
# UNIVERSIDAD DE CHILE
# FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
# DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION


## GENERACIÓN AUTOMATICA DE MÉTRICAS EN PROYECTOS DE SOFTWARE, A PARTIR DE LA ESPECIFICACIÓN DE REQUISITOS

**TESIS PARA OPTAR AL GRADO DE MAGISTER EN CIENCIAS MENCION COMPUTACION**


**ANDRÉS RODRIGO VERGARA ASTUDILLO**


PROFESOR GUIA:
SERGIO FABIÁN OCHOA

MIEMBROS DE LA COMISION:
MARÍA CECILIA RIVARA
NELSON BALOIAN
MARCELLO VISCONTI

SANTIAGO DE CHILE
ABRIL 2008

RESUMEN DE LA MEMORIA
PARA OPTAR AL GRADO DE
MAGISTER CS. MENCION COMPUTACIÓN
POR: ANDRÉS R. VERGARA ASTUDILLO
PROF. GUIA: SR. SERGIO OCHOA

# GENERACIÓN AUTOMÁTICA DE MÉTRICAS EN PROYECTOS DE SOFTWARE A PARTIR DE LA ESPECIFICACIÓN DE REQUISITOS

La especificación y el análisis de requisitos son actividades fundamentales, debido a que son los cimientos sobre los cuales se construirá una aplicación durante las siguientes etapas de desarrollo de un software. A diferencia de lo que ocurre en la construcción de un edificio, los cimientos o los requisitos en este caso, cambian y es necesario poder hacer seguimiento de estos. Es importante tener la posibilidad de manejar la trazabilidad de los cambios y poder cuantificar el impacto que pueden tener estas modificaciones sobre el proceso de desarrollo de software, tanto en la duración total del proyecto, como en su costo. Este trabajo propone automatizar ciertos mecanismos de trazabilidad y métricas que permitan reducir los costos y tiempos de desarrollo, además de disminuir los riesgos asociados al incumplimiento de requisitos.

Antes de partir con el desarrollo de los mecanismos de trazabilidad, se hizo un rediseño de ReqAdmin, que es la herramienta de código abierto resultante de mi memoria de ingeniería a la cual se le añadieron las alarmas y métricas definidas. Finalmente se incorporaron indicadores que sirven para diagnosticar el nivel de sanidad de los requisitos y de implementación del sistema.

El resultado final fue una herramienta que es capaz de administrar los requisitos por medio de un proceso genérico. Es fácil de usar y permite su operación en forma distribuida, facilitando así la comunicación entre los miembros del equipo de desarrollo. Permite la clasificación y simplifica la validación, control y seguimiento de los requisitos de un sistema. Se espera que esta herramienta ayude a mejorar tanto los productos obtenidos en la fase de análisis, como la visibilidad de esta fase sobre el proceso completo de desarrollo de software. Con esto se aliviará la gestión del proyecto, y por ende, mejorará la capacidad de predecir el cronograma, así como su resultado. La administración de requisitos propuesta probablemente permitirá una reducción del esfuerzo de desarrollo de los artefactos de análisis.

# Agradecimientos

En primer lugar quisiera agradecer a todos los estudiantes que probaron la herramienta y que aportaron con nuevas ideas y observaciones al desarrollo de ésta. Mis disculpas por cualquier mal rato que hayan tenido que pasar por su labor de “conejillos de indias”.

También tengo que agradecer a mi familia, en particular a mi madre, Patricia, por su constante estimulo y cariño.

A Sergio Ochoa por todo su apoyo y guía en este trabajo.

# INDICE GENERAL

# 1. Introducción

En un proyecto de software, la especificación, análisis y administración de requisitos son actividades fundamentales, debido a que constituyen la base sobre la cual se apoya el desarrollo durante el proyecto. Estas actividades dan origen a lo que se conoce hoy como ingeniería de requisitos [Hof01, Hoo00, Tha99]. En base a ellas, el desarrollador de software puede determinar las necesidades de un cliente, definir las características y el alcance de la solución de software que se proponga, y construir los modelos iniciales del sistema (diseño de la arquitectura, datos y comportamiento).

La especificación de requisitos es generalmente extensa, detallada y difícil de administrar. A esto hay que agregar que, en proyectos de software, los requisitos deben expresarse por escrito, ya que de esa manera es más fácil comunicarlos, considerarlos y evaluar su cumplimiento sobre el producto final. Por ende, la especificación debe ser completa y sin ambigüedad.

La especificación contiene requisitos con distintos niveles de granularidad, los cuales a su vez pueden corresponder a distintas categorías (requisitos de usuario/software, o requisitos funcionales/no funcionales/de calidad). Por otra parte, los requisitos mantienen múltiples relaciones entre sí, las cuales deben ser especificadas y administradas para evitar, tanto la pérdida como la aparición sin fundamento de requisitos a lo largo del proyecto. Relaciones similares también son mantenidas entre los requisitos y artefactos de software o casos de pruebas del sistema.

Especificar los requisitos con los que debe cumplir un software es sólo una parte del problema. Poco sirve una especificación de requisitos que es subutilizada durante un proyecto de desarrollo. Por esa razón, se requiere un proceso de administración de requisitos, a fin de que tanto el producto final como los productos intermedios adhieran a estas especificaciones. La especificación de requisitos y su administración siempre van de la mano. No tiene sentido una sin la otra, debido a que los cambios que se realicen en los requisitos también afectan a otros niveles del desarrollo. Una buena trazabilidad de los requisitos permite una gestión efectiva y eficiente de éstos, sin comprometer el desarrollo del sistema.

Para tener la posibilidad de especificar y administrar los requisitos en forma adecuada, especialmente en sistemas complejos, dichas especificaciones deben ser completas, simples, concisas y administrables. Esto involucra tanto a los requisitos, como a las relaciones entre ellos. ¿Cómo se podría lograr esto? Mediante la estandarización, formalización, clasificación, generación de indicadores de sanidad de requisitos y la organización de todo lo que el sistema debe hacer. En otras palabras al documentar formalmente y administrar, tanto los requisitos como las relaciones entre ellos. Básicamente esto implica trabajar en su gestión para lograr una buena trazabilidad del proyecto durante todo su desarrollo.

## *1.1 Justificación del trabajo*

La especificación y análisis de los requisitos puede parecer una tarea relativamente sencilla, pero como ocurre habitualmente, las apariencias engañan. En un proyecto de software, el contenido a comunicar (contexto y requisitos) es muy denso y a veces incluye conceptos propios del dominio de aplicación, los cuales son desconocidos para muchos de los diseñadores que utilizarán la especificación. Abundan las ocasiones para las malas interpretaciones debido a la falta de información (requisitos incompletos), especificaciones ambiguas, requisitos con distinto nivel de granularidad o relaciones no especificadas entre requisitos. En otras ocasiones, aunque los requisitos están bien especificados y validados, la falta de visibilidad de ellos sobre todo el proyecto, hace que el diseño de artefactos de software no considere completamente las especificaciones.

Es normal que durante un proyecto de software los requisitos cambien. Por lo tanto se requiere saber cuál es el impacto del cambio de uno o más requisitos, sobre la duración total y el costo del proyecto. Para poder determinar esto, se deben mantener los vínculos entre los requisitos, y entre éstos y los distintos productos intermedios del proyecto (módulos de software, casos de prueba, etc.). Básicamente se necesita un buen manejo de la trazabilidad del proyecto. Hasta el momento no se ha encontrado una herramienta capaz de realizar esto en forma automática, o bien que requiera poco esfuerzo por parte del usuario en cuanto al aprendizaje que tenga que realizar éste sobre la herramienta respectiva.

Por otra parte, se requiere también automatizar el proceso de generación y análisis de métricas e instrumentos de trazado, ya que los instrumentos actuales requieren, en su mayoría, procesamiento manual. Estos instrumentos juegan un rol importante a la hora de tomar decisiones respecto a cambios en el proyecto, evaluar su avance o determinar el nivel de sanidad de sus requisitos. Además, generalmente estas actividades deben ser realizadas en un período corto de tiempo, por lo tanto, se requiere que la generación y el análisis de métricas e instrumentos de trazado se realice lo más automática y rápidamente posible. Acá nuevamente se pone de manifiesto la necesidad de mantener y administrar los vínculos entre los requisitos, y entre éstos y los distintos productos intermedios del proyecto. Debido a lo antes expuesto, es altamente probable que el producto final tenga problemas de funcionalidad, o bien que las implementaciones terminen siendo parchadas a fin de cumplir con los requisitos.

Debido a la complejidad que comúnmente tiene la especificación y administración de los requisitos en un proyecto de software, se hace necesario el uso de una herramienta integral que apoye esta tarea, y que reduzca el esfuerzo de dar visibilidad y de mantener la trazabilidad de los requisitos. Esta herramienta debe poseer métricas e instrumentos de trazado que permitan detectar tempranamente irregularidades en la especificación, por ejemplo a través del análisis de una matriz de trazado, y del despliegue de métricas (indicadores) que expresen el nivel de sanidad de los requisitos.

Este trabajo de tesis define e implementa métricas que permiten detectar anomalías en la especificación de requisitos y las incluye en una herramienta integral que permite la especificación y administración de requisitos de usuario y de software, en base a ciertos mecanismos automatizados.

## *1.2 Hipótesis de trabajo*

Las hipótesis en las cuales se basó este trabajo son las siguientes:

H1: Un proceso automatizable de administración de requisitos, capaz de establecer la trazabilidad de estos elementos, ayuda a la detección temprana de inconsistencias en requisitos y en la relación de éstos con otros productos intermedios, como por ejemplo los módulos implementados.

H2: La generación automática de métricas que trabajen sobre los requisitos, permite a los desarrolladores de software obtener rápidamente un diagnóstico del estado del proyecto, en términos de avance, trabajo pendiente, pérdida/cambio de requisitos, requisitos prioritarios y posibles riesgos derivados de los requisitos o de su administración.

## *1.3 Objetivos de la tesis*

Uno de los propósitos de este trabajo fue definir métricas que fueran generadas automáticamente a partir de los requisitos de un proyecto de desarrollo de software. La definición de estas métricas y su implementación en una herramienta de administración de requisitos (en este caso ReqAdmin [Req05]), ayudaron a mejorar los productos obtenidos de la fase de análisis y la visibilidad de esta fase sobre el proceso completo. Los objetivos específicos que se derivan del objetivo general definido, son principalmente tres:

i. Desarrollar un proceso automatizable de administración de requisitos que siga un formato genérico y que permita ver, en cualquier momento, la trazabilidad de los requisitos de un sistema de software. Este proceso no está asociado a una metodología de desarrollo de software específica o a un paradigma de programación en particular.

ii. Definir y generar métricas que permitan mejorar la calidad de los productos desarrollados en la fase de análisis. Los indicadores muestran el nivel de sanidad de los requisitos, en forma tal que los riesgos puedan ser identificados en forma temprana. Esto permite también tomar decisiones a tiempo, para corregir situaciones no deseadas durante el proyecto.

iii. Desarrollar una herramienta de software que permita la administración integral de los requisitos. Ésta permite la clasificación, jerarquización, validación, además de facilitar el control y seguimiento de los requisitos de un sistema. De esa manera, se reduce el esfuerzo requerido para la gestión del proyecto, y por ende, mejorar la capacidad de predecir el cronograma del proyecto, así como su resultado. La administración de requisitos propuesta ha permitido una reducción del esfuerzo de desarrollo de los artefactos de análisis (costo y tiempo) en todos aquellos casos donde ha sido utilizada.

# 2. Antecedentes

Esta sección presenta y discute diversos conceptos que están involucrados a lo largo de este documento. A fin de brindarle al lector una mejor perspectiva para evaluar este trabajo, a continuación se introducen dichos conceptos.

## *2.1 Ingeniería de requisitos*

La administración y especificación de requisitos, también llamada ingeniería de requisitos cumple un papel primordial en el proceso de producción de software, ya que se enfoca en un área fundamental: “la definición de lo que se desea producir”. Su tarea principal consiste en la generación de especificaciones correctas que describan, en forma clara, consistente y completa, el comportamiento del sistema. Esto generalmente permite minimizar los problemas relacionados al desarrollo de sistemas.

RUP (Rational Unified Process) define a la ingeniería de requisitos como un “enfoque sistemático para recolectar, organizar y documentar los requisitos del sistema; también es el proceso que establece y mantiene acuerdos sobre los cambios de requisitos, entre los clientes y el equipo de desarrollo” [Ibm05]. Dentro de los beneficios que se obtienen de la Ingeniería de Requisitos, se pueden mencionar que:

- *Permite gestionar las necesidades del proyecto en forma estructurada*. Cada actividad de la ingeniería de requisitos consiste en una serie de pasos organizados y bien definidos.

- *Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados*. La ingeniería de requisitos proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempos y recursos necesarios.

- *Ayuda a disminuir los costos y retrasos del proyecto*. Esto se debe a que los objetivos están claros desde el inicio. En caso de cambios en los requisitos el cliente está obligado a renegociar costos y tiempos. De esa manera se evita caer en situaciones de "all-inclusive".

- *Mejora la calidad del software*. La calidad en el software tiene que ver con cumplir un conjunto de requisitos (funcionalidad, confiabilidad, rendimiento, etc.). A partir de la especificación de requisitos se puede converger en forma temprana a especificar las pruebas que se le realizarán al sistema, y a validar las características generales del producto en desarrollo.

- *Mejora la comunicación entre equipos*. La especificación de requisitos representa una forma de consenso entre clientes y desarrolladores. Si este acuerdo no ocurre, el proyecto no será exitoso.

- *Evita o disminuye los rechazos por parte de los usuarios finales*. La ingeniería de requisitos obliga al cliente/usuario a considerar sus requisitos cuidadosamente y a revisarlos dentro del marco del problema, por lo que los clientes / usuarios deberían estar involucrados durante todo el desarrollo del proyecto.

Muchos de los beneficios mencionados anteriormente, tales como la mejora en la predicción de un cronograma o de la calidad del software se logran con ayuda de una buena trazabilidad. En el siguiente capitulo se ahondará en este concepto.

## *2.2 Trazabilidad*

Un aspecto importante de la ingeniería de requisitos es el cambio y evolución de éstos, ya sea por petición externa del cliente, o por retroalimentación desde el proceso de diseño. Una buena trazabilidad permite una gestión eficaz de estos cambios, y un adecuado control entre los artefactos generados en la etapa de diseño a partir de los requisitos.

### 2.2.1 ¿Qué es trazabilidad?

La trazabilidad ha sido ampliamente identificada en la literatura como un factor de calidad del proceso y un promotor de la calidad del producto. Esta última es una característica que todo sistema debe poseer. Muchos estándares de desarrollo generados por organizaciones del gobierno de los Estados Unidos (por ejemplo, MIL-STD-2167-A [Roe91] y MIL-STD-498 [USD88]) requieren que se desarrollen documentos de trazabilidad como parte de la documentación regular de un proyecto. A continuación se mencionan varias definiciones de este concepto:

Edwars y Howell [Edw91] definen la trazabilidad como: “La técnica usada para proveer una relación entre los requisitos, el diseño y la implementación del sistema”. Palmer [Pal97] se refiere a ella de la siguiente forma: “La trazabilidad da la ayuda total en entender las relaciones que existen dentro y a través de los requisitos, diseño e implementación del software”. Esta relación permite a los diseñadores demostrar que el diseño resuelve los requisitos y ayuda al reconocimiento temprano de qué requisitos no son satisfechos por el diseño.

De acuerdo con el Glosario Estándar de la terminología de la Ingeniería de Software del IEEE [Ics07], trazabilidad está definida como “El grado en el cual una relación se puede establecer entre dos o más o productos del proceso de desarrollo, especialmente productos que tienen relaciones del tipo predecesor-sucesor o maestro-subordinado”.

Rational Unified Process (RUP) define el concepto de trazabilidad más genéricamente, como “la habilidad de trazar un elemento del proyecto a otro elemento relacionado del proyecto, especialmente aquellos relacionados con requisitos”.

Luego de haber leído todas estas definiciones podemos observar que “trazabilidad” es un concepto muy genérico, no muy difícil de entender. La trazabilidad puede ser vista como una característica del sistema, en el cual los requisitos son claramente vinculados a sus fuentes y a los artefactos creados durante el ciclo de vida del desarrollo del sistema. Mucha gente cree que trazabilidad implica construir una estructura jerárquica muy simple, con niveles bien definidos, y que representa una relación maestro-subordinado. Pero en la realidad ésta resulta ser más complicada, y por lo mismo, durante el desarrollo del proyecto no se actualiza toda la información necesaria, perdiendo así la oportunidad de mejorar la calidad y de responder a los cambios rápidamente. Pero ¿Cómo

evitar que esto suceda? Para contestar esta pregunta, primero necesitamos entender por qué necesitamos la trazabilidad y cuál es su ventaja.

### 2.2.2 ¿Por qué buscamos trazabilidad?

Según Sommerville [Som04] en lo que concierne a la ingeniería de requisitos, la trazabilidad se descompone en tres ámbitos distintos.

- *Trazabilidad de la fuente*: Aquella que vincula los requisitos con los actores involucrados que los propusieron y la razón de éstos.

- *Trazabilidad de requisitos*: Vincula los requisitos dependientes de otros, en el documento de requisitos. Esto permite evaluar el impacto de los cambios sobre los requisitos dependientes de los que reciben el cambio (propagación del cambio).

