Universidad de las Ciencias Informáticas
Facultad 5

HERRAMIENTA INTELIGENTE PARA LA CAPTURA DE REQUISITOS NO FUNCIONALES

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor(es): Edison Prado Lemus

Leandro José González Yero

Tutora: Ing. Irina Elena Argota Vega.

Co-Tutor: Yunerkis Prevot Urgellés.

Ciudad de la Habana, 2009

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Datos de Contacto

Tutor:

Ing. Irina Elena Argota Vega.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

e-mail: iargota@uci.cu

Ingeniero en Ciencias Informáticas, Profesor en Adiestramiento de la Universidad de las Ciencias Informáticas, con 1 año de experiencia en su desempeño laboral.

Co-Tutor:

Ing. Yunerkis Prevot Urgellés.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

e-mail: yprevot@uci.cu.

Ingeniero en Ciencias Informáticas, Profesor en Adiestramiento de la Universidad de las Ciencias Informáticas, con 1 año de experiencia en su desempeño laboral.

Dedicatoria

Dedicar este sueño a mis padres que son los que han hecho de mí el hombre que soy hoy, a mi hermano, mis amigos, toda mi familia, los vecinos de mi barrio que son mi otra familia, a los amigos de mi papa que me quieren como si fuese su hijo. En fin a todos aquellos que de una forma u otra creyeron que podía ser un ingeniero.

Leandro.

Dedicar este sueño que hoy se convierte en realidad a mis padres que siempre me han apoyado y me han dado fuerzas para llegar hasta aquí, a mi sobrinita querida que es mi fuente de inspiración. A toda mi familia que me quiere tanto y me apoyan en todo. A todos mis amigos que siempre creyeron en mí y siempre me alentaron en seguir adelante. En fin a todos los que de una forma u otra me guiaron hasta ser un ingeniero.

Edison.

Agradecimientos

Agradecer a todas mis amistades por ayudarme en los momentos difíciles de esta universidad, mis amigos Prado, André, Rigo, Licea, Edel, Yusnier, Argelio, Idelvis, Israel, Félix, Yordankis, Vanegas, Leyva, Hassan, Irais, entre otros por siempre estar ahí firme en los peores momentos. Agradecer también a mis tutores por aguantarme en estos meses y por su gran ayuda en la realización de este trabajo, a mis nuevos amigos de cuarto que en pocos meses de conocernos supieron apreciar mi amistad. En fin a todos los que posibilitaron estar aquí hoy.

Leandro.

Agradecer a mi madre y mi padre por apoyarme y quererme mucho y darme tantos consejos, y gracias a ellos estoy aquí hoy. A mi hermana querida del alma, a mi tía especial Omara, a mi primo y su novia y a toda mi familia. A mi compañero de tesis que sin él no hubiese sido posible realizar esta tarea. A mis amigos que han venido conmigo desde que me inicié como estudiante, Edel y Licea. A mis amigos que llevamos 5 años juntos, Yusnier, Argelio, Félix, Yordankis, Leyva, Vanegas, Idelvis, Bubaire, Israel, y a todos los compañeros de cuarto de este año que nos hemos integrado y convertido en poco tiempo en buenos amigos. A mis amistades Irais, Yaimara, Liannne, por aguantarme tanto. En fin a todos los que hicieron posible el desarrollo de este trabajo de diploma.

Edison

Resumen

Los Requisitos No Funcionales (RNF) especifican propiedades del sistema, como restricciones del entorno o la implementación, rendimiento, dependencia de las plataformas, facilidad de mantenimiento, extensibilidad y fiabilidad, es responsabilidad de Analistas de Requisitos la búsqueda de las verdaderos requisitos del sistema.

En el Polo de Hardware y Automática (PHA) de la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI) no se reutilizan las experticias de los Analistas de Requisitos en proyectos anteriores, dificultando la socialización del conocimiento y la productividad de los nuevos integrantes. No se utilizan métodos para la identificación y gestión de RNF, así como herramientas que agilicen este proceso, no encontrándose a nivel mundial una herramienta para la captura de los RNF, por lo que esta actividad se realiza de forma manual; por tal motivo los Analistas del Polo solicitaron la realización de una herramienta inteligente para la captura de los RNF a partir de las propiedades del sistema, siendo este el objetivo principal de la presente investigación.

Palabras clave: Ingeniería, requisitos no funcionales, sistemas expertos.

Tabla de contenido

Introducción

Bajo las ideas de nuestro Comandante en Jefe Fidel Castro “Las producciones intelectuales serán el sustento fundamental de Cuba. La idea es convertir la informática en una de las ramas más productivas y aportadoras de recursos para la nación” surge la Universidad de las Ciencias Informáticas concebida para que sus estudiantes y profesores centraran su labor en la docencia, investigación y producción, logrando convertirla en una universidad innovadora de excelencia científica, académica y productiva de forma tal que se gradúen profesionales integrales comprometidos con la patria, soporte de la informatización del país y la competitividad internacional de la industria cubana del software (1).

En el transcurso de vida de la Universidad, nacida en el calor de la batalla de ideas en el año 2002, ha alcanzado un enorme prestigio tanto nacional como internacionalmente, lo que se demuestra en la eficiencia de los productos desarrollados en la misma, esto ha provocado grandes demandas de software por lo que el centro se vio en la necesidad de crear varios Polos Productivos en sus 10 facultades, cada uno especializado en la producción de un conjunto determinado de software.

En la facultad 5 se creó el Polo de Hardware y Automática en el que existen algunos productos que se encuentran en etapa de desarrollo.

El proyecto *Supervisory Control and Data Acquisition* (SCADA) / (Control Supervisor y Adquisición de Datos) "Guardián del ALBA" es un software de supervisión, control y adquisición de datos desarrollado para la empresa petrolera PDVSA con la participación activa de Empresas Venezolanas y la UCI. Actualmente se encuentra funcionando en fase de prueba en el "Patio de Calderas" de Barinas, Venezuela. El proyecto ha sido realizado completamente con plataformas libres: la gestión de configuración, documental, gestión del proyecto y tareas se han mantenido actualizadas en este entorno de desarrollo, contribuyendo a alcanzar la Soberanía Tecnológica de Venezuela en el campo de la informática.

El proyecto “Supervisión Energética” con la Empresa Eléctrica (UNE) es un sistema cuyo objetivo es permitir el monitoreo del consumo de energía eléctrica mediante metro contadores de cada centro que esté conectado al software, entre sus funcionalidades se encuentran: la gestión de planes, extra-planes, usuarios energéticos de cada centro, la captura de los datos en dispositivos Ayudante Personal Digital (PDA), y la visualización de los datos capturados en una aplicación Web donde los clientes tengan acceso a los datos referentes a su consumo.

Otros se encuentran en la etapa de conceptualización como los proyectos con la Empresa de Telecomunicaciones de Cuba S.A (ETECSA), Ministerio de la Industria Básica (MINBAS), Cubapetróleo (CUPET).

El PHA posee cinco líneas de trabajo como son:

- ✓ Formación Posgrado y Pregrado, con el objetivo de capacitar a sus estudiantes y profesores en el área de la Automática y otros temas de producción y gestión de software.
- ✓ Cooperación Nacional e internacional, buscando la participación en esta área de instituciones nacionales e internacionales.
- ✓ Investigación, que permita la adquisición de conocimientos necesarios en el desarrollo de aplicaciones.
- ✓ Desarrollo de aplicaciones que permitan la reutilización del código fuente.
- ✓ Contrato de Proyectos, con clientes nacionales o internacionales que soliciten los servicios del Polo como vía de solución a sus necesidades.

En el PHA, no se reutilizan las experticias de los Analistas de Requisitos en proyectos anteriores, dificultando la socialización del conocimiento y la productividad de los nuevos integrantes. No se utilizan métodos para la identificación y gestión de Requisitos No Funcionales, así como herramientas que agilicen este proceso.

A raíz de la situación anteriormente descrita se identificó como **problema científico** de la presente investigación: La inexistencia de herramientas libres que permitan la captura de RNF en proyectos productivos.

El problema científico descrito genera como **objeto de estudio** el proceso de gestión de requisitos; siendo los Requisitos No Funcionales en proyectos productivos, el **campo de acción** de la presente investigación.

El **objetivo general** que se desea alcanzar es el desarrollo de una herramienta para la captura de RNF, aplicada a proyectos productivos del PHA.

Definiéndose así las siguientes **tareas de la investigación:**

- ✓ Identificación de conceptos, definiciones y teorías en el campo de los requisitos.
- ✓ Organización de los RNF por categorías de acuerdo a los requisitos extraídos de los proyectos productivos.
- ✓ Identificación de mecanismos o técnicas que permitan la implementación de la herramienta.
- ✓ Realización una Base de Datos (BD) que permita aplicar la técnica escogida para la captura de RNF.
- ✓ Selección de una herramienta para la implementación de la aplicación.
- ✓ Implementación del sistema para la captura de RNF, aplicándolo al PHA.

De acuerdo al problema científico plateado la **idea a defender** que guiará la investigación es: Realizar una correcta implementación de la herramienta para la toma de decisiones en la captura de RNF en proyectos productivos para lograr la automatización de dicho proceso, aplicándolo en proyectos del PHA.

Para el cumplimiento de estos objetivos se llevan a cabo varios **métodos y técnicas** de la investigación, que se mencionan a continuación.

A nivel teórico:

Métodos de análisis-síntesis: Para analizar las teorías y documentos referentes a la captura de requisitos no funcionales, permitiendo de esta manera la extracción de los elementos más importantes que se relacionan con este proceso.

Modelación: Para la caracterización o representación de las funcionalidades que tendrá el algoritmo de aprendizaje automático que permitirá una correcta captura de requisitos no funcionales.

A nivel empírico:

Experimento**:** En la elaboración de bases de conocimientos que puedan abarcar la mayor cantidad de información posible sobre los requisitos no funcionales y poder llegar a resultados satisfactorios en el desarrollo del sistema.

Entrevista**:** Apoyará a la incorporación de conocimientos mediante las entrevistas planificadas efectuadas a los especialistas tanto en rama de Inteligencia Artificial como en Ingeniería de Software.

Capítulo 1: Fundamentación teórica

En este capítulo se presentan conceptos fundamentales sobre Ingeniería de Requisitos (IR), en particular el concepto de Requisitos Funcionales (RF) y No Funcionales, con sus categorías, tipos y propiedades. Se definen las técnicas de captura, definición y validación de requisitos; además se muestran conceptos de la Inteligencia Artificial así como las técnicas de aprendizaje automático para su estudio. Además se muestra sobre qué Sistema Operativo será implementado el sistema, la metodología de desarrollo de software por la cual se regirá, la herramienta de modelado que se utilizará, así como el lenguaje de programación.

1.1 Requisitos

Los requisitos son importantes porque proporcionan la base para todo el trabajo de desarrollo que le sigue al software (2). Existen muchas versiones de definiciones de lo “requisitos”, a continuación se muestran algunas:

Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal (3).

Somerville lo define así: “Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste” (4).

Para un mayor entendimiento, se define por requisito, como la cualidad necesaria de un sistema, una declaración que identifique una capacidad, una característica, o un factor de calidad de un sistema que proporcione valor y utilidad al cliente o usuario. Los requisitos son las descripciones de los servicios y las restricciones que se generan durante el proceso de ingeniería de requisitos (5).

Un requisito puede ir desde una descripción abstracta de alto nivel de un servicio o una restricción del sistema hasta una especificación funcional matemática muy detallada (6).

Se hace necesario hacer una separación entre niveles de descripción; donde se encuentran los requisitos del usuario y los requisitos del sistema.

Los requisitos del usuario son las descripciones de lo que se espera que el sistema proporcione y las restricciones bajo las cuales debe funcionar. Ahora, los requisitos del sistema establecen con detalle las funciones, servicios y restricciones operativas del sistema. Estos requisitos deben de ser precisos, pues, son los que definen lo que se va a implementar (7)

Los requisitos de software pueden clasificarse en 2 categorías: Requisitos Funcionales y Requisitos No Funcionales.

Los **Requisitos Funcionales** son los que definen las funciones que el sistema será capaz de realizar.

Los **Requisitos No Funcionales** tienen que ver con características que de una u otra forma puedan limitar el sistema.

Los requisitos que se definan deben tener las siguientes propiedades:

Conjunto Único. Cada requisito debe ser declarado sólo una vez.

- ✓ **Normalizado**. Los requisitos no se deben traslapar (Por ejemplo, no se deben referir a otros requisitos ni a las capacidades de otros requisitos).
- ✓ **Conjunto interdependiente**. Se deben definir explícitamente las relaciones entre los requisitos individuales para mostrar cómo los requisitos están relacionados para formar el sistema completo.
- ✓ **Completo**. Una Especificación de Requisitos del Sistema (*System Requirements Specification* (SyRS). Debe incluir todos los requisitos dados por el cliente, así como aquellos requisitos necesarios para la definición del sistema.
- ✓ **Consistente**. El contenido de una SyRS debe ser consistente y sin contradicciones en el nivel de detalle, estilo de la declaración de los requisitos y en la presentación del material.
- ✓ **Acotado**. Los límites, alcance y el contexto de los requisitos del sistema deben ser identificados.
- ✓ **Modificable**. La SyRS debería ser modificable. Requisitos claros y sin traslapamientos contribuyen a lograrlo.
- ✓ **Configurable**. Las versiones deberían ser mantenidas a través del tiempo y a través de las instancias de la SyRS.
- ✓ **Verificable**: Un requisito es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o

pruebas.

- ✓ **Granular**. Éste debe ser el nivel de abstracción para el sistema que está siendo definido.
- ✓ **No ambiguo**: Un requisito no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.
- ✓ **Necesario**: Un requisito es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.
- ✓ **Son realistas.** Puede el sistema hacer lo que el cliente desea**.**
- ✓ **Rastreables.** Trazables, el origen de cada requisito está claro y se posibilita la referencia de cada uno de estos requisitos en desarrollos futuros o incrementos de la documentación (8).

1.1.1 Requisitos no funcionales

La presente investigación tendrá su basamento en los Requisitos No Funcionales, los cuales especifican propiedades del sistema, como restricciones del entorno o la implementación, rendimiento, dependencia de las plataformas, facilidad de mantenimiento, extensibilidad y fiabilidad(9). La figura 1 muestra los tipos de Requisitos no Funcionales.

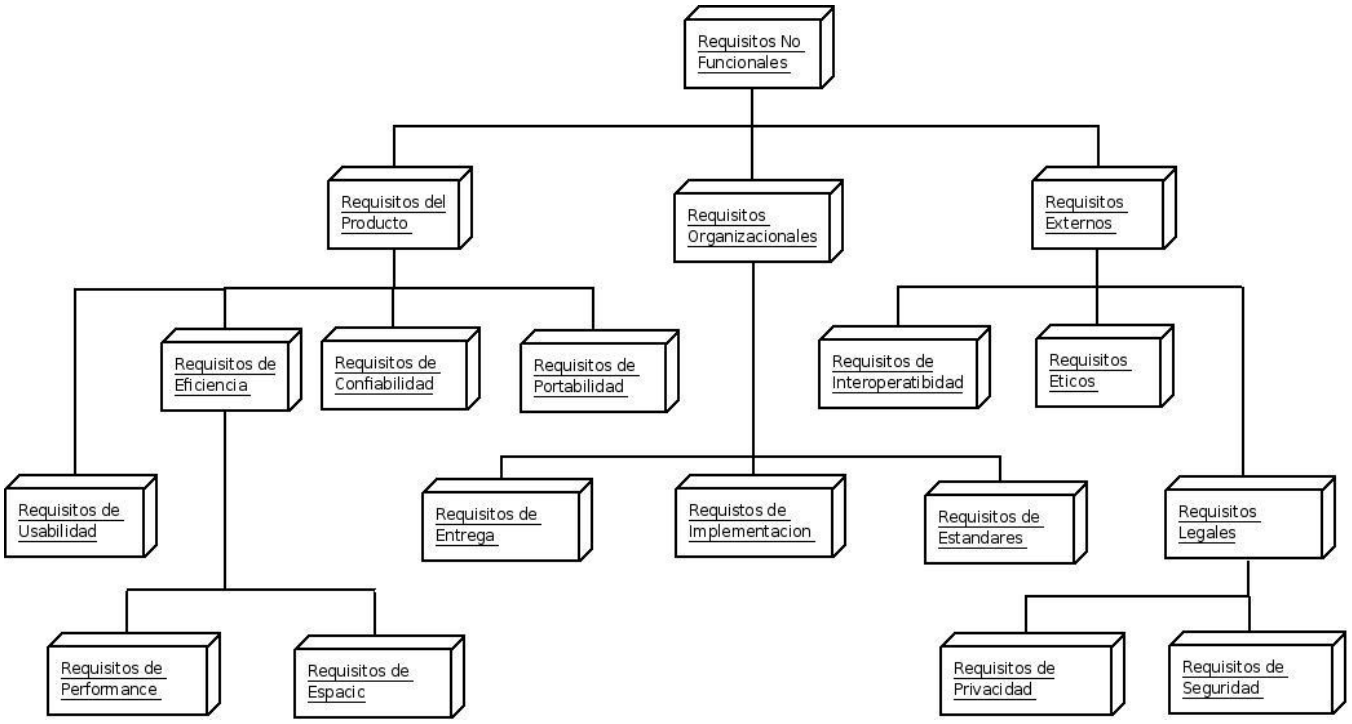


Figura N0. 1 Tipos de RNF.

