Universidad de las Ciencias Informáticas

"Facultad 15"

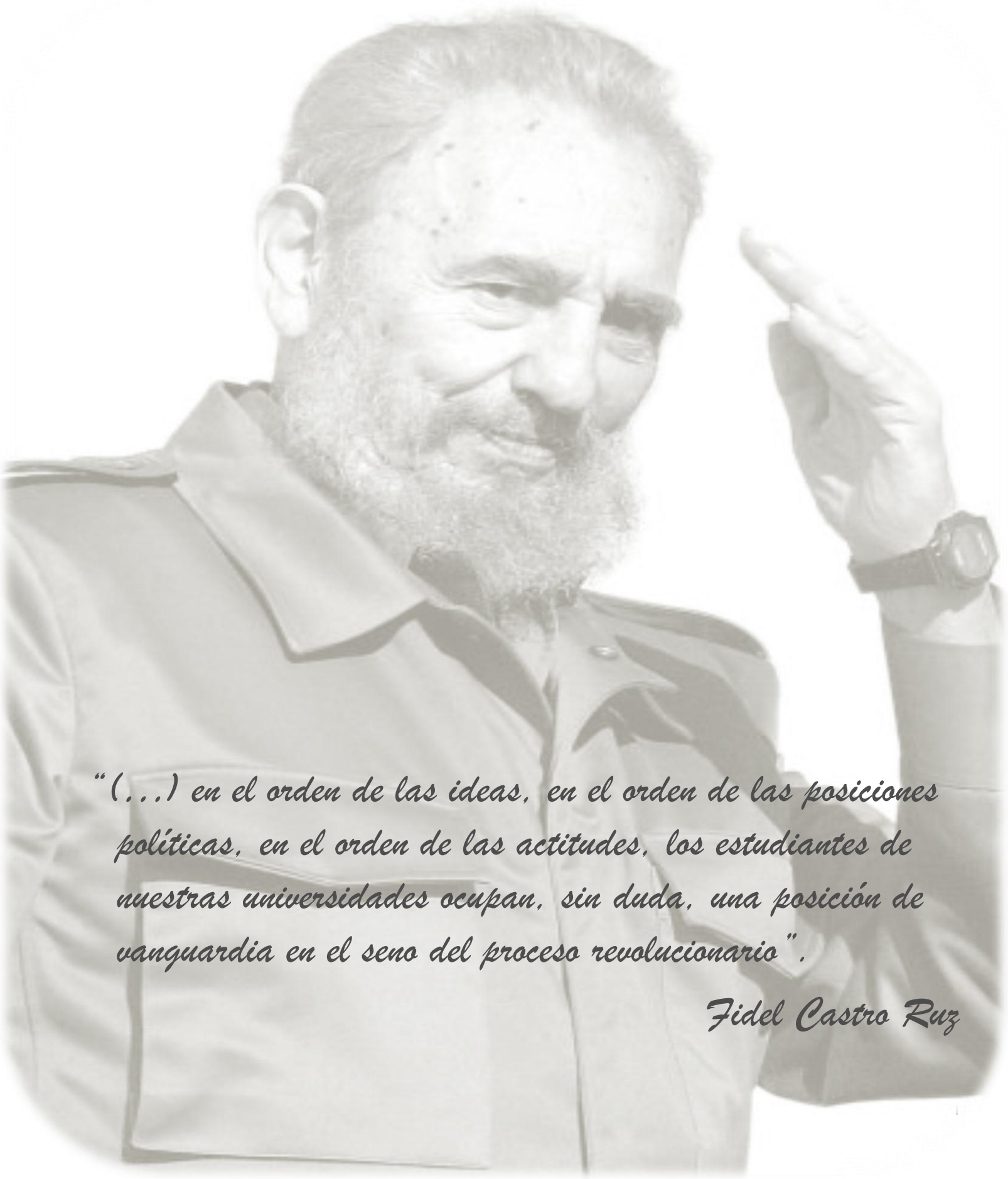
"Implementación de la herramienta SeReq para el seguimiento de los requisitos en el Centro de Informatización de la Gestión de Entidades"

Trabajo de diploma para optar por el título de ingeniero en ciencias informáticas

Autores: Yurisleydy González Hernández
Yusley González Rivera

Tutores: Ing. Yoan Arlet Carrascoso Puebla
Ing. Sandy Machado Scull

Ciudad de la Habana, Junio 2010



"(...) en el orden de las ideas, en el orden de las posiciones políticas, en el orden de las actitudes, los estudiantes de nuestras universidades ocupan, sin duda, una posición de vanguardia en el seno del proceso revolucionario".

Fidel Castro Ruz

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Informatización de la Gestión de Entidades de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____ _____

Yurisleydy González Hernández Yusley González Rivera

Firma de los autores

_____ _____

Yoan Arlet Carrascoso Puebla Sandy Machado Scull

Firma de los tutores

Síntesis de los tutores

Ing. Sandy Machado Scull

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el curso 2008-2009. Actualmente se desempeña como analista en el Centro de Informatización de la Gestión de Entidades.

Correo Electrónico: smachado@uci.cu

Ing. Yoan Arlet Carrascoso Puebla

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el curso 2007-2008. Actualmente se desempeña en el rol de Arquitecto de Datos de la Línea de Contabilidad y Finanzas del Centro de Informatización de la Gestión de Entidades.

Correo Electrónico: yacarrascoso@uci.cu

Clasificación de la tesis: Por roles.

*A **nuestros padres y hermanos**, por estar siempre y querernos tanto.*

*A toda **nuestra familia**, los que están y los que no, por ayudarnos a ser la persona que somos hoy.*

*A todos **nuestros amigos y amigas** que hemos conocido en estos cinco años que han sido especiales.*

A todos dedicamos este momento porque de todos es.

*A **nuestros padres, hermanos y el resto de la familia** que de una forma u otra nos han brindado su apoyo incondicional durante estos años.*

*A nuestros tutores **Sandy y Yoan** por su dedicación y comprensión en todo momento gracias.*

*A todos **nuestros amigos**, en especial a **Rolando y Javier** por brindarnos su apoyo desinteresado.*

A todos gracias.

Resumen

El seguimiento de los requisitos de software representa un amplio beneficio en el desarrollo de aplicaciones, pues permite mantener una traza directa de los requisitos identificados con el resto de los artefactos del sistema. Esta actividad en el Centro de Informatización de la Gestión de Entidades (CEIGE) no se está haciendo de forma eficiente porque no se cuenta con una herramienta, que permita tener acceso a la información de los requisitos en tiempo real por lo que se está necesitando de mucho esfuerzo y tiempo de las personas vinculadas.

Se realizó un estudio actual de varias herramientas existentes para el seguimiento de los requisitos y se llegó a la conclusión de que estas no cumplen con las características de la herramienta que se necesita en el CEIGE ya sea porque son software propietario lo cual no cumple con el principio de independencia tecnológica en el cual está basado actualmente el desarrollo de software en Cuba o porque presentan problemas de usabilidad.

Para el desarrollo de la herramienta se utilizó el marco de trabajo definido para el ERP Cuba (Sauxe), de este se usaron como lenguaje de programación del lado del servidor PHP, como gestor de base de datos PostgreSQL versión 8.3 y los marcos de trabajo que este define para la presentación, acceso a datos y lógica del negocio. La construcción de la solución se realizó basándose en el Modelo de Desarrollo y la arquitectura que se definieron para el ERP Cuba. Como resultado se obtuvo una solución capaz de facilitar el seguimiento de los requisitos en el CEIGE y brindar la información sobre estos al equipo de desarrollo.

Palabras clave

Herramienta, implementación, seguimiento de los requisitos, SeReq, traza.

Índice de contenidos

Índice de Figuras

Índice de Tablas

Introducción

Actualmente, en la Universidad de las Ciencias Informáticas (UCI) se está desarrollando un sistema para la planificación de recursos empresariales o ERP (por sus siglas en inglés, Enterprise Resource Planning). El proyecto encargado de desarrollar este sistema que lleva como nombre Cedrux es el proyecto ERP Cuba que es uno de los proyectos del Centro de Informatización de la Gestión de Entidades (CEIGE). Implementar este producto es un proceso largo, costoso, complejo y que requiere de gran cantidad de recursos humanos.

Durante el flujo de trabajo de Levantamiento de Requisitos realizado en la primera iteración del proyecto ERP Cuba, los analistas comprobaron que los requisitos en cada uno de los subsistemas están estrechamente relacionados y que pueden cambiar con frecuencia. Por esta razón se hace necesario conocer el efecto que puede tener la realización de un cambio en un requisito en el resto, en los artefactos implicados, así como en la implementación de los mismos en el proyecto.

Teniendo en cuenta que el equipo de desarrollo es numeroso, la trasmisión de información del seguimiento de los requisitos se hace difícil, sobre todo cuando la misma se necesita llevar de forma centralizada por los analistas principales del proyecto. Actualmente los analistas principales realizan esta actividad de forma manual, por lo que tienen que visitar los puestos de trabajo de los analistas en cada uno de los diferentes módulos y revisar detenidamente los documentos que contienen esta información para posteriormente actualizarla. Atendiendo a la gran dimensión del proyecto, esta actividad se torna engorrosa y pueden cometerse errores.

La actividad del seguimiento se puede facilitar con el empleo de una herramienta automatizada que ayude a gestionar la información referente a los requisitos y los cambios que se producen en ellos. Actualmente existen varias herramientas con esta finalidad como son: Caliber RM, IBM Rational RequisitePro, RaQuest, IRqA 4.0 y Open Source Requirements Management Tool (OSRMT). La mayoría de estas herramientas son software propietario lo cual no se corresponde con el principio de independencia tecnológica en el cual está basado actualmente el desarrollo de software en Cuba, por lo que no es factible su uso por la licencia que requieren. Por su parte, la herramienta OSRMT, que está basada en software libre, presenta serios problemas de usabilidad dado que los usuarios a menudo no encuentran el flujo de las acciones mientras trabajan con la herramienta además que las versiones no están planificadas

y una nueva versión podría no ser compatible con la anterior. De modo que ninguna de estas herramientas pueden ser utilizadas para el seguimiento de los requisitos en el CEIGE.

La **problemática** descrita anteriormente permitió definir el **problema científico**: El modo en que se realiza el seguimiento de los requisitos de software en el CEIGE dificulta el análisis de su trazabilidad y la transmisión de la información entre las partes implicadas en esta actividad.

El **objeto de estudio** es la Gestión de requisitos de software. Luego el **campo de acción** en el que se enmarca el trabajo es el seguimiento de requisitos.

A partir del problema planteado anteriormente el **objetivo general** que se persigue es implementar una herramienta que automatice el seguimiento de los requisitos de software en el CEIGE, que permita facilitar el análisis de su trazabilidad y la transmisión de la información, sobre los elementos que se afectan ante un cambio en un requisito, a las partes implicadas en esta actividad.

Para dar solución al problema y cumplir con el objetivo general se han trazado las siguientes **tareas de investigación:**

- Estudiar los procesos de gestión y seguimiento de requisitos.
- Analizar las herramientas existentes para el seguimiento de requisitos.
- Analizar los artefactos generados durante la etapa de análisis y diseño de la herramienta.
- Estudiar el marco de trabajo del proyecto ERP Cuba (Sauxe).
- Estudiar los estándares de codificación definidos por el equipo de trabajo de Arquitectura.
- Implementar las funcionalidades del sistema.
- Validar los resultados obtenidos.

Hipótesis:

Si se realiza el seguimiento de los requisitos de software de forma automatizada, entonces se podrá facilitar el análisis de trazabilidad de los requisitos y la transmisión de la información, sobre los elementos que se afectan ante un cambio en un requisito, a las partes implicadas en esta actividad.

El documento está estructurado por tres capítulos:

En el **Capítulo 1** se hace un estudio del estado del arte, donde se realiza la fundamentación teórica de la investigación. Se describe el objeto de estudio. También se muestran y describen algunos sistemas existentes para el seguimiento de los requisitos llegando a la conclusión de que ninguna de estas cumple con todas las condiciones de la herramienta que se necesita en el CEIGE. Finalmente se justifican las tendencias y tecnologías a utilizar para la implementación.

En el **Capítulo 2** se hace un análisis de los artefactos que fueron entregados por el analista, de los componentes ya existentes que pueden ser utilizados y de las principales estructuras de datos a usar. Finalmente se hace una descripción del modelo de componentes y análisis de un algoritmo no trivial de gran importancia en la solución.

En el **Capítulo 3** se realizan las pruebas de caja negra y analizan los resultados obtenidos. Además se hace una validación por especialista de los resultados.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se realiza un estudio sobre los aspectos teóricos necesarios para comprender el problema que se presenta. Se toma una posición científica acerca del concepto de gestión de requisitos, además de investigar sobre las herramientas que hoy en día permiten mantener el seguimiento de los requisitos. Se hace además un análisis de las herramientas y tecnologías seleccionadas para el desarrollo del sistema que se pretende implementar.

1.2 Gestión de Requisitos

Los requisitos pueden cambiar con frecuencia a lo largo del desarrollo del software y esto puede estar dado por varios factores. Los cambios pueden ser resultado de avances tecnológicos, modificaciones en leyes o regulaciones, cambios en las estrategias o ambiente del negocio, o bien porque los usuarios cambiaron su forma de pensar. Adicionalmente, se tiene que nuevos requisitos pueden surgir a medida que avanza el desarrollo del sistema o como efecto de la inclusión de otros requisitos. Cada vez que los requisitos sean modificados deben actualizarse los atributos de los requisitos y sus dependencias.

Por estas razones, el proceso de gestión de los requisitos adquiere una vital importancia, puesto que permite a los analistas del sistema supervisar y dar seguimiento a los cambios que se produzcan en ellos lo cual favorece el control del impacto que esto pueda causar sobre el sistema.

Roger Pressman describe la gestión de requisitos como “un conjunto de actividades que ayudan al equipo de desarrollo a identificar, controlar y darle seguimiento a los requisitos y los cambios en cualquier momento”. (Pressman, 2005)

Por otra parte, Bárbara McDonald Landazuri afirma que la gestión de requisitos es un componente esencial en el desarrollo de un proyecto de software ya que provee la dirección y alcance del proyecto. Expone además, que la gestión de requisitos es el proceso encargado de la identificación, asignación y seguimiento de los requisitos, incluyendo la verificación, la modificación y el control de su estado a lo largo del ciclo de vida del proyecto. (Landazuri, 2005)

Dichas definiciones sobre la gestión de requisitos son bastante completas, aunque los autores consideran que la definición más exacta de la gestión de requisitos es la planteada anteriormente por Pressman que de forma general, incluye en su concepto todas las actividades de este proceso.

Con la gestión efectiva de los requisitos será posible entender, desarrollar y entregar las funcionalidades que el cliente requiere, con un mínimo de afectación en el plazo y los recursos acordados garantizando la calidad en la producción de software. *"Los cambios de requisitos deben ser gestionados para asegurar que la calidad de los mismos se mantenga, los problemas suscitados por los cambios de requisitos podrían incurrir en altos costos, siendo el requisito factor crítico de riesgo."* (Landazuri, 2005)

Se propone que la gestión de requisitos sea flexible y adaptable para reunir las necesidades del proyecto. Las características del alcance de esta actividad variarán dependiendo de algunos factores claves como el tamaño y complejidad del proyecto, la experiencia de los clientes y del personal que trabaja en él, así como del propósito y uso del sistema que se construirá.

A pesar de que el proceso de gestión de requisitos es amplio, este trabajo de diploma se centrará en el seguimiento de los requisitos durante el ciclo de vida del software y las herramientas de gestión de requisitos que auxilian y/o automatizan estas tareas.

Específicamente, el seguimiento a los requisitos está sustentado en el concepto de trazabilidad de los requisitos que algunos autores han ofrecido.

