y un controlador.

G. Escalabilidad

La solución debe ser capaz de escalar con el crecimiento de usuarios, aplicaciones y dispositivos en la red. En caso de superarse la capacidad, se debe contar con un plan de acción para soportar las nuevas cargas de tráfico, incluyendo mecanismos de redundancia o segmentación de la red en dominios de control que compartan alcanzabilidad entre sus controladores.

H. Gestión

Para garantizar el éxito de la implementación de la solución se debe contar con una infraestructura de monitoreo y gestión robusta capaz de dar seguimiento y control a los aspectos de alcanzabilidad, desempeño, estabilidad y correctitud del funcionamiento SDN/OpenFlow.

I. Seguridad

Se deben proveer mecanismos de seguridad al tráfico de la red a través de la definición de reglas de filtrado en los controladores y en el desarrollo de programas de control que interactúen con el controlador y permitan al controlador manipular los caminos de flujos basado en el análisis de tráfico y en las estadísticas de la red.

J. Transferencia de Conocimientos

Para garantizar la operación de la nueva red es fundamental impartir el conocimiento del funcionamiento y operación de la red OpenFlow a los administradores de la red. Se debe hacer énfasis en los aspectos de configuración, actualización y validación de reglas en el controlador SDN, en la integración con aplicaciones externas, en los mecanismos de redundancia de los controladores y en los aspectos de programación de los controladores de la red.

VIII. CONCLUSIONES Y TRABAJOS FUTUROS

SDN es una arquitectura de red emergente que desacopla el plano de control y el plano de datos de los dispositivos y centraliza la inteligencia y el control de la red en un controlador central. La centralización de la inteligencia en una entidad central provee grandes beneficios en términos de operación, control y gestión de la red. El controlador SDN dispone la visión global de la red lo cual facilita y acelera los tiempos de respuesta en la resolución de problemas en la red y permite a los administradores de red implementar optimizaciones y aprovisionar nuevos servicios de manera ágil y expedita.

El análisis de los resultados reportados en la investigación de la arquitectura SDN/OpenFlow, permitió fundamentar un conjunto de consideraciones para apoyar a los administradores de red en el despliegue de redes SDN. Entre los aspectos más relevantes a considerar se encuentran los siguientes:

- Una solución SDN debe estar alineada con los objetivos y metas del negocio para la cual fue diseñada.
- Invertir una mayor cantidad de esfuerzo durante las pruebas de concepto de la solución permite detectar fallas a tiempo y agilizar los tiempos de despliegue en etapas posteriores de este.
- Un mecanismo para garantizar el funcionamiento de una nueva red SDN radica en contar con una buena infraestructura de monitoreo de red que permita efectuar chequeos y controles durante todas las etapas de un despliegue SDN.

Como trabajo futuro, planificamos desarrollar una herramienta que medición de desempeño para controladores SDN. Esta herramienta daría el tiempo de respuesta de un controlador cuando surge un cambio en un switch ya que este último debe comunicarse con el controlador para que le especifique las acciones a tomar. Similarmente, la herramienta podría inundar un controlador SDN de peticiones para estresarlo, y se reportaría la cantidad de peticiones que el controlador sería capaz de atender en condiciones de carga extrema. En este último caso, la herramienta se desarrollaría con una arquitectura distribuida de tal forma que se pueda someter una mayor carga de peticiones al controlador SDN.

REFERENCIAS

[1] Open Networking Foundation. *Software Defined Networking: The New Norm for Networks*. ONF White Paper. April 2012.

[2] Open Networking Foundation. *Migration Use Cases and Methods*. Migration Working Group. 2014.

[3] M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, J. Little, J. Van Reijendam, P. Weissmann, and N. Mckeown. *Maturing of OpenFlow and Software Defined Networking through Deployments*. Computer Networks: The International Journal of Computer and Telecommunications Networking. vol. 61, pp. 151-175. March 2014.

[4] J. Skorupa, M. Fabbi, and M. Fabbi. *Ending the Confusion about Software-Defined Networking: A Taxonomy*. Gartner Research, G00248592. March 2013.

[5] W. Xia, Y. Wen, C.H. Foh, D. Niyato, and H. Xie. *A Survey on Software-Defined Networking*. IEEE Communication Surveys & Tutorials, vol. 17, no. 1, pp. 27-51, First Quarter 2015. 2015.

[6] A. Lara, A. Kolasani, and B. Ramamurthy. *Network Innovation using OpenFlow: A Survey*. IEEE Communications Survey & Tutorials. August 2013.

[7] P. Goransson and C. Black. *Software Defined Networks, A Comprehensive Approach*. Morgan Kaufmann. May 2014.

[8] P. Morreale and J. Anderson. *Software Defined Networking Design and Deployment*. CRC Press. 2015.

[9] E. Haleplidis, K. Pentikousis, S. Denazis, J. Hadi Salim, D. Meyer, and O. Koufopavlou. *Software-Defined Networking (SDN): Layers and Architecture Terminology*. RFC 7426. Internet Research Task Force. January 2015.

[10] H. Kim and N. Feamster. *Improving Network Management with Software Defined Networking*. IEEE Communications Magazine. vol. 51, no. 2. pp. 114-119. 2013.

[11] A. Porxas. *Virtualization-enabled Adaptive Routing for QoS-aware Software-Defined*. Master thesis. Universitat Politècnica de Catalunya, Spain, December 2014.

[12] D. Kreutz, F. Ramos, et al. *Software-Defined Networking: A Comprehensive Survey*. Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76. January 2015.

[13] B. Pfaff and B. Davie. *The Open vSwitch Database Management Protocol*. RFC 7047. Internet Research Task Force. December 2013.

[14] A. Doria, J. Hadi, R. Hass, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern. *Forwarding and Control Element Separation (ForCES) Protocol Specification*. RFC 5810. Internet Research Task Force. March 2010.

[15] Y. Rekhter, T. Lid, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271. Internet Research Task Force. January 2006.

[16] J. Case, M. Fedor, M. Schoffstall, and C. Davin. *A Simple Network Management Protocol (SNMP)*. RFC 1098. Internet Research Task Force. May 1990.

[17] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman. *Network Configuration Protocol (NETCONF)*. RFC 6241. Internet Research Task Force. June 2011.

[18] L. Richardson and S. Ruby. *RESTful Web Services. Web Services for the Real World*. O'Reilly Media. May 2007.

[19] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California. Irvine, CA, USA, 2000.

[20] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi. *SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains*. Internet Engineering Task Force, Internet Draft. June 2012.

[21] JP. Vasseur and JL. Le Roux. *Path Computation Element (PCE) Communication Protocol (PCEP)*. RFC 5440. Internet Engineering Task Force. March 2009.

[22] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. *FlowVisor: A Network Virtualization Layer*. Deutsche Telekom Inc. R&D Lab, Stanford University, Nicira Networks. October 2009.

[23] A. Al-Shabibi, M. De Leenheer, A. Koshibe, G. Parulkar, B. Snow, M. Gerola, and E. Salvadori. *OpenVirteX: Make Your Virtual SDNs Programmable*, in proceedings of the Third Workshop on Hot Topics in Software Defined Networks (HotSDN 2014). Chicago, IL, USA, August 2014.

[24] Z. Bozakov and P. Papadimitriou. *AutoSlice: Automated and Scalable Slicing for Software-defined Networks*, in proceedings of the 2012 ACM Conference on CoNEXT Student Workshop (CoNEXT Student 2012), Nice, France, December 2012.

[25] S. Gutz, A. Story, C. Schlesinger, and N. Foster. *Splendid Isolation: A Slice Abstraction for Software-defined Networks*, in proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN 2012), Helsinki, Finland, August 2012.

[26] H. Kim and N. Feamster. *Improving Network Management with Software Defined Networking*. IEEE Communications Magazine, vol. 51, no. 2, pp. 114-119, 2013.

[27] T. Hinrichs, N. Gude, M. Casado, J. Mitchell, and S. Shenker. *Practical Declarative Network Management*, in proceedings of the First ACM Workshop on Research on Enterprise Networking (WREN 2009), Barcelona, Spain, August 2009.

[28] N. Foster, R. Harrison, M. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. *Frenetic: A Network Programming Language*, in proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming (ICFP 2011), Tokyo, Japan, September 2011.

[29] C. Monsanto, N. Foster, R. Harrison, and D. Walker. *A Compiler and Run-time System for Network Programming Languages*, in proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2012), Philadelphia, PA, USA, January 2012.

[30] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker. *Modular SDN Programming with Pyretic*. Usenix, The Advanced Computing Systems Association, vol. 38, no. 5, October 2013.

[31] ITU. *Management Framework for Open Systems Interconnection (OSI) for CCITT Applications. ITU Recommendation X.700*. September 1992.

[32] F. Maccioni. *Network Automation with Ansible 2.1 and Beyond* [PowerPoint slides]. Cisco. September 2016.

[33] M. Bjorklund. *The YANG 1.1 Data Modeling Language*. RFC 7950. Internet Research Task Force. August 2016.

[34] A. Bierman, M. Bjorklund, and K. Watsen. *RESTCONF Protocol*. RFC 8040. Internet Research Task Force. January 2017.

[35] gRPC. gRPC *A High Performance, Open-Source Universal RPC Framework*. http://www.grpc.io.

[36] Open Networking Foundation. *OF-CONFIG 1.2. OpenFlow Management and Configuration Protocol. ONF TS-016*. 2014.

[37] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. RFC 7348. Internet Engineering Task Force. August 2014.

[38] P. Garg and Y. Wang. *NVGRE: Network Virtualization using Generic Routing Encapsulation*. RFC 7637. Internet Engineering Task Force. September 2015.

[39] B. Davie and J. Gross. *A Stateless Transport Tunneling Protocol for Network Virtualization (STT)*. Internet Engineering Task Force, Internet Draft. April 2016.

[40] N. McKeown. *OpenFlow: Enabling Innovation in Campus Networks*. Stanford University. March 2008.

[41] J. Davis. *Introduction to Software-Defined Networking (SDN) and Network Programability*. CiscoLive BRKSDN-1014 [Power Point-Slides]. 2015.

[42] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.0.0 Implemented (Wire Protocol 0x01)*. December 2009.

[43] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.1.0 Implemented (Wire Protocol 0x02)*. February 2011.

[44] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.2.0 Implemented (Wire Protocol 0x03)*. December 2011.

[45] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.3.0 Implemented (Wire Protocol 0x04)*. June 2012.

[46] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.4.0 Implemented (Wire Protocol 0x05)*. October 2013.

[47] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.5.0 Implemented (Wire Protocol 0x06)*. December 2014.

[48] Linux Foundation. *OpenDaylight: Open Source SDN Platform*. https:// www.opendaylight.org.

[49] Project Floodlight. *Open Source Software for Building Software-Defined Networks*. http://www.projectfloodlight.org/floodlight.

[50] Ryu SDN Framework Community. *Component-Based Software Defined Networking Framework. Build SDN Agilely*. http://osrg.github.io/ryu.

[51] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou. *Feature-Based Comparison and Selection of Software Defined Networking (SDN) Controllers*, in proceedings of the 2014 World Congress on Computer Applications and Information Systems (WCCAIS 2014), Hammamet, Tunisia, January 2014.

[52] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky. *Advanced Study of SDN/OpenFlow Controllers*, in proceedings of the

9th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR 2013), Moscow, Russian Federation, October 2013.

[53] Big Switch Networks. *SDN Controller Unified Network Control Plane*. http://www.bigswitch.com/products/SDN-Controller.

[54] Brocade. *Brocade SDN Controller Data Sheet*, 2016.

[55] Cisco. *Cisco Open SDN Controller 1.2 Data Sheet*. October 2015.

[56] HPE. *Aruba VAN SDN Controller Software Data Sheet*. 2017.

[57] NEC. PF6800 ProgrammableFlow Controller. https://www.necam.com/sdn/Software/SDNController.

[58] B. Lantz, B. Heller, and N. McKeown. *A Network in a Laptop: Rapid Prototyping for Software-Defined Networks*, in proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets IX), Monterey, CA, USA, October 2010.

[59] J. Yan and D. Jin. *VT-Mininet: Virtual-time-enabled Mininet for Scalable and Accurate Software-Define Network Emulation*, in proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Network Research (SOSR 2015), Santa Clara, CA, USA, June 2015.

[60] B. Linkletter. *OFNet SDN Network Emulator*. http://www.brianlinkletter.com/ofnet-a-new-sdn-network-emulator.

[61] S. Wang, C. Chou, and C. Yang. *EstiNet OpenFlow Network Simulator and Emulator*, IEEE Communications Magazine, vol. 51, no. 9. September 2013.

[62] J. Postel. *Internet Control Message Protocol*, RFC 792, Internet Engineering Task Force, September 1981.

[63] Cisco Systems. *Introduction to Cisco IOS NetFlow - A Technical Overview*, May 2012.

[64] sFlow. *Traffic Monitoring using sFlow*. 2003.

[65] A. Myers. *JFlow: Practical Mostly-Static Information Flow Control*, in proceedings of the 26th ACM Symposium on Principles of Programming Languages (POPL 1999), San Antonio, TX, USA, January 1999.

[66] S. Chowdhury, Md. F. Bari, R. Ahmed, and R. Boutaba. *PayLess: A Low Cost Network Monitoring Framework for Software Defined Networks*, in proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS 2014), Krakow, Poland, June 2014.

[67] A. Tootoonchian, M. Ghobadi, and Y. Ganjali. *OpenTM: Traffic Matrix Estimator for OpenFlow Networks*, in proceedings of the 11th International Conference on Passive and Active Measurement (PAM 2010), Zurich, Switzerland, April 2010.

[68] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. Madhyastha. *Flowsense: Monitoring Network Utilization with Zero Measurement Cost*, in proceedings of the 14th International Conference on Passive and Active Measurement (PAM 2013), Hong Kong, China, March 2013.

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 2, pp. 34-45, Diciembre 2017
Selección de los Mejores Artículos de CoNCISa 2017

Propuesta para la Gestión de Proyectos Socio Tecnológicos del Programa Nacional de Formación en Informática

Iris Albarran [1], Liliana Silva [1], Colombia Amezquita [1], Marbella Castañeda [1]
iealbarranp@iutfrp.edu.ve, iutrc.liliana@gmail.com, ceamezquitaz@iutfrp.edu.ve, mdrcastanedar@iutfrp.edu.ve

[1] Departamento de Informática, Instituto Universitario de Tecnologia "Dr. Federico Rivero Palacio", Caracas, Venezuela

Resumen: A partir del año 2008, en el IUT "DR. Federico Rivero Palacio" comienza la gestión del Plan Nacional de Formación en Informática (PNFI). Este incorpora aspectos importantes como la formación por competencias y la definición del currículo por ejes transversales y longitudinales. El Proyecto Socio Tecnológico (PST), el cual forma parte del eje longitudinal, se plantea como una Unidad Curricular cuyas actividades en desarrollo propician la vinculación entre la Universidad y la Comunidad, mediante la satisfacción mutua de necesidades comunes. El objetivo de este trabajo es facilitar una propuesta que sustente la gestión y administración del PST y así garantizar que los productos de software generados por los estudiantes en la unidad curricular PST, sean de la satisfacción de las comunidades con las cuales se les vincula, adicional al logro académico esperado. Para ello se generó una propuesta de Gestión del Eje, aplicada a cuatro cohortes académicas.

Palabras Clave: Gestión-de-Proyectos; Proyecto-Socio Tecnológico; Comunidad.

Abstract: From 2008, El Plan Nacional de Formación en Informática (The National Plan on Computer Science Training, PNFI in Spanish) gets started. It involves important aspects such as a competence oriented education and a curriculum that is based on both a vertical and a horizontal axis. The Proyecto Socio Tecnológico (The Socio-Technological Project, PST in Spanish) which is part of the vertical axis, is a curricular unit (subject)that includes activities aimed to develop a connection between the university and the community by satisfying each other's needs. The objective of this study is to bring about a proposal to support the Implementation of The PST and guarantee that the software products that the students create satisfy their communities necessities, besides the expected academic achievement. With this in mind, a Gestión de Eje (Axis Management) is proposed and it will be taught to four academic cohorts.

Keywords: Projects Management; Socio-Technological Project; Community.

I. Introducción

En el año 2008, con el fin de regular los Programas Nacionales de Formación en Educación Superior, se crean los distintos Planes Nacionales de Formación (PNF). A partir de ese año se comienza a gestionar en el IUT "DR Federico Rivero Palacio" el Programa Nacional de Formación en Informática (PNFI), esto condujo a cambios en la gestión del currículo en cuanto a la estructuración y el abordaje de los conocimientos los cuales se enmarcan por ejes transversales y longitudinales. Esta estructuración y la forma de abordar el conocimiento fueron pensados con la finalidad de adecuar una currícula innovadora que orbitase alrededor de los retos de formación propios de la época y que son planteados por la UNESCO 2006.

El PNFI [1] plantea la formación total en cuatro (4) trayectos, generando en cada uno de los mismos una certificación o un título asociado a los conocimientos y competencias del perfil de egreso definido en el currículo. En cada uno de los trayectos se gestiona y administra un PST, el aprendiz a través de esta Unidad Curricular (UC) debe demostrar y aplicar los saberes asociados de las diferentes unidades curriculares generando un producto informático que dé solución a problemas reales de la comunidad, de manera que se refuerza la interacción Comunidad - Universidad.

Es importante resaltar que desde el año 2008 hasta la fecha, se han desarrollado productos para las comunidades a través de los PSTs. Los resultados desde el punto de vista académico, en la mayoría de los casos, suelen ser satisfactorios, no obstante, no siempre se logra satisfacer las necesidades de las comunidades. Pareciera que esto se debe a que los tiempos de desarrollo (tiempo académico) del software superan el tiempo requerido por el usuario para utilizar el producto. Otro factor que ocurre con menos frecuencia pero que ha incidido en las fallas de las soluciones a la comunidad, está relacionada a la disgregación del equipo de trabajo, que se interesó en desarrollar el proyecto en su inicio, pero no logra dar respuesta al ente externo, por no contar con todos los recursos humanos y operativos. Así mismo, pueden ocurrir otros factores de riesgo

en la aceptación del producto por parte de la Comunidad y esto se debe en algunos casos, a los cambios en las dependencias del ente, sobre todo en los niveles organizacionales decisores cuando el proyecto está avanzado y se originan nuevos requerimientos.

La problemática expresada anteriormente nos invita a reflexionar y a formularnos la siguiente interrogante: ¿De qué manera se puede lograr una solución en tecnología informática, real y adecuada, que satisfaga las necesidades de las comunidades en el tiempo requerido y que además, se garantice que los equipos de trabajo demuestren la aplicación rigurosa de los conocimientos y competencias que exige la academia?