Se distinguen tres tipos de RNF:

- ✓ **Requisitos del producto**. Especifican el comportamiento del producto; como los requisitos de desempeño en la rapidez de ejecución del sistema y cuánta memoria se requiere.
- ✓ **Requisitos organizacionales**. Se derivan de las políticas y procedimientos existentes en la organización del cliente y en la del desarrollador
- ✓ **Requisitos externos**. Se derivan de los factores externos al sistema y de su proceso de desarrollo.

Requisitos del producto

Eficiencia

Eficiencia es la habilidad del software para poner la cantidad mínima de demanda sobre los recursos de hardware como sea posible, los requisitos de eficiencia son concretos y ligados a las propiedades del hardware. Una eficiencia óptima requerirá una adaptación perfecta al ambiente particular de hardware y software. El sistema debe ser rápido y el tiempo de respuesta debe ser el mínimo posible.

Confiabilidad

Uno de los más importantes por parte del usuario es el de Fiabilidad debido a que debe de tener en cuenta la recuperación frente a fallos de conexión: asegurar que no se pierdan los datos del perfil definido por el usuario. Esto incluye no perderlos en el envío al servidor o en el envío a otras máquinas, como no perderlos si hay un fallo de conexión entre el receptor del usuario y el servidor. La recuperación frente a fallos del sistema: posibilidad de reiniciar el sistema. Verificar la Fiabilidad en la autenticación de los usuarios y la posibilidad de dar marcha atrás en la definición del perfil de cada usuario. Los requerimientos de Confiabilidad abarcan: Frecuencia y severidad de los fallos. Protección contra fallos. Recuperación. Predicción de fallos. Tiempo medio entre fallos.

Portabilidad

Portabilidad es la facilidad de transferir los productos software a diferentes entornos hardware y software. Su radio de acción no sólo se enfoca en las variaciones del hardware físico sino también de la máquina hardware-software, la que realmente se programa y que incluye el sistema operativo, el sistema de ventanas y otras herramientas fundamentales. Muchas de las incompatibilidades existentes entre las plataformas son injustificadas, y convierte a la portabilidad en un asunto primordial tanto para los que desarrollan como para los que usan el software.

Usabilidad

Los requisitos de usabilidad se recopilan usando técnicas como las entrevistas, las encuestas y la observación. Existen dos tipos de requisitos de usabilidad: los requisitos cualitativos y los requisitos cuantitativos, y ambos se relacionan con el concepto de métricas de usabilidad. Los requisitos cualitativos de usabilidad se concentran en las metas deseadas de usabilidad de un sistema, por ejemplo, “el sistema debe ser fácil de utilizar” o “debe haber satisfacción de parte del usuario”. Los requisitos cualitativos de usabilidad pueden ser subjetivos y no son siempre fáciles de medir o cuantificar. Los requisitos cuantitativos de usabilidad pueden expresarse como medidas específicas de desempeño. Ejemplos de estas medidas son el tiempo necesario para completar una tarea específica en un grupo específico de usuarios, o el número de errores por tarea, o el tiempo empleado en el uso de la documentación. Son requisitos asociados a la experiencia del uso de la aplicación. Los requisitos

de Usabilidad abarcan: Factores Estéticos, Interfaces de Usuario, Ayudas en Línea, Documentación para Usuarios.

Rendimiento y Escalabilidad

Se debe de tener en cuenta el requisito de rendimiento y escalabilidad porque convendría que los usuarios examinaran con detenimiento hasta qué punto el proyecto se ajusta a sus expectativas en cuanto a los tiempos de respuesta y es capaz de prestar servicio adecuadamente de acuerdo al tipo y tamaño para el que ha sido concebido. Los requisitos de rendimiento imponen condiciones sobre los requisitos funcionales como velocidad, tiempo de respuesta y uso de la memoria. La mayoría de los requisitos de rendimiento solamente son relevantes para cada CU y por lo tanto deben estar conectados a cada uno en particular.

Adaptabilidad

La Adaptabilidad es la facilidad con la cual puede adaptarse el sistema a los cambios, donde los requisitos son un factor de cambios de la aplicación.

Comprensibilidad

La información debe ser inteligible, fácil de comprender y accesible por los usuarios. Es el caso, del estilo de código empleado por los programadores, el cual debe representarse de forma clara.

Testabilidad

Un sistema testeable o verificable es aquel que en un proceso finito y no costoso obtiene resultados esperados.

Facilidad de Uso

La definición insiste en los diferentes niveles de experiencia de los posibles usuarios. Este requisito plantea uno de los mayores retos de los diseñadores de software preocupados por la facilidad de uso: cómo proporcionar explicaciones y guías detalladas a los usuarios novatos sin fastidiar a los usuarios expertos que quieren ir directo al grano. Una de las claves de la facilidad de uso es la simplicidad estructural. Un sistema bien diseñado, construido de acuerdo a una estructura clara y bien pensada, tiende a ser más fácil de aprender y usar que uno confuso. Los buenos diseñadores de interfaces siguen una política prudente. Hacen las menos suposiciones posibles sobre los usuarios. Cuando se diseña un sistema interactivo, se debe esperar que los usuarios sean miembros de la raza humana y

que sepan leer, mover un ratón, presionar un botón y teclear (lentamente); no mucho más. Si el software está dirigido a un área especializada de aplicación, se puede dar por supuesto que los usuarios están familiarizados con sus conceptos básicos. Pero incluso esto es arriesgado.

Software

En aplicaciones de software y hardware, los requisitos de software son las características que debe tener el software instalado en una computadora para poder soportar y/o ejecutar una aplicación o un dispositivo específicos. Contrasta con los requisitos de hardware. Los requisitos de software pueden ser: de sistema operativo, aplicaciones específicas instaladas, ciertas aplicaciones no instaladas en el mismo sistema, determinadas configuraciones en el sistema operativo o en ciertas aplicaciones.

Hardware

En aplicaciones de software y hardware, los requisitos de hardware son las características que debe tener el hardware de una computadora para poder soportar y/o ejecutar una aplicación o un dispositivo específicos. Contrasta con los requisitos de software. Los requisitos pueden ser:

- ✓ Requisitos mínimos de hardware, son las características mínimas (mínimo costo, y mínima antigüedad) indispensables para ejecutar la aplicación correctamente.
- ✓ Requisitos recomendados de hardware, son las características más apropiadas que debe tener el hardware para poder ejecutar una aplicación específica.
- ✓ Requisitos de Restricciones en el diseño y la implementación**,** especifica o restringe la codificación o construcción de un sistema, son restricciones que han sido ordenadas y deben ser cumplidas estrictamente.

Desempeño

Se refiere al tiempo mínimo de recuperación ante fallos, el tiempo mínimo para la recuperación ante aciagos fallos debe ser considerablemente corto para no perder la operatividad del sistema durante mucho tiempo. Los procesos y transacción masiva de datos deben estar definidos hacia y desde todas las estaciones de trabajo y sistemas asociados, sin efectos negativos sobre el rendimiento del sistema. Tiempos de respuestas a las funcionalidades: La respuestas a las diferentes funcionalidades solicitadas por el usuario deben estar dada en menos de 3 segundos. Si el Sistema se conecta a una BD, la misma debe permitir ser accedida, modificada , actualizada independientemente del número de

datos que maneje, los tiempos de consulta a la BD deben ajustarse de acuerdo a las funciones invocadas por los usuarios en lapsos de espera que permitan cumplir con la funcionalidad solicitada.

Apariencia o interfaz externa

Este tipo de requisito describe la apariencia del producto. Es importante destacar que no se trata del diseño de la interfaz en detalle sino que especifican cómo se pretende que sea la interfaz externa del producto. También pueden ser necesidades de cumplir con normas estándares, o con los estándares de la empresa para la cual se esté desarrollando el software.

Robustez

La robustez complementa la corrección. La corrección tiene que ver con el comportamiento de un sistema en los casos previstos por su especificación; la robustez caracteriza lo que sucede fuera de tal especificación. La robustez es por naturaleza una noción más difusa que la corrección. Puesto que tiene que ver aquí con casos no previstos por la especificación, no es posible decir, como con la corrección, que el sistema debería "realizar sus tareas" en tal caso. Donde las tareas son conocidas, el caso excepcional formaría parte de la especificación y regresaríamos al terreno de la corrección. Siempre habrá casos que la especificación no se contemple explícitamente. El papel del requisito de robustez es asegurar que si tal caso surgiese el sistema no causará eventos catastróficos; debería producir mensajes de error apropiados y terminar su ejecución limpiamente en lo posible.

Soporte

Brindan al usuario la facilidad de instalación, facilidad de mantenimiento, lo que requiere código y diseño documentado y facilidad de actualización hacia versiones más modernas. Son requisitos orientados al mantenimiento y soporte de la aplicación bajo su puesta en producción. Ellos abarcan: Mantenimiento, Estándares de Codificación e Instalación

Requisitos organizacionales

Entrega

Los requisitos de entrega garantizan la ausencia de errores y malentendidos para solicitar un presupuesto, reducir el plazo de entrega de resultados y es muy conveniente que a la hora de requerir un ensayo, se especifique muy claramente cuál es la causa o la razón por la que se solicita.

Implementación

El requisito de implementación tiene en cuenta varios aspectos como son el tipo de paradigma de programación a utilizar en el sistema, el tipo de entorno sobre el que vas a trabajar, el lenguaje de programación utilizado. En caso de que uses BD, se especificara también, además del servidor con que trabajaras. Además tendrá en cuenta las herramientas con las que modelaras tu sistema, en caso de hacerlo, así como la herramienta (IDE) en la que se va a desarrollar el sistema.

Requisitos externos

Interoperabilidad

Para que la interoperabilidad sea posible es condición primordial que se establezcan criterios de normalización y equivalencia entre los datos de manera que éstos queden identificados y puedan ser tratados de forma automática por los diferentes sistemas. La interoperabilidad no consiste en que se envíen informes y documentos entre los sistemas, sino en que los datos generados en un sistema puedan ser tratados por los otros y manejados de manera fiable por los profesionales que operan con los diferentes sistemas. La interoperabilidad es un reto importante que asegura que aplicaciones creadas con diferentes orígenes podrán integrarse con facilidad para trabajar juntas, algo que en el entorno del software propietario provoca no pocos quebraderos de cabeza y horas de consultoría. Si el entorno del software de código abierto es capaz de trabajar con brillantez en problema de la interoperabilidad, será sin duda un avance importante de cara a la popularización de este tipo de soluciones.

Legales

Describe la serie de reglamentos con los que debe cumplir el sistema. La parte legal y reglamentaria abarca: Reglas de Negocio, Reglamentación Legal.

Seguridad

La seguridad de un sistema no solo tiene en cuenta la seguridad del sistema sino, además, el ambiente en el que se usará el mismo. Por lo que se tiene que contemplar la seguridad física del lugar donde se usa la aplicación, los controles administrativos que se establecen de acceso al sistema y las regulaciones legales que afecta o determina su uso y que serán tenidas en cuenta si se incumple. La seguridad puede ser tratada en tres aspectos diferentes:

- ✓ Confidencialidad: La información está protegida de acceso no autorizado y divulgación. El

sistema debe ser capaz de mantener la calidad del dato de manera que garantice su integridad durante su aplicación. Si el sistema se conecta a una BD (BD), Cada vez que se detecta un cambio en el BD se debe replicar de manera automática los cambios para asegurar que los equipos de respaldo se mantengan actualizados. Si el sistema tiene interfaz con otros sistemas externos, el sistema debe permitir restablecer comunicación con otros sistemas externos tras la ocurrencia de una falla.

- ✓ Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos. Pueden incluir también mecanismos de chequeo de integridad y realización de auditorías.
- ✓ Disponibilidad: Significa que los usuarios autorizados se les garantizarán el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

1.2 Ingeniería de requisitos

“La Ingeniería de Requisitos es el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema”. (10).

En el ciclo de vida de los proyectos de software la Ingeniería de Requisitos es una de las fases más críticas. En los proyectos de software, la IR es fundamental para garantizar los resultados del sistema. Para entender qué es la IR se puede citar algunos conceptos que se han dado en el transcurso de los años por los estudiosos de la materia.

“La IR es la disciplina encargada de establecer los servicios que un sistema debe suministrar y las restricciones bajo las cuales debe operar” (11).

La IR tiene por objetivo la obtención de requisitos del sistema que se desea construir a cierto nivel de abstracción y se cumplen ciertas características, se hace un puente entre el espacio del problema y el espacio de la solución.

En la IR se identifica el propósito del sistema y el contexto en el que será usado. Los requisitos constituyen el enlace entre las necesidades reales de los clientes, usuarios y otros participantes

vinculados al sistema. La IR consiste en un conjunto de actividades y transformaciones que pretenden comprender las necesidades de un sistema software y convertir la declaración de estas necesidades en una descripción completa, precisa y documentada de los requisitos del sistema.

Trata de los principios, métodos, técnicas y herramientas que permiten descubrir, documentar y mantener los requisitos para sistemas basados en computadora, de forma sistemática y repetible.

La definición de las necesidades de un sistema es un proceso complejo, pues en él hay que identificar los requisitos que se deben cumplir para satisfacer las necesidades de los usuarios finales y de los clientes.

El proceso de especificación de requisitos se puede dividir en tres grandes actividades: Captura de Requisitos, Definición de Requisitos y Validación de requisitos.

1.2.1 Captura de requisitos

La Captura de requisitos es la actividad que permite al equipo de desarrollo de un sistema de software extraer de cualquier fuente de información disponible, las necesidades que debe cubrir.

El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa.

Durante la preparación y realización de las sesiones de elicitación de requisitos las técnicas más habituales son las entrevistas, el *Joint Application Development* (JAD) o Desarrollo Conjunto de Aplicaciones, el *brainstorming* / tormenta de ideas y la utilización de *escenarios*, más conocidos como Casos de Uso.

Entrevistas: resultan una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Existen muchos tipos de entrevistas y son muchos los autores que han trabajado en definir su estructura y dar guías para su correcta realización.

Básicamente, la estructura de la entrevista abarca tres pasos: identificación de los

entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista).

- ✓ A pesar de que las entrevistas son esenciales en el proceso de la captura de requisitos y con su aplicación el equipo de desarrollo puede obtener una amplia visión del trabajo y las necesidades del usuario, es necesario destacar que no es una técnica sencilla de aplicar. Requiere que el entrevistador sea experimentado y tenga capacidad para elegir bien a los entrevistados y obtener de ellos toda la información posible en un período de tiempo siempre limitado. Aquí desempeña un papel fundamental la preparación de la entrevista.
- ✓ JAD/Desarrollo conjunto de aplicaciones: esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Está basada en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación *What You See Is What You Get* (WYSIWYG)/ "lo que ves es lo que obtienes" es decir, durante la aplicación de la técnica se trabajará sobre lo que se generará. Tras una fase de preparación del JAD al caso concreto, el equipo de trabajo se reúne en varias sesiones. En cada una de ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación.

  Durante la sesión se discute en grupo sobre estos temas, llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema.

  Esta técnica presenta una serie de ventajas frente a las entrevistas tradicionales, ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado, pero requiere un grupo de participantes bien integrados y organizados.
- ✓ Tormenta de ideas: es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios.

  Como técnica de elicitación de requisitos, la tormenta de ideas puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de elicitación, cuando los requisitos son todavía muy difusos.