- *Trazabilidad de diseño*: Relaciona los requisitos con los módulos de diseño en los cuales son implementados. La información de aquellas relaciones sirve para evaluar el impacto de los requisitos propuestos en el diseño e implementación del sistema.

La importancia de la trazabilidad podemos decir que radica en su utilidad en diferentes aspectos de la gestión de requisitos:

*Verificación:* Permite verificar que los requisitos de software han sidos asignados tanto a diseño, como a código o a casos de prueba, asegurando de esta forma que sólo se han integrado funciones requeridas al producto.

*Reducción de Costos:* La trazabilidad permite asignar los requisitos del producto en forma temprana durante el ciclo de desarrollo. El costo de esperar hasta las fases de integración y pruebas del sistema para corregir defectos (en componentes no trazables) puede ser hasta 30 veces mayor que hacerlo en las etapas iniciales.

*Responsabilidad:* A la hora de auditar el proyecto, tanto interna como externamente, éste tendrá una mayor probabilidad de éxito si los datos están disponibles para los auditores y puedes probar que un requisito fue validado exitosamente por un caso de prueba asociado. Los administradores de proyectos tienen un mejor manejo de costos, y los clientes se aseguran de obtener el producto que pidieron.

*Administración del cambio:* Para cada cambio realizado en los requisitos es más fácil identificar qué elementos relacionados del diseño son afectados. Esto permite una documentación actualizada a medida que la implementación progresa. Además, los administradores pueden identificar los procedimientos de prueba que se deben volver a ejecutar para probar el cambio.

Por lo anterior podemos decir en pocas palabras que la trazabilidad sirve para lograr dos objetivos principales:

- Asegurar la calidad del producto, demostrando que éste tiene todas las capacidades que el cliente ha solicitado y ninguna que no haya pedido (construcción correcta del producto).

- Apoyar el análisis del impacto de los cambios sobre los requisitos, diseño y artefactos de software, así como las pruebas de software relacionadas.

Además de estas dos metas, hay algunas ventajas adicionales que son tangibles y que están relacionadas principalmente con el reporte del progreso del proyecto y para identificar el estado del proyecto. Básicamente es poder responder a las siguientes preguntas:

- ¿Cuál es el porcentaje de requisitos implementados?

- ¿Cuál es el porcentaje total de requisitos exitosamente implementados? (o sea, probados)

- ¿Cuál es el número de pruebas de aceptación de usuario que habría que volver a efectuar si se cambia la implementación de un requisito?

### 2.2.3 ¿Pre-requisitos de la trazabilidad?

La exigencia básica para trazar los requisitos, es una clara definición de los tipos de éstos. Los tipos son básicamente distintas clasificaciones de requisitos que resulten ser del mayor "sentido común" para el mapeo de éstos (funcionales, calidad, restricción, etc.). Para que un requisito sea trazable, éste debe ser referenciable, es decir tiene que existir la posibilidad de poder ser identificado sobre todo el ciclo de vida del proyecto.

### 2.2.4 ¿Cómo están trazados los requisitos?

Los requisitos están trazados explícitamente e implícitamente. La trazabilidad explicita requiere que se establezca manualmente a través de una relación entre dos o más tipos de requisitos. La trazabilidad implícita proviene de una relación inherente entre los ítems trazables. En la mayoría de los casos la trazabilidad es manejada explícitamente.

Existen diferentes enfoques para establecer la trazabilidad explicita. La relación de trazabilidad lógica entre dos requisitos (representados por sus identificadores) puede ser físicamente alcanzada en varias formas distintas. Las más comunes son las siguientes:

a) Dentro del documento de requisitos, en el lugar donde los requisitos son definidos (Ej. En referencias cruzadas)

b) Dentro del documento de requisitos, en una sección dedicada (Ej. En tablas de referencias cruzadas)

c) Externamente al documento de requisitos (Ej. A través de planillas Excel, bases de datos personalizadas o herramientas dedicadas a la administración de requisitos)

La opción a) es usualmente referida como "obtrusive traceability" debido a que los requisitos y la trazabilidad están definidos en el mismo lugar. Opciones b) y c) son referidas como "unobtrusive traceability", porque la trazabilidad es mantenida separadamente de los requisitos.

Rápidamente revisemos las diversas opciones planteadas: La alternativa a) debe ser desechada. No sólo porque el documento de requisitos puede ser difícil de mantener, sino también porque el lector se verá distraído por la información de trazabilidad dentro del documento, haciéndolo difícil de comprender. Además, la trazabilidad no es la única meta-información que se capturará para los requisitos. Otra información como prioridad, estado o riesgo también necesitan ser mantenidas. La alternativa b) representa un compromiso y puede resultar conveniente, sólo si la estrategia asegura que la trazabilidad sea establecida solamente dentro del documento o subiendo hacia los documentos estables. Esto hace que la opción c) sea al enfoque más deseable de trazabilidad.
Como se puede ver, en la mayoría de los casos establecer la trazabilidad requiere un esfuerzo manual. Consecuentemente un análisis costo-beneficio es generalmente el conductor para implementar la estrategia correcta de trazabilidad.

### 2.2.5 Trazabilidad en la práctica

Dado que las reglas de base para establecer y mantener trazabilidad son simples, a veces es duro de entender porqué muchos proyectos abandonan la trazabilidad durante el camino. Este síntoma es a menudo justificado con razones como “demasiado esfuerzo” y que los beneficios no son obvios. Estos pensamientos pueden ser explicados por lo siguiente:

i. No se ha definido ninguna estrategia de trazabilidad realizable para este proyecto.
ii. La trazabilidad no es vista como una parte integral de la ingeniería de los requisitos.
iii. Se gasta tiempo y esfuerzo, pero los beneficios nunca se observan.

La trazabilidad no puede ser hecha a medias, si uno no confía en su información de trazabilidad, es mejor no invertir el esfuerzo en primer lugar. Lo anterior puede ser sacado como moraleja de la siguiente historia: “…El proyecto está cercano a su fin. El administrador del proyecto recibe una petición de cambio seguida de una llamada del patrocinador de la empresa que quiere determinar rápidamente el impacto del cambio pedido en el proyecto. El requisito involucrado en la petición de cambio es rápidamente identificado: “La frecuencia de limpieza del ciclo”. El administrador del proyecto llama al analista responsable del área. El analista obtiene todos los documentos de requisitos relevantes y comienza a usar el motor de búsqueda del procesador de texto con algunas de las palabras claves relacionadas a los conceptos “Ciclo de Limpieza” y “Frecuencia”. El administrador ve esto y luce un poco perturbado y pregunta: “¿Por qué no usa la matriz de

trazabilidad que existió en cierto punto del proyecto?”. El analista responde rápidamente “Nadie se preocupó de mantener esas tablas actualizadas por lo que ¡no confío en ellas!”. Esto es lo que se puede evitar si logramos llevar a cabo ciertas buenas prácticas de trazabilidad de requisitos en nuestros proyectos.


FIGURA 2.1: Proceso de Software y Trazabilidad

La figura 2.1 intenta mostrar que durante el proceso de software todos los actores están involucrados (de una u otra forma) en la trazabilidad del proyecto, y por ende, si no se está dispuesto a hacer el esfuerzo de mantener actualizada la trazabilidad durante todo el desarrollo, es mejor que ni siquiera se haga el esfuerzo.

## Buenas Prácticas

A continuación se presentan algunas de las mejores prácticas que involucran una o más de las tres causas mencionadas recientemente, anotadas en paréntesis cuadrado.

*Establecer un nivel apropiado de trazabilidad.* No ser ambicioso, es decir ser consciente del nivel de detalle, las líneas de tiempo requeridas y el acercamiento o estrategia de desarrollo. Se debe pensar bien la estrategia y justificación para este “gran” esfuerzo [causa i de la sección anterior].

*Distribuir tus esfuerzos uniformemente.* Incluir el establecimiento de la trazabilidad como parte de del proceso de documentar los requisitos; no hacerlo una tarea separada. La trazabilidad tiene que estar sincronizada con las actualizaciones de los requisitos, si no se vuelve algo inútil [causas ii, iii].

*Aplicar el aseguramiento de calidad a la trazabilidad también.* Cuando se realizan revisiones o inspecciones, se debe incluir la trazabilidad, ya que esta también puede estar incorrecta. Primero hay que asegurarse que la trazabilidad está en su lugar, y luego validarla antes de que se utilice, ya que con esto aumenta la confianza de su uso [causas ii, iii].

*Ajustar y revisar tu estrategia de trazabilidad.* Una estrategia de trazabilidad que fue empleada en el pasado no se puede esperar que siga siendo válida por siempre. Los tipos de proyectos llevados a cabo por una organización, cambian o evolucionan con el tiempo. Por lo tanto, la estrategia necesita responder a estos cambios [causa i].

*Educar a tu equipo.* Trazabilidad es un esfuerzo del equipo. El equipo es responsable de realizar esta práctica y debe ser educado apropiadamente en el acercamiento de la trazabilidad del proyecto. Se debe entender y apreciar que es tan importante rastrear un requisito a su origen como lo es escribir aun definición del requisito de buena calidad [causas i, ii]

*Reflexionar durante el transcurso del proyecto pero no hacerlo una carga.* Ser diligente y reflexivo al establecer la información de trazabilidad, y no dejar que la discusión asuma el control de las reuniones del equipo [causa i].

La entrega iterativa e incremental de proyectos, anticipa y permite considerables cambios durante un proyecto, aumenta notoriamente la necesidad de trazabilidad sobre los métodos tradicionales de desarrollo del software. La capacidad de trazar tus requisitos correctamente es un prerrequisito esencial hacia conformidad de manejo en tu ambiente de negocio. Trazabilidad es un concepto esencial en el CMMI (Capability Maturity Model Integration) [Wkp07]. Esto lo podremos observar con más detalle en la siguiente sección.

### 2.2.6 Trazabilidad de los requisitos en modelos estandarizados de producción de software

Desde los inicios de la ingeniería de software, queda patente la dificultad para que los desarrollos generados alcancen un nivel de calidad óptimo, dentro de ciertos límites de tiempo y costo. Dada la naturaleza lógica del producto, se asume que la calidad de un sistema de software depende sobremanera de la calidad del proceso utilizado para desarrollarlo. Los modelos de evaluación y mejora de procesos, y su estandarización, han tomado un papel determinante en la identificación, integración, medición y optimización de las buenas prácticas existentes en la organización y desarrollo software.

En los siguientes puntos se revisará la trazabilidad de requisitos en los modelos CMMI e ISO/IEC 15504 respectivamente, con el objetivo de ver la importancia que se asigna en los modelos de calidad de software a la Trazabilidad.

#### CMMI

En el modelo CMMI, existe dentro del Nivel de madurez 2, el proceso de administración de requisitos. El objetivo de este proceso es administrar los requisitos y componentes del producto, y de esta forma poder identificar inconsistencias entre los requisitos y los planes del proyecto. Parte de la administración de los requisitos consiste en documentar los cambios de éstos y mantener la trazabilidad bidireccional, entre los requisitos fuentes y todos los artefactos/requerimientos que componen el sistema [Ram01]. La trazabilidad bidireccional es aplicada desde los productos finales a los requisitos y viceversa. La trazabilidad bidireccional se divide en trazabilidad vertical y horizontal. La trazabilidad vertical identifica el origen de los requisitos (necesidad del cliente) y hace seguimiento de éstos hasta los equipos de desarrollo e implementación del requisito, e incluso llegando eventualmente a la funcionalidad entregada al cliente. Cuando los requisitos están bien especificados, la trazabilidad desde el nivel inferior de la especificación de requisitos a la funcionalidad del software y viceversa es muy fácil de seguir. Por otra parte, la trazabilidad horizontal identifica los requisitos relacionados entre componentes del desarrollo. Por ejemplo, la trazabilidad horizontal seguiría requisitos relacionados a través de dos grupos de trabajo que desarrollan componentes asociados de un producto. La trazabilidad a través de estos dos grupos de trabajo les permite ver cuándo y cómo un cambio en un requisito (para uno de los componentes)

puede afectar a otro componente. Así, la trazabilidad horizontal permite al proyecto anticipar problemas potenciales (y atenuarlos o solucionar) antes de probar la integración [Cmm07].

### ISO/IEC 1554

ISO/IEC 15504 es un estándar internacional emergente, orientado a la evaluación y determinación de la capacidad y mejora continua de procesos de ingeniería de software. Este estándar apoya el desarrollo de un conjunto de medidas de capacidad estructuradas para todos los procesos del ciclo de vida y para todos los participantes. Es el resultado de un esfuerzo internacional de trabajo y colaboración, y tiene la innovación, en comparación con otros modelos, del proceso paralelo de evaluación empírica del resultado. En lo que respecta a la trazabilidad, este modelo hereda todas las características del modelo ISO 9000:2001. Este modelo dentro del punto 6 de control de calidad incorpora el ítem de trazabilidad e identificación, el cual describe que la organización identificará el producto/servicio por los medios que estime conveniente, donde aparece como un requisito la trazabilidad, en donde la organización controlará y registrará la identificación única del producto/servicio. La identificación se realiza conociendo cuál es el producto o el servicio que resulta de un requisito o proceso particular. Se necesita por lo tanto identificar un producto/servicio, los métodos usados y los documentos donde fueron definidos los requisitos del producto. Trazabilidad es conocer desde dónde vino un producto o servicio generado por algún proceso de la organización [Spc07].

## *2.3 Trabajos relacionados*

Con el pasar de los años la ingeniería de software ha introducido y popularizado una serie de estándares para medir y certificar la calidad de procesos y productos [Esa91, Six06]. Han surgido un número creciente de herramientas automatizadas para ayudar a definir y aplicar un desarrollo de software efectivo [Wie03]. A pesar de lo anterior, aún existe una alta incidencia de fallas en proyectos de software, retrasos en su desarrollo, presupuestos sobregirados y productos con problemas de calidad. Los procesos informales son la causa más común de estas falencias.

En la actualidad existen varias herramientas comerciales que intentan atacar este problema, a través de una especificación y administración semi-formal de los requisitos. Algunas de estas herramientas

son las siguientes: Rational RequisitePro [Ibm06], CaliberRM [Bor06], Analyst Pro [God06], Speedev [SDV06] o Verocel tool suite [Ver06]. Estas herramientas han mostrado ser útiles, pero su ámbito está restringido a un paradigma de desarrollo determinado (por ejemplo, la orientación a objetos [Som97]) o a una metodología de desarrollo específica (por ejemplo, RUP: Rational Unified Process [Kru00]). La mayoría de ellas sólo incluye una administración muy básica de los requisitos [Pre00].

Particularmente las dos primeras herramientas mencionadas anteriormente (RequisitePro y CaliberRM) están orientadas a la especificación de requisitos a través de casos de uso, utilizando los diagramas provistos por UML (Unified Modeling Language) [Fow99]. Eso significa que dichas herramientas son particularmente útiles si se utiliza una metodología de desarrollo tipo RUP o EUP (Enterprise Unified Process) [Amb05, Kru00].

La herramienta llamada Rational Requisite Pro permite crear vistas de trazabilidad que despliegan las relaciones padre e hijo de los casos de uso, gracias a las cuales se puede llegar a visualizar la forma en que un cambio en los requisitos puede afectar el flujo de la especificación del caso de uso. Además, gracias a esto también se pueden crear vistas particulares que ayudan a mejorar la visibilidad de los requisitos por sobre el proyecto como por ejemplo, la creación de vistas de las características que aún no se hayan implementado en cierto desarrollo. Esta herramienta también despliega avisos si algún requisito es modificado, y hasta posee la capacidad de enviar e-mails de notificación de cambios a las personas involucradas. Por otra parte, CaliberRM provee la posibilidad de definir una jerarquía consistente de tipos de requisitos a través de los cuales permite automatizar la trazabilidad de estos. Además permite almacenar los distintos puntos de vista de los involucrados en el desarrollo, con respecto a los cambios de un respectivo requisito, guardándolos en un historial.

En general los indicadores que despliegan las herramientas orientadas a casos de uso (en nuestro caso CaliberRM y RequisitePro) son principalmente avisos de los posibles conflictos que pueden ocurrir, entre lo que existe en el servidor central que almacena los requisitos y lo nuevo que se está intentando agregar, de forma similar a como ocurre cuando se utiliza un CVS [Cvs06].

En el caso de que se utilicen metodologías más tradicionales, concebidas antes de la era UML, dichas herramientas no sirven demasiado, ya que la estrategia de especificación y administración de requisitos es diferente. En este último escenario, se especifican requisitos de grano fino y se los

administra uno a uno. Aunque esto generalmente es una tarea pesada, este tipo de manejo de los requisitos minimiza la sorpresa al finalizar el proyecto. Analyst Pro, Speedev y Verocel Tool Suite se acercan más a este último enfoque en cuanto a la administración de requisitos.

Tanto Speedev como Verocel poseen un módulo para realizar consultas a los requisitos y chequear la trazabilidad de estos con los tests o casos de prueba a los cuales están vinculados. Además pueden revisar la cantidad de requisitos incumplidos o que posean alguna otra característica en particular. Adicionalmente, tanto Speedev como Analyst Pro permiten crear vistas o consultas predefinidas que facilitan la visibilidad de la administración y el estado de los requisitos.

# 3. Proceso de administración de requisitos

El proceso genérico de administración y especificación de requisitos está compuesto de varias tareas esenciales. A continuación serán presentadas secuencialmente, sin embargo, en un proceso de ingeniería de requisitos efectivo, estas actividades son aplicadas de manera continua y en orden variado.