Para Francisco Pinheiro, "la trazabilidad es la habilidad de definir, capturar y seguir las trazas dejadas por los requerimientos en otros elementos del ambiente de desarrollo de software, así como las trazas que estos elementos dejan en los requisitos". (Pinheiro, 2000)

Asimismo, la definición de trazabilidad, que se reconoce como la más difundida, "se concentra en la especificación de requisitos de software y requiere en primer lugar, que permita remontarse hacia atrás en el origen de los requerimientos y que también posibiliten referenciar a cada requerimiento en la documentación de desarrollo". (IEEE, 1998)

A partir de estas definiciones se puede entender el seguimiento de los requisitos como una actividad que posibilitará mostrar la relación existente entre los requisitos identificados y de estos con otros artefactos del sistema. Esto supone un amplio beneficio, en tanto ayudará al equipo de trabajo a determinar el impacto de los cambios, lo que favorecerá la reducción de los riesgos.

1.3 Herramientas para la gestión de requisitos

El uso de herramientas de gestión de requisitos es promovido para mejorar tanto la calidad como la productividad en el desarrollo de un proyecto de software. Lo que ha motivado a utilizar este tipo de herramientas es la complejidad al gestionar los requisitos.

Una herramienta de gestión de requisitos debe ser capaz de realizar algunas funciones como el registro, edición y seguimiento de requisitos a su origen así como la generación de informes.

Durante el estudio investigativo para elaborar el estado del arte del presente trabajo de diploma se estudiaron algunas herramientas que existen en el mundo para la gestión de los requisitos: Caliber RM, IBM Rational RequisitePro, RaQuest, IRqA 4.0 y OSRMT.

De Caliber RM, IBM Rational RequisitePro, RaQuest e IRqA 4.0 se pudo constatar que cumplen las necesidades que se exigen de una herramienta de gestión de requisitos para que sean incorporadas a las empresas. A pesar de esto, dichas herramientas constituyen software propietario lo cual no se corresponde con el principio de independencia tecnológica en el cual está basado actualmente el desarrollo de software de Cuba. Por esta razón no serán adoptadas por el CEIGE para el seguimiento de los requisitos de software.

Por otro lado, OSRMT es una herramienta basada en software libre que gestiona los requisitos del sistema durante todo el ciclo de vida del software. Puede ser explotada como aplicación de escritorio pero presenta problemas con la usabilidad, puesto que su diseño gráfico resulta incómodo al realizar algunas de sus funcionalidades. La versión para su empleo a través de la web es aún menos amigable para el usuario, quienes a menudo no encuentran el flujo de las acciones mientras trabajan con la herramienta ademas sus versiones no son planificadas por lo que las nuevas versiones podrían no ser compatibles con las anteriores.(ATOS, 2009).

Dado que es importante contar en el CEIGE con una herramienta, que con sólo disponer de un navegador web los usuarios puedan interactuar con ella y obtener la información necesaria para el seguimiento de los requisitos cuando se necesite y frente a las limitaciones que OSRMT presenta, se hace necesario desarrollar una nueva herramienta, basada en software libre, capaz de darle seguimiento a los requisitos y que pueda brindar toda la información a los interesados inmediatamente después de la realización de un cambio en los requisitos, con lo que estará apoyando a la determinación de los artefactos afectados con este.

1.4 Modelo de desarrollo

El modelo de desarrollo a utilizar es el definido por el proyecto ERP-Cuba para la realización del sistema Cedrux. Para tomar esta decisión se tuvieron en cuenta las características que posee este modelo las cuales se enuncian a continuación (GESTIÓN, 2009):

- **Centrado en la arquitectura:** La arquitectura orienta las prioridades del desarrollo y resuelve las necesidades tecnológicas y de soporte para el desarrollo.
- **Orientado a componentes:** Las iteraciones son orientadas por el nivel de significancia arquitectónica de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.
- **Iterativo e incremental:** Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.
- **Ágil y adaptable al cambio:** El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas.

1.5 Tecnologías para la implementación

El marco de trabajo adoptado para la implementación de la herramienta es Sauxe. El mismo ha sido definido por el proyecto ERP-Cuba y constituye un paquete tecnológico para el desarrollo de soluciones. La decisión está basada en el hecho de que se encuentra actualmente en explotación y se le brinda soporte a todas las herramientas que lo conforman. De las herramientas que incluye el Sauxe se utilizarán las que se presentan a continuación (Cabrera, 2010).

Sistema gestor de bases de datos

PostgreSQL versión 8.3 es un gestor de bases de datos relacional orientado a objetos, libre y gratuito.

Presenta las siguientes propiedades (PostqreSQL, 2010):

- Atomicidad: Asegura la realización de una operación, por lo que ante un fallo del sistema, esta no queda a medias.
- Consistencia: Posibilita la ejecución de aquellas operaciones que no van a romper las reglas y

directrices de integridad de la base de datos.

- Aislamiento: Mediante un sistema de acceso concurrente multiversión (*Multiversion concurrency control, MVCC)* asegura que una operación no pueda afectar a otras, de esta manera dos transacciones sobre la misma información no genera error.

Lenguaje de programación

El pre-procesador de hipertextos (PHP) es un lenguaje sencillo, de sintaxis cómoda y dispone de muchas librerías que facilitan en gran medida el desarrollo de las aplicaciones (php, 2010).

Características de PHP:

- Dispone de una conexión propia a varios sistemas de base de datos como: MySQL, PostgreSQL y Oracle.
- Incorpora bibliotecas que contienen funciones integradas para realizar útiles tareas relacionadas con la web.
- Es un producto de código abierto, soportado por una gran comunidad de desarrolladores que se encargan de encontrar y reparar los fallos de funcionamiento.
- Es un lenguaje multiplataforma.
- Permite las técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables.
- Posee tratamiento de errores.

De acuerdo con las facilidades expuestas anteriormente, PHP 5.2.4 será el lenguaje de programación en que se desarrollará la herramienta que propone el presente trabajo de diploma.

Zend Framework

Es un marco de trabajo de código abierto que está diseñado para PHP 5. Dentro de sus principales características están (Company, 2010):

- Módulos para manejar archivos en formato de documento portátil, canales de sindicación de noticias y servicios web.
- Incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para

consultar la base de datos.

- Completa documentación y pruebas de alta calidad.

Doctrine Framework

Doctrine Framework es un potente y completo sistema de mapas de relaciones de objetos (Object Relational Mapper, ORM por sus siglas en inglés) para PHP 5.2 ó superior con una base de datos con capas de abstracción incorporada. Brinda la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir, convertir clases a tablas de una base de datos. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos debido a que doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente (Doctrine, 2010).

ExtJS Framework

ExtJS Framework facilita las herramientas necesarias para la creación de aplicaciones web con excelentes gráficos, ya que posee una considerable colección de elementos para el diseño de interfaces: ventanas, pestañas, menús, tablas, entre otros. Brinda soporte para (Framework, 2009):

- Construir interfaces gráficas complejas y dinámicas.
- Comunicar datos de forma asíncrona con el servidor.
- Diversos navegadores como: Internet Explorer, Firefox, Safari y Opera.

Zend Studio para eclipse

Es el IDE para PHP más potente, ofreciendo al desarrollador de PHP la potencia de Zend Studio y el soporte multilenguaje de Eclipse y su enorme conjunto de extensiones (Zend, 2008).

Características de Zend Studio para Eclipse:

- No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- Soporte para PHP 4 y PHP 5.
- Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.

- Inserción automática de paréntesis y corchetes de cierre.
- Detección de errores de sintaxis en tiempo real.
- Integración mejorada con Zend Framework
- Integración con Zend Platform

Mozilla Firefox

Mozilla Firefox es un navegador de Internet desarrollado por la Corporación Mozilla. Se basa en el motor de renderizado Gecko, el cual se encarga de procesar el contenido de las páginas Web, desarrolladas en su mayor parte utilizando el lenguaje C++.
Incorpora bloqueo de ventanas emergentes, navegación por pestañas, marcadores dinámicos, compatibilidad con estándares abiertos, y un mecanismo para añadir funciones mediante extensiones (MozillaES, 2010).

1.6 Especificaciones de la arquitectura

Un elemento clave en el desarrollo del proceso de software es el diseño de la arquitectura, sobre ella se apoyan todas las representaciones de la estructura general de la aplicación a desarrollar, la misma es la que le da forma al software para que soporte todos los requisitos para alcanzar los objetivos y necesidades del sistema. La arquitectura proporciona una visión global del sistema a construir. Describe la estructura y la organización de los componentes del software, sus propiedades y las conexiones entre ellos (Reynoso, 2004).

Para el desarrollo de la herramienta se adopta la arquitectura definida por el proyecto ERP Cuba para el Cedrux, es una arquitectura basada en componentes que implementa el patrón Modelo-Vista-Controlador. Esta decisión está basada en la necesidad de que el sistema a crear sea escalable por lo que una arquitectura basada en componentes sentaría las bases para que se puedan incorporar otros componentes que implementen nuevas funcionalidades y cuando una nueva versión del sistema esté disponible, se puede remplazar la versión existente sin impacto en otros componentes.

1.6.2 MVC

Para la arquitectura de sistema, los componentes presentan una estructura, elaborada a partir del framework con que se trabaja (ExtJS, Zend, Doctrine) y de las necesidades del diseño. El elemento de

peso que pone el framework es la implementación del estilo o patrón arquitectónico Modelo-Vista-Controlador (MVC) (Codorníu, et al., 2009).

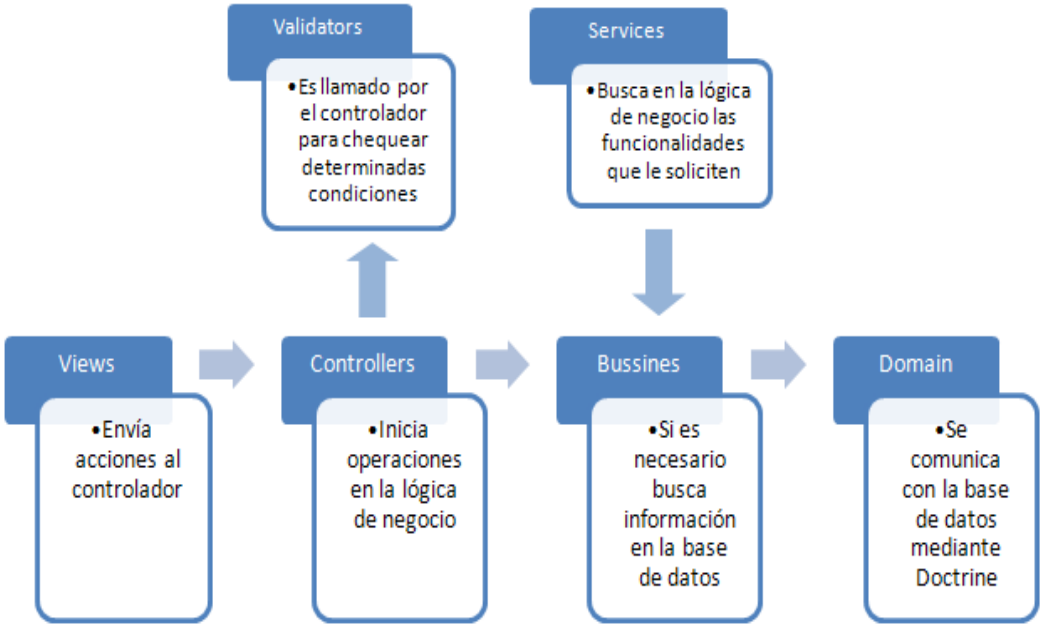


Figura 1. Flujo de trabajo dentro de un componente

Controllers: Contiene las clases controladores del componente. Son las que representan el Controller en el MVC. Su responsabilidad principal es comunicar las interfaces de usuario con la lógica de negocio de la aplicación.

Model:

- **Bussines**: Contiene las clases que implementan la lógica de negocio de la aplicación. Representa la capa de presentación.
- **Domain:** contiene las clases del dominio que heredan de las clases base. Representa la capa de datos de la aplicación.
    - **Generated**: contiene las clases que representan las tablas en la base de datos, es lo que se conoce como el mapeo de la base de datos.

Views:

- **Script:** Contiene los ficheros phtml, páginas de la aplicación.

- **Js:** Contiene los ficheros js incluidos dentro de cada phtml y que tienen toda la implementación de las interfaces de usuario.
- **Css:** Contiene estilos particulares del componente.

Validation: Contiene las clases que implementan las validaciones necesarias en cada uno de los escenarios del componente.

Estas se implementan aparte para separarlas de la lógica de negocio y se ejecutan a partir del mecanismo de AOP (Programación Orientada a Aspectos) del framework, que garantizan la posibilidad de configurar las validaciones que se deseen en cada acción de la aplicación.

Services: Contiene las clases de servicio, es la interfaz que brinda el componente al exterior, aquí se implementan todas las funcionalidades que el componente es capaz de brindar.

1.7 Pruebas de Software

Las pruebas de software son un elemento crítico para la garantía de calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación. La creciente percepción del software como un sistema y la importancia de los costos asociados a un fallo de este, están motivando a la creación de pruebas minuciosas y bien planificadas (Pressman, 2005).

El método de prueba que se va a aplicar al sistema es el de caja negra el cual se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Este tipo de prueba permite obtener un conjunto de condiciones de entrada que ejerciten de forma completa todos los requisitos funcionales de un programa, es un enfoque complementario que intenta descubrir diferentes tipos de errores como por ejemplo: errores de interfaz, errores en estructuras de datos o acceso a las bases de datos, entre otros (Pressman, 2005).

1.8 Complejidad ciclomática

La Complejidad ciclomática es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. La métrica, propuesta por Thomas McCabe en 1976, se basa en la representación gráfica del flujo de control del programa. De dicho análisis se desprende una medida cuantitativa de la dificultad de prueba y una indicación de la fiabilidad final. Cuando se utiliza en el

contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y proporciona el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. Se ha medido un gran número de programas, con el fin de establecer rangos de complejidad que ayuden al ingeniero de software a determinar la estabilidad y el riesgo inherente de un programa. La medida resultante puede ser utilizada en el desarrollo, mantenimiento y reingeniería para estimar el riesgo, costo y estabilidad. Algunos estudios experimentales indican la existencia de distintas relaciones entre la métrica de McCabe y el número de errores existentes en el código fuente, así como el tiempo requerido para encontrar y corregir esos errores (RIZZI, 2009).

Se suele comparar la complejidad ciclomática obtenida contra un conjunto de valores límite como se observa en la figura 2.

Complejidad Ciclomática	Evaluación de Riesgo
1-10	Programa simple, sin mucho riesgo
11-20	Más Complejo, riesgo moderado
21-50	Complejo, programa de alto riesgo
50	Programa no testeable, muy alto riesgo

Figura 2. Complejidad ciclomática contra evaluación de riesgo

Para conocer la complejidad del algoritmo es necesario calcular la complejidad ciclomática del mismo, para hacer dicho cálculo es necesario primero tener el código o el diseño del algoritmo, luego enmarcar cada instrucción del código con un número, que representa cada lugar del camino que puede seguir la secuencia del algoritmo (RIZZI, 2009).

Después de este paso, es necesario representar el grafo de flujo asociado, en el cual se representan distintos componentes como son los círculos que se denominan NODOS y representa una o más sentencias procedimentales. Las flechas se llaman ARISTAS y representan flujo de control. Una arista debe terminar en un nodo, aún cuando éste no represente ninguna sentencia procedimental. Las áreas delimitadas por aristas y nodos se denominan REGIONES (RIZZI, 2009).