  Para una sesión de tormenta de ideas requiere que se seleccione a los participantes y al jefe de la sesión, citarlos y preparar la sala donde se llevará a cabo la sesión. Los participantes en una sesión de tormenta de ideas para elicitación de requisitos son normalmente clientes,

usuarios, ingenieros de requisitos, desarrolladores y, si es necesario, algún experto en temas relevantes para el proyecto. Por tanto, se propone que se utilice la técnica de las entrevistas la cual se va a realizar al personal calificado que domine los temas de redes. Al realizar la técnica escogida a los administradores de red o aquellas personas dentro de la institución con experiencia y conocimientos, se evita que los usuarios o clientes del sistema de software den criterios falsos, pues estos en muchos casos no están adecuadamente preparados para diferenciar si los problemas e inconformidades presentadas en la ejecución diaria del sistema informático se deba a un requisito especifico de la red o a una falla propia del sistema utilizado. Por tanto, si se entrevista o se aplica una las técnicas anteriormente tratadas, los resultados de la misma no van a hacer satisfactorios en el levantamiento de los RNF de redes.

- ✓ *Concept Mapping* / Mapas Conceptuales: son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la ingeniería de requisitos, pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste. Sin embargo, deben ser usados con cautela porque en algunos casos pueden ser muy sugestivos y pueden llegar a ser ambiguos en casos complejos, si no se acompaña de una descripción textual.
- ✓ *Sketches y Storyboards* / (Esbozos y Guiones Gráficos): Está técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno Web. La misma consiste en representar sobre papel en forma muy esquemática las diferentes interfaces al usuario (esbozos). Estos esbozos pueden ser agrupados y unidos por enlaces, dando idea de la estructura de navegación (guiones gráficos).
- ✓ Casos de Uso: Aunque inicialmente se desarrollaron como técnica para la definición de requisitos, algunos autores proponen CU como técnica para la captura de requisitos. Los CU permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como CU) de un sistema. Un CU describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos (personas, otros sistemas.) que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios CU y un caso de uso puede interactuar con varios actores.
- ✓ La ventaja esencial de los CU es que resultan muy fáciles de entender para el usuario o cliente,

sin embargo carecen de la precisión necesaria si no se acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades.

- ✓ Cuestionarios y *Checklists* / (Listas de Chequeo): Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (listas de chequeo). Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista.
- ✓ Comparación de terminología: Uno de los problemas que surge durante la licitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada en forma complementaria a otras técnicas para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), una misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos (contraste).
- ✓ Ingeniería Inversa: su objetivo es obtener información técnica a partir de un producto accesible al público, con el fin de determinar de qué está hecho, qué lo hace funcionar y cómo fue fabricado.

  Es denominada ingeniería inversa porque avanza en dirección opuesta a las tareas habituales de ingeniería, que consisten en utilizar datos técnicos para elaborar un producto determinado. En general si el producto u otro material que fue sometido a la ingeniería inversa fueron obtenidos en forma apropiada, entonces el proceso es legítimo y legal (12).

1.2.2 Definición de requisitos

La definición de requisitos es el proceso donde se define el punto de partida del proyecto, se abordan las necesidades del sistema que se va a construir.

Expresa en lenguaje natural (con diagramas) los servicios y restricciones operacionales del sistema. Se genera con la información proporcionada por el cliente. La definición de requisitos es el proceso donde se define el punto de partida del proyecto, se abordan las necesidades del sistema que se va a construir. Para la realización de dicha actividad existe un gran número de técnicas, a continuación se

describen las más relevantes:

Lenguaje natural: Resulta una técnica muy ambigua para la definición de los requisitos. Consiste en definir los requisitos en lenguaje natural sin usar reglas para ello. A pesar de que son muchos los trabajos que critican su uso, es cierto que a nivel práctico se sigue utilizando.

Glosario y ontologías: La diversidad de personas que forman parte de un proyecto de software hace que sea necesario establecer un marco de terminología común. Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recogen y definen los conceptos más relevantes y críticos para el sistema. En esta línea se encuentra también el uso de ontologías, en las que no sólo aparecen los términos, sino también las relaciones entre ellos.

Plantillas o patrones: Esta técnica, recomendada por varios autores, tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta, menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado estructurado, el trabajo de rellenar las plantillas y mantenerlas, puede ser demasiado tedioso.

Escenarios: La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia una representación gráfica en forma de diagramas de flujo. El análisis de los escenarios, hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales del sistema.

Casos de uso: Como técnica de definición de requisitos han sido aceptados ampliamente los CU. Actualmente se ha propuesto como técnica básica del *Rational Unified Process* (RUP) / El Proceso Unificado Racional. Sin embargo, son varios los autores que defienden que pueden resultar ambiguos a la hora de definir los requisitos, por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.

Lenguajes Formales: Otro grupo de técnicas que merece la pena resaltar como extremo opuesto al lenguaje natural, es la utilización de lenguajes formales para describir los requisitos de un sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción formal, han sido aplicadas en el mundo de la ingeniería de requisitos desde hace años. Sin embargo, resultan muy complejas en su utilización y para ser entendidas por el cliente. El mayor inconveniente es que no favorecen la

comunicación entre cliente y analista. Por el contrario, es la representación menos ambigua de los requisitos y la que más se presta a técnicas de verificación automatizadas. (13)

1.2.3 Validación de requisitos

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos determina realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias.

Aún así, existen algunas técnicas que pueden aplicarse para ello:

Reviews / (Revisión): Está técnica consiste en la lectura y corrección completa de la documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar consistencias de la documentación o información faltante.

Auditorías: La revisión de la documentación con esta técnica consiste en un chequeo de los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada.

Matrices de trazabilidad: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.

Prototipos: Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final. (14) (15).

1.3 Inteligencia artificial

Actualmente el desarrollo de la computación se ha tornado imprescindible para la vida humana, es necesaria en casi todas las esferas en las que el hombre se desenvuelve. Ha propiciado el surgimiento de nuevas ciencias. Una de ellas es la Inteligencia Artificial (IA), que se ha convertido en las últimas

décadas en la más popular de todas, propiciando un acelerado incremento de la utilidad de los sistemas de cómputo.

La IA es un área de la investigación donde se combinan las computadoras, la fisiología y filosofía, reuniendo varios campos como la robótica y los Sistemas Expertos (SE), los cuales tienen en común la creación de máquinas que pueden pensar por medio de algoritmos. Básicamente lo que pretende la IA es crear una máquina secuencial programada que repita indefinidamente un conjunto de instrucciones generadas por un ser humano. Esta ciencia tiene implícitas varias ramas y dentro de ellas una de las más populares en la actualidad son los SE por la gran cantidad de aplicaciones que tienen. Una de las ramas de la IA es el Aprendizaje Automático, el cual se expondrá en el próximo epígrafe.

1.4 Aprendizaje automático

El Aprendizaje Automático es una rama de la IA cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. En muchas ocasiones el campo de actuación del Aprendizaje Automático se solapa con el de la Estadística, ya que las dos disciplinas se basan en el análisis de datos. Sin embargo, el Aprendizaje Automático se centra más en el estudio de la Complejidad Computacional de los problemas.

El Aprendizaje Automático tiene una amplia gama de aplicaciones, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos y robótica.

En la presente investigación se analizarán las siguientes técnicas de aprendizaje automático: los SE, Sistemas Basados en Reglas (SBR) y Sistemas Basados en Casos (SBC), las cuales permitirán realizar una selección conveniente para el desarrollo del sistema que se desea implementar para la captura de RNF.

1.4.1 Sistema de expertos

Se considera a alguien “experto” en un problema cuando este individuo tiene conocimiento especializado sobre dicho problema. En el área de los SE a este tipo de conocimiento se le llama

conocimiento sobre el dominio. La palabra dominio se usa para enfatizar que el conocimiento pertenece a un problema específico. (16)

Antes de la aparición del ordenador, el hombre ya se preguntaba si se le arrebataría el privilegio de razonar y pensar. En la actualidad existe un campo dentro de la inteligencia artificial al que se le atribuye esa facultad: el de los sistemas expertos. Estos sistemas también son conocidos como Sistemas Basados en Conocimiento, los cuales permiten la creación de máquinas que razonan como el hombre, restringiéndose a un espacio de conocimientos limitado. En teoría pueden razonar siguiendo los pasos que seguiría un experto humano (médico, analista, empresario, etc.) para resolver un problema concreto. Este tipo de modelos de conocimiento por ordenador ofrece un extenso campo de posibilidades en resolución de problemas y en aprendizaje. Su uso se extenderá ampliamente en el futuro, debido a su importante impacto sobre los negocios y la industria.

El conocimiento del experto se obtiene de alguna forma (ingeniería del conocimiento) y se organiza en una base de conocimientos, y en función de los datos disponibles de la aplicación (base de hechos o BD) se imita la forma de actuar del experto explorando en la base de conocimientos hasta encontrar la solución (motor de inferencia)

Ingeniería del conocimiento

Al proceso mediante el cual se genera dicha BD se le llama ingeniería del conocimiento: método apropiado que selecciona y organiza la información que contiene el o los SE. Todos los conocimientos que se obtienen deben ser estructurados de una forma correcta, todo este conocimiento se almacena en lo que se conoce como la base de conocimientos.

Base de Conocimientos

La base de conocimientos contiene el conocimiento especializado extraído del experto en el tema. Pero no es suficiente con conocer la base de conocimientos para entender la estructura y el funcionamiento de los SE, otro de los componentes esenciales de estos sistemas, es la BD o base de hechos.

Base de Datos o Base de Hechos

Contiene conocimiento sobre el Caso concreto en que se trabaja. También se registrarán en ella las conclusiones intermedias y los datos generados en el proceso de inferencia. Todos estos datos no son

suficientes, si no se tiene un sistema encargado de procesar y manipular toda la información para generar los resultados deseados, este sistema es conocido como Motor de inferencia.

Motor de Inferencia

El motor de inferencia es el "supervisor", un programa que está entre el usuario y la base de conocimientos, y que extrae conclusiones a partir de los datos simbólicos que están almacenados en las bases de hechos y de conocimiento. Dependen en gran medida de la representación elegida.

1.4.1.1 Ventajas y limitaciones de los sistemas expertos

Estos programas proporcionan la capacidad de trabajar con grandes cantidades de información, que son uno de los grandes problemas que enfrenta el analista humano que puede afectar negativamente a la toma de decisiones pues el analista humano puede depurar datos que no considere relevantes, mientras un SE debido a su gran velocidad de proceso analiza toda la información incluyendo las no útiles para de esta manera aportar una decisión más sólida.

Es evidente que para actualizar se necesita de reprogramación de estos (tal vez este sea una de sus limitaciones más acentuadas) otra de sus limitaciones puede ser el elevado costo en dinero y tiempo, además que estos programas son poco flexibles a cambios y de difícil acceso a información no estructurada.

Debido a la escasez de expertos humanos en determinadas áreas, los SE pueden almacenar su conocimiento para cuando sea necesario poder aplicarlo. Así mismo los SE pueden ser utilizados por personas no especializadas para resolver problemas. Además si una persona utiliza con frecuencia un SE aprenderá de él.

Por otra parte la inteligencia artificial no ha podido desarrollar sistemas que sean capaces de resolver problemas de manera general, de aplicar el sentido común para resolver situaciones complejas ni de controlar situaciones ambiguas.

1.4.2 Sistemas basados en reglas

En nuestra vida diaria encontramos muchas situaciones complejas gobernadas por reglas deterministas: sistemas de control de tráfico, sistemas de seguridad, transacciones bancarias.

Los sistemas basados en reglas son herramientas para tratar estos problemas. Las reglas deterministas constituyen la más sencilla de las metodologías utilizadas en sistemas expertos. La base de conocimiento contiene las variables y el conjunto de reglas que definen el problema, y el motor de inferencia obtiene las conclusiones aplicando la lógica clásica a estas reglas. Por regla se entiende una proposición lógica que relaciona dos o más objetos e incluye dos partes, la premisa y la conclusión. Cada una de estas partes consiste en una expresión lógica con una o más afirmaciones objeto-valor conectadas mediante los operadores lógicos y, o, o no. Una regla se escribe normalmente como: Si premisa, entonces conclusión".

1.4.3 Sistemas basados en casos.

Los SBC constituyen una de las más actuales tecnologías para construir Sistemas Basados en el Conocimiento para la toma de decisiones. Estos Sistemas utilizan el Razonamiento Basado en Casos como método de solución de problemas para resolver nuevas situaciones.

Es un paradigma de resolución de problemas, pero son precisamente las diferencias con el resto de los acercamientos de la inteligencia artificial las que lo hacen tan especial. En lugar de confiar únicamente en el conocimiento general del dominio del problema, o realizar asociaciones a lo largo de relaciones entre descripciones del problema y conclusiones, este paradigma es capaz de utilizar conocimiento específico de experiencias previas, es decir, situaciones de un problema concreto (Casos).

No existe consenso sobre qué información exactamente se debe representar en un Caso. Un Caso es un conjunto arbitrario de rasgos usado para describir un concepto particular. El conjunto de problemas resueltos (que pudieron ser resueltos con éxito o no) forma la Base de Casos o memoria permanente del sistema. En la memoria permanente se registran los problemas (Casos) resueltos y debe poseer una organización que le permita mostrar cualidades similares a la memoria humana, es decir: ser ilimitada. En la medida que la memoria crezca no se puede hacer más lenta. Debe permitir buscar directamente los elementos de memoria que sean relevantes a un problema.

Lo esencial para el trabajo de un SBC es que todos los Casos relevantes (similares, semejantes) al nuevo problema puedan recuperarse eficientemente desde la Base de Casos. Esto depende directamente de la forma en que están organizados los Casos en la base, a esa forma se le denomina

modelo de la Base de Casos, y a continuación se hará referencia a varios de estos. Se han propuesto diversos enfoques para modelar la Base de Casos, entre ellos están:

a) Organización de los Casos en una estructura plana.

b) Organización de los Casos en una estructura jerárquica compartida.

c) Organización de los Casos en redes de discriminación.

De los Sistemas Basados en Casos se tienen las siguientes ventajas:

- ✓ Adquisición de conocimiento: Razonan desde episodios específicos, lo cual evita el problema de descomponer el conocimiento del dominio y generalizarlo en reglas. Adicionalmente, los especialistas del dominio frecuentemente se resisten más a conformar un conjunto de reglas del dominio que a contar las historias de los casos que han encontrado.
- ✓ Flexibilidad en la representación del conocimiento: Pueden explotar múltiples tipos de conocimiento. El conocimiento reflejado se concentra no sólo en la representación de los Casos, sino también en la forma de su organización, la estrategia de recuperación y la estrategia de adaptación. Esto le brinda al diseñador del sistema mayor flexibilidad para escoger la mejor alternativa para representar el conocimiento requerido.
- ✓ Mantenimiento del conocimiento: Un usuario puede ser capaz de agregar nuevos Casos a la Base de Casos sin la intervención de los expertos.
- ✓ Aumento de la eficiencia al resolver un problema: La reutilización de las soluciones previas al resolver un problema, aumenta la eficiencia. El hecho de que se almacenen un Casos que resultó un fracaso permite advertir sobre problemas potenciales a evitar.
- ✓ Aumento de la calidad de las soluciones: Las soluciones sugeridas por Casos en dominios no entendidos completamente son más precisas que aquellas que surgieron por cadenas de reglas generalizadas. Esto es porque los Casos reflejan lo que realmente sucede.
- ✓ La aceptación del usuario: Las soluciones derivadas a partir del Razonamiento Basado en Casos están fundamentadas en Casos reales. Esto le permite al sistema justificar las decisiones al usuario.

Desventajas de los Sistemas basados en Casos:

- ✓ Añadir Casos requiere modificar la red.
- ✓ Mantener la red en forma óptima es costoso.
- ✓ Algunos Casos pueden no ser considerados dado el orden de las preguntas.
- ✓ También se pueden señalar otras desventajas, como pueden ser la dificultad para encontrar una estructura apropiada para describir el contenido de un Caso y decidir cómo la memoria de Casos debe ser organizada e indexada para un almacenamiento, recuperación y reutilización efectiva y eficiente, o que cuando el modelo del conocimiento general del dominio es incompleto, presenta información imprecisa, etc., resulta difícil su integración a la estructura de la Base de Casos. Estas desventajas inciden directamente en varios problemas que caracterizan a los SBC.

