MANTENIMIENTO DEL SOFTWARE EVOLUCIÓN 4.0

ALEXANDER ELIAS HERNANDEZ CUADRADO
ADRIANA JUDITH MONSALVE QUINTERO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

MANTENIMIENTO DEL SOFTWARE EVOLUCIÓN 4.0

ALEXANDER ELIAS HERNANDEZ CUADRADO

ADRIANA JUDITH MONSALVE QUINTERO

Trabajo de Grado para optar al título de
Ingenieros de Sistemas

Director
HUGO HERNANDO ANDRADE SOSA
Mag. Ingeniería de Sistemas

Codirector
EMILIANO DE JESUS LINCE MERCADO
Mag. Ingeniería de Sistemas

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2014

DEDICATORIA

Dedico este trabajo de grado a Dios, a mis padres, a mis hermanos y a mi novio. A Dios porque ha estado conmigo a cada paso que doy, cuidándome y dándome fortaleza para continuar, a mis padres, quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento. A mis hermanos por acompañarme en este proceso. A mi novio por ser mi gran apoyo y compañero de lucha, porque con su amor, confianza y paciencia me impulsó a seguir adelante, creyó en mí y es el motor de mi vida para luchar cada día por nuestros sueños. Es por ellos que soy lo que soy ahora. Los amo de todo corazón.

Adriana Judith Monsalve Quintero

DEDICATORIA

Dedico este trabajo de grado a Dios, a mis padres, a mis hermanos, a mi prima Yuranis que está en el cielo a la selección de futbol sala. A Dios porque ha estado conmigo a cada paso que doy, cuidándome y dándome fortaleza para continuar, a mis padres, quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento. A mis hermanos por acompañarme en este proceso. A mi prima por ser una luchadora de la vida y cumplidora de sueños, a mi selección de futbol sala de la UIS en cabeza de mi profe Rene y mis amigos Carlitos, Jhon, Pacho, Rolo, Tavo, Pita, Pollo, Isidro, Casillas, Caucho y otros que en el momento se me escapan por ser una gran experiencia de vida y brindar gratos momentos a mi vida.

Gracias a todos por apoyarme en esta meta. Los amo de todo corazón.

Alexander Elías Hernández Cuadrado

AGRADECIMIENTOS

Este Trabajo de Grado es un esfuerzo en el cual, directa o indirectamente, participaron varias personas leyendo, opinando, corrigiendo, teniéndonos paciencia, dándonos ánimo, acompañándonos en los momentos de crisis y en los momentos de felicidad.

Es por ello que damos las gracias primero que todo a Dios, ser supremo que nos bendice día a día y por quien estamos en este mundo, a la Universidad Industrial de Santander, porque a lo largo de estos años nos formó como personas de alta calidad ética, política y profesional, además nos permitió ampliar nuestros conocimientos y conocer personas que han dejado huella en nuestras vidas.

Queremos dar las gracias especialmente al Grupo SIMON de Investigación y a todos sus integrantes, en cabeza del profesor Hugo Hernando Andrade Sosa, quien con sus conocimientos y orientaciones nos guio a lo largo de este proyecto, que tuvo sus altibajos los cuales pudimos superar, gracias profesor por la confianza depositada, por la paciencia y por el apoyo que nos ha brindado en todo momento; de igual forma extendemos nuestros agradecimientos al desarrollador de Evolución 3.5 Mag. Emiliano de Jesús Lince Mercado, pilar fundamental de este trabajo quien con su amplio conocimiento y visión nos encaminó en la dirección correcta del proyecto.

Especial reconocimiento al Mag. Jorge Jair Moreno Chaustre por permitirnos conocer su trabajo y servir de guía para el nuestro; a los estudiantes de la Universidad Industrial de Santander y Universidad Cooperativa de Colombia quienes aportaron en la realización de pruebas.

Gracias a nuestras familias, por su apoyo incondicional, acompañamiento y espera a lo largo de este tiempo.

Queremos hacer extensiva nuestra gratitud al Departamento de Deportes de la Universidad Industrial de Santander, a sus profesores, a nuestro entrenador, a los compañeros de la selección masculina y femenina de Fútbol Sala de la UIS y amigos, que fueron testigos de este esfuerzo, que con sus palabras de ánimo nos impulsaron a la consecución de este objetivo.

Gracias a todos ellos por el conocimiento adquirido, las experiencias vividas, los lazos de amistad creados y los momentos compartidos.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

TITULO: MANTENIMIENTO DEL SOFTWARE EVOLUCION 4.0*

AUTORES: ALEXANDER ELIAS HERNANDEZ CUADRADO
ADRIANA JUDITH MONSALVE QUINTERO**

PALABRAS CLAVES: MANTENIMIENTO, SOFTWARE, DINÁMICA DE SISTEMAS, MODELADO, SIMULACIÓN, EVALUACION, PRUEBAS, EVOLUCIÓN.

DESCRIPCIÓN:

Este documento presenta la realización por etapas del Mantenimiento de la herramienta para el modelado y simulación con Dinámica de Sistemas llamada EVOLUCIÓN 4.0, perteneciente al grupo SIMON de la Universidad Industrial de Santander. Se comienza con la evaluación arquitectónica del software que toma como base la Tesis de Maestría "DISEÑO DE UNA ARQUITECTURA PARA UN ENTORNO DE MODELAMIENTO - SIMULACIÓN Y CREACION DE UN PROCESO PARA SU DESARROLLO POR UNA COMUNIDAD (I+D) desarrollada por el Mag. Jorge Jair Moreno Chaustre; evaluación que mediante la aplicación de un modelo de métricas, a través de herramientas computacionales, permitió conocer su estructura interna, el estado de sus atributos de calidad, los posibles problemas de diseño subyacentes y finalmente las directrices para su eventual mejora. Enseguida se aplica el estándar 1219 de la IEEE para Mantenimiento de Software, iniciando con la identificación de los errores presentes en la herramienta, los cuales fueron reportados por los usuarios, para su corrección. Se realizan diferentes tipos de pruebas como son de unidad, de integración, de sistema y de aceptación con usuarios, a fin de verificar el correcto funcionamiento del software y se complementa la documentación del mismo. Al finalizar se obtienen productos como un artículo que detalla las características del software, el cual fue publicado en la revista Latinoamericana de Dinámica de Sistemas, un videotutorial que enseña paso a paso cómo elaborar modelos y una página web a cerca de la herramienta. Los cambios hechos y la corrección de los errores se recopilan en una nueva versión del software EVOLUCIÓN 4.5.

* Trabajo de Grado

** Facultad de Ingenierías Fisico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Hugo Hernando Andrade Sosa, Mag. Ingeniería de Sistemas. Codirector: Mag. Emiliano de Jesús Lince Mercado, Mag. Ingeniería de Sistemas.

ABSTRACT

TITLE: SOFTWARE MAINTENANCE EVOLUTION 4.0*

AUTHORS: ALEXANDER ELIAS HERNANDEZ CUADRADO
ADRIANA JUDITH MONSALVE QUINTERO**

KEYWORDS: MAINTENANCE, SOFTWARE, SYSTEM DYNAMICS, MODELING, SIMULATION, EVALUATION, TESTING, EVOLUTION.

DESCRIPTION:

This paper presents the realization by stages of maintenance for the tool for modeling and simulation with System Dynamics called EVOLUTION 4.0, SIMON group belonging to the Industrial University of Santander.It begins with the architectural evaluation of software, which is based on the Master Thesis "DESIGNING AN ARCHITECTURE FOR MODELING ENVIRONMENT - SIMULATION AND CREATION OF A PROCESS FOR BY A COMMUNITY DEVELOPMENT (I & D) developed by Jair Moreno Jorge Mag Chaustre; evaluation by implementing a model metric, through computational tools, which allowed to know its internal structure, the status of their quality attributes, the possible underlying problems and ultimately design guidelines for possible improvement. Immediately applies the IEEE 1219 Standard for Software Maintenance, starting with the identification of errors in the tool, which were reported by users, for your correction. Performed different types of tests as are unit, integration, system and user acceptance, to verify the correct operation of the software and its documentation is complemented. The end products such as an article detailing the features of the software, which was published in the American Journal of Dynamic Systems, a video tutorial that teaches step by step how to develop models and a website about the tool. Changes made and correcting errors are collected in a new version of software EVOLUTION 4.5.

* Bachelor Thesis

** Facultad de Ingenierías Fisico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Hugo Hernando Andrade Sosa, Mag. Ingeniería de Sistemas. Codirector: Mag. Emiliano de Jesús Lince Mercado, Mag. Ingeniería de Sistemas.

INTRODUCCIÓN

El término mantener proviene del latín *manu tenere*, tener en la mano, en definitiva, controlar. Sin embargo, parece que no hay nada más difícil de gestionar que la evolución del software y las aplicaciones informáticas que se encuentran actualmente en las empresas [12][1].

En ocasiones se dice que sólo los productos software deficientes requieren mantenimiento. De hecho, la verdad es lo contrario: los productos deficientes se desechan, mientras que los buenos se reparan y mejoran, durante 10, 15 o incluso 20 años. Más aún, un producto software es un modelo del mundo real que está cambiando con frecuencia. Como consecuencia, el software tiene que recibir mantenimiento continuo para que siga mostrando con exactitud el mundo real [15][2].

Es por ello, que existe el interés por parte del Grupo SIMON de Investigación en desarrollar este proyecto, Mantenimiento del Software Evolución 4.0, debido a que es una herramienta para el modelamiento y simulación de fenómenos complejos con Dinámica de Sistemas, que es usada por estudiantes y profesores a nivel nacional e internacional, en labores académicas e investigativas como un instrumento para la realización de modelos de simulación, enfocados a la creación y recreación del conocimiento.

Por lo tanto, se requiere de una herramienta fiable[3], robusta y con documentación actualizada para los usuarios finales y los futuros desarrolladores. Esto se logra haciendo una evaluación del software, para

[1] [12] PIATTINI, Mario, VILLALBA, José, RUÍZ, Francisco, POLO, Macario y otros. "Mantenimiento del Software, Modelos, Técnicas y Métodos para la Gestión del Cambio". Alfaomega. 2001. 336 p.

[2] [15] SCHACH, Stephen. Ingeniería de Software Clásica y Orientada a Objetos. México: McGraw-Hill, 2006. p.

[3] Definición según IEEE Std 610.12-1990. Standard Glossary of Software Engineering Terminology.

- *Fiabilidad:* La capacidad de un sistema o componente para realizar sus funciones requeridas en condiciones indicadas durante un período especificado de tiempo.
- *Robustez:* El grado al cual un sistema o componente puede funcionar correctamente en la presencia de entradas inválidas o condiciones estresantes ambientales.

encontrar las falencias a nivel de software y documentación, y corregirlas mediante el mantenimiento del mismo.

1 ENTIDADES INTERESADAS

La elaboración de este proyecto es de interés para instituciones tales como:

1.1 UNIVERSIDAD INDUSTRIAL DE SANTANDER

Es de interés para la Universidad Industrial de Santander porque permite motivar a la comunidad académica en la inclusión de los estudiantes de pregrado a los diferentes procesos de investigación, contribuyendo a mejorar la calidad de la misma dentro y fuera de la institución, fomentando el desarrollo de la ciencia y la formación profesional de la juventud colombiana.

1.2 GRUPO SIMON DE INVESTIGACIÓN

Para el grupo SIMON de Investigación es de interés continuar mejorando una herramienta útil a los procesos de formación e investigación en modelamiento y simulación con Dinámica de Sistemas (DS) como Evolución, la cual tuvo su origen en el año de 1994 con la aparición de la versión 1.0 y se ha venido desarrollando a través de diferentes proyectos de grado (actualmente se encuentra disponible la versión 4.0), trabajos que se han realizados en medio de un proceso de análisis, diseño, desarrollo, evaluación y mejoramiento, conducente a obtener productos software con alto nivel de acabado y coherentes con la calidad de los mismos.

1.3 COMUNIDAD USUARIA DE LA HERRAMIENTA SOFTWARE EVOLUCIÓN

Es de vital importancia para la comunidad usuaria de Evolución tanto a nivel nacional e internacional, en los campos de docencia e investigación, disponer de una herramienta software para el modelamiento y simulación con Dinámica de Sistemas más robusta, con una documentación adecuada y que permita al usuario emplearla en labores académicas e investigativas, en la resolución de problemas, generación de conocimiento y toma de decisiones entre otros.

2 PLANTEAMIENTO DEL PROBLEMA

El mantenimiento de software, entendido como la adaptación de los sistemas a los nuevos requerimientos del negocio y su correcto funcionamiento, viene adquiriendo una importancia crítica, por lo tanto es necesario establecer una sistemática para su realización. Sistemática basada en metodologías, modelos de gestión, métricas, que tiene como objetivo mejorar la calidad del proceso y como consecuencia reducir costes.

El mantenimiento es una etapa de vital importancia en el ciclo de vida del software. En la actualidad muchos proyectos de pregrado realizados en la Universidad Industrial de Santander, incluyen la elaboración de herramientas software para diversas áreas y con muchas aplicaciones, las cuales tienen como prioridad el desarrollo y puesta en marcha de la mismas, finalizando con la etapa de pruebas y despliegue, dejando a un lado la etapa del mantenimiento, sin tener en cuenta en términos de esfuerzo y costes necesarios, las actividades que hay que realizar durante esta fase.

El grupo SIMON de Investigación perteneciente a la Escuela de Ingeniería de Sistemas de la Universidad Industrial de Santander, creó importantes aplicaciones orientadas al aprendizaje y a la simulación de fenómenos complejos en diferentes áreas, y mantiene como proyecto bandera el desarrollo de Evolución, herramienta software para el modelamiento y simulación con Dinámica de Sistemas (DS). El grupo SIMON y la herramienta Evolución han obtenido el reconocimiento en este campo en el ámbito nacional e internacional.

Evolución se ha convertido en un software de gran envergadura y complejidad; debido a esto, se ha abordado a través de diferentes tesis de grado, algunas de estas cubren el análisis y diseño de la herramienta, otras el núcleo y los últimos trabajos se centran en generar nuevos módulos. Con estos proyectos se

cumplió con las etapas del desarrollo de software, desde el análisis hasta la evaluación del software. Por lo tanto, es necesario dar inicio a la etapa de mantenimiento y así contemplar todas las fases del ciclo de vida del software, incluyendo la corrección de defectos y el mejoramiento del mismo.

Debido al uso continuo de la herramienta Evolución y a la realización de continuas modificaciones tras el desarrollo de nuevas versiones, se han encontrado falencias en el software. La realización de la tesis de maestría de J. Moreno [19][4] evidenció los aspectos en que se estaba fallando con Evolución.

En esta tesis de maestría se evaluó el código fuente, los requisitos y los diagramas UML de Evolución 3.5, a fin de conocer su estructura interna, el estado de sus atributos de calidad, los posibles problemas de diseño subyacentes y las directrices para su eventual mejora, con el propósito de verificar la calidad del software. En la evaluación de las métricas del modelo de atributos externos propuesto para Evolución, se encontraron fallos relacionados con violaciones a las reglas de diseño de los atributos de calidad. El autor de la tesis presenta varias sugerencias para el mejoramiento de Evolución, en aras de alcanzar una arquitectura más organizada, fácil de entender y mantener, y así mejorar el estado de sus atributos de calidad.

La documentación existente sobre Evolución tales como ayuda, manual del usuario y manual del programador, pertenecen a la versión 3.5. Esta documentación requiere ser actualizada, para contemplar los cambios realizados en la versión 4.0 y en la versión que se generará al finalizar este proyecto.

[4] [19] MORENO CHAUSTRE, Jorge Jair. Diseño de una arquitectura para un entorno de modelamiento – simulación y creación de un proceso para su desarrollo por una comunidad (I+D). Tesis de Maestría. Maestría en Informática. Bucaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas. 2006. 246 p.

Debido a todo lo anterior, surge la necesidad de un software fiable, más robusto y con una documentación actualizada, capaz de realizar sus funciones de forma correcta y por ello la importancia de este proyecto, Mantenimiento del Software Evolución 4.0.

3 DESCRIPCIÓN DE OBJETIVOS

3.1 OBJETIVO GENERAL

Realizar el mantenimiento de Evolución 4.0 a partir de la ejecución de pruebas al sistema, errores reportados por los usuarios y la evaluación hecha al software en la tesis de maestría "DISEÑO DE UNA ARQUITECTURA PARA UN ENTORNO DE MODELAMIENTO - SIMULACIÓN Y CREACION DE UN PROCESO PARA SU DESARROLLO CON UNA COMUNIDAD (I+D)".

3.2 OBJETIVOS ESPECÍFICOS

1. Desarrollar una nueva versión de Evolución que contemple corrección de errores detectados a través de pruebas, reportados por los usuarios, la evaluación hecha al software Evolución y los resultados obtenidos en la tesis de maestría de J. Moreno[5].
    a. Evaluar el software Evolución 4.0, a partir de la información disponible (tesis de maestría de J. Moreno, requisitos, diagramas UML, código fuente, manuales) utilizando herramientas computacionales para valorar los atributos de calidad externos e internos definidos para Evolución[6], a fin de encontrar fallos y proponer mejoras a la herramienta, consolidando los resultados en una ficha técnica.
    b. Corregir errores de sintaxis, de ejecución y de lógica presentes en el software, utilizando el lenguaje de programación Delphi, con el propósito de aportar al cumplimiento de los requisitos funcionales de Evolución.
    c. Realizar pruebas del sistema, de unidad, de integración, y de validación al software Evolución 4.0, utilizando herramientas de

[5] [19] MORENO CHAUSTRE, Jorge Jair. Diseño de una arquitectura para un entorno de modelamiento – simulación y creación de un proceso para su desarrollo por una comunidad (I+D). Tesis de Maestría. Maestría en Informática. Bucaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas. 2006. 246 p.

[6] Apartado 6.3.4 del Marco Teórico "Atributos de calidad utilizados para evaluar a Evolución 3.5 y 4.0".

pruebas[7] que faciliten esta labor, a fin de detectar defectos en el software y verificar el funcionamiento de Evolución.

   d. Completar la documentación de Evolución tales como ayuda, manual del usuario, manual del programador, elaborando el contenido faltante y reestructurando el que esté insuficiente, a fin de que la documentación quede actualizada, corregida y publicada en el sitio web del grupo SIMON de Investigación.

2. Brindar soporte a los usuarios de Evolución, a través de la creación de una página web en el servidor del grupo SIMON[8], con el contenido de la herramienta y un foro, para mantener un contacto frecuente y conocer las necesidades del usuario.
3. Elaborar y postular un artículo acerca de Evolución a una revista o como ponencia en un evento, para dar a conocer a toda la comunidad dinámico sistémica la herramienta como un software de modelado y simulación con Dinámica de Sistemas.

[7] Herramientas CAST (Computer Aided Software Testing).

[8] http://simon.uis.edu.co/joomla/home/index.php?option=com_content&view=article&id=215&Itemid=92

4 JUSTIFICACIÓN Y VIABILIDAD DEL PROYECTO

4.1 JUSTIFICACIÓN

Evolución es una herramienta que es usada por la comunidad de Dinámica de Sistemas a nivel nacional e internacional, en labores académicas e investigativas relacionadas con el modelado y la simulación. En particular, alrededor de 700 instituciones educativas de básica y media en Colombia, en las cuales la Universidad Industrial de Santander en convenio con Computadores Para Educar (CPE)[9] implementa una propuesta para llevar el modelado y la simulación a la educación. Debido a que se han detectando algunos errores de funcionamiento y limitaciones en la documentación de la herramienta, se hace necesario realizar actividades para el mantenimiento del software y así ofrecer a la comunidad usuaria un mejor producto.

La evaluación y mantenimiento del software son fases que por sus características en los trabajos de pregrado no se profundiza, se espera que este proyecto aporte a la apropiación de los métodos y herramientas de ésta etapa, para la realización de futuros proyectos en la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander, relacionados con el mantenimiento de software.

La realización del proyecto requiere de tiempo y dedicación por parte de los dos investigadores y directores del mismo, con el propósito de realizar la investigación exhaustiva de toda la documentación necesaria para la evaluación y mantenimiento del software, las herramientas CASE[10] y de programación apropiadas para la corrección de errores, la realización de pruebas y la complementación de la documentación de Evolución, a fin de

[9] CPE: Es el Programa Multi-Impacto del Gobierno Nacional, que viene impulsando, desde el año 2000, el desarrollo de las comunidades colombianas, reduciendo la brecha digital y de conocimiento a través del acceso, uso y aprovechamiento de las Tecnologías de la Información y las Comunicaciones en las comunidades educativas.

[10] *CASE:* Computer Aided Software Engineering (Ingeniería de Software Asistida por Ordenador). Aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

obtener una herramienta con mejores características, más robusta, fiable y mejor documentada, tanto como para el usuario final como para los futuros desarrolladores.

4.2 VIABILIDAD

La disponibilidad de recursos humanos, tecnológicos e informáticos tales como código fuente, documentación (manual del usuario, manual del programador, tesis de grado) y diagramas UML del software Evolución, además de herramientas[11] para realizar mantenimiento software, permiten que esta tesis de grado sea viable.

El grupo SIMON posee toda la información generada a partir del desarrollo de Evolución 3.5, también cuenta con el apoyo de sus desarrolladores, que pueden enriquecer el proceso con su experiencia y con la licencia de la herramienta Delphi 7 como lenguaje de programación; además dispone de material bibliográfico útil para el desarrollo de la tesis. La biblioteca central de la UIS tiene libros, revistas, bases de datos y tesis con temas como desarrollo, ingeniería y mantenimiento de software. Por esto se dice que la información necesaria es de fácil acceso.

Dado que el grupo SIMON de Investigación es rico en personal humano, sus instalaciones físicas facilitan la labor investigativa y la dinámica de trabajo hace que exista apoyo constante entre sus integrantes, se puede decir que se cuenta con recursos humanos y tecnológicos que facilitan el proceso.

[11] Herramientas CASE, de automatización del mantenimiento, de perfeccionamiento del código, de ingeniería inversa, de gestión de la configuración, de prueba.

5 MARCO TEÓRICO

5.1 SOFTWARE DE MODELADO Y SIMULACIÓN CON DINÁMICA DE SISTEMAS

El desarrollo de modelos de simulación es una aproximación ampliamente utilizada con el fin de analizar, comprender y predecir el funcionamiento de cualquier sistema en función de las variables que rigen su dinámica. “La posibilidad de realizar prácticas usando el ordenador y un software de construcción y simulación de modelos permite al usuario adquirir un conocimiento procedural de los conceptos y la actividad de modelado, al darle la posibilidad de realizar actividades como el diseño de modelos conceptuales y numéricos, simulación, análisis de sensibilidad, validación, etc. [7]”.

La Dinámica de Sistemas usa conceptos del campo del control realimentado para organizar información en un modelo de simulación por ordenador. Un ordenador ejecuta los papeles de los individuos en el mundo real. La simulación resultante revela implicaciones del comportamiento del sistema representado por el modelo [21].

Existen diversas herramientas de modelado y simulación tales como PowerSim, Ithink, Stella, Vensim, Evolución, entre otras, las cuales permiten conceptualizar, documentar, simular, analizar y optimizar modelos de Dinámica de Sistemas. Estas herramientas proveen una forma simple y flexible de construir modelos de simulación mediante diagramas causales y diagramas de Forrester[12].

Es importante realizar una comparación entre Evolución y otras herramientas software de modelado y simulación, a fin de distinguir sus semejanzas y diferencias, y así mejorar los aspectos en los que esté fallando el software.

[12] Los Diagrama de Forrester proporcionan una representación gráfica de los sistemas dinámicos modelando cualitativamente las relaciones entre las partes mediante símbolos que corresponden a una interpretación hidrodinámica del sistema.

5.2 MANTENIMIENTO DE SOFTWARE SEGÚN ESTÁNDAR IEEE 1219 - 1998

Para el Mantenimiento del Software existen los estándares: IEEE 1219 e ISO/IEC 14764.

ISO/IEC 14764: Éste estándar internacional describe el esqueleto del Proceso de Mantenimiento Software pero no específica los detalles de cómo implementar o ejecutar las actividades y tareas incluidas en el proceso.

IEEE 1219: Estándar que íntegramente se ocupa del proceso de Mantenimiento del Software. Esta norma describe un proceso iterativo para la gestión y ejecución de actividades de mantenimiento de software. Los criterios establecidos se aplican tanto a la planificación del Mantenimiento del Software mientras este está en desarrollo, como a la planificación y ejecución de las actividades de Mantenimiento para productos software existentes.

El estándar IEEE 1219 además establece los requisitos para el proceso, el control y la gestión de la planificación, ejecución y documentación de las actividades de mantenimiento de software. Este proceso de mantenimiento está dividido en fases. Dentro de cada una de estas fases, el estándar define una serie de procedimientos que se han de llevar a cabo y con los que se identifican la documentación, personas y productos software que intervienen.

Por todas estas razones se elige este estándar como guía para el mantenimiento del software Evolución.

Fases del Desarrollo en el Mantenimiento del Software.

- Identificación y Clasificación del Problema o de la Modificación.
- Análisis.
- Diseño.

- Implementación.
- Pruebas del Sistema.
- Pruebas de Aceptación.
- Liberación del Producto.

5.2.1 Identificación, clasificación y priorización del problema. En esta fase se identifican, clasifican y asignan una prioridad inicial a las modificaciones del software.

5.2.2 Análisis. En esta fase se estudia la viabilidad y el alcance de las modificaciones, que ya tenemos clasificadas y priorizadas, así como la generación de un plan preliminar de diseño, implementación, pruebas y liberación del software.

5.2.3 Diseño. A partir de toda la documentación existente del proyecto y del sistema que esté en producción, así como todo el código fuente y las bases de datos de la última versión, se va a realizar el diseño de las modificaciones del sistema en base a la documentación generada en la fase de análisis (análisis detallado, informe de requisitos, identificación de los elementos afectados, estrategia de pruebas y plan de implementación).

El primer paso en la fase de diseño será identificar los módulos software que van a ser objeto de modificación, con el fin de hacer constar la planificación de tareas y ver la previsión de las mejoras a introducir. Según se avanza en los módulos se realizarán las modificaciones oportunas a la documentación de los mismos. Esta documentación consiste de diagramas de flujo y control, esquemas, etc.

Para las modificaciones a realizar, se generará unos casos de pruebas que incluyan los elementos de seguridad y robustez. Además hay que identificar e incluir las pruebas de regresión necesarias.

En la documentación que se va generando en la fase de diseño, se tendrán que identificar y documentar los cambios que se realicen sobre los requisitos, y mantener al día una lista con las modificaciones que se van a llevar a cabo.

Mientras se realizan las modificaciones del software, se pueden correr una serie de riesgos, que se deben tener en cuenta para evitar complicaciones durante el mantenimiento y mantener la calidad del software, estos son:

- Uno de los riesgos que se corre a la hora de modificar el código fuente es la de generar nuevos errores, debido a cambios indebidos que se hagan sobre el mismo.
- Si no se lleva un control de versiones, es probable que no se sepa cuál es la última versión del software actualizada.

5.2.4 Implementación. En esta fase, se van a seguir unos determinados procesos que serán iterativos, y gradualmente incrementales, es decir, se irán repitiendo y desarrollando en mayor detalle, hasta obtener el resultado previsto por la organización. Estos procesos son:

- Codificación y pruebas de unidad.
- Integración.

5.2.4.1 Pruebas de Unidad. La prueba de unidad se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño del software: el componente o módulo de software. Durante la prueba de unidad, una tarea esencial es la prueba selectiva de las rutas de ejecución. Se deben diseñar casos de prueba para descubrir errores debidos a cálculos incorrectos, comparaciones erróneas o flujos de control inapropiados. Las pruebas de unidad se concentran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente[13].

[13] Ingeniería del Software. Un enfoque práctico. Roger S. Pressman.

5.2.4.2 Pruebas de Integración. La prueba de integración es una técnica sistemática para construir la arquitectura del software mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar componentes a los que se aplicó una prueba de unidad y construir una estructura de programa que determine el diseño[14].

5.2.5 Pruebas del Sistema. Son usualmente conducidas para asegurar que todos los módulos trabajan juntos sin error. Es similar a la prueba de integración pero con un alcance mucho más amplio.

Las pruebas del sistema examinan que tan bien el sistema cumple con los requerimientos de la organización y su utilidad, seguridad y desempeño. También prueba la documentación del sistema. Dentro de ellas están las Pruebas de Rendimiento, de Seguridad y de Recuperación.

5.2.5.1 Pruebas de Rendimiento. Prueban el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

- Requiere de instrumentación tanto software como hardware para los procesos de monitorización y medición.
- Se lleva a cabo durante todos los pasos de prueba.

5.2.5.2 Pruebas de seguridad. Verifica que los mecanismos de protección incorporados protegerán al sistema.

- Intenta conseguir las claves de acceso de cualquier forma.
- Ataca con software a medida.
- Bloquea el sistema.
- Provoca errores del sistema entrando durante su recuperación.

[14] Ingeniería del Software. Un enfoque práctico. Roger S. Pressman.

5.2.5.3 Pruebas de recuperación. Fuerza el fallo del software de varias formas y verifica que la recuperación se produce adecuadamente.