Para dar respuesta a esta interrogante nos plantearnos como objetivo: proporcionar una guía de Gestión, para el desarrollo del Proyecto Socio Tecnológico, de manera que se puedan garantizar proyectos reales para las comunidades y que los estudiantes logren integrar los conocimientos y competencias aprendidas en las unidades curriculares del trayecto, especialmente la aplicación de las metodologías y modelos establecidos por la Ingeniería del Software y los perfiles de egreso planteados en el PNFI. Para construir la propuesta, se realizó el análisis del currículo basado en el eje proyecto, los diferentes perfiles de egreso y los ejes transversales que agregan entre otros los conocimientos propios de la profesión. Adicionalmente se realizó la propuesta de gestión de los proyectos Informáticos y por último se estructuraron las estrategias de planificación y seguimientos por trayecto y por etapas.

II. Marco Referencial

A. *Programa Nacional de Formación en Informática (PNFI)*

Por Resolución Nº 2.963 del Ministerio del Poder Popular para la Educación Superior (MPPES), con fecha 14 de mayo de 2008, gaceta oficial Nº 38.930, el Ejecutivo resuelve regular los Programas Nacionales de Formación en Educación Superior. Para ello define en el Artículo 2:

"… se entiende por: Programas Nacionales de Formación en Educación Superior: El conjunto de actividades académicas, conducentes a títulos, grados o certificaciones de estudios de educación superior, creados por iniciativa del Ejecutivo Nacional, a través del Ministerio del Poder Popular para la Educación Superior, diseñados con la cooperación de Instituciones de Educación Superior Nacionales, atendiendo a los lineamientos del Plan de Desarrollo Económico y Social de la Nación, para ser administrados en distintos espacios educativos del territorio Nacional."

El Programa Nacional de Formación en Informática (PNFI) fue creado el 07 de octubre de 2008 en la resolución Nº 3147. En la actualidad el PNFI es gestionado en 33 instituciones entre las que se encuentran: Institutos Universitarios de Tecnología (IUT), los Colegios Universitarios (CU) y las Universidades Nacionales Experimentales (UNE).

El PNFI presenta un currículo por competencias.

B. *El Proyecto Socio Tecnológico (PST) en el Programa Nacional de Formación en Informática (PNFI)*

La Unesco (2013) [2] plantea tres aspectos importantes relacionados con el qué y cómo educar, ellos son:

1) la necesidad de adoptar y desarrollar un enfoque integral al aprendizaje que considere no solamente los conocimientos académicos, el desarrollo cognitivo y las capacidades, sino también las dimensiones "no cognitivas" – actitudes, valores, emociones, cualidades personales

2) la exigencia de considerar la dimensión aplicada del conocimiento, puesto que no solamente cuenta lo que se sabe sino también lo que se puede hacer con este saber

3) la importancia de repensar enteramente la estructura disciplinar tradicional del currículo, la organización de las experiencias de aprendizaje, la manera de enseñar y los sistemas de evaluación si se quiere promover el desarrollo efectivo de competencias.

Se ha entendido que el planteamiento anterior refleja el espíritu concebido en la fundamentación del PNFI. Al respecto, en la construcción del conocimiento en el mencionado Plan, se estructura en cinco ejes temáticos, tres ejes transversales que incluyen los ejes temáticos: (1) Epistemológico-Heurístico, (2) Socio-Cultural-Económico-Histórico-Ético Político y (3) Profesional y Estético-Lúdico y Ambiental; y dos ejes longitudinales conformados por (1) Proyecto Socio Tecnológico y (2) Formación Critica.

En la fundamentación del PNFI se destacan tres características a resaltar de proyecto [1]:

1) "El Proyecto Socio Tecnológico (PST) etimológicamente se relaciona con la palabra socio proveniente del latín, socius lo cual significa grupo humano. Por su parte, tecnológico se asocia con tecnología, correspondiente a fabricar objetos, productos o servicios y modificar el medio ambiente, lo cual genera una combinación adecuada a la formación del participante del programa, a su inserción y contacto con la realidad. En ese contexto es señalado por el Diccionario de la Real Academia Española (2007)."

2) "Constituye por tanto, el PST, el núcleo central del Programa Nacional de Formación en Informática, referido como una unidad curricular en cada uno de los trayectos con una importante carga crediticia. De este modo, representa un eje longitudinal transversal que orienta y define el resto de las demás unidades curriculares, desarrollándose de forma incremental, aumentando su nivel de complejidad y profundidad en cada trayecto."

3) "El PNFI propone el desarrollo de Proyectos Socio Tecnológicos como estrategia de aprendizaje que permite la construcción del conocimiento a partir del aprender haciendo, donde se propicia el reconocimiento en principio por el propio participante de sus conocimientos, habilidades y destrezas, que luego debe desarrollar a partir del Proyecto Socio Tecnológico convirtiéndose en crecimiento personal y confianza en el participante de su proceso formativo y del rol profesional a desempeñar".

El Proyecto Socio Tecnológico (PST) procura proporcionar un aprendizaje significativo que integre los saberes, valores, aptitudes, actitudes, habilidades y destrezas de un estudiante y donde el resultado se materialice en productos de software y que estos tengan un impacto dentro de las comunidades vinculadas proporcionando soluciones a problemas reales.

En el documento rector del PNFI se presentan las siguientes características del PST [1]:

- El planteamiento del proyecto se basa en un problema real local, regional o nacional que incorpore las áreas de los saberes de la informática.
- Provee oportunidades para que los participantes realicen investigaciones que les permitan aprender nuevos conceptos, aplicar la información y representar el conocimiento de diversas formas.
- Provee la posibilidad de trabajo en equipo y colaboración entre los participantes, profesores asesores y otras personas involucradas con el proyecto a fin de que el conocimiento sea compartido y distribuido.
- Posibilita el uso de herramientas cognitivas y ambientes de aprendizaje que motivan al participante a representar sus ideas y fomentar la construcción de sus conocimientos.
- Pueden abarcar más de un trayecto dependiendo de su objetivo y complejidad.
- Demandan la aplicación de conocimientos interdisciplinarios. Así, el participante puede apreciar la relación existente entre las diferentes disciplinas en el desarrollo de un proyecto en particular.
- Permiten la búsqueda de soluciones abiertas, dando así oportunidad al participante de generar nuevos conocimientos.

En general las propuestas de PST provienen de comunidades públicas o privadas las cuales plantean necesidades de automatización de acuerdo a la naturaleza de sus funciones. Se establece el vínculo Comunidad- Universidad a través de un PST asignado a uno o más grupos de estudiantes, dependiendo de la complejidad y el perfil de egreso del estudiante.

Si en este punto se triangula la propuesta de [1] y [2], encontramos que es un reto mejorar la Unidad Curricular Proyecto, la cual se considera además, un eje dentro del currículo y una estrategia de aprendizaje, de manera que garantice en dicha UC, el logro de las competencias de acuerdo al perfil y adicionalmente que el producto obtenido de ese logro sea de provecho directo a la comunidad con la cual se vincula.

C. Perfiles de Egreso del Programa Nacional de Formación en Informática (PNFI)

El perfil de egreso se refiere a las competencias logradas por un estudiante una vez alcanzado un nivel académico determinado, de acuerdo a lo planteado en el diseño curricular.

El PNFI presenta cuatro perfiles de egreso correspondiente a cada uno de los trayectos: dos constituyen títulos de nivel superior y dos son salidas intermedias o certificaciones. En la Tabla I, se muestran las salidas correspondientes a los perfiles según los Trayectos académicos.

Tabla I: Perfiles de Egreso por Trayectos

T I	T II	T III	T IV
Soporte Técnico a usuarios y Equipos	TSU en Informática	Desarrollador(a) de Aplicaciones	Ingeniero(a) en Informática

Según lo presentado en [1] el egresado como TSU del PNFI: "es un profesional con formación integral, que se desempeña con idoneidad operativa y ética profesional en la construcción de productos tecnológicos informáticos en armonía con la preservación del ambiente y del progreso de su entorno, aplicando los saberes para:

1) Desarrollar y mantener componentes de software bajo estándares de calidad, priorizando el uso de software libre.
2) Caracterizar, seleccionar, ensamblar, configurar y mantener equipos informáticos.
3) Interpretar el modelo de datos e implementar y mantener, de forma operativa, las bases de datos.
4) Instalar, configurar y administrar operativamente redes de área local, bajo estándares de calidad, priorizando el uso de software libre.
5) Participar técnicamente en el proceso de evaluación, selección e instalación de software".

Así mismo define al Ingeniero o la Ingeniera en Informática como: "un profesional con formación integral que se desempeña con idoneidad y ética profesional, en la conceptualización y construcción de productos tecnológicos informáticos en armonía con la preservación del ambiente y del progreso de su entorno, aplicando los saberes para:

1) Participar en la administración de proyectos informáticos bajo estándares de calidad y pertinencia social.
2) Auditar sistemas informáticos.
3) Desarrollar e implantar software bajo estándares de calidad y pertinencia social, priorizando el uso de plataformas libres.
4) Integrar y optimizar sistemas informáticos.
5) Diseñar, implementar y administrar bases de datos.
6) Diseñar, implementar y administrar redes informáticas bajo estándares de calidad, priorizando el uso de software libre."

En el mismo orden de idea las salidas intermedias constituyen un reconocimiento a las habilidades y destrezas adquiridas en el desarrollo de los PSTs en los que participa, y para obtenerlas se comprueba que posee los saberes necesarios. En el caso del certificado en "Soporte Técnico a Usuarios y Equipos" debe demostrar que posee los saberes para aplicar los conocimientos en:

- "Ensamblar, configurar y realizar mantenimiento preventivo y correctivo de equipos de computación de acuerdo a los requerimientos del usuario.
- Realizar soporte técnico a usuarios y equipos.
- Participar técnicamente en el proceso de evaluación, selección e instalación de software."

De manera similar, para obtener la certificación de "Desarrollador de Aplicaciones", en el tercer trayecto, el estudiante comprueba mediante las actividades de PST que es capaz de aplicar los conocimientos para:

- "Desarrollar y mantener aplicaciones y componentes de software,
- Interpretar el modelo de datos e implementar y mantener, de forma operativa, las bases de datos.
- Instalar, configurar y administrar operativamente redes de área amplia."

Se aprecia que algunos saberes se repiten en los perfiles de egreso, debido a que la formación es continua a lo largo de los trayectos y las competencias se van incrementando a lo largo de la ejecución del Plan.

D. Gestión de Proyectos Informáticos

Según [3] la gestión de un proyecto se define como: "Articular el método para alcanzar un objetivo único y no repetitivo en un plazo con principio y fin claros utilizando las técnicas que nos proporciona la gestión.", indican los autores además, que las tareas primordiales son: planificar y establecer estrategias adecuadas, organizar a los miembros y equipos para lograr los objetivos que se quieren alcanzar, controlar y comprobar si se están alcanzando dichos objetivos. Según estos autores "la organización de un proyecto consiste en diseñar la estructura con la que vamos a establecer las dependencias entre individuos, departamentos, cosas... dentro del proyecto. Asimismo, debemos asignar las tareas más idóneas para esas capacidades y el tiempo estimado para cumplir las tareas o funciones".

La gestión de los proyectos tiene dos aspectos que se resaltan en este trabajo, uno son las fases y el otro es el seguimiento y control del proyecto.

1) Fases de un Proyecto: Según la metodología PMBOK [4] "Las fases del proyecto son divisiones dentro del mismo proyecto, donde es necesario ejercer un control adicional para gestionar eficazmente la conclusión de un entregable mayor". Las fases de un proyecto se pueden relacionar de diferentes maneras una de forma secuencial donde el inicio de una depende de la culminación de la anterior, de una manera superpuesta, donde la fase siguiente comienza antes de que culmine la anterior y la tercera corresponde con una relación iterativa donde en un momento dado solo se planifica una fase y la siguiente depende del avance de proyecto y la generación de los e*ntregables en la etapa actual.* En la Figura 1, se muestra la estructura básica de un proyecto donde se agrupan diferentes componentes en cada fase.

Figura 1: Fases Principales de un Proyecto

Se observa que se propone como primera fase la planeación la cual incluye la definición del problema y la planificación propiamente dicha, en la segunda fase denominada ejecución se presenta la puesta en marcha, la fase productiva y la conclusión del proyecto, es decir contiene todas las fases de desarrollo del software incluyendo la transición del producto a la comunidad y finamente incorpora la fase de mantenimiento, recomendada para productos de software que ya se encuentran en producción.

Adicionalmente a lo planteado por [3], [4] incorpora algunos aspectos que deben considerarse en la fase de conclusión de proyectos, considerada en [4] como una fase y que incluye actividades como las siguientes:

- Obtener la aceptación del cliente o del patrocinador,
- Realizar una revisión tras el cierre del proyecto o la finalización de una fase,
- Registrar los impactos de la adaptación a un proceso,
- Documentar las lecciones aprendidas,
- Aplicar actualizaciones apropiadas a los activos de los procesos de la organización,
- Archivar todos los documentos relevantes del proyecto en el sistema de información
- Para la dirección de proyectos para ser utilizados como datos históricos y
- Cerrar las adquisiciones.

2) Seguimiento y Control del Proyecto: Las actividades de seguimiento y control, están constituidas por aquellas que reflejan todas las actividades requeridas para supervisar, analizar y regular el progreso y el desempeño del proyecto. El seguimiento de estas actividades arroja resultados que están relacionados con la dinámica propia del trabajo en cuestión, de manera que permite hacer correcciones, ajustes y algunos casos cambiar el rumbo inicialmente pautado. De manera que el seguimiento y control del proyecto es un proceso continuo y dinámico que requiere un esfuerzo constante de los involucrados. El grupo de seguimiento y control según [4] debe:

- Dar seguimiento y controlar el trabajo del proyecto
- Realizar control integrado de cambios
- Verificar el alcance
- Controlar el alcance
- Controlar el cronograma
- Controlar los costos
- Realizar control de calidad
- Informar el desempeño
- Dar seguimiento y control de riesgos
- Administrar las adquisiciones.

Esto permitirá a los involucrados: Controlar cambios, dar seguimiento a las actividades del proyecto y controlar los cambios de manera que se implementen cambios aprobados y no desvíe el alcance del proyecto.

E. Proceso Unificado de Desarrollo (RUP)

El proceso Unificado de Desarrollo fue creado por Booch, Jacobson, y Rumbaugh, en el año 2000 [5], con el fin de mitigar los riesgos en el proceso de desarrollo de software. Este proceso contiene todas las actividades necesarias para convertir los requisitos de un usuario en un sistema software, sigue un proceso iterativo e incremental.

Fases del Proceso Unificado de Desarrollo:

1) Fase de Comienzo o Inicio: el objetivo en esta etapa es concretar la visión del proyecto. Es la etapa donde se establecen los requerimientos y se delimita el alcance del proyecto. Finalmente se determinan los objetivos del ciclo de vida.

2) Fase de Elaboración: en esta fase se planifican las actividades a ejecutar y los recursos necesarios, haciendo énfasis en la arquitectura del software a utilizar. Al término de esta fase se llega al punto de no retorno del proyecto, ya que luego de ésta se debe afrontar la fase de construcción que es la más costosa y arriesgada.

3) Fase de Construcción: es la fase en la que se desarrolla el producto y se observa la evolución de la visión, la arquitectura y los planes hasta obtener la primera versión de lo que será manejado por el usuario final. En esta fase se hace especial énfasis en el control de las operaciones realizadas, administrando los recursos eficientemente de tal forma de minimizar costes, cumplir con el calendario y garantizar la calidad.

4) Fase de Transición: en esta fase se efectúa la transición del producto a los usuarios, lo cual incluye: manejo del producto, envío, entrenamiento, documentación, soporte y mantenimiento del producto hasta que el cliente esté satisfecho.

Cada una de estas fases es desarrollada mediante el ciclo de iteraciones. Una iteración es un ciclo de desarrollo completo dando como resultado una entrega de producto ejecutable.

III. Metodología

La necesidad de generar una propuesta de Gestión del eje longitudinal PST, que optimice la vinculación Universidad–Comunidad, se inicia con un proceso de entrevistas informales hechas a representantes de las comunidades que participaron como beneficiarios de los productos tecnológicos elaborados por estudiantes de las cohortes 2013-2014, 2014-2015, 2015-2016 y 2016-2017.

La propuesta se construye básicamente a partir del análisis documental de fuentes asociadas al problema abordado, el cual, aunado a la experiencia aportada por profesores del área, permitió extraer aspectos claves para la integración en una Guía, elementos que propicien: logro académico- satisfacción de la comunidad.

Elaborada la propuesta, y a fines de una primera valoración, se procedió a su aplicación en los trayectos correspondientes a las cohortes 2013-2014, 2014-2015, 2015-2016 y 2016-2017.

IV. Resultados

Los resultados se presentan en tres etapas. La primera refleja el análisis del currículo basado en el eje proyecto, la segunda presenta la propuesta de gestión de eje y la tercera etapa muestra la valoración de la propuesta.

A. Análisis del Currículo Basado en el Eje Proyecto

Se realizó el análisis de las Unidades Curriculares que contribuyen al desarrollo del eje longitudinal Proyecto por cada trayecto y que perfilan las competencias esperadas, con el fin de discernir la relación de estas con el perfil de egreso de cada trayecto.

En la Figura 2, denominada Visión Sistémica del Currículo, adecuada por [6], se muestra una estructuración simple del PNFI, especificándose en los laterales los ejes Longitudinales PST y Formación Crítica, y en el centro se hace la clasificación por áreas de las UCs que constituyen los ejes Transversales. Se agruparon bajo el eje de formación básica algunas UCs no incluidas en las áreas presentadas en el documento rector pero que han sido incluidas como cursos en el mismo.

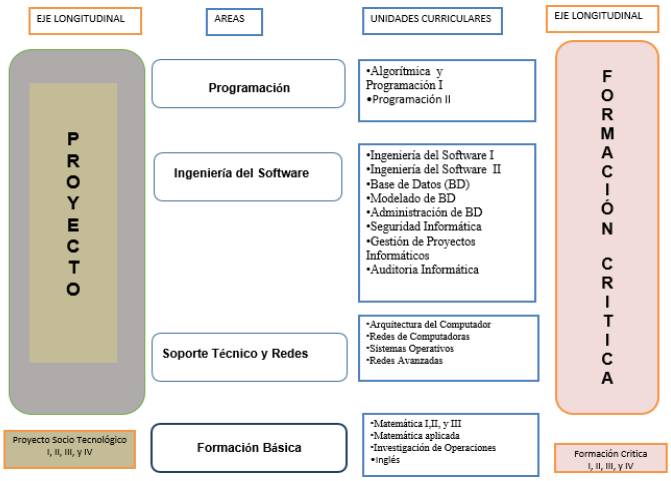

Figura 2: Visión Sistémica del Currículo

A continuación se presentan por cada trayecto una tabla (Tabla II, Tabla III, Tabla IV y Tabla V) donde se resaltan el conjunto de competencias requeridas por el aprendiz y el perfil de egreso pertinente al trayecto respectivo y que se relacionan con el PST. Ésta estructuración es tomada de [6] y es producto de la adecuación del análisis del documento rector. Las tablas se construyen tomando en cuenta todos los ejes de formación, de tal manera que se complemente el perfil técnico plasmado en el documento rector.