Seguidamente a la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código, el cálculo es necesario efectuarlo mediante tres vías, para concluir que fueron correctos es necesario que el resultado sea el mismo, las fórmulas para calcular son las siguientes:

- V (G) = A – N + 2

Donde A es el número de aristas en el grafo, N es el número de nodos. V se refiere al número ciclomático en teoría de grafos y G indica que la complejidad es una función del grafo.

- V (G) = P + 1

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

- V (G) = R

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo (RIZZI, 2009).

1.9 Conclusiones parciales

En el CEIGE, del modo que se realiza el seguimiento de los requisitos, dificulta el análisis de su trazabilidad y la transmisión de la información a las partes implicadas, pues no se cuenta con una herramienta automatizada que facilite esta actividad.

Se pudo constatar tras el estudio realizado en este capítulo que las herramientas disponibles para el seguimiento de los requisitos no son las más adecuadas para usar en el CEIGE. Por lo que se hace necesario implementar una nueva herramienta para el seguimiento de los requisitos.

Luego de seleccionar el modelo de desarrollo a seguir y concretar algunos aspectos sobre arquitectura de software, están creadas las condiciones para la implementación de la herramienta SeReq la cual automatizará el seguimiento de los requisitos en el CEIGE.

Capítulo 2: Descripción y análisis de la solución propuesta

2.1 Introducción

En este capítulo, se realiza una profunda valoración de los artefactos propuestos por el analista de sistema y se delimitan varios puntos importantes en el desarrollo de la herramienta como:

- Se describen las principales clases utilizadas en la implementación del mismo.
- Se describen los estándares de codificación para una mejor legibilidad del código.
- Se detalla la estructura de la herramienta.

2.2 Valoración del análisis y diseño propuesto por el analista

Del análisis propuesto se obtuvo el modelo conceptual, que permitió aumentar la comprensión del problema y contribuir a esclarecer la terminología o nomenclatura del dominio. Ver en el anexo 1 el modelo conceptual propuesto por el analista.

Durante la captura de requisitos del sistema a implementar, el analista identificó una serie de funcionalidades como se muestra en la tabla 1.

Tabla 1. Funcionalidades del subsistema

Agrupación de requisitos	Requisitos funcionales
Gestionar tipo de elemento de seguimiento.	• Adicionar tipo de elemento de seguimiento. • Modificar tipo de elemento de seguimiento. • Eliminar tipo de elemento de seguimiento. • Buscar tipo de elemento de seguimiento.

Gestionar tipo de atributo.	• Adicionar tipo de atributo. • Modificar tipo de atributo. • Eliminar tipo de atributo. • Buscar tipo de atributo.
Gestionar tipo de relación entre elementos de seguimiento.	• Adicionar tipo de relación entre elementos de seguimiento. • Modificar tipo de relación entre elementos de seguimiento. • Eliminar tipo de relación entre elementos de seguimiento. • Buscar tipo de relación entre elementos de seguimiento.
Gestionar elemento de seguimiento.	• Adicionar elemento de seguimiento. • Modificar elemento de seguimiento. • Eliminar elemento de seguimiento. • Buscar elemento de seguimiento.
Gestionar atributo de elemento de seguimiento.	• Adicionar atributo de elemento de seguimiento. • Modificar atributo de elemento de seguimiento. • Eliminar atributo de elemento de seguimiento.
Gestionar relación entre elementos de seguimiento.	• Adicionar relación entre elementos de seguimiento. • Modificar relación entre elementos de seguimiento. • Eliminar relación entre elementos

	de seguimiento. • Buscar relación entre elementos de seguimiento.
Realizar consulta.	• Realizar consulta.
Determinar estabilidad.	• Determinar estabilidad.

Los requisitos incluidos en las Agrupaciones de requisitos: Gestionar tipo de elemento de seguimiento, Gestionar tipo de atributo, Gestionar tipo de relación, Gestionar elemento de seguimiento, Gestionar relación entre elementos de seguimiento y Gestionar atributo de elemento de seguimiento permiten el registro de los cambios en los requisitos y favorecen la transmisión de la información al respecto, lo cual ofrece solución al problema planteado en la introducción del presente trabajo de diploma.

Adicionalmente, la agrupación de requisitos Realizar consulta facilita información de la dependencia entre elementos que puede ayudar a determinar al equipo de trabajo los artefactos que se ven afectados con la realización de un cambio.

El analista propuso la funcionalidad Gestionar Personas que incluía los requisitos para adicionar, eliminar y modificar personas y asociarlas a un grupo de trabajo, estos requisitos se decidió no implementarlos porque esta no era una funcionalidad básica exigida en herramientas para el seguimiento de los requisitos. Con esta decisión se ve afectado el modelo de datos del cual se eliminan las tablas Dat_Persona, Dat_Equipo y Dat_persona_Dat_Equipo. Además se incluyó una nueva tabla nom_subsistemas que guarda la información relacionada con los subsistemas del proyecto que es gestionado con la herramienta.

Los prototipos de interfaz de usuario propuestos por el analista tienen una muy buena estructura y organización pero algunos de estos a la hora de la implementación fueron modificados para lograr una interfaz gráfica más amigable al usuario y se eliminaron algunos que podían estar incluidos en otros para mejorar de esta forma la navegación al trabajar con la herramienta. Los prototipos modificados fueron: Asociar/desasociar tipo de atributo, Adicionar tipo de elemento de seguimiento, Adicionar elemento de seguimiento, Adicionar relación entre elementos de seguimiento, Determinar métrica, Realizar consulta y la Matriz de trazabilidad. Fue eliminado el prototipo de Asociar y Desasociar que se incluyó en la interfaz de Adicionar tipo de elemento y las consultas según tipo de elemento y tipo de atributo se incluyeron en la interfaz Gestionar elemento de seguimiento.

Los requisitos no funcionales también son importantes porque estos son los que hacen al producto atractivo, usable, rápido y confiable. A continuación se muestran los requisitos no funcionales con los que debe contar la herramienta:

Apariencia o interfaz externa

El sistema debe poseer una interfaz amigable al usuario, basada en web, brindando facilidades que permitan interactuar y navegar con el sistema, de forma fácil y rápida. El sistema debe cumplir con los estándares de diseño establecidos en el proyecto ERP Cuba (UCID, 2008).

Seguridad

El servidor de bases de datos y el servidor de aplicación estarán instalados en computadoras diferentes. Cada usuario realizará operaciones en la aplicación en dependencia de sus privilegios o niveles de acceso.

Soporte

Se debe brindar capacitación a los usuarios de la herramienta para el adecuado uso y explotación de la misma.

Del diseño propuesto se pudieron extraer las clases fundamentales que deben ser definidas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, dando una idea clara de lo que se debe implementar.

El diseño fue creado siguiendo patrones, que de manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, permitiendo llevar a cabo la implementación clara y limpia del subsistema.

2.3 Propuesta del sistema

Al iniciar la sesión el sistema verificará los privilegios que posee el usuario registrado, permitiendo acceder solo a las funcionalidades que tenga acceso y las distintas acciones definidas según el rol que desempeña.

Aparecerá un botón de inicio con las siguientes funcionalidades:

- **Seguridad:** Permite mantener la seguridad del software que es donde se le da permiso a los usuarios para acceder de acuerdo a su rol a las funcionalidades correspondientes en el sistema. En el caso de un usuario administrador que podría ser un analista principal tendría acceso a las funcionalidades de configuración ya que solo un administrador puede configurar en el sistema y a las demás funcionalidades tienen acceso todos los usuarios.
- **Configuración:** Solo pueden acceder los usuarios con permisos de administración del sistema, las funcionalidades que incluye posibilitan registrar los diferentes tipos de elementos y asociarle los tipos de atributos determinados para después poder registrar elementos de seguimiento. Ver la figura 3 que se presenta a continuación.

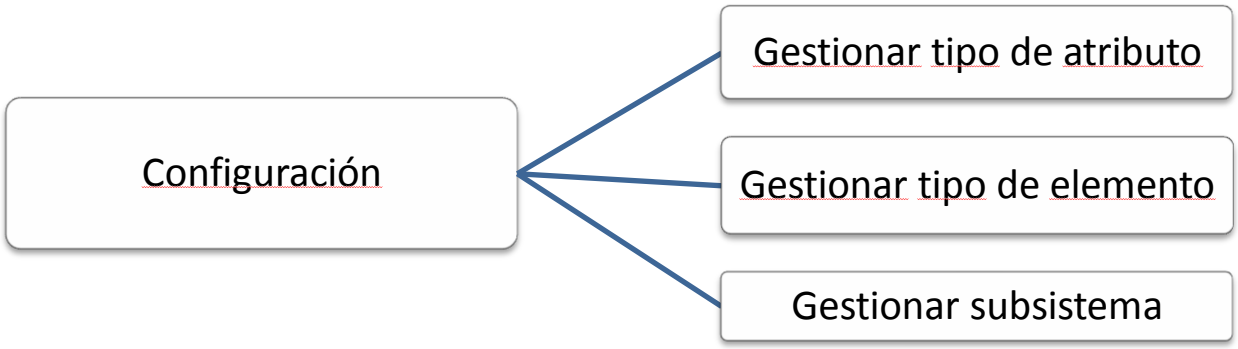


Figura 3. Mapa de navegación que presenta las funcionalidades de configuración de la herramienta SeReq

- *Gestionar tipo de atributo:* Permite adicionar, modificar y eliminar un tipo de atributo, que después se asocia a un tipo de elemento de seguimiento.
- *Gestionar tipo de elemento:* Permite adicionar, eliminar y modificar los tipos de elementos que se necesiten en el proyecto.
- *Gestionar subsistema:* Permite adicionar, modificar y eliminar los subsistemas existentes en el proyecto para poder ubicar los elementos en el subsistema al que pertenece y así poder obtener una información más específica de los mismos en caso que se desee.

- **Funcionalidades SeReq:** Se encuentran las funcionalidades concretas del seguimiento de los requisitos. Se pueden hacer consultas por criterios de búsqueda y conocer el estado de un elemento de seguimiento y además determinar los artefactos que se pueden afectar con un

cambio. Ver la figura 4 que se presenta a continuación.

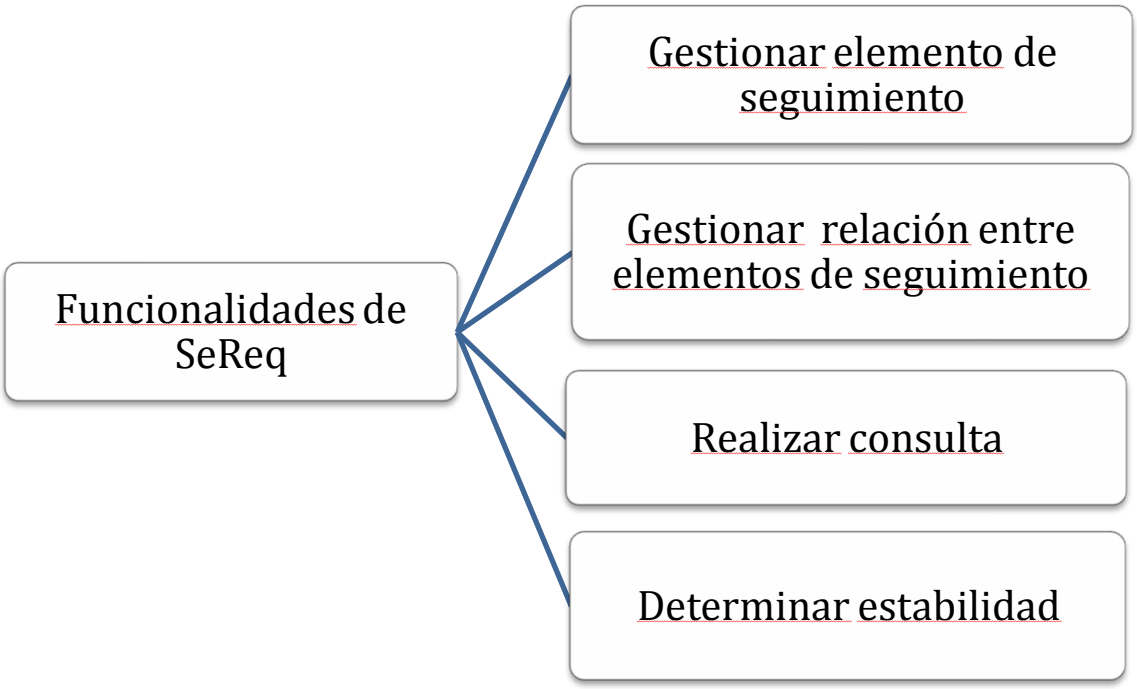


Figura 4. Mapa de navegación con las funcionalidades comunes para todos los usuarios de la herramienta SeReq

- *Gestionar elemento de seguimiento:* Que permite al usuario adicionar, eliminar y modificar un elemento de seguimiento en dependencia del tipo de elemento que sea y se le puede dar valor a los tipos de atributos que se le asociaron. Además brinda información acerca de la trazabilidad de los requisitos o elementos de seguimiento que estén registrados en el sistema a los usuarios a través de los criterios de búsqueda que se presentan.
- *Gestionar relación entre elementos de seguimiento:* Se establecen las relaciones de dependencia entre elementos de seguimiento.
- *Realizar consulta:* Permite hacer consultas brindando información de la dependencia que pueden tener un grupo de tipos de elementos determinados con otros, para en caso de un cambio poder facilitar la determinación de los elementos afectados con este cambio. La consulta muestra una matriz de trazabilidad con los tipos de elementos que se seleccionen anteriormente para las filas y para las columnas y sus relaciones.

- *Determinar estabilidad:* Permite conocer la estabilidad de un grupo determinado de requisitos que pertenecen a un determinado subsistema del proyecto o de todos los requisitos del proyecto.

2.4 Análisis de reutilización y utilización de componentes, código o módulos

A continuación se muestran los componentes que son utilizados para la implementación la herramienta SeReq.

- El subsistema Acaxia, se utiliza para controlar los roles y usuarios con permisos para acceder a las funcionalidades. De esta forma se puede permitir o denegar el acceso tan profundamente como se requiera y quedaría definido un sistema de administración centralizado de seguridad que puede ser configurado para una política restrictiva tan severa como se desee. Cada usuario tendrá su perfil y podrá configurarlo según entienda. El sistema permite la administración de conexiones ya que cada sistema puede configurar a qué servidores, gestores, bases de datos y esquemas conectarse, el usuario también tiene su usuario de base de datos para conectarse a esta y realizar las acciones que este usuario tenga asignadas (Seguridad, 2010).
- Sauxe se utiliza como marco de trabajo para facilitar el trabajo a la hora de la implementación porque tiene las tecnologías, herramientas y framework ya muy bien definidos y además contiene un conjunto de componentes implementados y sus interfaces bien definidas, estos componentes se pueden utilizar y redefinir.

2.5 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos. A partir de los requisitos definidos y las principales funcionalidades identificadas, el sistema se estructuró en 2 componentes siguiendo los lineamientos de la arquitectura del proyecto ERP Cuba. En la figura 5 se muestra la aplicación SeReq dividida en componentes que facilitan su organización en partes más manejables.