1.5 Sistema operativo GNU/Linux: Distribución GNU/Linux

Este proyecto será desarrollado a partir de las herramientas que brinda el software libre. Buscando la independencia tecnológica que este brinda al posibilitar la libertad de uso y distribución de los programas sin incurrir en litigios de licenciamiento o asuntos legales.

El Proyecto *Debian* es una asociación de personas que han hecho causa común para crear un Sistema Operativo (SO) libre, bajo la *GNU General Public License* (GNU GPL*)* / La Licencia Pública General de GNU. Basado en el conocido y distribuido núcleo de Linux la distribución es llamada Debian.

Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU; de ahí el nombre: GNU/Linux.

Debian es la única distribución importante de GNU/Linux mantenida solamente por voluntarios, es decir, sin un enfoque comercial, esto tiene ventajas y desventajas.

En primer lugar, las personas que se dedican a *Debian* tienen una alta motivación en participar en la misma, y se actualiza la distribución diariamente, apareciendo paquetes nuevos de software constantemente. Al mismo tiempo, existe un compromiso de calidad, no se desea distribuir software con errores.

En segundo lugar, dada su actitud abierta a la participación de todos, en el mismo espíritu original de Linux, constantemente hay personas que se unen a *Debian* para participar aportando su granito de arena, no solamente haciendo paquetes de programas, sino colaborando en el Servidor de Web,

traduciendo documentación de *Debian*, documentando fallos, o ayudando a los usuarios a través de las listas de correo que mantiene la comunidad.

Como desventajas tiene un mayor componente técnico que otras distribuciones. También, es posible que ciertos paquetes no estén tan actualizados como debieran, quizás porque sus desarrolladores han dejado de actualizarlos y nadie se ha hecho cargo. Sin embargo esto es algo que todos los desarrolladores tratan de evitar y, aunque cada desarrollador mantiene sus paquetes, no es raro que otro desarrollador (incluso un usuario) envíe una nueva versión del paquete para arreglar un problema o actualizarlo.

Se decide usar la distribución *Debian* porque es una (distribución de GNU/Linux) de desarrollo muy estable, por lo que los paquetes que se desarrollen en él y quieran ejecutarse utilizando cualquier distribución siempre serán estables. A diferencia de otras distribuciones tiene un magnífico soporte de estabilidad en las aplicaciones (no requieren ser compiladas en la máquina que las esté usando). Los módulos del *Lighweight Directory Access Protocol* (LDAP) / Protocolo Ligero de Acceso a Directorios se pueden ejecutar sin problemas permitiendo que los usuarios usen sus sesiones en cualquier máquina dentro del área de trabajo, ahorrando recursos de hardware.

1.6 Metodología de desarrollo de software: OpenUP

El proceso de gestión *Open Unified Process* (OpenUP) / (Proceso Unificado Abierto), es una parte del marco de desarrollo Eclipse (*open source (*código abierto*)*, y desarrollado por la fundación Eclipse). Fue liberado por el Eclipse Process Framework (EPF). Se construyó sobre una donación realizada por *International Business Machines* (IBM) del Basic Unified Process (BUP) / Proceso Unificado Básico. Fue entregada a Eclipse a fines de 2005 y renombrado como OpenUP en 2006.

El OpenUP es un Proceso Unificado que aplica propuestas de gestión ágil como son desarrollo iterativo e incremental dentro del ciclo de vida, tratando de ser manejable en relación con el RUP, ya que mantiene sus características esenciales. Debemos estar claros que el uso de OpenUP se debe realizar cuando se tiene un equipo pequeño, cuando se quiere evitar ser cargado excesivamente con metodologías formales improductivas.

OpenUP es un proceso de desarrollo de software completo ya que puede ser manifestado como todo el proceso para construir un sistema. Es extensible ya que en el proceso se puede agregar o adaptar según lo vayan requiriendo los sistemas, sin embargo solo está incluido lo fundamental (menos de 20

artefactos). OpenUP es ágil, ligero y proporciona una comprensión detallada del proyecto, beneficiando a clientes y desarrolladores sobre productos a entregar y su formalidad. Se centra en una arquitectura temprana para reducir al mínimo los riesgos y organizar el desarrollo. Es la metodología utilizada por desarrolladores de alto nivel en todo el mundo por sus altas cualidades administrativas. Su desarrollo es dirigido por CU.

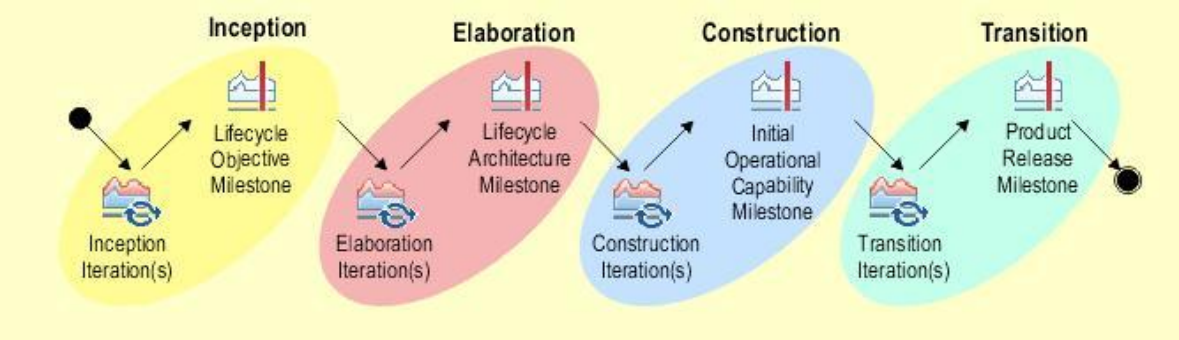


Fig. NO. 2 Ciclo de vida de la metodología OpenUP

1.7 Herramienta de modelado UML: Visual Paradigm

Visual Paradigm para *Unified Modeling Language* (UML) / Lenguaje Unificado de Modelado es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Algunas de las características que presenta son:

Interoperabilidad con modelos UML 2.0 (meta modelos UML 2.x para plataforma Eclipse) a través de XMI, esta es una nueva característica. Ingeniería inversa: código a modelo, código a diagrama. Generación de código - Modelo a código, diagrama a código. Generación de código: modelo a código, diagrama a código, entre otras.

1.8 Lenguaje de programación: C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. C++ permite trabajar tanto a alto como a bajo nivel.

Se escoge este lenguaje de programación porque C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, y por ser un lenguaje orientado y multiparadigma.

1.9 Biblioteca QT

Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario. Es utilizada en *K Desktop Environment* (KDE) / un entorno de escritorio para sistemas como GNU/Linux. Utiliza el lenguaje de programación C++ de forma nativa y además existen *bindings / adaptaciones* para C, Python (PyQt), Java (Qt Jambi), Perl (PerlQt), Gambas (gb.qt), Ruby (QtRuby), PHP (PHP-Qt) y Mono (Qyoto) entre otros lenguajes de programación.

El API de la biblioteca cuenta con métodos para acceder a Bases de Datos mediante *Structured Query Language* (SQL)/ Lenguaje de consulta estructurado, así como uso de de *Extensible Markup Language* (XML)/ Lenguaje de Marcas Extensible y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.

1.10 Eclipse

Eclipse es una herramienta muy potente y de libre distribución, es una plataforma de herramientas universales, un Entorno Integrado de Desarrollo (IDE) abierto y extensible.

Pese a que Eclipse esté escrito en su mayor parte en Java (salvo el núcleo), se ejecute sobre máquina virtual de ésta y su uso más popular sea como un IDE para Java, Eclipse es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, etc.

La característica clave de Eclipse es la extensibilidad. Eclipse es una gran estructura formada por un núcleo y muchos plugins / complementos que van conformando la funcionalidad final. La forma en que los plugins interactúan es mediante interfaces o puntos de extensión; así, las nuevas aportaciones se integran sin dificultad ni conflictos.

Se decide utilizar este IDE debido a que hay una gran documentación sobre cómo integrar QT con Eclipse y C++, además por la condición de ser un IDE muy potente y de libre distribución y estar considerado como una plataforma universal.

Capítulo 2: Características y diseño del sistema

En el presente capítulo se realiza una descripción del sistema, donde se muestra, el alcance del mismo, así como la técnica de aprendizaje propuesta, se muestra la especificación de los requisitos, el diagrama de CU del sistema con su descripción, los diagramas de secuencia, la arquitectura del sistema, y los diagramas de clases con sus descripciones.

2.1 Alcance del sistema.

Debido a que no se reutiliza la experticia de los analistas en proyectos anteriores, la inexistencia de una herramienta que agilice la identificación y gestión de los RNF, en el PHA.

El sistema propuesto será capaz de capturar los RNF para el PHA. Para lograrlo se tienen árboles de decisiones, donde cada árbol es un requisito, (Anexo 5), esto permite una mayor facilidad a la hora de capturar y mostrar al usuario los RNF. Se crea una base de conocimientos con dichos árboles, y así a través de la técnica utilizada el experto realiza la captura de requisitos. El sistema una vez que captura todos los RNF, esto se convierte en un Caso, lo guarda en la base de Casos y lo muestra en una plantilla para su uso y además el sistema será capaz de reutilizar un Caso anterior.

2.2 Técnica de aprendizaje propuesta

Una vez analizada las distintas técnicas utilizadas en los SE como son los SBC y SBR, los cuales son los más comunes, se han recopilado conocimientos de los mismos. Debido a que no existen herramientas a nivel mundial que faciliten la captura de RNF, no se puede utilizar básicamente una en particular, por lo que se crea un nueva herramienta la cual permite la captura de los RFN, lo que es de gran importancia, ya que el resultado de la misma será utilizado en una de las fases más importantes en el proceso de desarrollo de Software, que es la captura de requisitos, por lo que va ser de gran ayuda para los Analista de requisitos del PHA.

Para el desarrollo de esta herramienta se han propuesto las dos técnicas mencionadas SBC y SBR.

Se decide utilizar el SBC debido a que el sistema una vez que realiza una captura de RNF esta se convierte en un Caso específico, el cual se guarda para un futuro uso, esto contribuye al mantenimiento del conocimiento. La posibilidad de recuperar un Caso anterior, aumenta la eficiencia

desde el punto de vista de la rapidez en la captura de los RNF, lo cual es uno de los objetivos para lo que fue creada la herramienta. El hecho de que se almacenen Casos que resultaron ser un fracaso permite advertir sobre problemas potenciales a evitar. Lo esencial de un SBC es que los Casos similares puedan ser recuperados de manera eficiente desde la Base de Casos, esto depende directamente de la forma en que están organizados los Casos en la base, generalmente están organizados de esta manera:

a) Organización de los Casos en una estructura plana.

b) Organización de los Casos en una estructura jerárquica compartida.

c) Organización de los Casos en redes de discriminación.

En la herramienta realizada se obtienen los Casos Similares a través de los proyectos a los cuales él esté vinculado , o sea los Casos que contenga un proyecto van a ser similares para ese proyecto, de forma tal que el experto que va a trabajar con la herramienta podrá recuperar cualquiera de los Casos tratados anteriormente. Otro de los motivos por el cual se decide utilizar esta técnica es porque las soluciones derivadas a partir del Razonamiento Basado en Casos están basadas en Casos reales y esto permite justificar las decisiones al cliente o analista que lo utilice.

El objetivo de la herramienta es capturar los RNF y una de las vías que presenta la misma es a través de una encuesta, la cual da la impresión de que el cliente o el analista se está comunicando con un experto en el tema, para lograr esto se recurrió a los SBR, los cuales están formados por los datos (hechos o evidencia) y el conocimiento (el conjunto de reglas almacenado en la base de conocimiento) y un motor de inferencia que usa a ambos para obtener nuevas conclusiones o hechos. En la herramienta los datos están almacenados en una BD en la cual aparecen una serie de preguntas y respuestas que al ser seleccionadas por el cliente o analista se va generando la base de conocimientos, por lo que en la herramienta no están definidas las reglas como tal, debido a que la información referente a los RNF es muy versátil, o sea puede cambiar con facilidad. El motor de inferencia va a funcionar a través de la forma en que están organizados los datos en la BD; la misma está organizada tipo árbol de decisión, donde sus nodos serán las preguntas, y sus respuestas serán capturadas en aquellas que fueron afirmativas. Esta forma de modelar la BD permite que todos los nodos puedan ser visitados, además cumple con la comunicación hombre-sistema, la cual va a ser fundamental a la hora de capturar los RNF.

2.3 Especificación de requisitos.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Para el sistema la aplicación que se está desarrollando, ellos son:

RF1 El sistema debe permitir al analista seleccionar los requisitos no funcionales de acuerdo a las características de su proyecto y basados en las categorías de los mismos.

RF1.1 El sistema debe ser capaz mostrar el listado de los RNF.

RF1.2 El sistema debe mostrar el significado de algunas preguntas de difícil comprensión.

RF1.3 El sistema debe ser capaz de mostrar una sugerencia al cliente en caso de que el no sepa tomar una decisión al leer una pregunta.

RF1.4 El sistema debe permitir al analista insertar una nueva respuesta en la BD.

RF1.5 El sistema deber ser capaz de guardar reporte de los RNF.

RF1.6 El sistema deber permitir guardar un Caso en la BD

RF1.7 El sistema de debe ser capaz de reutilizar un Caso anteriormente registrado.

RF2 El sistema debe permitir registrar participantes en la encuesta.

Los requisitos no funcionales son propiedades o cualidades que el producto debe cumplir. Para el sistema son los siguientes:

Usabilidad

CSUS1 La Herramienta luego de instalada, deberá ejecutarse con calidad en cualquier plataforma.

Confiabilidad

CSFI1 El Sistema debe ser capaz de mantener la calidad del dato de manera que garantice su integridad durante su aplicación, procesamiento y almacenamiento en la BD.

CSFI2 Cada vez que se detecte un cambio en la BD se debe replicar de manera automática los cambios.

Funcionamiento

CSFU1 La BD debe permitir ser accedida, modificada, actualizada independientemente del número de datos que maneje.

Soporte

CSSO1 El Sistema debe ser:

- ✓ De fácil instalación, configuración y puesta en marcha.
- ✓ De arquitectura abierta y distribuida, modular, de capacidad escalable.
- ✓ Programado orientado a objeto.

CSSO2 El Sistema deberá ejecutarse sobre cualquier plataforma y utilizará PostgreSQL como Sistema gestor de BD.

Rendimiento

CSRE1 Los tiempos de respuesta del Sistema serán de corta duración.

Interfaces del sistema

IU1 El diseño del Sistema debe estar enfocado a la interfaz de GNOME, debe ser sencillo y funcional, de fácil operación, basado en ventanas y de rápida adaptación para los analistas.

Coacciones del sistema

Implementación

CSIM1 El Sistema debe ejecutarse en diversas plataformas de hardware y software.

CSIM2 El Sistema se realizará usando el lenguaje de programación C++, para la realización de la interfaz gráfica se utilizará la biblioteca gráfica QT 4 y como herramienta para programación en

Eclipse. Para la modelación de los diagramas UML se utilizará el Visual Paradigm y como metodología de desarrollo de software el OpenUP.

CSIM3 El Sistema debe cumplir con los lineamientos necesarios para la producción de software libre y la comunidad de desarrollo y soporte.

Portabilidad

CSPO1 El Sistema debe ser multiplataforma, debe ejecutarse de manera óptima sin importar el Sistema operativo que utilice el usuario.

Seguridad

CSSE1 El Sistema debe garantizar que la información sea registrada, visualizada, eliminada y actualizada de forma segura.

Para la especificación de los requisitos remitirse al documento "Especificación de Requisitos".