- Si la recuperación es automática hay que evaluar la corrección de la inicialización, de los mecanismos de recuperación del estado del sistema, de la recuperación de los datos y del proceso de rearranque.
- Si la recuperación no es automática, hay que evaluar los tiempos medios de reparación para determinar si están dentro de unos límites aceptables.

5.2.6 Pruebas de Aceptación. Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios.

5.2.7 Liberación del producto. Una vez probado completamente el sistema, estamos a punto de pasar a la fase de liberación de la versión modificada. Los pasos a seguir para instalar la nueva versión serán los siguientes:

- Notificar a la comunidad de usuarios.
- Desarrollar una versión de archivo del sistema para salvaguarda del mismo.
- Realizar la instalación y formación para el cliente. Se ha de proveer del material de sistema necesario para facilitar la utilización a los usuarios.
- Se ha de completar la Documentación de Descripción de la Versión

5.3 EVALUACIÓN ARQUITECTÓNICA DEL SOFTWARE EVOLUCIÓN 4.0

Este capítulo se concentra en la satisfacción del objetivo específico 1.1 de este trabajo, el cuál propone:

- *"Evaluar el software Evolución 4.0, a partir de la información disponible (tesis de maestría de J. Moreno, requisitos, diagramas UML, código fuente, manuales) utilizando herramientas computacionales para valorar los atributos de calidad externos e internos definidos para Evolución[15], a fin de encontrar fallos y proponer mejoras a la herramienta, consolidando los resultados en una ficha técnica."*

Para llevar a cabo este objetivo, se realizó una evaluación arquitectónica que mediante la aplicación de un modelo de métricas sobre Evolución 4.0, permitió conocer su estructura interna, el estado de sus atributos de calidad, los posibles problemas de diseño subyacentes y finalmente las directrices para su eventual mejora, temas que se abordan a continuación.

Cabe aclarar que la versión 4.0 de Evolución, es la adaptación de la versión 3.5 de la herramienta con el componente adicional FIS (Sistema de Inferencia Borrosa), por lo tanto esta evaluación fue realizada en dos partes:

- Primero se realizó la evaluación arquitectónica de Evolución 3.5 por parte del Ing. Jorge Jair Moreno Chaustre[16] como parte fundamental e indispensable en la elaboración de una arquitectura software para un ESMS - MI[17], que tenía como punto de partida la herramienta software Evolución 3.5, debido a la concepción arquitectónica madura para realizar un tratamiento sobre la escalabilidad hacia un entorno más sofisticado para el modelado y la simulación, como uno de los objetivos planteados en su

[15] Apartado 6.3.4 del presente capítulo "Atributos de Calidad utilizados para evaluar a Evolución 4.0".
[16] Magister en Informática UIS.
[17] ESMS - MI: Entorno Software de Modelado y Simulación de Modelos Integrados.

tesis de maestría[18]. Es por esta razón que se hace referencia a dicho trabajo.

- Seguidamente se efectuó la evaluación de Evolución 4.0, por lo tanto sólo se tuvo en cuenta la documentación (diagramas UML, código fuente, manuales) del componente FIS (Sistema de Inferencia Borrosa), versión que fue realizada mediante tesis de grado[19] desarrollada por integrantes del grupo Simon.

Es de aclarar que la metodología, objetivos, información a recopilar, herramientas utilizadas y demás apartados de la Evaluación Arquitectónica del Software, fueron realizados de la misma manera para la versión 3.5 y 4.0 de Evolución, pero aquí solo mostraremos los resultados obtenidos para Evolución 4.0, los resultados de la versión anterior se encuentran recopilados en el Anexo_1_Tesis_de_Maestria.

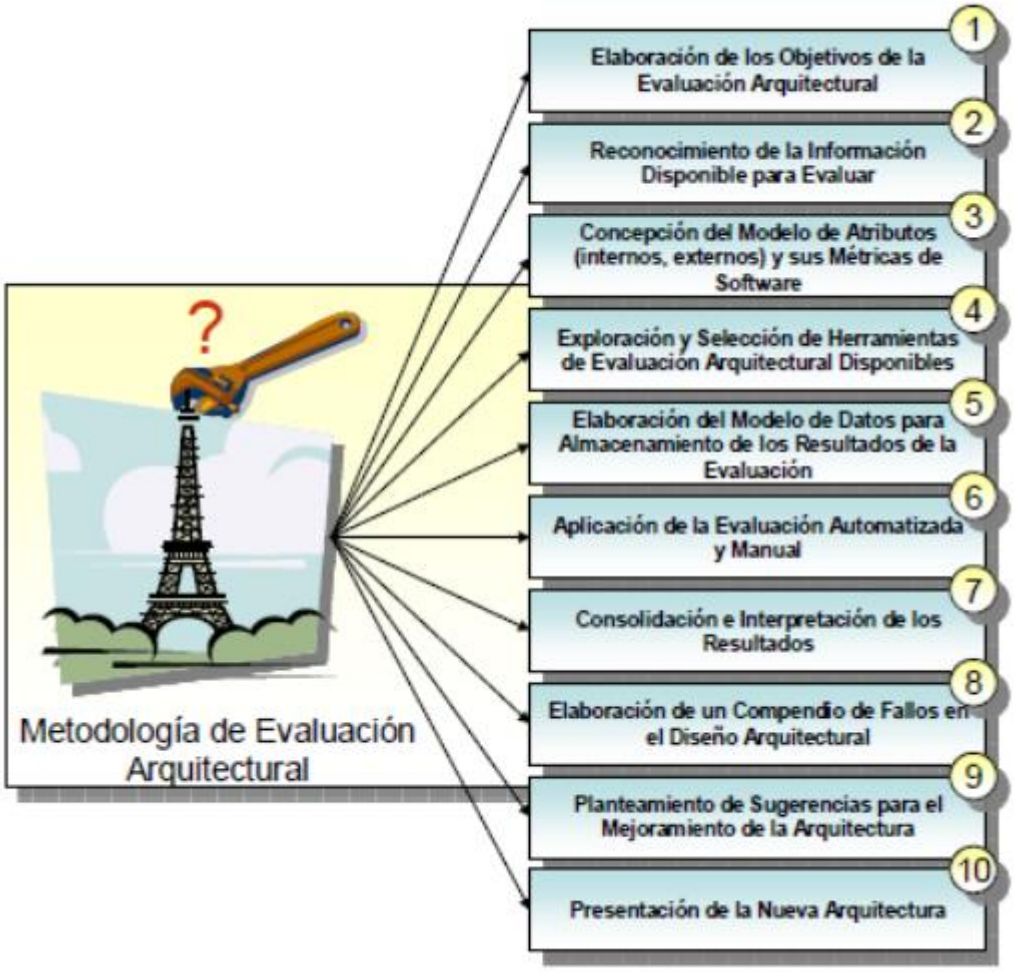
5.3.1 Metodología de Evaluación Arquitectónica. La metodología de evaluación arquitectónica aplicada a las versiones 3.5 y 4.0 de Evolución comprendió diez (10) pasos que abarcan desde el planteamiento de los objetivos mismos de la evaluación, hasta el nuevo diseño arquitectónico de Evolución 4.0 cuya arquitectura abierta al desarrollo en comunidad le permitirá convertirse en el nuevo ESMS-MI[20]. En la Figura 1, se ilustran los diez (10) pasos que guiaron la evaluación arquitectónica de Evolución 3.5 y 4.0. Estas etapas se describen a continuación en la misma secuencia que se muestra en la Figura 1.

[18] MORENO CHAUSTRE, Jorge Jair. Diseño de una arquitectura para un entorno de modelamiento – simulación y creación de un proceso para su desarrollo por una comunidad (I+D). Tesis de Maestría. Maestría en Informática. Bucaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas. 2006. 246 p.

[19] MACHADO MENDOZA, Gesman David y GONZALEZ PEREZ, Cesar Eduardo. Componente de sistema de inferencia difusa (FIS) para Evolución 3.5. Tesis de Pregrado. Bucaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas. 2006. 124 p.

[20] ESMS - MI: Entorno Software de Modelado y Simulación de Modelos Integrados.

Figura 1. Metodología Propuesta para Evaluación Arquitectónica[21]



5.3.2 Objetivos de Evaluación Arquitectónica.

- Verificar el estado de conformidad de la descripción arquitectónica actual de Evolución 4.0; y la arquitectura ya implementada en el código fuente entregado por SIMON.
- Valorar el estado de la arquitectura implementada con el propósito de encontrar fallos de diseño y sobre esta base proponer mejoras.
- Documentar la Arquitectura implementada de Evolución 4.0 integrando las mejoras sugeridas en la medida de lo posible.

[21] Fuente de información: Tesis de maestría "Diseño de una Arquitectura para un Entorno de Modelamiento-Simulación y Creación de un Proceso para su Desarrollo por una Comunidad (I+D)". Ing. Jorge Jair Moreno Chaustre.

5.3.3 Reconocimiento de la información disponible. La información disponible para llevar a cabo la evaluación arquitectónica de la herramienta Evolución 4.0 se obtuvo mediante la tesis de grado Componente de sistema de inferencia difusa (FIS) para Evolución 3.5[22]. La fuente de información disponible se distribuye en tres categorías: requisitos, diagramas y código fuente. El código fuente pertenece al lenguaje de alto nivel Delphi 7.0 y la versión de UML utilizada en los diagramas es la 2.0. A continuación en la Tabla 1, se enumeran los artefactos disponibles para la evaluación arquitectónica de Evolución 4.0.

Tabla 1. Información disponible para evaluar EVOLUCIÓN 4.0

Categoría	Información Disponible
Requisitos	Especificaciones de Casos de Uso: 14
	Número de Diagramas de Casos de Uso: 2
Diagramas	Número de Diagramas de Clase: 8
	Número de Diagramas de Secuencia: 3
	Número de Diagramas de Componentes: 0
	Número de Diagramas de Paquetes: 0
Código	Número de Clases: 57
	Número de Paquetes: 44
	Líneas de Código: 4.349

5.3.4 Atributos de calidad utilizados para evaluar a Evolución 3.5 y 4.0. Los atributos o características conforman la noción de calidad de un producto software y pueden ser externos o internos. Los atributos externos son *"visibles"* externamente (de ahí su nombre) por ejemplo la *confiabilidad* y la *mantenibilidad* pueden ser medidas en términos de cómo el software se relaciona con su entorno y solo cuando el producto ha sido creado. En cambio un atributo interno puede ser medido en términos del producto mismo, por ejemplo el *tamaño*, el *acoplamiento* y la *cohesión* pueden determinarse a partir de su representación UML o su código fuente. La descripción de los atributos de calidad y las métricas propuestas para valorar dichos atributos se encuentran definidos a continuación.

[22] MACHADO MENDOZA, Gesman David y GONZALEZ PEREZ, Cesar Eduardo. Componente de sistema de inferencia difusa (FIS) para Evolución 3.5. Tesis de Pregrado. Bucaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas. 2006. 124 p.

5.3.4.1 Arquitectura y sus Atributos de Calidad Externos. En la Figura 2, se ilustran las cualidades esenciales externas que un buen diseño arquitectónico debe reflejar según Pressman (2001), las cuáles se definen brevemente a continuación:

- **La conformidad funcional:** Una buena arquitectura de calidad debería implementar todos los requisitos explícitos contenidos en el modelo de análisis y debe acomodar todos los requisitos implícitos que desea el cliente.

- **Adaptabilidad:** Consiste en el nivel de esfuerzo requerido para realizar cambios en una arquitectura.

- **Modularidad:** Esta cualidad se concentra en promover el principio del ocultamiento de la información.

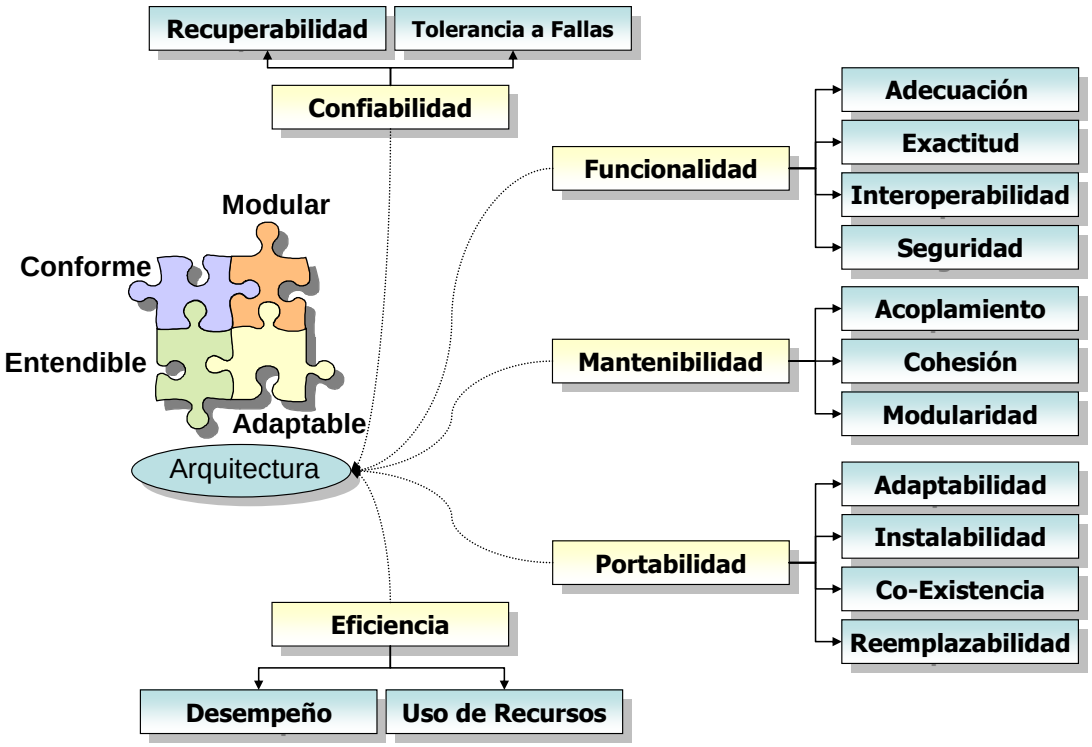
- **Entendible:** De acuerdo con Somerville (1998), el entendimiento estará afectado por: la cohesión, el acoplamiento, la nominación, la documentación y la complejidad.

    - **Cohesión:** Es una consecuencia del ocultamiento de la información. La meta es hacer que los componentes sean lo más cohesivos posible, asegurándose de que sus responsabilidades estén altamente relacionadas entre sí, ofreciendo el mayor nivel de funcionalidad posible, afirma Pfleeger (1998).

    - **Acoplamiento:** De acuerdo con Pressman (2001), este es un indicador de la fuerza de interconexión entre los componentes o elementos de la arquitectura. Los sistemas altamente acoplados tienen una fuerte interconexión, lo que se refleja en una gran dependencia entre sus componentes internos y una alta sensibilidad al mantenimiento. Mantener el acoplamiento en un nivel bajo es saludable para la robustez de la arquitectura, afirma Pfleeger (1998).

La Figura 2 muestra los atributos externos de la arquitectura que pueden medirse con el propósito de valorar su calidad y en la Tabla 2 se describen estos mismos atributos en relación con sus atributos subordinados. En consecuencia, la información que se especifica a continuación, se constituye en el modelo de métricas externo concebido para valorar las versiones 3.5 y 4.0 de Evolución.

Figura 2. Atributos de Calidad de la Arquitectura[23]



[23] Fuente de información: Tesis de maestría "Diseño de una Arquitectura para un Entorno de Modelamiento-Simulación y Creación de un Proceso para su Desarrollo por una Comunidad (I+D)". Ing. Jorge Jair Moreno Chaustre.

Tabla 2. Descripción de los Atributos Externos de Calidad

Atributo de Calidad	Definición	Sub-Atributo	Definición
Funcionalidad	Según Kazman (2001) consiste en la habilidad de un sistema para realizar el trabajo para el cual fue concebido.	**Adecuación / Corrección**	Grado en el que un programa satisface las especificaciones y cumple los objetivos del usuario.
		Exactitud / Fiabilidad	Grado en el que un programa se espera que realice su función con una precisión requerida. Por ejemplo: entrega de cálculos numéricos confiables.
		Interoperabilidad	Capacidad de interacción con sistemas externos.
		Seguridad / Integridad	Capacidad de brindar soporte al control de acceso a la información privada.
Confiabilidad	De acuerdo con Barbacci (1995), esta consiste en la medida de la habilidad que tiene un sistema para mantenerse operativo a lo largo del tiempo.	**Tolerancia a Fallas**	Capacidad para manejar excepciones.
		Recuperabilidad	Existencia de mecanismos o dispositivos de software para reestablecer el nivel de desempeño y recuperar datos.
Eficiencia	De acuerdo con Bass (2003), esta se refiere a la cantidad de comunicación e interacción existente entre los componentes del sistema.	**Desempeño**	Entrega de un rendimiento deseable durante la ejecución.
		Uso de Recursos	Utilización eficiente de los recursos aún en condiciones de escasez.
Mantenibilidad	Según Bosch (1999), esta consiste en la capacidad de modificar el sistema	**Acoplamiento**	Medida de la dependencia e interacción entre componentes.

Atributo de Calidad	Definición	Sub-Atributo	Definición
	de manera rápida y a bajo costo.	**Cohesión**	Consiste en el grado en que están relacionadas las responsabilidades de una clase o componente.
		Modularidad	Número de componentes que dependen de otro.
Portabilidad	De acuerdo con Pressman (2001), esta consiste en la habilidad del sistema para ser ejecutado en diferentes ambientes de computación.	**Adaptabilidad**	Capacidad para adaptarse durante su ejecución a una configuración cambiante de recursos.
		Instalabilidad	Facilidad de instalación en diversos ambientes de computación.
		Coexistencia	Facilidad para operar adecuadamente sin sufrir o causar perturbaciones de otro software ya instalado.
		Reemplazabilidad	Lista de componentes reemplazables para cada componente.

5.3.4.2 Arquitectura y sus Atributos de Calidad Internos. Una vez determinado el modelo de atributos externos, el siguiente paso consiste en concebir y armonizar una colección de métricas internas que valoren el estado de la información disponible (véase numeral 6.3.3) y a su vez permitan establecer el estado de algunos atributos externos del numeral anterior. A continuación, se presenta el proceso de concepción para un modelo de métricas interno que valorará la arquitectura implementada de las versiones 3.5 y 4.0 de Evolución.

De acuerdo con (Vásquez, et al. 2001), la armonización de un modelo de métricas interno exige trabajar simultáneamente elementos en las siguientes cinco (5) dimensiones (ver Figura 3): los compromisos o requisitos que debe cumplir, la forma en que debe medirse (naturaleza), lo que debe medir

(atributos), cuando debe medirse (fase o etapa del proceso de desarrollo) y por último con qué nivel de detalle (granularidad). La siguiente tabla relaciona de forma breve, cada uno de las dimensiones mostradas en la Figura 3 en correspondencia con sus elementos contenidos. Además se indica por cada elemento, las métricas aplicables en cada uno.

Figura 3. Características de un Modelo de Métricas para Evolución 3.5 y 4.0[24]

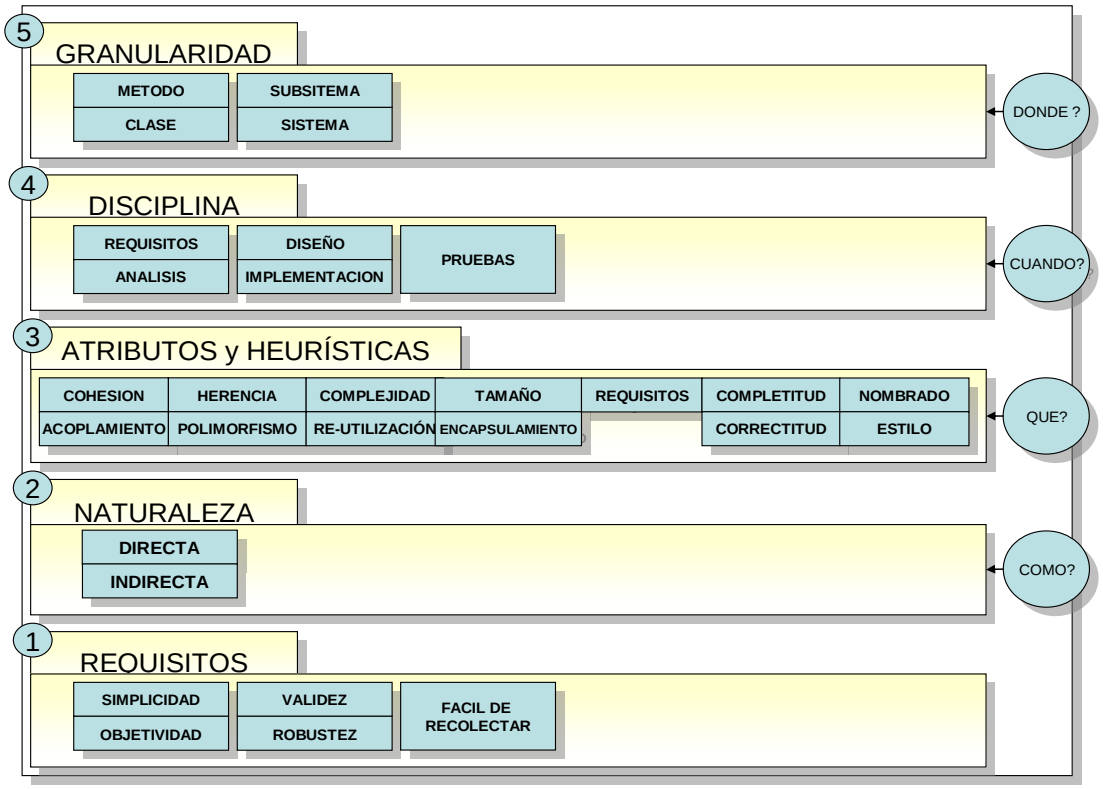


Tabla 3. Atributos de Calidad Internos

DIMENSION	ELEMENTO	DESCRIPCION
REQUISITOS	**SIMPLICIDAD**	La definición y uso de la métrica debe ser simple.
	OBJETIVIDAD	Diferentes personas han de darle valores idénticos. Esto le da consistencia evitando interpretaciones individuales.

[24] Fuente de información: Tesis de maestría "Diseño de una Arquitectura para un Entorno de Modelamiento-Simulación y Creación de un Proceso para su Desarrollo por una Comunidad (I+D)". Ing. Jorge Jair Moreno Chaustre.

DIMENSION	ELEMENTO	DESCRIPCION
	VALIDEZ	El coste y esfuerzo para obtener la medida debe ser razonable.
	ROBUSTEZ	Debe ser insensible a cambios no relevantes, permitiendo realizar comparaciones útiles.
	FACIL RECOLECCION	Debe medir lo que se supone que mide. Esto proporciona fiabilidad.
NATURALEZA	**DIRECTA**	Solo precisan de un único atributo para medirlo.
	INDIRECTA	No miden un atributo directamente. En su lugar, combinan varias métricas directas para medirlo.
ATRIBUTOS	**MODIFICABILI-DAD Y DEFECTOS POTENCIALES DE REQUISITOS**	De acuerdo con Kamstiems y Rombach (1997), detectar tempranamente los problemas en la definición de requerimientos es crucial a la hora de mejorar la calidad del proceso de desarrollo. Esto se debe a que el costo de reparar los defectos, se incrementa a medida que el proyecto avanza. Existen dos enfoques para valorar la calidad de los requerimientos: • **Manual:** Este enfoque aborda aspectos relacionados con la estabilidad, ambigüedad y trazabilidad entre requerimientos. • **Automatizado:** Este enfoque utiliza **NPL** (Natural Language Processing) para conocer el vocabulario usado, estilo de redacción, grado de concordancia entre la sintaxis y la semántica del texto, información faltante, oraciones no conectadas entre otros. La modificabilidad es una de las propiedades deseables acerca de la especificación de los requerimientos. IEEE (1993) define una especificación de requerimientos modificables como aquella estructura y estilo de texto que soporta el cambio con facilidad y de forma consistente conservando su esencia. Saeki (2003) propone un conjunto de métricas para valorar la modificabilidad de los diagramas de casos de uso. La idea básica consiste en que si un caso de uso necesita un cambio, probablemente otros casos de uso relacionados también lo necesiten. Las métricas propuestas son: **NOD[25] , NUCT[26].**

[25] Acrónimo de **Number of Dependencies**. (Saeki 2003).
[26] Acrónimo de **Number of Case Use Types**. (Saeki 2003).

DIMENSION	ELEMENTO	DESCRIPCION
		De otra parte, las métricas propuestas por Bernárdez et al. (2004) permiten la predicción de defectos potenciales en la especificación de requerimientos que a la larga impactarán la facilidad de mantenimiento a los mismos. Las métricas concebidas para tal propósito son: **NOS[27], NOAS[28], NOSS[29], NOUS[30], NOCS[31], NOE[32], NIE[33], RAAS[34], RASS[35], RUCS[36], CC[37], NAU[38], NMU[39] y NSCU[40].**
	COHESION	La cohesión consiste en el grado en que están relacionadas las responsabilidades de una clase o componente. Una alta cohesión indica que un elemento de diseño presenta pocas responsabilidades estrechamente relacionadas, al mismo tiempo que entrega una alta funcionalidad y es fácil de entender por separado. Por otra parte, un elemento con baja cohesión, presenta, muchas responsabilidades no relacionadas, no puede entenderse fácilmente si está aislado y será más difícil de reutilizar y mantener. Las métricas de cohesión están estrechamente relacionadas con las de tamaño. **LCOM[41]** es una métrica para valorar cohesión.

[27] Acrónimo de **Number of Steps of the Use Case**. (Bernárdez et al., 2004).
[28] Acrónimo de **Number of actor action steps of the use case**. (Bernárdez et al., 2004).
[29] Acrónimo de **Number of system action steps of the use case.** (Bernárdez et al., 2004).
[30] Acrónimo de **Number of use case action steps of the use case**. (Bernárdez et al., 2004).
[31] Acrónimo de **Number of conditional steps of the use case.** (Bernárdez et al., 2004).
[32] Acrónimo de **Number of Exceptions of the use case.** (Bernárdez et al., 2004).
[33] Acrónimo de **Number times the use case is included or extends others.** (Bernárdez et al., 2004).
[34] Acrónimo de **Rate of actor action steps of the use case.** (Bernárdez et al., 2004).
[35] Acrónimo de **Rate of system action steps of the use case.** (Bernárdez et al., 2004).
[36] Acrónimo de **Use case action steps of the use case.** (Bernárdez et al., 2004).
[37] Acrónimo de **Cyclomatic Complexity of the use cases.** (Bernárdez et al., 2004).
[38] Acrónimo de **Number of Actors Associated to a Use Case.** (Kim y Boldyreff., 2002).
[39] Acrónimo de **Number of messages associated with a Use Case.** (Kim y Boldyreff., 2002).
[40] Acrónimo de **Number of System Classes Associated with a Use Case).** (Kim y Boldyreff., 2002).
[41] Acrónimo de **Lack of Cohesion in Methods**. (Chidamber y Kemerer, 1994).

DIMENSION	ELEMENTO	DESCRIPCION
	ACOPLAMIENTO	Consiste en la medida de la dependencia entre los elementos de un diseño. Una alta dependencia, causará necesariamente un impacto sobre la calidad del producto final, específicamente sobre la facilidad de mantenimiento y de prueba. Un buen principio de diseño consiste en minimizar estas dependencias. Las métricas: **CBO**[42]**, COF**[43]**, Dep_Out**[44]**, Dep_In**[45]**, NumAssEl_ssc**[46]**, NumAssEl_sb**[47]**, NumAssEl_nsb**[48]**, EC_Attr**[49]**, IC_Attr**[50]**, EC_Par**[51]**, IC_Par**[52] **, StimSent**[53]**, StimRecv**[54]**, MsgSent**[55]**, MsgRecv**[56] están concebidas para valorar el acoplamiento de un sistema y arrojarán como resultado a aquellos elementos de diseño que presentan una alta densidad de fallas.

[42] Acrónimo de **Coupling Between Objects**. (Chidamber y Kemerer, 1994).
[43] Acrónimo de **Coupling Factor**. (Abreu y Melo, 1996).
[44] El número de dependencias (asociaciones) donde la clase es el cliente. (Catálogo de SDMetrics).
[45] El número de dependencias (asociaciones) donde la clase es el proveedor. (Catálogo de SDMetrics).
[46] Para una clase en el paquete ***p***, cuenta únicamente elementos asociados en el mismo paquete ***p.*** (Catálogo de SDMetrics).
[47] Para una clase en el paquete ***p***, cuenta únicamente elementos asociados en los paquetes (contenidos o contenedores) de ***p.*** (Catálogo de SDMetrics).
[48] Para una clase en el paquete ***p***, cuenta únicamente elementos asociados en otros paquetes, **no** contenidos ni contenedores de ***p.*** (Catálogo de SDMetrics).
[49] Número de veces que la clase es usada externamente como un atributo. (Catálogo de SDMetrics).
[50] Número de atributos en la clase que tienen como tipo a otras clases o interfaces. (Briand et. al., 1997).
[51] Número de veces que la clase es utilizada como parámetro. (Briand et. Al 1997).
[52] Número de Parámetros en la clase cuyo tipo es clase o interfaz. (Briand et. Al 1997).
[53] Número de estímulos que los objetos de la clase, envían a objetos de otras. (Briand et. Al 1997).
[54] Número de estímulos que los objetos de la clase reciben de objetos de otras. (Briand et. Al 1997).
[55] Número de mensajes que las instancias de la clase envían a instancias de otras.(Briand et. Al 1997).
[56] Número de mensajes que las instancias de la clase reciben de instancias de otras.(Briand et. Al 1997).