En la Tabla II, Atributos del Trayecto I, la contribución directa y fundamentalmente de las competencias técnicas para la construcción del PST en el Trayecto I, corresponde a la Arquitectura del Computador. La integración con el eje socio crítico permite al estudiante potenciar e integrar las características éticas sociales y de participación ciudadana, que son requeridas para el trabajo en equipo y respeto a la propiedad intelectual, entre otras. La UC Algorítmica y Programación provee las competencias básicas y técnicas para el desarrollo de componentes de software que son requeridas en el Trayecto II, en conjunto con otras UC.

Por otra parte, en la Tabla III Atributos del Trayecto II, a través de estos componentes curriculares mostrados, el estudiante aprende herramientas, metodologías y lenguajes de programación que le permiten abordar adecuadamente el desarrollo del PST. Es importante resaltar que el alcance de la aplicación informática que debe construir el equipo de trabajo en un PST, se limita al desarrollo de componentes. En este sentido, la definición de componentes además de otros conceptos son asumidos por la UC Ingeniería del Software. Los conocimientos y competencias adquiridos por los aprendices en el Trayecto I son acumulativos y se refuerzan y potencian a través del eje Longitudinal. Hasta el Trayecto II el estudiante éticamente puede realizar actividades de soporte y además desarrollar componentes de software.

En el Trayecto III se incorporan elementos técnicos de mayor profundidad, así mismo el proceso de abstracción del estudiante se ha desarrollado complementado con las UCs del componente de formación general y el eje socio crítico. El PST en este trayecto le permite al estudiante concebir un proyecto de una complejidad más alta. El equipo de trabajo está en capacidad de desarrollar un sistema completo partiendo desde el análisis de necesidades y requerimientos, construyendo el

diseño e implementación de soluciones automatizadas completas y finalmente realizando un conjunto de pruebas que pondría a tono el sistema. Desde el punto de vista académico y de la gestión de PST como UC, el alcance del proyecto es mayor, por lo que se requiere por parte del equipo docente especialistas del área técnica, evaluar proyectos de alta complejidad para ser distribuidos a los equipos de estudiantes interesados en brindar solución a través de un producto software a una comunidad y así mismo, demostrar sus sólidos conocimientos en la aplicación de metodologías, herramientas, métodos y análisis de problemas complejos. También, en este momento, el estudiante estará en capacidad de desarrollar productos aplicando estándares de calidad y hacer propuestas innovadoras desde su PST.

Tabla II: Atributos del Trayecto I

Competencias Profesionales	Unidades Curriculares	Proyecto I
Aplicar el análisis lógico y el razonamiento inductivo y deductivo en la solución de problemas, a partir del desarrollo alcanzado en el pensamiento abstracto. Operar una computadora personal conociendo sus características y funcionalidades. Caracterizar y seleccionar equipos de computación. Ensamblar y configurar equipos de computación de acuerdo a los requerimientos del usuario. Realizar el mantenimiento preventivo y correctivo de equipos de computación. Realizar soporte a usuarios y equipos de computación. Desarrollar algoritmos de mediana complejidad, implementarlos y ponerlos a punto usando los estándares adecuados. Utilizar el idioma inglés para manejar bibliografía técnica relacionada con la informática. Fomenta la cultura de la innovación para contribuir con la soberanía y seguridad tecnológica.	***Programación*** Algorítmica y Programación ***Soporte Técnico*** Arquitectura del Computador ***Cursos*** Matemáticas I Formación Critica I Inglés	***Certificación en Soporte Técnico a Usuarios y Equipos*** Plantear alternativas de soluciones ante situaciones y problemas reales, relacionados con soporte técnico a usuarios y equipos.

En el Trayecto IV, se da por sentado que los aprendices cuentan con todos los saberes acumulados desde el inicio de la carrera al momento de desarrollar un PST. Ahora corresponde incorporar un nuevo elemento en el desarrollo del software, el cual se refiere a la seguridad informática. En resumen: el estudiante desde el punto de vista técnico e integral está en capacidad de cumplir con todas las etapas planteadas en el desarrollo de sistemas, aplicando además criterios de calidad y seguridad informática. Tiene la capacidad de analizar y desarrollar proyectos complejos, pudiendo continuar con proyectos realizados en el Trayecto III o sobre cualquier otro proyecto que haya sido desarrollado en la comunidad y que el estudiante pueda añadir valor agregado desde su conocimiento y competencia.

Tabla III: Atributos del Trayecto II

Competencias Profesionales	Unidades Curriculares	Proyecto II
Aplicar el pensamiento y la reflexión lógica en la organización y formalización de conocimientos relacionados con el cálculo integral y conceptos básicos matemáticos en otras áreas de saberes. Elaborar algoritmos de alta complejidad utilizando estructuras de datos en memoria interna y externa, así como, programarlos en un lenguaje de alto nivel. Desarrollar y mantener componentes de software, bajo estándares de calidad, priorizando el uso de software libre. Instalar, configurar y administrar operativamente redes de área local, bajo estándares de calidad, priorizando el uso de software libre. Interpretar modelos de datos Implementar y mantener bases de datos de pequeña y mediana complejidad. Desarrollar algoritmos para manipular bases de datos de pequeña a mediana complejidad. Fomenta la cultura de la innovación para contribuir con la soberanía y seguridad tecnológica.	***Programación*** Programación II ***Ingenieria del Software*** Ingenieria del Software I Base de Datos (BD) ***Soporte Técnico*** Redes Computador ***Cursos*** Matemáticas II Formación Critica II Inglés	***Técnico Superior Universitario o Técnica Superior Universitaria en Informática"*** Plantear alternativas de soluciones ante situaciones y problemas reales, relacionados con soluciones informáticas Diseño y desarrollo de la aplicación informática, acorde al alcance del proyecto Mostrar el módulo funcional realizado

Tabla IV: Atributos del Trayecto III

Competencias Profesionales	Unidades Curriculares	Proyecto III
Desarrollar aplicaciones informáticas basadas en los principios de la ingeniería de software. Aplicar estándares de calidad, usabilidad y accesibilidad en el desarrollo de aplicaciones informáticas. Elaborar la documentación técnica de una aplicación informática. Instalar, configurar y manejar sistemas operativos en equipos de computación. Diseñar bases de datos. Aplicar técnicas estadísticas y de la programación matemática para apoyar la toma de decisiones. Fomenta la cultura de la innovación para contribuir con la soberanía y seguridad tecnológica.	***Ingenieria del Software*** Ingenieria del Software II Modelado de Base de Datos (BD) ***Soporte Técnico*** Sistemas Operativos ***Cursos*** Matemática Aplicada Investigacion de Operaciones Formación Critica III Inglés	***Desarrollador de Aplicaciones*** Diseño y desarrollo de la aplicación informática Seleccionar y justificar la Metodología de desarrollo de software a utilizar en el proyecto socio tecnológico. Discutir acerca de Metodologías de Desarrollo de Software Metodologías Tradicionales Vs Ágiles. Visión general de las distintas metodologías de desarrollo de software. (Rup, Watch, Merinde, Xp, Scrum, Otras). Productos entregables de un proyecto de desarrollo de software

Tabla V: Atributos del Trayecto IV

Competencias Profesionales	Unidades Curriculares	Proyecto IV
Gestionar proyectos informáticos, aplicando estándares reconocidos de calidad y pertinencia social. Aplicar metodologías para realizar auditorías a sistemas informáticos. Administrar bases de datos. Diseñar, implementar y administrar redes informáticas, bajo estándares de calidad, priorizando el uso de software libre. Aplicar los principios básicos de la seguridad informática. Utilizar el idioma inglés para comunicar los resultados de los proyectos desarrollados mediante el uso de la terminología técnica del idioma inglés. Fomenta la cultura de la innovación para contribuir con la soberanía y seguridad tecnológica.	***Ingenieria del Software*** Administración de BD Seguridad Informática Gestión de Proyectos Informáticos Auditoria Informática ***Soporte Técnico*** RedesAvan-zadas ***Cursos*** Investigacion de Operaciones Formación Critica IV Inglés	***"Ingeniero o Ingeniera en Informática"*** Diagnóstico, diseño, ejecución, gestión, implantación y evaluación de proyecto informático bajo estándares de calidad y pertinencia social

B. Propuesta de Gestión de los PST

A continuación se presenta la propuesta para la gestión de los PSTs. En la misma se presenta el objetivo general, los objetivos específicos, la estrategia para la ejecución del proyecto, las etapas del proyecto y la guía para el desarrollo de los proyectos de software.

La propuesta de gestión de PSTs, se basa en aspectos observados durante la ejecución del PNFI a lo largo de los años de su ejecución. Uno de los aspectos a resaltar es que el 85% de los proyectos que se han implementado en el instituto utilizan metodologías de desarrollo hasta la fase de construcción, lo que implica que cuando requiere pasar a producción se generan errores y la mayoría de las veces termina como software inutilizado. Otra situación observada es que la selección de las comunidades es mayoritariamente realizada por los estudiantes. En este sentido, se dificulta el proceso de transición una vez que los estudiantes egresan de la institución, sobre todo en los Trayectos II y IV que proporcionan un título universitario que le permite insertarse directamente en el mercado laboral. La propuesta hace énfasis en que los proyectos deben llegar a la fase de Transición de acuerdo a lo establecido en la Ingeniería del Software.

Se plantea en este aspecto la aplicación de un enfoque sistémico en el cual el equipo de docentes de Proyecto del Departamento de Informática sea el ente que canalice y asigne prioridades para la aceptación y desarrollo de los proyectos. De esta forma, existiría un mayor compromiso de parte de los docentes, los estudiantes y también las comunidades que se atienden y que deben colaborar en todas las fases del proyecto.

A continuación se presenta esquemáticamente la propuesta de gestión:

1) Objetivo General: Gestionar los proyectos Socio-tecnológicos desarrollados por los estudiantes del PNFI del IUT "Dr. Federico Rivero Palacio".

2) Objetivos Específicos:

- Seleccionar los proyectos a gestionar.
- Seleccionar las metodologías en el ámbito tecnológico a aplicar.
- Seguimiento a la Aplicación de la metodología seleccionada para los proyectos.

3) Estrategias para la Ejecución del PST: El Programa Nacional de Formación en Informática (PNFI) establece un número de 6h semanales presenciales para Proyecto Socio-Tecnológico. En este tiempo se debe orientar a los equipos de trabajo de proyecto en el trayecto correspondiente, tomando en cuenta el perfil y las competencias a aplicar, tal como se mostró en el análisis del currículo que se realizó en la fase anterior. Ahora bien, para garantizar la calidad del PST, se ha trabajado con una división interna del proyecto en dos bloques. Un bloque que se ha denominado Proyecto Metodológico, en el cual se realizan las actividades propias de una investigación, tales como: plantear el problema a resolver, los objetivos, las bases teóricas y metodológicas y normas APA, entre otras. El segundo bloque se denominó Proyecto Técnico: en donde se realizan actividades complementarias que refuerzan los conocimientos y competencias técnicas para la ejecución exitosa del PST. Adicionalmente, existe la figura de un especialista técnico o tutor que trabaja en conjunto con el equipo de estudiantes y además se vinculan con la comunidad que requiere el software. La estrategia que se plantea para lograr el objetivo, es emplear los principios establecidos por la Ingeniería del Software y el enfoque de Sistemas y Gestión de Proyectos Informáticos, para realizar el seguimiento del proyecto y asegurar la calidad del producto a entregar a la comunidad que ha solicitado el proyecto desde los inicios del mismo. Esto serviría para potenciar el área de Ingeniería del Software en la formación de los estudiantes del instituto y la gestión propiamente dicha, fomentando la aprehensión de la competencia profesional desde el inicio mismo del proyecto de manera práctica. Así como estrechar los vínculos Comunidad-Universidad necesarios para garantizar un desempeño efectivo del eje de Proyecto.

1) Etapas de la Ejecución del PST: Se propone un esquema para la gestión de PSTs adecuando RUP [5]. En este sentido, se plantean cuatro etapas, que consideran las distintas fases de RUP, dependiendo del trayecto del proyecto y se incorpora una etapa de selección de proyectos, la cual es administrada por el grupo docente encargado de proyectos.

2) Guía para el Desarrollo de Proyectos de Software: Para el desarrollo del software se propone como base la aplicación de una guía de gestión de proyectos de Software. Se hace énfasis en los proyectos de desarrollo de software que constituyen tres cuartos del eje longitudinal PST, la naturaleza de los proyectos de soporte técnico a usuarios y equipos plantean otros tipos de metodologías no presentadas en este documento. Lo cual no implica que las fases de esta guía puedan ser adaptadas para el PST del Trayecto I.

C. Etapas de la Ejecución de los PSTs

Para la etapa de ejecución, se diseñó un cronograma que permite realizar un seguimiento y control dentro de cada etapa.

De esta manera se puede observar en la Tabla VI una estructuración donde para cada uno de los trayectos, se especifica una fecha para realizar el análisis y selección del PST del conjunto de los PSTs, esto con la finalidad de que

pueda convertirse en un posible elegible, la determinación del inicio, diagnóstico o construcción (etapa 2), inventario técnico, elaboración o transición (etapa 3), la ejecución, construcción o transición (etapa 4) y el desarrollo y entrega del producto e informe (etapa 5). En definitiva, la Tabla VI representa un cronograma que permite asignarle tiempo (en 9 meses o 36 semanas) para la realización de cada una de las fases del PST del trayecto correspondiente. Esta tabla, sirve de guía para presentar las diferentes fases en el eje longitudinal. La primera columna contiene las fases previstas, y las columnas sucesivas contienen el tiempo expresado en meses. Una intercepción de una fila con una columna indica el mes en que debería realizarse esa fase para el proyecto especificado. De manera que se puede ubicar, dada una fase de un proyecto de un trayecto dado, el mes en que debería realizarla o dado un mes cuál debería ser el avance en cuanto a las etapas del proyecto de un trayecto dado.

La explicación de cada una de las etapas se expresa posteriormente a la presentación de la Tabla VI.

Tabla VI: Cronograma de Etapas Vs Tiempo

Etapas/Tiempo (en meses)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Ejecución PST Etapa 1	X	X							
Ejecución PST Etapa 2			X						
Ejecución PST Etapa 3				X	X				
Ejecución PST Etapa 4						X	X	X	
Ejecución PST Etapa 5									X

1) Ejecución PST Etapa 1: en esta etapa el objetivo es analizar y seleccionar las comunidades y los proyectos asociados a ellas. Las comunidades provienen del entorno público, departamentos internos, empresas privadas, convenios interinstitucionales y podrán ser propuestas por las autoridades, auto postulándose o por los estudiantes. Se propone establecer al menos dos reuniones con las comunidades.

- 1era reunión: Diagnóstico general: se determinan las necesidades de automatización y/o soporte técnico de la comunidad participante.
- 2da reunión: Presentación de la comunidad. Una vez determinadas las propuestas de proyectos factibles a realizar en ese trayecto académico. Los PSTs ya estarán clasificados por parte de los expertos (profesores especialistas de PST) de acuerdo a las prioridades establecidas ya sea por compromisos de convenios interinstitucionales, necesidades internas a la institución, proyectos de investigación o continuidad de proyectos anteriores, entre otros. Los estudiantes democráticamente escogen el proyecto con el cual sienten afinidad para trabajar.

2) Ejecución PST Etapa 2: esta etapa depende del trayecto en el cual se está realizando el proyecto. De forma tal que, en el trayecto I corresponderá con el Diagnóstico técnico, en el trayecto II y III corresponderá a la fase de inicio o equivalente y en trayecto IV a la fase de construcción o equivalente.

3) Ejecución PST Etapa 3: en el trayecto I corresponde al inventario técnico, en el trayecto II y trayecto III a la fase de elaboración o equivalente y en el trayecto IV a la Auditoría de Sistemas.

4) Ejecución PST Etapa 4: en el trayecto I corresponde a la acción de corregir las fallas del equipo de hardware o proponer la inclusión de componentes nuevos que forman parte de un hardware completo, en el trayecto II y III corresponde a la fase de construcción o equivalente y en el trayecto IV corresponde a la fase de transición.

5) Ejecución PST Etapa 5: esta etapa corresponde a las actividades de cierre académico, tales como entrega de informes, presentación oral, entre otras.

D. Guía para el Desarrollo del PST

Para el desarrollo del software se propone como base la aplicación de una guía de gestión de proyectos de Software. Se hace énfasis en los proyectos de desarrollo de software que constituyen tres cuartos del eje longitudinal PST, la naturaleza de los proyectos de soporte técnico a usuarios y equipos plantean otros tipos de metodologías no presentadas en este documento.

En las fases de desarrollo de un sistema informático, presentadas en [3] se indica que la primera es la Planeación, no obstante al trabajar con proyectos académicos, con grupos integrados por dos o más estudiantes, es importante establecer relaciones de trabajo y organizacionales entre ellos. Según [4], la cultura organizacional tiene una influencia en cómo los proyectos son ejecutados, así mismo en [7] y [8], se plantea la importancia de organizar, definir y conocer el equipo que trabajará en un proyecto determinado. Es por ello que para los proyectos académicos en particular se añade una fase inicial relacionada con el manejo del grupo. Así mismo es importante crear un marco regulatorio de aspectos que deben ser alcanzados, ese marco debe ser flexible y no limitativo, de manera que todos los proyectos tengan un mínimo académico requerido de acuerdo a los perfiles deseados y donde se respete la naturaleza del PST. Este conjunto de requisitos mencionados anteriormente constituyen los procesos de dirección del proyecto. Así mismo en un proyecto académico-comunitario es necesario incorporar una fase de cierre del proyecto donde se incluyen actividades propias de la finalización de un proyecto académico y directamente relacionado con las directrices de evaluación, las cuales no son consideras en este proceso directamente.

La guía de gestión plantea fundamentalmente las etapas siguientes:

- Conformación de grupo
- Análisis preliminar
- Desarrollo de la propuesta de solución
- Presentación de propuesta
- Aplicación de la Metodología de desarrollo de software en las etapas de: Análisis, diseño implementación, pruebas y transición
- Cierre de proyecto.
- Estas etapas se complementan con el desarrollo de los siguientes componentes:
- Proceso de Planificación, fundamental para el control seguimiento y evaluación de los estudiantes participantes del proyecto
- Proceso de control de versiones que permita mantener los desarrollos de software
- Proceso de organización documental, que permita mantener los distintos productos generados en el trabajo en la aplicación de la metodología, separándolos

dependiendo de los actores, por ejemplo la comunidad, el documento de PST, proyecto técnico, entre otros.