2.4 Diagrama de casos de uso

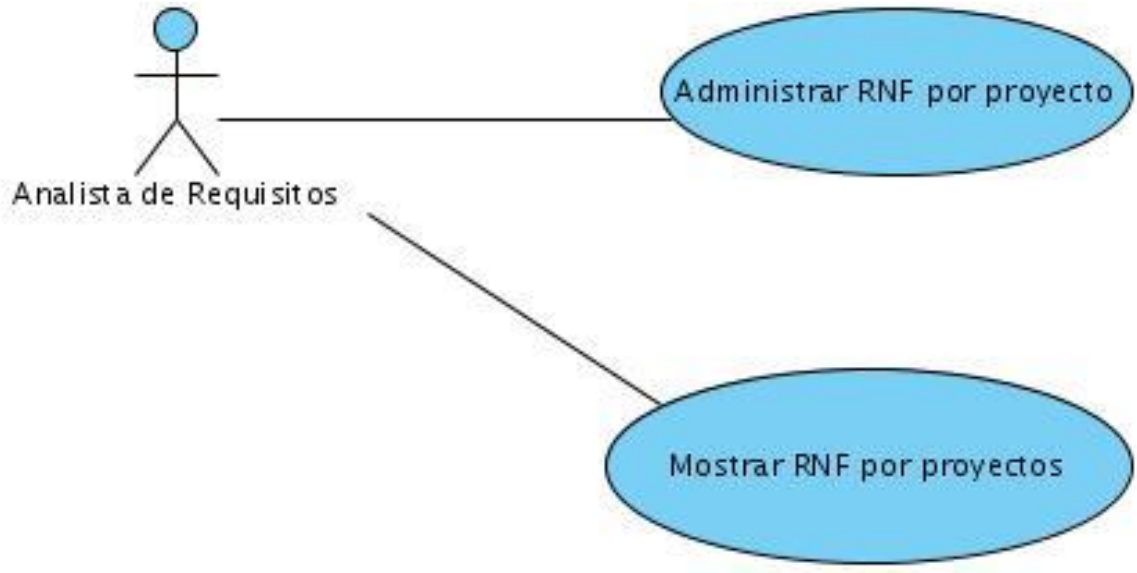


Fig.3 Diagrama de Casos de uso

2.5 Descripción de los casos de uso

Caso de Uso: Administrar RNF por proyecto

1 Breve Descripción

Este Caso de Uso le permite al Analista de Requisitos administrar los RNF del proyecto, realizando las actividades siguientes: Registrar datos, seleccionar RNF, listar RNF, consultar significado de la pregunta , sugerencia de respuesta a la pregunta, insertar una nueva respuesta, guardar reporte de los RNF, guardar el nuevo caso en la BD.

2 Breve Descripción de los Actores

2.1 Analista de requisitos

El Analista de Requisitos es el responsable de la captura de los RNF del Sistema.

3 Precondiciones

- ✓ Debe existir en la BD las preguntas y respuestas de las características generales de proyectos productivos, así como RNF asociados, de forma tal que permita al Analista de Requisitos la realización de la encuesta.
- ✓ Se sugiere estar presente para la realización de la encuesta el/los Cliente (s) junto al/los Analista(s).

4 Flujo Básico de Eventos

4.1 Seleccionar Requisito No Funcional

4.1.1 El Sistema le muestra al Analista de Requisitos un menú con dos opciones:

- ✓ Iniciar Captura de RNF para un Nuevo Proyecto.
- ✓ Consultar RNF de proyectos anteriores.

4.1.2 El Analista de Requisitos selecciona la opción “Iniciar Captura de RNF para un Nuevo Proyecto”.

4.1.3 El Sistema muestra una interfaz para registrar los siguientes datos:

- ✓ Nombre del Proyecto. Nombre del Proyecto Productivo al que se le realizará la captura de RNF. (Campo Obligatorio)
- ✓ Nombre del Analista. Nombre del Analista que desarrollará la encuesta. (Campo Obligatorio)
- ✓ Fecha. Fecha de realización de la encuesta. (Campo Automático)

4.1.4 El Analista de Requisitos registra los datos y selecciona la opción “Registrar” [T]

4.1.5 El Sistema verifica la completitud de los datos (Alterno 5.1) y guarda dicha información.

4.1.6 El Sistema comienza la encuesta mostrando las preguntas y posibles respuestas correspondientes a las características generales de los proyectos, organizadas según las Categorías de RNF, dichas preguntas deben ser contestadas en función de dar respuesta a las necesidades del Cliente.

4.1.7 El Analista de Requisitos contesta correctamente las preguntas realizadas. (Alterno 5.2) (Alterno 5.3) (Alterno5.4).

4.1.8 El Sistema muestra los RNF correspondientes a cada respuesta emitida, indicándole al final de la encuesta un mensaje de “Se ha culminado la Selección de RNF con éxito”.

4.1.9 El Sistema muestra el listado de RNF.

4.2 Guardar caso de RNF

4.2.1 El Analista de Requisitos selecciona “Guardar” el caso con todos los RNF seleccionados. [T]

4.2.2 El Sistema guarda el nuevo Caso tratado.

4.3 Guardar listado como

4.3.1 El Analista de Requisitos selecciona “Guardar como” el listado de los RNF.

4.3.2 El Sistema guarda el listado de RNF en formato (.pdf), con el nombre deseado y la ruta donde se almacenará.

4.4 Fin del caso de uso

4.4.1 El Analista de Requisitos solicita al Sistema cerrar la aplicación.

4.4.2 El Sistema cierra la aplicación.

5 Flujos alternos

5.1 Error al guardar información por datos incompletos

5.1.1 El Sistema verifica que existe información incompleta, mostrándole un mensaje de error al Analista de Requisitos.

5.1.2 El Analista de Requisitos acepta el mensaje de error.

5.1.3 El Sistema regresa al paso 4.1.3 del flujo básico.

5.2 Consultar significado de la pregunta

5.2.1 En el paso 4.1.7, el cliente no comprende una pregunta que se le realiza, el Analista de Requisitos selecciona la opción “Significado de la Pregunta”.

5.2.2 El Sistema le muestra una breve explicación de la pregunta.

5.2.3 El Analista de Requisitos selecciona “Aceptar”.

5.2.4 El Sistema retorna al paso 4.1.6 del flujo básico.

5.3 Sugerencia de respuesta a la pregunta.

5.3.1 En el paso 4.17 del flujo básico, el Analista de Requisitos decide no tomar una decisión, pues al consultar el significado de la pregunta no lo comprendió o no quedó satisfecho, entonces selecciona la opción “Sugerencia”.

5.3.2 El Sistema basándose en respuestas emitidas con anterioridad en las encuestas a otros proyectos, le muestra al Analista de Requisitos una Sugerencia de Respuesta a la Pregunta que se le realiza.

5.3.3 El Analista de Requisitos selecciona “Aceptar”.

5.3.4 El Sistema retorna al paso 4.1.6 del flujo básico.

5.4 Insertar nueva respuesta

5.4.1 El Analista de Requisitos mientras realiza la encuesta de Selección de Requisitos No Funcionales, no está de acuerdo (ni él, ni el cliente) con la respuesta mostrada para determinada pregunta, por lo que selecciona la opción “Insertar Nueva Respuesta”.

5.4.2 El Sistema le muestra una interfaz donde debe introducir la respuesta.

5.4.3 El Analista de Requisitos introduce la respuesta y selecciona “Aceptar”.

5.4.4 El Sistema verifica la completitud de los datos (Alterno 5.5).

5.4.5 El Sistema guarda en la BD la información emitida.

5.5 Error al guardar información por datos incompletos

5.5.1 En el paso 5.4.4, el Sistema verifica que existe información incompleta, mostrándole un mensaje de Error al Analista de Requisitos.

5.5.2 El Analista de Requisitos acepta el mensaje de error.

5.5.3 El Sistema retorna al paso 4.1.2 del flujo básico.

6 Poscondiciones

Si el Caso de Uso finaliza correctamente, se logró:

- ✓ Registrar datos.
- ✓ Seleccionar RNF.
- ✓ Listar RNF
- ✓ Consultar significado de la pregunta.
- ✓ Sugerencia de respuesta a la pregunta.
- ✓ Insertar una nueva respuesta
- ✓ Exportar el nuevo Caso.
- ✓ Guardar el nuevo caso en la BD

Si el caso de uso no terminó correctamente se mostró el mensaje de error correspondiente.

Caso de Uso: Mostrar RNF por proyectos

1 Breve descripción

Este caso de uso le permite al Analista de Requisitos ver los RNF de un proyecto dado y reutilizar los mismos, en caso de considerarlo conveniente.

2 Breve descripción de los actores

2.1 Analista de requisitos

El Analista de Requisitos es el responsable de reutilizar los RNF dado un proyecto.

3 Precondiciones

Debe existir en la BD, Requisitos No Funcionales de proyectos anteriores, de forma tal que se puedan re-utilizar.

Se sugiere que el/los Cliente(s) esté(n) junto al/los Analista(s) para la búsqueda de RNF.

4 Flujo Básico de eventos

4.1 Consultar RNF de proyectos anteriores

4.1.1 El Analista de Requisitos selecciona la opción “Consultar RNF de proyectos anteriores”

4.1.2 El Sistema muestra una interfaz con la siguiente información:

- ✓ Proyectos. Nombre de los Proyectos antes analizados.
- ✓ Vista de Casos. Activa la opción “Casos Similares”
- ✓ Casos Similares. Número de los casos existentes con respecto al proyecto seleccionado.

  Listado de los RNF que van a ser reutilizados.

4.1.3 El Analista de Requisitos selecciona todos los campos.

4.1.4 El Sistema le muestra el Listado de RNF, junto a la opción “Re-Utilizar”, donde le permite re-utilizar el Listado de RNF registrado con anterioridad.

4.2 Reutilizar RNF de un proyecto registrado con anterioridad

4.2.1 El Analista de Requisitos selecciona la opción “Re-Utilizar”.

4.2.2 El Sistema le muestra una interfaz con las siguientes opciones:

- ✓ Modificar. Modificar los RNF seleccionados.
- ✓ Guardar sin Modificar. Guardar íntegramente los RNF seleccionados.

4.2.3 El Analista de Requisitos selecciona “Modificar”. Alterno (5.1)

4.2.4 El Sistema muestra una interfaz con los RNF para su modificación y una opción “Mostrar RNF Modificados”.

4.2.5 El Analista de Requisitos modifica los RNF deseados y selecciona la opción “Mostrar RNF Modificados”

4.3 Registrar datos

4.3.1 El Sistema muestra una interfaz con la siguiente información:

- ✓ Nombre del Proyecto. Nombre del Proyecto Productivo al que se le realizará la captura de RNF. (Campo Obligatorio)
- ✓ Nombre del Analista. Nombre del Analista que desarrollará la encuesta. (Campo Obligatorio)
- ✓ Fecha. Fecha de realización de la encuesta. (Campo Automático)

4.3.2 El Analista de Requisitos registra los datos y selecciona la opción “Registrar” [T]

4.3.3 El Sistema verifica la completitud de los datos (Alterno 5.2)

4.4 Guardar listado

4.4.1 El Sistema muestra al Analista de Requisitos las opciones de “Guardar” y “Guardar Como”.

4.4.2 El Analista de Requisitos selecciona “Guardar” (Alterno 5.3) el caso con todos los RNF seleccionados. [T]

4.4.3 El Sistema guarda el Listado de RNF como un nuevo Caso registrado.

4.5 Fin del caso de uso

4.5.1 El Analista de Requisitos solicita al Sistema cerrar la aplicación.

4.5.2 El Sistema cierra la aplicación.

5 Flujos alternos

5.1 Reutilizar RNF de un proyecto registrado con anterioridad sin modificar.

5.2.1 En el paso 4.2.3 el Analista de Requisitos selecciona la opción “Guardar sin Modificar”.

5.2.2 El Sistema retorna al paso 4.4.2 del flujo básico.

5.2 Error al completar datos

5.2.1 El Sistema verifica que existe información incompleta, mostrándole un mensaje de error al Analista de Requisitos.

5.2.2 El Analista de Requisitos acepta el mensaje de error.

5.2.3 El Sistema retorna al paso 4.3.1 del flujo básico.

5.3 Guardar el Listado como PDF

5.3.1 El Analista de Requisitos selecciona "Guardar Como" el listado de los RNF.

5.3.2 El Sistema guarda el listado de RNF en formato (.pdf), con el nombre deseado y la ruta donde se almacenará.

6 Poscondiciones

- ✓ Si el Caso de Uso finaliza correctamente, se logró:
- ✓ Consultar RNF de proyectos anteriores.
- ✓ Re-Utilizar RNF de un Proyecto registrado con anterioridad y Sin Modificar.
- ✓ Registrar Datos.
- ✓ Guardar Caso de RNF.

Si el caso de uso no terminó correctamente se mostró el mensaje de error correspondiente.

2.6 Diagramas de interacción en el diseño

En el diseño orientado a objetos la interacción es el comportamiento que comprende un conjunto de mensajes intercambiados entre un conjunto de objetos dentro de un contexto para lograr un propósito. Un mensaje es la especificación de una comunicación entre objetos (unidad de comunicación entre objetos).

Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones. Los diagramas de interacción muestran cómo se comunican los objetos en una interacción. El UML define dos tipos de diagramas de interacción: los Diagramas de Colaboración y los Diagramas de Secuencia. En este trabajo se realizaron los Diagramas de Secuencia de los CU arquitectónicamente significativos. (Ver Anexo 1 al 4)

2.7 Arquitectura modelo vista controlador.

La arquitectura Modelo Vista controlador (MVC) fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo, genera una representación visual del mismo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia a él.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al mismo. En la figura 5 se muestra cómo funciona el sistema basado en la arquitectura definida.

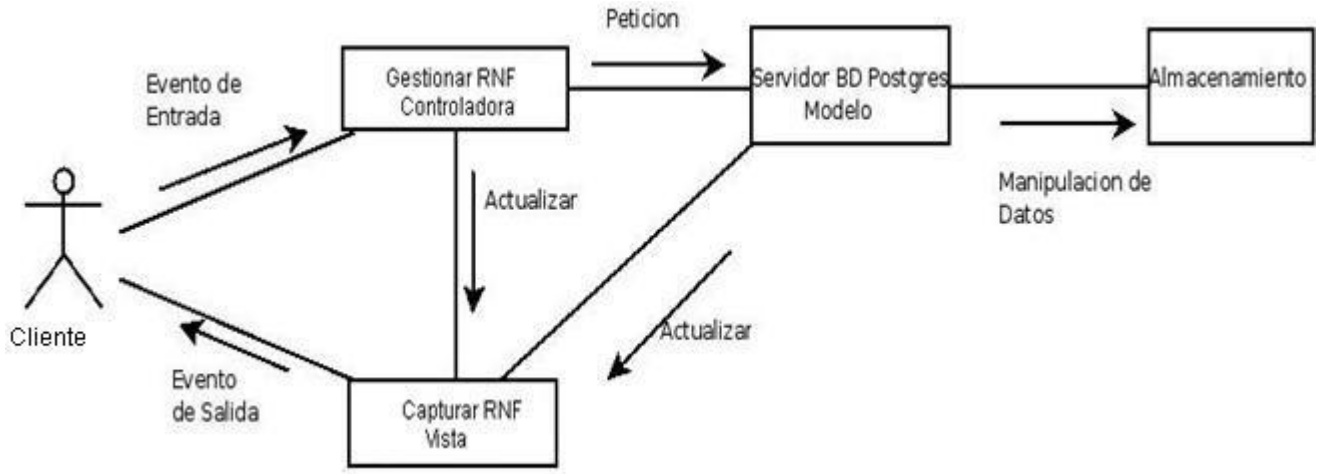


Fig. 4 Arquitectura del sistema

2.8 Modelo de diseño.

Este modelo es el punto de partida de la Implementación. Descompone las tareas de implementación en sub-tareas manejables que, en muchos casos, se pueden realizar de forma simultánea. Es una abstracción del sistema. La estructura del sistema no puede cambiar, aunque sí los detalles de la implementación. Como sus características fundamentales se tiene que es un modelo físico, no genérico y específico para una implementación. Puede tener cualquier número de estereotipos físicos sobre las clases en dependencia del lenguaje en el que se haya decidido implementar la aplicación. Es un modelo dinámico, centrado en la secuencia. Debe ser mantenido durante todo el ciclo de vida del software y finalmente da forma al sistema mientras que intenta preservar la estructura definida por el modelo de análisis lo más posible.