DIMENSION	ELEMENTO	DESCRIPCION
	HERENCIA	La herencia debe valorarse con base en aspectos tales como: profundidad y anchura del árbol de herencia, número de ancestros y descendientes de un elemento de diseño, polimorfismo, sobre-escritura de métodos entre otros. Por ejemplo: el entendimiento de un elemento de diseño ubicado en un nivel determinado del árbol de herencia, depende del entendimiento de sus ancestros, así mismo una modificación en un elemento de diseño, afectará necesariamente a todos sus descendientes si no se ha previsto la situación. El uso adecuado o inadecuado de la herencia puede afectar a la calidad del producto. Sin embargo, la herencia es poco usada en los elementos de diseño, en otras palabras, pocas clases del sistema estarán involucradas en relaciones de herencia, de aquí que las métricas relacionadas con la herencia, son poco usadas y difíciles de aplicar. Algunas métricas propuestas son: **NumAnc[57], NumDesc[58]**, **NOC[59]**, **MIF[60], AIF[61], SIX[62] y DIT[63].**
	POLIMORFISMO	Esta propiedad indica la posibilidad de que una operación tome muchas formas. Es decir, permite referirse a objetos de clases diferentes mediante el mismo elemento de programa y realizar la misma operación de diferentes formas, según sea el objeto que se referencia en ese momento. La métrica propuesta para valorar polimorfismo es: **POF[64]**.

[57] Cuenta el número total de ancestros de la clase. Si no se usa herencia múltiple, entonces arroja el mismo valor que DIT. (Chidamber y Kemerer, 1994).
[58] Cuenta el número total de descendientes de la clase a través de toda una jerarquía. (Lake 1994).
[59] Cuenta el número total directo de descendientes de una clase. (Chidamber y Kemerer, 1994).
[60] Acrónimo de **Method Inheritance Factor**. (Abreu y Melo, 1996).
[61] Acrónimo de **Attribute Inheritance Factor**. (Abreu y Melo, 1996).
[62] Acrónimo de **Specialisation Index per Class**. (Lorentz y Kidd, 1994).
[63] Calcula el camino más largo desde la clase raíz de una jerarquía hasta la clase. (Chidamber y Kemerer, 1994).
[64] Acrónimo de **Polymorphism Factor**. (Abreu y Melo, 1996).

DIMENSION	ELEMENTO	DESCRIPCION
	COMPLEJIDAD	La complejidad mide el grado de conectividad (relaciones/dependencias) entre los elementos de un diseño. Por ejemplo: el número de llamados entre métodos de una misma clase puede considerarse como una medida de la complejidad de la clase. Una alta complejidad, impacta directamente sobre la facilidad de entendimiento del modelo y su posterior esfuerzo para realizarle pruebas. En la práctica, la complejidad está estrechamente relacionada con el tamaño. Algunas métricas propuestas para valorar la complejidad son: **RFC[65], WMC[66], v(G)[67],** y **MsgSelf[68].**
	REUTILIZACION	El uso correcto de la herencia en los sistemas, permite alcanzar niveles de reutilización deseables. La métrica propuesta para valorar este atributo es: **NOC[69].**
	ENCAPSULAMIENTO	El encapsulamiento (ocultamiento de la información) permite que un objeto pueda acceder a sus datos mediante sus propios métodos al mismo tiempo que los esconde de los demás objetos, esto asegura que los objetos no pueden cambiar el estado interno de otros objetos de maneras inesperadas; solamente los propios métodos internos del objeto pueden acceder a su estado. Las métricas propuestas para valorar el encapsulamiento son: **MHF[70], AHF[71].**

[65] Acrónimo de **Response For a Class**. (Chidamber y Kemerer, 1994).
[66] Acrónimo de **Weighted Methods per Class**. (Chidamber y Kemerer, 1994).
[67] Acrónimo de **Average**. (McCabe 1996).
[68] Cuenta el número de mensajes que instancias de esta clase envía a sí misma o a instancias de la misma clase. (Lee et. al., 1995).
[69] Acrónimo de **Number of Children**. (Chidamber y Kemerer, 1994).
[70] Acrónimo de **Method Hiding Factor**. (Abreu y Melo, 1996).
[71] Acrónimo de **Attribute Hiding Factor**. (Abreu y Melo, 1996).

DIMENSION	ELEMENTO	DESCRIPCION
	TAMAÑO	Las métricas **(LOC[72], NOM[73], NumAttr[74], NumOps[75], NumPubOps[76], Setters[77] y Getters[78])** orientadas a medir el tamaño del diseño básicamente consisten en el conteo de los elementos contenidos en el modelo. Por ejemplo el número de operaciones en una clase o el número de clases en un paquete. Este tipo de métricas son adecuadas para estimar adecuadamente el coste y esfuerzo para la implementación, revisión, prueba o mantenimiento. Generalmente estas estimaciones son la base para la asignación de personal durante la planificación del proyecto. La facilidad de mantenimiento, reuso y entendimiento son afectados por el número de componentes involucrados en un elemento de diseño. Por último, el tamaño es el principal factor a considerar a la hora de estimar el costo de un proyecto de software.
HEURISTICAS (REGLAS DE DISEÑO)	**CLASE NO USADA**	**La clase no es usada en ninguna parte.** En otras palabras, la clase no tiene clases hijas, dependencias, o asociaciones ni tampoco es usada como parámetro en algún método de otra clase ni como atributo. Probablemente, se necesite modelar los clientes de la clase o considerar borrarla del modelo. Esta regla pertenece a la característica de la COMPLETITUD y su severidad se considera como “Alta”. (Riel 96). **Nota:** Para modelos que fueron generados a partir del código (reverse engineering), la herramienta SDMetrics puede reportar violaciones falsas a esta regla**.** Esto sucede únicamente para las clases que son referenciadas en la implementación de los métodos a través de variables.

[72] Acrónimo de **Lines of Code per Method**. (Chidamber y Kemerer, 1994).
[73] Acrónimo de **Number of Messages Send**. (Lorentz y Kidd, 1994).
[74] Acrónimo de **Number of Attributes for a Class.** (Lorentz y Kidd, 1994).
[75] Acrónimo de **Number of Operations in a Class.**
[76] Acrónimo de **Number of Public Operations in a Class.**
[77] Acrónimo de **Number of Operations with a name starting with 'Set'.**
[78] Acrónimo de **Number of Operations with a name starting with 'Get', 'Is', or 'Has'.**

DIMENSION	ELEMENTO	DESCRIPCION
	CLASE NO REPRESENTA-DA	**La clase del código no es representada en ninguna parte del modelo.** En otras palabras, la clase que fue hallada mediante la generación del código al modelo, no se encuentra representada en ningún diagrama de la documentación oficial de la herramienta. Esta regla pertenece a la característica de la COMPLETITUD y su severidad se considera como “Alta”.
	PAQUETE NO REPRESENTA-DO	**El Paquete del Código no es representado en ninguna parte del modelo.** En otras palabras, el paquete que fue hallado mediante la generación del código al modelo, no se encuentra representado en ningún diagrama de la documentación oficial de la herramienta. Esta regla pertenece a la característica de la COMPLETITUD y su severidad se considera como “Alta”.
	CLASE ENORME	**La clase es enorme también conocida como clase “dios”.** Esta situación se presenta cuando una clase tiene entre atributos y operaciones más de 60. Este tipo de clases son un cuello de botella para el mantenimiento del software, también causan problemas de confiabilidad e indican una debilidad en la concepción de una buena arquitectura orientada a objetos. Esta regla pertenece a la categoría de ESTILO y su severidad de considera “Media”. (Fowler, 1999).
	OPERACIÓN DUPLICADA	**La clase tiene operaciones duplicadas.** Consiste en que existen en la clase, dos o más operaciones con signaturas idénticas (nombre de operación y lista de parámetros con sus tipos). Las signaturas de operación deben ser únicas para las clases. Esta regla pertenece a la categoría de CORRECTITUD y su severidad se considera como “Alta”. Esta regla está sugerida en los estándares OMG[79] 03 y OMG 05 que todo modelo UML válido debe cumplir.

[79] Consorcio OMG (Object Management Group).

DIMENSION	ELEMENTO	DESCRIPCION
	NOMBRE DE CLASE INCORRECTO	**La clase ha sido representada con un nombre que no se corresponde con el código.** En la documentación oficial de la herramienta el nombre de la clase es diferente al nombre real que fue encontrado en el código fuente de la misma. Esta regla pertenece a la categoría de CORRECTITUD y su severidad de considera como "Media".
	REFERENCIAS CIRCULARES	**La clase tiene referencias circulares.** Las dependencias circulares deberían evitarse. Las clases que participan en el ciclo, no pueden probarse ni reusarse de forma independiente. Entre más clases participen en el ciclo, peor es el problema, especialmente si las clases residen en diferentes paquetes. Debe revisarse el diseño para eliminar el ciclo de dependencia. Esta regla pertenece a la categoría de "ESTILO" y su severidad se considera "Media". (SDMetrics – Measurement Catalog).
	SOBRE- ESCRITURA DE ATRIBUTO HEREDADO	**La clase define un atributo del mismo nombre que un atributo heredado.** Durante la generación de código, esta situación puede ocultar inadvertidamente el atributo de la clase padre. Debería cambiarse el nombre del atributo en la clase hija. Esta regla pertenece a la categoría de NOMBRADO y su severidad se considera como "Media". (Ramírez et. al., 2004).
DISCIPLINA	**REQUISITOS**	**Métricas Aplicables:** NOD, NUCT, NOS, NOAS, NOSS, NOUS, NOCS, NOE, NIE, NOAS/NOS, NOSS/NOS, NOUS/NOS y CC
	DISEÑO	**Métricas Aplicables:** CBO, COF, Dep_Out, Dep_In, NumAssEL_ssc, NumAssEL_sb, NumAssEL_nsb, EC_Attr, IC_Attr, IC_Par, StimSent, StimRecv, MsgSent, MsgRecv, NumAnc, NumDesc, MsgSelf, LCOM, RFC, WMC, MIF, SIX, DIT, POF y NOC.
	IMPLEMENTACION	**Métricas Aplicables:** LCOM, v(G), MHF, AHF, MIF, SIX, POF, y LOC.
	PRUEBAS	**Métricas Aplicables:** ICC[80], SCC[81] y UCC[82].
GRANULARIDAD	**METODO**	**Las métricas aplicables en este contexto son:** CBO,

[80] Acrónimo de **Inhheritance Context Coverage.** (IPL 1999).
[81] Acrónimo de **State-based Context Coverage.** (IPL 1999).
[82] Acrónimo de **Multi-trheaded Context Coverage.** (IPL 1999).

DIMENSION	ELEMENTO	DESCRIPCION
		COF, RFC, WMC, MsgSelf, v(G), MIF, AIF, LOC y NOM.
	CLASE	**Las métricas aplicables en este contexto son:** CBO, COF, RFC, WMC, v(G), MHF, AHF, MIF, AIF, SIX, DIT, POF, NOC, LOC, Dep_Out, Dep_In, NumAssEL_ssc, NumAssEL_sb, NumAssEL_nsb, EC_Attr, IC_Attr, IC_Par, StimSent, StimRecv, MsgSent, MsgRecv, NumAnc, NumDesc.
	PAQUETE	**Las métricas aplicables en este contexto son:** CBO, COF, RFC, WMC, v(G), MIF, AIF y LOC.
	SISTEMA	**Las métricas aplicables en este contexto son:** CBO, COF, RFC, WMC, v(G), MIF, AIF y LOC.
	MODELO DE CASOS DE USO	**Las métricas aplicables en este contexto son:** NOD, NUCT, NOS, NOAS, NOSS, NOUS, NOCS, NOE, NIE, NOAS/NOS, NOSS/NOS, NOUS/NOS y CC.

En la carpeta Anexos/Anexo1_Tesis_de_Maestria/Anexos/Métricas_de_Calidad_para_Arquitectura**,** se describe de forma detallada las métricas propuestas para evaluar la arquitectura de software de la herramienta Evolución 3.5 y 4.0. Las métricas se organizan por atributos internos de la arquitectura (tamaño, acoplamiento, cohesión, encapsulamiento, herencia, polimorfismo y requisitos). Adicionalmente cada métrica está acompañada de su definición, valoración y autor(es)**.**

En el siguiente numeral, se enuncian las herramientas utilizadas para apoyar la evaluación arquitectónica de Evolución 3.5 y 4.0. Para cada herramienta se provee una breve descripción así como una ilustración de su interfaz gráfica de usuario.

5.3.5 Exploración y selección de herramientas disponibles para realizar la evaluación. Durante el desarrollo de la evaluación arquitectónica, se exploró en diversas fuentes (como Internet) en busca de herramientas que pudieran llevar a cabo parcial o totalmente la evaluación de todos los artefactos

(requisitos, diagramas, modelos y código) disponibles para el software Evolución 3.5 y 4.0. A continuación se provee una breve descripción de las herramientas encontradas y seleccionadas para apoyar la evaluación arquitectónica.

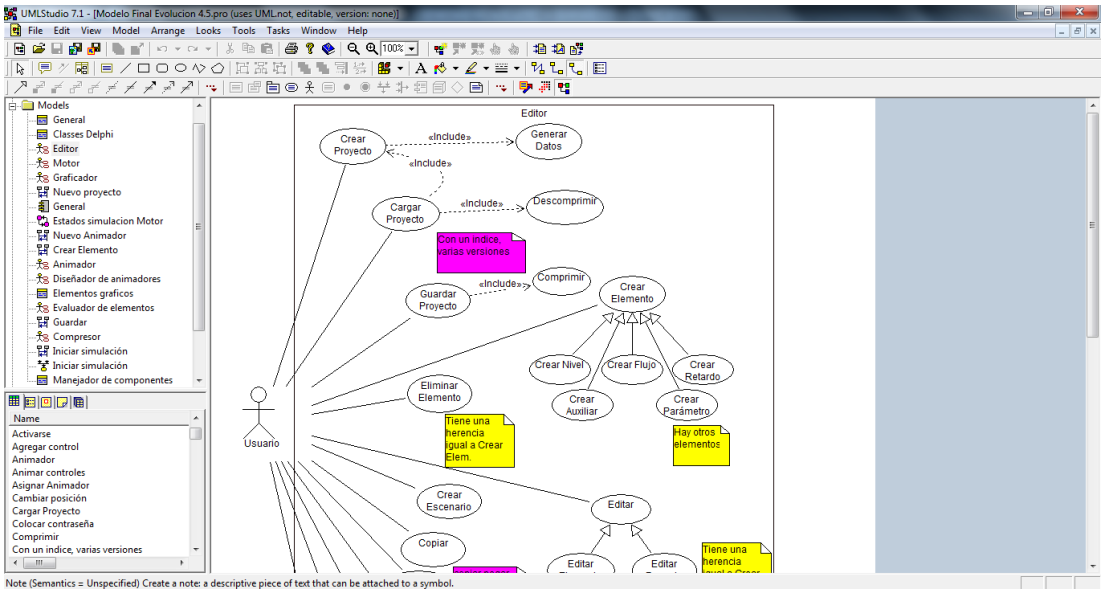
- **UMLStudio.** Esta herramienta ayuda a los desarrolladores de software a ser más sistemáticos y productivos en sus proyectos. Por medio del modelado, los desarrolladores pueden visualizar, analizar y comunicar sus ideas de forma más efectiva en el marco de sistemas de software grandes y complejos. Además de los modelos gráficos, esta herramienta permite la generación automática de código fuente genérico que luego puede ser completado y adecuado a las necesidades de los equipos de desarrollo. Esta herramienta CASE[83] provee las siguientes capacidades: Uno o más editores para la creación de diagramas en notación UML[84], características de validación sintáctica y semántica de los diagramas, un generador de código automático y capacidad de ingeniería inversa y un repositorio en el cuál los diagramas y sus atributos puedan almacenarse y recuperarse. Todos los modelos entregados por SIMON, están elaborados en esta herramienta. La Figura 4 muestra el aspecto de esta herramienta. Como limitaciones de la versión usada pueden encontrarse las siguientes:

    - Las capacidades de importación/exportación están restringidas.
    - De igual forma la herramienta sólo permite abrir ficheros propietarios a diferencia de otras que aceptan formatos como XMI[85] y XML[86].
    - Sus capacidades de ingeniería inversa, no incluyen a Delphi.

[83] *CASE:* Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora.
[84] *UML:* Unified Modeling Language. Lenguaje Unificado de Modelado
[85] XMI o XML Metadata Interchange (XML de Intercambio de Metadatos) es una especificación para el Intercambio de Diagramas.

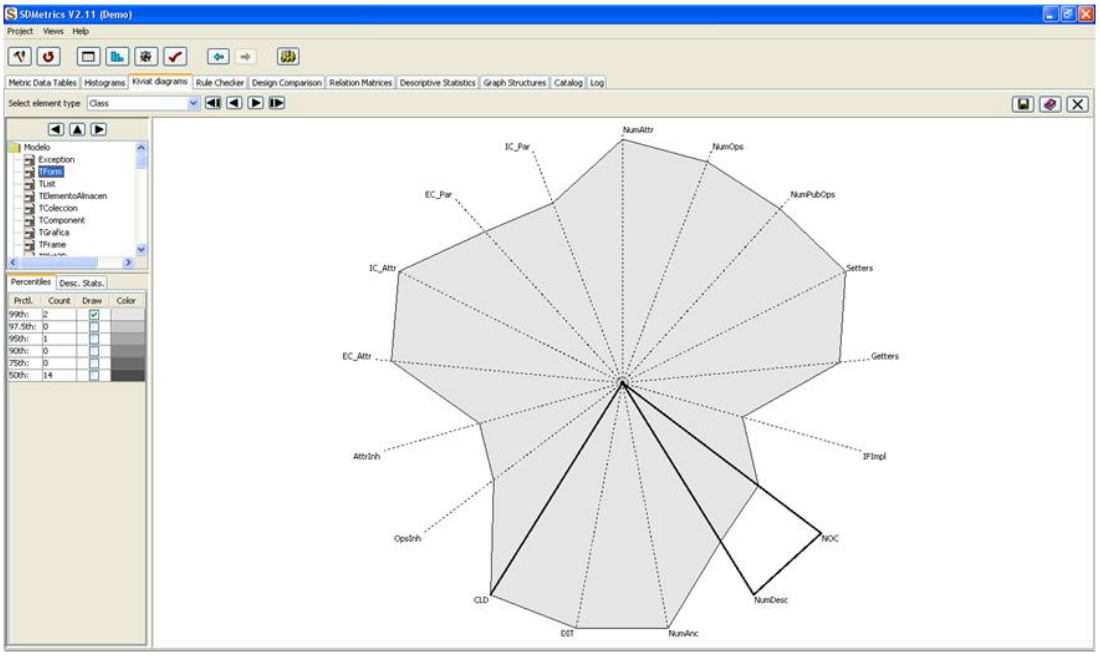
[86] XML: Extensible Markup Language.

Figura 4. Interfaz de UMLStudio



- **SDMetrics.** Esta herramienta provee algunas capacidades automáticas para analizar y valorar un modelo ya generado a partir de código fuente con relación a un conjunto de métricas de software y heurísticas de diseño. Necesita como entrada un archivo en formato XMI y como salida provee tres vistas: Métrica, Elemento y Tabla. Las métricas que es capaz de evaluar están agrupadas en: Tamaño, Acoplamiento, Complejidad y Herencia. De otra parte las heurísticas de diseño valoradas se agrupan en: Completitud, Correctitud, Nombrado y Estilo entre otras. Debido a que la versión utilizada es una ***"Demo"***, las capacidades de exportación de datos a otros formatos está restringida, por lo tanto el copiado y organización de estos últimos tuvo que hacerse manualmente. La Figura 5 muestra el aspecto de la herramienta.

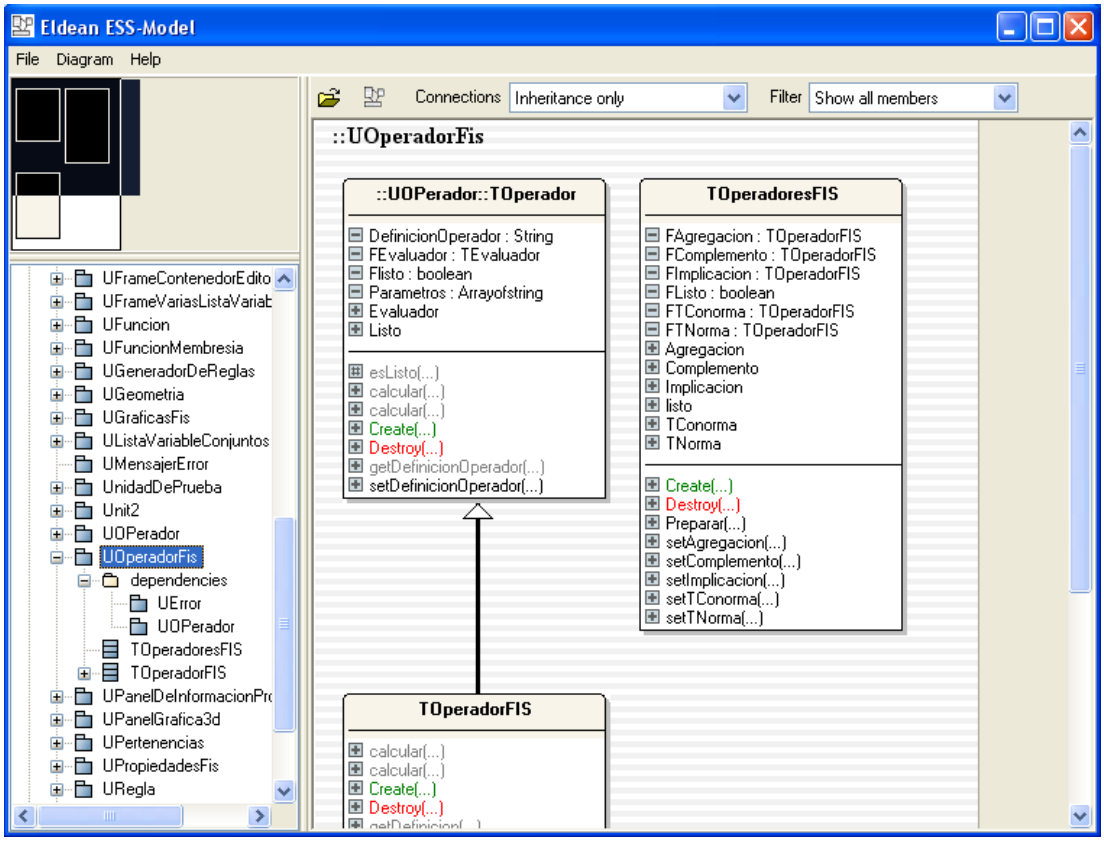
Figura 5. Interfaz de SDMetrics



SDMetrics®

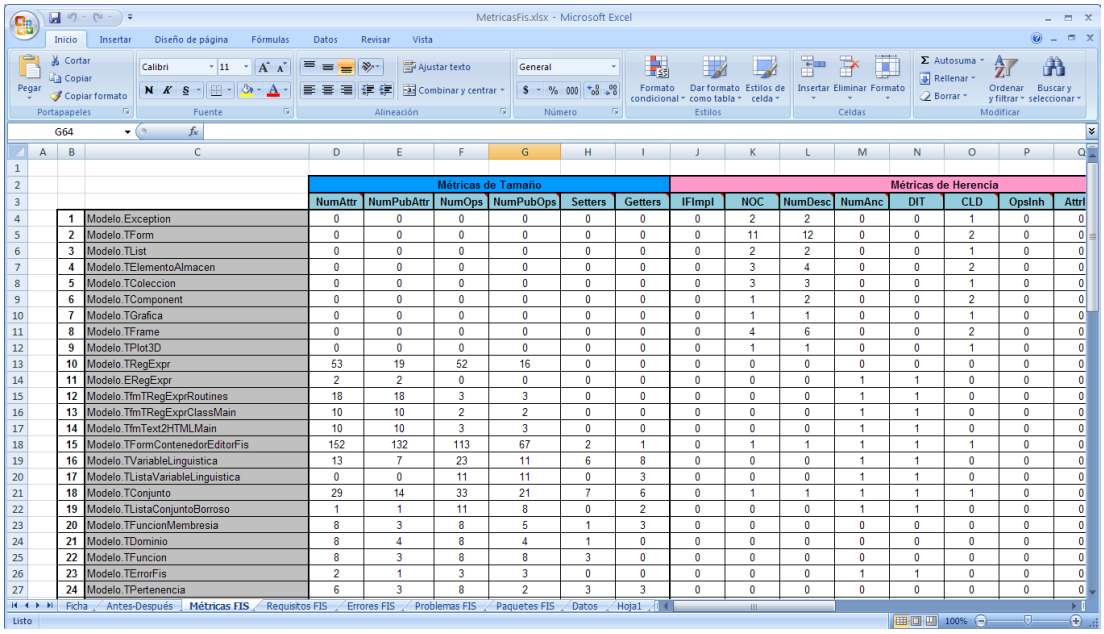
- **EssModel**. Esta herramienta permite la generación de un archivo en formato XMI a partir de código fuente escrito en C, Java o Delphi. Por otra parte, permite la navegación y edición a través del modelo generado a partir del código fuente con el propósito de crear la documentación del mismo en formato de HTML. El uso de este software fue indispensable debido a que es capaz de generar el archivo XMI que necesita SDMetrics para evaluar al código fuente, además fue bastante útil a la hora de calcular algunas métricas importantes de forma manual que SDMetrics no provee. La Figura 6 muestra el aspecto de la herramienta.

Figura 6. Interfaz de EssModel



- **Microsoft Excel.** Esta herramienta sirvió para la tabulación, organización e interpretación de los datos derivados de la evaluación arquitectónica realizada sobre Evolución 3.5 y 4.0. La Figura 7 muestra el aspecto de la herramienta.