A continuación se desarrollan las etapas de la guía de gestión propuesta.

1) La Conformación de Grupo: se refiere a las actividades conducentes a integrar bajo una misión al grupo, cohesionarlos y que entiendan sus ventajas y diferencias, de manera que tengan herramientas para el manejo de conflictos, minimizando el riesgo de ruptura del equipo en algún momento de la ejecución del mismo. En esta etapa, los estudiantes interactúan como equipo y escogen líderes en aspectos técnicos, organizacionales, comunicacionales, entre otros. Además definen su misión, visión, metas y estrategias; como grupo. Esto es importante porque todos los estudiantes deberán rotar y ejercer los diferentes roles presentes en el perfil, a lo largo de la ejecución del proyecto y caracteriza cultura de los miembros del grupo y permite establecer la organización del mismo. En esta etapa los estudiantes definen su primera planificación de actividades generales a desarrollar en el proyecto, a través de un diagrama de Gantt.

2) Análisis Preliminar: para el análisis se prevén las actividades de recolección de información. Análisis del sistema actual, entrevistas con la comunidad, realización de encuestas, entrevistas a los usuarios, determinación de tendencias, indagación de estándares de desarrollo del tipo de aplicaciones que se pretende desarrollar, entre otras que puedan surgir por la naturaleza del proyecto. Una vez culminada esta actividad de recolección se genera la lista de requerimientos. La lista de requerimientos permite formular el modelo del negocio y los casos de uso generales. Adicionalmente el estudiante debe incorporar los requerimientos no funcionales de acuerdo a las normas de calidad y estándares de desarrollo. Esto es fundamental, pues con ellos se generará la propuesta de solución.

3) Desarrollo de la Propuesta de Solución: consiste en determinar en qué consistirá el proyecto, evaluar su factibilidad estimando costos, duración y esfuerzo. Para ello es necesario guiar a los estudiantes a utilizar alguna herramienta de estimación de costo y esfuerzo; se sugiere la utilización de alguna basada en los casos de uso. Aunque generalmente de la estimación pueden despreciarse los costos por ser un PST, las estimaciones del esfuerzo, horas hombre y duración del proyecto resultan importantes para la toma de decisiones en el proyecto y para la planificación. Adicionalmente los estudiantes deben indagar en la comunidad sobre cuál es la tecnología con la que cuentan para el proyecto y con cuál tecnología cuentan los estudiantes para el desarrollo del mismo ya sea propia, de la comunidad o del Instituto o universidad.

4) Presentación de la Propuesta o Anteproyecto ante la Comunidad: en esta actividad los estudiantes presentan a la comunidad el resumen de los casos de uso que determinaron en el análisis preliminar, presentan un prototipo de cómo funcionaría el sistema y la factibilidad, la estimación y los riesgos del desarrollo del mismo. Así mismo, se presenta la planificación del proyecto general y un diagrama PER CPM para mostrar las actividades criticas del mismo. La comunidad puede visualizar la posible solución, la duración, los riesgos, las actividades críticas que puedan poner en riesgo la planificación, entre otras y hacer un nuevo intercambio para fijar el alcance real del proyecto durante el trayecto, que puede ser mayor o menor, de acuerdo a los resultados obtenidos en la estimación. Cuando se habla de menor no significa que no se desarrollará la aplicación completa a la comunidad sino que pueden tomarse otras alternativas, como que el mismo grupo desarrolle un porcentaje de los requerimientos y quede para el próximo trayecto el resto de las funcionalidades, esto en el caso de Trayecto III y IV. En el caso de Trayecto II, se acorta el número de componentes que puede desarrollar. En todo caso los involucrados documentan esta situación mediante un acta de requerimientos.

5) Aplicación de la Metodología de Desarrollo de Software: aun cuando los estudiantes realizan un análisis preliminar a fin de presentar la propuesta, en este punto se comienza a desarrollar como tal el proyecto de software de manera que una vez acordados los requisitos del sistema, el estudiante proceda con las demás actividades planteadas en la metodología realizando el análisis detallado de los requerimientos. La actividad subsiguiente sería proceder a realizar el diseño, generando para ello los artefactos necesarios de acuerdo a la metodología que esté utilizando, diagramas de clase, de interacción, de secuencia, de estado, etc. Todos aquellos que permitan explicar el funcionamiento del sistema. Así mismo se debe diseñar el modelo de datos correspondiente, valiéndose de herramientas como el diagrama de clases, análisis del discurso, entre otros. Esta etapa debe ir acompañada de su respectiva planificación de la fase de diseño del proyecto, la cual incluye las posibles iteraciones según la metodología. Se recomienda que la planificación se haga por roles, es decir cada uno de los estudiantes debe identificar las tareas a desarrollar en cada fase y preferiblemente todos los integrante del equipo deben participar proporcionalmente. También debe incluir la entrega de productos, las reuniones de grupo con el asesor y de ser requerido, con la comunidad. El seguimiento de la planificación en esta fase, mínimo debe hacerse semanalmente y mostrar los avances porcentuales de las actividades.

6) Implementación: basado en que el diseño ha sido óptimo el desarrollo debe concentrarse en codificar los requerimientos e implantar el modelo físico de la base de datos. La codificación conlleva también la revisión con pruebas de caja blanca y caja negra. La verificación del avance puede hacerse de manera individual con el asesor técnico o de forma grupal, donde cada quien muestra su avance, de manera semanal como mínimo. Durante el desarrollo es posible utilizar metodologías ágiles, pues en ese punto se supone el conocimiento pleno de negocio y sus interacciones. Las iteraciones se concentran en resolver los problemas de codificación y avance del proyecto en el menor tiempo posible.

7) Pruebas: Para realizar esta actividad se recomienda que los estudiantes conozcan diferentes softwares de pruebas para requerimientos Funcionales y No funcionales de manera que puedan aplicar las verificaciones según el tipo de sistema. Este proceso genera documentos que se utilizan para planificar las correcciones pertinentes. En esta etapa lo relevante en la planificación es determinar cuáles serán las pruebas a ejecutar y cuales insumos se necesita para ello. Una vez realizadas las pruebas deben generarse los informes técnicos con los hallazgos, de manera que se solventen en tiempos cortos.

8) Transición: en algunos casos los equipos de trabajo realizan sus actividades en ambientes colaborativos y el proyecto se va integrando a medida que se desarrolla, en otros

casos, trabajan de manera separada y es necesario integrar los componentes desarrollados. Una vez culminado el proyecto, es necesario planificar cómo se implantará en la comunidad destino. Es necesario considerar la disposición de la comunidad para permitir a los estudiantes manejar sus activos o involucrar a los mismos en los equipos de trabajo de la organización. De igual forma se prevé la realización de las pruebas de instalación e integración, la migración de datos y el entrenamiento a los usuarios, entre otros. Finalmente se requiere que la comunidad realice las pruebas de aceptación para cada uno de los requerimientos convenidos. En este punto se considera culminado el proyecto y pasa a la fase de producción en la comunidad.

9) Cierre del Proyecto: en este punto pareciera culminado el proyecto, sin embargo, deben realizarse algunas actividades que permitan cerrar el proceso. La relación Comunidad-universidad se verá fortalecido y se preparará para la apertura de nuevos ciclos de proyectos. Dentro de las actividades pautadas. Se presentan entonces los informes técnicos referentes al proceso y los manuales de usuarios. Se planifica la presentación final y la entrega del trabajo final de proyecto. Una vez culminados todos los proyectos es importante considerar la evaluación del grupo docente de proyectos, generación del informe final y presentación ante las autoridades. También se generan candidatos a presentaciones en congresos, eventos científicos y otras actividades académicas, sociales, científicas; relacionadas con los proyectos.

E. Valoración de la Propuesta de Gestión

Se aplicó la propuesta de Gestión de PST a cuatro cohortes comprendidas entre 2013 y 2017, en la gestión diurna del PNFI.

Las peticiones de proyectos se clasificaron en cuatro grupos:

- solicitudes internas a la institución, procedentes de departamentos académicos o dependencias administrativas
- solicitudes procedentes de instituciones públicas del Estado Venezolano, tales como instituciones educativas, Consejos Comunales, Ministerios Públicos, entre otros
- peticiones procedentes de Organizaciones no Gubernamentales (ONG) en ese grupo están generalmente fundaciones sin fines de lucro
- Solicitudes del sector privado, procedente de empresas generalmente desarrolladoras de software o fundaciones privadas.

En total se analizaron un total de 79 proyectos, repartidos en las cuatro cohortes estudiadas.

En la Tabla VII se muestra la cantidad de proyectos asignados a cada tipo de comunidad. Como se observa la distribución de los proyectos es variable, el hecho de no asignar proyectos a un sector no implica que no hubo peticiones del mismo.

En la Tabla VIII, se muestra la distribución según el tipo de comunidad y los trayectos. El mayor número de proyectos se reparte en los trayectos iniciales, dado que en ellos se concentra la mayor cantidad de estudiantes. La disminución que se observa en el número de PST entre el Trayecto II y el Trayecto III, se debe a que algunos estudiantes al obtener el título de TSU, deciden no continuar con la carrera hacia la ingeniería. Por otra parte se observa que en el Trayecto IV, aparece un número de proyectos para el sector privado que no proviene de trayectos anteriores, correspondiente al sector privado, esto surgió por la migración de estudiantes del turno nocturno a la gestión diurna, como era de esperarse continuaron con sus comunidades y proyectos.

Tabla VII: Asignación de Proyectos por Comunidad y Cohorte

Cohortes / Comunidad	2013-2014	2014-2015	2015-2016	2016-2017
Proyectos Internos	9	6	9	10
Proyectos Sector público	8	6	4	9
Fundaciones y ONGs	0	1	1	1
Sector Privado	1	0	9	5
Total	18	13	23	25

En total se asignaron 16 proyectos de soporte técnico al Trayecto I, 29 proyectos de desarrollo de componentes al Trayecto II, que representaban gestiones administrativas sencillas los cuales incluían hasta tres procesos de: creación, reportes, actualización y eliminación. Así mismo se asignaron 17 proyectos de mayor complejidad a equipos de trabajo del Trayecto II y se atendieron proyectos rezagados en Trayecto IV para un total de 17 PSTs.

Tabla VIII: Asignación Según el Tipo de Comunidad y los Trayectos

Trayectos / Comunidad	I	II	III	IV
Proyectos Internos	12	9	7	6
Proyectos Sector Público	0	10	9	8
Fundaciones y ONGs	0	2	1	0
Sector Privado	4	8	0	3
Total	16	29	17	17

En la Tabla IX, aparece el total de proyectos culminados satisfactoriamente, cumpliendo con los requisitos académicos y los requerimientos de la comunidad.

Se observa que en el Trayecto I, solo el 37% de los proyectos se culminó satisfactoriamente. De acuerdo al estudio realizado, esto se debe que los estudiantes abandonan la UC proyecto I. Otro factor es que las instituciones declinan ante la posibilidad de ejecución del PST, sobre todo cuando se trata de soporte técnico a equipos hardware. Es importante resaltar que el total de proyectos culminados en general tienen el 100% de satisfacción en la comunidad y en su mayoría corresponde con actualizaciones de software, instalación del sistema operativo Linux y entrenamiento a usuarios.

Para el Trayecto II, se evidencia que los proyectos internos presentan problemas para la implantación del mismo y por lo tanto queda archivados, aun cuando cumple las expectativas de funcionamiento. Presumiblemente los estudiantes entregan el producto que no ha sido probado exhaustivamente y falla al ponerse en funcionamiento. En algunos casos no se realiza la migración de datos y la comunidad decide seguir usando su sistema ante la imposibilidad de tener el historial requerido. Otro aspecto tiene que ver con los recursos técnicos humanos,

disponibles en la organización, de manera que si no tienen expertos de software, la aplicación queda sin soporte de mantenimiento y por ende cae en desuso lo que finalmente genera insatisfacción. No así para las instituciones privadas que tienen equipos humanos para gestionar los componentes de software que piden realizar, en general éstas tienen un 100% de satisfacción en los componentes recibidos al culminar el proyecto. En general se observó que las comunidades satisfechas regresaron a la institución solicitando mejoras y nuevos requerimientos para el software.

En el Trayecto III, los estudiantes desarrollaron un porcentaje del 70% de la aplicación requerida. Se estableció la entrega de documentos hasta el desarrollo alcanzado y se realizó la presentación de los logros a la comunidad. En general las comunidades quedan satisfechas con los logros alcanzados, al observar los avances del proyecto, aun cuando el software no esté concluido la puesta en marcha del mismo en la comunidad es posible, y esa instalación es tratada como una versión de prueba. Un aspecto negativo es que en caso de que algunos proyectos queden reprobados por incumplimiento por parte de los estudiantes con los requerimientos académicos, la comunidad queda desatendida, hasta una nueva apertura de trayecto.

En el Trayecto IV, se continúan con los proyectos del Trayecto III, en las cohortes de estudio, tres grupos de estudiantes del turno nocturno realizaron el cambio al turno diurno, por ello las cifras coinciden, pero en realidad hubo proyectos que reprobaron o desertaron. No obstante se culminaron el 76% de los proyectos, los cuales cumplieron con todas las fases de la ingeniería del software y la aceptación de la comunidad. En este punto es importante aclarar que un grupo puede culminar el 100% de los requerimientos en el Trayecto III, no obstante debe continuar con el mismo para aplicarle los conceptos referidos a la seguridad, administración de base de datos, auditoria de sistemas, entre otras competencias que adquiere en el último trayecto del Plan de Formación.

Tabla IX: Proyectos Culminados Satisfactoriamente por Trayectos

Trayectos / Comunidad	I	II	III	IV
Proyectos Internos	6	1	7	5
Proyectos Sector Público	0	3	9	5
Fundaciones y ONGs	0	1	1	0
Sector Privado	0	8	0	3
Porcentaje de satisfacción	37,5%	44,8%	17	76,4%

A través de la valoración de la propuesta y los resultados obtenidos, se evidencia que la aplicación de la misma es una adecuada alternativa de solución que permite una vinculación y relación enriquecedora de la Comunidad-Universidad, ya que se logra cumplir con el fin académico y con la construcción de un producto final que queda operativo en la organización. Igualmente se muestra que hay aspectos que pueden mejorarse, como es el manejo de grupo y la gestión de conflictos, entre otros.

La propuesta deja en evidencia que los estudiantes de Trayecto IV, tienen los conocimientos y competencias para lograr construir un producto que quede totalmente operativo y cumplir con los aspectos académicos requeridos. Los proyectos del Trayecto II, solo alcanzan algunas fases de la ingeniería del software y es posible que sea la razón de falla al ser instalado en la comunidad, no obstante funcionan muy bien, cuando la comunidad tiene un equipo de especialistas que tome el componente desarrollado para integrarlos a un sistema. Igualmente los proyectos del Trayecto III, resultan inconclusos con algunos aspectos de calidad del software, pero se instalan como versiones preliminares, hasta completar la versión final en el Trayecto IV. Los proyectos de Trayecto I, finalizados muestran excelentes resultados en la satisfacción de la comunidad, esto se debe a que los estudiantes logran dejar operativos los equipos o diagnosticar fallas que luego puedan ser atendidas por los especialistas.

V. Conclusiones

En este trabajo se presenta una propuesta de Gestión de Proyectos para el eje longitudinal Proyecto Socio Tecnológico. Esta propuesta es el resultado de la adecuación y combinación de la metodología de desarrollo de sistemas RUP y la de gestión de proyectos PMBOK.

La propuesta fue aplicada en las cohortes 2013-2014, 2014-2015, 2015-2016 y 2016-2017 y la valoración de los resultados obtenidos demostró resultados favorables en la culminación de proyectos y el grado de satisfacción del producto en la relación Universidad-Comunidad.

La gestión del PST con miras a satisfacer tanto las necesidades de formación del estudiantado conforme al perfil de competencias expresadas en el PNFI, como las de los beneficiarios del proyecto, implica una negociación previa entre las partes (Universidad- Comunidad), donde se establezcan los alcances de cada PST, de tal forma que el estudiantado en cuestión, solo hará aportes significativos a la comunidad, en función de las competencias exigidas en el pensum de estudio.

Para garantizar que todos los proyectos lleguen a la fase de Transición de acuerdo a lo establecido en la Ingeniería del Software, se estima conveniente que sea el personal docente involucrado en la gestión de los Proyectos quienes canalicen y asignen prioridades en la aceptación y desarrollo de proyectos.

La calidad del PST ejecutado por los estudiantes, implica la puesta en práctica de conocimientos no solo en el ámbito de la Informática, sino además como investigadores. Esto es posible, distribuyendo el tiempo asignado a PST, en dos bloques: Proyecto Metodológico y Proyecto Técnico.

En el análisis curricular se demuestra que los proyectos de software deben cumplir todas las fases de la ingeniería de software y éstas solo pueden ser completadas por estudiantes del Trayecto IV.

Para el Trayecto I, la organización del proyecto condujo a resultados exitosos, ya que todos los proyectos ejecutados satisficieron las necesidades de soporte técnico a usuarios y equipos de la comunidad.

Los proyectos de Trayecto II, solo alcanzan algunas fases de la ingeniería del software y es posible que sea la razón de falla al ser instalado en la comunidad, funcionan muy bien, cuando la comunidad tiene un equipo de especialistas que tome el componente desarrollado para integrarlos a un sistema.

Un aspecto que aún queda abierto es por cuánto tiempo se mantienen los proyectos en funcionamiento en las comunidades, se propone el tema como trabajo futuro para mejorar la propuesta actual.

REFERENCIAS

[1] MPPEU. Ministerio de Poder Popular Para la Educación Universitaria *Plan Nacional de Formación en Informatica*. pp. 11, pp. 14, pp. 22, pp. 28, pp. 29. 2012.

[2] M. Amadio, R. Operetti, J. Tedeski. *Ejes de Formación y Enfoques Curriculares*. UNESCO. Boletin no. 13. Ginebra, Suiza. 2013.

[3] G. Maigua, E. Lopez. *Buenas Prácticas en la Dirección y Gestión de Proyectos Informáticos*. Edutecne. Argentina. 2012.

[4] Project Management Institute. *Guia de los Fundamentos para la Dirección de Proyectos (PMBOK Guide)*. 2 Edition. Estados Unidos. 2008.

[5] I. Jacobson, G. Booch, J. Rumbaugh,. *El Proceso Unificado de Desarrollo de Software*, Addison Wesley. 2000.

[6] C. Amezquita, M. Castañeda. *Plan de Formación y Capacitación del Perfil Profesional Docente del Departamento de Informática*, Departamento de Informática, IUTFRP. 2015.

[7] R. Pressman. *Ingenieria del Software un Enfoque Práctico*. 7 edición. Estados Unidos. McGrawHill. 2012.