## *3.1 Actividades relevantes*

Dependiendo del tamaño del proyecto y del modelo de proceso de software utilizado para el ciclo de desarrollo, las actividades de la ingeniería de requisitos varían tanto en número como en nombres. La tabla 3.1 muestra algunos ejemplos de las actividades identificadas para cada proceso. A pesar de las diferentes interpretaciones que cada desarrollador tenga sobre el conjunto de actividades mostradas en la siguiente tabla, podemos identificar cinco actividades principales, las cuales son:

- Análisis del Problema
- Evaluación y Negociación
- Especificación
- Validación
- Evolución

| Modelo | Oliver & Steiner 1996 | EIA / IS-632 | IEEE Std 1220-1994 | CMM nivel repetitivo (2) | RUP |
|---|---|---|---|---|---|
| | | | | | |
| **Actividades** | Evaluar la información disponible | Análisis de requisitos | Análisis de requisitos | Identificación de requisitos | Análisis del problema |
| | Definir métricas efectivas | Análisis funcional | Estudio de los requisitos | Identificación de restricciones del sistema a desarrollar | Comprender las necesidades de los involucrados |
| | Crear un modelo del comportamiento del sistema | Síntesis | Validación de requisitos | Análisis de los requisitos | Definir el sistema |
| | Crear un modelo de los objetos | Análisis y control del sistema | Análisis funcional | Representación de los requisitos | Analizar el alcance del proyecto |
| | Ejecutar el análisis | | Evaluación y estudio de funciones | Comunicación de los requisitos | Modificar la definición del sistema |
| | Crear un plan secuencial de construcción y pruebas | | Verificación de funciones | Validación de requisitos | Administrar los cambios de requisitos |
| | | | Síntesis | | |
| | | | Estudio y evaluación del diseño | | |
| | | | Verificación física | | |
| | | | Control | | |

TABLA 3.1. Actividades de la IR para diferentes modelos de procesos de Ingeniería de Software [Itm05], [SST05], [Wkp05]

A continuación se detallará en cada una de las cinco actividades principales identificadas.

### 3.1.1 Análisis del problema

Esta es la primera de las cinco actividades anteriormente mencionadas. El objetivo de esta actividad es entender las verdaderas necesidades del negocio. Antes de describir qué pasos deben cumplirse en ella, debemos tener una definición clara del término “Problema”. A través de la definición del problema, podemos ver entonces que la actividad de “Análisis del problema” tiene

por objetivo que se comprendan las dificultades del negocio, se evalúan las necesidades iniciales de todos los involucrados en el proyecto y que se proponga una solución de alto nivel para resolverlo. Durante el análisis del problema, se realizarán una serie de pasos para garantizar un acuerdo entre los involucrados, basados en los problemas reales del negocio. Estos pasos son los siguientes:

- *Comprender el problema que se está resolviendo*. Es importante determinar quién tiene el problema realmente, considerar dicho problema desde una variedad de perspectivas y explorar muchas soluciones desde diferentes puntos de vista.

- *Construir un vocabulario común*. Debe confeccionarse un glosario en dónde se definan todos los términos que tengan significados comunes y que serán utilizados durante el proyecto. Esta acción es sumamente beneficiosa ya que reduce los términos ambiguos desde el principio.

- *Identificar a los afectados por el sistema*. Esto evita que existan sorpresas a medida que avanza el proyecto. Las necesidades de cada afectado, son discutidas y sometidas a debate durante la ingeniería de requisitos, aunque esto no garantiza que vaya a estar disponible toda la información necesaria para especificar un sistema adecuado.

- *Definir los límites y restricciones del sistema*. Esto es sumamente importante, se debe tener claro lo que se está construyendo y lo que no, para de esta forma entender la estrategia del producto a corto y largo plazo. Debe determinarse cualquier restricción ambiental, presupuestaria, temporal, técnica y de factibilidad que limite el sistema que se va a construir.

### 3.1.2 Evaluación y negociación de los requisitos

La diversa gama de fuentes de las cuales provienen los requisitos, hace necesaria una evaluación de los mismos con el objetivo de determinar si tienen un nivel aceptable de riesgo, tomando en cuenta la factibilidad técnica y económica, para el cliente. Dependiendo de esta evaluación inicial, en esta etapa se pretende limitar las expectativas del cliente apropiadamente,

tomando como referencia los niveles de abstracción y descomposición de cada problema presentado. Los principales pasos de esta actividad son:

- *Descubrir problemas potenciales*: Identificar requisitos ambiguos, incompletos, inconsistentes, etc.

- *Clasificar los requisitos*: En este paso se deben priorizar los requisitos, identificar la importancia que tienen en término de implementación y así establecer la secuencia en que ocurrirán las actividades de diseño y prueba de cada requisito. La prioridad de cada requisito dependerá de las necesidades que tenga el negocio. En base a esta, cada requisito puede ser clasificado como crítico, deseable o innecesario. Una vez hecha esta categorización puedo tomar como estrategia general el incluir los críticos, discutir los deseables y descartar los innecesarios. Antes de decidir la inclusión de un requisito debe analizarse su costo, complejidad y una cantidad de otros factores.

- *Evaluar factibilidades y riesgos*: Involucra la evaluación de factibilidades técnicas (¿pueden implementarse los requisitos con la tecnología actual?), factibilidades operacionales (¿puede ser el sistema utilizado sin alterar el organigrama actual?), factibilidades económicas (¿ha sido aprobado por los clientes el presupuesto?).

### 3.1.3 Especificación de requisitos de software

La especificación de requisitos de software es el resultado final de las actividades de análisis y evaluación de requisitos. Los clientes y usuarios utilizan está especificación para comparar si lo que se está proponiendo, coincide con las necesidades de la empresa. Los analistas y programadores la utilizan para determinar el producto que debe desarrollarse. El personal de pruebas elaborará las pruebas en base a este documento. Para el administrador del proyecto sirve como referencia y control de la evolución del sistema.

La estandarización del documento de esta etapa es fundamental pues ayudará, entre otras cosas, a facilitar la lectura y escritura de la misma. Será un documento familiar para todos los involucrados, además de asegurar que se cubren todos los tópicos importantes.

### 3.1.4 Validación de requisitos

Esta actividad permite demostrar que los requisitos definidos en el sistema son los que realmente quiere el cliente, además que revisa que no se haya omitido ninguno, que no sean ambiguos, inconsistentes o redundantes. Esta validación debe garantizar que todos los requisitos presentes en el documento de especificación sigan los estándares de calidad. No debe confundirse la actividad de evaluación de requisitos con la validación de requisitos. La evaluación verifica las propiedades de cada requisito, mientras que la validación revisa el cumplimiento de las características de la especificación de requisitos. La validación de requisitos es importante pues de ella depende que no existan elevados costos de mantenimiento para el software desarrollado.

### 3.1.5 Evolución de requisitos

Los requisitos son una manera de comprender mejor el desarrollo de las necesidades de los usuarios y como los objetivos de la organización pueden cambiar, por lo tanto, es esencial planear posibles cambios a los requisitos cuando el sistema sea desarrollado y utilizado. La actividad de evolución es un proceso externo que ocurre a lo largo del ciclo de vida de proyecto. Es común que en desarrollos largos los requisitos cambien por diferentes razones Las más frecuentes son:

- Porque al analizar el problema, no se hacen las preguntas correctas a las personas correctas.
- Porque cambió el problema que se estaba resolviendo.
- Porque los usuarios cambiaron su forma de pensar, sus percepciones o necesidades.
- Porque cambió el ambiente de negocios.
- Porque cambió el ambiente sobre el cual se desenvuelve el negocio.

Un cambio en los requisitos involucra modificar el tiempo en el que se va a implementar una característica en particular, modificación que a la vez puede tener impacto en otros requisitos. En

vista que las peticiones de cambios provienen de muchas fuentes deben ser encausadas en un solo proceso. Esto se hace con la finalidad de evitar problemas y conseguir estabilidad en los requisitos.

## *3.2 Estándar de la European Space Agency (ESA)*

Anteriormente se han mencionado varios estándares de desarrollo y se ha especificado en que consiste la ingeniería de requisitos y cuales son las fases o actividades que la componen. El trabajo realizado durante el desarrollo de esta tesis se ha basado principalmente en el estándar de la European Space Agency (ESA) [ESA91] para el desarrollo de un proyecto de software. Las principales razones por las cuales se tomó esta decisión son:

- Puede ser adaptado a la realidad de una empresa específica.
- Puede ser aplicado en forma parcial, según la naturaleza del problema a resolver.
- Es simple, por lo tanto con poca práctica se puede hacer buen uso de él.

El estándar de la ESA propone un conjunto de actividades para abordar el ciclo de vida del software. Este ciclo se inicia cuando el producto de software se concibe, y termina cuando este ya no está disponible para su uso. Un modelo de ciclo de vida, estructura las actividades de un proyecto en fases, y define las actividades de cada una de estas. En particular, las fases que define la ESA son las siguientes:

- UR: Definición de requisitos de usuarios.
- SR: Definición de requisitos de software.
- AD: Definición del diseño arquitectónico.
- DD: Diseño detallado y producción del código.

- TR: Transferencia del software a operaciones.

- OM: Operación y mantenimiento.

A su vez cada una de estas etapas está delimitada los siguientes hitos:

- Aprobación del Documento de Requisitos de Usuario (URD).

- Aprobación del Documento de Requisitos de Software (SRD).

- Aprobación del Documento de Diseño Arquitectónico (ADD).

- Aprobación del Documento de Diseño Detallado (DDD), el manual de software de usuario (SUM), el código, y la declaración de terminación para la revisión o testeo de aceptación provisional.

- Declaración de aceptación provisional y la confección del Documento de Transferencia de Software (STD).

- Declaración de aceptación final y entrega del Documento de Historia del Proyecto (PHD).

La siguiente figura muestra las fases y los ítems de cada una de estas que contempla este estándar.

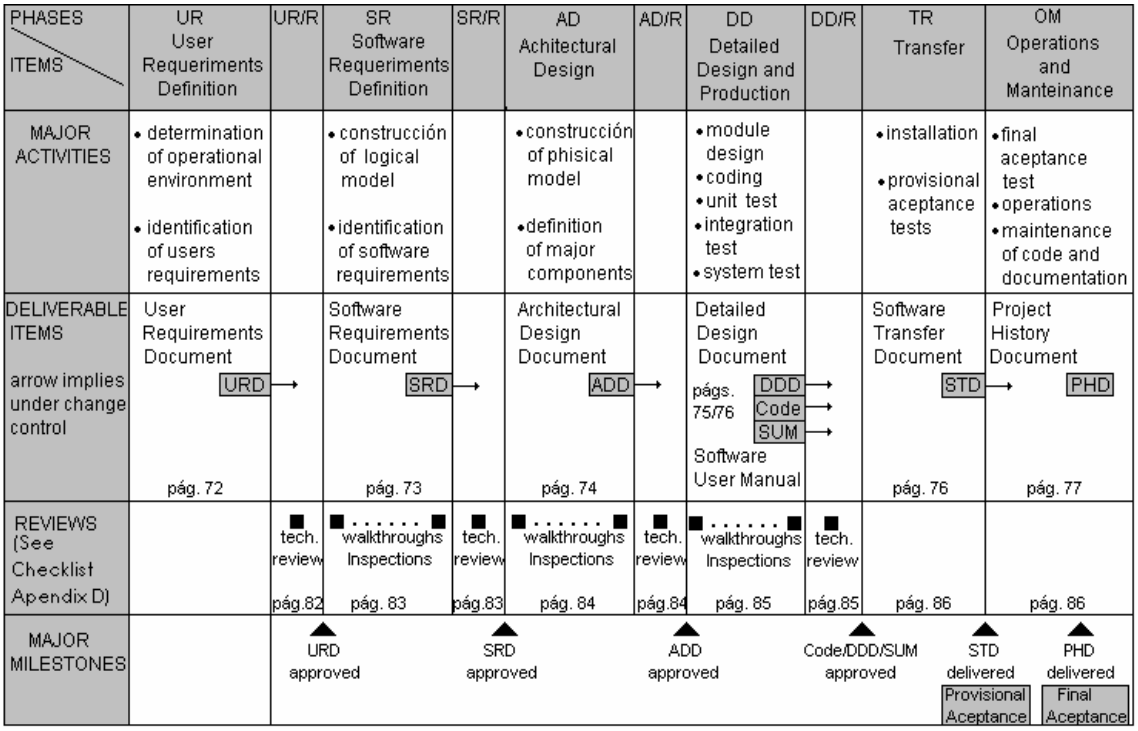


FIGURA 3.1 Estructura de un ciclo/iteración de software según el estándar de la ESA.

A continuación una descripción de las fases mencionadas anteriormente.

### 3.2.1 Fase de requisitos de usuario (UR)

Esta fase también puede ser llamada como la fase de definición del problema del ciclo de vida. El propósito principal de esta fase es refinar una idea acerca de la tarea a ser desarrollada. La definición de requisitos de usuario debe ser responsabilidad del usuario. La experiencia de los ingenieros de software, ingenieros de hardware y personal de operación debe ser usada para ayudar a definir y revisar los requisitos de usuario, aunque el apoyo de los desarrolladores en esta etapa varía según la familiaridad de los usuarios con el software. Los requisitos de software deben ser capturados ya sea por medio de entrevistas, encuestas, formularios o prototipos. Durante esta fase se debe generar un Documento de Requisitos de Usuario (URD). La revisión (UR/R) debe ser hecha por los usuarios, los ingenieros de hardware y software, y por el administrador del proyecto. Antes de completar la UR/R se debe construir un Plan de Administración del Proyecto de Software que muestre todas las fases siguientes con estimación de recursos respectiva.

### 3.2.2 Fase de requisitos de software (SR)

Esta fase es la etapa de análisis del problema del ciclo de vida. Su propósito es analizar el estado de los requisitos de usuario y producir un conjunto de requisitos de software lo más completa, correcta y consistente posible.

La definición de Requisitos de Software debe ser responsabilidad del desarrollador. Los participantes de esta fase deben incluir al usuario, ingenieros de software, ingenieros de hardware y personal de operación. Todos ellos tienen distinto concepto del producto final y ese concepto debe ser analizado y luego sintetizado en un completo y consistente listado de requisitos en los cuales todos estén de acuerdo (etapa de revisión de requisitos de software, SR/R).

Es vital que en esta etapa se haga una descripción del modelo del software, que especifique lo que debe hacer y no cómo debe hacerlo, por lo que debe omitirse en lo posible terminología de implementación. Además puede ser necesario crear prototipos para clarificar requisitos de software y completar requisitos de usuarios. El entregable principal es el Documento de Requisitos de Software (SRD). Cada proyecto de software debe tener este documento. En esta fase se debe revisar el Plan de Administración del Proyecto de Software.

### 3.2.3 Fase de diseño arquitectónico (AD)

El propósito de esta fase es el de definir la estructura del software. El modelo construido en la fase de requisitos de software es el punto de partida. Este modelo es transformado en diseño arquitectónico asignando funciones a componentes de software y definiendo el control y flujo de datos entre ellos. Esta fase pude considerar varias iteraciones del diseño.

En esta fase deben ser identificadas las dificultades técnicas o partes críticas del diseño. Además puede ser necesario realizar prototipos del software para confirmar las suposiciones básicas del diseño. El ítem entregable que constituye la salida formal de esta fase es el Documento de Diseño Arquitectónico (ADD), el cual debe ser producido para cada proyecto de software. El ADD debe ser formalmente revisado por los ingenieros de software y de hardware, por los usuarios y por el administrador del proyecto, durante el proceso de revisión del Diseño Arquitectónico (AD/R).

Durante la fase de Diseño Arquitectónico, debe construirse un Plan de administración de Proyecto de Software, que describa el resto del proyecto. Este plan debe contener una estimación del costo del proyecto, apuntando a minimizar el margen de error (idealmente no debe superar el 10%). También se debe producir planes detallados para la fase DD.

### 3.2.4 Fase de diseño detallado y producción (DD)

El propósito de esta fase es el de detallar el diseño del software, codificarlo, documentarlo y probarlo. En forma concurrente con la codificación y las pruebas, se produce el Documento de Diseño Detallado (DDD) y el Manual de Software de Usuario (SUM). Inicialmente, el DDD y SUM contienen las secciones correspondientes a los niveles superiores del sistema. A medida que el diseño progresa a niveles más bajos, se añaden sub-secciones relacionadas. Al final de la fase, los documentos están completos y, junto con el código constituyen los ítems entregables de esta etapa.

Durante esta fase, debe realizarse las actividades de pruebas unitarias, de integración y de sistema de acuerdo con los planes de verificación establecidos en las fases de requisitos de software y Diseño Arquitectónico. También debe verificarse la calidad del software. Los tres entregables (código, DDD y SUM), deben ser revisados formalmente por los ingenieros de software y el administrados, durante le proceso de Revisión del Diseño Detallado (DD/R). Al final del proceso de revisión, el software puede considerarse como listo para la prueba de aceptación provisional.

### 3.2.5 Fase de transferencia (TR)

El propósito de esta fase es el de establecer que el software cumple con los requisitos especificados en el URD. Esto es hecho instalando el software y realizando las pruebas de aceptación. Si el software demuestra que provee las capacidades requeridas, este puede ser aceptado provisionalmente y con ello puede empezar la operación. Durante la fase de transferencia, debe producirse el Documento de Transferencia de Software (STD), que documente el proceso de transferencia de software al equipo de operaciones.