Figura 7. Interfaz de Microsoft Excel

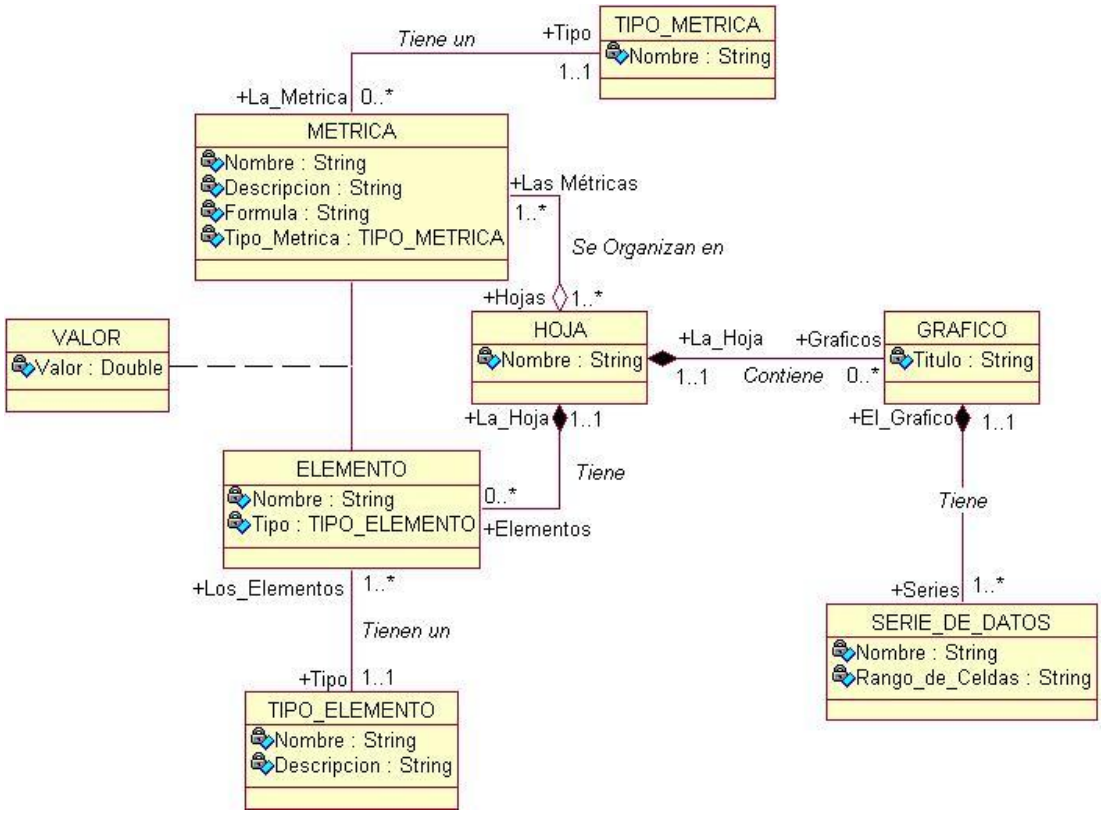


MetricasFis.xlsx - Microsoft Excel

			Métricas de Tamaño						Métricas de Herencia							
			NumAttr	NumPubAttr	NumOps	NumPubOps	Setters	Getters	IFImpl	NOC	NumDesc	NumAnc	DIT	CLD	OpsInh	Attrl
1	Modelo.Exception		0	0	0	0	0	0	0	2	2	0	0	1	0	0
2	Modelo.TForm		0	0	0	0	0	0	0	11	12	0	0	2	0	0
3	Modelo.TList		0	0	0	0	0	0	0	2	2	0	0	1	0	0
4	Modelo.TElementoAlmacen		0	0	0	0	0	0	0	3	4	0	0	2	0	0
5	Modelo.TColeccion		0	0	0	0	0	0	0	3	3	0	0	1	0	0
6	Modelo.TComponent		0	0	0	0	0	0	0	1	2	0	0	2	0	0
7	Modelo.TGrafica		0	0	0	0	0	0	0	1	1	0	0	1	0	0
8	Modelo.TFrame		0	0	0	0	0	0	0	4	6	0	0	2	0	0
9	Modelo.TPlot3D		0	0	0	0	0	0	0	1	1	0	0	1	0	0
10	Modelo.TRegExpr		53	19	52	16	0	0	0	0	0	0	0	0	0	0
11	Modelo.ERegExpr		2	2	0	0	0	0	0	0	0	1	1	0	0	0
12	Modelo.TfmTRegExprRoutines		18	18	3	3	0	0	0	0	0	1	1	0	0	0
13	Modelo.TfmTRegExprClassMain		10	10	2	2	0	0	0	0	0	1	1	0	0	0
14	Modelo.TfmText2HTMLMain		10	10	3	3	0	0	0	0	0	1	1	0	0	0
15	Modelo.TFormContenedorEditorFis		152	132	113	67	2	1	0	1	1	1	1	1	0	0
16	Modelo.TVariableLinguistica		13	7	23	11	6	8	0	0	0	1	1	0	0	0
17	Modelo.TListaVariableLinguistica		0	0	11	11	0	3	0	0	0	1	1	0	0	0
18	Modelo.TConjunto		29	14	33	21	7	6	0	1	1	1	1	1	0	0
19	Modelo.TListaConjuntoBorroso		1	1	11	8	0	2	0	0	0	1	1	0	0	0
20	Modelo.TFuncionMembresia		8	3	8	5	1	3	0	0	0	0	0	0	0	0
21	Modelo.TDominio		8	4	8	4	1	0	0	0	0	0	0	0	0	0
22	Modelo.TFuncion		8	3	8	8	3	0	0	0	0	0	0	0	0	0
23	Modelo.TErrorFis		2	1	3	3	0	0	0	0	0	1	1	0	0	0
24	Modelo.TPertenencia		6	3	8	2	3	3	0	0	0	0	0	0	0	0

5.3.6 Elaboración del modelo de datos para la evaluación. Una vez reconocido el universo de artefactos disponible (requisitos, diagramas y código) y las métricas que se aplicarían sobre estos, se determinó un diseño de datos simple sobre la herramienta Microsoft Excel que permitiera la tabulación de los valores de cada métrica por cada elemento de evaluación (caso de uso, clase, paquete e interfaz). A continuación en la Figura 8 se ilustra el modelo de datos que subyace en el libro de Microsoft Excel "*Métricas_Evolución_4.0.xls*".

Figura 8. Modelo de Datos para la Hoja de Excel "Métricas_Evolución_4.0.xls"



En la Tabla 4 se provee una breve descripción de cada concepto mostrado en la Figura 8.

Tabla 4. Elementos del Modelo de Datos de Evaluación Arquitectónica

CONCEPTO	DESCRIPCION
TIPO__METRICA	Las métricas calculadas pertenecen a varias categorías: Tamaño, Acoplamiento, Herencia, Complejidad etc. En el archivo "Métricas_Evolución_4.0.xls" se agruparán visualmente las métricas según su tipo (Tamaño, Herencia, Acoplamiento etc.).
METRICA	Corresponde a la métrica que se valora, la cual se describe por su acrónimo, descripción (motivación y autor) y su fórmula. Cada métrica es calculada para todos y cada uno de los elementos según corresponda.
ELEMENTO	Corresponde al objeto evaluado, el cuál puede ser una clase, un paquete, un caso de uso, entre otros. Los elementos se agrupan por módulos u hojas de cálculo en el libro "Métricas_Evolución_4.0.xls".
TIPO_ELEMENTO	Indica su naturaleza: Clase, Diagrama, Caso de Uso, Paquete entre otros.

CONCEPTO	DESCRIPCION
VALOR	Almacena el resultado final de la fórmula de cada métrica por cada elemento evaluado.
HOJA	Corresponde a las hojas del libro "Métricas_Evolución_4.0.xls". Cada hoja, almacena un conjunto de métricas y elementos relacionados, aunque es posible que existan solo hojas de resultados. Se crearon alrededor de 6 hojas de cálculo en el archivo Métricas_Evolución_4.0.xls, y una cantidad menor de hojas de gráficos en el archivo Gráficos_Evolución_4.0.xls.
GRAFICO	Se incrustan en las hojas con el objeto de resumir y consolidar visualmente los datos tabulados. Se proveen tres tipos de perspectivas en los gráficos: Comparación entre métricas del mismo tipo, comparación de métricas de todos los tipos para elementos del mismo módulo y comparación entre atributos para el módulo. Los gráficos usados son del tipo *barras horizontales* con el propósito de facilitar la comparación. Además se incluyeron otros gráficos de más alto nivel con el objeto de soportar las conclusiones arrojadas. Todos los gráficos se encuentran en el archivo Gráficos_Evolución_4.0.xls.
SERIE_DATOS	Cada gráfico se *alimenta* de un conjunto de datos que puede agruparse por series, las cuáles son representadas por distintos colores con el objeto de diferenciarlas unas de otras.

Finalmente es importante destacar que aunque el código fuente no refleja el empaquetamiento sugerido por el modelo elaborado en la herramienta UMLStudio**,** ni el documento de tesis "Evolución 3.5. herramienta software para el modelamiento y simulación con dinámica de sistemas", los archivos Métricas_Evolución_4.0.xls y Gráficos_Evolución_4.0.xls[87] sí reflejan esta estructura con el objeto de respetar y mantener el espíritu original de los autores en la concepción de Evolución 3,5 y 4.0**,** aspecto que hará más comprensible a los desarrolladores de SIMON interpretar los datos de la evaluación arquitectónica aquí presentada.

Los apartados que se mencionan a continuación, son parte de la METODOLOGÍA DE EVALUACIÓN ARQUITECTURAL, los cuales se desarrollan en el Capítulo 7.1 RESULTADOS DE LA EVALUACIÓN ARQUITECTÓNICA.

[87] Nombre de los archivos obtenidos en los resultados de la Evaluación Arquitectónica de Evolución 4.0

- Aplicación de la Evaluación Automatizada y Manual
- Consolidación e Interpretación de Resultados
- Elaboración de un Compendio de Fallos en el Diseño Arquitectural
- Planteamiento de Sugerencias para el Mejoramiento de la Arquitectura
- Presentación de la Nueva Arquitectura

5.4 ANTECEDENTES

5.4.1 TESIS DE MAESTRIA "DISEÑO DE UNA ARQUITECTURA PARA UN ENTORNO DE MODELAMIENTO - SIMULACIÓN Y CREACION DE UN PROCESO PARA SU DESARROLLO POR UNA COMUNIDAD (I+D)".

AUTOR: ING. JORGE JAIR MORENO CHAUSTRE.

Esta tesis propone un diseño lógico para la arquitectura de un Entorno Software de Modelado y Simulación (ESMS) que permita extender la funcionalidad entregada por la herramienta Evolución (desarrollada por SIMON), hacia la integración de componentes de modelado y simulación para otras herramientas de representación de conocimiento, desarrolladas por grupos de investigación (I+D) geográficamente dispersos, pertenecientes a la comunidad latinoamericana de modelado y simulación en dinámica de sistemas.

Además, con el propósito de apoyar el trabajo de los equipos (I+D) geográficamente dispersos cuando trabajan cooperativamente (desarrollando componentes para Evolución), este proyecto propone una metodología de desarrollo de software en comunidad soportada mediante una plataforma de servicios en Internet, que integran herramientas adecuadas para la gestión y documentación de proyectos, así como el control, comunicación y coordinación de los equipos de desarrollo bajo un ambiente distribuido.

La tesis de J. Moreno plantea la creación de una arquitectura de software que tiene como punto de partida la herramienta Evolución 3.5, debido a que su concepción arquitectónica es la más madura para realizar un tratamiento sobre

la escalabilidad hacia un entorno más sofisticado para el modelamiento y la simulación. Por tanto para la concepción de la nueva arquitectura, se consideró fundamental e indispensable la realización de una evaluación arquitectónica que mediante la aplicación de un modelo de métricas sobre Evolución 3.5, permitió conocer su estructura interna, el estado de sus atributos de calidad, los posibles problemas de diseño subyacentes y finalmente las directrices para su eventual mejora.

Finalmente, este trabajo, propone el diseño mismo de un proyecto comunitario de desarrollo de software para implementar las características especificadas en la arquitectura lógica concebida para el ESMS[24], usando la metodología de software propuesta sobre la plataforma de servicios seleccionada.

Por las métricas utilizadas para evaluar los atributos de calidad y los resultados obtenidos en la evaluación de la herramienta Evolución 3.5, se hace referencia a dicha tesis de maestría.

5.4.2 TESIS DE PREGRADO "EVOLUCION 3.5 HERRAMIENTA SOFTWARE PARA EL MODELAMIENTO Y SIMULACION CON DINAMICA DE SISTEMAS".

AUTORES: MARIO CUELLAR YENERIS Y EMILIANO LINCE MERCADO.

La Dinámica de Sistemas orienta el proceso de construcción de un modelo matemático estructural de un fenómeno y posibilita simular su comportamiento dinámico en el transcurrir del tiempo, o de otra variable independiente.

Evolución 3.5 es un software que permite modelar y simular con Dinámica de Sistemas, para su desarrollo se utilizaron técnicas de Programación Orientada a Objetos (POO), diseño basado en componentes, patrones de diseño y el Lenguaje Unificado de Modelado (UML), se utilizó el modelo en espiral como metodología de desarrollo.

[24] *ESMS:* Entorno Software de Modelado y Simulación.

Está conformado por un Editor que ofrece una amplia variedad de elementos y herramientas para el desarrollo de modelos y su documentación, un eficaz Motor que realiza los cálculos que se necesitan para la simulación y un Presentador de resultados que permite ver de diferente manera, los resultados de la simulación e interactuar con ésta, utilizando componentes como gráficas en dos y tres dimensiones, tablas, dial, barras de desplazamiento, etiquetas, etc. Este proyecto además del software produjo dos documentos: Manual del Usuario y Manual del Programador, que son de gran ayuda para sus lectores (usuarios de la herramienta y futuros desarrolladores, respectivamente); este proyecto en sí mismo constituye una experiencia de Ingeniería de Software de lata complejidad y hace parte de un desarrollo macro del grupo SIMON, para ofrecer a la comunidad nacional e internacional, un soporte para el modelado y la simulación de fenómenos complejos con Dinámica de Sistemas.

Esta tesis es tomada como referencia para el desarrollo de este proyecto, debido a que es la base de la herramienta software Evolución a la que se le va a realizar el mantenimiento.

5.4.3 TESIS DE PREGRADO “PROTOTIPO WEB PARA CONTROL DE VERSIONES SOFTWARE Y DOCUMENTACION EN LINEA, APLICADO AL SERVIDOR DE LA ESCUELA DE INGENIERIA DE SISTEMAS – UIS, CORMORAN”.

AUTORA: ANGELICA MARIA NIÑO DIAZ.

Este proyecto desarrollado para el servidor Cormorán de la Escuela de Ingeniería de Sistemas (EISI) de la Universidad Industrial de Santander, tiene como propósito administrar proyectos y documentación alojados en él. Permite centralizar proyectos e información en general, facilitar el desarrollo distribuido de aplicaciones y mantener la documentación del servidor actualizada y en línea.

Para llevar a cabo estas operaciones se utiliza la metodología de control de versiones, que es la administración de múltiples revisiones de una misma unidad de información. Es comúnmente utilizado en ingeniería y desarrollo de software para manejar el historial de documentos digitales tales como código fuente de aplicaciones y otra información crítica que puede ser trabajada en un grupo de personas.

Debido a que el mantenimiento de la herramienta software Evolución requiere de continuos cambios al código y la documentación, se hace necesario llevar un control de versiones con el propósito de administrar el acceso a un conjunto de ficheros y mantener un historial de los cambios realizados, es decir, facilitar dicho mantenimiento.

La tesis de pregrado hace referencia a diversas herramientas para el control de versiones software, con el objetivo de lograr el desarrollo de proyectos que administren de forma segura código, documentación y elementos multimedia.

Para escoger la herramienta indicada para el manejo de control de versiones, se tuvieron en cuenta diversos aspectos[25], y se llegó a la conclusión que la más apropiada es SUBVERSION de la organización Tigris.org, ya que cumple con la mayoría de requisitos. Por todo lo anterior, se tiene en cuenta esta tesis de pregrado para llevar a cabo el control de versiones de Evolución 4.0.

[25] Los aspectos evaluados y el documento completo se encuentra en la url: http://tangara.uis.edu.co/biblioweb/tesis/2008/125763.pdf

6 RESULTADOS

6.1 EVALUACIÓN ARQUITECTÓNICA DEL SOFTWARE EVOLUCIÓN 4.0

6.1.1 Aplicación de la Evaluación. La evaluación arquitectónica de EVOLUCIÓN 3.5 y 4.0 se constituyó a partir de tres valoraciones: automática, semi-automática y manual. La primera se llevo a cabo por medio de la herramienta SDMetrics, la segunda mediante la herramienta Ms-Excel y la tercera a través de EssModel. El procedimiento para realizar la valoración automática de EVOLUCIÓN 4.0 fue el siguiente:

1. Abrir el directorio "***DesarrolloFIS***[88]" mediante el comando "**Open Folder**" de la herramienta **EssModel**.

2. Generar el archivo en formato **XMI** a partir del código fuente de **EVOLUCION 4.0** mediante el comando "**Export Model to a XMI-File**" de la herramienta **EssModel**. El archivo generado tuvo un tamaño de 1.82 MB, arrojando un total de 57 clases y 44 paquetes.

3. Abrir el archivo **XMI** generado en el paso anterior y ejecutar el comando "***Calculate Metrics***" en la herramienta **SDMetrics**.

4. Copiar manualmente[89] cada fila de valores de métricas por cada clase del código fuente de **EVOLUCION 4.0**, en el archivo "Metricas_Evolución_4.0.xls" según corresponda.

Por otra parte, el procedimiento para valorar manualmente el estado de algunas métricas no contempladas por **SDMetrics** fue el siguiente:

1. Abrir el directorio "***DesarrolloFIS***" en el comando "**Open Folder**" de la herramienta **EssModel**.

[88] Carpeta que contiene el código fuente del componente FIS que se desarrolló para Evolución 4.0.

[89] El copiado manual de los datos se requirió debido a que la versión de SDMetrics fue una Demo.

2. Recorrer manualmente el árbol de paquetes o clases que provee **EssModel** con el propósito de buscar la información necesaria para calcular la métrica. Generalmente esta información se deriva a partir del conteo manual de algunos aspectos clave involucrados en la fórmula de la métrica.

3. Consignar el resultado en el archivo "Metricas_Evolución_4.0.xls" según corresponda.

Nota: Este mismo procedimiento se realizó para la evaluación arquitectónica de EVOLUCIÓN 3.5, utilizando el código fuente correspondiente a esta versión. Los archivos obtenidos contenían una gran cantidad de datos, debido al número de clases y paquetes de esta herramienta, los cuales están contenidos en la tesis de maestría de J. Moreno[90].

6.1.2 Consolidación e interpretación de resultados. Una vez finalizada la evaluación de las métricas del modelo propuesto en el numeral 6.3.4 usando toda la información disponible (numeral 6.3.3) y las herramientas descritas en el numeral 6.3.5, se alcanzó el diligenciamiento completo del archivo Metricas_Evolución_4.0.xls cuyo modelo de datos se especificó en el numeral 6.3.6. A continuación se describe la interpretación de datos relacionados con la evaluación arquitectónica de EVOLUCION 3.5 de manera parcial[91] y EVOLUCIÓN 4.0 en forma detalla.

[90] Tesis de maestría "Diseño de una Arquitectura para un Entorno de Modelamiento-Simulación y Creación de un Proceso para su Desarrollo por una Comunidad (I+D)". Ing. Jorge Jair Moreno Chaustre.

[91] Los resultados completos de la Evaluación Arquitectónica de Evolución 3.5 se encuentran en la tesis de maestría "DISEÑO DE UNA ARQUITECTURA PARA UN ENTORNO DE MODELAMIENTO - SIMULACIÓN Y CREACION DE UN PROCESO PARA SU DESARROLLO CON UNA COMUNIDAD (I+D)".

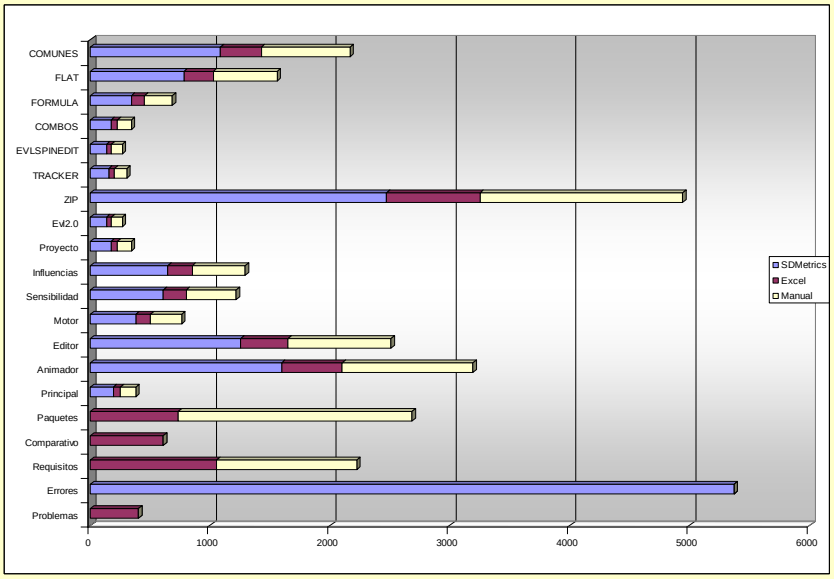
6.1.3 Ficha Técnica

Evolución 3.5	Evolución 4.0
En total se recopilaron 31.503 datos agrupados en 2.184 registros y distribuidos en 20 hojas de cálculo de Excel. SDMetrics calculó de forma automática 15.478 datos (49,13% del total), mientras que 5.987 datos (19% del total) fueron calculados por Ms-Excel de forma semi-automática. Por último, 10.038 datos (31,86% del total) se contabilizaron manualmente con la ayuda de la herramienta EssModel. El esfuerzo total en horas/hombre para la realización de la evaluación e interpretación de resultados, arrojó un valor de 287 horas/hombre. Esta información es útil a la hora de estimar futuros esfuerzos de evaluación arquitectónica y/o de justificar un desarrollo orientado hacia esta área de la ingeniería del software.	En total se recopilaron 3.486 datos agrupados en 256 registros y distribuidos en 5 hojas de cálculo de Excel. SDMetrics calculó de forma automática 1.836 datos (52.66% del total), mientras que 670 datos (19.21% del total) fueron calculados por Ms-Excel de forma semiautomática. Por último, 980 datos (28.11% del total) se contabilizaron manualmente con la ayuda de la herramienta EssModel.

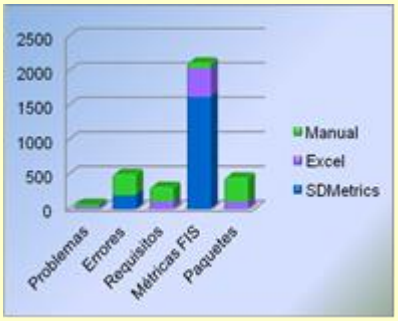
La Figura 9 ilustra la situación, a la luz de los aspectos y formas de registro de métricas para Evolución 3.5 y 4.0.

Figura 9. Cantidad de Datos Recopilados por Método y por Aspecto para EVOLUCIÓN 3.5 y 4.0*

EVOLUCIÓN 3.5



EVOLUCIÓN 4.0

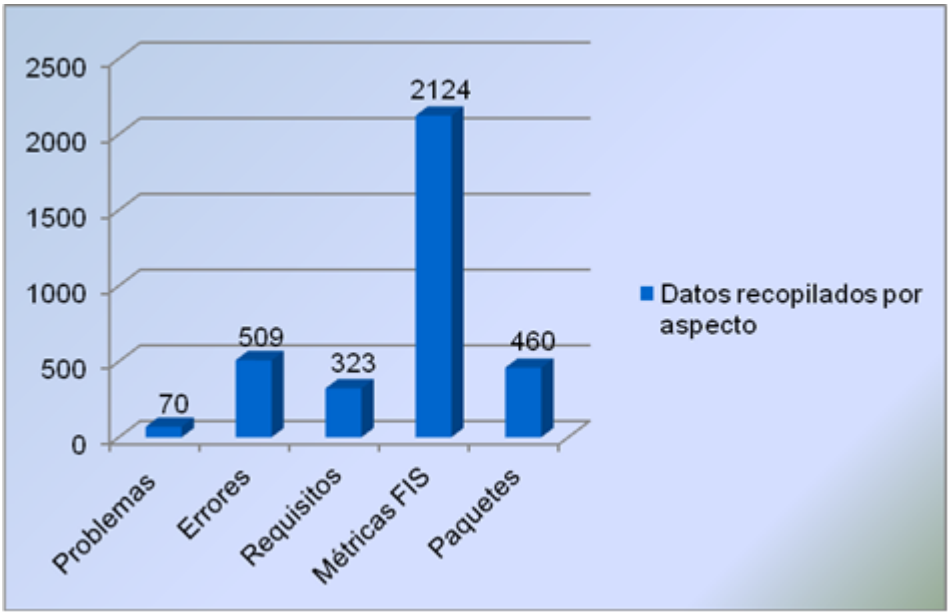


***Nota:** Los métodos utilizados para la recopilación de la información son: automático con la herramienta SDMetrics, semiautomático con Excel y manual con EssModel. Los aspectos hacen referencia a las diferentes carpetas contenidas dentro del código fuente de Evolución 3.5 y 4.0.

Por otra parte, en la Figura 10, se muestra la cantidad de datos registrado por cada uno de los aspectos lógicos[92] de la evaluación arquitectónica de **Evolución 4.0**, allí se puede notar que la mayor densidad de datos se encuentra concentrada en el aspecto “Métricas FIS”.

[92] Aspectos tenidos en cuenta para realizar la evaluación arquitectónica de EVOLUCIÓN 4.0.

Figura 10. Cantidad de Datos Recopilados por Aspecto



En la Tabla 5, se muestra la cantidad de datos recopilados (automática, semiautomática o manualmente) por cada uno de los 5 aspectos contemplados en la evaluación arquitectónica de EVOLUCION 4.0**.**

Tabla 5. Cantidad de Datos Recopilados

	METODO				
Aspecto	**Automático (SDMetrics)**	**Semiauto-mático (Ms-Excel)**	**Manual**	**Total de Datos por Aspecto**	**Proporción Densidad de Datos por Aspecto**
Problemas	0	35	35	**70**	**2.01%**
Errores	204	0	305	**509**	**14.60%**
Requisitos	0	113	210	**323**	**9.27%**
Métricas FIS	1632	414	78	**2124**	**60.93%**
Paquetes	0	108	352	**460**	**13.20%**

6.1.3.1 Organización del Informe de Evaluación Arquitectónica. El informe de evaluación arquitectónica para EVOLUCIÓN 4.0 se compone de dos archivos de Ms-Excel mutuamente relacionados Métricas_Evolución_4.0.xls y Gráficos_Evolución_4.0.xls**.** A continuación se describe el contenido del archivo

Métricas_Evolución_4.0.xls mediante una tabla que especifica el propósito, número de datos, registros, métricas y gráficos que corresponden a cada módulo de la herramienta EVOLUCIÓN 4.0. En cada caso se proveen enlaces a los directorios o documentos según corresponda.

Tabla 6. Descripción del Informe de Evaluación Arquitectónica

Hoja	Descripción
Problemas	El propósito fundamental de esta hoja consiste en enumerar y categorizar los problemas relacionados con las heurísticas (COMPLETITUD, CORRECTITUD, NOMBRADO y ESTILO) de diseño para modelos orientados a objetos y escritos en UML. Además muestra el estado final de los factores de Acoplamiento, Herencia y Encapsulamiento para el sistema. En el archivo Gráficos_Evolución_4.0.xls se ilustran estos datos mediante 4 gráficos.
	Número de Datos: 70 corresponde al 2.01% del total
	Número de Registros: 39
	Número de Datos Calculados Automáticamente: 0
	Número de Datos Calculados Semiautomáticamente: 35
	Número de Datos Calculados Manualmente: 35
Métricas FIS	El propósito fundamental de esta hoja consiste en mostrar el estado final de los grupos de métricas relacionadas con el acoplamiento, tamaño, herencia, encapsulamiento y complejidad para todos los módulos de la herramienta Evolución 4.0, permitiendo realizar una comparación del estado de cualquier módulo respecto a los demás. En el archivo Métricas_Evolucion_4.0.xls se muestran estos datos mediante tablas. Se calcularon 36 métricas.
	Número de Datos: 2124 corresponde al 60.93% del total
	Número de Registros: 59
	Número de Datos Calculados Automáticamente: 1632
	Número de Datos Calculados Semiautomáticamente: 414
	Número de Datos Calculados Manualmente: 78
Errores	El propósito fundamental de esta hoja consiste en enumerar para todos los módulos de la herramienta Evolución 4.0 las violaciones a las reglas de buen diseño o heurísticas para COMPLETITUD, CORRECTITUD, NOMBRADO y ESTILO. En el archivo Métricas_Evolucion_4.0.xls, se muestran estos datos mediante tablas.
	Número de Datos: 509 corresponde al 14.60% del total
	Número de Registros: 95
	Número de Datos Calculados Automáticamente: 204
	Número de Datos Calculados Semiautomáticamente: 0
	Número de Datos Calculados Manualmente: 305
	El propósito fundamental de esta hoja consiste en mostrar el estado final de los las

Hoja	Descripción
Requisitos	métricas relacionadas con los casos de uso que se proporcionaron en la documentación oficial de la herramienta Evolución 4.0, permitiendo valorar el estado del modelo de requisitos. En el archivo Métricas_Evolucion_4.0.xls, se muestran estos datos mediante tablas. Se calcularon 19 métricas.
	Número de Datos: 323 corresponde al 9.27% del total
	Número de Registros: 17
	Número de Datos Calculados Automáticamente: 0
	Número de Datos Calculados Semiautomáticamente: 113
	Número de Datos Calculados Manualmente: 210
Paquetes	El propósito fundamental de esta hoja consiste en mostrar el estado final de los grupos de métricas relacionadas con el acoplamiento, tamaño, herencia, encapsulamiento y complejidad para todos los paquetes de la herramienta Evolución 4.0, permitiendo realizar una comparación del estado de cualquier paquete respecto a los demás. En el archivo Métricas_Evolucion_4.0.xls, se muestran estos datos mediante tablas. Se calcularon 10 métricas.
	Número de Datos: 460 corresponde al 13.20% del total
	Número de Registros: 46
	Número de Datos Calculados Automáticamente: 0
	Número de Datos Calculados Semiautomáticamente: 108
	Número de Datos Calculados Manualmente: 352

6.1.4 Elaboración de un compendio de fallos arquitectónicos. Luego de exponer la estructura del informe de evaluación arquitectónica, se describe a continuación, un compendio de los errores o fallos (violaciones a las buenas prácticas de diseño propuestas en las heurísticas o reglas descritas en la Tabla 3. Atributos de Calidad Internos) encontrados al final del proceso de evaluación. Cada violación a la regla estará acompañada de una recomendación general para corregirla en el numeral 7.1.5. Además con el propósito de simplificar la lectura de este documento, se ha compilado el mismo compendio de fallos más extendido en la hoja “Compendio” del Archivo “Metricas_Evolucion_4.0.xls”. Los aspectos más destacados relacionados con violaciones a las reglas o heurísticas se enuncian a continuación:

A continuación se muestran los resultados obtenidos para las heurísticas de diseño para modelos orientados a objetos y escritos en UML, valorados en EVOLUCIÓN 3.5 y 4.0 como son: Completitud, Correctitud, Nombrado y Estilo.