[8] A. Weitzenfeld, *Ingeniería de Software Orientada a Objetos con UML, Java e Internet*, México*:* Edamsa impresiones, S.A. de C.V, pp. 68–73. 2008.

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 2, pp. 46-58, Diciembre 2017
Selección de los Mejores Artículos de CoNCISa 2017

Propiedades Algebraicas y Decidibilidad del Transformador de Predicados wp sobre la Teoría de Conjuntos

Federico Flaviani[1]
fflaviani@usb.ve

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

Resumen: En este trabajo se presentan nuevas propiedades algebraicas del transformador de predicado wp sobre los operadores $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\forall$, $\exists$, min y max, demostrados independientemente del lenguaje de programación, usando propiedades generales de la semántica denotacional de los lenguajes. Adicionalmente se muestra un resultado que habla sobre la decidibilidad y cerradura del transformador de predicados wp sobre el lenguaje de programación GCL y usando aserciones escritas en el lenguaje de la teoría de conjuntos de Zermelo-Frankel-Skolem. En este trabajo se muestra que calcular wp de una instrucción de GCL y una aserción escrita en el lenguaje ZFS, es decidible y es otra aserción escrita en ZFS. No necesariamente se puede decidir el valor de verdad de dicha aserción resultante de calcular wp, aun teniendo todos los valores de sus variables libres, por lo que el resultado no contradice la indecidibilidad del problema de la parada.

Palabras Clave: Precondición más Débil; Semántica Denotacional; Decidibilidad; GCL.

Abstract: In this paper, new algebraic properties of the predicate transformer wp are presented on the operators $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\forall$, $\exists$, min and max, demonstrated independently of the programming language, using general properties of the denotational semantics of the languages. Additionally, a result that speaks about the decidability and closure of the predicate transformer wp on the GCL programming language with assertions written in the Zermelo-Frankel-Skolem set theory language is shown. In this paper it is shown that calculating wp of a GCL statement and a written assertion in the ZFS language is decidable and is another written assertion in ZFS. It can not necessarily decide the truth value of said assertion resulting from calculating wp, even if all the values of your free variables are given, so that the result does not contradict the undecidability of the halting problem.

Keywords: Weakest Precondition; Denotational Semantics; Decidability; GCL.

I. Introducción

La lógica de Dijkstra [1] para la corrección de programas se basa en el transformador de predicados wp (weakest precondition), que es básicamente una función sintáctica de dos variables que devuelve de forma simbólica la precondición más débil de una instrucción $inst$ dado una postcondición $Post$ (usando la notación clásica de funciones de dos variables, la notación $wp(inst, Post)$ se refiere al resultado de aplicarle a la función wp, los argumentos $inst$ y $Post$, este resultado es la precondición más débil, simbólicamente hablando, de la instrucción $inst$ con la postcondición $Post$). El uso sucesivo de wp permite ir calculando precondiciones más débiles entre instrucción e instrucción, desde el final del programa hasta el inicio.

Dijkstra en [1] estableció las reglas que definen la función de transformación sintáctica wp según el párrafo siguiente:

Si $B, B_0, \ldots, B_n$ y $S, S_0, \ldots, S_n$ son expresiones booleanas e instrucciones del lenguaje GCL respectivamente, si se abrevia IF y Do como las instrucciones $if\ B_0 \to S_0 [] \ldots [] B_n \to S_n\ fi$ y $do\ B \to S\ od$ respectivamente y si se denota $domain(B_0, \ldots, B_n)$ como un predicado que de satisfacerse en un estado, ninguna de las expresiones B_i, al evaluarse en ese estado, incurren en una operación ilegal (como dividir entre 0), entonces:

- $wp(SKIP, Post) := Post$
- $wp(y_{i_1}, \ldots, y_{i_k} := Exp_1, \ldots, Exp_k, Post) := domain(Exp_1, \ldots, Exp_k) \wedge Post[y_{i_1}, \ldots, y_{i_k} := Exp_1, \ldots, Exp_k]$
- $wp(S_0; S_1, Post) := wp(S_0, wp(S_1, Post))$
- $wp(IF, Post) := domain(B_0, \ldots, B_n) \wedge (B_0 \vee \cdots \vee B_n) \wedge (B_0 \Rightarrow wp(S_0, Post)) \wedge \ldots \wedge (B_n \Rightarrow wp(S_n, Post))$
- $wp(Do, Post) := (\exists k | k \geq 0 : H_k(Post))$ en donde $H_k(Post)$ es un predicado que satisface las

ecuaciones:

$$H_0(Post) \equiv domain(B) \wedge \neg B \wedge Post$$

$$H_k(Post) \equiv$$

$$H_0(Post) \vee (domain(B) \wedge B \wedge wp(S, H_{k-1}(Post)))$$

para $k \geq 1$

La definición recursiva anterior de $H_k(Post)$, tiene la desventaja de que la regla que define a wp para la instrucción Do esta escrita en lógica de segundo orden, por lo que no es aplicable directamente. Es necesario resolver la recurrencia de fórmulas $H_k(Post)$ primero, antes de aplicar la regla de wp para el Do, y también es necesario demostrar que la solución a esa recurrencia se puede escribir en el mismo lenguaje en que esta escrito $Post$, esto con el objetivo de demostrar que la función sintáctica $wp(S, .)$ es cerrada con respecto al lenguaje en que se escribe las aserciones (Es decir que si una postcondición $Post$ esta escrito en cierto lenguaje, entonces $wp(S, Post)$ es equivalente a una fórmula escrita en el mismo lenguaje).

Por otro lado existe un álgebra del transformador de predicado wp sobre los operadores $\wedge$, $\vee$ y $\Rightarrow$, determinada por las siguientes reglas

- $wp(S, false) \equiv false$
- $wp(S, P \wedge Q) \equiv wp(S, P) \wedge wp(S, Q)$
- $wp(S, P \vee Q) \equiv wp(S, P) \vee wp(S, Q)$ si S es una instrucción determinística
- Si $P \Rightarrow Q$, entonces $wp(S, P) \Rightarrow wp(S, Q)$

En este trabajo se demuestran propiedades del álgebra de wp distintas a las anteriores.

A. Contribución

Asumiendo que las aserciones de este trabajo se escriben en el lenguaje de la teoría de conjuntos de Zermelo-Frankel-Skolem, se demuestra la cerradura de $wp(S, .)$ sobre este lenguaje para cualquier instrucción S de GCL. Adicionalmente se muestra que, usando el teorema de las definiciones recursivas de la teoría de conjuntos, si se tiene un predicado $Post$, se muestra de forma constructiva una fórmula en ZFS que es equivalente a $wp(S, Post)$, demostrando que el problema del calculo de wp es decidible sobre el lenguaje ZFS.

El problema de la parada para la instrucción S, es equivalente al problema de conocer el valor de verdad de la aserción $wp(S, true)$ conociendo todos los valores de las variables libres de $wp(S, true)$ (que corresponden a los valores del estado inicial del programa antes de la ejecución). La decidibilidad del cálculo de $wp(S, Post)$ sobre el lenguaje ZFS no es una contradicción a la indecidibilidad del problema de la parada, ya que el problema de conocer el valor de verdad de la aserción resultante de escribir $wp(S, Post)$ en el lenguaje de la teoría de conjuntos de ZFS, teniendo todos los valores de las variables libres de la aserción, no es decidible, ya que la fórmula puede contener combinaciones de cuantificadores $\forall$ y $\exists$ sobre conjuntos infinitos, obligando a que la única forma de conocer el valor de verdad de la aserción sea haciendo una demostración matemática elaborada, y el problema de decidir si una fórmula es un teorema de ZFS, no es decidible.

Por otro lado en este trabajo usando semántica denotacional se demuestran propiedades algebraicas del transformador de predicados wp sobre los operadores $\wedge$ y $\vee$, cuando uno de los predicados en conjunción o disyunción no es modificado o es modificado determinísticamente por $wp(S, .)$. También se demuestran propiedades algebraicas de wp sobre los operadores $\neg$, $\Rightarrow$ $\forall$, $\exists$, min y max.

B. Trabajos Relacionados

Originalmente en [1] la definición recursiva de wp que se expuso al inicio no incluía la función sintáctica $domain$ en sus reglas, esto fue corregido en [2], donde lo incorpora a la regla de wp de la asignación, pero no en las demás reglas como se definió al inicio de la introducción. Una justificación de la incorporación de $domain$ en las reglas de wp del IF y Do, se encuentra en [3], donde se hace una revisión de la semántica denotacional de GCL incluyendo el estado $abort$. La función sintáctica $domain$, aplica sobre expresiones, pero su incorporación en las reglas de construcción de wp del IF y Do, traen dificultades adicionales que no se tenían en [2]. Para manejar estas dificultades, en [4] se definió la función sintáctica $support$, que viene siendo el análogo a $domain$, pero aplica sobre instrucciones en lugar de expresiones.

Gracias a la incorporación de la función sintáctica $support$ en la teoría de wp, es posible demostrar algunas propiedades algebraicas nuevas del transformador de predicados wp, dichas demostraciones se encuentran en este trabajo usando propiedades generales de la semántica denotacional, de esta forma los resultados de estas propiedades algebraicas no dependen del lenguaje de programación usado. Algunas de las propiedades algebraicas de wp que se enuncian aquí, ya fueron enunciadas en [5], en donde se afirma que las demostraciones se pueden realizar usando inducción estructural sobre el tamaño de la instrucción, pero dichas demostraciones no se encuentran en [5].

Por otro lado en [6] se demuestra un teorema de cerradura del transformador de predicados $wp(S, Post)$ sobre el lenguaje de la aritmética de Peano. La demostración muestra una forma constructiva de resolver la recurrencia $H_k(Post)$ usando las funciones β de Gödel, lo cual demuestra que el cálculo de $wp(S, Post)$ es decidible sobre el lenguaje de la aritmética. En [4] se tomó la misma idea de [6] pero resolviendo la recurrencia $H_k(Post)$ usando la versión numerable del teorema de recursión transfinita que es conocido, como el teorema de las definiciones recursivas, sin embargo en [4] no se toma en cuenta el aspecto constructivo del teorema de las definiciones recursivas y por lo tanto no se mostró un resultado de decibilidad del cálculo de wp. En este trabajo se retoma la demostración de [4] considerando los aspectos constructivos de la demostración con el objetivo de demostrar la decidibilidad del cálculo de wp sobre el lenguaje de la teoría de conjuntos de ZFS.

En el área de derivación automática de invariantes ha habido un interés reciente en los últimos años [7]-[15], adicionalmente

existen aplicaciones como [16][17] que pueden calcular invariantes para ciclos donde las expresiones de las asignaciones del cuerpo del ciclo son todas lineales o traducibles a sistemas de transición lineales, de igual forma en [18] se encuentra otra técnica que es aplicable sólo a ciclos donde el cuerpo es traducible a una transformación afín de espacios vectoriales. Aplicaciones basadas en lógica de Hoare y separación tenemos a [19][20][21] y basadas en wp se encuentra [22], sólo que funciona para programas no estructurados.

El resultado sobre la decidibilidad del cálculo de wp abre posibilidades en el campo de la derivación automática de invariantes para algunos casos, ya que una precondición más débil de un ciclo es un invariante. Sin embargo, la precondición más débil obtenida del teorema de decidibilidad, no es en general una aserción cuyo valor de verdad es decidible cuando se tienen todos los valores las variables libres de la aserción, de esta forma no siempre es práctico usar este teorema de decidibilidad para derivar automáticamente un invariante, y más aún si se cuenta con alguna otra técnica, que en el caso en cuestión, pueda calcular un invariante equivalente y decidible (que el valor de verdad de la fórmula sea decidible cuando se tienen todos los valores de las variables libres de la aserción).

C. Estructura del Artículo

A continuación se presentan tres secciones de las cuales, en la primera de ellas se exponen todas las definiciones de semántica denotacional de [3] necesarias para demostrar los teoremas de las siguientes secciones. En la sección siguiente, se demuestran nuevas propiedades algebraicas del transformador de predicados wp. En la última sección se demuestra el teorema que afirma que el cálculo de wp sobre GCL y la teoría de conjuntos es decidible sobre el lenguaje de la teoría de conjuntos de ZFS.

II. Semántica Denotacional de un Lenguaje de Programación

Algunas de las propiedades del transformador de predicados wp que se encuentran en la siguiente sección, se enunciaron por primera vez en [5], donde se afirma que su demostración se hace por inducción estructural sobre el tamaño de la instrucción. Demostrar propiedades por inducción estructural tiene la desventaja de que la demostración depende de las instrucciones que tenga el lenguaje GCL, es decir, si en un futuro se extiende el lenguaje GCL con nuevas instrucciones, entonces sería necesario revisar todas las demostraciones por inducción estructural que se han hecho en la teoría, para incluir los casos correspondientes a las nuevas instrucciones.

Demostraciones independiente al lenguaje de programación son posibles usando semántica denotacional, en donde se consideran sólo las hipótesis generales que tiene una interpretación de una instrucción. A continuación se presenta un resumen de la semántica denotacional para lenguajes de programación propuesta en [3].

Definición (Espacio de Estados del Algoritmo)**.** *Se considera el siguiente algoritmo*

$$
\begin{array}{l}
[Const\ \overline{x} : \overline{T}; \\
\ Var\ \overline{y} : \overline{T'}; \\
\ S \\
]
\end{array}
$$

Donde S es el código del algoritmo, $\overline{x}$ es la lista de constantes del algoritmo y $\overline{T}$ es la lista de tipos de cada $\overline{x}$, $\overline{y}$ es la lista de variables del algoritmo y $\overline{T'}$ es la lista de tipos de cada $\overline{y}$. Si $\overline{T} = T_1, T_2, \ldots, T_n$ y $\overline{T'} = T_{n+1}, T_{n+2}, \ldots, T_{n'}$, entonces se define el espacio de estados del algoritmo anterior como

$$\prod_{i=1}^{n'} T_i$$

Ejemplo. *Se considera el siguiente algoritmo:*

$$
\begin{array}{l}
[Const\ n : Entero; \\
Var\ x : Real; \\
\quad z : Real; \\
S \\
]
\end{array}
$$

Como las constantes y variables del algoritmo son n, x y z de tipos Entero, Real y Real respectivamente, entonces el espacio de estados del algoritmo anterior es $\mathbb{Z} \times \mathbb{R} \times \mathbb{R}$.

Ejemplo. *Se considera el siguiente algoritmo:*

$$
\begin{array}{l}
[Const\ n : Entero; \\
Var\ A : arreglo\ [3..7)\ de\ Reales; \\
\quad z : Real; \\
S \\
]
\end{array}
$$

Como un arreglo de tipo T, es una función de una parte de los enteros a T, entonces el espacio de estados del algoritmo del ejemplo es $\mathbb{Z} \times \mathbb{R}^{[3..7)} \times \mathbb{R}$

Notación. *En un algoritmo con espacio de estados Esp se denotará como $\vec{x}$ a la lista de constantes y variables que se encuentra en el orden en que fueron declaradas, es decir, $\vec{x} = \overline{x}||\overline{y}$, donde $\overline{x}$ y $\overline{y}$ son las listas de constantes y variables de la definición de espacio de estados y $||$ es el operador de concatenación de listas.*

Notación. *Para simplificar la notación, el vector $(\vec{x}) = (\overline{x}, \overline{y})$ se denota simplemente como $\vec{x}$, por lo que la notación $\vec{x}$ puede entenderse según el contexto como una lista de variables de tipo sintáctica $\overline{x}||\overline{y}$, ó como una tupla $(\overline{x}, \overline{y})$.*

Por ejemplo en la fórmula $Post(\vec{x}, \overline{Y})$, la notación $\vec{x}$ se interpreta como lista, queriendo decir $Post(\overline{x}, \overline{y}, \overline{Y})$, en cambio en una fórmula como $\vec{x} \in Esp$, la notación $\vec{x}$ se interpreta como tupla, queriendo decir $(\overline{x}, \overline{y}) \in Esp$.

Definición. *Sea un algoritmo con espacio de estados Esp y se toma un elemento $abort \notin Esp$, entonces se define el espacio de estados extendido al abort como $Esp' := Esp \cup \{abort\}$*

Un algoritmo además de la descripción del espacio de estados consta de frases del lenguaje que se llaman instrucciones y

dentro de las instrucciones se encuentran otras frases llamadas expresiones de tipo T (donde T es un conjunto). Dichas frases se interpretan denotacionalmente con la función de interpretación $\mathcal{E}$. Dicha función de interpretación toma una expresión Exp sintácticamente hablando y devuelve una función con rango en T, que se denota $\mathcal{E}[\![Exp]\!]$.

Definición. *En un algoritmo con espacio de estados Esp, una expresión Exp de tipo T, es una frase del lenguaje que se interpreta como una función*

$$\mathcal{E}[\![Exp]\!] : Dom(\mathcal{E}[\![Exp]\!]) \subseteq Esp \to T.$$

Nota. *Se deja abierta a cualquier posibilidad la sintaxis de las expresiones y el valor de su función de interpretación, ya que la teoría que aquí se desarrolla es válida para cualquier tipo de expresión y función de interpretación.*

Notación. *Dada una expresión Exp dentro de un algoritmo con espacio de estados Esp, se denota como $domain(Exp)$ a una fórmula con variables libres $\vec{x}$, tal que*

$$Dom(\mathcal{E}[\![Exp]\!]) = \{\vec{x} \in Esp | domain(Exp)\}$$

y si $Exp_1, \ldots, Exp_n$ son expresiones del mismo algoritmo se define

$$domain(Exp_1, \ldots, Exp_n) :=$$
$$domain(Exp_1) \wedge \cdots \wedge domain(Exp_n)$$

A continuación se define el concepto de instrucción e interpretación de la misma. Para dar semántica denotacional a las instrucciones, se usará una función de interpretación que se denota $\mathcal{C}$, dicha función recibe una instrucción S sintácticamente hablando y devuelve una interpretación, que se denota como $\mathcal{C}[\![S]\!]$, que no es más que una relación de la teoría de conjuntos.

Definición. *En un algoritmo con espacio de estados Esp una instrucción S es una frase del lenguaje que se interpreta como una relación*

$$\mathcal{C}[\![S]\!] : Esp' \to Esp'$$

tal que el dominio de $\mathcal{C}[\![S]\!]$ es todo Esp',

$$\mathcal{C}[\![S]\!](\{abort\}) = \{abort\}$$

y

$$\overline{x'} = \overline{x} \text{ si } (\overline{x'}, \overline{y'}) \in \mathcal{C}[\![S]\!](\{(\overline{x}, \overline{y})\}) \text{ y } \overline{x'} \in \overline{T}.$$

Ejecutar esta instrucción para unos valores iniciales $\vec{x_0}$ se entiende como evaluar la relación anterior en los valores $\vec{x_0}$ donde la salida de la ejecución pudiera ser cualquier imagen del punto $\vec{x_0}$.