2.8.1 Diagrama de clases del diseño

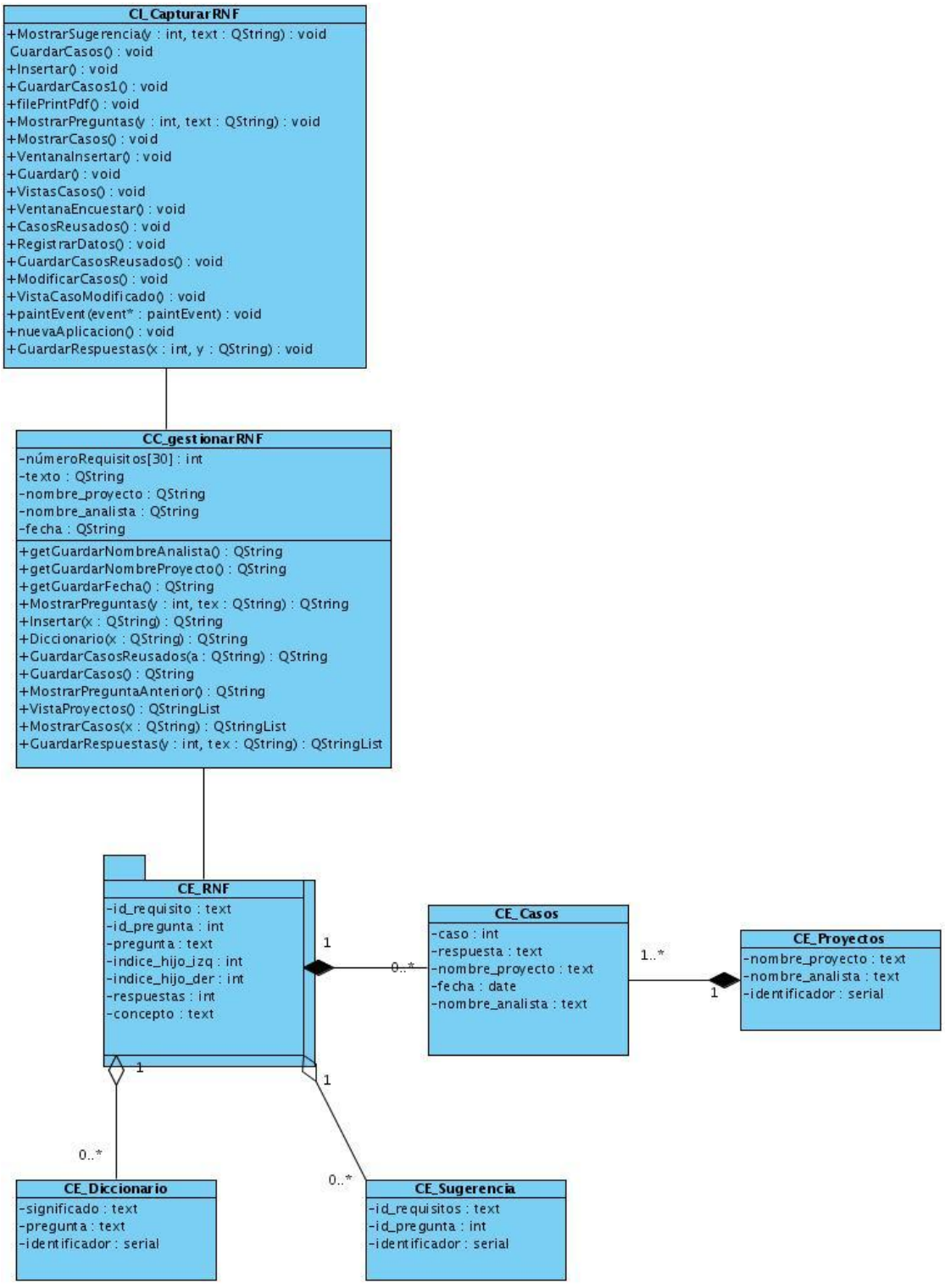


Fig. 5 Diagrama de clases

2.8.2 Descripción de las clases

Nombre: GestionarRNF	
Descripción: Clase que realiza todas las acciones necesarias para la captura de RNF	
Tipo de clase: Controladora	
Atributo	**Tipo**
valoractualIzq	Int
valoractualDer	
nombre_proyecto	text
nombre_analista	text
numeroRequisito[30]	Int
Para cada responsabilidad:	
Nombre:	MostrarPreguntas(y : int, tex : QString) : QString
Descripción:	Método encargado de mostrar las preguntas al cliente o analista.
Nombre:	Insertar(x : QString) : QString
Descripción:	Método encargado de insertar una nueva respuesta a la BD.
Nombre:	GuardarCasos() : QString
Descripción:	Método que permite guardar un Caso en la Base de Casos.
Nombre:	GuardarCasosReusados(a : QString) : QString
Descripción:	Método que permite guardar un Caso anterior en la Base de Casos.
Nombre:	GuardarRespuestas(y : int, tex : QString) : QStringList
Descripción:	Método que permite guardar las respuestas en la BD
Nombre:	MostrarCasos(x : QString) : QStringList
Descripción:	Método que permite mostrar la lista de Casos que tiene un proyecto.

Tabla 1 Descripción de la clase controladora CapturarRNF

Nombre: RNF	
Descripción: Es la clase que realiza la persistencia de los RNF	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_requisito	text
id_pregunta	int
pregunta	text
indice_hijo_izq	int
indice_hijo_der	int
respuestas	text
concepto	text
Para cada responsabilidad:	
Descripción:	
Nombre:	

Tabla 2 Descripción de la clase entidad RNF

Nombre: Casos	
Descripción: Es la clase que realiza la persistencia de los Casos	
Tipo de clase: Entidad	
Atributo	**Tipo**

caso	Int
respuesta	text
nombre_proyecto	text
fecha	date
nombre_analista	text
Para cada responsabilidad:	
Descripción:	
Nombre:	

Tabla 3 Descripción de la clase entidad Casos

Nombre: Proyectos	
Descripción: Es la clase que realiza la persistencia de los Proyectos	
Tipo de clase: Entidad	
Atributo	**Tipo**
nombre_proyecto	text
nombre_analista	text
identificador	serial
Para cada responsabilidad:	
Descripción:	
Nombre:	

Tabla 4 Descripción de la clase entidad Proyectos

Nombre: Sugerencia	
Descripción: Es la clase que realiza la persistencia de las sugerencia que brinda el sistema para una pregunta dada.	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_requisitos id_pregunta identificador	text int serial
Para cada responsabilidad:	
Descripción:	
Nombre:	

Tabla 5 Descripción de la clase entidad Sugerencia

Nombre: Diccionario	
Descripción: Es la clase que realiza la persistencia de la posibilidad de mostrar el significado de una pregunta no entendida.	
Tipo de clase: Entidad	
Atributo	**Tipo**
significado pregunta identificador	text text serial
Para cada responsabilidad:	

Descripción:	
Nombre:	

Tabla 6 Descripción de la clase entidad Diccionario

2.8.3 Modelo entidad-relación

A continuación se presenta el Modelo Entidad-Relación (MER), que opera con los conceptos de entidad y relación, siendo las entidades los elementos que existen y están bien diferenciados entre sí, que poseen propiedades y entre los cuales se establecen relaciones.

La gran ventaja de los MER, está en que puede ser comprendida por personas que no sean especialistas. El número de entidades en una base de datos es por lo general considerablemente menor que el número total de elementos de datos. Así usando el concepto de una entidad como una abstracción para esos objetos del mundo real sobre el cual se desean coleccionar informaciones, se simplifican grandemente las fases de análisis de los requisitos y diseño conceptual

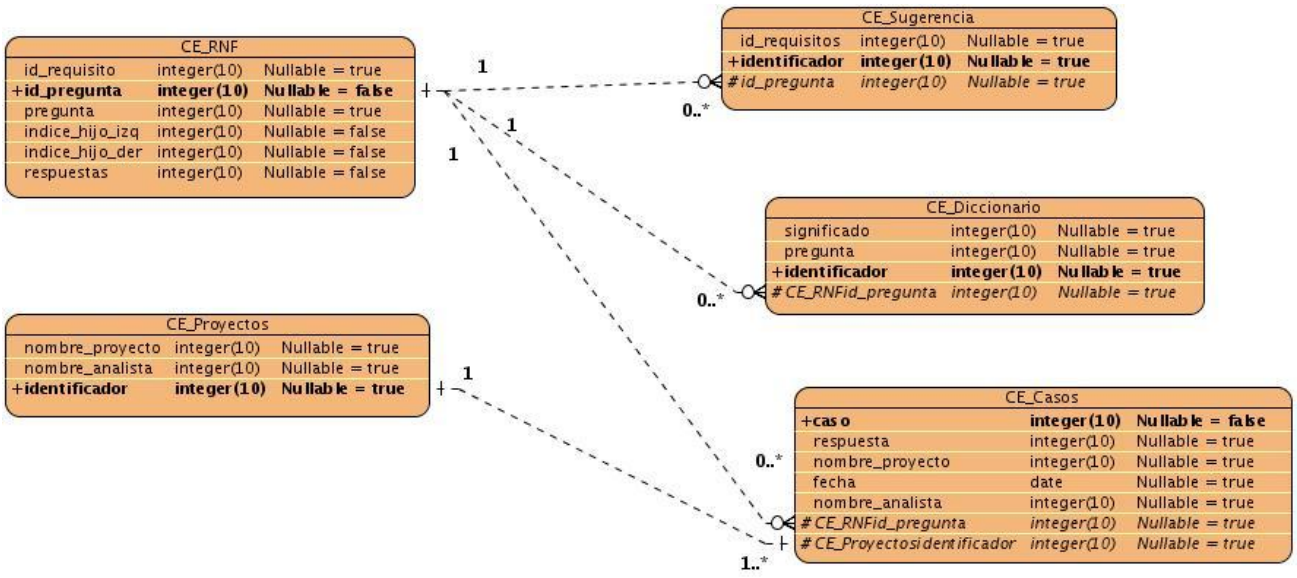


Fig. 6 Modelo entidad-relación

2.9 Diagrama de despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados, es decir, se sitúa el software en el hardware que lo contiene. El sistema está compuesto por una PC cliente donde va a estar la aplicación además de la BD.

Fig.7 Diagrama de despliegue

Capítulo 3: Pruebas al sistema

En el presente capitulo se realiza el proceso de pruebas al sistema para validar las funciones del mismo, para esto se realiza el plan de pruebas, el diseño de casos de pruebas, además se expone un manual de usuario basado en los prototipos funcionales del sistema, también se propone un seguimiento y control al sistema y se muestran los resultados presentes y visones futuras.

3.1 Pruebas de software

Las pruebas de software centran sus objetivos en la detección de errores con el propósito de verificar y revelar la calidad de un producto software. Las pruebas de software, es un proceso usado para identificar posibles fallos de implementación, calidad, o usabilidad del sistema desarrollado.

Pruebas de sistema.

Las Pruebas de Sistema verifican que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema total. Está constituida por varios tipos de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema.

Pruebas de funcionalidad

El objetivo principal de este tipo de prueba es asegurar el buen funcionamiento de los requisitos funcionales, incluyendo la entrada de datos, la navegación, procesamiento y obtención de resultados. Las pruebas de funcionalidad verifican el procesamiento, recuperación e implementación adecuada de las reglas del negocio con el propósito de determinar la extensión en la que la aplicación satisface las funcionalidades que el sistema debe cumplir.

Pruebas de caja negra

“Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de

caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca.” (17)

Objetivos del flujo de trabajo de prueba

- ✓ Encontrar y documentar los defectos que puedan afectar la calidad del software.
- ✓ Validar que el software trabaje como fue diseñado.
- ✓ Validar y probar los requisitos que debe cumplir el software en ese momento.
- ✓ Validar que los requisitos fueron implementados correctamente.

3.2 Plan de pruebas

La realización de un buen Plan de Pruebas es la piedra angular, y en consecuencia, el principal factor crítico de éxito para la puesta en práctica de un proceso de pruebas que permita entregar un software de mejor nivel. No obstante a que cada esfuerzo o proceso de pruebas puede ser diferente y específico, la mayor parte de los proyectos informáticos, sean de nuevos desarrollos o de mantenimiento de aplicaciones, tienen un marco común para la realización de las pruebas.

En el diseño de Casos de Pruebas se utilizaron las siguientes notaciones:

CP: Casos de Pruebas.

SCU: Sección de Casos de Uso.

3.2.1 Introducción

El desarrollo de aplicaciones informáticas implica una serie de actividades de desarrollo de software en las que es bastante común cometer errores. Esta es la principal razón por la que se necesita probar el software periódicamente con el objetivo de poder encontrar y solucionar estos errores aun cuando hay tiempo, y cuando su solución es viable al no implicar grandes atrasos en la terminación del software.

Cada proceso de pruebas a realizar debe ser planificada con antelación, para lograr de esta manera que los resultados alcanzados sean los esperados y que el proceso de pruebas tenga la calidad necesaria y cumpla con los objetivos por los que se realiza.

3.2.2 Propósito

El propósito de este Plan de Pruebas es definir la estrategia de pruebas de la Herramienta inteligente para la captura de RNF, además se pretende determinar con antelación los recursos necesarios para el mismo, y exponer los RF y CU que serán probados.

3.2.3 Alcance

El alcance de este Plan de Pruebas comprende la planificación para la ejecución de las Pruebas de Sistema a la herramienta inteligente para la captura de RNF. Estas pruebas son usualmente conducidas para comprobar que todos los módulos trabajan juntos sin error.

3.2.4 Especificaciones de software y hardware

- ✓ Software
- ✓ Sistema Operativo Linux o Windows.
- ✓ Gestor de BD PostgreSQL.
- ✓ Hardware
- ✓ Computadora Cliente

3.2.5 Descripción de los requisitos.

Caso de uso: Administrar RNF por proyectos

Este Caso de Uso le permite al Analista de Requisitos administrar los RNF del proyecto, realizando las actividades siguientes: registrar datos, seleccionar RNF, listar RNF, consultar significado de la

pregunta , sugerencia de respuesta a la pregunta, guardar reporte de los RNF, guardar el caso en la BD.

RF a probar:

- ✓ Seleccionar RNF
- ✓ Guardar Casos
- ✓ Registrar datos

Caso de uso: Mostrar RNF por proyectos

RF a probar:

- ✓ Recuperar Caso modificándolo

SCU: Insertar nueva respuesta

RF a probar:

- ✓ Insertar nueva respuesta

3.2.6 Estrategia de pruebas.

Luego de realizar un minucioso análisis sobre el estado de terminación de la aplicación a probar y los objetivos de la realización del Plan de Pruebas, se determinó que en este caso la estrategia de pruebas quedaría definida de la siguiente forma:

Definición del nivel de pruebas**: Pruebas de Sistema.**

Definición del tipo de pruebas**: Pruebas de Funcionalidad.**

Definición del método de prueba**: Prueba de Caja Negra.**

3.3 Diseño de casos de pruebas

Es un conjunto de condiciones o variables bajo las cuales se determina si los requisitos de una aplicación son parcial o completamente satisfactorios.

Caso de uso: Administrar RNF por proyecto

Descripción general

Este Caso de Uso le permite al Analista de Requisitos administrar los Requisitos No Funcionales del proyecto.

CP1 Seleccionar RNF

Descripción

Este caso de prueba le permite al analista seleccionar los RNF organizados por categorías.

Flujo central

El sistema muestra las preguntas a ser respondidas, el menú y 4 botones activos, los cuales son: Acepto, No acepto y Sugerencia, Significado.

El analista responde las preguntas según las peticiones del cliente, y de esta forma quedan seleccionados los RNF organizados por categorías.

Condiciones de ejecución

- ✓ Debe existir en la BD las preguntas y respuestas de las características generales de proyectos productivos, así como requisitos no funcionales asociados, de forma tal que permita al Analista de Requisitos la realización de la encuesta.
- ✓ El analista debe registrarse antes de comenzar la selección de RNF.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones

Seleccionar los RNF		El sistema debe seleccionar los RNF mostrar los mismos en una interfaz organizados por categorías.	El sistema muestra en una nueva interfaz el listado de RNF organizados por categorías.	

Tabla 7 CP1 Seleccionar RNF

CP2 Guardar Casos.

Descripción

Este caso de prueba permite al analista guardar un caso en la BD.

Flujo central

- ✓ Una vez mostrada la lista de RNF en una nueva interfaz, la misma contiene un menú con 2 opciones, “Guardar Casos” y “Guardar como *Portable Document Format (PDF)* /formato de documento portátil”.
- ✓ El analista selecciona la opción “Guardar Casos”.
- ✓ El analista completa todos los campos selecciona el botón “Guardar”.

Condición de ejecución

- ✓ El analista debe de haber terminado de seleccionar todos los RNF

Iteraciones

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observacion**

				es
Guardar Caso		El sistema debe guardar el Caso en la BD.	El sistema guarda el Caso en la BD	

Tabla 8 CP2 Guardar Caso

CP3 Registrar datos

Descripción

Este caso de prueba permite registrar los datos de la encuesta realizada.