Tabla 7. Resultados de la Heurística de Diseño “Completitud”

COMPLETITUD	
Evolución 3.5	**Evolución 4.0**
En total 4081 problemas encontrados.	En total 110 problemas encontrados.
a) Del 100% de los paquetes encontrados en el código fuente, ninguno fue representado en el modelo de la documentación oficial de Evolución 3.5 afectando la COMPLETITUD y CORRECTITUD del sistema en este aspecto. En total 239 de los 239 paquetes presentan esta violación a la regla. b) Además el 63% de las clases encontradas en el código fuente no están representadas en el modelo de UMLStudio que entregó SIMON. En total 282 de las 448 clases presentan esta violación a la regla. c) Lo anterior se agrava al determinar que sólo el 15% de la documentación UML (casos de uso y diagramas de secuencia) del sistema está completa. De lo que apenas existe se encontraron 158 problemas de especificación y completitud para casos de uso. Además el modelo de interacción está incompleto en al menos un 88% arrojando 3176 problemas de completitud. d) El 50% de las clases en el código, aparentemente no se usan. Esto se debe a que en estos casos, las clases son usadas como tipos de parámetros o variables en procedimientos de otras. En total 226 de las 448 clases presentan esta violación aparente a la regla. e) En la distribución de violaciones a la COMPLETITUD discriminada por módulos, puede observarse una marcada carencia de COMPLETITUD en los módulos: animador, editor, influencias, motor y comunes. Así mismo los componentes que más aportan a la degradación de la COMPLETITUD, son los componentes ZIP y FLAT.	a) Del 100% de los paquetes encontrados en el código fuente, ninguno fue representado en el modelo de la documentación oficial de Evolución 4.0 afectando la COMPLETITUD y CORRECTITUD del sistema en este aspecto. En total 44 de los 44 paquetes presentan esta violación a la regla. b) Además el 30% de las clases encontradas en el código fuente no están representadas en el modelo de UMLStudio que entregó SIMON. En total 17 de las 57 clases presentan esta violación a la regla. c) Sólo el 89% de la documentación UML (casos de uso y diagramas de secuencia) del sistema está completa. De lo que apenas existe se encontraron 30 problemas de especificación y completitud para casos de uso. d) El 33% de las clases en el código, aparentemente no se usan. Esto se debe a que en estos casos, las clases son usadas como tipos de parámetros o variables en procedimientos de otras. En total 19 de las 57 clases presentan esta violación aparente a la regla.

Tabla 8. Estado de la COMPLETITUD Antes y Después de las Mejoras para EVOLUCIÓN 3.5

	COMPLETITUD											
	ANTES						DESPUES					
	Clases No Usadas	Clases No Representadas	Paquetes No Representados	Casos de Usos No Especificados	Diags. Interacción	Total Problemas	Clases No Usadas	Clases No Representadas	Paquetes No Representados	Casos de Usos No Especificados	Diags. Interacción	Total Problemas
PRINCIPAL	2	0	3	N.D	25	30	2	0	0	N.D	0	2
ANIMADOR	33	39	66	64	560	762	33	0	0	0	0	33
EDITOR	27	4	24	36	271	362	27	0	0	0	0	27
MOTOR	1	0	7	58	80	146	1	0	0	0	0	1
SENSIBILIDAD	21	26	22	N.D	208	277	21	0	0	N.D	0	21
INFLUENCIAS	12	34	13	N.D	272	331	12	0	0	N.D	0	12
PROYECTO	1	0	3	N.D	8	12	1	0	0	N.D	0	1
EVL2.0	1	0	1	N.D	8	10	1	0	0	N.D	0	1
ZIP	92	124	31	N.D	992	1239	92	0	0	N.D	0	92
TRACKER	1	1	1	N.D	8	11	1	0	0	N.D	0	1
EVLSpinEdit	0	1	1	N.D	8	10	0	0	0	N.D	0	0
COMBOS	1	3	1	N.D	24	29	1	0	0	N.D	0	1
FORMULA	7	12	5	N.D	80	104	7	0	0	N.D	0	7
FLAT	14	35	30	N.D	280	359	14	0	0	N.D	0	14
COMUNES	13	3	31	N.D	352	399	13	0	0	N.D	0	13
TOTALES	**226**	**282**	**239**	**158**	**3176**	**4081**	**226**	**0**	**0**	**0**	**0**	**226**

Figura 11. Distribución de Fallos de COMPLETITUD para EVOLUCIÓN 4.0

Tabla 9. Resultados de la Heurística de Diseño "Correctitud"

CORRECTITUD	
Evolución 3.5	**Evolución 4.0**
En total 25 problemas encontrados.	En total 11 problemas encontrados.
a) El 5% de las clases encontradas en el código fuente, presentan un nombre incorrecto en relación con los diagramas de clase provistos en la documentación oficial de Evolución 3.5. En total 23 de las 448 clases presentan esta violación a la regla. b) Mientras que el 0.45% del total de clases tiene operaciones duplicadas. En total 2 de las 448 clases presentan esta violación a la regla.	a) El 19% del total de clases tiene operaciones duplicadas. En total 11 de las 57 clases presentan esta violación a la regla. b) La distribución de violaciones a la CORRECTITUD se ilustra a continuación en la Tabla 14.

c) En la distribución de violaciones a la CORRECTITUD discriminada por módulos, puede observarse una marcada carencia de CORRECTITUD en los módulos: *principal, animador, editor, proyecto y comunes*, en estos módulos la ambigüedad nominal entre el código y el modelo es muy alta.	

Tabla 10. Estado de la CORRECTITUD antes y Después de las Mejoras para EVOLUCIÓN 4.0

	CORRECTITUD					
	ANTES			DESPUES		
	Operaciones Duplicadas	Nombre Incorrecto	Total Problemas	Operaciones Duplicadas	Nombre Incorrecto	Total Problemas
PRINCIPAL	0	1	1	0	0	0
ANIMADOR	0	6	6	0	0	0
EDITOR	0	13	13	0	0	0
MOTOR	2	0	2	2	0	2
SENSIBILIDAD	0	0	0	0	0	0
INFLUENCIAS	0	0	0	0	0	0
PROYECTO	0	1	1	0	0	0
EVL2.0	0	0	0	0	0	0
ZIP	0	0	0	0	0	0
TRACKER	0	0	0	0	0	0
EVLSpinEdit	0	0	0	0	0	0
COMBOS	0	0	0	0	0	0
FORMULA	0	0	0	0	0	0
FLAT	0	0	0	0	0	0
COMUNES	0	2	2	0	0	0
Totales	**2**	**23**	**25**	**2**	**0**	**2**

Figura 12. Distribución de Problemas de CORRECTITUD en EVOLUCIÓN 4.0

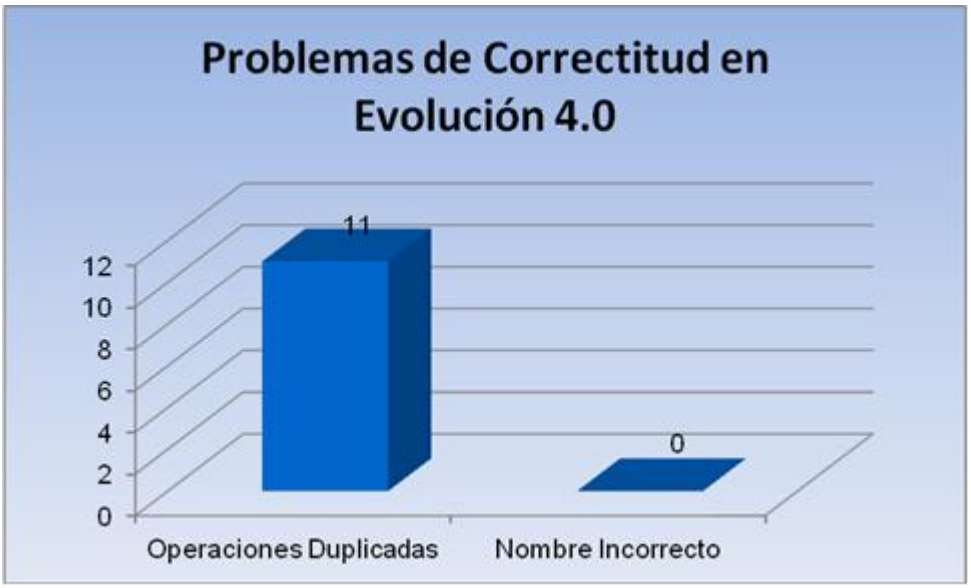


Tabla 11. Resultados de la Heurística de Diseño "Nombrado y Estilo"

NOMBRADO y ESTILO	
Evolución 3.5	**Evolución 4.0**
En total 87 problemas encontrados.	En total 4 problemas encontrados.
a) El 8% de las clases tiene entre atributos y operaciones más de 60, en otras palabras son clases enormes. En total 35 de las 448 clases presentan esta violación a la regla. Estas clases usualmente se convierten en cuellos de botella para el mantenimiento, son causa de problemas de confiabilidad y signo de una débil concepción en la arquitectura orientada a objetos. b) Mientras que el 2% de las clases padece problemas de ciclos de dependencia circular. En total 9 de las 448 clases presentan esta violación a la regla. Es posible encontrar serios inconvenientes a la horade probar y re-usar de forma independiente estas clases. El problema se agrava si las clases provienen de distintos paquetes.	a) El 7% de las clases tiene más de 60 atributos y operaciones, en otras palabras son clases enormes. En total 4 de las 57 clases presentan esta violación a la regla. Estas clases usualmente se convierten en cuellos de botella para el mantenimiento, son causa de problemas de confiabilidad y signo de una débil concepción en la arquitectura orientada a objetos. b) En la Tabla 15 se ilustra la distribución de problemas encontrado para la categoría de

c) El 10% de las clases del sistema, sobre-escribe sus atributos heredados. En total 43 de las 448 clases presentan esta violación a la regla. d) En la distribución de problemas encontrado para la categoría de NOMBRADO y ESTILO, discriminada por módulos, puede observarse una marcada concentración de problemas en el componente FLAT (mayor cantidad de clases enormes), seguido del animador, influencias (mayor cantidad de clases con sobre-escritura de atributos heredados) y el componente ZIP.	NOMBRADO y ESTILO.

Tabla 12. Estado de NOMBRADO y ESTILO Antes y Después de las Mejoras para EVOLUCIÓN 4.0

	NOMBRADO y ESTILO							
	ANTES				DESPUES			
	Clases Enormes	Clases con Referencias Circulares	Clases que Sobre-Escriben Atributos Heredados	Total Problemas	Clases Enormes	Clases con Referencias Circulares	Clases que Sobre-Escriben Atributos Heredados	Total Problemas
PRINCIPAL	1	0	0	1	1	0	0	1
ANIMADOR	8	0	8	16	8	0	8	16
EDITOR	3	0	1	4	3	0	1	4
MOTOR	0	2	1	3	0	2	1	3
SENSIBILIDAD	0	0	5	5	0	0	5	5
INFLUENCIAS	2	0	11	13	2	0	11	13
PROYECTO	2	0	0	2	2	0	0	2
EVL2.0	0	0	0	0	0	0	0	0
ZIP	3	0	7	10	3	0	7	10
TRACKER	0	0	0	0	0	0	0	0
EVLSpinEdit	0	0	0	0	0	0	0	0
COMBOS	0	0	0	0	0	0	0	0
FORMULA	0	2	2	4	0	2	2	4
FLAT	16	2	4	22	16	2	4	22
COMUNES	0	3	4	7	0	3	4	7
Totales	**35**	**9**	**43**	**87**	**35**	**9**	**43**	**87**

Figura 13. Distribución de Problemas de NOMBRADO y ESTILO en EVOLUCIÓN 4.0

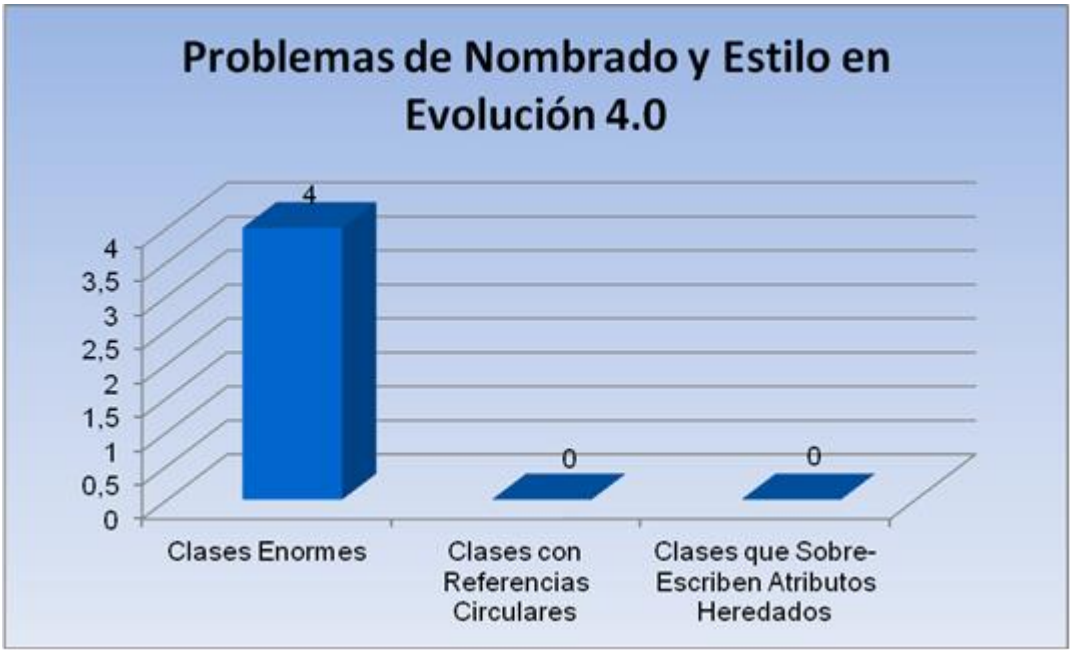


Tabla 13. Resultados del Factor Acoplamiento

ACOPLAMIENTO	
Evolución 3.5	**Evolución 4.0**
a) El factor de acoplamiento para las clases del sistema arrojó un valor del 2%. Es un acoplamiento bajo considerando que el máximo número de acoplamientos del sistema es TC^2-TC=200.256, donde TC=448 clases. Esto es posible al uso intensivo de componentes. Aún así, existen 3239 dependencias entre las clases del sistema, lo cual es complejo de entender si el modelo está desordenado, incompleto y poco especificado. b) En el mismo sentido, el factor de acoplamiento para paquetes del sistema arrojó un valor del 5%, considerado bajo dado que el número	a) El factor de acoplamiento para las clases del sistema arrojó un valor del 11.72%. Es un acoplamiento bajo considerando que el máximo número de acoplamientos del sistema es TC^2-TC=3.192, donde TC=57 clases. Esto es posible al uso intensivo de componentes. Aún así, existen 374 dependencias entre las clases del sistema, lo cual es complejo de entender si el modelo está desordenado, incompleto y poco especificado. b) En el mismo sentido, el factor de acoplamiento para paquetes del sistema arrojó un valor del 17.02%, considerado bajo dado que el número máximo de acoplamientos del sistema para los

máximo de acoplamientos del sistema para los paquetes es TP^2- TP=50.882, donde TP=239. Sin embargo este dato es engañoso debido a que en la mayoría de los casos, un paquete está ocupado por solo una clase. Aún así existen 2562 dependencias entre paquetes, lo cual se torna complejo de entender cuando el modelo está pobremente descrito o desordenado.	paquetes es TP^2-TP=1.892, donde TP=44. Sin embargo este dato es engañoso debido a que en la mayoría de los casos, un paquete está ocupado por solo una clase. Aún así existen 322 dependencias entre paquetes, lo cual se torna complejo de entender cuando el modelo está pobremente descrito o desordenado.

Tabla 14. Resultados del Factor Herencia

HERENCIA	
Evolución 3.5	**Evolución 4.0**
a) De los 8611 atributos totales de las clases del sistema, 3516 son atributos heredados, arrojando un valor de herencia en atributos del 41%, lo cual implica un uso intensivo de este mecanismo. b) Asimismo, de las 8891 operaciones totales de las clases del sistema, 4168 corresponden a operaciones heredadas es decir se utiliza este mecanismo un 47% de las veces.	a) De los 906 atributos totales de las clases del sistema, 235 son atributos heredados, arrojando un valor de herencia en atributos del 26%, lo cual implica un uso moderado de este mecanismo. b) Asimismo, de las 846 operaciones totales de las clases del sistema, 205 corresponden a operaciones heredadas es decir se utiliza este mecanismo un 24% de las veces.

Tabla 15. Resultados del Factor Encapsulamiento

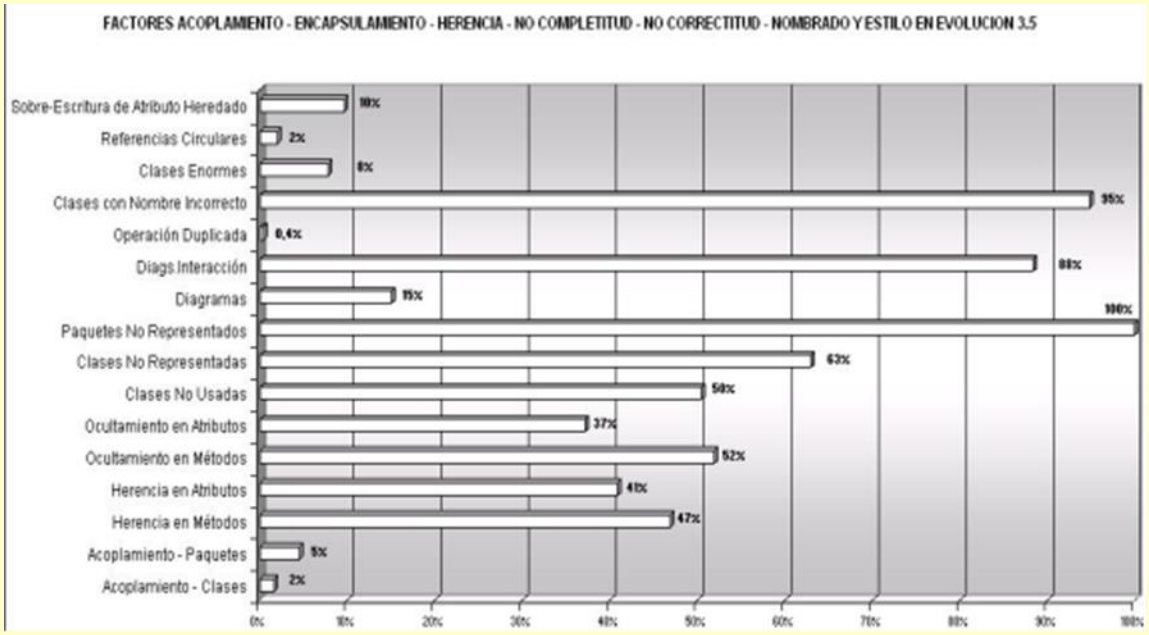
ENCAPSULAMIENTO	
Evolución 3.5	**Evolución 4.0**
a) De los 5095 atributos totales de las clases del sistema, 3203 son atributos públicos, es decir que sólo el 37% de los atributos del sistema se está ocultando.	a) De los 671 atributos totales de las clases del sistema, 459 son atributos públicos, es decir que sólo el 32% de los atributos del sistema se está ocultando.

b) Asimismo, de las 4723 operaciones totales de las clases del sistema, 2269 corresponden a operaciones públicas es decir se oculta el 52% de los métodos.	b) Asimismo, de las 641 operaciones totales de las clases del sistema, 391 corresponden a operaciones públicas es decir se culta el 39% de los métodos.

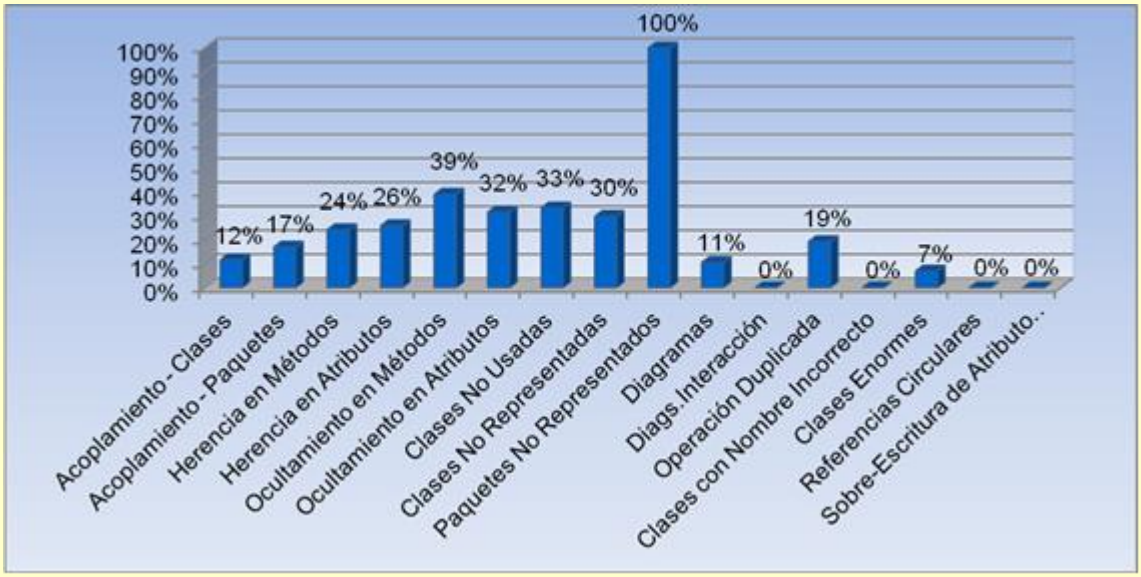
Por último, la Figura 14, ilustra el estado de los factores de Completitud, Correctitud, Nombrado, Estilo, Documentación, Acoplamiento, Herencia y Encapsulamiento para el sistema. Los datos mostrados son independientes unos de otros y deben tomarse en relación a su propia categoría.

Figura 14. Estado final de los Atributos de la Arquitectura de EVOLUCION 3.5 y 4.0

EVOLUCIÓN 3.5



EVOLUCION 4.0



6.1.5 Sugerencias para el mejoramiento arquitectónico. A continuación se describen los cambios que deben realizarse a la arquitectura de la herramienta Evolución 4.0 con el propósito de mejorar el estado de sus atributos en aras de alcanzar una arquitectura más organizada, fácil de entender, mantener y extender en el marco de una comunidad (i+d). Con el objeto de facilitar la lectura, se organizarán las sugerencias en el mismo orden de aparición y categorías que los del numeral anterior.

- **COMPLETITUD.** En total 110 problemas encontrados por resolver.

a) **Sugerencia 1.** Representar en el modelo de la documentación oficial de Evolución 4.0 el 100% de los paquetes encontrados en el código fuente. En total se resolverían 44 de los 44 problemas encontrados.

b) **Sugerencia 2.** Representar en el modelo que entregó SIMON el 30% de las clases encontradas en el código fuente que aún no están representadas. En total se resolverían 17 problemas.

c) **Sugerencia 3.** Completar el 11% de la documentación UML (casos de uso y diagramas de secuencia) del sistema, resolviendo al instante 30 problemas de especificación y completitud para estos artefactos.

d) **Conclusión.** Al realizar las mejoras sugeridas se habrán resuelto un total de 91 problemas de COMPLETITUD, mejorando notablemente la facilidad de comprensión de la arquitectura de Evolución 4.0, por parte de una comunidad de desarrollo. A continuación se muestra el conteo de problemas de COMPLETITUD antes y después de las mejoras:

Tabla 16. Problemas de COMPLETITUD antes y después de las mejoras

	COMPLETITUD											
	ANTES						DESPUES					
	Clases No Usadas	Clases No Representadas	Paquetes No Representados	Casos de Uso No Especificados	Diagramas de Interacción	TOTAL PROBLEMAS	Clases No Usadas	Clases No Representadas	Paquetes No Representados	Casos de Uso No Especificados	Diagramas de Interacción	TOTAL PROBLEMAS
Desarrollo FIS	19	17	44	30	0	**110**	19	0	0	0	0	**19**

- **CORRECTITUD.** En total 11 problemas encontrados por resolver.

a) **Sugerencia 1.** Debe revisarse las operaciones duplicadas de las clases afectadas y eliminar si es el caso, una de ellas. En total 11 errores se resolverían.

b) **Conclusión.** Al realizar las mejoras sugeridas se habrán resuelto un total de 11 problemas de CORRECTITUD, aportando facilidad de comprensión y facilidad de mantenimiento debido a la coherencia entre el modelo y el

código fuente. Sin embargo, los 11 problemas de operación duplicada no se resolverán, sólo se dejarán indicados. A continuación se muestra el conteo de problemas de CORRECTITUD antes y después de las mejoras:

Tabla 17. Problemas de CORRECTITUD antes y después de las mejoras

	CORRECTITUD					
	ANTES			DESPUES		
	Operaciones Duplicadas	Nombre Incorrecto	TOTAL PROBLEMAS	Operaciones Duplicadas	Nombre Incorrecto	TOTAL PROBLEMAS
Desarrollo FIS	11	0	**11**	11	0	**11**

- **NOMBRADO y ESTILO.** En total 4 problemas encontrados por resolver.

a) **Sugerencia 1.** En total 4 (7%) de las 57 clases son clases enormes. Debe considerarse su distribución en clases más pequeñas. Si no existe justificación para esta violación entonces se deben resolver estos 4 problemas.

b) **Conclusión.** Al realizar las mejoras sugeridas se resolverían 4 problemas de NOMBRADO y ESTILO. Sin embargo debe advertirse que emprender este esfuerzo es bastante delicado y propenso a errores, por lo tanto se sugiere realizar estos cambios en presencia de los desarrolladores de Evolución 4.0. En la siguiente tabla se especifica cómo quedó el estado de esta categoría de NOMBRADO y ESTILO en el contexto de la nueva arquitectura.

Tabla 18. Problemas de NOMBRADO Y ESTILO antes y después de las mejoras

	NOMBRADO Y ESTILO							
	ANTES				DESPUES			
	Clases Enormes	Clases con Referencias Circulares	Clases que Sobre-Escriben Atributos Heredados	TOTAL PROBLEMAS	Clases Enormes	Clases con Referencias Circulares	Clases que Sobre-Escriben Atributos Heredados	TOTAL PROBLEMAS
Desarrollo FIS	4	0	0	**4**	4	0	0	**4**

En caso de que el lector del presente documento se interese en alcanzar un mayor nivel de detalle acerca de los datos recaudados a partir de la evaluación arquitectónica de Evolución 4.0, se recomienda ver el Anexo “Métricas Evolución 4.0”.

Para la realización de la Evaluación Arquitectónica del software Evolución 3.5 y 4.0 se hizo necesario realizar una investigación exhaustiva sobre *“Análisis y Evaluación Arquitectónica”*, la cual, arrojó una metodología de diez (10) etapas y la armonización de un modelo de métricas para la arquitectura. Es importante destacar que mediante la metodología concebida, se valoró la calidad arquitectónica de una herramienta real, desarrollada desde hace más de veinte (20) años al interior del grupo de Investigación en Modelado y Simulación SIMON de la UIS. Dicha herramienta con un tamaño de más de 60.000 líneas de código fue valorada mediante un modelo de hasta 38 métricas propuestas

por diversos autores y agrupadas por atributos de *"tamaño", "herencia", "encapsulamiento" y "complejidad"*, también, se aplicaron 4 heurísticas de diseño arquitectónico para determinar problemas relacionados con la *"Completitud", "Correctitud", "Nombrado y Estilo"*.

- **Para Evolución 3.5:** Esta valoración implicó un esfuerzo enorme, arrojando un total de 31.503 datos dispuestos en 2.184 registros que se organizaron en 20 aspectos de la herramienta. Por otro lado, se encontraron hasta 4.407 problemas arquitectónicos de los cuáles 4.179 problemas se resuelven en la nueva arquitectura del ESMS-MI[93], quedando 228 problemas que requieren de la atención de los desarrolladores aún sin resolver. Más adelante y luego de cuatro (4) iteraciones, la arquitectura resultante con un total de: 6 Diagramas de Casos de Uso, 33 Diagramas de Paquetes, 239 Diagramas de Clases, 1 Diagrama de Componentes, 77 Casos de Uso, 448 clases y 245 paquetes, refleja el espíritu inicial de los desarrolladores de Evolución 3.5 organizando todos los aspectos de la herramienta de forma que el código fuente y el modelo se correspondan mutuamente sin distorsiones favoreciendo la completitud y correctitud de la arquitectura del ESMS-MI.