Adicionalmente, se define el soporte $supp(\mathcal{C}[\![S]\!])$ como el conjunto de los estados que no resultan en un $abort$ al ejecutar la instrucción, es decir,

$$supp(\mathcal{C}[\![S]\!]) := \{\vec{x} \in Esp | abort \notin \mathcal{C}[\![S]\!](\{\vec{x}\})\}$$

Nota. *Note que como el dominio de $\mathcal{C}[\![S]\!]$ es todo Esp' entonces, $\mathcal{C}[\![S]\!](\{x\}) \neq \emptyset$ para todo $x \in Esp'$*

Definición. *Si $\overline{x}$ y $\overline{y}$ son la lista de constantes y variables de un algoritmo A en el orden en que fueron declaradas, $\overline{Y}$ una lista de variables distintas a $\overline{x}$ y $\overline{y}$, $\overline{Y_0}$ una lista de valores de tipos iguales a las variables $\overline{Y}$, S una instrucción, $Pre(\overline{x},\overline{y},\overline{Y})$ y $Post(\overline{x},\overline{y},\overline{Y})$ predicados que solo tienen como variables libres a $\overline{x}$, $\overline{y}$ y $\overline{Y}$ y las familias de conjuntos*

$$Dom_{\overline{Y}} := \{(\overline{x}, \overline{y}) \in Esp | Pre(\overline{x}, \overline{y}, \overline{Y})\}$$

y

$$Rgo_{\overline{Y}} := \{(\overline{x}, \overline{y}) \in Esp | Post(\overline{x}, \overline{y}, \overline{Y})\},$$

entonces una tripleta de Hoare dentro del algoritmo A es un predicado que tiene la forma

$$\{Pre(\overline{x}, \overline{y}, \overline{Y_0})\} S \{Post(\overline{x}, \overline{y}, \overline{Y_0})\}$$

y es verdadera si y sólo si

$$\mathcal{C}[\![S]\!](Dom_{\overline{Y_0}}) \subseteq Rgo_{\overline{Y_0}}$$

Notación. *En caso en que Pre y $Post$ no tengan valores fijos instanciando las variables libres $\overline{Y}$, entonces la tripleta*

$$\{Pre(\vec{x}, \overline{Y})\} S \{Post(\vec{x}, \overline{Y})\}$$

es una abreviación de

$$(\forall \overline{Y} | : \{Pre(\vec{x}, \overline{Y})\} S \{Post(\vec{x}, \overline{Y})\}),$$

es decir, que para cada instancia $\overline{Y_0}$ de valores fijos para las variables $\overline{Y}$, se tiene que

$$\{Pre(\vec{x}, \overline{Y_0})\} S \{Post(\vec{x}, \overline{Y_0})\}$$

es verdadera.

Nota. *Como $\overline{Y}$ es una lista de variables ligadas en el predicado $\{Pre(\vec{x}, \overline{Y})\}\ S\{Post(\vec{x}, \overline{Y})\}$ según la notación anterior, entonces el nombre de de éstas no importa y pueden cambiarse según convenga.*

Nota. *Como $Dom_{\overline{Y}} := \{\vec{x} \in Esp | Pre(\vec{x}, \overline{Y})\}$ y $Rgo_{\overline{Y}} := \{\vec{x} \in Esp | Post(\vec{x}, \overline{Y})\}$, entonces para evitar redundancia se convendrá que en $Pre(\vec{x}, \overline{Y})$ y $Post(\vec{x}, \overline{Y})$ no aparece el predicado $\vec{x} \in Esp$, y dicho predicado siempre se tomará como una hipótesis sobrentendida.*

Definición. *Sea R una relación de $A \times B$ y un subconjunto $Rgo \subseteq B$, entonces*

$$M := \{x \in A | R(\{x\}) \neq \emptyset \wedge R(\{x\}) \subseteq Rgo\}$$

se denomina como "dominio máximo de R con rango Rgo".

Lema 1. *Dado un algoritmo con espacio de estados Esp, una instrucción S y un predicado $Post(\vec{x}, \overline{Y})$ con $\vec{x} \in Esp$ y $\overline{Y_0}$ lista de valores del mismo tipo que $\overline{Y}$, entonces el dominio máximo de $\mathcal{C}[\![S]\!]$ y $Rgo_{\overline{Y_0}}$ es igual a*

$$\{\vec{x} \in Esp | \vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge$$
$$(\forall y| : y \in R_S \restriction_{supp(\mathcal{C}[\![S]\!])} (\{\vec{x}\}) \Rightarrow Post(y, \overline{Y_0}))\}$$

Donde $Post(y, \overline{Y_0})$ es una notación simplificada que significa

$$(\exists \overline{y'} | y = (\overline{x}, \overline{y'}) : Post(\overline{x}, \overline{y'}, \overline{Y_0}))$$

Demostración: Ver [3] ■

Definición. *Dado un algoritmo cuyo espacio de estados es Esp con lista de constantes y variables igual a $\vec{x}$ y la tripleta $\{Pre\}S\{Post\}$ tiene como variables libres distintas a las del espacio de estado a $\overline{Y}$, entonces se dice que Pre es una precondición más débil de S y $Post$ si y sólo si $\{\vec{x} \in Esp | Pre(\vec{x}, \overline{Y_0})\}$ es el dominio máximo de la relación $\mathcal{C}[\![S]\!]$ con rango $Rgo_{\overline{Y_0}}$ para cualquier valor $\overline{Y_0}$ de las variables libres $\overline{Y}$.*

Nota. *Según la definición anterior la precondición más débil de la instrucción S y $Post$ es equivalente a*

$$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge (\forall y | : y \in R_S \restriction_{supp(\mathcal{C}[\![S]\!])} (\{\vec{x}\}) \Rightarrow Post(y, \overline{Y}))\}$$

De modo que en el lenguaje de la teoría de conjuntos, el predicado $wp(S, Post(\vec{x}, \overline{Y}))$ es equivalente al predicado anterior.

III. No Determinismo y Propiedades de wp y $support$

En esta sección se introduce la función de tipo sintáctica $support$, que es análoga a la función $domain$ salvo, que $support$ aplica a instrucciones mientras que $domain$ aplica a expresiones. Incluir en la teoría del transformador de predicados wp a la función sintáctica $support$, tiene la ventaja de que con su ayuda, pueden enunciarse propiedades de tipo algebraicas de wp sobre operadores como $\neg$, $\Rightarrow$ min y max.

Los siguientes teoremas son propiedades algebraicas del transformador de predicados wp, los lemas principales serán demostrados usando aserciones escritas en el lenguaje de la teoría de conjuntos de ZFS y usando la fórmula de precondición más débil que se deduce de la semántica denotacional al final de la sección anterior. Dicha fórmula tiene la ventaja de ser independiente del lenguaje de programación, ya que sólo usa el concepto general de interpretación de una instrucción $\mathcal{C}[\![S]\!]$, sin usar hipótesis de como es el comportamiento específico de la instrucción S al ser interpretada.

Lema 2. $wp(S, P \wedge Q) \equiv wp(S, P) \wedge wp(S, Q)$

La interpretación de la instrucción S es una relación $\mathcal{C}[\![S]\!]$, si esta relación es determinística con respecto a las coordenadas $i_1, \ldots, i_k$, entonces la relación se comporta como una función sobre esas coordenadas. Dichas funciones se les pondrá el nombre de "funciones componentes" y se denotará como $\mathcal{C}[\![S]\!]^j$ a la función componente de la coordenada j.

Ejemplo.

$[Const\ x : Real;$
$Var\ y :\ Entero;$
$\quad z :\ Real;$
$if\ y \geq 0\ \rightarrow$
$\ \ y := -y;$
$\ \ z := z/x$
$[]\ y \geq 3\ \rightarrow$
$\ \ y := 3;$
$\ \ z := z/x$
fi
$]$

Denotemos el cuerpo del algoritmo anterior por S. El espacio de estados Esp del algoritmo es $\mathbb{R} \times \mathbb{Z} \times \mathbb{R}$, por lo tanto si $\vec{x} \in Esp$, entonces $\vec{x}$ es de la forma (x, y, z) en donde la tercera coordenada es modificada por $\mathcal{C}[\![S]\!]$ de forma determinística. De esta forma, la tercera coordenada es gobernada por la función componente

$$\mathcal{C}[\![S]\!]^3 : supp(\mathcal{C}[\![S]\!]) \subseteq \mathbb{R} \times \mathbb{Z} \times \mathbb{R} \to \mathbb{R}$$

$$\mathcal{C}[\![S]\!]^3(x, y, z) := \frac{z}{x}$$

y todo elemento perteneciente a $\mathcal{C}[\![S]\!](\{(x, y, z)\})$ con $(x, y, z) \in supp(\mathcal{C}[\![S]\!])$ es de la forma

$$(x, y', \mathcal{C}[\![S]\!]^3(x, y, z))$$

para algún $y' \in \mathbb{Z}$

Lema 3. *Sean $\overline{x}$ y $\overline{y}$ la lista de constantes y variables declaradas en un algoritmo respectivamente y $\overline{Y}$ una lista de variables de especificación. Sea S una instrucción que se comporta determinísticamente sobre los valores de las variables $y_{i_1}, \ldots, y_{i_k}$ de la lista $\overline{y}$. Sea Q un predicado y $P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y})$ un predicado que sólo depende de $\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y}$, entonces*

$$wp(S, P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y}) \vee Q(\overline{x}, \overline{y}, \overline{Y})) \equiv$$

$$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge$$

$$(P(\overline{x}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \overline{Y}) \vee wp(S, Q(\overline{x}, \overline{y}, \overline{Y})))$$

Donde $\mathcal{C}[\![S]\!]^j(\vec{x})$ es la función componente de $\mathcal{C}[\![S]\!]$ en la coordenada j

Demostración: Según la última fórmula de la sección anterior se tiene que la precondición más débil de una instrucción S y la postcondición $Post$ es equivalente a:

$$(\overline{x}, \overline{y}) \in supp(\mathcal{C}[\![S]\!]) \wedge$$

$$(\forall y | : y \in \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x}, \overline{y})\}) \Rightarrow$$

$$(\exists \overline{y'} | y = (\overline{x}, \overline{y'}) : Post(\overline{x}, \overline{y'}, \overline{Y}))).$$

Como la interpretación $\mathcal{C}[\![S]\!]$ de la instrucción S modifica las coordenadas $i_1, \ldots, i_k$ de forma determinística con las funciones componentes $\mathcal{C}[\![S]\!]^{i_j}(\vec{x})$, entonces el predicado anterior es equivalente a:

$$(\overline{x}, \overline{y}) \in supp(\mathcal{C}[\![S]\!]) \wedge$$

$$(\forall y | : y \in \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x}, \overline{y})\}) \Rightarrow$$

$$(\exists y'_1, \ldots, y'_{i_1 - 1}, y'_{i_1 + 1}, \ldots, y'_{i_k - 1}, y'_{i_k + 1}, \ldots, y'_m |$$

$$y = (\overline{x}, y'_1, \ldots, y'_{i_1 - 1}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, y'_{i_k - 1}, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \ldots, y'_m) :$$

$$Post(\overline{x}, y'_1, \ldots, y'_{i_1 - 1} \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, y'_{i_k - 1}, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \ldots, \overline{Y})))$$

Si la postcondición $Post$ es $P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y}) \vee Q(\overline{x}, \overline{y}, \overline{Y})$, entonces la precondición más débil de la instrucción S con esta postcondición es equivalente a:

$$(\overline{x},\overline{y}) \in supp(\mathcal{C}[\![S]\!])\wedge$$
$$(\forall y| : y \in \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x},\overline{y})\}) \Rightarrow$$
$$(\exists y'_1,\ldots,y'_{i_1-1},y'_{i_1+1},\ldots,y'_{i_k-1},y'_{i_k+1},\ldots,y'_m|$$
$$y = (\overline{x},y'_1,\ldots,y'_{i_1-1},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,y'_{i_k-1},\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\ldots,y'_m) :$$
$$P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y})\vee$$
$$Q(\overline{x},y'_1,\ldots,\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),y'_{i_1+1},\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),y'_{i_k+1},\ldots,\overline{Y})))$$

$\equiv$

$$(\overline{x},\overline{y}) \in supp(\mathcal{C}[\![S]\!])\wedge$$
$$(\forall y| : y \in \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x},\overline{y})\}) \Rightarrow$$
$$P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y})\vee$$
$$(\exists y'_1,\ldots,y'_{i_1-1},y'_{i_1+1},\ldots,y'_{i_k-1},y'_{i_k+1},\ldots,y'_m|$$
$$y = (\overline{x},y'_1,\ldots,y'_{i_1-1},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,y'_{i_k-1},\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\ldots) :$$
$$Q(\overline{x},y'_1,\ldots,y'_{i_1-1},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,y'_{i_k-1},\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\ldots,\overline{Y})))$$

$\equiv$

$$(\overline{x},\overline{y}) \in supp(\mathcal{C}[\![S]\!])\wedge$$
$$(\forall y| : y \in \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x},\overline{y})\}) \Rightarrow$$
$$P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y})\vee$$
$$(\exists\overline{y'}|y = (\overline{x},\overline{y'}) : Q(\overline{x},\overline{y'},\overline{Y})))$$

$\equiv$

$$(\overline{x},\overline{y}) \in supp(\mathcal{C}[\![S]\!])\wedge$$
$$(\forall y| : y \notin \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x},\overline{y})\})\vee$$
$$P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y})\vee$$
$$(\exists\overline{y'}|y = (\overline{x},\overline{y'}) : Q(\overline{x},\overline{y'},\overline{Y})))$$

$\equiv$

$$(\overline{x},\overline{y}) \in supp(\mathcal{C}[\![S]\!])\wedge$$
$$(P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y})\vee$$
$$(\forall y| : y \notin \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x},\overline{y})\})\vee$$
$$(\exists\overline{y'}|y = (\overline{x},\overline{y'}) : Q(\overline{x},\overline{y'},\overline{Y}))))$$

$\equiv$

$$(\overline{x},\overline{y}) \in supp(\mathcal{C}[\![S]\!])\wedge$$
$$(P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y})\vee$$
$$(\forall y| : y \in \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x},\overline{y})\}) \Rightarrow$$
$$(\exists\overline{y'}|y = (\overline{x},\overline{y'}) : Q(\overline{x},\overline{y'},\overline{Y}))))$$

$\equiv< p \wedge (q \vee r) \equiv p \wedge (q \vee (p \wedge r)) >$

$$(\overline{x},\overline{y}) \in supp(\mathcal{C}[\![S]\!]) \wedge (P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y})\vee$$
$$((\overline{x},\overline{y}) \in supp(\mathcal{C}[\![S]\!])\wedge(\forall y| : y \in \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x},\overline{y})\}) \Rightarrow$$
$$(\exists\overline{y'}|y = (\overline{x},\overline{y'}) : Q(\overline{x},\overline{y'},\overline{Y})))))$$

$\equiv$<Lema 1>

$$(\overline{x},\overline{y}) \in supp(\mathcal{C}[\![S]\!])\wedge$$
$$(P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y}) \vee wp(S,Q(\overline{x},\overline{y},\overline{Y})))$$

■

Lema 4. *Sean $\overline{x}$ y $\overline{y}$ la lista de constantes y variables declaradas en un algoritmo respectivamente y $\overline{Y}$ una lista de variables de especificación. Sea S una instrucción que se comporta determinísticamente sobre los valores de las variables $y_{i_1},\ldots,y_{i_k}$ de la lista $\overline{y}$. Sea $P(\overline{x},y_{i_1},\ldots,y_{i_k},\overline{Y})$ un predicado que sólo depende de $\overline{x},y_{i_1},\ldots,y_{i_k},\overline{Y}$, entonces*

$$wp(S,P(\overline{x},y_{i_1},\ldots,y_{i_k},\overline{Y})) \equiv$$
$$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y})$$

Donde $\mathcal{C}[\![S]\!]^j(\vec{x})$ es la función componente de $\mathcal{C}[\![S]\!]$ en la coordenada j

Demostración: Inmediato tomando Q como $false$ en el Lema 3 y usando $wp(S,false) \equiv false$ ■

Lema 5. *Sea S una instrucción (determinística o no) tal que se comporta de forma determinística sobre las variables del predicado $P \vee Q$, entonces*

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

Demostración: Si P depende de $\overline{x}$, $\overline{Y}$ y de las variables $y_{i_1},\ldots,y_{i_k}$ y Q depende de $\overline{x}$, $\overline{Y}$ y de las variables $y_{j_1},\ldots,y_{j_{k'}}$, entonces:

$$wp(S,P(\overline{x},y_{i_1},\ldots,y_{i_k},\overline{Y}) \vee Q(\overline{x},y_{j_1},\ldots,y_{j_{k'}},\overline{Y}))$$

$\equiv$<Lema 4>

$$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge (P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y})\vee$$
$$Q(\overline{x},\mathcal{C}[\![S]\!]^{j_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{j_{k'}}(\vec{x}),\overline{Y}))$$
$\equiv$
$$(\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge P(\overline{x},\mathcal{C}[\![S]\!]^{i_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{i_k}(\vec{x}),\overline{Y}))\vee$$
$$(\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge Q(\overline{x},\mathcal{C}[\![S]\!]^{j_1}(\vec{x}),\ldots,\mathcal{C}[\![S]\!]^{j_{k'}}(\vec{x}),\overline{Y}))$$

$\equiv$<Lema 4>

$$wp(S,P) \vee wp(S,Q)$$ ■

Lema 6. *Sea P un predicado y S una instrucción que no modifica los valores de las variables de P, entonces*

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

donde $support(S)$ es un predicado que depende de las constantes y variables declaradas en el programa, tal que un

estado lo satisface si y sólo si la instrucción S no aborta al ser ejecutado en dicho estado.

Por ejemplo $true$ es un predicado que para cualquier S, se tiene que S no modifica sus variables, por lo que una forma de calcular $support(S)$ es calculando $wp(S, true) \equiv support(S) \wedge true \equiv support(S)$. Por ejemplo si S es la instrucción

$$
\begin{array}{l}
if\ a > -3 \rightarrow \\
\quad b := b/a \\
[]\ a \leq -3 \rightarrow \\
\quad b := 2 \\
fi,
\end{array}
$$

entonces

$$
\begin{array}{l}
wp(S, true) \\
\equiv \\
(a > -3 \Rightarrow domain(b/a) \wedge true[b := b/a]) \wedge \\
(a \leq -3 \Rightarrow true[b := 2]) \\
\equiv \\
(a > -3 \Rightarrow a \neq 0) \wedge true
\end{array}
$$

Con lo que $support(S) \equiv a > -3 \Rightarrow a \neq 0$.

A continuación se demostrará el Lema 6.