Flujo central

- ✓ El sistema muestra una interfaz con los datos necesarios para comenzar la encuesta, los cuales son los siguientes: Nombre del analista presente, nombre del proyecto al que se le va a realizar la captura de RNF, y la fecha.
- ✓ El analista registra los datos correctamente.
- ✓ El sistema muestra un mensaje de que los datos fueron registrados con éxitos.

Condición de ejecución

Iteraciones

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**

Registrar datos correctamente		El sistema debe mostrar un mensaje "Los datos han sido registrados con éxito"	El sistema muestra un mensaje "Los datos han sido registrados con éxito"	
Registrar datos sin completar algunos campos.		El sistema debe mostrar un mensaje "Complete los campos"	El sistema muestra un mensaje "Complete los campos"	

Tabla 9 CP3 Registrar datos

CP4 Insertar nueva respuesta

Descripción

Este caso de prueba permite al analista insertar una nueva respuesta en la BD.

Flujo central

- ✓ Durante el desarrollo de la encuesta el sistema muestra un botón “insertar” donde puedes insertar una nueva respuesta, en caso de no estar de acuerdo con la que él muestra.
- ✓ El analista selecciona el botón “Insertar”.
- ✓ El sistema muestra una interfaz con un campo para introducir la nueva respuesta y dos botones “Aceptar” y “Cerrar”
- ✓ El analista selecciona el botón “Aceptar”.
- ✓ El sistema guarda la respuesta en la BD.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Insertar nueva respuesta		El sistema debe insertar una nueva respuesta en la BD.	El sistema inserta una nueva respuesta en la BD	
	Insertar nueva respuesta sin completar el campo necesario.	El sistema debe mostrar un mensaje notificando que debe de completar el campo para insertar la respuesta.	El sistema muestra un mensaje "Debe llenar los campos"	

Tabla 10 CP4 Insertar nueva respuesta

Condición de ejecución.

- El cliente debe de estar en desacuerdo con una pregunta que retorne una respuesta.

Caso de Uso Mostar RNF por proyectos

Descripción General

Este Caso de Uso permite mostrar los RNF de un proyecto dado y permite reutilizar los mismos.

CP1 Recuperar Caso modificándolo

Descripción

Este caso de prueba permite recuperar un caso ya sea modificando el que va a recuperado o recuperándolo íntegramente.

Flujo central

- ✓ El sistema le muestra una nueva interfaz con el Caso a recuperar y un botón “Re-Usar”
- ✓ El analista selecciona el botón “Re-Usar”.
- ✓ El sistema muestra dos opciones “Modificar”: reutilizar el caso con la posibilidad de realizar cambios en el mismo, “Guardar sin Modificar”: reutilizar el caso íntegramente.
- ✓ El analista selecciona la opción “Modificar”
- ✓ El sistema muestra una interfaz con el Caso para realizar los cambios deseados.
- ✓ El analista realiza los cambios deseados y presiona la opción “Mostrar Modificado”
- ✓ El sistema muestra una interfaz para registrar los datos: “Nombre del proyecto”, “Nombre del analista”, y “Fecha”,
- ✓ El analista completa los campos y selecciona “Siguiente”.
- ✓ El sistema muestra una interfaz con el Caso y sus datos y las opciones de “Guardar Listado” y “Guardar como PDF”
- ✓ El analista selecciona la opción “Guardar Listado”
- ✓ El sistema guarda el nuevo caso en la BD.
- ✓ El analista selecciona en el menú la opción “Guardar como PDF” para obtener el reporte del Caso recuperado.
- ✓ El sistema muestra una interfaz para guardar el reporte del Caso en el formato “.PDF”, donde podrá elegir el nombre y la ruta donde será guardado el Caso.
- ✓ El analista obtiene el Caso recuperado.

Condición de ejecución

- ✓ Debe existir al menos un Caso anterior en la BD.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Recuperar Casos, modificándolo registrándose correctamente		El sistema debe mostrar el Caso recuperado en un documento y guardarlo en la BD	El sistema muestra el Caso recuperado en un documento y lo guarda en la BD	
	Recuperando el caso, dejando campos vacíos.	El sistema debe mostrar un mensaje indicando que debe llenar los campos vacíos.	El sistema muestra un mensaje" debe llenar los campos vacíos"	

Tabla 11 CP5 Recuperar Caso modificándolo

3.4 Manual de usuario

El presente Manual de Usuario, tiene como propósito dar a conocer de una manera detallada y sencilla, el proceso que se lleva a cabo a través del uso de la Herramienta para la captura de RNF, con el objetivo de que los analistas se familiaricen con la aplicación.

3.4.1 Requisitos básicos

Tener una computadora con la aplicación y la BD.

Tener instalado QT y la librería pqxx

3.4.2 Acceso a la aplicación

La aplicación estará a su disposición en el escritorio de la PC y no necesita instalación.

3.4.3 Seleccionar RNF

Para seleccionar los RNF debe de seguir los siguientes pasos:

- ✓ Seleccionar en el menú principal la opción “Iniciar Captura de RNF para un nuevo proyecto”.

  Aparecerá una nueva interfaz para registrar los datos.
- ✓ Registra todos los datos.
- ✓ Presionar “Iniciar Encuesta”

Se muestran las preguntas a ser respondidas por el cliente, y 4 botones, los cuales son: “Acepto”, “No acepto” y “Sugerencia” y “Significado” se debe ir respondiendo las preguntas a través de los botones “Acepto” y “No acepto” hasta completar la selección de todos los RNF.

3.4.4 Mostrar listado de RNF

- ✓ Para mostrar el listado de de los RNF sólo es necesario seleccionar todos los RNF y una vez realizado esto el sistema muestra automáticamente el listado de los mismos en una interfaz.

3.4.5 Mostrar significado

Para observar el significado de una pregunta se deben seguir los siguientes pasos:

- ✓ Seleccionar el botón “Significado”

El sistema mostrará una breve explicación de la pregunta.

3.4.6 Mostrar sugerencia

Para observar una sugerencia debe seguir los siguientes pasos:

Seleccionar el botón “Sugerencia”.

Se muestra una interfaz con la sugerencia que da el sistema sobre si acepta o no la pregunta en que se encuentra durante la realización de la encuesta.

3.4.7 Insertar respuesta

Para insertar una nueva respuesta debe seguir los siguientes pasos:

Durante el desarrollo de la encuesta usted tiene la posibilidad de insertar una nueva respuesta, en caso de no estar de acuerdo con la que el sistema le brinde.

- ✓ Aparece en la interfaz a la izquierda un botón “Insertar”.
- ✓ Selecciona el botón “Insertar”.

  Aparece una nueva interfaz con un campo para introducir la nueva respuesta y dos botones “Aceptar” y “Cerrar”
- ✓ Selecciona el botón “Aceptar” y queda guardada su respuesta en la BD.

3.4.8 Guardar Casos

Para guardar un caso debe seguir los siguientes pasos:

- ✓ Cuando el sistema muestra en una interfaz la lista de requisitos, la misma contiene, un menú con 2 opciones, “Guardar Caso” y “Guardar como PDF”.
- ✓ Selecciona “Guardar Caso”.

  De esta manera quedará guardado este caso en la BD.

3.4.9 Guardar reporte de los RNF

Para guardar el reporte de los RNF seleccionados debe seguir los siguientes pasos:

Cuando el sistema muestra en una interfaz la lista de requisitos, la misma contiene, el un menú con 2 opciones, “Guardar Caso” y “Guardar como PDF”.

- ✓ Selecciona la opción “Guardar como PDF”.

Aparecerá una nueva ventana que le posibilitará guardar el reporte de los RNF en el formato “.PDF”, y elegir el nombre y la ruta donde guardará dicho reporte.

3.4.10 Recuperar Casos modificándolo

Para la recuperación de un caso anterior realizándole modificaciones debe seguir los siguientes pasos:

- ✓ Selecciona el botón “Re-Usar”.

Se muestra dos opciones “Modificar”: recuperar el caso con la posibilidad de realizar cambios en el mismo, “Guardar sin Modificar”: reutilizar el caso íntegramente.

- ✓ Selecciona la opción “Modificar”

Se muestra una interfaz con el Caso para realizar los cambios deseados.

- ✓ Realiza los cambios deseados y presiona la opción “Mostrar Modificado”.

Se muestra una interfaz para registrar los datos: “Nombre del proyecto”, “Nombre del analista”, y “Fecha”,

- ✓ Completa los campos y selecciona “Siguiente”.

Se muestra una interfaz con el Caso y sus datos y las opciones de “Guardar Listado” y “Guardar como PDF”.

- ✓ Selecciona la opción “Guardar Listado”

Se guarda el nuevo caso en la BD.

- ✓ Selecciona en el menú la opción “Guardar como PDF” para obtener el reporte del Caso recuperado.

- ✓ Se muestra una interfaz para guardar el reporte del Caso en el formato ".PDF", donde podrá elegir el nombre y la ruta donde será guardado el Caso.

Obtienes el Caso recuperado.

3.4.10.1 Recuperar Caso sin modificar

Para recuperar un caso sin modificar debes seguir los siguientes pasos:

- ✓ Selecciona la opción "Guardar sin Modificar".

  Se muestra una interfaz para registrar los datos: "Nombre del proyecto", "Nombre del analista", y "Fecha",

- ✓ Completa los campos y selecciona "Siguiente".

  Se muestra una interfaz con el Caso y sus datos y las opciones de "Guardar Listado" y "Exportar a PDF"

- ✓ Selecciona la opción "Guardar Listado"
- ✓ Se guarda el nuevo caso en la BD.
- ✓ Selecciona en el menú la opción "Exportar a PDF" para obtener el reporte del Caso recuperado.

  Se muestra una interfaz para guardar el reporte del Caso en el formato ".PDF", donde podrá elegir el nombre y la ruta donde será guardado el Caso.

  Obtienes el Caso recuperado.

3.5 Seguimiento y control de los RNF

El proceso de especificación de requisitos se puede dividir en tres grandes actividades: Captura de Requisitos, Definición de Requisitos y Validación de requisitos.

El presente trabajo se centra en la captura de RNF. La Captura de requisitos es la actividad que permite al equipo de desarrollo de un sistema de software extraer de cualquier fuente de información disponible, las necesidades que debe cubrir.

El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es

desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa. Aún así existen problemas a la hora de capturar los RNF en el PHA, por lo que se crea la herramienta inteligente para la captura de los RNF.

Una vez creada dicha herramienta deberá ser distribuida hacia todos los Analistas de Requisitos del PHA para que sea validada y se familiaricen con la misma.

Se consultará con los analistas para ver si la herramienta les parece amigable, fácil de usar, navegable útil y eficiente.

Cuando sea validada se debe elaborar una estrategia de motivación sobre el trabajo realizado donde se proponen las siguientes actividades:

- ✓ Explicar la importancia de la herramienta a los Analistas de Requisitos del PHA.
- ✓ Realizar encuentros y debates sobre la herramienta realizada en el proyecto para que sea de conocimiento de los demás equipos de trabajo de líneas diferentes en el mismo proyecto.

Conclusiones

Se puede concluir al término de la presente investigación sobre el desarrollo de una herramienta inteligente para la captura de los RNF en los proyectos del PHA, que el objetivo general fue cumplido. Alcanzándose los siguientes resultados:

- ✓ Se identificaron conceptos, definiciones y teorías en el campo de los requisitos, los que facilitaron la comprensión del tema a desarrollar.
- ✓ Organización de los RNF según el tipo, permitiendo obtener sus dependencias y relaciones.
- ✓ Se desarrolló la herramienta inteligente para la captura de requisitos no funcionales en proyectos productivos, teniendo como base un encuestador de preguntas asociadas a las características del proyecto bajo análisis.

Recomendaciones

Una vez desarrollada la herramienta inteligente para la captura de RNF, inicialmente enfocada en el PHA, se presentan los siguientes aspectos que son necesarios para darle continuación a este trabajo:

- ✓ Seguir investigando acerca del objeto de estudio y del campo de acción planteados para tener un conocimiento más profundo de los mismos.
- ✓ Hacer extensible la herramienta hacia otros Polos de la Universidad.
- ✓ Dar mantenimiento periódicamente tanto al sistema como a la BD, con el objetivo de extender su vida útil.

Referencias Bibliográficas

1. **(ONE), Oficina Nacional de Estadísticas de Cuba.** *Panorama económico y social.* Cuba : s.n., 2008.

2. **R, Ralph Young.** *The Requirements Engieneering .* London : s.n., 2004.

3. IEEE Standards Association. *standards.ieee.* [En línea] 24 de Mayo de 2006. [Citado el: 14 de Abril de 2009.] http//standards.ieee.org.

4. **Sommerville, Ian.** *Ingeniería del Software.* México DF : Editorial Pearson.

5. **Straub, Dr.Pablo.** *Universidad Diego Portales.* [En línea] 2007. http://www.udp.cl.

6. **Chaves, Michel Arias.** *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.* Costa Rica : Revista InterSedes, 2007. Vol. VI, 10. ISSN 1409-4746.

7. **Grupo de Investigación.** KYBELE. [En línea] 2007. [Citado el: 14 de Enero de 2009.] http://kybele.escet.urjc.es.

8. **IEEE std 1233.** *Guía para el desarrollo de Especificaciones de Requerimientos de Sistemas.* 1998.

9. **RUMBAUGH y I.J.G.B.J.** *El Proceso Unificado de Desarrollo de Software.* 2000.

10. **Durán, Amador Toro y Bernárdes, Beatriz Jiménez.** *Metodología para la elicitación Requisitos de sistemas software.* Sevilla : s.n., 2000.

11. **Sommerville, Ian y Sawyer, Pete.** *Requirements Engineering.* 1997.

12. **Dávila, Nicolás Davyt.** *Una guía para extraer, analizar, especificar y validar los requerimientos de un proyecto.* Uruguay : s.n., 2001.

13. **A, Ortas.** *Aproximación de la Ingeniería de Requerimientos.* Uruguay : s.n., 2001.

14. **Raghavan, S, Zelesnik, G y Ford, G.** Software Engineering Institute, Carnegie Mellon University. [En línea] 1994. [Citado el: 25 de febrero de 2009.] //www.sei.cmu.edu. CMU/SEI-93-EM-010.

15. **Medina, Carod Martínez.** ficcte. *Facultad de Informática, Ciencias de la Comunicación y Técnicas Especiales.* [En línea] 2000. [Citado el: 18 de marzo de 2009.] http://ficcte.unimoron.edu.ar.

16. **Castro, Marcel.** SISTEMAS EXPERTOS. *strix.* [En línea] 2002. [Citado el: 20 de Abril de 2009.] http://strix.ciens.ucv.ve/~iartific/Material/PP_Sistemas_Expertos.pdf.

17. **S.Pressman, Roger.** *Ingeniería de Software un enfoque práctico.* 5ta Edición.

BIBLIOGRAFÍA CONSULTADA

1. **Castro, Marcel.** SISTEMAS EXPERTOS. *strix.* [En línea] 2002. [Citado el: 20 de Abril de 2009.] http://strix.ciens.ucv.ve/~iartific/Material/PP_Sistemas_Expertos.pdf.

2. **Chaves, Michel Arias.** *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.* Costa Rica : Revista InterSedes, 2007. Vol. VI, 10. ISSN 1409-4746.

3. **Dávila, Nicolás Davyt.** *Una guía para extraer, analizar, especificar y validar los requerimientos de un proyecto.* Uruguay : s.n., 2001.

4. **Durán, Amador Toro y Bernárdes, Beatriz Jiménez.** *Metodología para la elicitación Requisitos de sistemas software.* Sevilla : s.n., 2000.

5. **Larman, Craig.** *UML y Patrones: Introducción al análisis y diseño orientado a objetos.* México : Universitarios, 1999. ISBN: 970-17-0261-1.

6. **(ONE), Oficina Nacional de Estadísticas de Cuba.** *Panorama económico y social.* Cuba : s.n., 2008.