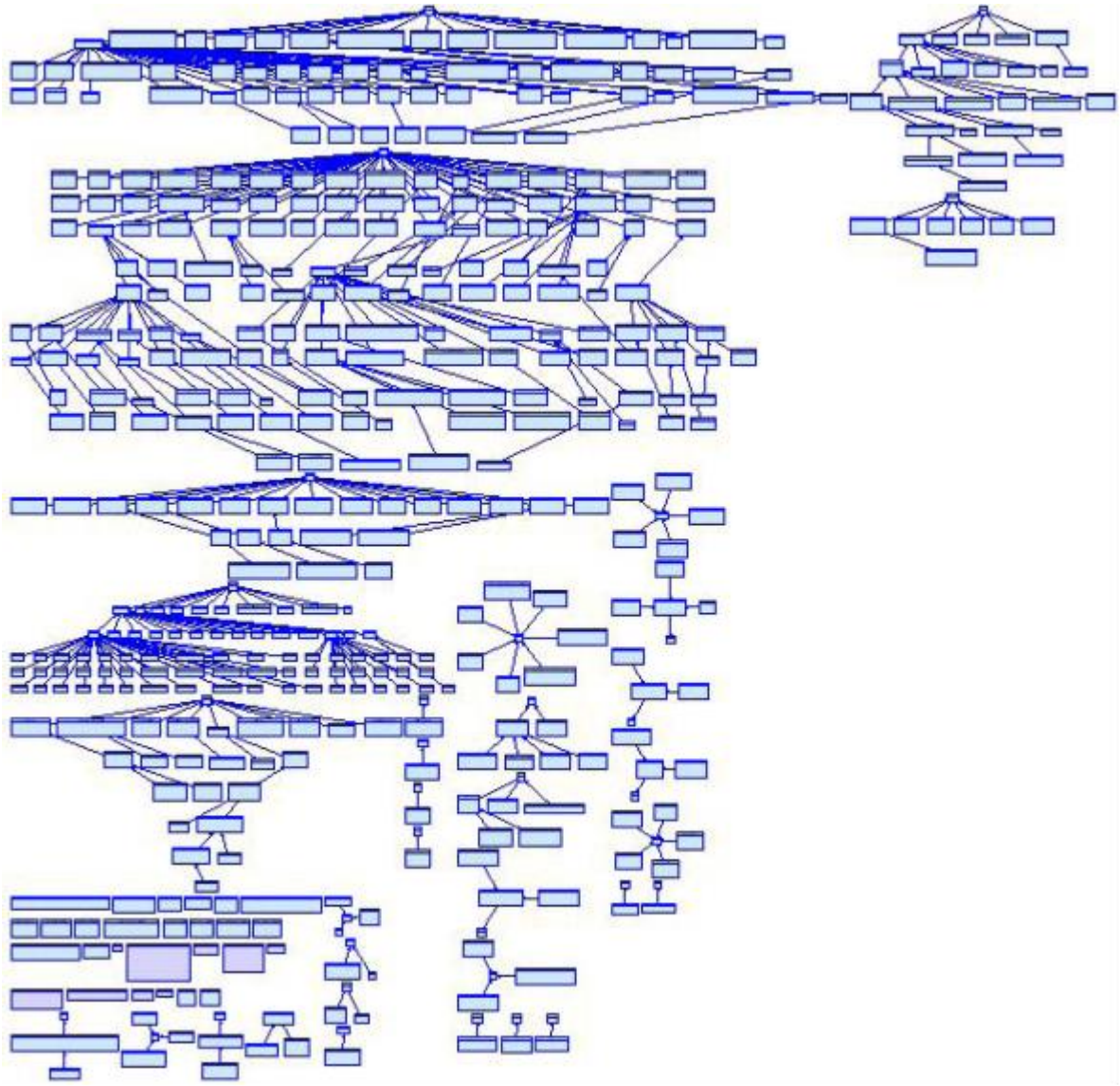
- **Para Evolución 4.0:** La valoración de esta versión obtuvo una menor cantidad de datos, arrojando un total de 3.486 datos dispuestos en 256 registros que se organizaron en 5 aspectos de la herramienta. Por otro lado, se encontraron 125 problemas arquitectónicos de los cuáles 91 problemas se resuelven en la nueva documentación del software, quedando 34 problemas que requieren de la atención de los desarrolladores aún sin resolver.

Finalmente, se hizo evidente la escasez de herramientas que permitieran realizar de forma automática la totalidad del análisis y evaluación arquitectónicos. Solamente se encontraron tres herramientas semi-adecuadas para tal propósito. Sin embargo se logró aprovechar la combinación estas herramientas mediante técnicas automáticas, semi-automáticas y manuales

[93] ESMS-MI: Entorno Software de Modelado y Simulación para Modelos Integrados.

para cubrir todos los aspectos deseados del análisis y evaluación arquitectónicos.

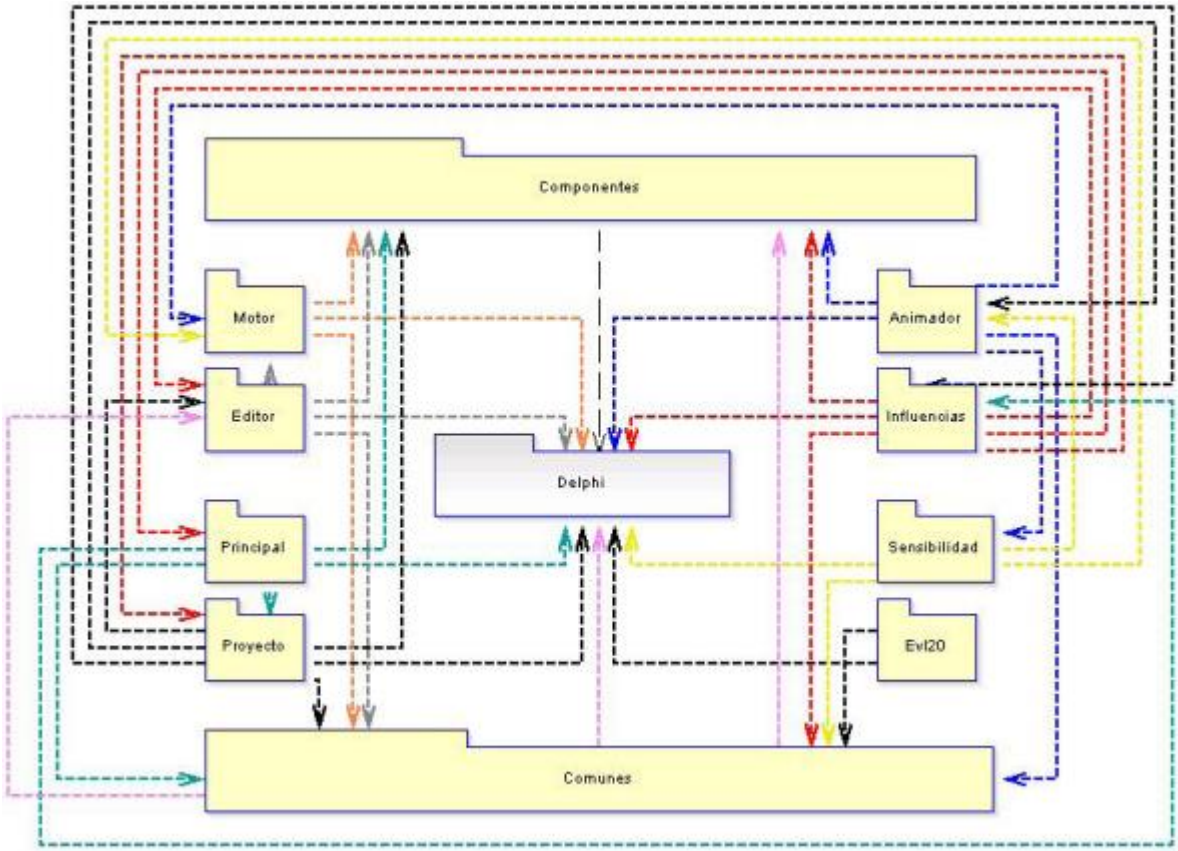
Figura 15. Arquitectura de Evolución 3.5 antes de la Evaluación Arquitectónica



Además el nuevo modelo arquitectónico propone en primer lugar, un conjunto de requisitos funcionales y no funcionales para extender la funcionalidad de Evolución 3.5 hacia el nuevo ESMS-MI, además de algunas mejoras para organizar aún más los elementos de software en el marco de una arquitectura

por capas la cual permitirá involucrar algunos patrones de diseño que se estarían violando en la antigua arquitectura.

Figura 16. Arquitectura de Evolución 3.5 después de la Evaluación Arquitectónica



6.2 EVALUACIÓN DEL SOFTWARE EVOLUCIÓN 4.0 SEGÚN EL ARTÍCULO "CRITERIA FOR SIMULATION SOFTWARE EVALUATION" JALAL NIKOUKARAN - VLATKA HLUPIC - RAY J. PAUL.

En el año de 1998 en la Conferencia de Simulación de Invierno[94] Jalal Nikoukaran, Vlatka Hlupic y Ray J. Paul publican una lista comprensiva de criterios estructurados en un marco de trabajo jerárquico para la evaluación de software de simulación. Los criterios están estructurados en siete grupos principales y varios subgrupos. La jerarquía puede ser utilizada para obtener una mejor visión de las características del software y como guía para poner a prueba y analizar los paquetes de modelado y simulación.

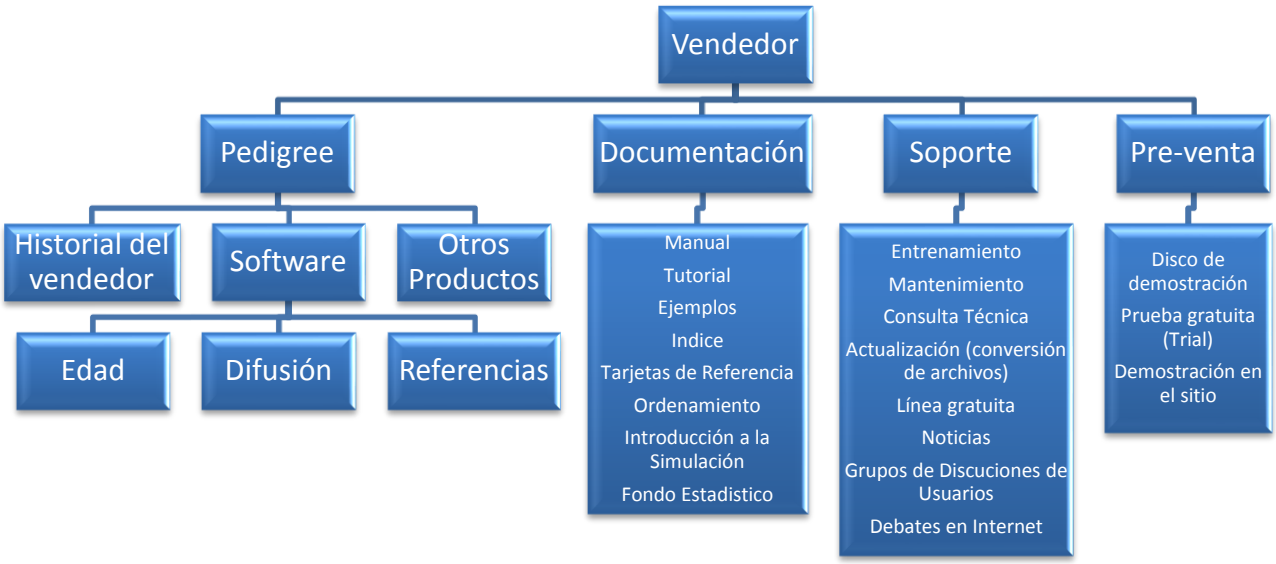
Figura 17. Grupos de Criterios Principales de la Jerarquía

Se evaluó el software Evolución 4.0 teniendo en cuenta los criterios que presenta este artículo, a fin de valorar las características de la herramienta como un software de modelado y simulación. A continuación se muestran los resultados:

[94] WSC: Winter Conference Simulation por sus siglas en inglés. La Conferencia de Simulación de Invierno es el principal foro internacional para la difusión de los avances recientes en el campo de la simulación de sistemas.

- **Vendedor**

Figura 18: Criterios Relacionados al Grupo Vendedor



➢ **Pedigree:** Se refiere a la historia del vendedor y del software, la relación que tiene con otros productos y la divulgación de la herramienta.

Historia del vendedor	Evolución inició con una propuesta del ingeniero Hugo Hernando Andrade Sosa profesor de la Universidad Industrial de Santander, fundador del grupo SIMON de Investigación en Modelamiento y Simulación adscrito a la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander. Este es un grupo de gran trayectoria enfatizado en la Dinámica de Sistemas, el Pensamiento Sistémico y otras áreas de la Educación.
Software	Evolución 4.0 es un software para el modelado y simulación de fenómenos complejos con Dinámica de Sistemas, para su desarrollo utiliza técnicas de Programación Orientada a Objetos, diseñado basado en componentes, patrones de diseño y el Lenguaje Unificado de Modelado.
Otros productos	Software de Modelado y Simulación: ° Evolución 2.0a y 3.5 ° Homos (Herramienta Software para el Modelamiento y Simulación basado en Objetos y Reglas). ° Simuis Software Educativo: ° MAC (Micromundos de Simulación para el Aprendizaje de las Ciencias de primero a undécimo grado).
Edad	Evolución 4.0 es un esfuerzo académico e investigativo que es realizado al interior del grupo de investigación SIMON desde hace más de 15 años. Se ha

	venido realizando a través de proyectos de pregrado por parte de los estudiantes pertenecientes al grupo de investigación.
Difusión	Evolución viene siendo utilizado en muchas universidades y colegios como un software académico de modelado y simulación tanto a nivel nacional e internacional. Ha sido difundido a través de ponencias en diferentes congresos latinoamericanos y encuentros colombianos de Dinámica de Sistemas. También vía web, a través de la página del grupo SIMON http://simon.uis.edu.co/joomla
Referencias	▪ *Universidad Industrial de Santander.* El grupo SIMON de investigación en Modelado y Simulación hace parte de los grupos de investigación de la UIS, la universidad pública del oriente colombiano de gran renombre y prestigio por su calidad académica. Se puede encontrar en la Internet a través de https://www.uis.edu.co

➢ **Documentación:** Es la información que presenta el software a los usuarios para su conocimiento y manejo tales como manuales, tutoriales, ejemplos, etc.

Evolución cuenta con manuales para el usuario y el programador que describen las características y funciones del software, desde la instalación, pasando por los componentes y los diferentes comandos de la herramienta. También presenta un videotutorial que explica el funcionamiento de la interfaz de Evolución, así como sus elementos y exponen un ejemplo práctico de población por prototipos. El software posee un sitio web propio para realizar descargas de las diferentes versiones, además el portal presenta una descripción de la herramienta, su historia, aplicaciones, ejemplos, manuales, preguntas frecuentes y foro. Toda esta documentación se encuentra en la web a través de la página del grupo SIMON http://simon.uis.edu.co/joomla.

➢ **Soporte:** Es el apoyo que tienen los usuarios.

El proyecto de grado Mantenimiento del Software Evolución 4.0 es el encargado del mantenimiento y actualización de la herramienta en su versión 4.0, con el propósito de corregir los errores que presente el software. Así mismo está disponible la página web y el foro para debatir temas relacionados con Evolución. De igual forma en el grupo SIMON de investigación se

adelantan proyectos de grado para la generación de nuevas versiones del producto con más funcionalidades.

- **Pre-venta**

Evolución es una herramienta software para uso académico exclusivamente, es producido por la Universidad Industrial de Santander que posee todos los derechos, por lo tanto no es un software comercial y se puede descargar gratuitamente desde la página del grupo SIMON, en la pestaña descargas. Es así como no existen pruebas gratuitas (trial) del producto, ya que se descarga la versión completa del software, pero si una demostración de su funcionamiento a través de los videotutoriales.

- **Software**

- **Modelo y entrada**

Para la construcción del modelo con Evolución, el usuario cuenta con medios como el ratón y el teclado. El modelo puede ser realizado de forma gráfica y definido a través de ecuaciones en el Diagrama de Flujo Nivel. Se representa en un graficador que simula el comportamiento del modelo. Evolución presenta menús con comandos para realizar el modelo y cajas de diálogo que guían al usuario a realizar algún procedimiento. También se muestra un botón de ayuda que provee de asistencia al usuario para realizar el modelo.

Se puede diseñar el modelo en un Diagrama de Influencias o en un Diagrama de Flujo Nivel, pero éste último es el que se representa en la simulación. La unión de modelos realizados previamente puede ser un submodelo para un gran modelo, Evolución cuenta con este elemento que puede ser de gran utilidad. También tiene una librería de funciones estándar (aleatorias, de conversión, especiales, lógicas, matemáticas, trigonométricas y misceláneos), propias de Evolución y funciones definidas por el usuario, lo cual permite que cada usuario cree su propia biblioteca de funciones para que haga más cómoda la creación de modelos. Esto es posible a través de una Biblioteca de

Vínculos Dinámicos (DLL), implementadas en cualquier lenguaje de programación de alto nivel como C++, Visual C++, Visual Basic, Delphi, etc.

El rechazo de entradas ilegales o no permitidas previene muchos errores que pueden ocurrir durante la ejecución del modelo, esto se muestra a través de mensajes de error cuando el usuario realiza algún procedimiento indebido o declara de forma incorrecta alguna variable.

- **Ejecución**

Evolución requiere que determinemos los valores iniciales de las variables antes de iniciar la corrida de simulación. El control de la velocidad de la ejecución del modelo es una característica que posee Evolución, lo cual puede reducir el tiempo de ejecución del modelo, pues puede hacerlo paso a paso para verificar el comportamiento de las variables simuladas.

- **Animación**

Evolución le permite al usuario crear animadores con objetos de animación que permiten visualizar los resultados de una simulación de diferente manera, inclusive algunos de estos controles permiten controlar e interactuar con la simulación y se pueden modificar sus propiedades.

La presentación gráfica de los modelos en la pantalla puede hacerse en dos o tres dimensiones, con diversas propiedades de tamaño, color, estilo, forma. Se pueden crear otras vistas para representar las variables.

Evolución permite iniciar, reanudar, pausar, parar y ejecutar paso a paso la simulación del modelo; tiene la posibilidad de hacer zoom, rotar y cambiar el estilo de los puntos de corte en la gráfica.

- **Prueba y eficiencia**

La función paso de simulación permite al usuario ejecutar el modelo paso a paso y observar los cambios en cada estado. Las puntos de quiebre en

Evolución se puede utilizar para determinar algunos puntos en el tiempo para detener o iniciar la simulación del modelo.

La longitud de los nombres de las variables es de 16 caracteres y no permite declararlos con tildes, sólo letras y números.

Se pueden ver los valores de las variables en la barra de estado y sobre la variable misma. También en la gráfica se pueden observar los valores que toma a medida que corre la simulación.

- **Salida**

El software Evolución permite al usuario producir informes personalizados. Estos informes pueden ser administrados para su presentación. Puede enviar las salidas a un archivo .mev que es la extensión para modelos creados con Evolución y a dispositivos como la impresora, además se pueden ver las salidas en otro tipo de aplicaciones software como los MAC, que usan componentes de Evolución para simulaciones. Se pueden exportar como texto las variables declaradas, su definición y descripción en el Diagrama de Influencias o el Diagrama de Flujo Nivel, de igual forma las imágenes generadas en éstos diagramas.

Los resultados de la simulación con Evolución pueden ser presentados en forma de gráficos así como en forma de tabla. Estos gráficos pueden ser visualizados en la pantalla, cambiar dinámicamente con el progreso de la corrida de simulación del modelo.

El análisis de la salida es una cuestión importante en Evolución. El software lo hace de dos maneras, un Análisis de Sensibilidad por Variación de Parámetros y uno por Variación de Escenarios con el que se pueden comparar los resultados para diferentes valores de las variables. Se muestran estadísticas de las variables usadas en el análisis de sensibilidad como la media aritmética, desviación estándar y varianza.

- **Usuario**

Figura 19: Criterios Categorizados en el Grupo Usuario

El software Evolución es portable ya que el usuario puede elaborar un modelo en una máquina y ejecutarlo en otra máquina con una configuración diferente. El software puede ser ejecutado en versiones Windows 95 en adelante. Es necesario tener conocimientos previos en la Dinámica de Sistemas para poder desarrollar los modelos de forma correcta. Evolución es un software gratuito para uso académico exclusivamente.

Como conclusión vemos que Evolución cuenta con muchas características que debe tener un software de simulación, como se menciona en los criterios de este artículo[95] desde los tres puntos de vista: el vendedor, el software y el usuario.

6.3 MANTENIMIENTO DE SOFTWARE SEGÚN ESTÁNDAR IEEE 1219 - 1998

6.3.1 Identificación, clasificación y priorización del problema. Para el caso de Evolución se agrupó en una tabla todas las modificaciones que se realizaron al software teniendo en cuenta los siguientes aspectos:

- Asignación de un Número de Identificación.
- Clasificación del tipo de mantenimiento.

[95] CRITERIA FOR SIMULATION SOFTWARE EVALUATION. Jalal Nikoukaran, Vlatka Hlupic, Ray J. Paul.

- Análisis de la modificación para determinar si se acepta, se deniega o se evalúa.
- Realizar una estimación preliminar de la magnitud de la modificación.
- Priorizar la modificación.
- Asignar Solicitudes de Modificación a bloques de tareas planificadas para su implementación.

6.3.2 Análisis. En este análisis se tuvo en cuenta los resultados obtenidos en la evaluación arquitectónica y la ingeniería inversa hecha al código fuente de Evolución 4.0, que mediante la aplicación de un modelo de métricas sobre el software, permitió conocer su estructura interna, el estado de sus atributos de calidad, los posibles problemas de diseño subyacentes y finalmente las directrices para su eventual mejora, así como sus componentes y las interrelaciones que existen entre ellos.

6.3.2.1 Análisis de Viabilidad. Analizadas todas las modificaciones, se decidió que era viable la corrección de los problemas que presentaba el software Evolución, puesto que se cuenta con las herramientas necesarias tanto a nivel humano, de hardware y software para corregir los errores y de esta manera presentar un software funcional para el usuario final.

También es viable la realización de nuevas modificaciones, que no afectan el funcionamiento de Evolución y en cambio dan información acerca de las variables del modelo que se esté realizando.

6.3.2.2 Análisis Detallado. Los elementos afectados con el mantenimiento de la herramienta son el software y la documentación de Evolución. En el software se realizarán cambios concernientes a los elementos y sus propiedades, principalmente del Diagrama de Influencias, así como a controles que no están funcionando correctamente.

- **Lista de Modificaciones Preliminares**

La lista de modificaciones es la tabla de modificaciones realizadas al software Evolución 4.0, donde se presentan los cambios a realizarse en el Editor de Diagrama de Influencias, en el Editor de Diagrama de Flujo Nivel, en el Presentador de Resultados y en el Animador.

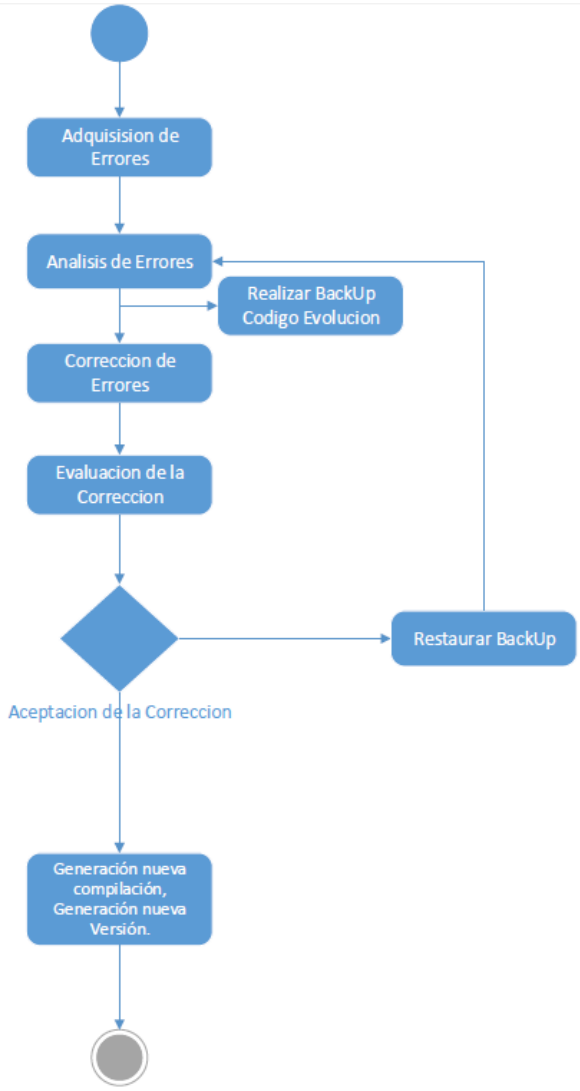
- **Plan de Pruebas**

Se diseñó un plan de pruebas, que reúne toda la información necesaria para planear y controlar el esfuerzo de probar el software Evolución. Este plan especifica los objetivos del mismo, así como las herramientas utilizadas para la elaboración de pruebas. El Plan de Pruebas se encuentra en el Anexo_3_Plan_de_Pruebas.

- **Plan de Implementación**

Se realiza una evaluación arquitectónica, basada en métricas de calidad, para verificar el estado del software. Se irán corrigiendo los errores y se van liberando nuevas versiones de Evolución con los errores ya corregidos. Enseguida se llevan a cabo las pruebas verificando que no aparezcan nuevos errores, luego se documenta el software y se libera la última versión.

Figura 20. Metodología utilizada en la Corrección de Errores de Código



- **Adquisición de Errores**: Los errores fueron recopilados de usuarios expertos, de las pruebas de aceptación de la retroalimentación de errores antes corregidos, de las pruebas de aceptación y de las pruebas de unidad.
- **Análisis:** En este proceso se lleva acabo el estudio del error, que lo produce y como se propone como se puede dar solución al mismo.
- **Realizar BackUp Código Evolución:** Al terminar el análisis del error y tener una posible solución se realiza un copia de seguridad del código.
- **Corrección de Errores:** Se hace la codificación de la solución propuesta en el procedimiento de análisis.

- **Evaluación de la Corrección:** Se realizan pruebas de la codificación de la solución del error en tiempo de ejecución si esta arroja los resultados esperados es aprobada la corrección.
- **Restaurar BackUp:** Este proceso se realiza si y solamente si la evaluación de la corrección no es aprobada.
- **Generación nueva compilación, generación nueva versión:** Al aprobar la evaluación de la corrección se realiza una nueva compilación del software.

6.3.3 Diseño. El primer paso en la fase de diseño será identificar los módulos software que van a ser objeto de modificación, estos componentes son: el Editor de Diagramas de Influencias, el Editor de Diagramas de Flujo Nivel, el Presentador de Resultados y el Animador. Estos cambios están representados en detalle en la Tabla de Errores de Evolución que se encuentra en los anexos. Luego de realizar las modificaciones, se generarán unos casos de pruebas, para verificar el buen funcionamiento del software. Además hay que identificar e incluir las pruebas de regresión necesarias.

6.3.4 Implementación. En este apartado se realizaron las pruebas de unidad y de integración.

6.3.4.1 Pruebas de Unidad.

Herramienta seleccionada. En el mercado existen diversas herramientas para realizar pruebas de unidad, como son Test, JUnit, Rational Robot, entre otras, las cuales trabajan con plataformas como Java o Microsoft Visual Studio .NET. También está presente ***DUnit***, una herramienta para la realización de pruebas de unidad de software desarrollado con Delphi. Es una adaptación para Delphi de una herramienta para Java llamada JUnit, desarrollada por Kent Beck. Todas las herramientas descendientes de JUnit consisten en una serie de clases que auxilian en la preparación y codificación de casos de prueba y algunos mecanismos auxiliares, que en conjunto permiten ejecutar y verificar el

cumplimiento de los casos de prueba. Además provee una interfaz que permite automatizar la ejecución de grupos de casos de prueba. En el caso de DUnit dicha interfaz se puede ejecutar desde Delphi o por separado. Además es una herramienta de software libre, por lo cual se puede extender. En internet se puede obtener gratuitamente desde el sitio http://sourceforge.net/projects/dunit.

Por todas estas razones, se escogió esta herramienta por ser la que mejor se adapta para realizar las pruebas de unidad al software Evolución, software desarrollado en Delphi 7.

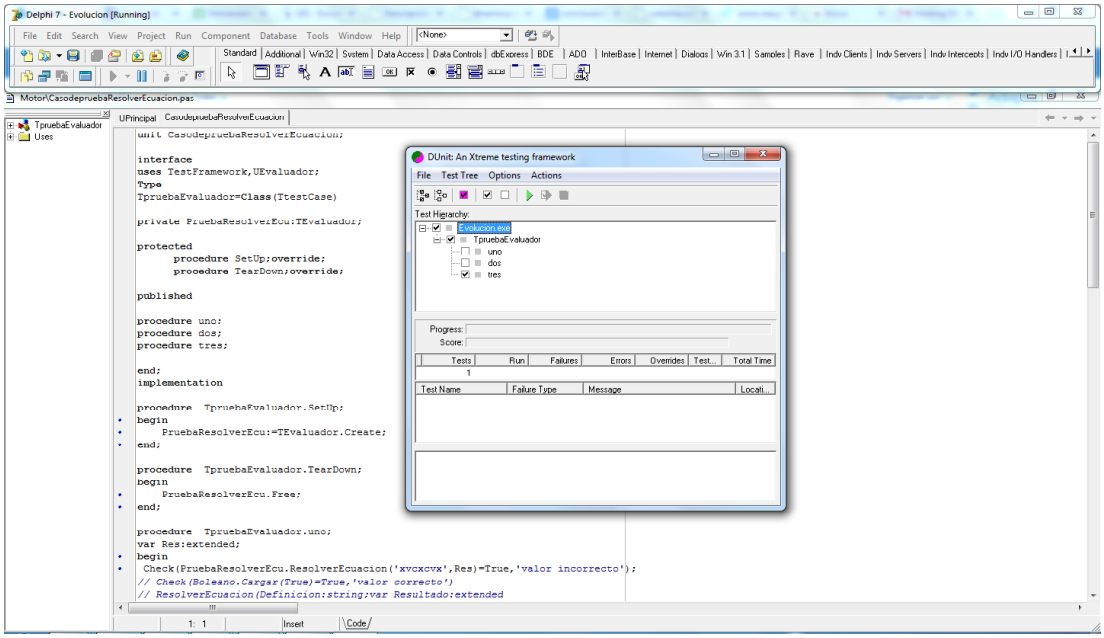
Metodología usada para la realización de las Pruebas de Unidad.