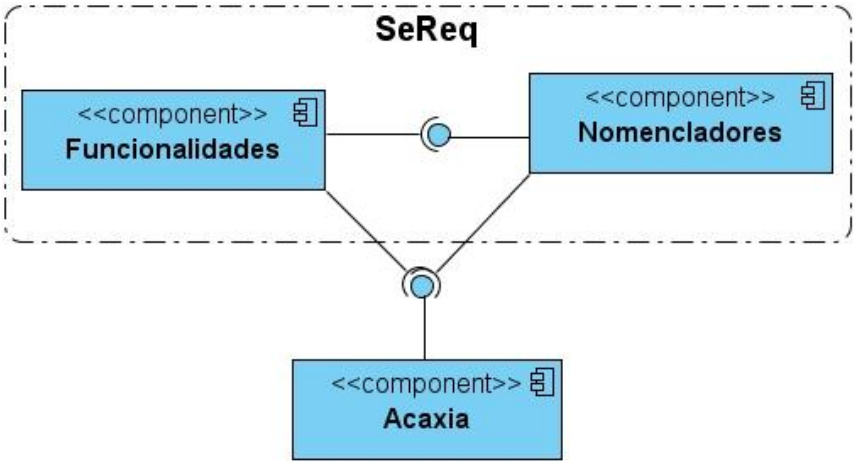


Figura 5. Diagrama de componentes de SeReq

El componente **Nomencladores** se encarga de la gestión de los objetos que manejan los usuarios de la herramienta y brinda un conjunto de funcionalidades a través de su interfaz *NomencladoresService.* Los servicios que este componente brinda son:

- obtenerTipoelementoPorID
- obtenerTipoelem
- obtenerRequisitos
- obtenerTipoatributoPorID
- obtenerEstabilidad
- obtenerSubsistemaPorID
- obtenerTiporelacionPorID
- listarRelaciones y cargarValoresValidos.

El componente **Funcionalidades** abarca las funciones que se pueden realizar con los objetos registrados en el sistema como la realización de consultas. No brinda servicios y consume todos los que brinda el componente Nomencladores.

En el diagrama se representa el componente Acaxia, que posibilita la gestión de la seguridad del sistema. Este no está incluido en la herramienta SeReq en sí, sino que ha sido implementado por el equipo de arquitectura del proyecto ERP Cuba y sus servicios son utilizados. Algunos de estos servicios son:

- Crear los roles.
- Crear usuarios a los cuales se le asigna un rol de los existentes.
- Asignar permiso a un determinado usuario de un rol a una acción dentro de una funcionalidad del sistema.
- Asignar permiso a un determinado usuario de un rol a una funcionalidad del sistema.

2.5.1 Integración entre componentes

La comunicación entre los elementos del MVC de cada componente se realiza mediante llamadas a métodos o eventos de forma directa.

Entre diferentes módulos y componentes, la integración se basa en el patrón Inversión de Control (IoC) y se realiza a través de un componente incluido en el framework, que permite operar sobre distintos esquemas en la base de datos realizando las transacciones adecuadas. En dicho componente se define el fichero ioc.xml que contiene la ubicación de cada uno de los componentes y los servicios que ofrecen las clases services correspondientes. De esta manera, la puerta de entrada de cada componente es el paquete de las clases de servicios que buscan en los modelos o entidades las funcionalidades requeridas.

Con la integración se persigue obtener una forma eficiente y flexible de combinar recursos internos o externos de los componentes o subsistemas usando:

- IoC Interno: para la integración entre componentes de un subsistema.
- IoC Externo: para la integración entre subsistemas.

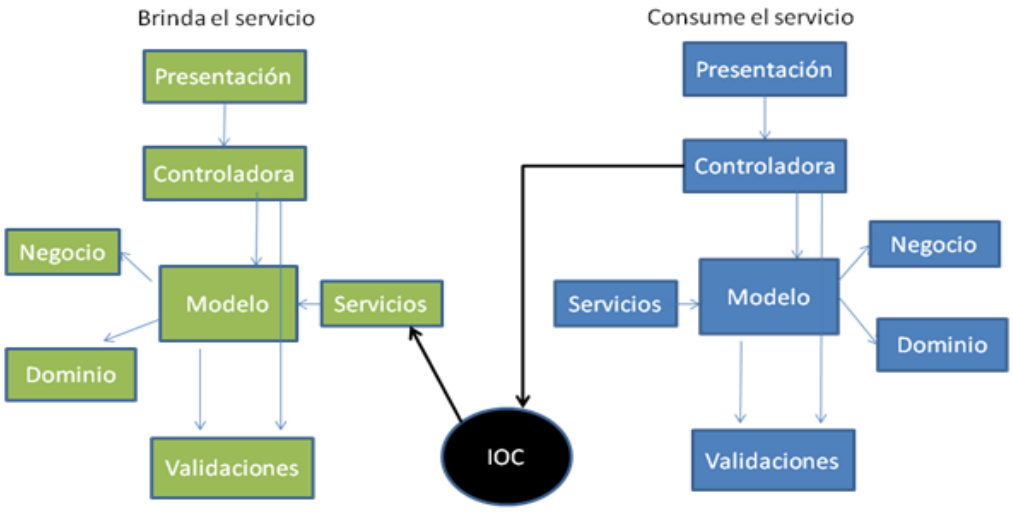


Figura 6. Integración entre componentes

2.6 Modelo de datos

El modelo de datos como se dijo en el epígrafe 2.2 Valoración del análisis y diseño propuesto por el analista fue modificado, actualmente este cuenta con un total de 10 tablas de las cuales 4 son nomencladores.

- nom_tipoatributo: Tabla que guarda la información relacionada con los tipos de atributos existentes en el sistema.
- nom_tipoatributo_nom_tipoelemento: Tabla que contiene la información de la relación entre las tablas nom_tipoatributo y nom_tipoelemento.
- nom_tipoelemento: Tabla que guarda la información relacionada con los tipos de elementos de seguimiento existentes en el sistema.
- nom_tiporelacion: Tabla que guarda la información relacionada con los tipos de relaciones entre elementos de seguimiento existentes en el sistema.
- nom_subsistemas: Tabla que guarda la información relacionada con los subsistemas existentes en el sistema.
- dat_elemento: Tabla que guarda la información relacionada con los elementos de seguimiento existentes en el sistema.
- dat_atributo: Tabla que guarda la información relacionada con los atributos existentes en el

sistema.

- dat_relacion: Tabla que guarda la información relacionada con las relaciones entre elementos de seguimiento existentes en el sistema.
- dat_metrica: Tabla que guarda la información relacionada con las métricas existentes en el sistema.
- dat_formula: Tabla que guarda la información relacionada con las fórmulas existentes en el sistema.

En el anexo 2 se muestra el modelo de datos de SeReq.

2.7 Estándares de código

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código.

Los estándares de codificación en el marco del proyecto ERP Cuba establecerán las pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenimiento a lo largo del tiempo. El sistema a implementar se rige por estos estándares, pues son una guía para el desarrollo y la implementación desde el punto de vista arquitectónico estandarizan el código a implementar (UCID, 2008).

Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación Pascal Casing.

Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación Camel Casing, y con sólo leerlo se reconoce el propósito de la misma.

Nomenclatura de las variables

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación Camel Casing, y comenzando con un prefijo según el tipo de datos.

2.8 Descripción de las principales clases

En las siguientes tablas se muestran las principales clases de la implementación:

En la clase GestElemSegController del componente Funcionalidades es donde se programan los métodos que dan solución a las funcionalidades que permiten gestionar elementos de seguimiento. Ver en la tabla 2 la descripción de esta clase.

Tabla 2. Clase controladora GestElemSegController

Nombre: GestElemSegController	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
init()	Constructor de la clase.
cargarElementosAction()	Responsabilidad de devolver los elementos de seguimiento existentes.
avanzadaAction()	Es una búsqueda avanzada según el idtipoelemento y atributo.
cargarAtributosAction()	Responsabilidad de cargar dado un elemento de seguimiento sus atributos.
adicionarAtribAction()	Responsabilidad de adicionar los atributos asociados a un tipo de elemento.

adicionarAction()	Responsabilidad de adicionar los elementos de seguimiento.
cargarValoresValidosAction()	Responsabilidad de dado un tipo de atributo devolver sus valores validos.
cargarTipoAtribAction()	Responsabilidad de devolver los tipos de atributo dado de un tipo de elemento.
cargarComboTEAction()	Responsabilidad de devolver los tipos de elementos.
modificarAtribAction()	Modifica los tipos de atributos de un elemento de seguimiento.
eliminarAtribAction()	Elimina los atributos de un elemento de seguimiento.
eliminarAction()	Elimina elementos de seguimiento.
modificarAction()	Modifica un elemento de seguimiento.

En la clase GestRelEntreElemController del componente Funcionalidades es donde se programan los métodos que dan solución a las funcionalidades que permiten gestionar las relaciones entre elementos de seguimiento. Ver en la tabla 3 la descripción de esta clase.

Tabla 3. Clase controladora GestRelEntreElemController

Nombre: GestRelEntreElemController	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
init()	Constructor de la clase.
cargarRelacionAction()	Cargar las relaciones existentes.
eliminarAction()	Responsabilidad de eliminar relaciones.
modificarAction()	Responsabilidad de modificar las relaciones.
adicionarAction ()	Responsabilidad de adicionar una nueva relación.
cargarcomboTRAction()	Responsabilidad de cargar en un combo los tipos de relaciones.

En la clase GestTipoAtribController del componente Nomencladores es donde se programan los métodos que dan solución a las funcionalidades que permiten gestionar los tipos de atributos para los elementos de seguimiento. Ver en la tabla 4 la descripción de esta clase.

Tabla 4. Clase controladora GestTipoAtribController

Nombre: GestTipoAtribController	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**

init()	Constructor de la clase.
cargarTipoAtribAction()	Responsabilidad de cargar los tipos de atributos en un GridPanel.
adicionarAction()	Responsabilidad de adicionar un tipo de atributo.
eliminarAction()	Responsabilidad de eliminar uno o varios tipos de atributos.
modificarAction()	Responsabilidad de modificar un tipo de atributo.

En la clase RealizarConsultaController del componente Funcionalidades es donde se programan los métodos que dan solución a las funcionalidades que permiten realizar consultas sobre los elementos de seguimiento. Ver en la tabla 5 la descripción de esta clase.

Tabla 5. Clase controladora RealizarConsultaController

Nombre: RealizarConsultaController	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
init()	Constructor de la clase.
cargarComboFilasAction()	Responsabilidad de cargar los tipos de elementos en el combo de las filas.
cargarComboColumnAction()	Responsabilidad de cargar los tipos de elementos en el combo de las columnas.

cargarmatrizAction()	Responsabilidad de cargar la matriz de trazabilidad.

En la clase DatElemento del componente Funcionalidades es donde se realizan las consultas pertinentes para obtener información de la base de datos de los elementos de seguimiento. Ver en la tabla 6 la descripción de esta clase.

Tabla 6. Clase modelo del dominio DatElemento

Nombre: DatElemento	
Tipo de clase: Entidad	
Para cada responsabilidad:	
Nombre:	**Descripción:**
setUp()	Constructor de la clase.
eliminarElementos($idelemento)	Eliminar elementos de seguimiento.
busquedaAvanzada ($limite = 10,$inicio = 0 ,$idtipoelemento, $idtipoatributo, $atributo)	Busca un elemento de seguimiento determinado dado uno de los tres criterios de búsqueda determinado.
contar ()	Devuelve la cantidad de elementos de seguimiento.
getElementos ($limite = 10,$inicio = 0)	Devuelve elementos de seguimiento y tipos de elementos.
existeCodigo($codigo)	Es un boolean que devuelve true si el código pasado por parámetros existe.
existeNombre($nombre)	Es un boolean que devuelve true si el

	nombre pasado por parámetros existe.

En la clase DatRelacion del componente Funcionalidades es donde se realizan las consultas pertinentes para obtener información de la base de datos de las relaciones de los elementos de seguimiento. Ver en la tabla 7 la descripción de esta clase.

Tabla 7. Clase modelo del dominio DatRelacion

Nombre: DatRelacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
idelemento1, idelemento2, idtipoelemento, idrelacion	varchar, date, varchar, varchar, numeric, varchar, numeric, numeric
Para cada responsabilidad:	
Nombre:	**Descripción:**
setUp()	Constructor de la clase.
eliminar($idrelacion)	Eliminar las relaciones que se desee entre elementos de seguimiento.
cargarcombo()	Devuelve las relaciones.
Buscar($idelemento1)	Busca un elemento de seguimiento.
BuscarElemento ($limite = 20, $start =0, $pTipoRelacion, $deElem)	Busca los elementos que tengan un tipo de relación determinado.

existeRelacion($e1, $e2)	Es un boolean que devuelve true si existe la relación entre los dos elementos de seguimiento que se le pasa por parámetros.
mostrarRelacion($id1,$id2)	Muestra los nombres de los tipos de relación entre dos elementos.

2.9 Algoritmos no triviales

Dentro de la solución existen algoritmos que resuelven problemas específicos de gran significancia en el cumplimiento del objetivo que se persigue con la implementación de la herramienta. En este caso se encuentra el algoritmo que se encarga de obtener las relaciones entre todos los elementos de un tipo con todos los elementos de otro tipo que pueden ser seleccionados desde la interfaz (Realizar consulta) por el usuario y mostrarlas en una matriz de trazabilidad que al hacer coincidir fila con columna si tienen relación se muestra el tipo de relación que presentan. Esta información permite al usuario saber la dependencia entre elementos de seguimiento para en caso de que ocurra un cambio en uno de estos elementos saber que otros elementos se afectan también. Ver en la figura 7 la matriz de trazabilidad.

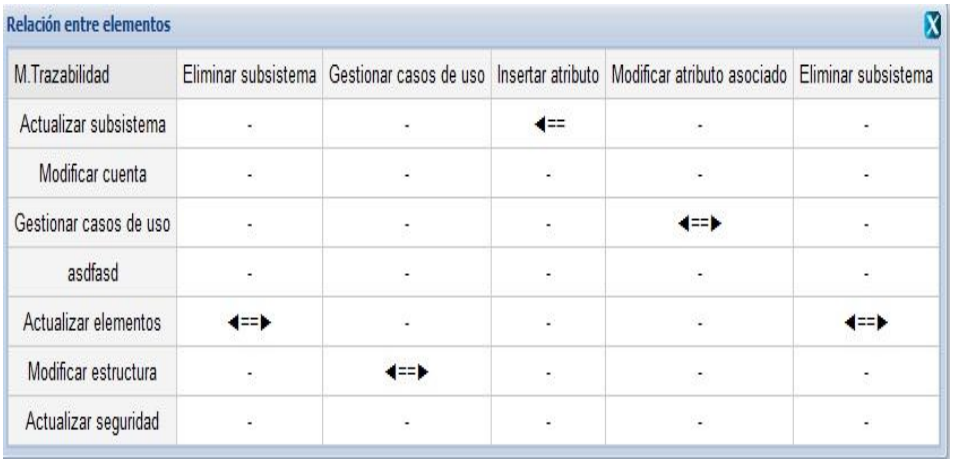


Figura 7. Matriz de trazabilidad

A continuación se presenta el algoritmo que permite esta funcionalidad al cual se le va a calcular la complejidad ciclomática.