### 3.2.6 Fase de operación y mantenimiento (OM)

Una vez que el software ha entrado en operación, debe ser monitoreado cuidadosamente para confirmar que cumpla con los requisitos definidos en el URD. Algunos de dichos requisitos, tales como los referentes a disponibilidad, puede tomar algún tiempo validarlos. Cuando el software ha pasado todas las pruebas de aceptación, entonces recién puede ser finalmente aceptado.

El documento de Historia del proyecto (PHD) resume la información administrativa de importancia acumulada en el transcurso del proyecto. Este documento debe ser generado después de la aceptación final. Debe ser rehecho al final del ciclo de vida, con información acumulada durante la fase de operación y mantenimiento. Después de la aceptación final, el software puede ser modificado para corregir errores no detectados durante fases anteriores, o porque aparecen nuevos requisitos. A esto se le denomina mantenimiento. Durante todo el período de operación, se debe hacer un esfuerzo especial para mantener la documentación al día. Información sobre fallas y caídas debe ser registrada para establecer datos para el establecimiento de métricas de calidad de software para proyectos siguientes.

# 4. Indicadores y alarmas

A continuación se describirán los indicadores y las alarmas que se definieron en este trabajo, indicando cuál es su función y para qué pueden utilizarse. Hay que recordar que el desarrollo de estas alarmas e indicadores, fue concebido para mejorar la trazabilidad de los proyectos y por ende, facilitar la labor de los analistas.

## *4.1 Indicadores*

La segunda hipótesis de trabajo planteaba que la generación automática de indicadores permitiría a los desarrolladores de software obtener rápidamente un diagnóstico del estado del proyecto, en términos de avance, trabajo pendiente, perdida/cambio de requisitos y prioridades. Con esto se mejorará tanto la calidad de los productos desarrollados en la fase de análisis como la visibilidad de esta fase sobre el proceso completo de desarrollo permitiendo que los riesgos sean identificados en forma temprana. Gracias a esto se logra tomar decisiones a tiempo y así corregir situaciones no deseadas durante el proyecto. Los indicadores definidos en este trabajo son; matrices de trazado, vista rápida de requisitos, estadísticas y un árbol de relaciones, los cuales serán descritos a continuación.

### 4.1.1 Matrices de trazado

Es la técnica más común y aplicable a cualquier modelo de desarrollo. Consiste, tal como lo dice su nombre, en crear una matriz cuyas columnas y filas correspondan a dos características relacionadas y su intersección se marca con una cruz, si es que existe una correlación entre ellas. Esta herramienta hace posible el análisis de las relaciones entre sus elementos. Se definieron las siguientes vistas disponibles para las matrices en la herramienta:

- *Requisitos de usuario vs. requisitos de software:* Con esta vista se puede ver claramente la correlación entre requisitos de usuario y requisitos de software.

- *Requisitos de software vs. módulos:* Gracias a esta vista se puede visualizar de que forma están mapeados los requisitos en los distintos módulos implementados.

- *Requisitos vs. casos de prueba:* Por medio de esta vista se puede observar de que forma los requisitos, tanto de usuario como de software, están vinculados a los distintos casos de pruebas ingresados en la herramienta.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Proyecto :Grupo 2: Sistemas de Publicaciones del DCC | | | | | |
| 2 | | RS0001 | RS0002 | RS0003 | RS0004 | RS0005 |
| 3 | RU0001 | X | | | | |
| 4 | RU0002 | | X | | | |
| 5 | RU0003 | | | X | | |
| 6 | RU0004 | | | | X | |
| 7 | RU0005 | | | | | X |

FIGURA 4.1. Matriz de trazado

Una ventaja de estas matrices es que si se actualizan continuamente, pueden servir como herramienta de soporte muy útil para el personal encargado del mantenimiento y las pruebas del sistema.

### 4.1.2 Vista rápida de requisitos

Esta vista es básicamente para facilitar la visibilidad de los requisitos (de usuario y de software) por parte de los integrantes del grupo de desarrollo. Se trata más que nada de un despliegue del código y la descripción del requisito permitiendo además realizar filtros por el estado del requisito (cumple, no cumple, ambiguo), por la prioridad (crítica, deseable, innecesaria) y por el tipo de usuario al que esté asociado.


FIGURA 4.2. Vista rápida de requisitos.

### 4.1.3 Despliegue de árbol de relaciones

Al igual que la vista rápida de requisitos, este árbol fue concebido para facilitar la visualización de una forma rápida de algunas inconsistencias en el estado de los productos de desarrollo. Para lograrlo se utilizó un código de colores para que sea fácil identificar situaciones anómalas en el estado de los productos a simple vista (un ejemplo de esto sería encontrar un requisito de usuario cuyo estado es “no cumple” vinculado a un requisito de software con estado “cumple”). Las vistas disponibles son

- *Requisitos de usuarios vs. requisitos de software:* En esta vista al seleccionar un requisito de usuario se muestran los requisitos de software y los tipos de usuario asociados a él utilizando el código de colores anteriormente mencionado.

- *Requisitos de software vs. requisitos de usuario:* Esta vista es complementaria a la anterior. Aquí se despliegan los requisitos de usuario y los tipos de usuario asociados al requisito de software seleccionado.

- Requisitos vs. casos de prueba: Este árbol de relaciones muestra como nodos los casos de pruebas y como hojas los requisitos asociados a cada nodo o caso de prueba y los tipos de usuario asociados al caso de prueba.

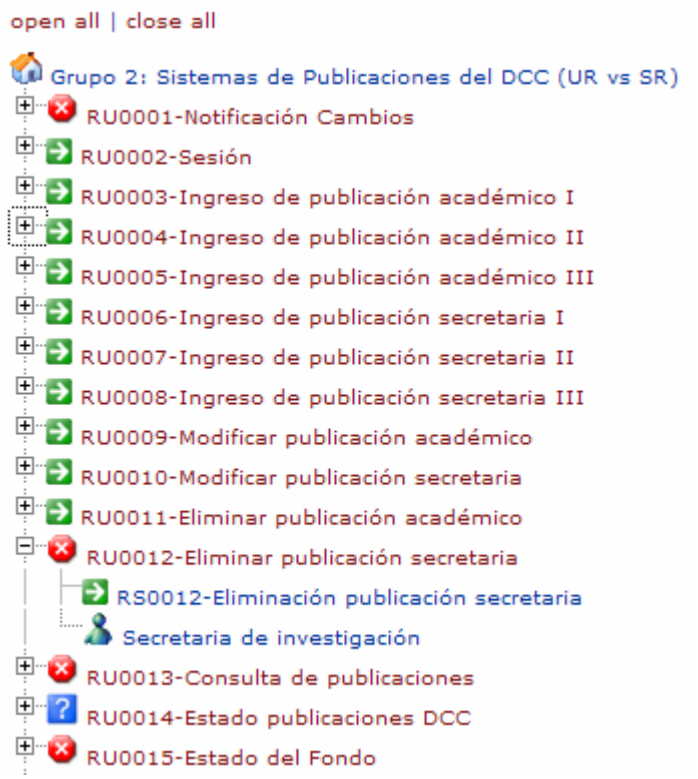


FIGURA 4.3. Árbol de relaciones (UR vs. SR).

### 4.1.4 Estadísticas

Además de los indicadores anteriores señalados, también existe un despliegue de estadísticas generales. Están subdivididos en cuatros secciones; requisitos de usuario, requisitos de software, casos de prueba y módulos. Con respecto a las estadísticas que se despliega y que relacionadas con los requisitos de usuario podemos decir que son los siguientes:

- Número de requisitos de usuario por cada tipo (funcional, calidad y restricción).
- Cantidad de requisitos de usuario agrupados por prioridad (crítica, deseable, innecesaria).
- Número de requisitos de usuario por estado (cumplidos, no cumplidos, ambiguos)

- Cantidad de requisitos de usuario por estabilidad (transable, intransable).
- Número de requisitos de usuario críticos no cumplidos.
- Número de requisitos de usuario deseables no cumplidos.
- Número de requisitos de usuario innecesarios no cumplidos.
- Número de requisitos de usuario que no tienen tipo de usuario asociado.
- Cantidad de requisitos de usuario sin requisitos de software asociado.

Básicamente se puede decir que estas estadísticas pueden servir al jefe de proyecto para llevar un historial del avance del proyecto, comparando en las distintas semanas los distintos valores entregados por la aplicación. Con esto se puede saber al momento cuantos requisitos importantes quedan aún por cumplir y tener una mejor noción de cómo está el estado del proyecto.

En lo que se refiere a los requisitos de software, las estadísticas son muy similares a las anteriores:

- Cantidad de requisitos de software agrupados por tipo (funcional, usabilidad, mantenibilidad, etc.).
- Número de requisitos de software por prioridad (crítica, deseable, innecesaria).
- Número de requisitos de software agrupados por estado (cumplido, no cumplido, ambiguo).
- Cantidad de requisitos de software agrupados por estabilidad (transable, intransable).
- Número de requisitos de software críticos no cumplidos.
- Número de requisitos de software deseables no cumplidos.
- Número de requisitos de software innecesarios no cumplidos.

- Cantidad de requisitos de software sin algún tipo de usuario asociado.

- Cantidad de requisitos de software sin requisito de usuario asociado.

Como se puede ver, las estadísticas de los requisitos de software son muy similares a los de los requisitos de usuario y apuntan al objetivo de facilitar la tarea de conocer el estado actual del proyecto sabiendo, por ejemplo, cuántos requisitos están cumplidos o si tienen un tipo de usuario asociado.

Para los casos de prueba se despliegan los siguientes indicadores:

- Número total de casos de prueba en el sistema.

- Cantidad de casos de prueba asociados a requisitos de usuario.

- Cantidad de casos de prueba asociados a requisitos de software.

- Número de casos de prueba sin tipos de usuario asociados.

Las estadísticas de los casos de prueba son menos detalladas que las que existen para los requisitos de usuario o de software. Fueron creadas con la intención de mejorar la noción de cuantos casos de prueba existen en la herramienta vinculadas al proyecto.

Finalmente para los módulos, solo se despliegan dos indicadores que mencionamos a continuación:

- Número de módulos totales ingresados en el sistema.

- Número de módulos sin requisitos de software asociados.

Con respecto a estos dos últimos indicadores es importante tenerlos en cuenta, ya que gracias a ellos podemos saber cuantos módulos existen que no están asociados a algún requisito, es decir módulos ingresados al sistema pero que no están contemplados en la etapa de análisis.

## 4.2 Alarmas

Además de los indicadores, también se desarrollaron alarmas en los requisitos de usuario y de software. Con alarmas, nos referimos a unos indicadores que se despliegan cuando algún requisito tiene potencialmente algún problema en la especificación y es conveniente que el analista lo revise con más cuidado.

Las alarmas implementadas para los requisitos de usuario se listan a continuación:

- *Requisitos de usuario con más de cinco requisitos de software asociados:* Se definió esta alarma debido a que esta situación puede ser producto de una mala especificación en los requisitos. Podría significar que el requisito es demasiado complejo y podría ser conveniente dividirlo.

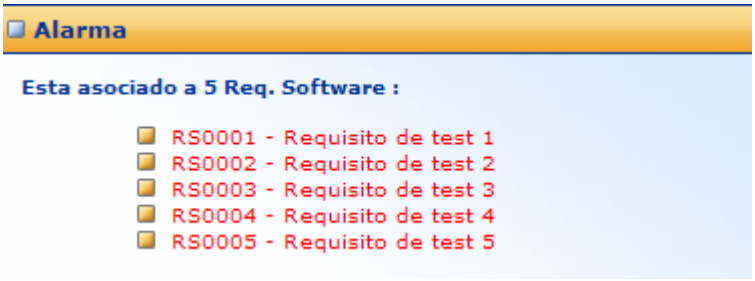


FIGURA 4.4. Ejemplo de alarma de asociación.

- *Requisitos de usuario sin requisitos de software asociados:* Si esto ocurre significa que existe algo anómalo en la especificación.


FIGURA 4.5. Segundo ejemplo de alarma de asociación.

- *Prioridad de requisitos de usuario es menor a la máxima prioridad de los requisitos de software asociados:* Esta alarma fue concebida para chequear las prioridades en la especificación. Un ejemplo de esta situación es lo que se muestra en la figura 4.6 donde existe un requisito de usuario con prioridad deseable, asociado a un requisito de software con prioridad crítica.

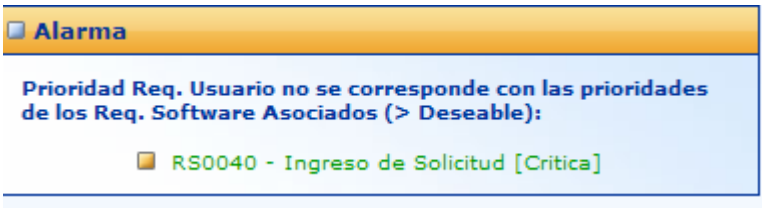


FIGURA 4.6. Alarma de prioridades de requisitos de usuario.

Para los requisitos de software se crearon las siguientes alarmas:

- *Requisitos de software con tres o más requisitos de usuario asociado:* Al igual que con los requisitos de usuario, esta alarma se definió debido a que esta situación se puede deber a una mala especificación. Cabe la posibilidad de que alguno de los requisitos deba dividirse.

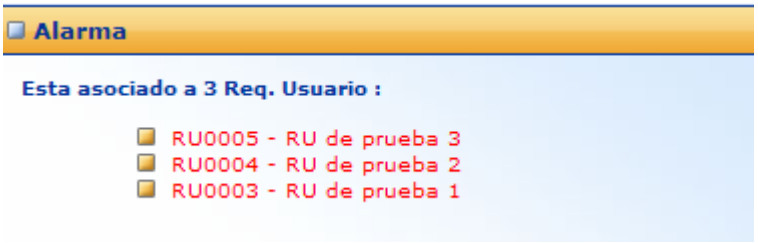


FIGURA 4.7. Alarma de asociación de requisitos de software.

- *Requisitos de software sin requisito de usuario asociado:* Esta es una situación que no debe ocurrir. En la etapa de análisis no puede existir un requisito de software que no esté asociado a un requisito de usuario.


FIGURA 4.8. Segundo ejemplo de Alarma de asociación de requisitos de software.

- *Requisito de software intransable asociado a requisito de usuario transable:* Esta alarma permite detectar esta inconsistencia en las especificaciones y así evitar malos entendidos con el cliente. En la figura 4.9 se puede visualizar este caso.

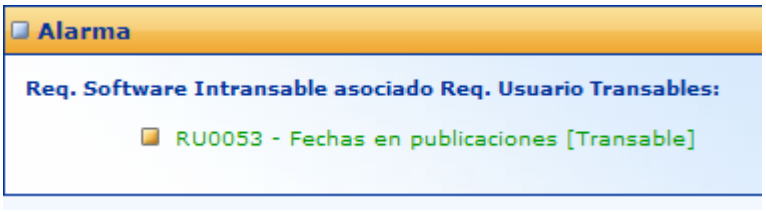


FIGURA 4.9. Alarma de estabilidad de requisitos de software.

- *Requisitos de software con prioridad menor o igual a la máxima prioridad de los requisitos de usuario asociados:* Esta situación se produce posiblemente por un error en la especificación. En la siguiente figura se puede observar la alarma que se genera en el caso de que un requisito de software de prioridad crítica está asociado a un requisito de usuario de prioridad deseable.

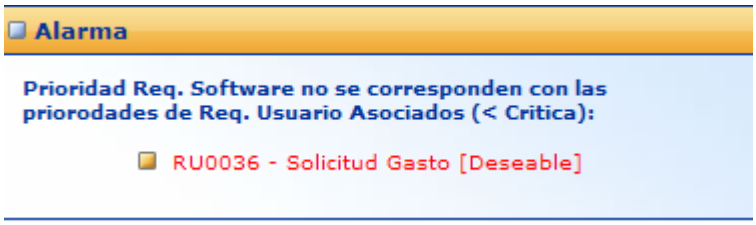


FIGURA 4.10. Alarma de prioridad de requisitos de software.

Todos los indicadores y alarmas mencionadas han sido pensados para facilitar la tarea de los analistas y ayudarlos a descubrir potenciales problemas en las especificaciones. Si es necesario hacer un cambio en algún requisito, con las herramientas de visualización, es más fácil tener una noción de lo que involucra esta modificación dentro del proyecto. La automatización de estos mecanismos de trazabilidad e indicadores, pueden ayudar a reducir los costos y tiempos de desarrollo, y a disminuir los riesgos asociados al incumplimiento de requisitos. Esto se traduciría en productos más baratos y confiables.

# 5. La Herramienta: ReqAdmin

Esta sección presenta los detalles de la implementación de ReqAdmin, y de los indicadores incluidos. Como parte de dicha descripción se detalla su ambiente operacional, su diseño y su interfaz de usuario. El modelo de datos de ReqAdmin se puede observar en el anexo (A.2). Además de lo anterior, también se entrega una explicación acerca del formato utilizado para representar los requisitos utilizando la herramienta.

El propósito de la herramienta es facilitar la administración y visibilidad de los requisitos de un proyecto de software para todos los involucrados en el desarrollo de este.

Existen dos tipos de actividades que se pueden realizar, utilizando ReqAdmin, para administrar requisitos: actualización de datos (inserción, modificación) y obtención de reportes e indicadores.

Por medio de esta herramienta se pretende facilitar la especificación de los requisitos que describan el comportamiento de un sistema a desarrollar, además de servir como un medio de información para todas las personas involucradas en el proyecto debido a su carácter web. La ingeniería de requisitos depende de una intensa comunicación entre clientes, analistas y equipo de trabajo. Con ReqAdmin se pretende superar en parte las barreras de comunicación que aparecen durante este proceso.