Basándose en el Manual del programador y el diagrama de clases de la herramienta se escogieron las clases fundamentales. Las clases, sus métodos y sus respectivos casos de prueba.

Cabe aclarar que no se aislaron las unidades del código de Evolución 4.5 para llevarlas a un proyecto aparte del FrameWork de DUnit como sugería la documentación. Después de una larga discusión entre los ejecutantes de este proyecto y sus directores se decidió utilizar el Framework DUnit al interior de todo el código de Evolución 4.5 a fin de que todos los componentes y recursos del software trabajen en pos del método que se va a probar.

Esto se hizo de la siguiente forma: se generó una versión paralela de Evolución 4.5 y se incluyó la unidad Casos de prueba para cada método escogido, en la cual van los procedimientos que contienen los casos de prueba. Además en el apartado *uses* de la unidad Evolución se añadieron las clases *TestFramework y GUITestRunner*, para que se pueda ejecutar la suite de Dunit y la clase *Casodeprueba in 'Casodeprueba.pas'*; para que se utilice la clase prueba; en el apartado *begin* de este mismo se agregó *GUITestRunner.RunRegisteredTests* y se deshabilitó la línea *Application.Run.*

Figura 21. Interfaz de DUnit.



6.3.5 Pruebas del Sistema.

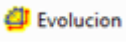
6.3.5.1 Pruebas de Rendimiento

Herramienta Seleccionada. Hoy en día es muy común encontrar infinidad de herramientas para realizar pruebas de rendimiento como son: HP Load Runner, JMeter, Selenium, inTrace, ProDelphi, entre muchas mas, las cuales permiten realizar pruebas a una gran variedad de aplicaciones, incluyendo las de movilidad, Ajax, Flex, HTML 5, Web, .NET, Java, GWT, Silverlight, SOAP, Citrix, ERP y las heredadas. Por estar diseñadas para otro tipo de aplicaciones diferentes a Evolución (Aplicación hecha en Delphi), por su elevado costo o porque solo evalúan unas cuantas rutinas y no el conglomerado del software, se descartan para realizar las pruebas de rendimiento a Evolución 4.5.

Una que si supera las expectativas es AQTime (Version 8.10 Built 772) una completa herramienta de la compañía SmartBear Software diseñada para evaluar el rendimiento de aplicaciones y eliminar fugas y cuellos de botella.

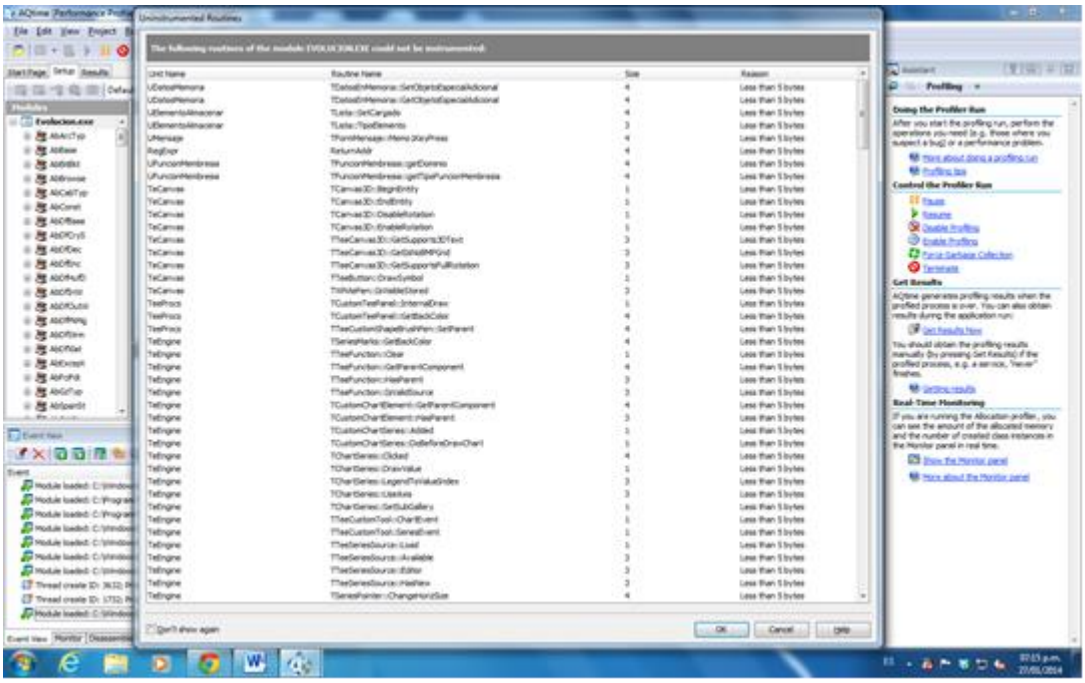
Es una herramienta de rendimiento de perfiles y depurador de uso de la memoria para aplicaciones Windows y compiladores .NET, (Delphi, Visual Basic, Visual C++ y C++ Builder) y de las rutinas de VBScript y JScript de 32 - y 64-bits. AQTime puede perfilar casi cualquier tipo de ejecutable (exe, dll, ocx, bpl, cpl, servicios NT, ISAPI y aplicaciones ASP.NET, COM, DCOM y servidores COM+). Además, puede perfilar .NET y ciertos tipos de secuencias de comandos. Puede perfilar plantillas, rutinas sobrecargadas y recursivas, y otras formas especiales de código. Por estas razones se escogió dicha herramienta para realizar las pruebas de rendimiento a Evolución 4.5, además es muy completa y facilita el entendimiento de los resultados gracias a los manuales, tutoriales y ayuda que posee.

Metodología usada para la realización de las Pruebas de Rendimiento.

Para realizar las pruebas de rendimiento sobre Evolución 4.5, en AQTime se debe crear un nuevo proyecto, agregar un nuevo módulo que será el archivo ejecutable de Evolución 

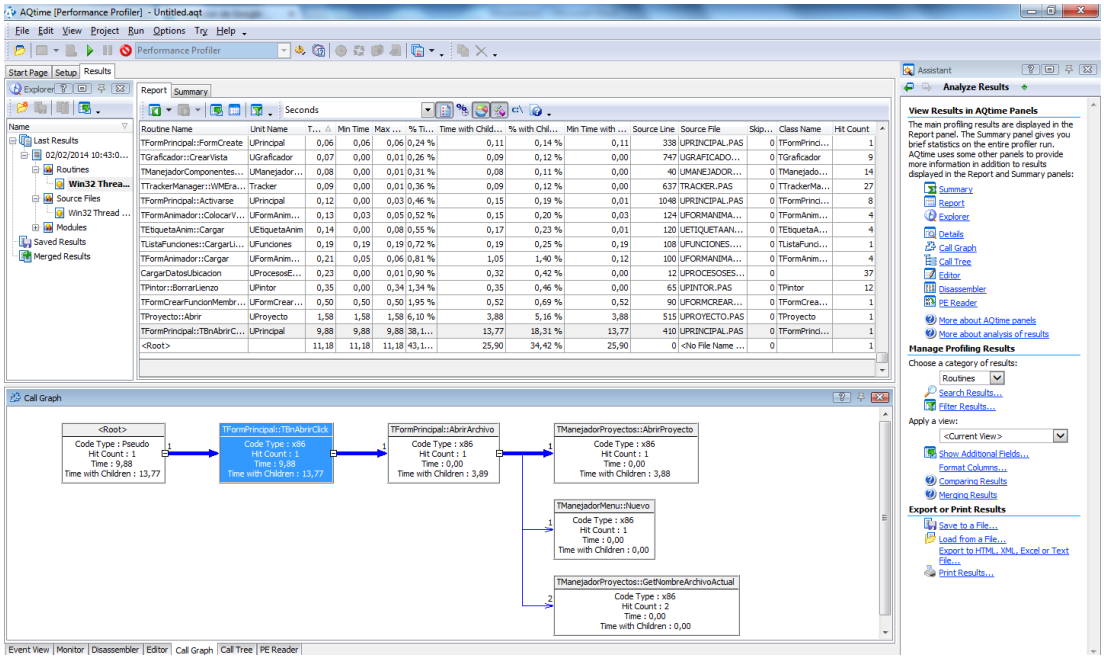
(.exe), escoger las opciones de perfilado y darle correr (F5.) La herramienta de pruebas primero que todo realiza un proceso llamado instrumentación, en el cual solo perfila aquellas rutinas que pueden ser medibles, las que no cumplen ciertos requisitos definidos por AQTime se muestran en un cuadro de diálogo como aparecen en la Figura 22. Para el caso de Evolución 4.5, la mayoría de éstas rutinas no pueden ser instrumentadas por ocupar menos de 5 bytes de código binario.

Figura 22. Rutinas de EVOLUCION.EXE que no pueden ser instrumentadas con AQTime.



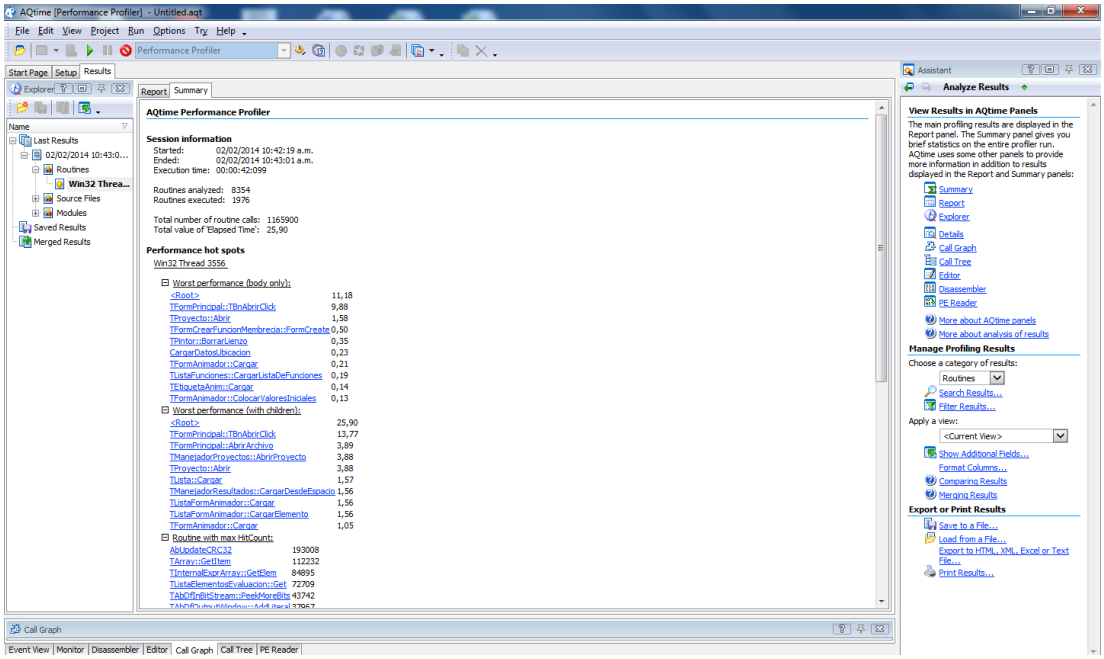
Enseguida se inicia el software Evolución 4.5, desde allí se pueden elaborar modelos con Dinámica de Sistemas y los resultados de la prueba de rendimiento se generan en varios paneles, mostrando las unidades, clases y rutinas utilizadas desde que se carga la aplicación hasta que se crea el modelo y los tiempos gastados en ejecutar estas rutinas. Dichos resultados se presentan en forma de tablas, código ensamblador, árbol de llamadas, gráficos de llamadas, entre otros.

Figura 23. Resultados con AQTime.



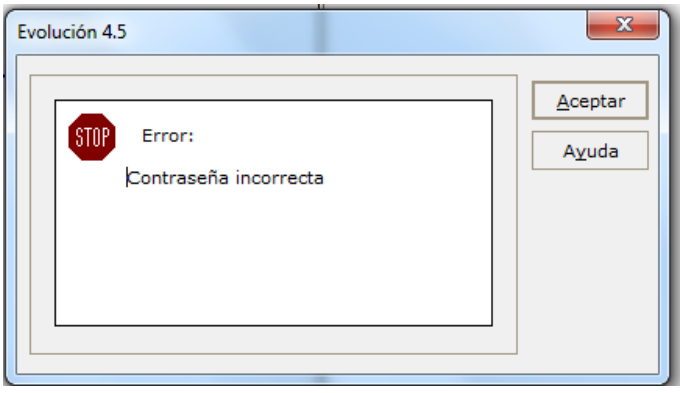
AQTime genera un resumen al finalizar la prueba, donde detalla la Información de la sesión, los puntos calientes de rendimiento (cuellos de botella), la configuración de ejecución, las opciones del perfilador y la información del sistema. En el Anexo_5_Resultados_Pruebas_de_Rendimiento se encuentran los resultados obtenidos con la realización de un modelo en Evolución, para un mejor entendimiento de la herramienta, donde se muestran las rutinas, clases utilizadas y los tiempos gastados por cada una de ellas.

Figura 24. Resumen del Análisis de Rendimiento.



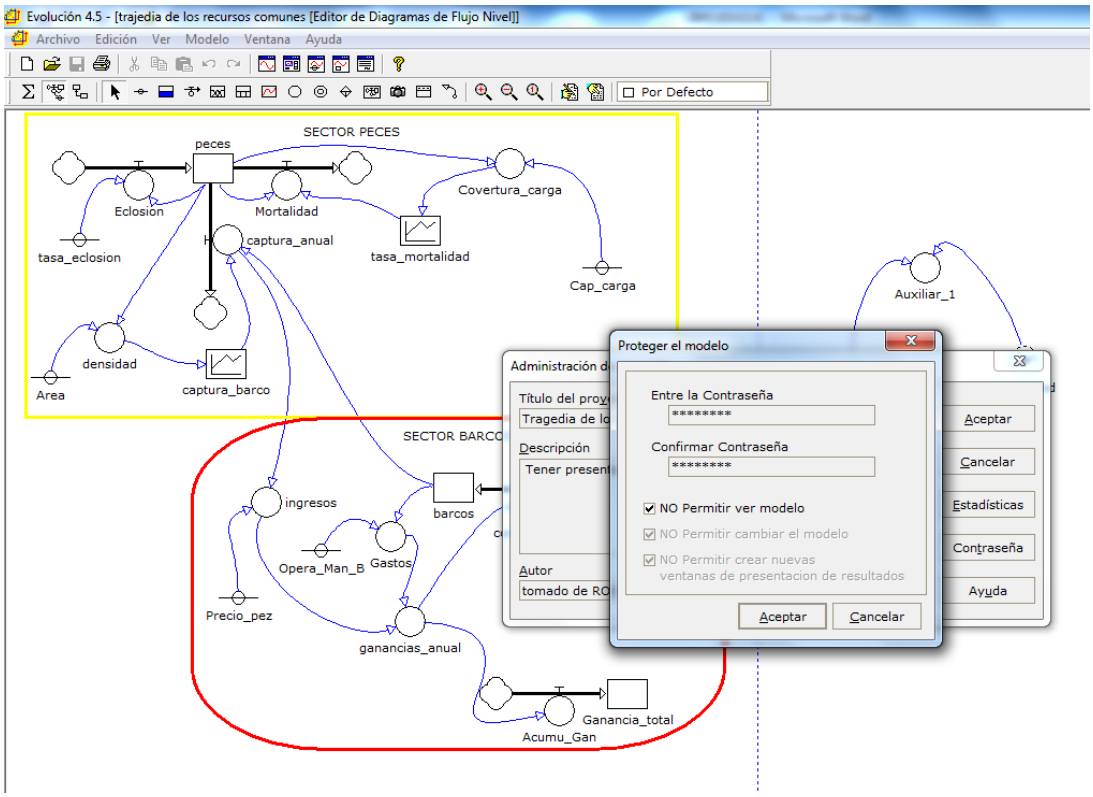
6.3.5.2 Pruebas de seguridad. Evolución 4.5 cuenta con la posibilidad de proteger el modelo con la creación de una contraseña de máximo 8 caracteres. Se debe ingresar por el menú “Archivo”, luego “Administración” y se abre un cuadro de diálogo, que le permite ingresar la contraseña. El usuario puede elegir entre “No permitir ver el modelo”, “No permitir cambiar el modelo” y “No permitir crear nuevas ventanas de presentación de resultados”. Si la contraseña es errónea la aplicación muestra un mensaje de error como el que aparece en la figura 25.

Figura 25. Mensaje de error contraseña incorrecta



Si el usuario malicioso intenta ingresar una contraseña errónea para abrir un modelo en 3 ocasiones seguidas, el software se sale del cuadro de diálogo “Contraseña de desbloqueo” y cierra los editores de Diagrama de Flujo Nivel y Diagrama de Influencias. Estas opciones fueron probadas con varios modelos cumpliendo a satisfacción cada una de ellas.

Figura 26 Pruebas de Seguridad con Evolución 4.5



6.3.5.3 Pruebas de recuperación. Evolución 4.5 no cuenta con una forma de recuperar los datos cuando el modelo no ha sido guardado, esto se comprobó cerrando de manera abrupta el sistema, después de haber creado un modelo sin haberlo guardado. Al reiniciar el sistema y volver a abrir Evolución 4.5 el modelo creado anteriormente no apareció.

6.3.6 Pruebas de Aceptación. Estas pruebas se desarrollaron con grupos de estudiantes de varias Universidades de Bucaramanga con el fin de analizar las funcionalidades de Evolución 4.5. Para tal fin se diseñaron Formatos de Pruebas para evaluar los componentes de Evolución 4.5, donde el estudiante con conocimientos previos en el software probaba cada uno de los íconos, funciones y barras de la interfaz, entre otras, las formas de acceder y ejecutar la herramienta a través del teclado, ratón y barras de menús.

Tabla 19. Ficha Técnica Pruebas de Aceptación

Universidad	Materia	Docente	No. Estudiantes	Casos de Prueba
Universidad Industrial de Santander	Pensamiento Sistémico	Hugo Hernando Andrade Sosa	12	131
Universidad Cooperativa de Colombia	Modelación I	Gina Paola Maestre	13	49

El formato de prueba diseñado consta de un encabezado con la información concerniente a los desarrolladores del proyecto y las entidades que intervienen en el mismo. Enseguida se especifica el caso de prueba, las condiciones para ejecutarlo, los pasos a seguir, los valores que debe utilizar, el resultado esperado y la casilla de verificación. Posteriormente esta ubicado un recuadro con las observaciones y la decisión de aprobación o fallo del caso de prueba.

Figura 27.Formato de Prueba utilizado para las Pruebas de Aceptación

Universidad Industrial de Santander	**UNIVERSIDAD INDUSTRIAL DE SANTANDER** **GRUPO SIMON DE INVESTIGACIÓN**	Grupo SIMON de Investigación
	Aplicación: Software de Modelado y Simulación con Dinámica de Sistemas	
	Nombre: Evolución 4.5	
	Responsables: Adriana Judith Monsalve Quintero	**Estado:** En desarrollo
	Alexander Elías Hernández Cuadrado	**Página:** 5 de 12

Formato Casos de Pruebas			
ID Caso de Uso:	CU - 01 Editor	**Tipo de Prueba:**	Prueba de Aceptación
ID Nombre Escenario:	01-05 Eliminar Elemento	**Autor del Caso de Prueba:**	Adriana Monsalve
ID Nombre Caso de Prueba:	01-05-01 Eliminar Elemento en el Editor de Diagramas de Influencias	**Nombre del Probador:**	
		Fecha de Ejecución:	

Condición(es) para que se ejecute el Caso de Prueba:

Haber creado un elemento en el Editor de Diagramas de Influencias. Luego debe seleccionar el elemento y probar con todas las opciones para eliminarlo.
Las relaciones de información y de material, el ciclo, el sector y el clon no tienen la opción de ser eliminadas con la opción clic derecho/eliminar elemento.
Las relaciones de información y de material no tienen la opción de ser eliminadas con la opción edición/eliminar.

Para la Ejecución del ID del Caso de Prueba:

Paso	Condición	Valor(es)	Resultado Esperado	Resultado Obtenido
Edición/Eliminar		Elemento	Al seleccionar un elemento y realizar cualquiera de las operaciones para eliminarlo, éste desaparece del editor de diagramas de influencias.	☐
Clic derecho/Eliminar Elemento				
Tecla Suprimir				
Edición/Eliminar		Ciclo, Clon, Sector		☐
Clic derecho/Eliminar Elemento				☐
Tecla Suprimir				☐
Tecla Suprimir		Relación de Información		☐
		Relación de Material		☐

Criterios de Aprobación del Caso de Prueba:	Si se cumplen en un 100% los resultados esperados				
Observaciones:					
Decisión de Aprobación del Caso de Prueba:	**Aprobó:**		**Falló:**		**(marque con una X el resultado obtenido)**
Documentos Anexos:					

Metodología usada para la realización de las Pruebas de Aceptación.

En la realización de esta prueba se generaron casos de prueba específicos para cada uno de los componentes de la interfaz de la herramienta. Estos casos de prueba se ejecutaron en Evolución 4.5 teniendo en cuenta las condiciones y siguiendo los pasos definidos en los Formatos de Pruebas en Excel. Después se comparó el resultado generado con el esperado del caso de prueba, si ambos resultados coinciden damos clic en el recuadro *"Resultado Obtenido"* y marcamos con una X la casilla *"Aprobó"*, de lo contrario debían apuntar las observaciones y adjuntar los anexos (imágenes, modelos, pasos que conllevaron al error) involucrados en el fallo.

Figura 28. Pasos a seguir dentro del Formulario de Pruebas

Para la realización de estas pruebas se crearon 5 libros de Excel compuesto de 49 casos de prueba, cada uno con 1 o más opciones a verificar, de las cuales fallaron 5 casos de prueba. En el Anexo_6_Resultados_Pruebas_Aceptacion del libro se encuentran los resultados obtenidos de estas pruebas.

Tabla 20. Componentes de Evolución 4.5 probados

Funcionalidades de Evolución	Casos de Prueba	Opciones ejecutadas dentro de los casos de prueba	Casos de Prueba que fallaron	Funcionalidad que presentó fallo
Editor de Diagramas de Influencias	12	75	2	Opciones Deshacer y Rehacer
Editor de Diagramas de Flujo Nivel	14	95	0	---
Motor	7	23	0	---
Graficador	7	37	0	---
Animador	9	62	1	Track
TOTAL	**49**	**292**	**3**	**---**

Al término de estas pruebas se comprobó que del total de los casos de prueba ejecutados de Evolución 4.5 fallaron el 6.12%.

Las funciones matemáticas, lógicas, de conversión y trigonométricas de Evolución 4.5 se probaron haciendo la comparación entre los resultados

obtenidos por el software y los generados por Excel. Para cada función se creó un caso de prueba, el cual fue ejecutado en las dos herramientas. Luego se verificó si los resultados obtenidos coincidían.

Para probar las funciones de Evolución 4.5 se crearon 82 casos de prueba encontrándose un (1) fallo, la Función ATANH (Arco Tangente Hiperbólica) mostró un cuadro de error y no calculó el valor que se esperaba. Los fallos encontrados se reportaron a la fase de la adquisición de errores definida en el numeral 7.3.3 para su corrección. Algunas funciones son propias de Evolución, por lo tanto no se podían comparar con Excel. Los resultados obtenidos de esta prueba se encuentran en el Anexo_6_Resultados_Pruebas_Aceptación.

Figura 29. Funciones Probadas de Evolución 4.5

FUNCIONES QUE MANEJA EVOLUCIÓN 4.5

FUNCIÓN	TIPO DE FUNCIÓN	DEFINICIÓN	SIGNIFICADO	FORMA DE COMPARAR	COMPARADA CON	RESULTADO EN EVOLUCIÓN	RESULTADO EN EXCEL	FUNCIONA
+	Matemática	+	Adición y Más unario.	Sumar dos valores numéricos.	Excel	Suma los números escogidos	Suma los números escogidos	☑
-	Matemática	-	Substracción y Menos unario.	Restar dos valores numéricos.	Excel	Resta los números escogidos	Resta los números escogidos	☑
*	Matemática	*	Multiplicación.	Multiplicar dos valores numéricos.	Excel	Multiplica los números escogidos	Multiplica los números escogidos	☑
/	Matemática	/	División.	Dividir dos valores numéricos.	Excel	Divide los números escogidos	Divide los números escogidos	☑
<	Lógica	<	Menor que.	Comparar si un número es menor que otro.	Excel	1 ó 0 (1=VERDADERO, 0=FALSO)	VERDADERO ó FALSO	☑
<=	Lógica	<=	Menor o Igual que.	Comparar si un número es menor e igual que otro.	Excel	1 ó 0 (1=VERDADERO, 0=FALSO)	VERDADERO ó FALSO	☑
<>	Lógica	<>	Diferente que.	Comparar si un número es diferente que otro.	Excel	1 ó 0 (1=VERDADERO, 0=FALSO)	VERDADERO ó FALSO	☑
=	Lógica	=	Igual que.	Comparar si un número es igual que otro.	Excel	1 ó 0 (1=VERDADERO, 0=FALSO)	VERDADERO ó FALSO	☑
>	Lógica	>	Mayor que.	Comparar si un número es mayor que otro.	Excel	1 ó 0 (1=VERDADERO, 0=FALSO)	VERDADERO ó FALSO.	☑
>=	Lógica	>=	Mayor o Igual que.	Comparar si un número es mayor e igual que otro.	Excel	1 ó 0 (1=VERDADERO, 0=FALSO)	VERDADERO ó FALSO. MAYOR.O.IGUAL: Prueba si un número es mayor que el valor de referencia, donde 1 = Verdadero y 0 = Falso.	☑

Al término de esta prueba se comprobó que del 100% de las funciones de Evolución 4.5 ejecutadas, fallaron el 0.82%.

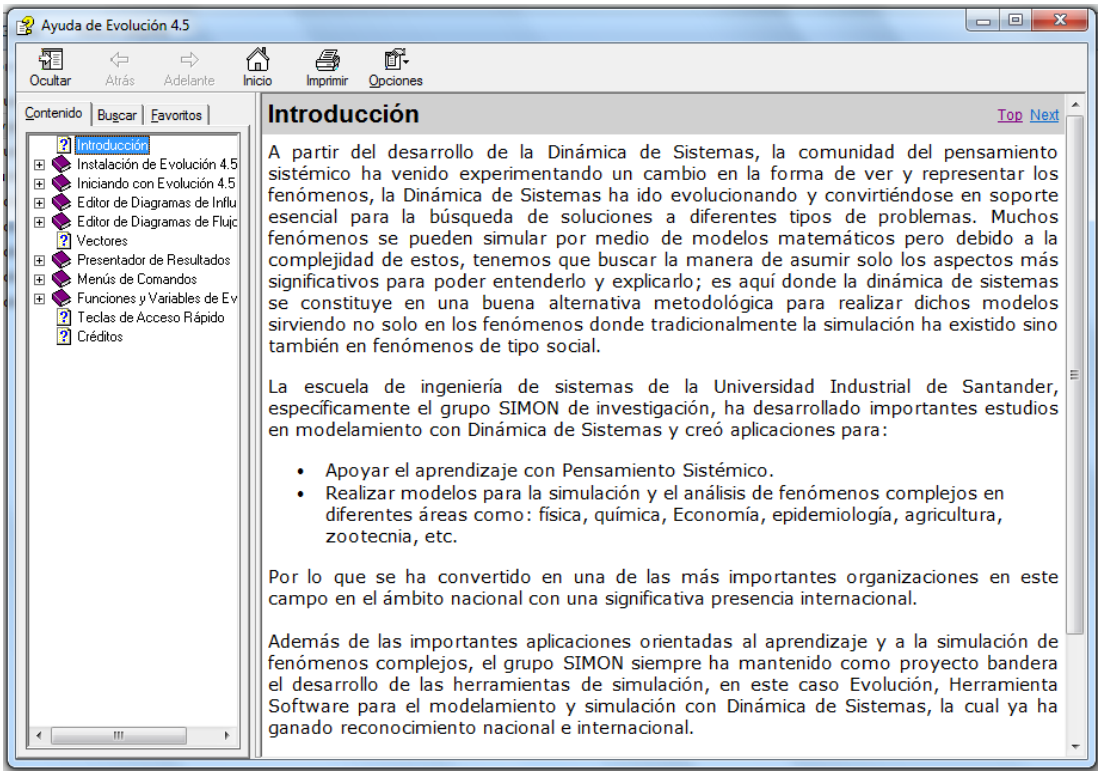
6.3.7 Liberación del producto. Se dejó a disposición de los usuarios la versión 4.5 de Evolución con los errores ya corregidos.