1. Se obtiene desde la base de datos los dos arreglos de los tipos de elementos que se seleccionaron en la interfaz realizar consulta.
2. Después se compara si alguno de los arreglos está vacío muestra un mensaje diciendo que no hay elementos de ese tipo para poner en la fila o la columna.
3. En caso de que los dos arreglos tengan elementos se compara cada elemento del arreglo fila con todos los del arreglo columna y si tienen relación manda el elemento de la fila, el de la columna y el nombre de la relación para mostrarlo en la matriz, en caso de que no tengan relación manda el carácter ¨-¨ para mostrarlo de esa forma en la matriz.

```
function cargarmatrizAction()
{//1
    $idtipoelemfilas = $this->_request->getPost ( 'idtipoelemento' );//1
    $idtipoelemcolumn = $this->_request->getPost ( 'id' );//1
    $fila = NomTipoelemnto::getElemento ($idtipoelemfilas);//1
    $colum = NomTipoelemnto::getElemento ($idtipoelemcolumn);//1

    if(count($fila) == 0 || count($colum) == 0) //2
    {
        $matriz = array('columna'=>1,'fila'=>1,'datos'=>$datos);//3

    }//4
    else //5
    {
        $relacion = DatRelacion :: mostrarRelacion($fila[0]['idelemento'],$colum[0]['idelemento']);//6
        for($i=0; $i<count($fila); $i++) //7
        {
            for($j=0; $j<count($colum); $j++) //8
            {
                $relacion = DatRelacion :: mostrarRelacion($fila[$i]['idelemento'],$colum[$j]['idelemento']);//9

                $datos[$i][$j]= $relacion;//10
            }//11
        }//12

        $matriz = array('columna'=>$colum,'fila'=>$fila,'datos'=>$datos);//13
    }//14
    echo json_encode ( $matriz );//15

}//16
```

Figura 8. Código del algoritmo cargarmatrizAction()

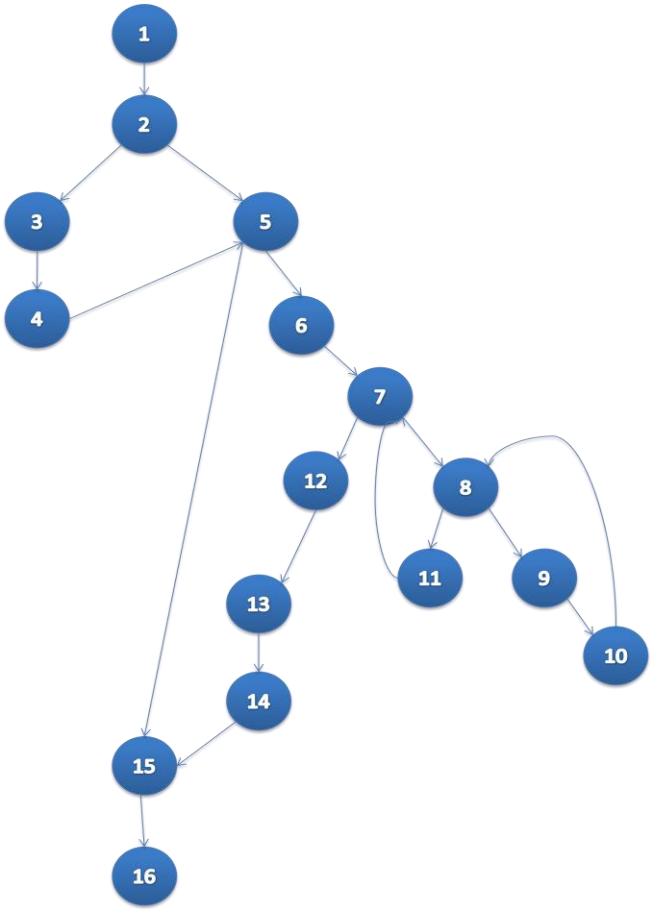


Figura 9. Grafo de flujo del algoritmo presentado

Seguidamente a la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código por las tres fórmulas pertinentes.

Por la fórmula 1:

- V (G) = A – N + 2

  V (G) = (19-16) + 2

  V (G) = 5

Por la fórmula 2:

- V (G) = P + 1

  V (G) = 4+1

V (G) = 5

Por la fórmula 3:

- V (G) = R

  V (G) = 5

Realizado el cálculo por las tres vías necesarias se llega a la conclusión de que el algoritmo presentado anteriormente tiene una complejidad ciclomática de 5, dando una visión de que existen a lo sumo cinco caminos lógicos por donde recorrer el algoritmo. Al tener una complejidad 5, la evaluación de riesgo es de un algoritmo sin mucho riesgo. A pesar de ser un algoritmo sin mucho riesgo el valor que aporta como funcionalidad es de vital importancia en una herramienta para el seguimiento de los requisitos.

2.10 Conclusiones parciales

Con el desarrollo de este capítulo se obtuvo la solución de la herramienta SeReq para el seguimiento de los requisitos con el objetivo de facilitar el modo en que se hace el seguimiento de los requisitos en el CEIGE y transmitir la información sobre la realización de un cambio en un elemento de seguimiento a las partes implicadas.

La valoración del diseño propuesto por el analista fue muy importante pues favoreció en gran medida a la correcta conformación de los componentes a implementar para resolver el problema planteado.

Se garantizó además, a partir de la arquitectura propuesta, el intercambio de información entre los componentes existentes y se sentaron las bases para la incorporación de nuevos componentes que incluyan nuevas funcionalidades que mejoren en un futuro la herramienta.

Se realizó un análisis de los estándares de codificación lo que permitió lograr una mayor uniformidad en el código y se hizo una breve descripción de algunas de las clases más importantes del sistema que permite ver internamente las funcionalidades que estas brindan.

Capítulo 3: Validación de la solución propuesta.

3.1 Introducción

Durante el proceso de desarrollo de un software, los errores pueden comenzar a aparecer incluso desde el mismo momento en que fueron definidos los objetivos y estos a su vez especificados de forma errónea e imperfecta; de la misma forma en los posteriores pasos del diseño y desarrollo. A raíz de la incapacidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software debe ser acompañado de una actividad que garantice su calidad.

En el desarrollo de este capítulo se valida mediante pruebas si con el software realizado se ha cumplido el objetivo planteado con la calidad requerida.

3.2 Aplicación de pruebas de caja negra

Para el desarrollo de la prueba se utilizará el requisito Adicionar elemento de seguimiento que está dentro de gestionar elemento de seguimiento. El objetivo del mismo es persistir en la base de datos la información referente a los elementos de seguimiento que se han creado en la entidad dando la posibilidad de modificarla y eliminarla.

Condiciones de ejecución

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Debe existir al menos un tipo de elemento de seguimiento.
- Se debe seleccionar la opción Funcionalidades de SeReq.
- Se debe seleccionar la opción **Gestionar elemento de seguimiento.**

En la tabla 8 se muestra el requisito a probar.

Tabla 8. Requisito a probar

Nombre del requisito	Descripción general	Escenarios de prueba	Flujo del escenario
1: Adicionar Elemento de seguimiento.	El sistema permite adicionar elementos de seguimiento, persistiendo la información en la base de datos y buscarla en caso necesario.	**EP: 1.1** Adicionar un elemento de seguimiento introduciendo los datos correctamente al presionar el botón **Aceptar**.	Se presiona el botón **Adicionar**. Se introducen correctamente lo datos. Se presiona el botón **Aceptar.** Se muestra la notificación: "El elemento se adicionó satisfactoriamente.". Se presiona el botón **Aceptar** de la notificación. Se muestra el GridPanel con el nuevo elemento.
		EP: 1.2 Adicionar un elemento de seguimiento introduciendo datos inválidos al presionar el botón **Aceptar.**	Se presiona el botón **Adicionar.** Se introducen los datos correspondientes insertando algunos datos inválidos. Se presiona el botón **Aceptar.**

	EP 1.3: Adicionar un elemento de seguimiento dejando campos requeridos en blanco al presionar el botón **Aceptar.**	Se presiona el botón **Adicionar**. Se introducen los datos correspondientes en el formulario y deje al menos un campo requerido en blanco. Se presiona el botón **Aceptar**.
	EP 1.4: **Cancelar.**	Se presiona el botón **Adicionar.** Se introducen o no los datos en los campos. Se presiona el botón **Cancelar.**

En la tabla 9 se muestra la descripción de las variables.

Tabla 9. Descripción de los datos que tiene la ventana del requisito Adicionar Elemento de seguimiento

No	Nombre de campo	Tipo	Válido	Inválido
1	Código.	Campo de texto.	En este campo de texto puede escribirse cualquier tipo de letra, número o carácter extraño.	Vacío.
2	Nombre.	Campo de texto.	En este campo de texto puede escribirse solamente letras.	Vacío.

3	Tipo elemento.	Combo box.	Se selecciona uno de los tipos de elementos.	Vacío.
4	Descripción.	Campo de texto.	En este campo de texto puede escribirse cualquier tipo de letra, número o carácter extraño.	NA
5	Autor	Campo de texto.	En este campo de texto puede escribirse solamente letras.	Vacío
6	Fecha	Campo fecha.	Se selecciona una de las fechas que muestra el componente y no se puede escribir nada en este campo que no sea lo que en él aparece.	Vacío

En la siguiente tabla se muestran los juegos de datos a probar y los resultados obtenidos para estos.

Tabla 10. Juegos de datos a probar y los resultados obtenidos para estos

Id del escenario	Escenario	Código	Nombre	Tipo de elemento	Autor	Fecha	Descripción	Respuesta del sistema	Respuesta de la prueba
EP 1.1	Adicionar un elemento de seguimiento	V(R1)	V(Adicionar Elemento)	V(Requisito Funcional)	V(Yuri)	V(10/06/2010)	V(Probando)	Un mensaje que diga ¨El elemento se adicionó satisfactoriamente¨	Mostró el mensaje ¨ El elemento se adicionó satisfactoriamente ¨

	introduciendo datos válidos.	V(R2)	V(Eliminar elemento)	V(Requisito funcional)	V(Yuri)	V(09/06/2010)	V(Vacío)	Un mensaje que diga ¨El elemento se adicionó correctamente¨	Mostró el mensaje ¨El elemento se adicionó correctamente¨
EP 1.2	Adicionar un elemento de seguimiento dejando campos requeridos en blanco.	I(Vacío)	V(Nomencladores)	V(Componente)	V(Yuri)	V(10/06/2010)	V(Vacío)	Un mensaje que diga ¨Por favor verifique que hay valor(es) incorrecto(s) ¨ Se muestran los campos de texto en rojo y un mensaje indicando que esos campos son obligatorios. En el caso del campo Fecha, el sistema	Mostró el mensaje ¨ Por favor verifique que hay valor(es) incorrecto(s) ¨
		V(AR3)	I(Gestionar relaciones)	V(Agrupación de requisitos)	V(Yuri)	V(10/06/2010)	V(Vacío)		
		V(R4)	V(Adicionar personas)	I(Vacío)	V(Yuri)	V(10/06/2010)	V(Vacío)		
		V(R5)	V(Modificar personas)	V(Requisito funcional)	I(Vacío)	V(10/06/2010)	V(Prueba)		
		V(AR6)	V(Gestionar Elementos)	V(Agrupación de requisitos)	V(Yuri)	I(Vacío)	V(Vacío)		

								debe mostrar por defecto la fecha del día en que se intenta realizar la inserción y tener deshabilitados los días después del actual para que no se puedan escoger.	
EP 1.3	Adicionar un elemento de seguimiento introduciendo errores en los datos.	I(R;)	V(Gestionar personas)	V(Agrupación de requisitos)	V(Yuri)	V(10/06/2010)	V(Vacío)	Un mensaje que diga ¨Por favor verifique que hay valor(es) incorrecto(s) ¨ Se muestran los campos de texto en rojo y un mensaje	Mostró el mensaje ¨ Por favor verifique que hay valor(es) incorrecto(s) ¨
		V(R7)	I(Requisito 22)	V(Requisito funcional)	V(Yuri)	V(08/06/2010)	V(Vacío)		
		V(R8)	V(Modificar elementos)	V(Requisito funcional)	I(Yuri 88)	V(07/06/2010)	V(prueba;)		

								indicando que esos campos son obligatorios.	
EP 1.4	Cancelar.	NA	NA	NA	NA	NA	NA	Se cancela la operación. Se cierra la ventana.	Se cierra la ventana.

De esta forma se le aplicaron las pruebas de caja negra a todas las funcionalidades del sistema. A continuación se resumen la cantidad de escenarios de prueba utilizados para cada funcionalidad, así como el número de no conformidades para los casos donde fueron encontradas estas. Los valores porcentuales que representan estos resultados se muestran en la figura 10 (Resultados obtenidos por escenarios al aplicar las pruebas de caja negra).

Funcionalidades en Gestionar tipo de elemento de seguimiento:

1. Adicionar tipo de elemento de seguimiento: 5 escenarios.
2. Modificar tipo de elemento de seguimiento: 4 escenarios.
3. Eliminar tipo de elemento de seguimiento: 4 escenarios.
4. Búsqueda: 3 escenarios.

Funcionalidades en Gestionar tipo de atributo de elemento de seguimiento:

1. Adicionar tipo de atributo de elemento de seguimiento: 5 escenarios.
2. Modificar tipo de atributo de elemento de seguimiento: 4 escenarios.
3. Eliminar tipo de atributo de elemento de seguimiento: 4 escenarios.
4. Búsqueda: 3 escenarios.

Funcionalidades en Gestionar subsistemas:

1. Adicionar subsistema: 5 escenarios.
2. Modificar subsistema: 4 escenarios.

3. Eliminar subsistema: 4 escenarios.
4. Búsqueda: 2 escenarios.

Funcionalidades en Gestionar elemento de seguimiento:

1. Adicionar elemento de seguimiento: 5 escenarios.
2. Modificar elemento de seguimiento: 4 escenarios.
3. Eliminar elemento de seguimiento: 4 escenarios.
4. Búsqueda: 4 escenarios.
5. Búsqueda avanzada: 8 escenarios, 4 no conformidades.
6. Adicionar atributo de elemento de seguimiento: 5 escenarios.
7. Eliminar atributo de elemento de seguimiento: 2 escenarios.
8. Modificar atributo de elemento de seguimiento: 4 escenarios.

Funcionalidades en Gestionar relación entre elementos de seguimiento:

1. Adicionar relación entre elementos de seguimiento: 5 escenarios.
2. Modificar relación entre elementos de seguimiento: 4 escenarios.
3. Eliminar relación entre elementos de seguimiento: 2 escenarios.
4. Búsqueda: 3 escenarios, 1 no conformidad.

Realizar consulta:

1. Realizar consulta: 4 escenarios de prueba.

Determinar Estabilidad:

1. Determinar estabilidad: 4 escenarios de prueba.

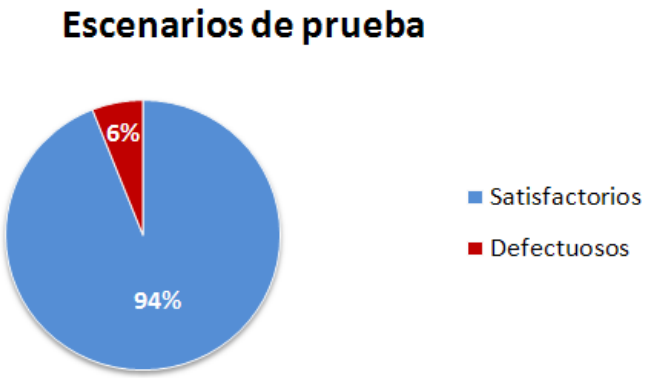


Figura 10. Resultados obtenidos por escenarios al aplicar las pruebas de caja negra

Como evidencian los resultados solo el 6 % de los escenarios de pruebas presentan no conformidades o sea son defectuosos.

Estas no conformidades fueron cubiertas para dejar en un 100% de funcionamiento el sistema.

3.3 Validación por especialista

Después de finalizada la herramienta se presentó ante un especialista que validara si las funcionalidades que esta presenta podían resolver el problema planteado. La especialista fue la ingeniera Tahirí Rivero Álvarez, precisamente porque es la analista principal del proyecto ERP Cuba actualmente y aquí fue donde se presentó la problemática planteada además es la persona que lleva el seguimiento de los requisitos en el proyecto y tiene conocimientos de las tareas que esta actividad conlleva. Su valoración fue la siguiente:

"La herramienta cumple con los requerimientos que establece la universidad para llevar el seguimiento de los requisitos" (Alvarez, 2010).

Para dar este criterio comparó las funcionalidades que brinda la herramienta con las establecidas por Calisoft para el seguimiento de los requisitos.