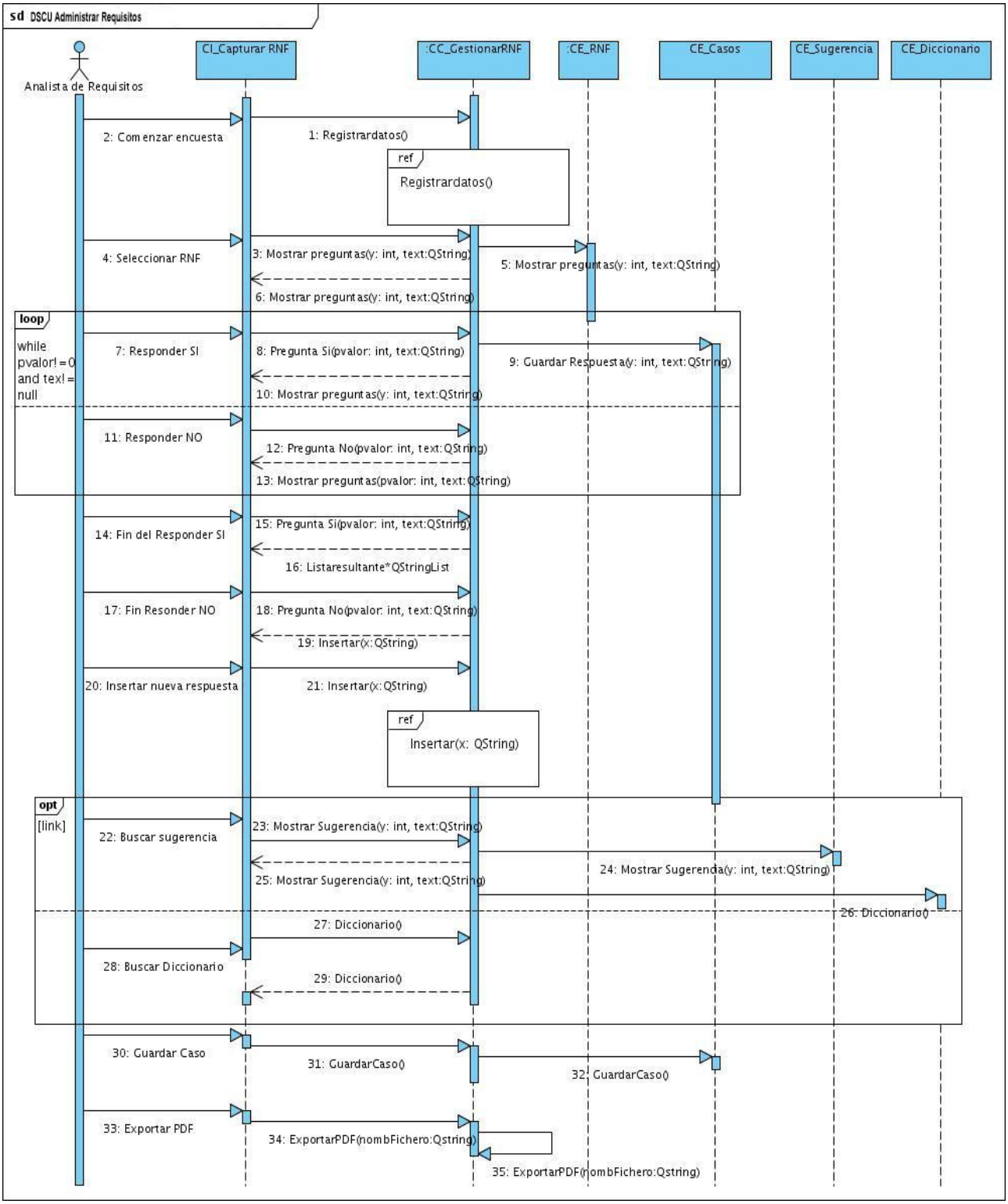
7. **R, Ralph Young.** *The Requirements Engieneering .* London : s.n., 2004.

8. **Sommerville, Ian.** *Ingeniería del Software.* México DF : Editorial Pearson.

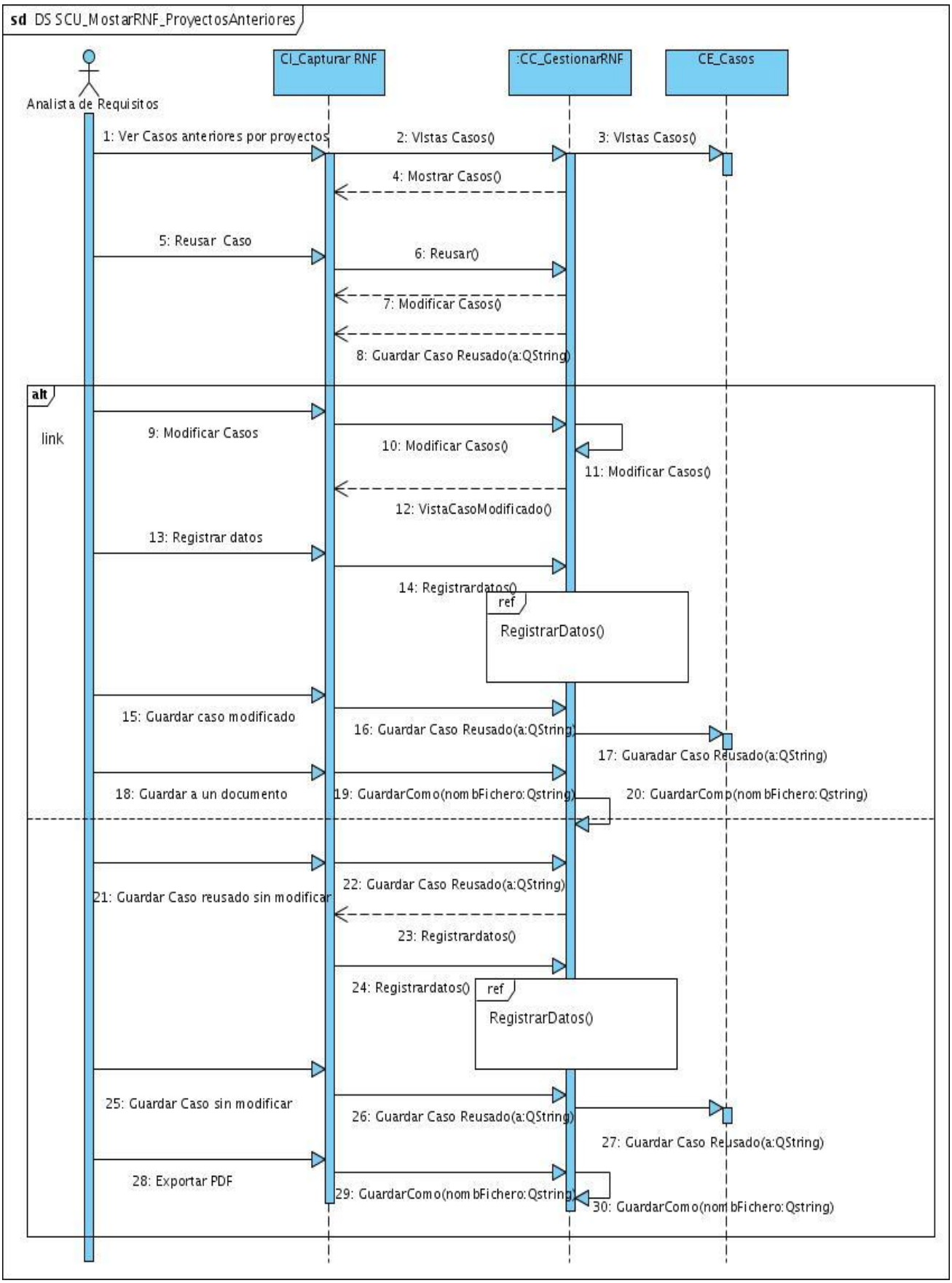
9. **Sommerville, Ian y Sawyer, Pete.** *Requirements Engineering.* 1997.

10. **S.Pressman, Roger.** *Ingeniería de Software un enfoque práctico.* 5ta Edición.

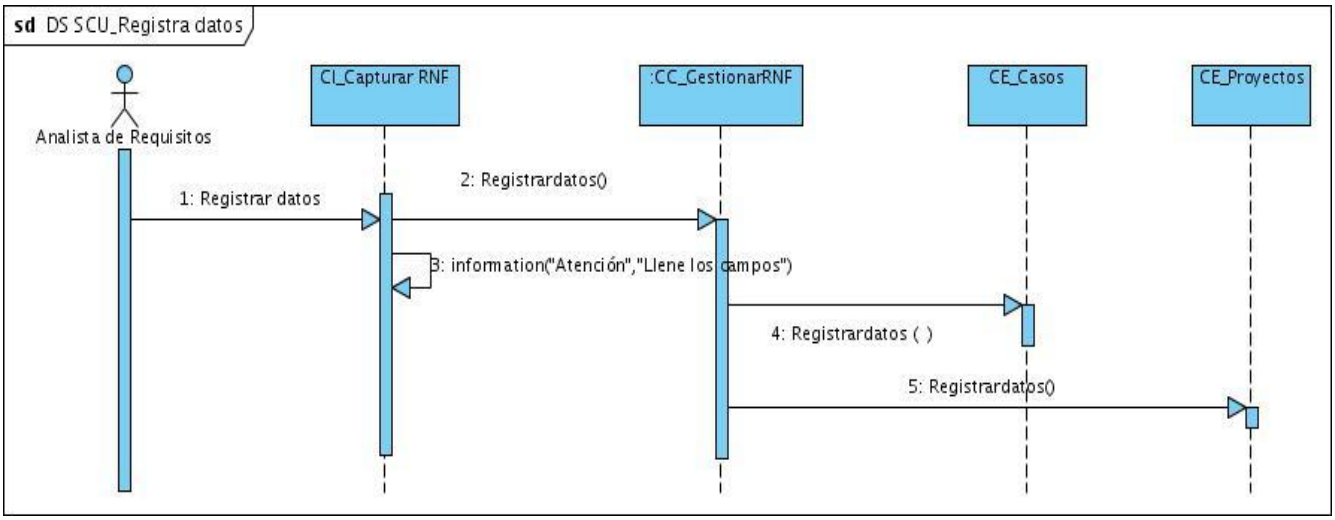
Anexos



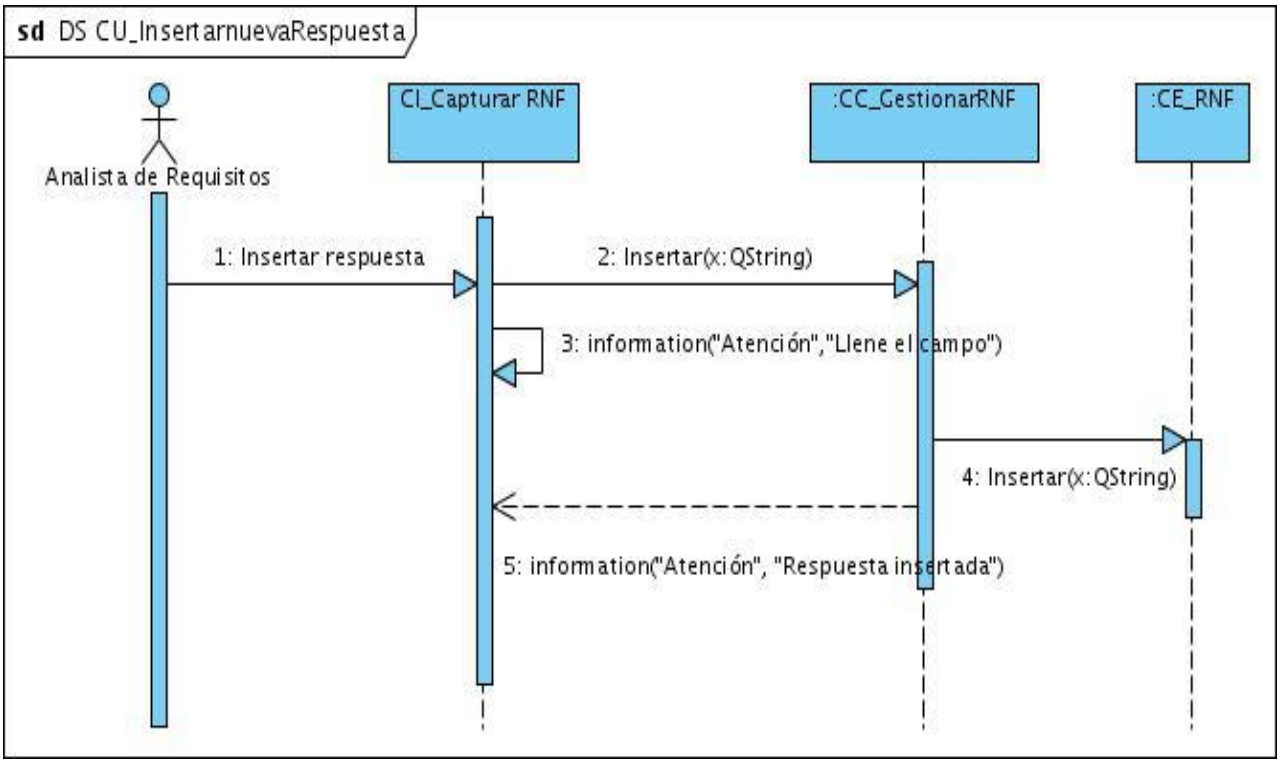
Anexo 1- DSCU Administrar RNF por proyecto


sd DS SCU_MostarRNF_ProyectosAnteriores
Analista de Requisitos
CI_Capturar RNF
:CC_GestionarRNF
CE_Casos
1: Ver Casos anteriores por proyectos
2: VIstas Casos()
3: VIstas Casos()
4: Mostrar Casos()
5: Reusar Caso
6: Reusar()
7: Modificar Casos()
8: Guardar Caso Reusado(a:QString)
alt
link
9: Modificar Casos
10: Modificar Casos()
11: Modificar Casos()
12: VistaCasoModificado()
13: Registrar datos
14: Registrardatos()
ref
RegistrarDatos()
15: Guardar caso modificado
16: Guardar Caso Reusado(a:QString)
17: Guaradar Caso Reusado(a:QString)
18: Guardar a un documento
19: GuardarComo(nombFichero:Qstring)
20: GuardarComp(nombFichero:Qstring)
21: Guardar Caso reusado sin modificar
22: Guardar Caso Reusado(a:QString)
23: Registrardatos()
24: Registrardatos()
ref
RegistrarDatos()
25: Guardar Caso sin modificar
26: Guardar Caso Reusado(a:QString)
27: Guardar Caso Reusado(a:QString)
28: Exportar PDF
29: GuardarComo(nombFichero:Qstring)
30: GuardarComo(nombFichero:Qstring)

Anexo 2- DSCU Mostrar RNF por proyectos



Anexo 3 – DS SCU Registrar Datos



Anexo 4 – DS SCU Insertar nueva respuesta

Anexo-5 Sección del árbol de software

Glosario de términos

A:

Auditorías: Esta técnica consiste en un chequeo de los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso.

Adaptabilidad: Es la facilidad con la cual puede adaptarse el sistema a los cambios, donde los requisitos son un factor de cambios de la aplicación.

B:

Base de Datos (BD): Es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso.

Base de Conocimiento: Contiene el conocimiento especializado extraído del experto en el tema.

C:

Caso de Uso: Secuencias de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de las secuencias.

Clase: Es la declaración o abstracción de un objeto cuando se programa según el paradigma de orientación a objetos.

Código abierto: Relativo al software para el cual el código fuente está disponible en forma gratuita.

D:

Desarrollo conjunto de aplicaciones (JAD): Esta técnica está de captura de requisitos está basada en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación.

E:

Eficiencia: Es la habilidad del software para poner la cantidad mínima de demanda sobre los recursos de hardware como sea posible.

Entidad: Es la representación de un objeto o concepto del mundo real que se describe en una base de datos.

Glosario de Términos

I:

Ingeniería de requisitos (IR): Es el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema.

Ingeniería del conocimiento: Proceso mediante el cual se genera dicha BD se le llama ingeniería del conocimiento.

Inteligencia artificial (IA): Lo que pretende la IA es crear una máquina secuencial programada que repita indefinidamente un conjunto de instrucciones generadas por un ser humano.

M:

Metodología: Se refiere a los métodos de investigación que se siguen para alcanzar una gama de objetivos en una ciencia.

Modelo Vista Controlador (MVC): Es una arquitectura de software que separa el modelo de datos de una aplicación, la interfaz de usuario, y la lógica de control en tres distintos componentes de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos.

Multiplataforma: Poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

P:

Paradigma: Es un modelo o patrón en cualquier disciplina científica u otro contexto epistemológico.

R:

Requisitos: Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.

Requisitos no funcionales (RNF): Tienen que ver con características que de una u otra forma puedan limitar el sistema.

S:

Sistema: Es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. Es una aplicación informática.

Software: Los requisitos de software son las características que debe tener el software instalado en una computadora para poder soportar y/o ejecutar una aplicación o un dispositivo específicos.