Habitualmente se tiene problemas para visualizar las relaciones existentes entre requisitos. Por medio de ReqAdmin deberían disminuir estos problemas ya que permite manejar y visualizar estas relaciones a través de matrices de trazado y árboles de relaciones. También permite diferenciar los grados de importancia de los requisitos mediante un campo de prioridad con tres estados (Crítico, Deseable o Innecesario).

Además de lo mencionado anteriormente la herramienta puede generar documentos con los listados de requisitos lo que facilita la tarea al analista de requisitos y le permite tener acceso a estos documentos al resto del grupo de trabajo. Pero cómo podemos enmarcar el uso de ReqAdmin dentro de las actividades relevantes de la ingeniería de requisitos enunciadas en 3.1 (Análisis del problema, evaluación y negociación, especificación de requisitos de software, validación y evolución). A continuación tomaremos en cuenta este punto.

Durante la etapa de análisis del problema, donde se pretende entender las necesidades del negocio, con la herramienta desarrollada se pueden enunciar los requisitos de usuario y asociarlos a los distintos tipos de usuarios involucrados. Con esto se puede identificar a los afectados con el sistema y evita que existan sorpresas a medida que avanza el proyecto. Por medio de los requisitos de restricción se pueden definir los límites del sistema. Además de lo anterior, la herramienta sirve como repositorio central de los requisitos, debido a que provienen de varias fuentes.

En la evaluación y negociación de requisitos, esta herramienta puede jugar un papel fundamental ya que permite etiquetar a los requisitos como ambiguos y por medio de sus relaciones se pueden ver que tan consistentes son. Además permite priorizar los requisitos e identificar su estabilidad, lo cual dependerá de las necesidades del negocio.

El resultado final de las actividades de análisis y evaluación de requisitos es la especificación de requisitos de software. ReqAdmin permite esta especificación además de que es capaz de generar el documento respectivo. Por medio de la matriz de trazado (requisitos de usuario vs. requisitos de software) se puede verificar el estado de la especificación de requisitos y determinar el producto que debe desarrollarse. El personal de pruebas elaborará los casos de prueba en base a esto y al administrador del proyecto le sirve como referencia y para poder controlar la evolución del sistema.

Debido a que ReqAdmin es una herramienta que mejora la visibilidad de los requisitos para todos los involucrados entonces se facilita la etapa de validación de requisitos donde se quiere demostrar que los requisitos definidos en el sistema son los que realmente quiere el cliente.

Finalmente ReqAdmin también tendrá un rol fundamental en la fase de evolución de requisitos. Por medio de esta herramienta pueden ser enrutadas todas las peticiones de cambio que provienen generalmente de muchas fuentes en un solo proceso, lo que permite evitar problemas y conseguir estabilidad en los requisitos.

Esta herramienta está inscrita como proyecto en ChileForge (http://chileforge.cl) el cual es un repositorio de proyectos Open Source cuyo objetivo es impulsar el desarrollo y la adopción de software libre a nivel nacional, tanto por parte de personas naturales y como empresas. Se hizo esto con el propósito de difundir el software. Se encuentra en este lugar bajo licencia GPL.

## *5.1 Ambiente de Operación*

El desarrollo de la herramienta se realizó sobre HTML, Javascript, PHP, Apache y MySQL debido a que cuentan con varios atributos deseables: fácil de desarrollar y por tanto de extender, con capacidad de mostrar una interfaz intuitiva, dónde es más sencillo lograr un ambiente distribuido y que si es desarrollada correctamente puede operar en múltiples plataformas sin mayores inconvenientes. En nuestro caso la herramienta ha sido probada tanto en Linux como Windows funcionando sin problemas.

Una de las características deseables para la herramienta es la facilidad de instalación. Para Microsoft Windows hay soluciones como XAMPP que resumen la instalación a un par de clicks. Linux no es realmente un problema pues hasta sus distribuciones más sencillas tienen incluido PHP, Apache y MySQL. Por otro lado la instalación se realiza una sola vez por el lado del servidor, lo que libera a los usuarios de este trámite.

El propósito de desarrollar esta aplicación bajo la filosofía del Software Libre es pensando principalmente en la mejora de la herramienta, ya que de esta forma se puede añadir la oportunidad de incorporar los aportes de otras personas. La herramienta ha sido desarrollada y probada utilizando el navegador Mozilla Firefox 2.0. Por problemas de compatibilidad de Microsoft Internet Explorer 6.0 con la especificación estándar de CSS (Cascading Style Sheets) no se puede desplegar la información correctamente en este navegador por el momento.

Para la construcción de las planillas Excel se utilizó una biblioteca PHP de código abierto llamada Spreadsheet: WriteExcel desarrollada por Xavier Moguer [Mog01] y que se encuentra disponible para su uso bajo licencia GPL. Las versiones de las herramientas utilizadas para el desarrollo de la aplicación son las siguientes:

- Lenguaje de programación: PHP versión 5.1.4.
- Servidor de Base de Datos: MySQL 5.0.21
- Servidor Web: Apache 2.2.2

## *5.2 Formato de Especificación de Requisitos*

Los requisitos en general deben presentar una serie de características tanto individualmente como en grupo. Dentro de estas características se puede mencionar:

- *Necesario:* Un requisito es necesario si su omisión provoca una deficiencia en el sistema a construir.

- *Conciso:* Un requisito es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en el futuro.

- *Completo:* Un requisito está completo si no necesita ampliar detalles en su redacción, es decir, si se necesita la información suficiente para su comprensión.

- *Consistente:* Un requisito es consistente si no es contradictorio con otro requisito.

- *No ambiguo:* Un requisito no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

- *Verificable:* Un requisito es verificable cuando puede ser cuantificado de manera que permita hacer uso de algún método de verificación.

Para la especificación de los requisitos se tomó en cuenta que ésta debería ser lo más simple posible, con el fin de facilitar e incentivar el uso de la herramienta y evitar la inclusión de datos superfluos. Haciendo uso de las recomendaciones del estándar de ingeniería de software de la ESA (European Space Agency) [Esa91], del formato propuesto por Tom Gilb [Gil88] y de la experiencia de desarrolladores de software, se definió un conjunto de atributos comunes para los requisitos de usuario y de software. Aunque estos requisitos se especifican de la misma manera, ellos se administran en forma diferente. La planilla definida para especificar los requisitos contiene los siguientes atributos:

| Atributo | Descripción |
|---|---|
| Identificador | Este es un código único que sirve para identificar o reconocer el requisito. Para los requisitos de usuarios se utilizará el formato RUXXXX y para los de software RSXXXX |
| Nombre | Nombre en lenguaje normal del requisito |
| Descripción | Descripción del requisito. Que aspectos involucra, en qué consiste, etc. |
| Prioridad | Prioridad asociada al requisito, esta puede ser crítica, deseable o innecesaria. Un requisito es crítico si afecta una operación crítica del negocio. Si existe algún proceso que se quiera incluir para mejorar los procesos actuales, estamos ante un requisito deseable y si se trata de un requisito informativo o que puede esperar para fases posteriores, el requisito es catalogado como innecesario. |
| Fuente | Documento o persona desde la cual surgió el requisito. |
| Estabilidad | Este campo tiene como propósito señalar si el requisito puede o no puede estar sujeto a cambio durante el ciclo de vida del software (transable o intransable). El estándar de la ESA lo define como estable o no estable. |
| Tipo | A que tipo de requisito pertenece. Puede ser Funcional, Calidad, etc. |
| Estado | Estado actual del requisito dentro del desarrollo (Cumple, No Cumple, Ambiguo) |
| Listado de Usuarios | Son los tipos de usuarios que están asociados al requisito. |
| Caso de Prueba | Caso con el cual se probará si se cumple o no con el requisito en el sistema. |

El cambio principal entre la especificación de requisitos de usuario y de software radica en la asignación de la categoría a la que un requisito pertenece. En el caso de los requisitos de usuario, las categorías definidas son las siguientes:

- *Funcional:* Describe una capacidad que tiene que ser soportada por el software. Este tipo de requisito debe ser alcanzado sí o sí.

- *De Calidad:* Representan un atributo de calidad del producto, y se pueden expresar en una escala de medición. La calidad final del software dependerá del logro de estos objetivos.

- *De Restricción:* Indican con qué restricciones se desarrollará y/o funcionará el software. Esto es en función de costos, tiempos, personal, ambiente operacional, hardware, redes, etc.

De la misma manera, también se generó una clasificación para los requisitos de software. Las categorías definidas para los requisitos de software son las siguientes:

- *Funcionales:* Indican cuáles deben ser las capacidades del software. Se derivan del modelo lógico.

- *Interfaz:* Especifican el hardware, software o elementos de bases de datos con los que el sistema o sus componentes interactúan o se comunican.

- *Operacionales:* Especifican la forma en que correrá el sistema y como se comunicará con los operadores humanos. Incluyen todas las interfaces de usuario, interacción humano-computador, y requisitos logísticos y organizacionales.

- *Recursos (Ambiente Operacional):* Especifican los límites superiores de los recursos físicos tales como capacidad de procesamiento, memoria principal, espacio en disco, etc.

- *Usabilidad:* Estos son los relacionados con el esfuerzo de uso, y la evaluación del uso, realizada por los usuarios.

- *Mantenibilidad:* Requisitos relacionados con el esfuerzo de hacer modificaciones. Especifican cuan fácil es reparar fallas y adaptar el software a nuevos requisitos

- *Portabilidad:* Tiene que ver con la habilidad de ser transferido de un ambiente a otro.

- *Confiabilidad:* Son aquellos que están relacionados con la capacidad de mantener un nivel adecuado de servicio, bajo ciertas condiciones y por cierto tiempo. Especifican los tiempos medios entre fallas aceptables.

- *Interoperabilidad:* Habilidad de interactuar con determinados sistemas.

- *Rendimiento:* Establecen valores numéricos para variables medibles que guardan relación con el rendimiento del sistema.

- *Documentación:* Especifican requisitos particulares del proyecto para la documentación.

- *Escalabilidad:* Especifica la capacidad del sistema para mantener, si no mejorar, su rendimiento medio conforme aumenta el número de usuarios.

Las categorías recién mencionadas, tanto para los requisitos de usuario como para los de software son una simplificación de las categorías mencionadas por el estándar de ingeniería de software de la ESA (European Space Agency) [Esa91]. Esto con el fin de hacer a la aplicación lo más simple y amigable posible.

## *5.3 Proceso de especificación de Requisitos, Casos de Prueba e Indicadores*

El proceso de especificación de requisitos genérico en el cual está basado el diseño de esta herramienta consiste en tres grandes procesos: UR (Requisitos de Usuario), SR (Requisitos de Software) y AD en conjunto con DD (Diseño Arquitectónico y Diseño detallado y Producción). El primer proceso (UR) consiste en la definición de los requisitos de usuario, fase que tiene como producto final el URD o documento de requisitos de usuario. Luego de aprobada esta fase y teniendo como entrada el URD se procede a la especificación y análisis de requisitos de software (SR) el cual tiene como salida el documento de requisitos de software (SRD). En base al SRD se define el diseño arquitectónico del sistema a construir, el cual contempla los módulos que pueden ser ingresados en ReqAdmin. La revisión de que se cumplan los requisitos de usuario y de software se hace por medio de los casos de prueba que se pueden definir en la herramienta desarrollada. A continuación se muestra un diagrama del proceso anteriormente descrito.

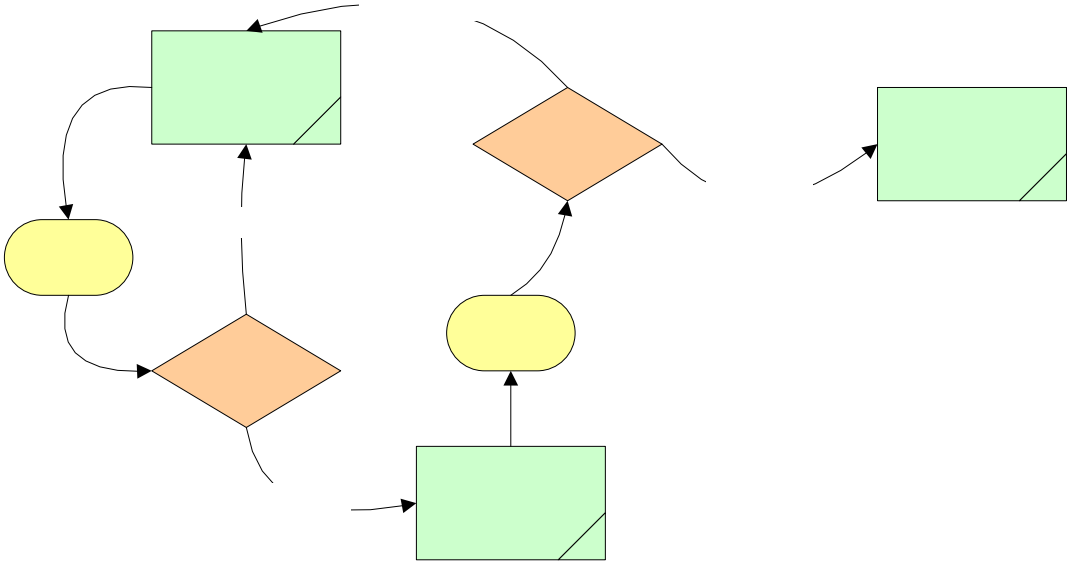

FIGURA 5.1. Diagrama del proceso de especificación de requisitos

Para ayudar con el proceso de especificación de requisitos, ReqAdmin cuenta con módulos de proyectos, tipos de usuario, requisitos de usuario, requisitos de software, casos de prueba y módulos. Cada uno de ellos está diseñado para ayudar en cada etapa del proceso de especificación de requisitos. Los pasos a seguir dentro de la herramienta para su utilización son los siguientes:

- Crear un nuevo proyecto e ingresar los distintos tipos de usuarios que usarán el software a desarrollar.

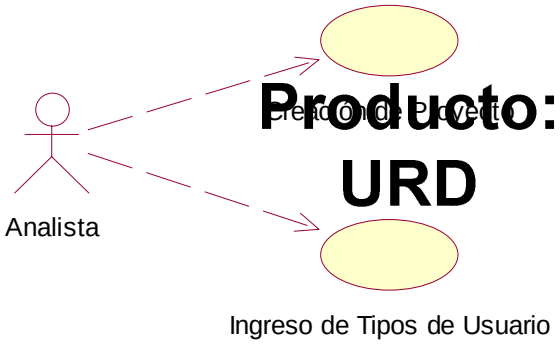


FIGURA 5.2. Ingreso de Proyectos y Tipos de Usuario

- Ingresar los distintos requisitos de usuario, vinculándolos con los tipos de usuario respectivos. Se debe ingresar la prioridad del requisito, su estabilidad, a que categoría corresponde y su estado que originalmente es “No Cumple”.

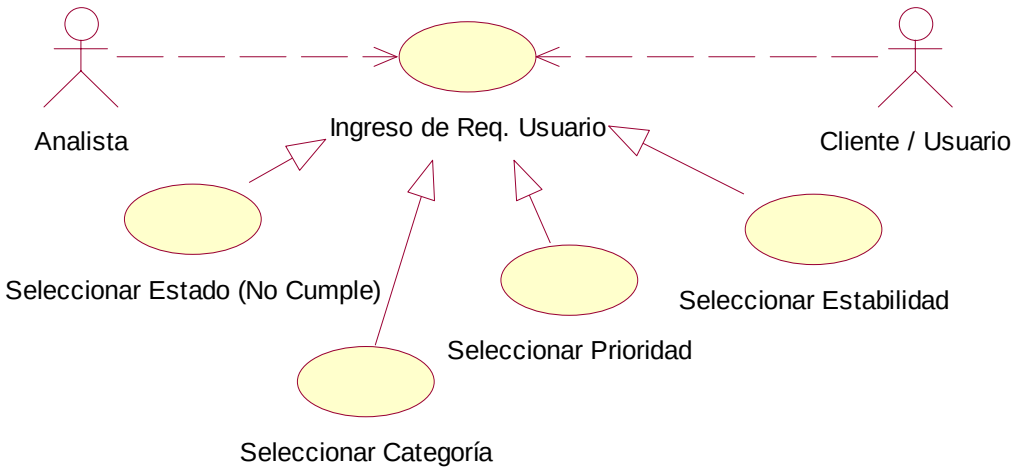


FIGURA 5.3. Ingreso de Requisitos de Usuario

- Luego de las revisiones correspondientes (UR/R) se procede al ingreso de requisitos de software al sistema. Básicamente se sigue el mismo procedimiento que con los requisitos de usuario pero estos deben ser vinculados a requisitos de software.