Elementos a tener en cuenta en el seguimiento de los requisitos según Calisoft (CALISOFT, 2009):

- Identificar elementos de seguimiento.
- Definir relaciones entre los elementos de seguimiento a los requisitos.
- Especificar los atributos de seguimiento de cada elemento.
- Registrar elementos de seguimiento a los requisitos.
- Definir las dependencias.
- Obtener información de seguimiento.

A continuación se presentan los **problemas que resuelve** la herramienta según la especialista:

- Permite mediante una búsqueda sencilla obtener indicadores como la estabilidad de los requisitos por subsistemas y de manera general en todo el proyecto, lo cual aunque no es imposible de llevar en un Excel, la realidad es que es mucho más complejo y engorroso para las personas encargadas de llevar el seguimiento de los requisitos.

- Permite contar con una herramienta personalizada acorde a las necesidades del proyecto, lo que implica que el responsable de llevar el seguimiento de los requisitos no tiene que desgastarse entendiendo las funcionalidades de una herramienta que no cumpla con sus expectativas.
- Permite mantener una trazabilidad de los requisitos en ambos sentidos, o sea hacia el negocio y hacia la implementación de estos en el sistema, lo cual era muy complejo de llevar en un Excel, donde además de que existían otros indicadores, existía un gran número de subsistemas.
- Sirve de apoyo a la toma de decisiones, pues ante una solicitud de cambios se pueden ver los elementos con los que se relaciona y decidir si se procede o no a realizar el cambio en la versión objetivo del requisito, según el impacto que requiere realizar el cambio. Como se llevaba en el ERP, esto no era posible de conocer.
- Permite hacer transparente para los que utilizan la herramienta los cálculos que se requieren para establecer el estado de algún indicador o algún reporte que se desee obtener.
- Permitirá tener de forma centralizada y pública toda la información referente a los requisitos de cada subsistema lo cual le permitirá al analista pueda consultar el estado de los requerimientos de los subsistemas con los que se relaciona su subsistema. Esto anteriormente era imposible conocer.
- Desde el punto de vista de gestión se puede identificar el subsistema que mayor estabilidad tuvo en los requisitos y por tanto identificar el mejor equipo de análisis.

Recomendaciones importantes hechas por la especialista

- Poder obtener la línea base de los requisitos por iteraciones.
- Incrementar el número de reportes.
    - Requisitos por procesos de negocio.
    - Requisitos por componentes.
    - Requisitos por subsistema.
    - Requisitos por complejidad.
    - Requisitos por prioridad.
    - Visualizar las relaciones existentes entre las necesidades de la organización y los requisitos

funcionales para verificar si todas las necesidades se satisfacen o el impacto de los errores en los requisitos y el producto de software.

- Porciento o estado de implementación de los requisitos.

- Adicionarle funcionalidades para graficar, relaciones y estado de avances.

La herramienta cumple además con lo establecido en el artefacto oficial que está definido para el seguimiento de los requisitos en el proyecto ERP Cuba (2009).

3.4 Conclusiones parciales

En este capítulo se hizo un análisis de los resultados de las pruebas de caja negra aplicadas al sistema a través de los diferentes casos de pruebas que cubrían las funcionalidades permitiendo encontrar errores que afectaban el funcionamiento de este. Los errores encontrados con estas pruebas fueron corregidos lo que permitió que actualmente el sistema esté al 100% de funcionamiento.

También se presentó la herramienta ante un especialista que dio una validación donde deja plasmado que esta cumple con las funcionalidades básicas del seguimiento de requisitos, lo que permitió comprobar que las funcionalidades que brinda facilitan el seguimiento de los requisitos en el CEIGE. Además las recomendaciones hechas permitirán que en un futuro SeReq pueda incorporar nuevas funcionalidades haciendo más fácil el seguimiento de los requisitos para las partes implicadas.

Conclusiones generales

Durante el presente trabajo se realizaron valiosos aportes, pues se partió de un problema específico y se llegó a una solución que abarca las necesidades del sistema.

- La valoración del análisis y diseño propuesto por el analista aunque se le hicieron un conjunto de cambios permitió ahorrar tiempo a la hora del desarrollo.
- El estudio de las tecnologías permitió explotar al máximo las bondades de las herramientas para el desarrollo.
- Las pruebas realizadas al software, posibilitaron detectar errores que afectaban el funcionamiento del sistema.
- Se cumplió el objetivo propuesto de realizar la implementación de la herramienta, basado en los requisitos funcionales, obteniendo como resultado la implementación de un sistema capaz de facilitar el seguimiento de los requisitos en el CEIGE y brindarle la información sobre la trazabilidad de los mismos a las partes implicadas con esta actividad.

Recomendaciones

Las recomendaciones propuestas para la continuidad del presente trabajo son:

- Integrar a la solución el componente reporteador dinámico para ampliar el conjunto de consultas posibles a los elementos de seguimiento.
- Realizar una prueba piloto de la herramienta en un porciento de las líneas de trabajo del proyecto ERP Cuba.
- Permitir llevar un histórico de las consultas que se realicen.
- Permitir llevar un histórico de los cambios que se realicen.
- Agrupar los requisitos por subsistema y por paquetes.
- Poder imprimir cualquier reporte.
- Adicionar funcionalidades para graficar, relaciones y estado de avances.

Referencias bibliográficas

ATOS, ITA, TID, GERMINUS, UPM. 2007. [En línea] 21 de Septiembre de 2007. [Citado el: 21 de Enero de 2007.] http://www.ines.org.es/vulcano/wp-content/uploads/2007/09/d6-estudio-de-herramientas-de-certificacion-tid.pdf.

Alvarez, Tahirí Rivero. 2010. *Validacion de Sereg.* Ciudad de la Habana : s.n., 2010.

Cabrera, Marianela Tenrero. 2010. *Manual de instalación de las herramientas para Sauxe.* Universidad de las Ciencias Informáticas. Ciudad de la Habana : s.n., 2010. Manual.

CALISOFT. 2009. *Guía de trazabilidad.* Universidad de las Ciencias Informáticas. 2009. págs. 1-16.

—. 2010. *Manual de instalación de las herramientas para Sauxe 1.5 .* 2010.

Codorníu, César Lage y Fernández, Ing. Osmar Leyet. 2009. *Arquitectura del ERP.* 2009.

Company, zend The PHP. 2010. zend The PHP Company. [Online] 2010. http://www.zend.com/en/community/framework.

Corporation, Microsoft. 2009. *La Arquitectura Orientada a Servicios (SOA) de Microsoft.* 2009.

Doctrine. 2010. Doctrine-PHP Object Persistence Libraries and More. *Doctrine.* [Online] 2010. http://www.doctrine-project.org/.

Framework, Ext JS Cross-Browser Rich Internet Application. 2009. Ext JS Cross-Browser Rich Internet Application Framework. *Sencha.* [Online] 2009. http://www.sencha.com/products/js/.

GESTIÓN, CENTRO DE SOLUCIONES DE. 2009. *Definición del ciclo de vida de los proyectos de desarrollo de software v1.0.* 2009.

Gómez Baryolo, Oiner and Mena, Grette Leydi. 2008. *Especificaciones de la Arquitectura del proyecto ERP Cuba.* 2008.

2009. *Guía para el seguimiento a los requisitos, 2.0.* Consultoria y proyectos, Universidad de las ciencias informaticas. Ciudad de la Habana : s.n., 2009. Plantilla.

IEEE. 1998. *IEEE Guide to Software Requirements Specifications.* 1998.

Landazuri, Barbara A. McDonald. 2005. *Definición de Perfiles en Herramientas de Gestión de Requisitos.* Madrid : s.n., 2005.

Lago, Ramiro. 2007. *Patrón "Modelo-Vista-Controlador".* [Online] Abril 2007. http://www.proactiva-calidad.com/java/patrones/mvc.html.

Lidia Fuentes, José M. Troya y Antonio Vallecillo. 2008. *Desarrollo de Software Basado en Componentes.* [prod.] Universidad de Málaga. España, Málaga, España : Universidad de Málaga, 2008.

Mañas, J. 2004. [En línea] 2004. http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s22.

MozillaES. 2010. La comunidad de Mozilla en español. *MozillaES.* [Online] 2010. http://www.mozillaes.org/.

Pérez, Yadenis Piñero. 2010. *RELACIÓN DE HERRAMIENTAS Y COMPONENTES.* Universidad de las ciencias informaticas. Ciudad de la Habana : s.n., 2010. Plantilla.

php. 2010. php. [Online] 2010. http://www.php.net/docs.php.

Pinheiro, Francisco. 2000. *Formal and Informal Aspects of Requirements Tracing.* III Workshop de Engenharia de Requisitos WER '2000. Rio de Janeiro : s.n., 2000.

PostqreSQL. 2010. PostqreSQL. [Online] 2010. http://www.postgresql.org/docs/8.3/static/release-8-3.html.

Pressman, Roger S. 2005. *Ingenieria del Software un enfoque practico.* La Habana : Felix Varela, 2005. pp. 171-187.

Reynoso, Carlos Billy. 2004. *Introducción a la Arquitectura de Software.* Buenos Aires : UNIVERSIDAD DE BUENOS AIRES, 2004.

RIZZI, I. F. M. COMPLEJIDAD CICLOMÁTICA,. 2009. [Online] 2009. http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/RIZZICOMPLEJIDAD.Pdf.

Seguridad, Subsistema de. 2010. *Acaxia (Sistema de Gestión Integral de Seguridad).* 2010.

UCID. 2008. *Estándar de diseño de interfaces para las aplicaciones de gestión.* Universidad de las Ciencias Informáticas. Ciudad de la habana : s.n., 2008. Plantilla.

Zend. 2008. Zend Framework. [Online] 2008. [Cited: febrero 5, 2010.] http://www.zend.com/products/studio/.

Bibliografía Consultada

Alvarez, Tahirí Rivero. 2010. *Validacion de Sereg.* Ciudad de la Habana : s.n., 2010.

Anaya, Víctor and Letelier, Patricio. *SmarTTrace: Una Herramienta para Trazabilidad de Requisitos en Proyectos basados en UML.* Valencia.

ATOS, ITA, TID, GERMINUS, UPM. 2007. [En línea] 21 de Septiembre de 2007. [Citado el: 21 de Enero de 2007.] http://www.ines.org.es/vulcano/wp-content/uploads/2007/09/d6-estudio-de-herramientas-de-certificacion-tid.pdf.

Alvarez, Tahirí Rivero. 2010. *Validacion de Sereg.* Ciudad de la Habana : s.n., 2010.

Cabrera, Marianela Tenrero. 2010. *Manual de instalación de las herramientas para Sauxe.* Universidad de las Ciencias Informáticas. Ciudad de la Habana : s.n., 2010. Manual.

—. 2010. *Manual de instalación de las herramientas para Sauxe 1.5 .* 2010.

CALISOFT. 2009. *Guía de trazabilidad.* Universidad de las Ciencias Informáticas. 2009. págs. 1-16.

Codorníu, César Lage y Fernández, Ing. Osmar Leyet. 2009. *Arquitectura del ERP.* 2009.

Company, zend The PHP. 2010. zend The PHP Company. [Online] 2010. http://www.zend.com/en/community/framework.

Corporation, Microsoft. 2009. *La Arquitectura Orientada a Servicios (SOA) de Microsoft.* 2009.

Doctrine. 2010. Doctrine-PHP Object Persistence Libraries and More. *Doctrine.* [Online] 2010. http://www.doctrine-project.org/.

Dorfman, R.H. Thayer and M. 1997. *Software Requirements Engeneering.* 1997. Vol. IEEE Computer Society Press.

Echavarría, Yaimí Márquez Alpízar y Yenni Valdés. 2008. *Procedimiento general de pruebas de Caja Blanca aplicando la técnica del Camino Básico.* Universidad de las Ciencias Informáticas. Ciudad de la Habana : s.n., 2008.

Framework, Ext JS Cross-Browser Rich Internet Application. 2009. Ext JS Cross-Browser Rich Internet Application Framework. *Sencha.* [Online] 2009. http://www.sencha.com/products/js/.

GESTIÓN, CENTRO DE SOLUCIONES DE. 2009. *Definición del ciclo de vida de los proyectos de desarrollo de software v1.0.* 2009.

Gómez Baryolo, Oiner and Mena, Grette Leydi. 2008. *Especificaciones de la Arquitectura del proyecto ERP Cuba.* 2008.

2009. *Guía para el seguimiento a los requisitos, 2.0.* Consultoria y proyectos, Universidad de las ciencias informaticas. Ciudad de la Habana : s.n., 2009. Plantilla.

IEEE. 1998. *IEEE Guide to Software Requirements Specifications.* 1998.

Landazuri, Barbara A. McDonald. 2005. *Definición de Perfiles en Herramientas de Gestión de Requisitos.* Madrid : s.n., 2005.

Lago, Ramiro. 2007. *Patrón "Modelo-Vista-Controlador".* [Online] Abril 2007. http://www.proactiva-calidad.com/java/patrones/mvc.html.

Lidia Fuentes, José M. Troya y Antonio Vallecillo. 2008. *Desarrollo de Software Basado en Componentes.* [prod.] Universidad de Málaga. España, Málaga, España : Universidad de Málaga, 2008.

Mañas, J. 2004. [En línea] 2004. http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s22.

MozillaES. 2010. La comunidad de Mozilla en español. *MozillaES.* [Online] 2010. http://www.mozillaes.org/.

Pérez, Yadenis Piñero. 2010. *RELACIÓN DE HERRAMIENTAS Y COMPONENTES.* Universidad de las ciencias informaticas. Ciudad de la Habana : s.n., 2010. Plantilla.

php. 2010. php. [Online] 2010. http://www.php.net/docs.php.

Pinheiro, Francisco. 2000. *Formal and Informal Aspects of Requirements Tracing.* III Workshop de Engenharia de Requisitos WER '2000. Rio de Janeiro : s.n., 2000.

PostqreSQL. 2010. PostqreSQL. [Online] 2010. http://www.postgresql.org/docs/8.3/static/release-8-3.html.

Pressman, Roger S. 2005. *Ingenieria del Software un enfoque practico.* La Habana : Felix Varela, 2005. pp. 171-187.

Reifer, Donald. 1994. *Encyclopedia of Software Engeneering.* s.l. : J.J Marciniack, 1994. págs. 1043-1054.

Reynoso, Carlos Billy. 2004. *Introducción a la Arquitectura de Software.* Buenos Aires : UNIVERSIDAD DE BUENOS AIRES, 2004.

RIZZI, I. F. M. COMPLEJIDAD CICLOMÁTICA,. 2009. [Online] 2009. http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/RIZZICOMPLEJIDAD.Pdf.

Seguridad, Subsistema de. 2010. *Acaxia (Sistema de Gestión Integral de Seguridad).* 2010.

Silva, Andrés. 2009. [En línea] 2 de Octubre de 2009. [Citado el: 9 de Febrero de 2010.] http://asilva.wordpress.com/2006/11/01/herramientas-libres-de-gestion-de-requisitos/.

Scull, Sandy Machado. 2009. *Análisis y diseño de la herramienta SeReq para el seguimiento de los requisitos en el proyecto ERP Cuba.* Universidad de las Ciencias Informáticas. Ciudad de la Habana : s.n., 2009. Trabajo de diploma.

UCID. 2008. *Estándar de diseño de interfaces para las aplicaciones de gestión.* Universidad de las Ciencias Informáticas. Ciudad de la habana : s.n., 2008. Plantilla.

Vega Miniet, Yanet. 2008. *Plantilla DCS - Especificación de Requisitos 3.0.* 2008. Equipo de Analistas Principales del proyecto ERP Cuba.

Zend. 2008. Zend Framework. [Online] 2008. [Cited: febrero 5, 2010.] http://www.zend.com/products/studio/.