Demostración: Si P depende de $\overline{x}$, $\overline{Y}$ y de las variables $y_{i_1}, \ldots, y_{i_k}$, entonces

$wp(S, P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y}))$

$\equiv$<Lema 4>

$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge P(\overline{x}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \overline{Y})$

Como la instrucción S no modifica los valores de las variables $y_{i_1}, \ldots, y_{i_k}$, entonces las funciones componentes $\mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x})$ son funciones identidad y por lo tanto la expresión anterior es equivalente a:

$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y})$

$\equiv$<Notación>

$support(S) \wedge P$ ■

Lema 7. $wp(S, P) \Rightarrow support(S)$

Demostración:
$wp(S, P)$

$\equiv$<Lema 1>

$$
\begin{array}{l}
\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge \\
\qquad (\forall y| : y \in R \restriction_{supp(\mathcal{C}[\![S]\!])} (\{\vec{x}\}) \Rightarrow P(y, \overline{Y}))
\end{array}
$$

$\Rightarrow$<Debilitamiento>

$\vec{x} \in supp(\mathcal{C}[\![S]\!])$

$\equiv$<Notación>

$support(S)$ ■

Lema 8. *Sean P y Q predicados y S una instrucción que se comporta determinísticamente sobre los valores de las variables de P, entonces*

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

Demostración: Si P depende de $\overline{x}$, $\overline{Y}$ y de las variables $y_{i_1}, \ldots, y_{i_k}$, entonces

$wp(S, P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y}) \vee Q(\overline{x}, \overline{y}, \overline{Y}))$

$\equiv$<Lema 3>

$$
\begin{array}{c}
\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge \\
(P(\overline{x}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \overline{Y}) \vee wp(S, Q(\overline{x}, \overline{y}, \overline{Y})))
\end{array}
$$

$\equiv$

$$
\begin{array}{c}
(\overline{x}, \overline{y}) \in supp(\mathcal{C}[\![S]\!]) \wedge \\
(((\overline{x}, \overline{y}) \in supp(\mathcal{C}[\![S]\!]) \wedge P(\overline{x}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \overline{Y})) \vee \\
wp(S, Q(\overline{x}, \overline{y}, \overline{Y})))
\end{array}
$$

$\equiv$<Lema 4>

$$
\begin{array}{c}
(\overline{x}, \overline{y}) \in supp(\mathcal{C}[\![S]\!]) \wedge \\
(wp(S, P(\overline{x}, \overline{y}, \overline{Y})) \vee wp(S, Q(\overline{x}, \overline{y}, \overline{Y})))
\end{array}
$$

$\equiv$<Notación>

$$
\begin{array}{c}
support(S) \wedge \\
(wp(S, P(\overline{x}, \overline{y}, \overline{Y})) \vee wp(S, Q(\overline{x}, \overline{y}, \overline{Y})))
\end{array}
$$

■

Lema 9. *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

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

Demostración:
$wp(S, P \wedge Q)$

$\equiv$<Lema 2>

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

$\equiv$<Lema 6>

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

$\equiv$<Lema 7>

$P \wedge wp(S, Q)$

Por otro lado si S no modifica los valores de las variables de P, entonces S se comporta determinísticamente sobre los valores de las variables de P, por lo tanto se puede aplicar el Lema 8 de la siguiente manera:

$wp(S, P \vee Q)$

$\equiv$<Lema 8>

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

$\equiv$<Lema 6>

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

Lema 10. *Sea P un predicado y S una instrucción que se comporta determinísticamente sobre los valores de las variables de P, entonces*

$$wp(S, \neg P) \equiv support(S) \wedge \neg wp(S, P)$$

Demostración: Si P depende de $\overline{x}$, $\overline{Y}$ y de las variables $y_{i_1}, \ldots, y_{i_k}$, entonces:

$wp(S, \neg P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y}))$

$\equiv$<Lema 4>

$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge \neg P(\overline{x}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \overline{Y})$

$\equiv$<Absorción>

$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge (\vec{x} \notin supp(\mathcal{C}[\![S]\!]) \vee$
$\neg P(\overline{x}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \overline{Y}))$
$\equiv$
$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge \neg(\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge$
$P(\overline{x}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \overline{Y}))$

$\equiv$<Lema 4>

$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge \neg wp(S, P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y}))$

$\equiv$<Notación>

$support(S) \wedge \neg wp(S, P)$ ■

Lema 11. *Sean P y Q predicados y S una instrucción que se comporta determinísticamente sobre los valores de las variables de P, entonces*

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

Demostración: Como S es una instrucción que se comporta determinísticamente sobre los valores de las variables de P, entonces S es una instrucción que se comporta determinísticamente sobre los valores de las variables de $\neg P$, entonces:

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

$\equiv$<Lema 8>

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

$\equiv$<Lema 10>

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

Lema 12. *Sean P y Q predicados y S una instrucción que no modifica los valores de las variables de P, entonces*

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

Demostración: Como S es una instrucción que no modifica los valores de las variables de P, entonces S es una instrucción que se comporta determinísticamente sobre las variables de P, entonces:

$wp(S, P \Rightarrow Q)$

$\equiv$<Lema 11>

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

$\equiv$<Lema 6>

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

Lema 13. *Sea P un predicado, S una instrucción que se comporta determinísticamente sobre los valores de las variables de P, y ϵ una variable no declarada en el programa, entonces*

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

Demostración: Si P depende de $\overline{x}$, $\overline{Y}$ y de las variables $y_{i_1}, \ldots, y_{i_k}$, entonces:

$wp(S, (\exists \epsilon | : P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y})))$

$\equiv$<Lema 4>

$\vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge$
$(\exists \epsilon | : P(\overline{x}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \overline{Y}))$

$\equiv <$ ϵ no ocurre en el vector de variables declaradas $\vec{x}$ $>$

$(\exists \epsilon | : \vec{x} \in supp(\mathcal{C}[\![S]\!]) \wedge$
$P(\overline{x}, \mathcal{C}[\![S]\!]^{i_1}(\vec{x}), \ldots, \mathcal{C}[\![S]\!]^{i_k}(\vec{x}), \overline{Y}))$

$\equiv$<Lema 4>

$(\exists \epsilon | : wp(S, P(\overline{x}, y_{i_1}, \ldots, y_{i_k}, \overline{Y})))$ ■

Lema 14. *Sean P y R predicados, S una instrucción que se comporta determinísticamente sobre los valores de las*

variables de P y no modifica los valores de las variables de R, y ϵ una variable no declarada en el programa, entonces

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

Demostración: Como S no modifica los valores de las variables de R, entonces S se comporta determinísticamente sobre los valores de las variables de R y como adicionalmente S se comporta determinísticamente sobre los valores de las variables de P, entonces S se comporta determinísticamente sobre los valores de las variables de $R \wedge P$.

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

$\equiv$<Lema 13>

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

$\equiv$<Lema 9>

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

Lema 15. *Sea S una instrucción, P un predicado y ϵ una variable no declarada en el programa, entonces*

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

Demostración: Dado que $\mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])}$ es una relación, es fácil demostrar que la fórmula de $wp(S, P)$ del final de la sección anterior es equivalente a

$$(\overline{x}, \overline{y}) \in supp(\mathcal{C}[\![S]\!]) \wedge$$
$$(\forall\overline{y'}| : (\overline{x}, \overline{y'}) \in \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x}, \overline{y})\}) \Rightarrow$$
$$P(\overline{x}, \overline{y'}, \overline{Y}))$$

Como ϵ no ocurre en la lista de variables libres $\overline{x}, \overline{y}$, entonces de la fórmula $(\forall\epsilon| : wp(S, P))$ se puede sacar del $\forall\epsilon$ el lado izquierdo del $\wedge$, el $\forall\overline{y'}$ y el antecedente de $\Rightarrow$ quedando

$$(\overline{x}, \overline{y}) \in supp(\mathcal{C}[\![S]\!]) \wedge$$
$$(\forall\overline{y'}| : (\overline{x}, \overline{y'}) \in \mathcal{C}[\![S]\!] \restriction_{supp(\mathcal{C}[\![S]\!])} (\{(\overline{x}, \overline{y})\}) \Rightarrow$$
$$(\forall\epsilon| : P(\overline{x}, \overline{y'}, \overline{Y})))$$

$\equiv$

$$wp(S, (\forall\epsilon| : P))$$

■

Lema 16. *Sean P y R predicados, S una instrucción que no modifica los valores de las variables de R, y ϵ una variable no declarada en el programa. Si $(\exists\epsilon| : R) \equiv true$, entonces*

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

Demostración:
$wp(S, (\forall\epsilon|R : P))$
$\equiv$
$wp(S, (\forall\epsilon| : \neg R \vee P))$

$\equiv$<Lema 15>

$(\forall\epsilon| : wp(S, \neg R \vee P))$

$\equiv$<Lema 9>

$(\forall\epsilon| : support(S) \wedge (\neg R \vee wp(S, P)))$
$\equiv$
$support(S) \wedge (\forall\epsilon| : \neg R \vee wp(S, P))$
$\equiv$
$support(S) \wedge (\forall\epsilon|R : wp(S, P))$

$\equiv$< $(\exists\epsilon| : R) \equiv true$ >

$(\forall\epsilon|R : support(S) \wedge wp(S, P))$

$\equiv$<Lema 7>

$(\forall\epsilon|R : wp(S, P))$ ■

Lema 17. *Sea R un predicado y S una instrucción que se comporta determinísticamente sobre los valores de las variables de R. Si i_f es una variable no declarada en el programa y ϵ es una expresión en donde S no modifica los valores de sus variables, entonces*

$$wp(S, \epsilon \neq (min\ i_f|R : i_f)) \equiv$$
$$support(S) \wedge \epsilon \neq (min\ i_f|wp(S, R) : i_f)$$

y

$$wp(S, \epsilon = (min\ i_f|R : i_f)) \equiv$$
$$support(S) \wedge \epsilon = (min\ i_f|wp(S, R) : i_f)$$

Demostración: $\epsilon \neq (min\ i_f|R : i_f)$ es equivalente a

$(\neg(\exists i_f| : R) \Rightarrow \epsilon \neq \infty) \wedge$
$((\exists i_f| : R) \Rightarrow \neg(R[i_f := \epsilon]) \vee (\exists i_f| : R \wedge i_f < \epsilon))$.

De modo que:

$wp(S, \epsilon \neq (min\ i_f|R : i_f))$

$\equiv$< $wp(S, P) \Rightarrow support(S)$ para cualquier P >

$support(S) \wedge wp(S, \epsilon \neq (min\ i_f|R : i_f))$
$\equiv$
$support(S) \wedge wp(S, (\neg(\exists i_f| : R) \Rightarrow \epsilon \neq \infty) \wedge$
$((\exists i_f| : R) \Rightarrow \neg(R[i_f := \epsilon]) \vee (\exists i_f| : R \wedge i_f < \epsilon)))$

$\equiv$<Lema 2>

$support(S) \wedge wp(S, \neg(\exists i_f| : R) \Rightarrow \epsilon \neq \infty) \wedge$
$wp(S, (\exists i_f| : R) \Rightarrow \neg(R[i_f := \epsilon]) \vee (\exists i_f| : R \wedge i_f < \epsilon))$

$\equiv$<Lema 11>

$support(S) \wedge (wp(S, \neg(\exists i_f| : R)) \Rightarrow wp(S, \epsilon \neq \infty)) \wedge$
$(wp(S, (\exists i_f| : R)) \Rightarrow wp(S, \neg(R[i_f := \epsilon]) \vee$

$$(\exists i_f| : R \wedge i_f < \epsilon)))$$

$\equiv$<Lema 8>

$$support(S) \wedge (wp(S, \neg(\exists i_f| : R)) \Rightarrow wp(S, \epsilon \neq \infty)) \wedge$$
$$(wp(S, (\exists i_f| : R)) \Rightarrow wp(S, \neg(R[i_f := \epsilon])) \vee$$
$$wp(S, (\exists i_f| : R \wedge i_f < \epsilon)))$$

$\equiv$<Lema 10>

$$support(S) \wedge (\neg wp(S, (\exists i_f| : R)) \Rightarrow wp(S, \epsilon \neq \infty)) \wedge$$
$$(wp(S, (\exists i_f| : R)) \Rightarrow \neg wp(S, R[i_f := \epsilon]) \vee$$
$$wp(S, (\exists i_f| : R \wedge i_f < \epsilon)))$$

$\equiv$<Lemas 13>

$$support(S) \wedge (\neg(\exists i_f| : wp(S, R)) \Rightarrow wp(S, \epsilon \neq \infty)) \wedge$$
$$((\exists i_f| : wp(S, R)) \Rightarrow \neg wp(S, R[i_f := \epsilon]) \vee$$
$$(\exists i_f| : wp(S, R \wedge i_f < \epsilon)))$$

$\equiv$<Lema 2>

$$support(S) \wedge (\neg(\exists i_f| : wp(S, R)) \Rightarrow wp(S, \epsilon \neq \infty)) \wedge$$
$$((\exists i_f| : wp(S, R)) \Rightarrow \neg wp(S, R[i_f := \epsilon]) \vee$$
$$(\exists i_f| : wp(S, R) \wedge wp(S, i_f < \epsilon)))$$

$\equiv$<Lema 6>

$$support(S) \wedge (\neg(\exists i_f| : wp(S, R)) \Rightarrow \epsilon \neq \infty) \wedge$$
$$((\exists i_f| : wp(S, R)) \Rightarrow \neg wp(S, R[i_f := \epsilon]) \vee$$
$$(\exists i_f| : wp(S, R) \wedge i_f < \epsilon))$$

$\equiv$< S no modifica las variables de ϵ y i_f >

$$support(S) \wedge (\neg(\exists i_f| : wp(S, R)) \Rightarrow \epsilon \neq \infty) \wedge$$
$$((\exists i_f| : wp(S, R)) \Rightarrow \neg(wp(S, R)[i_f := \epsilon]) \vee$$
$$(\exists i_f| : wp(S, R) \wedge i_f < \epsilon))$$
$\equiv$
$$support(S) \wedge \epsilon \neq (min\ i_f | wp(S, R) : i_f)$$

Por otro lado

$$wp(S, \epsilon = (min\ i_f | R : i_f))$$
$\equiv$
$$wp(S, \neg(\epsilon \neq (min\ i_f | R : i_f)))$$

$\equiv$<Lema 10>

$$support(S) \wedge \neg wp(S, \epsilon \neq (min\ i_f | R : i_f))$$

$\equiv$<Resultado anterior>

$$support(S) \wedge \neg(support(S) \wedge \epsilon \neq (min\ i_f | wp(S, R) : i_f))$$
$\equiv$
$$support(S) \wedge (\neg support(S) \vee \epsilon = (min\ i_f | wp(S, R) : i_f))$$

$\equiv$<Absorción>

$$support(S) \wedge \epsilon = (min\ i_f | wp(S, R) : i_f)$$ ■

Lema 18. *Sea R un predicado y S una instrucción que se comporta determinísticamente sobre los valores de las variables de R. Si i_f es una variable no declarada en el programa y ϵ es una expresión en donde S no modifica los valores de sus variables, entonces*

$$wp(S, \epsilon \neq (max\ i_f | R : i_f)) \equiv$$

$$support(S) \wedge \epsilon \neq (max\ i_f | wp(S, R) : i_f)$$

y

$$wp(S, \epsilon = (max\ i_f | R : i_f)) \equiv$$

$$support(S) \wedge \epsilon = (max\ i_f | wp(S, R) : i_f)$$

Demostración: La demostración es análoga a la del Lema 17 pero sustituyendo la expresión $i_f < \epsilon$ por $i_f > \epsilon$. ■

Nota. *las demostraciones de estas propiedades con aserciones escritas en el lenguaje de la teoría de conjuntos de ZFS, implican la veracidad de las mismas en un fragmento de un lenguaje que incluya la teoría de ZFS y que sea cerrado sobre los operadores $\wedge$, $\vee$, $\neg$, $\Rightarrow$ $\forall$, $\exists$, min, max y $wp(S, .)$.*

Por ejemplo el lenguaje de las aserciones decidibles de [2] (aserciones donde conocer el valor de verdad, si se tienen todos los valores de las variables libres de la aserción, es decidible) es tal, que se puede extender para poder escribir todos los axiomas de ZFS consistentemente, a modo que el lenguaje de [2] pudiera entenderse, como un fragmento de ZFS. En este lenguaje de [2] las propiedades de esta sección son ciertas para toda instrucción S tal que $wp(S, .)$ sea cerrado, ya que el fragmento es cerrado sobre $\wedge$, $\vee$, $\neg$, $\Rightarrow$ $\forall$, $\exists$, min y max.

Para mostrar la afirmación de la nota anterior para el Lema 5 se supone que S es una instrucción que se comporta determinísticamente sobre las variables de P y Q, que son predicados escritos en el lenguaje de las aserciones decidibles de [2] y $wp(S, .)$ es cerrado en ese lenguaje. Como puedo escribir consistentemente los axiomas de ZFS con la sintaxis del lenguaje de [2], entonces este lenguaje puede entenderse como un fragmento de ZFS y por lo tanto el lema 5 es cierto obteniendo

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

$\equiv$<Lema 5>

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

Como $wp(S, .)$ es cerrado sobre el lenguaje de [2], entonces $wp(S, P)$ y $wp(S, Q)$ son fórmulas del lenguaje de [2], y como dicho lenguaje es cerrado sobre el operador $\vee$, entonces $wp(S, P) \vee wp(S, Q)$ es una fórmula en el lenguaje de [2]. Esto demuestra que el lema 5 es cierto si se restringen las aserciones a este lenguaje y $w(S, .)$ es cerrado en el fragmento.

IV. Cerradura y Decidibilidad del Cálculo de wp

Del lenguaje y axiomatización de la teoría de conjuntos de ZFS no es directo que todo conjunto definido recursivamente exista, sin embargo todo conjunto que quiera definirse de forma recursiva, puede definirse con una fórmula en el lenguaje de ZFS que es equivalente a la recursión inicial. El proceso de conseguir la fórmula del lenguaje de primer orden de ZFS, que define equivalentemente el conjunto que inicialmente se encontraba definido recursivamente, se conoce como "meta-teorema de recursión transfinita". Dado una definición de un conjunto hecho de forma recursiva, dicho metateorema muestra de forma constructiva, cuál es la fórmula dentro del lenguaje de ZFS, que define al mismo conjunto.