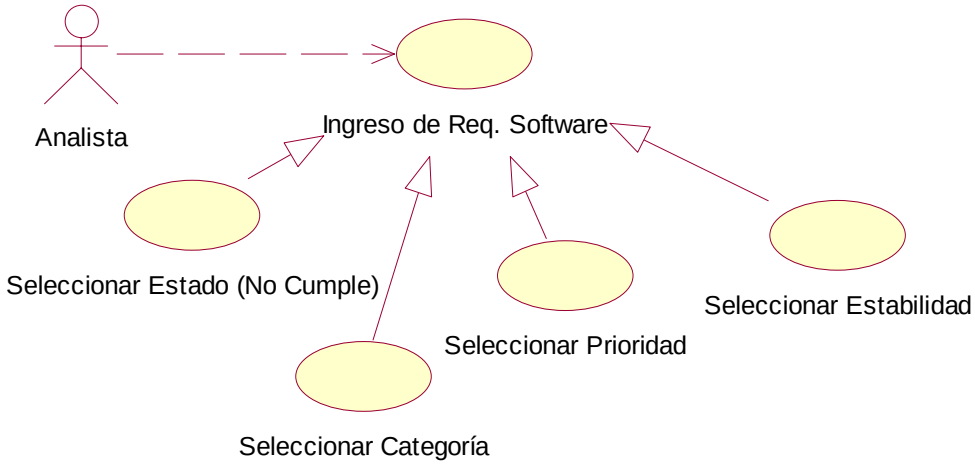


FIGURA 5.4. Ingreso de Requisitos de Software

- Se definen las pruebas de usuario con el cliente y se ingresan en el módulo de casos de prueba.

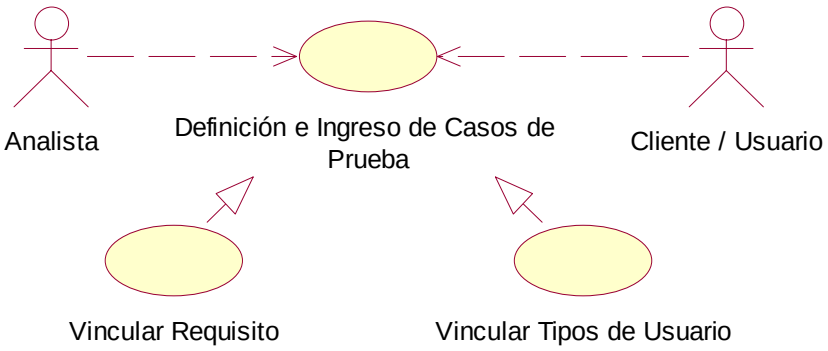


FIGURA 5.5. Ingreso de Casos de Prueba

- Finalizada esta primera etapa de requisitos, se procede con el diseño arquitectónico de la herramienta. ReqAdmin posee una sección de ingreso de módulos en el cual deben ingresarse los artefactos resultantes de la etapa de diseño. A estos artefactos se les debe vincular los requisitos de software respectivos.

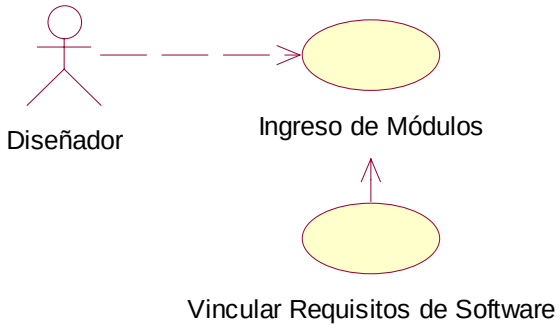


FIGURA 5.6. Ingreso de módulos

- Luego de lo anterior, se procede a iniciar la etapa de testing del software desarrollado hasta el momento, por medio de los casos de prueba ingresados al sistema anteriormente. Dependiendo de los resultados obtenidos, se deben modificar los estados de los requisitos involucrados (estos estados son “Cumple”, “No Cumple” y “Ambiguo”)

## *5.4 Diseño de la Herramienta*

La herramienta está pensada en un modelo de tres capas, es decir una primera capa de datos, luego una segunda capa, que contempla la lógica del negocio, hasta llegar finalmente a la capa de la interfaz o capa de presentación. Físicamente la herramienta está estructurada en directorios que contienen distintos scripts de PHP. Los directorios presentes son los siguientes:

- *Home:* Aquí se encuentran la página de inicio index.php y otros scripts que sirven para el ingreso. Por medio de estas se puede llegar al resto de las scripts PHP disponibles.

- *Frames:* Tal como el nombre del directorio lo indica, aquí se encuentran los scripts que sirven para la interfaz y para el funcionamiento de los frames que utiliza el sitio.

- *Includes:* Este directorio es fundamental en el funcionamiento de ReqAdmin. En él se encuentran las clases que interactúan con la base de datos, el archivo donde se definen las constantes de conexión al servidor, la librería que se utiliza para escribir archivos Excel, envío de e-mail y las clases que despliegan el código HTML.

- *Media:* Aquí se pueden encontrar javascripts que utiliza la herramienta, además de las imágenes e iconos que se visualizan en ella.

- *Processors:* En este directorio se encuentran los scripts que se encargan de procesar los datos y enviarlos a los scripts respectivos de la capa de datos.

Como se mencionó anteriormente el directorio includes posee clases que permiten el funcionamiento del sistema: entre ellas se encuentran MySQLDB, Session, Mailer y ErrorHandler. El propósito de la clase MySQLDB es simplificar la tarea de acceder a la información de la base de datos del sitio, todos los scripts de la herramienta acceden a la base de datos para insertar o modificar datos por medio de esta clase. La clase Session ayuda a manejar las acciones de la sesión del usuario en el navegador, además permite manejar el tracking de los usuarios conectados y de las visitas. Mailer que tiene como funcionalidad enviar e-mails a los usuarios registrados, esta clase trabaja bajo el supuesto de que esté instalado en el servidor, sendmail. ErrorHandler es una clase que esta pensada para que se encargue del manejo de los errores que puedan ocurrir durante la

interacción del usuario con la herramienta como por ejemplo error en el ingreso de algún campo u omisión de este.

## *5.5 Mapa del Sitio*

A continuación se muestra un mapa del sitio desarrollado. La estructura de navegación consta de ocho módulos, los cuales a su vez tienen subsecciones con funciones particulares cada una (Figura 5.7). Las secciones en las que está compuesta la herramienta son las siguientes:

- *Inicio*: En este módulo se despliega la página de bienvenida al sitio y las funciones de administración como editar los datos de la cuenta y para los usuarios con perfil de administrador de la herramienta, poder agregar nuevos usuarios o proyectos.

- *Proyectos:* Esta sección está orientada al manejo de los proyectos. Las funcionalidades que posee son la de ingresar un nuevo proyecto, editarlo o seleccionarlo. También existe la posibilidad de copiar el proyecto y ver cual ha sido la actividad de este.

- *T. Usuario:* Este módulo está desarrollado con scripts PHP que permiten la administración de los tipos de usuarios asociados a cada proyecto. Estos tipos de usuario se pueden tanto ingresar en el sistema como editar.

- *Req. Usuario:* En esta sección se encuentran los scripts orientados a la administración de requisitos de usuario. Permiten la visualización de estos, ingresar nuevos requisitos, asociarlos a tipos de usuario y editarlos.

- *Req. Software:* Los scripts que se utilizan en este módulo son similares a los de requisitos de usuario, permiten las mismas funcionalidades, la principal diferencia son los tipos definidos para uno y otro. Para los requisitos de usuario hay tres tipos distintos definidos y para los de software once.

- *Casos Prueba*: Este módulo está enfocado en los casos de prueba que se pueden asociar tanto a los requisitos de software como de usuario.

- Módulos: Esta sección está enfocada en la administración de los módulos o interfaces a desarrollar en el proyecto. Tiene las funcionalidades de crear, editarlos y asociarlos a los requisitos de software respectivos.

- *Indicadores:* Esta sección fue desarrollada para el despliegue de los indicadores de sanidad que posee la herramienta como las matrices de trazado, árbol de relaciones, estadísticas y vista rápida de requisitos. Permite descargar además todos los datos ingresados en los proyectos (requisitos de usuario, de software, casos de prueba y módulos) por medio de documentos en formato RTF (Rich Text Format). Las matrices de trazado se pueden descargar en formato de una planilla de cálculo excel.

En la figura 5.7 se pueden apreciar las subsecciones que están señaladas de color naranjo y las secciones principales de color celeste. La página de inicio está marcada de color amarillo.

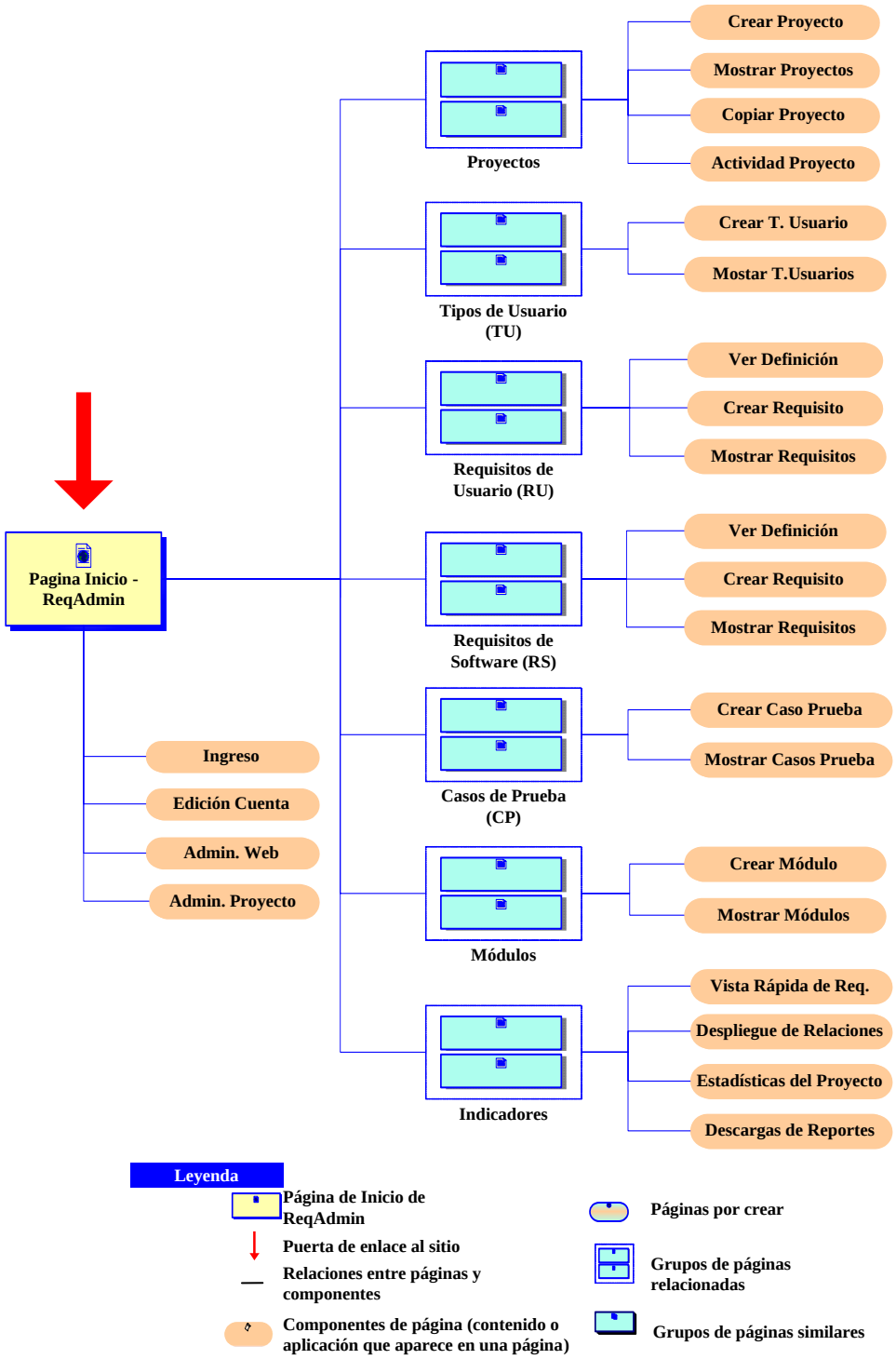


FIGURA 5.7. Mapa de la herramienta Web ReqAdmin

## *5.6 Interfaces de Usuario*

En la sección anterior se presentó un diagrama en el cual se puede visualizar la estructura de ReqAdmin. A continuación se muestran las interfaces de usuario de la herramienta desarrollada, la cual contempla una página principal de ingreso y otras ocho opciones que corresponden a: inicio, proyectos, tipos de usuario, requisitos de usuario, de software, casos de prueba, módulos e indicadores. A estas opciones se puede acceder por medio de una barra de navegación que provee la herramienta (Figura 5.8).


FIGURA 5.8. Barra de navegación superior de la herramienta

### 5.6.1 Página de inicio

Esta es la página de inicio del sistema. Permite entrar al sitio por medio del ingreso de un nombre de usuario y una contraseña.

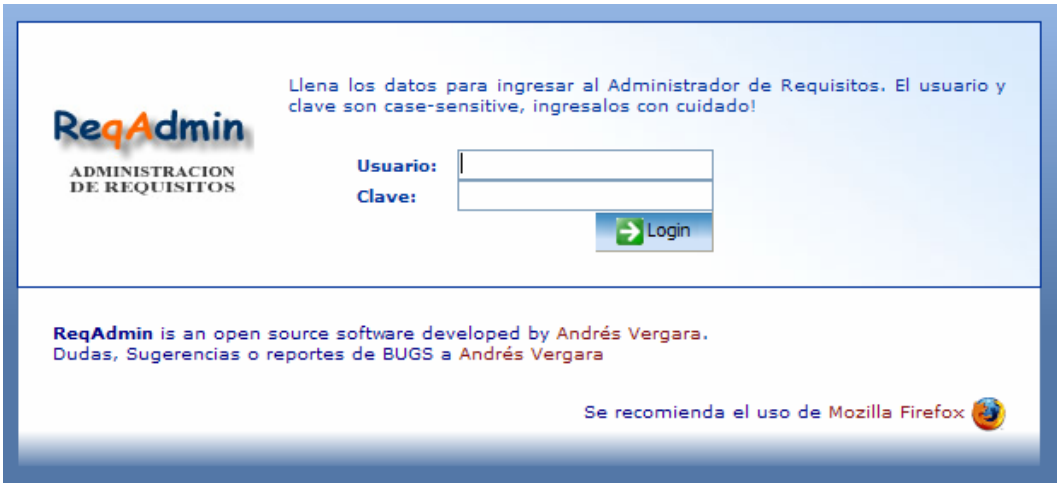


FIGURA 5.9. Formulario de ingreso al sistema

Una vez que un usuario ha ingresado al sistema se despliega la siguiente página (Figura 5.10.) en la cual se muestra un pequeño mensaje de bienvenida. En la barra de navegación se puede ver desplegadas las opciones para la página de inicio que son editar cuenta, administración web y administración de proyecto. En editar cuenta se permite cambiar la contraseña y e-mail de contacto. Los usuarios con privilegios de administrador pueden acceder a la sección de administración web donde se pueden ingresar nuevos usuarios y administrar los permisos, lo mismo sucede con administración de proyecto.


FIGURA 5.10. Página de Bienvenida al sistema

En la parte superior de la herramienta se puede observar el siguiente menú (Figura 5.11):

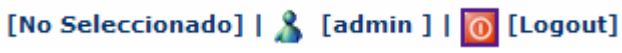


FIGURA 5.11. Menú superior

La primera frase entre paréntesis se refiere al proyecto actual del sistema que se encuentra seleccionado, lo segundo se refiere al usuario que ha ingresado a ReqAdmin, en este caso admin, y el tercero es la opción para que el usuario salga del sistema.

### 5.6.2 Requisitos de usuario

La fase de requisitos de usuarios es la que define el problema y es responsabilidad del usuario o cliente. Mediante esta etapa se refina la idea de la tarea a ser desarrollada. Cuando se hace click sobre el menú de requisitos de usuarios aparece un submenú con tres opciones: ver definición, crear requisito, mostrar requisitos. Cuando se accede a ver definición, lo primero que se puede apreciar es la definición de requisitos de usuario y más abajo la definición de cada una de la categorías definidas para ellos, que en este caso son Funcional, Calidad y Restricción. En la siguiente figura se muestra como se despliegan estas definiciones en el sistema.

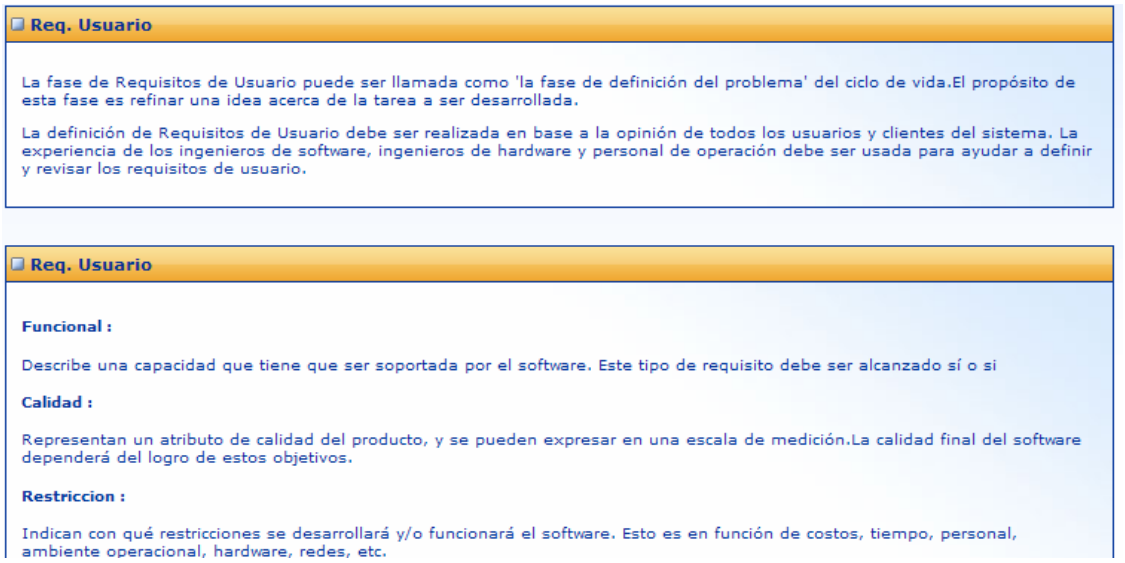


FIGURA 5.12. Despliegue de definiciones de requisitos de usuarios en la herramienta

La herramienta despliega los requisitos que están ingresados en el sistema hacia abajo en forma de un listado, con un filtro en el cual se pueden seleccionar los distintos tipos de requisitos de usuario, esto con el propósito de mejorar la visualización. Se puede ver la forma del despliegue en la figura 5.13.