Anexos

Anexo 1. Modelo conceptual propuesto por el analista

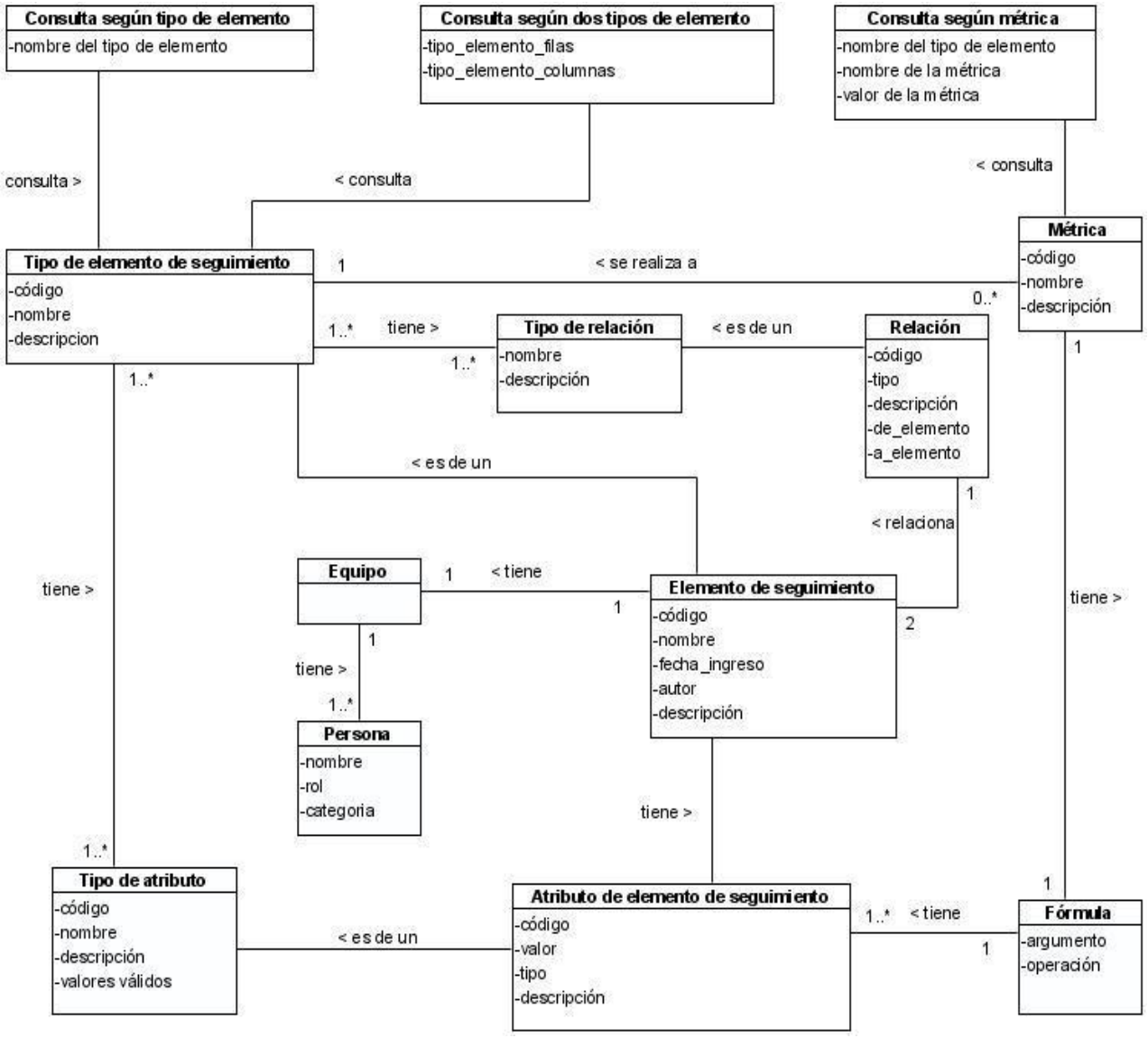


Figura 11. Modelo conceptual

Anexo 2. Modelo de Datos de SeReq

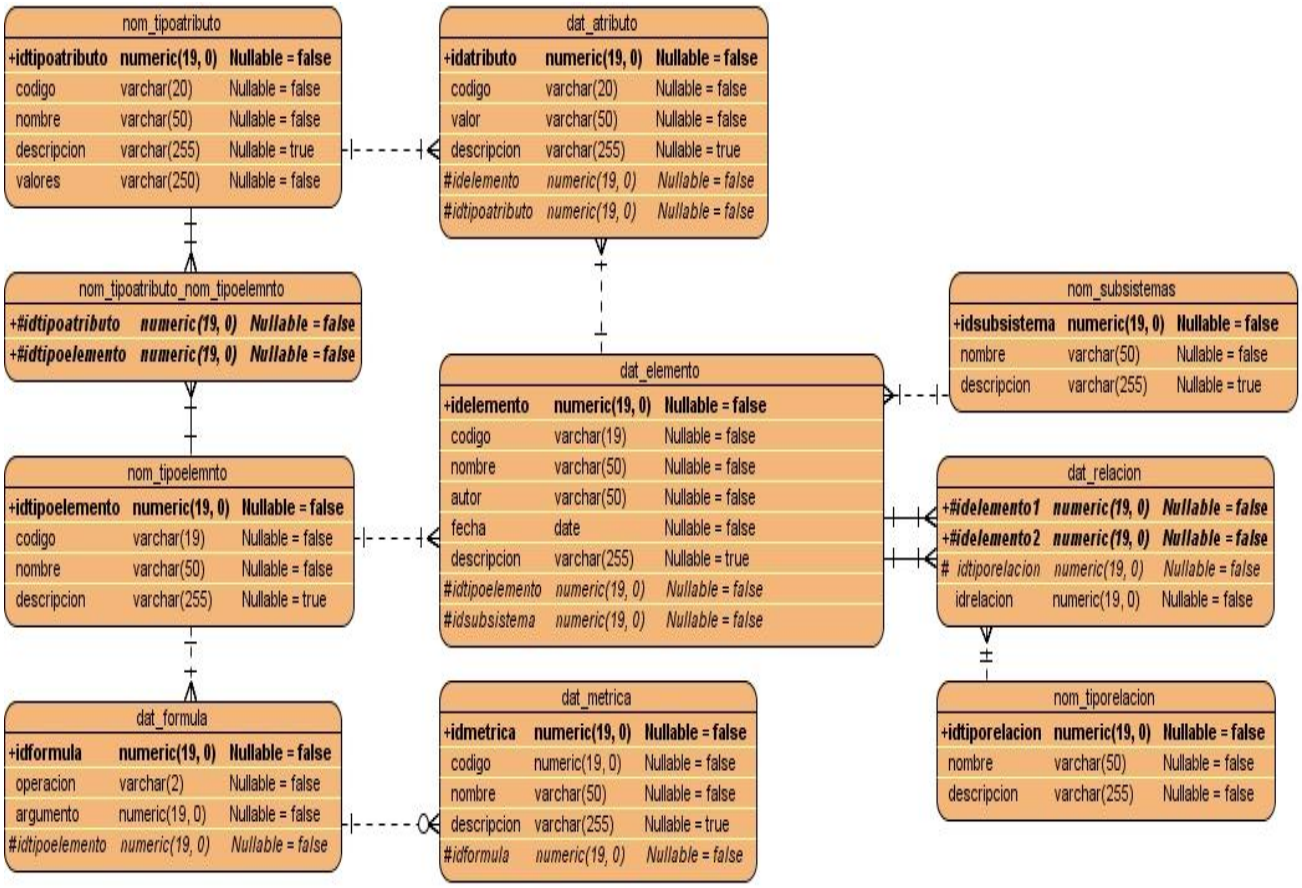


Figura 12. Modelo de Datos de SeReq

Anexo 3. Diagrama de clases persistentes del sistema

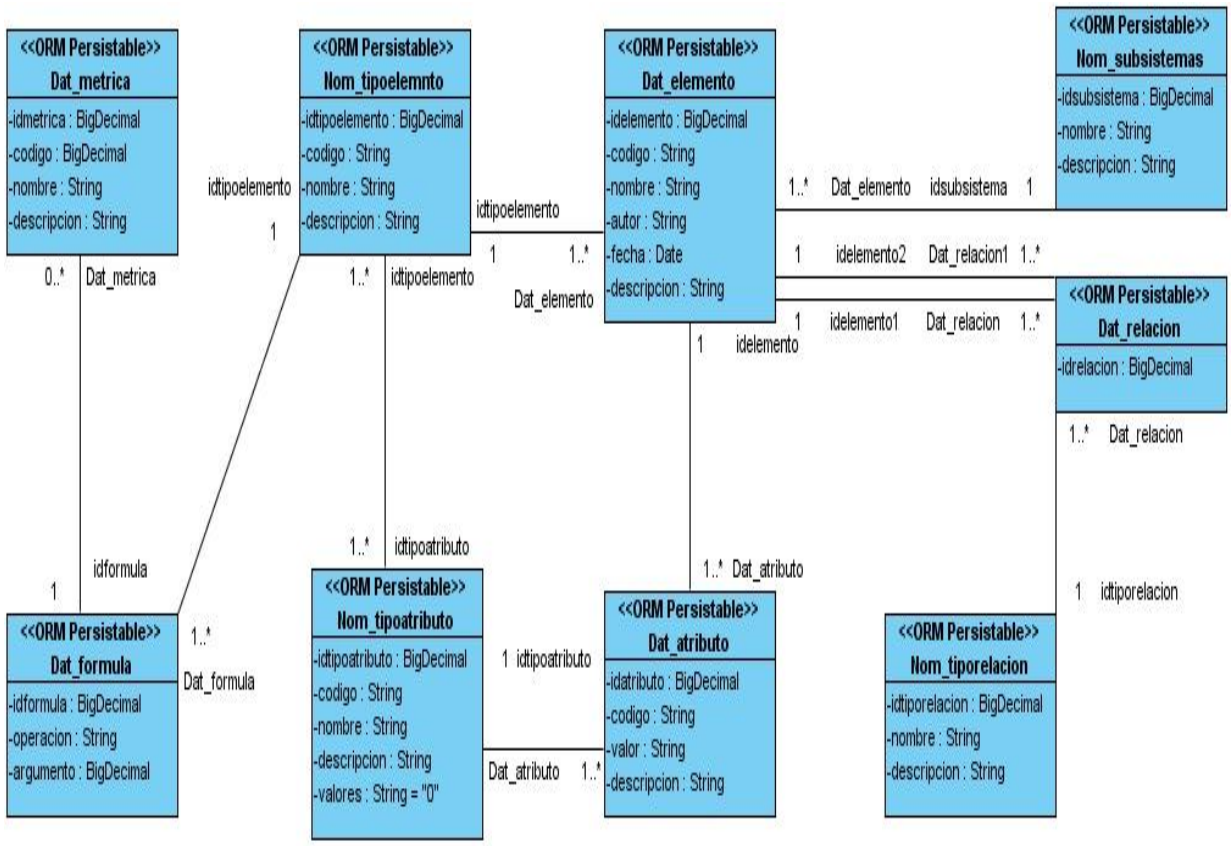


Figura 13. Diagrama de clases persistentes del sistema

Anexo 4. Interfaces de SeReq

Figura 14. Interfaz de inicio de SeReq

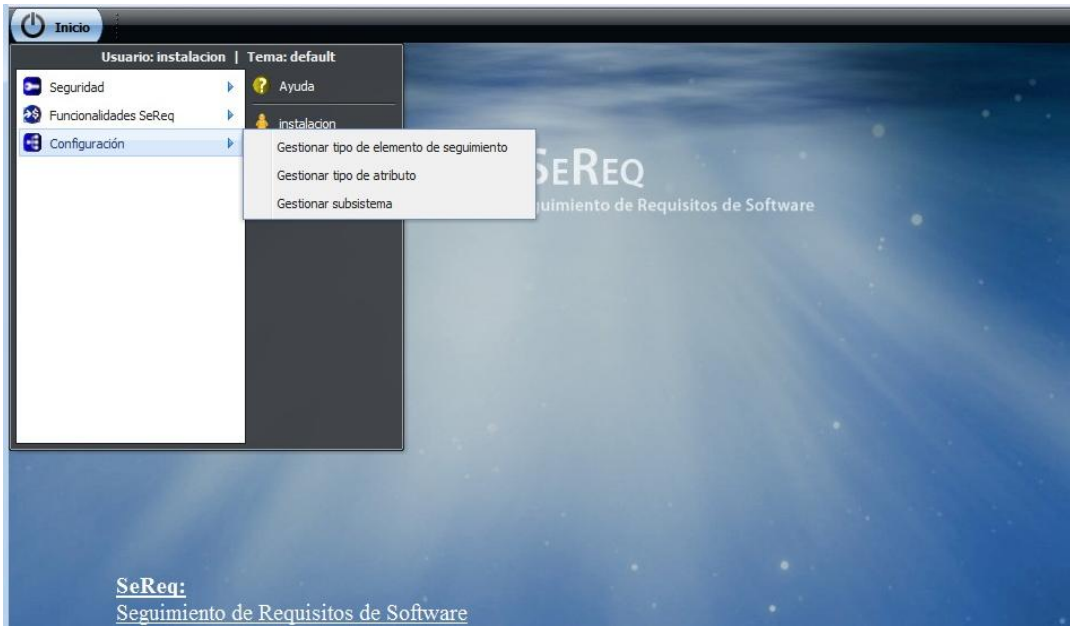


Figura 15. Interfaz con las funcionalidades de Configuración

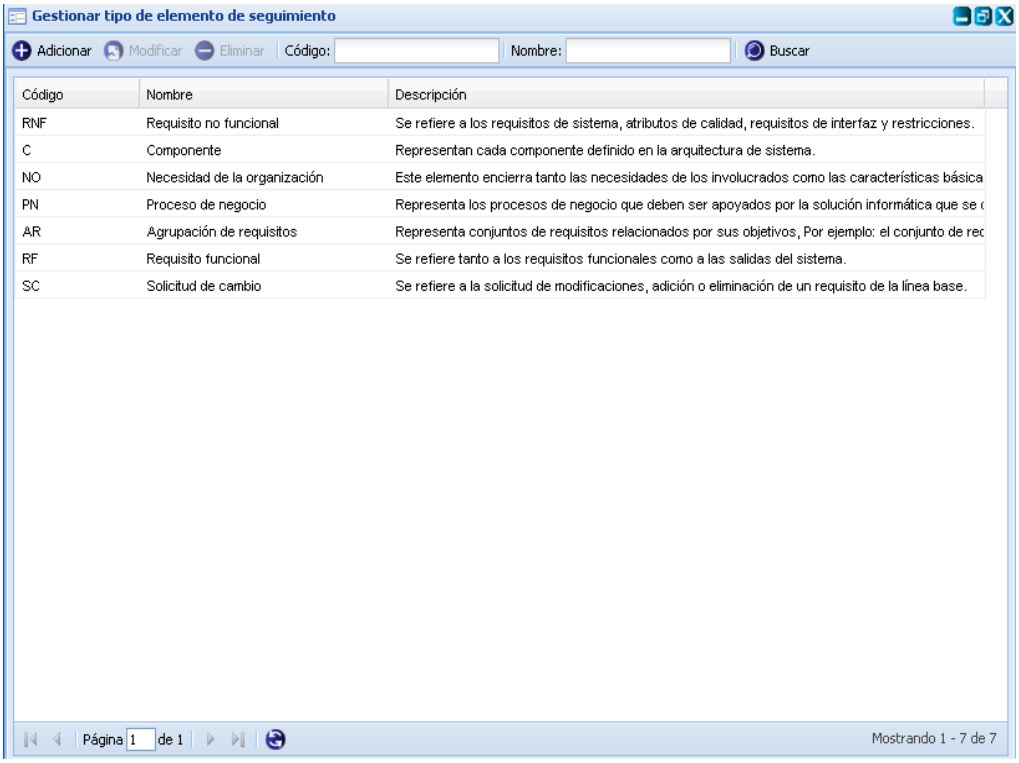


Figura 16. Interfaz que gestiona los tipos de elementos

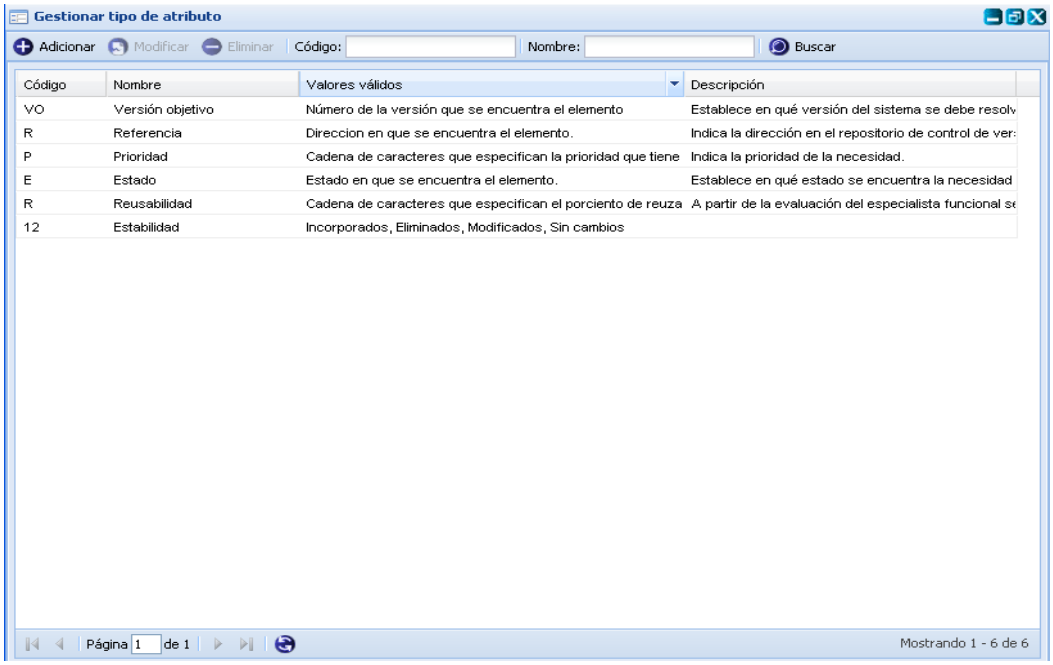


Figura 17. Interfaz que gestiona los tipos de atributos

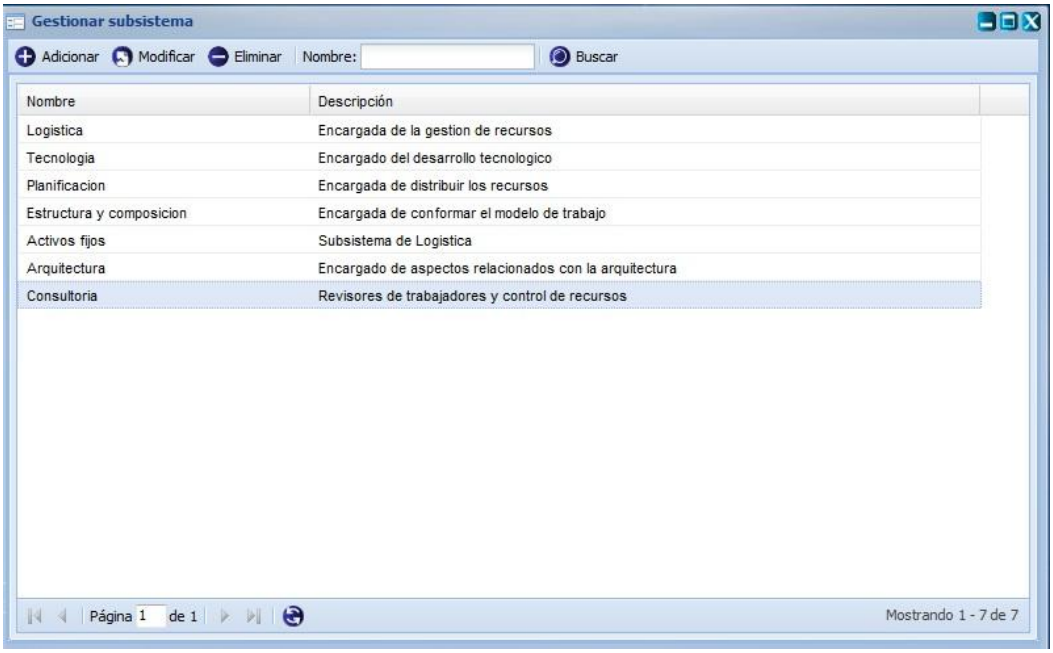


Figura 18. Interfaz que gestiona los subsistemas

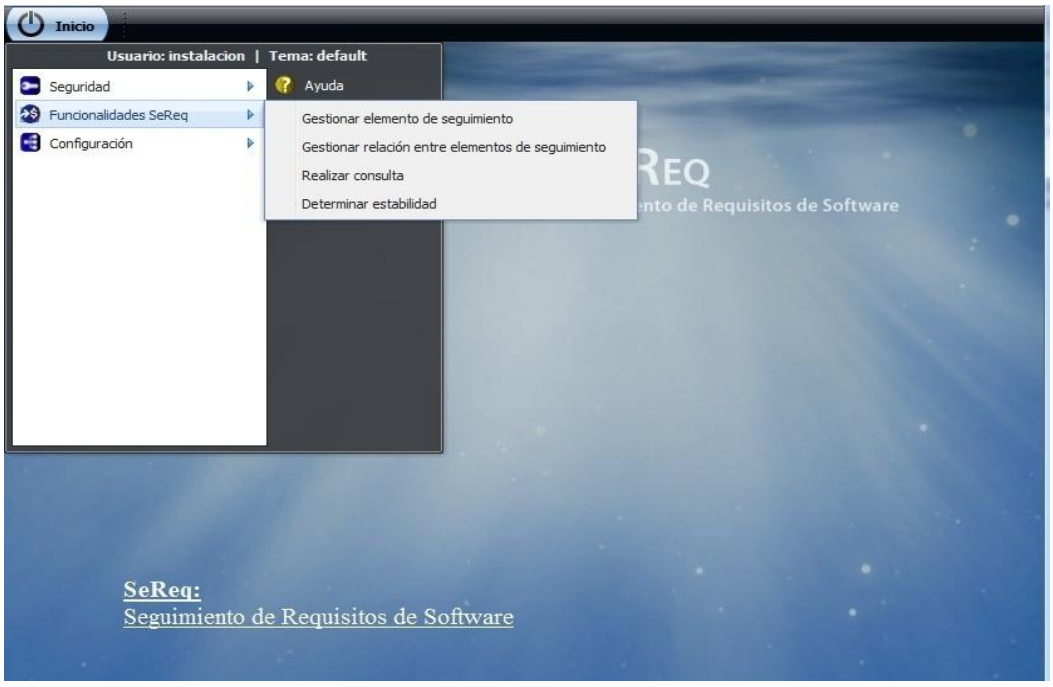


Figura 19. Interfaz de las funcionalidades de SeReq

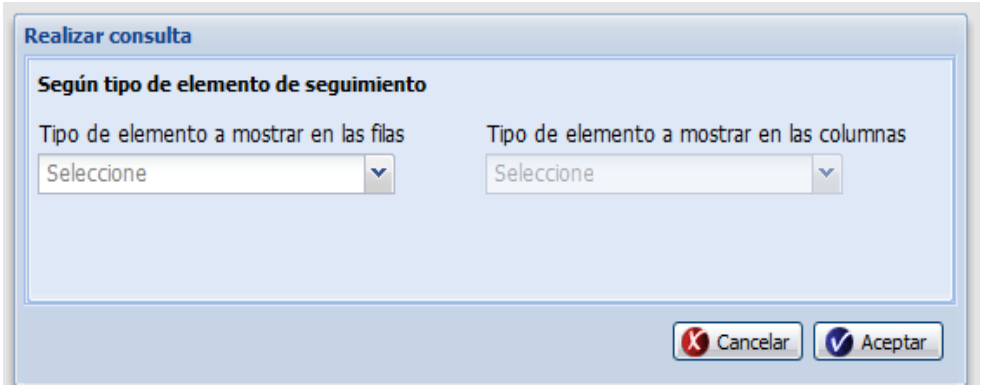


Figura 20. Interfaz que permite realizar consultas

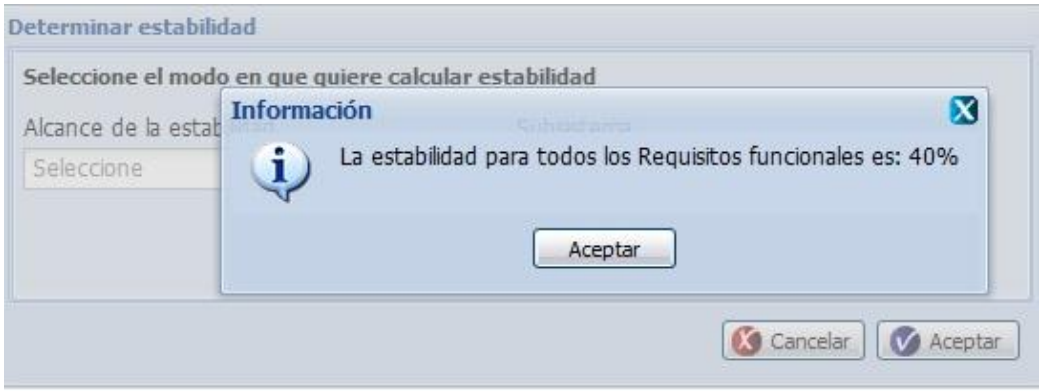