Una demostración detallada del metateorema de recursión transfinita se encuentra en [23], sin embargo es más general de lo que se necesita para esta sección, ya que es válido para hacer recursión sobre cualquier ordinal. En esta sección se usará una versión de dicho teorema restringido a ω, cuyo enunciado es:

Teorema 1. *Si se tiene un predicado φ tal que satisface $(\forall k, F| : (\exists! y| : \varphi(k, F, y)))$. Definiendo $G(k, F)$ como el único y tal que $\varphi(k, F, y)$. Entonces se puede escribir una fórmula ψ donde lo siguiente es demostrable:*

1) $(\forall k| : (\exists! y| : \psi(k, y)))$*, es decir ψ define una función F tal que $\psi(k, F(k))$*
2) $(\forall k | k \in \omega| : F(k) = G(k, F \restriction_{k-1}))$

Una explicación verbosa del teorema anterior sería que la expresión $F(k) = G(k, F \restriction_{k-1})$ es una definición recursiva del conjunto $F(k)$ y la fórmula $\psi(k, y)$ es una versión en el lenguaje de ZFS, que define por comprensión un conjunto y que viene siendo igual $F(k)$. La idea de la demostración es la siguiente:

Demostración: Se define $\psi(k, y)$ como

$$(k \notin \omega \wedge y = \emptyset) \vee (k \in \omega \wedge (\exists d, h | App(d, h) : k \in d \wedge h(k) = y))$$

En donde $App(d, h)$ es un predicado definido como

$$esFuncion(h) \wedge$$
$$d = Dom(h) \subseteq \omega \wedge (\forall m)(m \in d \Rightarrow m - 1 \subseteq d) \wedge$$
$$(\forall m)(m \in d \Rightarrow \varphi(m, h \restriction_{m-1}, h(m)))$$

El resto de la demostración consiste en demostrar $(\exists! y| : \psi(k, y))$ por inducción fuerte y luego que la función $F(k)$ existe usando el axioma de reemplazo. ∎

Nota. *Note que la demostración del teorema anterior dice que si se tiene la fórmula del predicado φ, entonces se tiene la fórmula para $\psi(k, y)$, que es una fórmula escrita en el lenguaje de primer orden de la teoría de conjuntos de ZFS. Toda ocurrencia de un conjunto $F(k)$ definido recursivamente en algún predicado $P(\dots, F(k), \dots)$ dentro de una fórmula de ZFK, se entiende como una abreviación de la fórmula $(\forall R | \psi(k, R) : P(\dots, R, \dots))$*

Teorema 2. *Si el lenguaje que se usa para escribir los predicados de las aserciones en GCL, es el lenguaje de primer orden de la teoría de conjuntos de Zermelo-Frankel-Skolem, entonces por cada caso particular de predicado $Post$, de expresión B_0 e instrucción S_0, existe una fórmula escrita en el lenguaje de primer orden de la teoría de conjuntos de Zermelo-Framkel-Skolem, que es equivalente a $wp(do\ B_0 \to S_0\ od, Post)$*

Demostración: Se definen recursivamente las siguientes instrucciones

$$If := if B_0 \to S_0 [] \neg B_0 \to SKIP fi$$
$$Do_0 := if \neg B_0 \to SKIP fi$$
$$Do_{k+1} := If; Do_k.$$

Como la interpretación de una secuenciación de instrucciones es la composición de las interpretaciones, entonces la interpretación de la instrucción Do_k satisface la siguiente recurrencia:

$$\mathcal{C}[\![Do_0]\!] := \mathcal{C}[\![if \neg B_0 \to SKIP fi]\!]$$

$$\mathcal{C}[\![Do_{k+1}]\!] := \mathcal{C}[\![Do_k]\!] \circ \mathcal{C}[\![If]\!]$$

Por otro lado la fórmula definida por

$$\varphi(k, F, y) :=$$
$$(k = 0 \wedge y = \mathcal{C}[\![Do_0]\!]) \vee$$
$$(k \neq 0 \wedge k \in \omega \wedge esFuncion(F) \wedge y = F(k-1) \circ \mathcal{C}[\![If]\!]) \vee$$
$$(\neg(k = 0 \vee (k \neq 0 \wedge k \in \omega \wedge esFuncion(F))) \wedge y = F)$$

satisface que:

$$(\forall k, F| : (\exists! y| : \varphi(k, F, y))).$$

Si se define a $G(k, F)$ como el único y que satisface $\varphi(k, F, y)$, entonces por el teorema 1, se tiene que existe una formula ψ que satisface que:

1) $(\forall k| : (\exists! y| : \psi(k, y)))$, es decir ψ define una función F tal que $\psi(k, F(k))$
2) $(\forall k | k \in \omega| : F(k) = G(k, F \restriction_{k-1}))$

Como $G(k, F)$ en notación de llaves es la función a trozos

$$G(k, F) = \begin{cases} \mathcal{C}[\![Do_0]\!] & si \quad k = 0 \\ F(k-1) \circ \mathcal{C}[\![If]\!] & si \quad k \in \omega\ y \\ & \quad esFuncion(F) \\ F & sino \end{cases}$$

entonces la fórmula

$$(\forall k | k \in \omega| : F(k) = G(k, F \restriction_{k-1}))$$

es equivalente a la recurrencia que define a $\mathcal{C}[\![Do_k]\!]$ arriba y por lo tanto, $F(k)$ debe ser igual a $\mathcal{C}[\![Do_k]\!]$. Por esta razón, como se tiene que F es una función tal que $F(k)$ es el único valor en el que $\psi(k, F(k))$ es verdad, entonces cuando el predicado

$$\psi(k, R)$$

sea cierto, debe ocurrir que $R = \mathcal{C}[\![Do_k]\!]$.

Por otro lado en [3], se demostró que

$$(\exists k | k \in \omega \wedge k \geq 0 : \mathcal{C}[\![Do_k]\!](\{\vec{x}\}) \subseteq Rgo_{\overline{Y}})$$

es un predicado equivalente a $wp(do\ B_0 \rightarrow S_0\ od, Post)$. Sin embargo, usando el predicado ψ, la fórmula anterior dentro de ZFS es una abreviación de

$$(\exists k | k \in \omega \wedge k \geq 0 : (\forall R | \psi(k, R) : R(\{\vec{x}\}) \subseteq Rgo_{\overline{Y}}))$$

Como $\psi(k, R)$ esta definida como

$$(k \notin \omega \wedge R = \emptyset) \vee$$

$$(k \in \omega \wedge (\exists d, Do | App(d, Do) : k \in d \wedge Do(k) = R))$$

entonces la fórmula anterior se puede escribir equivalentemente como

$$(\exists k | k \in \omega \wedge k \geq 0 : (\exists d, Do | App(d, Do) : k \in d \wedge Do(k)(\{\vec{x}\}) \subseteq Rgo_{\overline{Y}}))$$

la cual está escrita en el lenguaje de primer orden de la teoría de conjuntos. ■

Como puede observarse la fórmula de la precondición más débil de la demostración del teorema anterior, se consigue de forma constructiva, ya que el predicado $\psi(k, R)$ se extrae del metateorema de recursión transfinita, y como se tiene la fórmula explícita para φ, se puede construir la fórmula de $\psi(k, R)$ y por ende la fórmula de la precondición más débil dentro del lenguaje de primer orden de la teoría de conjuntos.

Por lo dicho en el párrafo anterior, este último teorema muestra que el cálculo de $wp(Do, .)$ para cualquier instrucción Do es decidible dentro del lenguaje de la teoría de conjuntos y por ende, como las reglas de cálculo de wp (que se enunciaron en la introducción) para las instrucciones distintas de Do, son aplicables directamente sobre lenguajes de primer orden, entonces el cálculo de $wp(S, .)$ para cualquier instrucción S es decidible sobre ZFS.

De la demostración se observa que la aserción $wp(Do, Post)$ que se extrae del teorema 2 es muy complicada, incluso la verificación del valor de verdad de dicha aserción teniendo los valores de las variables libres puede ser no decidible. Esto es debido a que el problema de la parada se puede escribir equivalentemente, como el problema de verificar el valor de verdad de la precondición $wp(S, true)$ para valores iniciales de las variables y constantes del programa S, es decir, para no contradecir la indecidibilidad del problema de la parada, debe ocurrir que verificar el valor de verdad de la aserción $wp(S, true)$, no es decidible en general.

Un ejemplo de uso del teorema 2 para mostrar la complejidad que pueden tener las aserciones calculadas es el siguiente, en donde el invariante del ciclo es calculado con la fórmula de wp para Do del teorema 2:

$[Var\ N, i, s :\ Enteros;$
$\quad s, i := 0, 0;$
$\quad \{Inv : (\exists k | k \in \omega \wedge k \geq 0 : (\exists d, Do | App(d, Do) :$
$\qquad\qquad k \in d \wedge Do(k)(\{(N, i, s)\}) \subseteq Rgo_{\overline{Y}}))\}$
$\quad do\ i \neq N \rightarrow$
$\qquad s, i := s + i, i + 1$
$\quad od$
$\{Post :\ s = \sum_{j=0}^{N-1} j\}$
$]$

Como las únicas variables libres que tiene Inv son N, i y s y el ciclo esta precedido de una instrucción de asignación, podemos calcular la precondición más débil de todo el programa aplicando la regla de wp de la asignación obteniendo

$$(\exists k | k \in \omega \wedge k \geq 0 : (\exists d, Do | App(d, Do) : k \in d \wedge Do(k)(\{(N, i, s)\}) \subseteq Rgo_{\overline{Y}}))[s, i := 0, 0]$$
$$\equiv$$
$$(\exists k | k \in \omega \wedge k \geq 0 : (\exists d, Do | App(d, Do) : k \in d \wedge Do(k)(\{(N, 0, 0)\}) \subseteq Rgo_{\overline{Y}}))$$
$\equiv$<Definición de $Rgo_{\overline{Y}}$ >
$$(\exists k | k \in \omega \wedge k \geq 0 : (\exists d, Do | App(d, Do) : k \in d \wedge Do(k)(\{(N, 0, 0)\}) \subseteq \{\langle N, i, s\rangle \in \mathbb{Z}^3 | s = \sum_{j=0}^{N-1} j\})))$$
$\equiv$<Definición de App >
$$(\exists k | k \in \omega \wedge k \geq 0 : (\exists d, Do | esFuncion(Do) \wedge d = Dom(Do) \subseteq \omega \wedge (\forall m | m \in d : m - 1 \subseteq d) \wedge (\forall m | m \in d : \varphi(m, Do \restriction_{m-1}, Do(m))) : k \in d \wedge Do(k)(\{(N, 0, 0)\}) \subseteq \{\langle N, i, s\rangle \in \mathbb{Z}^3 | s = \sum_{j=0}^{N-1} j\})))$$
$\equiv$<Definición de φ >
$$(\exists k | k \in \omega \wedge k \geq 0 : (\exists d, Do | esFuncion(Do) \wedge d = Dom(Do) \subseteq \omega \wedge (\forall m | m \in d : m - 1 \subseteq d) \wedge (\forall m | m \in d : (m = 0 \wedge Do(m) = \mathcal{C}[\![Do_0]\!]) \vee (m > 0 \wedge Do(m) = Do \restriction_{m-1} (m - 1) \circ \mathcal{C}[\![If]\!]) : k \in d \wedge Do(k)(\{(N, 0, 0)\}) \subseteq \{\langle N, i, s\rangle \in \mathbb{Z}^3 | s = \sum_{j=0}^{N-1} j\})))$$

En donde $\mathcal{C}[\![Do_0]\!]$ y $\mathcal{C}[\![If]\!]$ se pueden calcular usando la definición de la semántica denotacional de la instrucción if de GCL definida en [3], con lo que se tiene que

$$\mathcal{C}[\![Do_0]\!] \equiv id_{\{\langle N,i,s\rangle \in \mathbb{Z}^3 | i = N\}} \cup (\{\langle N, i, s\rangle \in \mathbb{Z}^3 | i \neq N\} \times \{abort\}) \cup \{\langle abort, abort\rangle\}$$

y

$$\mathcal{C}[\![If]\!] \equiv \{\langle\langle N, i, s\rangle, \langle N', i', s'\rangle\rangle \in \mathbb{Z}^3 \times \mathbb{Z}^3 | \langle N', i', s'\rangle = \langle N, i + 1, s + i\rangle\} \circ id_{\{\langle N,i,s\rangle \in \mathbb{Z}^3 | i \neq N\}}) \cup id_{\{\langle N,i,s\rangle \in \mathbb{Z}^3 | i = N\}} \cup \{\langle abort, abort\rangle\}$$

El cálculo de esta aserción se hizo aplicando directamente las reglas de wp y las definiciones de la semántica denotacional de GCL, lo cual es automatizable, sin embargo la complejidad y tamaño de la aserción es considerable.

Nota. *Como una muestra de lo innecesariamente compleja que puede resultar una aserción calculada usando el teorema 2, observe que se puede demostrar por inducción sobre N, que esta última precondición más débil calculada, es equivalente a*

$0 \leq N$, que es un predicado mucho más simple. Esto muestra que es recomendable usar el teorema 2 sólo cuando no se puede calcular la precondición más débil del programa con otras técnicas.

V. Conclusiones

El teorema de decidibilidad del cálculo de wp provee una alternativa para calcular precondiciones más débiles de forma automática, sin embargo estas precondiciones, en muchos casos, no serían aserciones decidibles. Por esta razón el teorema sugiere una aplicación de software para el cálculo de wp que maneje aserciones no decidibles de forma simbólica.

La idea práctica es usar todas las técnicas conocidas para calcular precondiciones o invariantes, y en caso de que estas técnicas fallen en el cálculo de una aserción decidible, la aplicación arroja como última opción, la aserción resultante del teorema de decidibilidad aquí mostrado.

Una aplicación de este estilo trabajaría en muchas ocasiones con aserciones no decidibles, por lo que no tiene sentido manejar estas aserciones dentro del lenguaje de programación en cuestión, porque en estos casos las aserciones no serían programables. Por esta razón sugiero que una aplicación que haga uso del teorema de decibilidad aquí mostrado, maneje las aserciones como comentarios al código. Por ejemplo se pudiera desarrollar un IDE que inserte a modo de comentario, de forma automática la precondición más débil de cada instrucción. Un IDE de este tipo siempre puede computar una aserción entre todas las instrucciones del programa, aunque algunas de ellas sean no decidibles, pudiera ser provechoso, porque es posible que en el cálculo sucesivo de wp, las aserciones se simplifiquen y se obtenga, al final del proceso, una precondición más débil de todo el programa, que sea una aserción decidible.

Por otro lado desde el punto de vista teórico, los resultados aquí presentados muestran que la teoría de wp de Dijkstra es aplicable sobre la teoría de conjuntos, y por ende también el teorema de la invariancia y todas las reglas de Hoare derivadas de wp. De esta forma se pueden usar las técnicas de corrección formal sobre algoritmos con tipos de datos pertenencientes a la teoría de conjuntos. Por ejemplo, se puede usando wp o reglas de Hoare, corregir algoritmos donde los tipos de las variables son objetos como ordinales, cardinales, filtros, ultrafiltros, espacios topológicos, etc.

Referencias

[1] E. W. Dijkstra. *Guarded Commands, Nondeterminacy and Formal Derivation of Programs*. Communications of the ACM, vol. 18, no. 8, pp. 453-457, 1975.

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

[3] 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.

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

[5] F. Flaviani. *Calculation of Invariants Assertions*, en las memorias de la XLIII Conferencia Latinoamericana en Informática (CLEI 2017), Córdoba, Argentina, Septiembre 2017.

[6] G. Winskel. *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, 1993.

[7] J. Berdine, A. Chawdhary, B. Cook, D. Distefano, and P. OHearn. *Variance Analyses from Invariance Analyses*, in proceedings of the 34th Annual Symposium on Principles of Programming Languages, Nice, France, January 2007.

[8] E. Rodríguez Carbonnell and D. Kapur. *Program Verification using Automatic Generation of Invariants*, in proceedings of the First International Colloquium on Theoretical Aspects of Computing, Guiyang, China, September 2004.

[9] J. Carette and R. Janicki. *Computing Properties of Numeric Iterative Programs by Symbolic Computation*. Fundamentae Informatica, vol. 80, no. 1, pp. 125-146, March 2007.

[10] M. A. Colon, S. Sankaranarayana, and H. B. Sipma. *Linear Invariant Generation using non Linear Constraint Solving*. Computer Aided Verification, vol. 2725, pp. 420-432, 2003.

[11] M. D. Ernst, J. H Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao. *The Daikon System for Dynamic Detection of Likely Invariants*. Science of Computer Programming, 2006.

[12] J.C. Fu, F. B. Bastani, and I-L. Yen. *Automated Discovery of Loop Invariants for High Assurance Programs Synthesized using AI Planning Techniques*, in proceedings of the 11th High Assurance Systems Engineering Symposium (HASE 2008), pp. 333-342, Nanjing, China, December 2008.

[13] L. Kovacs and T. Jebelean. *Automated Generation of Loop Invariants by Recurrence Solving in Theorema*, in proceedings of proceedings of the 6th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2004), pp. 451-464, Timisoara, Romania, September 2004.

[14] L. Kovacs and T. Jebelean. *An Algorithm for Automated Generation of Invariants for Loops with Conditionals*, in proceedings of the Computer-Aided Verification on Information Systems Workshop (CAVIS 2005), 7th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005), pp. 16-19, Timisoara, Romania, September 2005.

[15] S. Sankaranarayana, H. B. Sipma, and Z. Manna. *Non Linear Loop Invariant Generation Using Groebner Bases*, in proceedings of the ACM SIGPLAN Symposium on Principles of Programming Languages (POLP 2004), pp. 381-329, Venice, Italy, January 2004.

[16] A. Gupta and A. Rybalchenko. *InvGen: An Efficient Invariant Generator*, in proceedings of the International Conference on Computer Aided Verification (CAV 2009), Grenoble, France, June 2009.

[17] Stanford Invariant Generator, 2006, http://theory.stanford.edu/~srirams/Software/sting.html

[18] E. Rodríguez-Carbonell and D. Kapur. *Generating all Polynomial Invariants in Simple Loops*. Journal Symbolic Computation, vol. 42, no. 4, pp. 443-476, April 2007.

[19] S. Magill, A. Nanevski, E. Clarke, and P. Lee. *Inferring Invariants in Separation Logic for Imperative List-processing Programs*, in proceedings of the Third Workshop on Semantics, Program Analysis, and Computing Environments for Memory Management (SPACE 2006), Charleston, SC, USA, January 2006.

[20] J. Berdine, B. Cook, and S. Ishtiaq, *SLAyer: Memory Safety for Systems-Level Code*, in proceedings of the 23rd International Conference on Computer Aided Verification (CAV 2011), Snowbird, UT, USA, July 2011.

[21] C. Varming and L. Birkedal. *Higher-order Separation Logic in Isabelle/HOLCF*. Electronic Notes in Theoretical Computer Science, vol. 218, pp. 371-389, October 2008.

[22] M. Barnett, K. Rustan, and M. Leino, Microsoft Research. *Weakest-Precondition of Unstructured Programs*, in proceedings of the 6th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, (PASTE 2005), pp. 82-87, Lisbon, Portugal, September 2005.

[23] K. Kunen. *Set Theory*. College Publications, London, UK, 2013.

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