FIGURA 5.13. Despliegue de requisitos de usuarios en la herramienta

Como se puede ver en la figura 5.13 se despliegan todos los datos del requisito y en la última columna existe la opción de editar el requisito y además aparece un icono amarillo con un signo de interrogación que significa que el sistema ha encendido una alarma con respecto a ese requisito. Cuando hacemos click sobre ese ícono, se abre el siguiente pop-up (Figura 5.14)

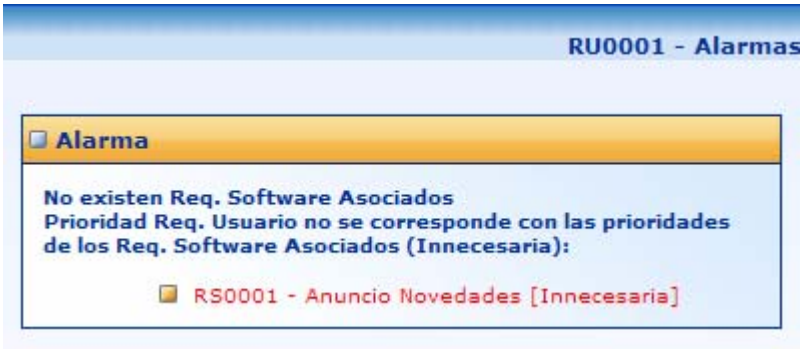


FIGURA 5.14. Ejemplo de despliegue de alarma.

Cuando se hace click en editar aparece lo siguiente (Figura 5.15):

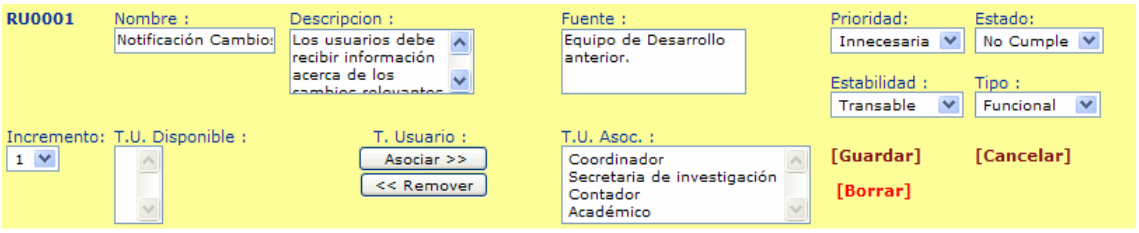


FIGURA 5.15. Edición de un requisito de usuario

En esta pantalla se pueden editar todos los campos del requisito de usuario seleccionado. Se puede agregar un nuevo tipo de usuario o por ejemplo cambiar su prioridad. Cuando queremos crear un

nuevo requisito, hacemos click sobre la opción respectiva del menú de navegación (Figura 5.10) y aparece el siguiente formulario:


FIGURA 5.16. Formulario de ingreso de un requisito de usuario

En el formulario (Figura 5.16) se pueden seleccionar el tipo de requisito, su estabilidad, prioridad y el estado actual. Además, se pueden asociar al requisito algún tipo de usuario. El sistema asigna automáticamente un identificador al requisito que se va a ingresar. En el campo nombre de este formulario de ingreso, se debe escribir un apodo en lenguaje natural del requisito y en la descripción una breve reseña de este. En el campo fuente, se propone que se añada el origen del requisito. Los campos estabilidad, tipo, prioridad, y estado son listas de selección y funcionan de la misma forma, esto quiere decir que se selecciona la opción deseada dentro de un listado. Dentro de las listas de selección, la estabilidad se refiere a si el requisito estará sujeto a cambio durante el ciclo de la vida del software (transable o intransable). En el campo tipo, se selecciona a que categoría corresponde el requisito y con esto, se permite su clasificación dentro del sistema. El campo prioridad sirve principalmente al desarrollador para armar su plan de trabajo, debido a que este campo representa la importancia del requisito para el cliente. Finalmente el campo estado, tiene como función especificar el status del requisito en el desarrollo actual. Inicialmente todos los requisitos parten con el estado “No Cumple” a menos que el usuario lo modifique. A medida que

avance el desarrollo, el estado puede cambiar a “Cumple” o “Ambiguo”, según lo dicten las pruebas realizadas. Hay que destacar que para facilitar la visualización se adoptó un código de colores para el estado de los requisitos de usuario. Para el estado “Cumple” se utiliza el verde, azul para “Ambiguo” y finalmente rojo para “No Cumple”. Además de los campos anteriores, hay un último elemento que hay que rellenar en el formulario para ingresar un requisito de usuario, el cual se refiere a los tipos de usuario. Este campo corresponde a listas de selección múltiple, mediante las cuales el analista debe seleccionar los tipos de usuarios que estarán vinculados al requisito. Los tipos de usuario son los distintos perfiles de usuario que van a utilizar el software o sistema a desarrollar (gerente, secretaria, etc.) y que por lo mismo, tienen distintos requisitos que generalmente se cruzan unos con otros y que en algunas situaciones pueden ser excluyentes. Generalmente no se hace un control o seguimiento muy fuerte de esta situación, pero con ReqAdmin se pretende controlar esta condición y así minimizar las fallas en la especificación de requisitos.

### 5.6.3 Requisitos de software

En la etapa de requisitos de software se propone analizar el estado de requisitos de usuario con el fin de lograr una especificación de lo que el software tiene que hacer lo más completa, correcta y consistente posible. La definición de los requisitos de software es responsabilidad del desarrollador. El propósito de este módulo de la herramienta es ayudar en la especificación de los requisitos de software. Al igual que en el modulo de requisitos de usuario, existen tres opciones; ver definición, crear requisito, mostrar requisitos. Básicamente todo este módulo funciona de la misma forma que el módulo de requisitos de usuarios, la única diferencia es que los tipos definidos son distintos y a los requisitos de software además de asociar tipos de usuario hay que asociar requisitos de usuario. Un ejemplo de cómo se visualizan los requisitos de software en la herramienta se puede ver en la figura 5.17.


FIGURA 5.17. Despliegue de requisitos de software en ReqAdmin

Como se puede observar en la figura la visualización es prácticamente igual a lo que ocurre con los requisitos de usuario, la única diferencia son los filtros de tipo de requisitos y que aquí además se asocian requisitos de usuario a los de software. Se continúa como se puede ver con el código de colores para ver el estado de los requisitos; verde si es que “Cumple”, rojo si es “No Cumple” y azul si es “Ambiguo”.

Para agregar un requisito de software se accede al siguiente formulario (Figura 5.18) En él se pueden seleccionar el tipo de requisito, su estabilidad, prioridad y el estado actual. Además se pueden asociar al requisito algún tipo de usuario y los requisitos de usuarios asociados. Para la edición de algún requisito se accede a un formulario similar al visto en la sección anterior de requisitos de usuario (Figura 5.15)

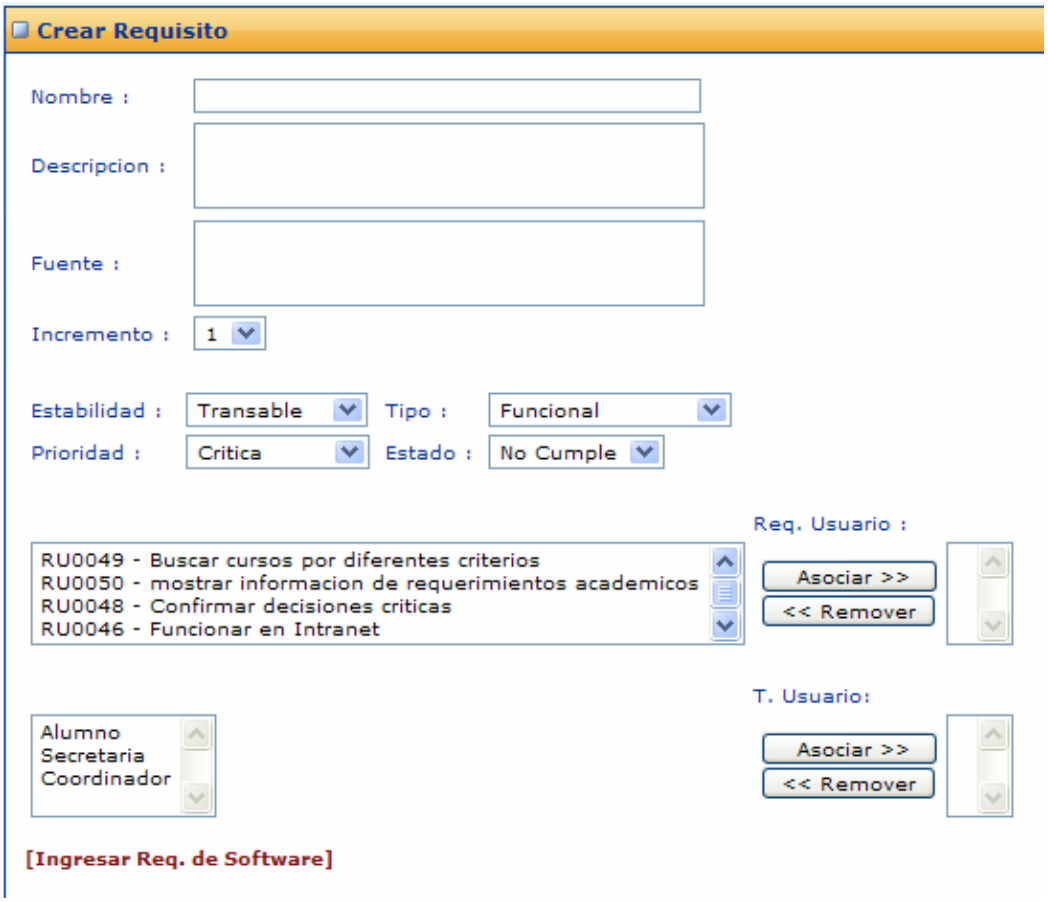


FIGURA 5.18. Formulario de ingreso de un requisito de software.

### 5.6.4 Casos de prueba

Esta sección está destinada a especificar lo que son las pruebas para detectar errores en el sistema. Estas deberían ser llevadas por personas distintas a los diseñadores de los programas, tanto para evitar una simple verificación de que el programa funcione correctamente, como para probar que ese programa ha sido concebido e interpretado de forma correcta. Los casos de prueba deben ser escritos tanto para condiciones de entrada inválidas o inesperadas, como para condiciones válidas y esperadas.

En el sistema, los casos de prueba se despliegan como lo muestra la figura 5.19. Estos se pueden asociar tanto a requisitos de software como de usuarios. Además de lo anterior también se puede asociar al tipo de usuario que involucra el caso en cuestión.


FIGURA 5.19. Despliegue de Casos de Prueba en ReqAdmin

Cuando se desea ingresar un nuevo caso de prueba se llega a un formulario como el de la figura 5.20. En éste se selecciona el tipo de requisito que se desea vincular (usuario o software) y luego se selecciona el requisito respectivo para asociarlo al caso de prueba. El tipo de usuario se vincula por medio de listas de selección múltiple. El tipo de usuario vinculado puede ser más de uno. Este tipo de usuario indica quién deberá realizar la prueba especificada. Estos tipos de prueba son conocidos como “pruebas de usuario” ya que se enfocan más a cumplir 100% los requisitos definidos por el usuario en etapas anteriores que si en algún caso no se llegasen a cumplir a cabalidad, deben ser refinados y finalmente aprobados por el usuario final.

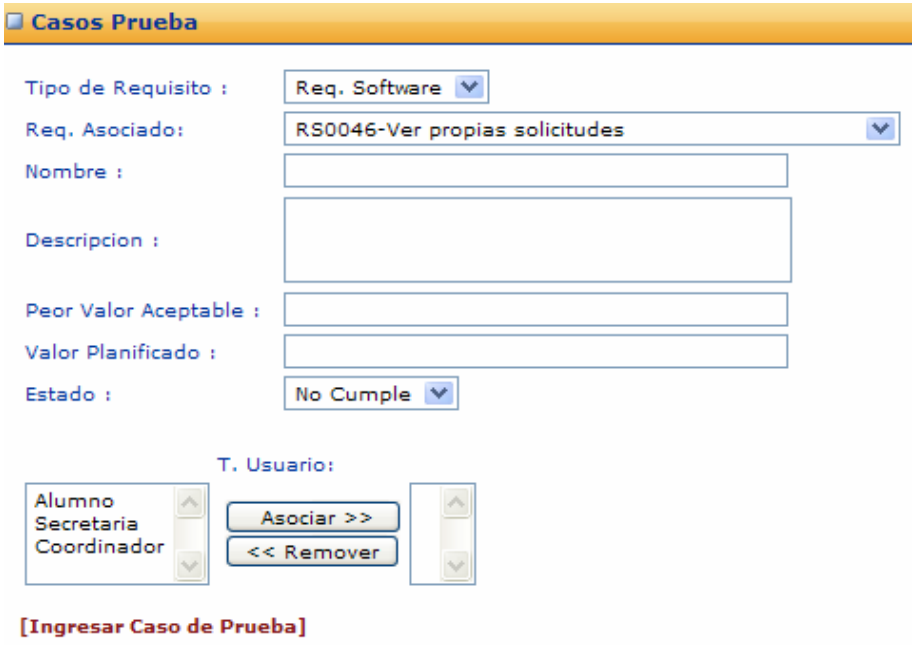


FIGURA 5.20. Formulario de Ingreso de un Caso de Prueba en ReqAmin

### 5.6.5 Módulos

Esta sección esta diseñada para ayudar en las especificaciones de los módulos del sistema a desarrollar. Con módulos nos referimos a componentes de alto nivel que encapsulan sub-elementos, en este caso esos sub-elementos serían los requisitos de software. Los módulos en la práctica resultan ser interfaces, páginas web o scripts. La herramienta despliega los módulos como se puede ver en la figura 5.21. Se utiliza al igual que en las secciones anteriores el código de colores para visualizar el estado de los requisitos; verde es “Cumple”, rojo es “No Cumple” y azul “Ambiguo”.

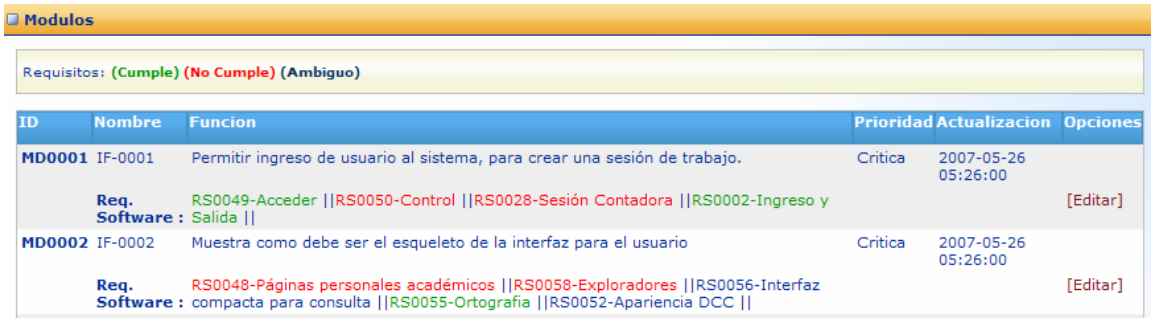


FIGURA 5.21. Despliegue de Módulos en ReqAdmin

Para agregar un módulo a la herramienta se accede por medio del menú de navegación a un formulario como el de la figura 5.22. En este formulario se ingresan el nombre del módulo, su función, se selecciona su prioridad y se le agregan los requisitos de software que están asociados al módulo.

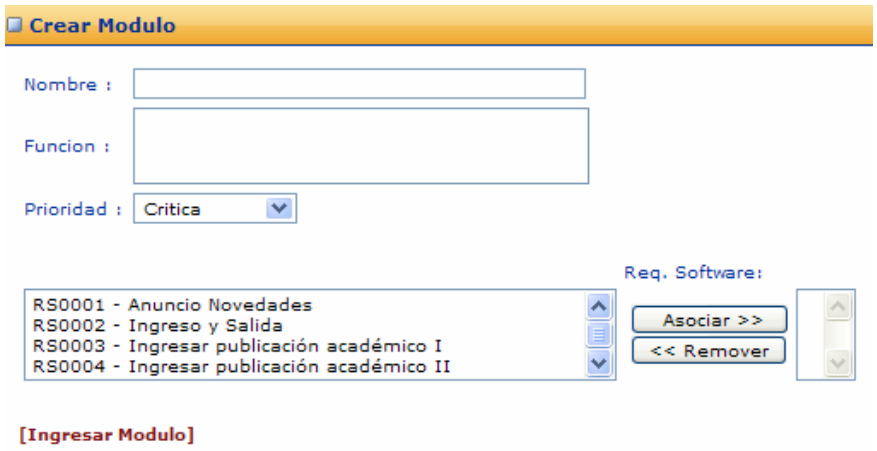


FIGURA 5.22. Formulario de ingreso de módulos en ReqAdmin

### 5.6.6 Tipos de usuario

Esta sección está destinada  a ingresar a los tipos de usuario que están involucrados en cada proyecto. Esto se realiza para que luego sean vinculados con cada requisito y de esta forma poder saber que requisito está relacionado con que tipo de usuario y con esto hacer un mejor control de los requisitos. En el sistema los tipos de usuario se despliegan como lo muestra la figura 5.23.