Figura 21. Interfaz que permite determinar la estabilidad

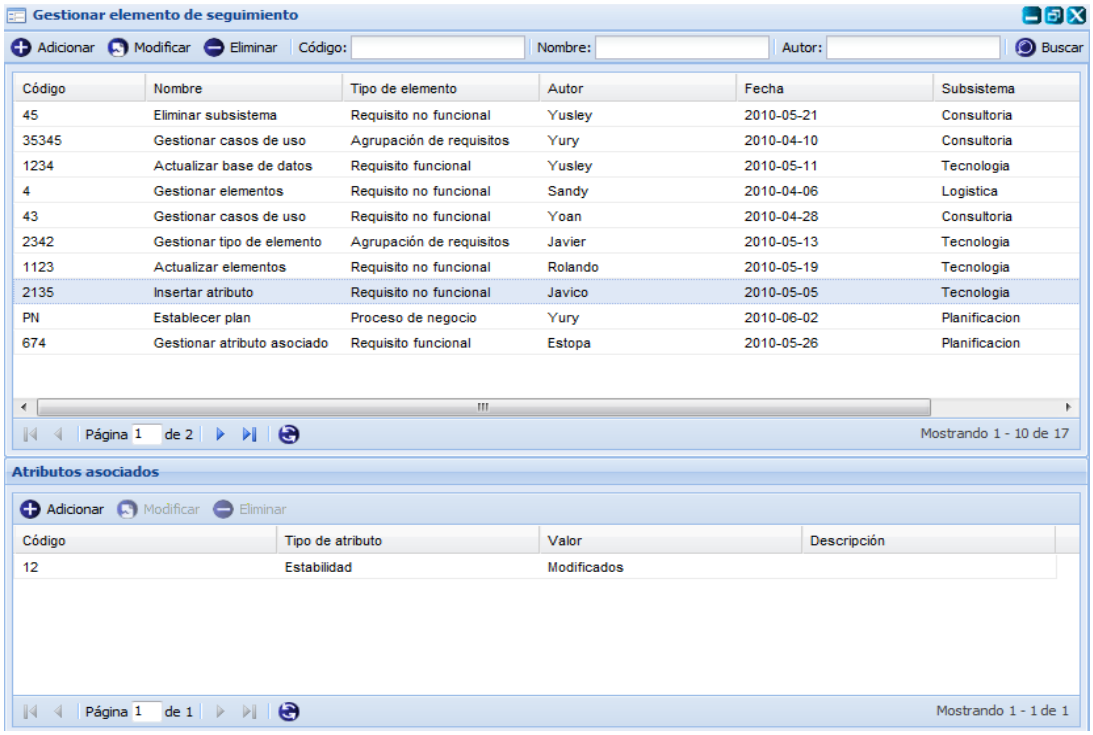


Figura 22. Interfaz que permite gestionar elementos de seguimiento

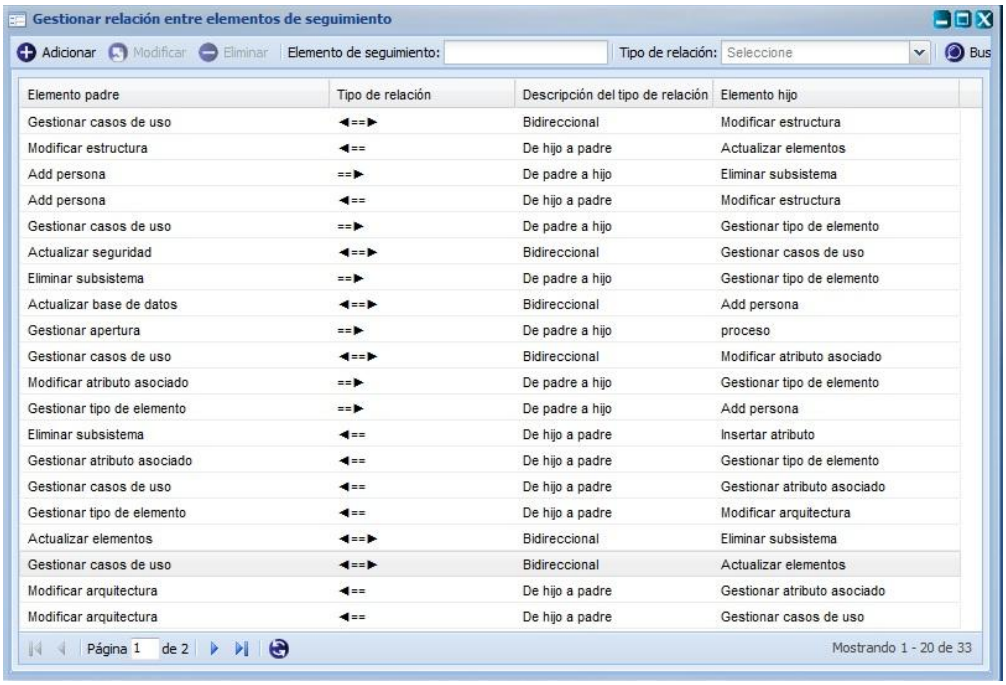


Figura 23. Interfaz que permite gestionar relaciones entre elementos de seguimiento

Glosario de términos

UCI: Universidad de las Ciencias Informáticas

CEIGE: Centro de informatización de la gestión de entidades.

ERP: Sistemas de planificación de recursos empresariales (Enterprise Resource Planning, ERP por sus siglas en inglés).

OSRMT: Herramienta libre para la gestión de los requisitos (Open Source Requirements Management Tool).

MDOC: Modelo de Desarrollo Orientado a Componentes.

MVCC: Es un sistema de acceso concurrente multiversión (Multiversion concurrency control).

HTTPS: Protocolo de Web para intercambio de información segura.

ORM: El Mapeo Objeto Relacional es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

NA: No aplica es cuando no es importante lo que se escribe en los campos del formulario si se va a cancelar o se puede poner cualquier término.