6.4 DOCUMENTACIÓN

Entre la documentación del software Evolución, se encuentran los diagramas UML, los manuales de usuario y del programador, la ayuda y un videotutorial.

6.4.1 Ayuda de Evolución 4.5. La ayuda está incorporada en el software a través de botones en varios cuadros de diálogo y en la barra de menús. Allí se muestra todo el contenido del software con gráficos que facilitan al usuario el entendimiento de Evolución y de todos sus componentes.

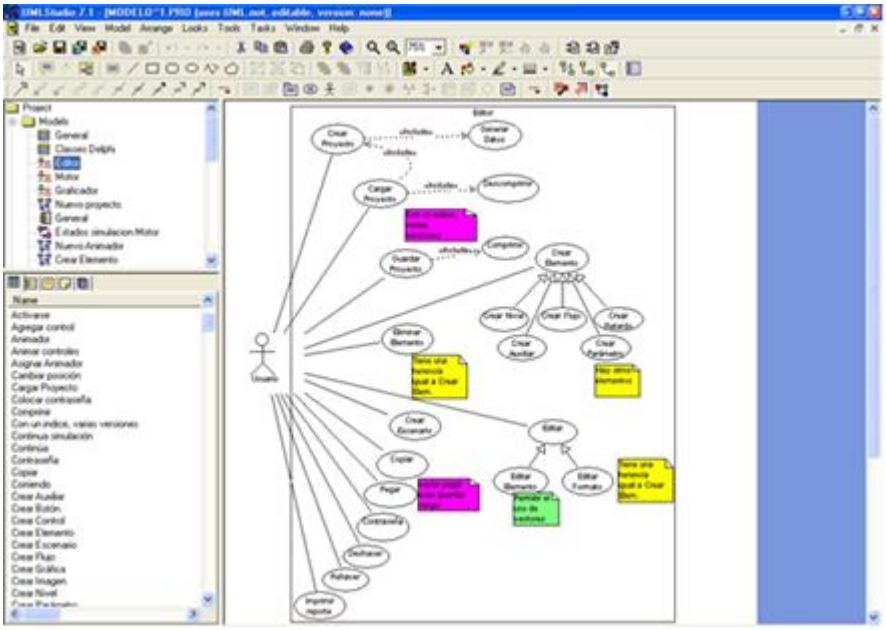
Imagen de la ayuda.



6.4.2 Diagramas UML. Evolución posee diagramas UML que permiten modelar sistemas diseñados a objetos. Entre ellos se encuentran los Diagramas de Casos de Uso, Diagramas de Clase, Diagramas de Secuencia, Diagramas de Actividad y Diagramas de Estado, que representan el comportamiento y la estructura del software.

Los diagramas de Evolución se realizaron con UML Studio 7.1, una herramienta de modelado ampliamente utilizada. En el Anexo_7_Documentacion se encuentran los diagramas mencionados anteriormente.

Figura 30. Interfaz de UML Studio



6.4.3 Manuales de Evolución 4.5. El software Evolución 4.5 cuenta con un Manual del Usuario que explica los requisitos para la instalación del software, los elementos que forman un modelo, ya sea para diseñar el Diagrama de Flujo Nivel o de Influencias, las funciones, el presentador de resultados, los animadores, vectores, comandos y demás propiedades de la herramienta. Es una guía muy útil para el usuario, pues lo asesora con todo lo relacionado al funcionamiento de Evolución. Está disponible en la página web del software[96] en la pestaña manuales.

También existe un Manual del Programador que explica los componentes básicos de Evolución como son el Editor, el Motor y el Animador, los patrones de diseño que se tuvieron en cuenta para desarrollar el proyecto, así como

[96] http://simon.uis.edu.co/joomla/home/index.php?option=com_content&view=article&id=214&Itemid=96

objetos especiales de Evolución (empaquetador de listas, documento de Word, datos en memoria, hilo evento, evaluador, compresor). Este manual ayuda al programador de Evolución a entender la lógica del software, las clases y los procedimientos o métodos con los que se llevaron a cabo la programación de la herramienta.

6.4.4 Video tutorial de Evolución 4.5. Existen muchas herramientas en el mercado actual para realizar videos y cada una de ellas posee capacidades distintas que los hace sobresalir de sus competidores. En este caso, Camtasia Studio es uno de los programas más conocidos y bastante fácil de manejar.

De la mano de TechSmith, Camtasia se impone como uno de los mejores software de captura de pantalla. Su fácil uso y su versatilidad, lo convierten en una muy buena alternativa a la hora de colocar presentaciones en la web.

Por todas estas características se escogió este programa para realizar el videotutorial que explica paso a paso el ejemplo de una población de conejos, así como el funcionamiento de la interfaz de Evolución, sus elementos, menús, ventanas y propiedades. Es 1 prototipo que empieza con un modelo elemental y va aumentando con nuevas variables que aparecen y reflejan el comportamiento esperado para una población. Este videotutorial se encuentra publicado en la página de Evolución en la pestaña manuales y en el Anexo_7_Documentacon.

6.5 PAGINA WEB DE EVOLUCIÓN 4.5

Para satisfacer el Objetivo Específico No. 2 y *c*on el propósito de brindar soporte a los usuarios de la herramienta de Modelado y Simulación, se creó la página web del software Evolución 4.5 con contenido específico, para conocer y facilitar el uso de la misma.

Anteriormente solo se encontraba en Internet el instalador de diferentes versiones de Evolución, la cual se podía descargar desde la página

http://simon.uis.edu.co/WebSIMON/software/indsof.php. Debido a que no había mayor información acerca de esta herramienta de modelado y simulación publicada en Internet, fue necesario documentar el software con la elaboración de una página web que mostrara los comienzos de Evolución desde los años 90, sus usos y aplicaciones hasta el día de hoy, llegando de esta manera a usuarios de la Dinámica de Sistemas en diferentes partes del mundo y así puedan conocer esta herramienta de modelado y simulación llamada Evolución 4.5.

La url para acceder al sitio es: http://simon.uis.edu.co/joomla/home/index.php?option=com_content&view=article&id=215&Itemid=92, la cual se encuentra en el servidor del grupo Simon con los siguientes temas:

- Descripción
- Historia
- Usos y Aplicaciones
- Ejemplos
- Descargas
- Manuales
- Preguntas Frecuentes
- Foro

Cada link presenta información concerniente al software Evolución con ejemplos, gráficos y manuales para facilidad del usuario.

Figura 31. Página Web de Evolución

6.6 ARTICULO "EVOLUCIÓN HERRAMIENTA SOFTWARE PARA EL MODELADO Y SIMULACIÓN CON DINÁMICA DE SISTEMAS".

Autores:

HUGO HERNANDO ANDRADE SOSA

Ingeniero de Sistemas UIS

Magister en Informática UIS

Profesor Titular Escuela de Ingeniería de Sistemas e Informática

Universidad Industrial de Santander

handrade@uis.edu.co

EMILIANO DE JESUS LINCE MERCADO

Ingeniero de Sistemas UIS

Magister en Informática UIS

Universidad Industrial de Santander

emilianolince@hotmail.com

ALEXANDER ELIAS HERNANDEZ CUADRADO
Estudiante X Semestre de Ingeniería de Sistemas
Universidad Industrial de Santander
otroalex1980@hotmail.com

ADRIANA JUDITH MONSALVE QUINTERO
Estudiante X Semestre de Ingeniería de Sistemas
Universidad Industrial de Santander
adrianamonsalve8@hotmail.com

Resumen: Se pretende dar a conocer a la comunidad dinámico-sistémica una herramienta para el modelado y simulación con Dinámica de Sistemas llamada EVOLUCIÓN, este software es un desarrollo colombiano que se ha llevado a cabo en el grupo SIMON de Investigación en Modelado y Simulación de la Universidad Industrial de Santander. Evolución es una herramienta que está en español, desarrollada por la comunidad latinoamericana, esto nos da la posibilidad y flexibilidad de adaptarla a nuestras necesidades específicas de investigación y aplicación. Este artículo describe las características de Evolución, su historia y lo que viene para la herramienta en un futuro.

Evento:

Con motivo del 7° Congreso Latinoamericano de Dinámica de Sistemas y 7° Encuentro Colombiano de Dinámica de Sistemas, realizado en Santa Marta - Colombia del 4 al 7 de Noviembre de 2009, se elaboró y postuló el artículo **"EVOLUCIÓN HERRAMIENTA SOFTWARE PARA EL MODELADO Y SIMULACIÓN CON DINÁMICA DE SISTEMAS"**, para dar a conocer a la comunidad en general el software Evolución y todas sus características.

Este artículo también fue postulado para la Revista Latinoamericana de Dinámica de Sistemas perteneciente a la Universidad de Talca en Chile, el cual fue aceptado y publicado en el Volumen 5, Numero 1 de fecha Marzo de 2011,

de dicha revista y puede ser visto en la siguiente dirección: http://dinamica-sistemas.mty.itesm.mx/docs/RDS_6_1_4.pdf

Es así como se cumple el objetivo N° 3 del plan del proyecto Mantenimiento del Software Evolución 4.0, el cual consiste en elaborar y postular un artículo acerca de Evolución a una revista o como ponencia en un evento, para dar a conocer a toda la comunidad dinámico sistémica la herramienta como un software de modelado y simulación con Dinámica de Sistemas. En los Anexo_8_Articulo se encuentra el artículo completo, las memorias del Congreso y el certificado de Ponente de uno de los autores.

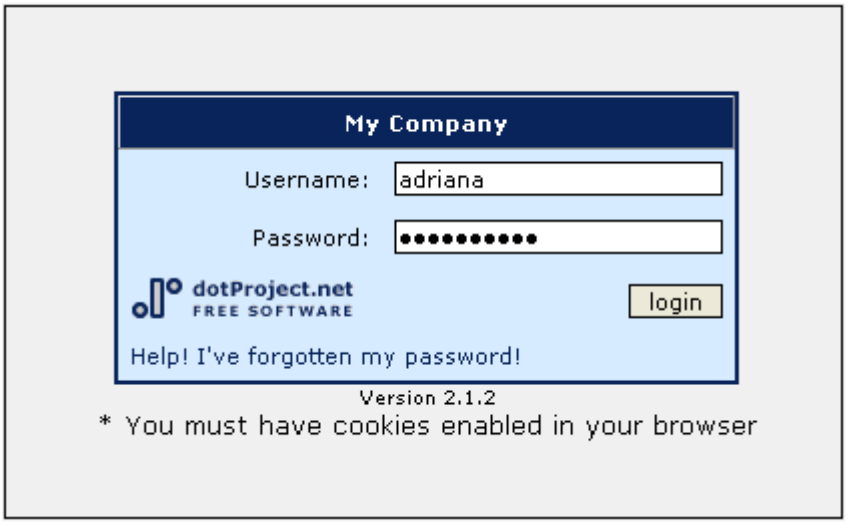
Figura 32. Imagen oficial del 7° Congreso Latinoamericano de Dinámica de Sistemas

7 CRONOGRAMA DE ACTIVIDADES

El cronograma de actividades se realizó en la herramienta Dotproject. Esta plataforma cuya interfaz permite la gestión de proyectos que nace como alternativa a los costosos productos de Microsoft y otras costosas herramientas disponibles en el mercado. Sus requerimientos esenciales son: Interfaz de usuario simple y consistente, funcionalidad para la gestión de proyectos, libre uso-acceso y de código abierto. Algunos servicios de la plataforma incluyen: Gestión de Usuarios, E-mail, Gestión de información de clientes y compañías, Listado de proyectos, Listado jerárquico de tareas, Repositorio de Archivos, Lista de Contactos, Calendario, Foro, Acceso restringido a los recursos.

El portal de ingreso es http://simon.uis.edu.co/dotproject, para el cual se debe ser usuario y tener una clave de acceso al sitio.

Figura 33. Portal de ingreso a Dotproject



8 CONCLUSIONES

Para la culminación de este trabajo cabe destacar que para satisfacer nuestro objetivo general propuesto, se realizaron diferentes esfuerzos para alcanzar cada uno de los objetivos específicos ya que estos son distintos en sus contextos, por esta razón a continuación se enunciaran las conclusiones para cada objetivo específico:

El objetivo específico No.1 plantea:

> *"Desarrollar una nueva versión de Evolución que contemple corrección de errores detectados a través de pruebas, reportados por los usuarios, la evaluación hecha al software Evolución y los resultados obtenidos en la tesis de maestría de J. Moreno[97].*
>
> a. *Evaluar el software Evolución 4.0, a partir de la información disponible (tesis de maestría de J. Moreno, requisitos, diagramas UML, código fuente, manuales) utilizando herramientas computacionales para valorar los atributos de calidad externos e internos definidos para Evolución[98], a fin de encontrar fallos y proponer mejoras a la herramienta, consolidando los resultados en una ficha técnica.*
> b. *Corregir errores de sintaxis, de ejecución y de lógica presentes en el software, utilizando el lenguaje de programación Delphi, con el propósito de aportar al cumplimiento de los requisitos funcionales de Evolución.*
> c. *Realizar pruebas del sistema, de unidad, de integración, y de validación al software Evolución 4.0, utilizando herramientas de pruebas[99] que faciliten esta labor, a fin de detectar defectos en el software y verificar el funcionamiento de Evolución".*

[97] [19] MORENO CHAUSTRE, Jorge Jair. Diseño de una arquitectura para un entorno de modelamiento – simulación y creación de un proceso para su desarrollo por una comunidad (I+D). Tesis de Maestría. Maestría en Informática. Bucaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas. 2006. 246 p.

[98] Apartado 6.1.1 del Marco Teórico "Atributos de calidad utilizados para evaluar a Evolución 3.5 y 4.0".

[99] Herramientas CAST (Computer Aided Software Testing).

d. *Completar la documentación de Evolución tales como ayuda, manual del usuario, manual del programador, elaborando el contenido faltante y reestructurando el que esté insuficiente, a fin de que la documentación quede actualizada, corregida y publicada en el sitio web del grupo SIMON de Investigación".*

Para la resolución del inciso a) del objetivo específico No. 1, el primer paso que se realizó fue hacer un estudio profundo de la tesis de maestría de J. Moreno, que implico el análisis de la información obtenida, video conferencias continuas con su desarrollador y una visita del Mg. Jorge Jair Moreno Chaustre a la ciudad de Bucaramanga, en razón a que labora en la Universidad del Cauca, con el motivo exclusivo de reunirse tanto con el desarrollador de Evolución 3.5 como con los desarrolladores de este proyecto, para orientar y revisar la Evaluación Arquitectónica de Evolución 4.0.

Se debe aclarar que la diferencia fundamental entre Evolución 3.5 y Evolución 4.0 es la inclusión del componente FIS en la última versión del software, esto conllevo a centrarnos en aplicar la Evaluación a este componente solamente, puesto que la tesis de maestría de J. Moreno se encargó de la valoración de la versión 3.5 de Evolución. Por lo anterior en la valoración de los atributos de calidad internos y externos de Evolución 4.0 se obtuvieron una menor cantidad de datos, arrojando un total de 3.486 datos dispuestos en 256 registros que se organizaron en 5 aspectos de la herramienta. Por otro lado, se encontraron 125 problemas arquitectónicos de los cuáles 91 problemas se resuelven en la nueva documentación del software, quedando 34 problemas que requieren de la atención de los desarrolladores aún sin resolver.

Se volvió a evidenciar la escasez de herramientas que permitieran realizar de forma automática la totalidad del análisis y evaluación arquitectónicos como se concluyó igualmente en la tesis de maestría. Solamente se encontraron tres herramientas para tal propósito. Sin embargo se logró aprovechar la combinación de estas herramientas mediante técnicas automáticas, semi-

automáticas y manuales para cubrir todos los aspectos deseados del análisis y evaluación arquitectónicos.

Finalmente se obtuvo un conocimiento amplio de la gran cantidad de métricas existentes para valorar el software Evolución 4.0 y a la vez se asimilaron dichos resultados facilitando la comprensión y agrupación de estos en las heurísticas de diseño contempladas dentro de la valoración arquitectural.

Para el inciso b) se realizó en primera medida el estudio y aprendizaje de la herramienta Delphi 7, en la cual se desarrolló el Software Evolución 4.0 y la comprensión del código fuente. La corrección de errores de sintaxis, de ejecución y de lógica es un ámbito muy delicado de llevar a cabo; se debe ser muy cauteloso y ceñirse a la metodología planteada para la corrección de errores de código y así evitar introducir nuevos errores al codificar las soluciones y no afectar la arquitectura del software.

Después de llevar a cabo esta corrección se generó una nueva compilación del software Evolución contribuyendo así con su fiabilidad y robustez.

Por último se deja claro que no a todos los errores y bugs se les dio solución debido a la naturaleza de los mismos, pero se entrega una tabla recomendando las posibles soluciones a estos para que en un próximo mantenimiento a la herramienta sean abordados.

Para el inciso c) se investigaron las posibles herramientas de pruebas que ofrecía el mercado, seguido de esto se pasó a escoger las más apropiadas para cada tipo de prueba. Se encontraron pruebas que al llevarlas de la teoría a la práctica fueron bastante complicadas de aplicar a Evolución. Como por ejemplo las pruebas de unidad, que en la teoría encontrada se encaminan mucho a las funciones que contienen algoritmos aritméticos. Por su parte en la codificación de Evolución la mayoría de los cálculos están encapsulados en los procedimientos y no en las funciones, además presenta el manejo de

excepciones en gran parte de su estructura haciendo de esto un fuerte en la recepción de datos entre las funciones y asegurando una mayor robustez.

Se realizaron pruebas de rendimiento a Evolución 4.5 utilizando la herramienta AQTime que permitió analizar el software observando las rutinas y clases que eran utilizadas al ejecutar alguna acción y el tiempo que gastaban en hacerlo. Se evidenció que los procesos ejecutados en Evolución 4.5 correspondían a las rutinas llamadas dentro del código y en efecto los tiempos de su ejecución eran más tardíos con respecto a las demás rutinas.

Para la realización de las pruebas de aceptación al software Evolución 4.5, se hizo un estudio minucioso de las funcionalidades de la interfaz de la herramienta, plasmándolo en los formatos de prueba, que permitieron medir el comportamiento esperado de cada una de estas. Las pruebas se llevaron a cabo con la ayuda de estudiantes universitarios, que probaron en detalle los menús, iconos y botones de la interfaz gráfica, mediante casos de prueba previamente elaborados El porcentaje que arrojaron estas pruebas evidencia que Evolución 4.5 cumple con los requisitos de funcionalidad y tiene un nivel alto de aceptación.

Para el inciso d) se actualizó y complementó la documentación de Evolución 4.5 tales como ayuda, manual del usuario, manual del programador y diagramas UML, además se creó un videotutorial, que explica paso a paso como crear un modelo, así como el funcionamiento de la interfaz de Evolución, sus elementos, menús, ventanas y propiedades; tanto el manual del usuario como el videotutorial quedaron publicados y a disposición del usuario final en el sitio web del grupo Simon de Investigación.

Tras haber abordado estos cuatro tópicos se generó una nueva versión del software Evolución 4.5, una herramienta fiable, robusta y con una documentación completa y actualizada para los futuros usuarios y desarrolladores.

En cuanto al objetivo específico No. 2 que propone:

> *"Brindar soporte a los usuarios de Evolución, a través de la creación de una página web en el servidor del grupo SIMON[100], con el contenido de la herramienta y un foro, para mantener un contacto frecuente y conocer las necesidades del usuario".*

Para el cumplimiento de este objetivo, el grupo Simon de investigación contaba con toda la infraestructura web para montar este página, por ello se aprovechó el sitio del grupo para alojar el contenido concerniente a Evolución 4.5 dentro del link *Software*, además existía un conocimiento previo del gestor de páginas web Joomla por parte de los desarrolladores, para cargar el contenido de la página a la plataforma. También se realizó una investigación con respecto a los diferentes proyectos de grado que a través del tiempo contribuyeron a la construcción de Evolución.

El objetivo específico No. 3 plantea:

> *"Elaborar y postular un artículo acerca de Evolución a una revista o como ponencia en un evento, para dar a conocer a toda la comunidad dinámico sistémica la herramienta como un software de modelado y simulación con Dinámica de Sistemas."*

Para alcanzar los resultados se realizó una investigación exhaustiva de otras herramientas software para modelado y simulación con Dinámica de Sistemas existentes en el mercado y del mismo software Evolución, ahondando en sus características, funciones y prestaciones y de esta manera poder realizar un cuadro comparativo entre ellas.

En el artículo "EVOLUCIÓN HERRAMIENTA SOFTWARE PARA EL MODELADO Y SIMULACIÓN CON DINÁMICA DE SISTEMAS" se relata la historia de cómo se ha desarrollado el software Evolución desde sus inicios, las herramientas con que cuenta para el modelado y simulación, en qué estado se

[100] http://simon.uis.edu.co/joomla/home/index.php?option=com_content&view=article&id=215&Itemid=92

encuentra y hacia donde se quiera llevar en un futuro. Se hizo la presentación formal del artículo a nivel internacional, demostrando que con sus características y funcionalidades esta a la altura de otras herramientas de modelado y simulación con Dinámica de Sistemas existentes en el mercado.

Este artículo no solo fue ponencia en el 7° Congreso Latinoamericano de Dinámica de Sistemas y 7° Encuentro Colombiano de Dinámica de Sistemas, realizado en Santa Marta - Colombia del 4 al 7 de Noviembre de 2009 sino que fue aceptado y publicado en la Revista Latinoamericana de Dinámica de Sistemas perteneciente a la Universidad de Talca en Chile, en su Volumen 5, Numero 1 de fecha Marzo de 2011. Motivo que nos llenó de orgullo y satisfacción por el esfuerzo realizado y aprendizaje obtenido.

Sin duda para nosotros como desarrolladores elaborar y postular este artículo fue una gran experiencia, primero al escribir en conjunto con el Profesor Hugo Hernando Andrade Sosa, experto en el tema y el Ingeniero Emiliano de Jesús Lince Mercado, desarrollador de Evolución 3.5, no fue tarea fácil, pero gracias a sus orientaciones y el conocimiento adquirido a través del proyecto pudimos realizarlo.

Después de haber terminado este proyecto se puede concluir que para realizar el mantenimiento de software es primordial tener toda la documentación del mismo, tanto el código fuente actualizado, los diagramas UML, como los manuales del programador y del usuario. Corroborar que el código fuente a modificar corresponda a la versión que está liberada y que sea la que está siendo utilizada por los usuarios. Si esta documentación está incompleta se pierde tiempo al tratar de comprender el funcionamiento de la herramienta y puede que no se haga de la manera correcta dificultando así nuestra labor de mantenimiento.

Por otra parte con este proyecto se le aporta al grupo Simon una base teórica y práctica para probar y documentar los proyectos venideros en fase de desarrollo.

9 RECOMENDACIONES

Se recomienda cambiar a Evolución 4.5 de Delphi 7 a una versión más actualizada de Delphi de 2010 en adelante, por las facilidades de automatización de pruebas que presentan estas nuevas versiones.

Generar una versión de Evolución 4.5, que sea susceptible al cambio de idioma, en razón al uso de la herramienta en diversas instituciones a nivel internacional.

Se implemente una funcionalidad de autoguardado a Evolución 4.5, a fin de contribuir a la recuperación de datos en caso de interrupción abrupta del sistema.

Se haga un mantenimiento periódico de la herramienta por el desarrollo de nuevas versiones, trabajos de grado asociadas, su difusión y utilización nacional e internacional.

Realizar una evaluación formal a los nuevos desarrollos para Evolución que en la actualidad siguen aportando a la construcción del software y que contemplen la valoración de la arquitectura de la herramienta a fin de conformar la noción de calidad del producto.

BIBLIOGRAFÍA

Libros

[1] ANDRADE, H., DYNER, I., ESPINOSA, A., LOPEZ, H., y SOTAQUIRA, R. Pensamiento Sistémico: Diversidad en búsqueda de Unidad. División de Publicaciones UIS. 2001. 423 p.

[2] ANDRADE SOSA, Hugo Hernando, GOMEZ FLOREZ, Luis Carlos. "Tecnología Informática en la Escuela". Tercera Edición. División de Publicaciones UIS. 2008. 461p.

[3] BRAUDE, Eric J. "Ingeniería de Software, Una perspectiva orientada a objetos". Primera Edición. Alfaomega. 539 p.

[4] CARVALHO, A. M. B. R.; CHIOSSI, Thelma Cecília dos Santos. Introdução à Engenharia de Software. 02. ed. Campinas: Editora da Unicamp, 2001. v. 1. 148 p.

[5] FRAZER, A. "Reverse engineering – hype, hope or here?", P.A.V. Hall, Software Reuse and Reverse Engineering in Practice. Chapman & Hall. 1992.

[6] MYERS, G. J. "The Art of Software Testing". Nueva York. John Wiley & Sons. 1979.

[7] HESTENES, D. "Towards a Modeling Theory of Physics Instruction. American Journal of Physics". 1987.

[8] LARMAN, Craig. "UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado". Segunda Edición. Prentice Hall. Madrid, 2003. 624 p.

[9] PFLEEGER, Shari Lawrence. "Ingeniería de Software, Teoría y Práctica". Primera Edición. Buenos Aires: Pearson Education. 2002. 792 p.

[10] PIATTINI, Mario G., CALVO, José A., CERVERA, Joaquín., FERNANDEZ, Luis. "Análisis y Diseño de Aplicaciones Informáticas de Gestión. Una perspectiva de Ingeniería del Software". Alfaomega. 2004. 710 p.

[11] PIATTINI, Mario G., GARCÍA Félix O. "Calidad en el desarrollo y mantenimiento del software". Alfaomega. 2003. 310 p.

[12] PIATTINI, Mario, VILLALBA, José, RUÍZ, Francisco, POLO, Macario y otros. “Mantenimiento del Software, Modelos, Técnicas y Métodos para la Gestión del Cambio”. Alfaomega. 2001. 336 p.
[13] POSTON, R., SEXTON, M. Evaluating and Selecting Testing Tools. IEEE Software, mayo 1992, p. 33-42
[14] PRESSMAN, Roger S. “Ingeniería del Software un Enfoque Práctico”. Quinta Edición. McGrawHill. 2001. 640 p.
[15] SCHACH, Stephen. Ingeniería de Software Clásica y Orientada a Objetos. México: McGraw-Hill, 2006. p.
[16] SOMMERVILLE, I. “Software Engineering”. Cuarta Edición. USA: Addison-Wesley, 1992
[17] TEXEIRA, Steve; PACHECO, Xavier. “Guía de Desarrollo Delphi 5”. Prentice Hall, 2000.

Trabajos de Grado

[18] CUELLAR YENERIS, Mario y LINCE MERCADO, Emiliano. Evolución 3.5 herramienta software para el modelamiento y simulación con dinámica de sistemas. Tesis de pregrado. Ingeniero de Sistemas. Bucaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas. 2003. 179 p.
[19] MORENO CHAUSTRE, Jorge Jair. Diseño de una arquitectura para un entorno de modelamiento – simulación y creación de un proceso para su desarrollo por una comunidad (I+D). Tesis de Maestría. Maestría en Informática. Bucaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas. 2006. 246 p.
[20] NIÑO DIAZ, Angélica María. Prototipo web para control de versiones software y documentación en línea, aplicado al servidor de la Escuela de Ingeniería de Sistemas – UIS, Cormoran. Tesis de pregrado. Ingeniero de Sistemas. Bucaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas. 2008. 156 p.

Internet

[21] FORRESTER, Jay. Abril 29 de 1991. System Dynamics and the Lessons of 35 Years. <http://sysdyn.clexchange.org/sdep/papers/D-4224-4.pdf>. [Con acceso el 04-05-09].

[22] JURISTO Natalia, MORENO, Ana M., VEGAS Sira. ”Técnicas de Evaluación de Software”. Internet: <http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf>

[23] LARGO GARCIA, Carlos Alberto, MARIN MAZO Erledy. “Guía Técnica para Evaluación de Software”. Internet: <http://www.puntoexe.com.co/site/index.php/productos/53-guiasoft>

[24] MARTÍNEZ PÁRRAGA, Juan Angel. Estándar IEEE 1219 de Mantenimiento del Software. Internet: <http://alarcos.inf-cr.uclm.es/per/fruiz/cur/mso/comple/IEEE1219.pdf>

[25] NIKOUKARAN, Jalal; HLUPIC, Vlatka; PAUL, Ray J. Criteria for simulation software evaluation. <http://portal.acm.org/citation.cfm?id=293172.293256>

[26] Norma Técnica Colombiana NTC 1486 - Documentación. Presentación de tesis, trabajos de grado y otros trabajos de investigación. Internet: <http://www.ulibertadores.edu.co/?idcategoria=1796>

[27] Software Engineering, IEEE Standard 610.12-1990, IEEE 1993.