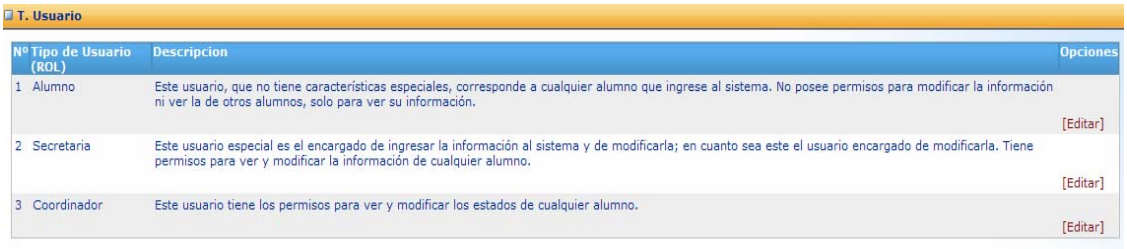


FIGURA 5.23. Despliegue de tipos de usuario

### 5.6.7 Indicadores

Esta sección de la herramienta desarrollada tiene como propósito desplegar indicadores del proyecto que permitan mejorar la calidad de los productos desarrollados en la fase de análisis y de forma tal que los riesgos puedan ser identificados en forma temprana y así poder corregir situaciones no deseadas durante el proyecto. En este módulo se pueden acceder a las siguientes cuatro secciones; descarga de reportes, estadística de proyecto, despliegue de relaciones y vista rápida de requisitos.

La generación de matrices de trazado que realiza la herramienta lo hace por medio de hojas de cálculo Excel (Figura 5.24). Se pueden seleccionar de diversos tipos (requisitos de usuario versus requisitos de software, artefactos de diseño versus requisitos de software, etc.). Además, existe una opción que permite descargar una matriz que agrupa todas las matrices en un documento Excel.

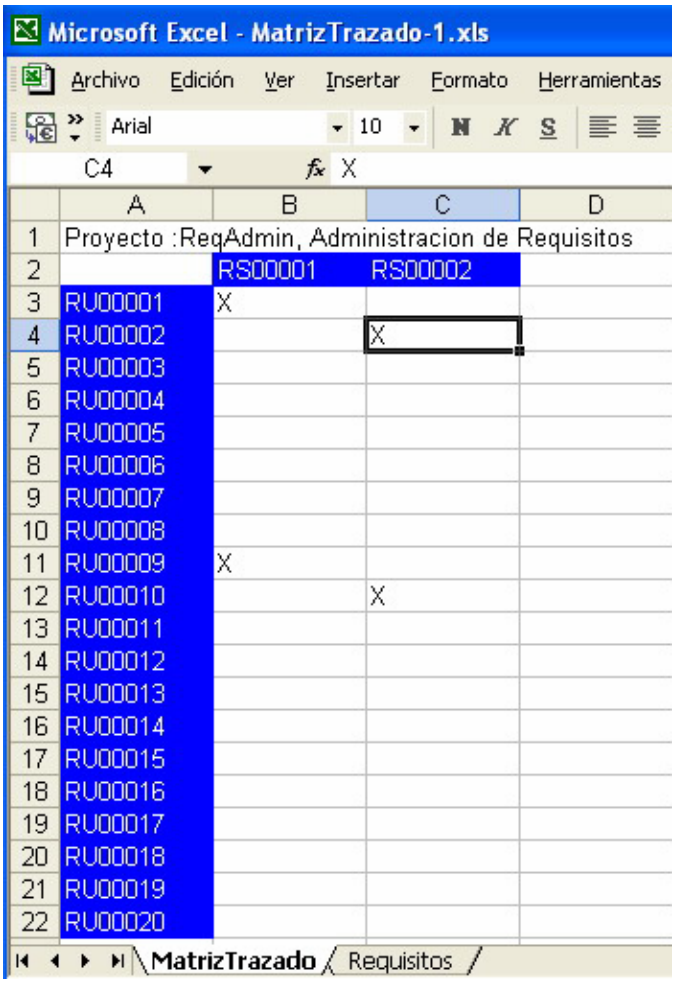


FIGURA 5.24. Matriz de trazado (requisitos de usuario versus requisitos de software)

También en la sección de indicadores existe un reporte de requisitos el cual sirve para desplegar tanto requisitos de usuario como requisitos de software. Se pueden seleccionar la prioridad de los requisitos que se quieren visualizar, así como su estado o el tipo de usuario al que esté asociado (Figura 5.25). El propósito de este reporte es mejorar la visualización de los requisitos.

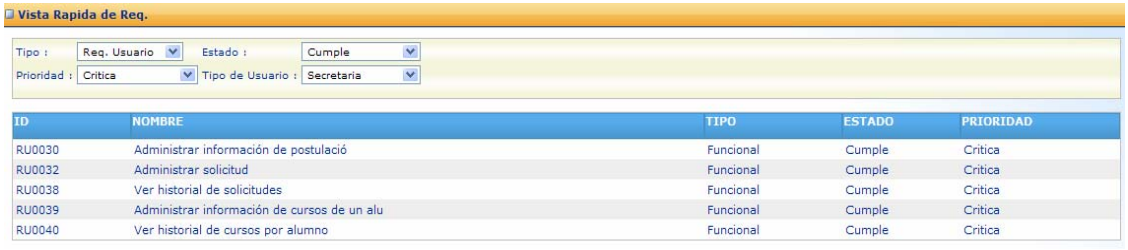


FIGURA 5.24.Reporte de Requisitos

Otra funcionalidad que se ha agregado a la herramienta es el despliegue de un árbol de relaciones. Las vistas disponibles para este árbol de relaciones son requisitos de usuario vs. requisitos de software, requisitos de software vs. requisitos de usuario y casos de prueba vs. requisitos. Gracias a esta nueva característica de la herramienta, es fácil identificar situaciones anómalas como por ejemplo un requisito de software cuyo estado sea “Cumple” (en la herramienta representado por un icono verde) asociado con un requisito de usuario con el estado “No Cumple” (ícono rojo) tal como se puede ver en la figura 5.25 en el requisito de software RS0012.


FIGURA 5.25. Árbol de relaciones en ReqAdmin

Además de las funcionalidades ya descritas, esta sección también es capaz de desplegar algunas estadísticas del proyecto como por ejemplo la cantidad de requisitos críticos no cumplidos, cuantos no tienen ningún tipo de usuario asociado, la cantidad de requisitos ambiguos, etc. Un ejemplo de cómo la herramienta despliega estos indicadores es la figura 5.26


FIGURA 5.26. Despliegue de estadísticas de proyecto

## 5.6.8 Proyectos

En esta sección están todos los proyectos que se encuentran disponibles en el sistema. Al igual que en el resto de los módulos anteriores, la sección de proyectos está disponible un menú con opciones, que en esta oportunidad permite ver los proyecto, ingresar nuevos proyectos, copiar un proyecto y ver su actividad. Los proyectos en la herramienta son desplegados tal como lo muestra la figura 5.27.

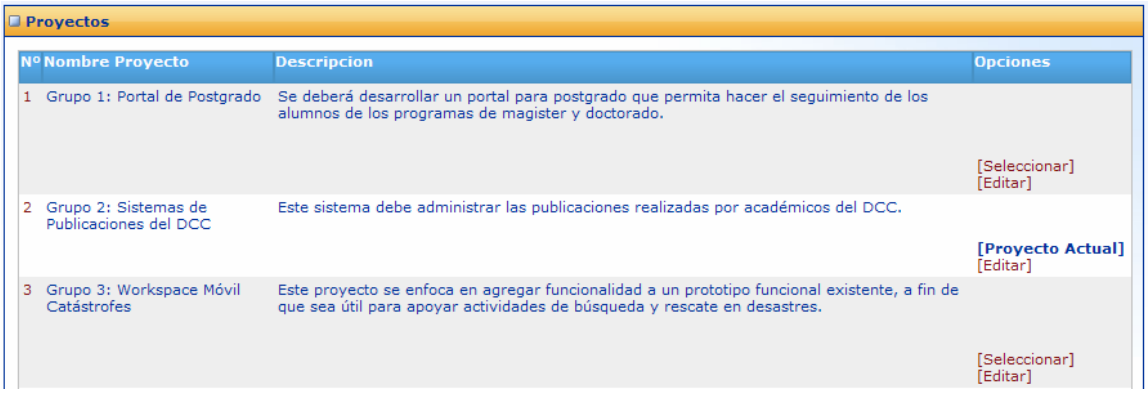


FIGURA 5.27. Despliegue de proyectos

El proyecto actualmente seleccionado aparece con la leyenda “Proyecto Actual”, además también está la opción de seleccionar el proyecto o editarlo. En el caso que se necesite ingresar un nuevo proyecto se accederá a un formulario similar al de la siguiente figura.

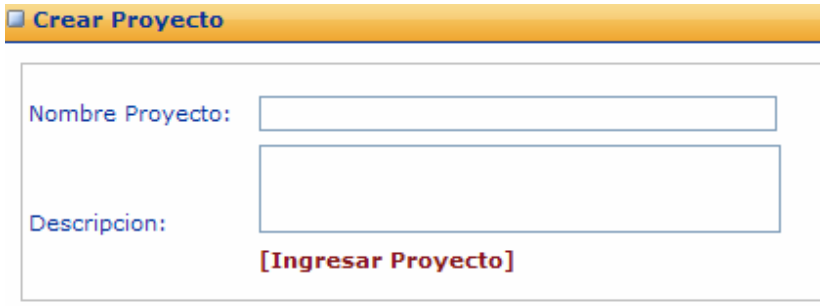


FIGURA 5.28. Formulario de ingreso de proyectos

Como se mencionó anteriormente existen otras dos funcionalidades que son copiar proyecto y ver actividad del proyecto. La primera de estas funcionalidades es básicamente hacer una copia del proyecto seleccionado en el estado actual. Esto sirve para guardar hitos y poder hacer comparaciones a través del tiempo. La segunda funcionalidad permite al usuario saber que elementos del proyecto han sido actualizados y por quien (por elementos nos referimos a requisitos de usuario, requisitos de software, casos de prueba o módulos).

# 6. Resultados obtenidos

A continuación se detalla cómo fueron validadas las hipótesis, los resultados obtenidos y cuáles son los aportes o contribuciones que hace esta herramienta, tanto a la ingeniería de requisitos como a la ingeniería de software en general.

## *6.1 Validación de la hipótesis*

Se utilizó el curso CC51A, Ingeniería de Software, como fuente de validación y experimentación durante el desarrollo de este trabajo. La herramienta desarrollada se utilizó en el curso como apoyo al desarrollo de los proyectos del semestre. La retroalimentación obtenida sirvió para hacer los ajustes respectivos a la herramienta.

Se validó preliminarmente la hipótesis H1 dando acceso sólo a las métricas de la herramienta. Sin embargo, sólo a aquellos que ejercieron el rol de “téster”, se les permitió acceder al módulo de trazabilidad de los requisitos. De esta forma los téster fueron capaces de detectar en forma más temprana, mayor número de inconsistencias que los analistas.

La hipótesis H2 se validó realizando inspecciones a los distintos proyectos durante su desarrollo. Previo a cada inspección se le solicitó al administrador del proyecto y a los analistas que hicieran un diagnóstico respecto al estado del proyecto, en términos de avance, trabajo pendiente, pérdida/cambio de requisitos, requisitos prioritarios y posibles riesgos derivados de los requisitos o de su administración. Los analistas utilizaron las métricas como apoyo para realizar su diagnostico; mientras que el administrador del proyecto podía utilizar cualquier otra herramienta que no fuera parte del sistema propuesto. Luego de que ambos entregaran su propuesta, se realizó la inspección al proyecto encontrándose en general que los diagnósticos de los analistas estuvieron más precisos que los administradores de proyecto.

Este trabajo contempló sólo la validación preliminar de las hipótesis, ya que un proceso de validación más contundente y definitivo requeriría de un enorme esfuerzo de experimentación.

## *6.2 Contribuciones de la ingeniería de requisitos*

La ingeniería de requisitos está enfocada en entregar soluciones flexibles y de bajo costo para especificar y administrar requisitos durante el desarrollo de un producto de software. Uno de sus objetivos principales consiste en la generación de especificaciones correctas que describan con claridad, en forma consistente y compacta, el comportamiento del sistema. Aunque esta tarea no es fácil, la ingeniería de requisitos ha mostrado que el esfuerzo vale la pena. Algunos de los beneficios más importantes son los siguientes:

- Permite gestionar las necesidades del proyecto en forma estructurada. Cada actividad de la ingeniería de requisitos consiste en una serie de pasos organizados y bien definidos.

- Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados. La ingeniería de requisitos proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempos y recursos necesarios.

- Ayuda a disminuir los costos y retrasos del proyecto. Esto se debe a que los objetivos están claros desde el inicio. En caso de cambios en los requisitos, el cliente está obligado a renegociar costos y tiempos. De esa manera se evita caer en situaciones de “all-inclusive”.

- Mejora la calidad del software. La calidad en el software tiene que ver con cumplir un conjunto más o menos largo de requisitos (funcionales, de calidad y de restricción) más o menos complejos. A partir de la especificación de requisitos se puede converger en forma temprana a especificar las pruebas que se le realizarán al sistema, y a validar las características generales del producto en desarrollo.

- Mejora la comunicación entre miembros del equipo. La especificación de requisitos representa una forma de consenso entre clientes y desarrolladores. Si este acuerdo no ocurre, el proyecto no será exitoso.

- Reduce la probabilidad de rechazo de productos por parte de los usuarios finales. La ingeniería de requisitos obliga al cliente/usuario a considerar y revisar los requisitos del producto en desarrollo en forma periódica. Por lo tanto, los clientes/usuarios son parte creadora del resultado final, y por ende se reduce la probabilidad de rechazo del producto.

Por lo antes mencionado, se asume que la automatización de los mecanismos de trazabilidad y la creación de métricas podría ayudar a reducir los costos y tiempos de desarrollo, y a disminuir los riesgos asociados al incumplimiento de requisitos. Esto se traduciría en productos más baratos y confiables. Por esa razón, esta tesis se enfocó en desarrollar una herramienta de administración de requisitos, que fuera capaz de proveer métricas que indican el nivel de sanidad de los requisitos. Se estima que el uso de esta herramienta ayuda a mejorar los productos obtenidos en la etapa de análisis, reduce el esfuerzo involucrado en el desarrollo de éstos, mejora la visibilidad de la fase de análisis sobre el resto del proyecto de desarrollo y permite la trazabilidad de los requisitos. Además permite el acceso distribuido a un repositorio compartido de requisitos.

# 7. Conclusiones y trabajo a futuro

Durante esta tesis se desarrolló una herramienta que permite especificar los requisitos y las relaciones existentes entre ellos siguiendo un formato genérico. Esta herramienta permite la clasificación de los requisitos y ayuda con lo que respecta a la validación, control y trazabilidad de los requisitos de un sistema facilitando su gestión.

La aplicación es capaz de generar indicadores que permiten visualizar el nivel de sanidad de los requisitos, de forma tal que los riesgos pueden ser identificados en etapas más tempranas, lo que permitirá tomar decisiones a tiempo para corregir situaciones no deseadas, reduciendo de esta forma el esfuerzo requerido para la gestión del proyecto. Las herramientas de trazabilidad incluidas en la herramienta ayudan a detectar de forma visual las incongruencias que se pueden encontrar en las especificaciones por medio de un código de colores. Además de lo anterior, la herramienta es capaz de generar ciertas alarmas en el caso de que un requisito cumple ciertas condiciones que llevan a la sospecha de que puede existir un error en su especificación.

Las principales características de la herramienta desarrollada son las siguientes:

- *Fácil de instalar:* Este objetivo se logró satisfactoriamente, considerando que los usuarios no deben instalar nada especial, pues la aplicación se utiliza desde un navegador. Sin embargo la instalación a nivel servidor puede resultar un poco más complicada.

- *Fácil de usar:* Sería necesario un estudio más riguroso para poder afirmar este punto con fuerza. Sin embargo es posible conjeturar que la herramienta puede usarse sin mayores dificultades, debido a la simplicidad de sus interfaces.

- *Permite la operación distribuida:* Su diseño cliente-servidor permite que la información sea ingresada y visualizada de forma distribuida. Los datos permanecen centralizados y visibles para todo el grupo de trabajo.

- *Auto contenida:* El usuario no debe tener más que un navegador para utilizar todas las potencialidades de la herramienta. Sin embargo, el lado servidor debe contar con varios

servicios como un servidor Web, una base de datos y otros programas y bibliotecas instaladas. Es importante notar que la mayoría de estos servicios son de uso común y no deberían imponer dificultades especiales al habilitarlos.

- *Multiplataforma:* El usar una interfaz Web, hace que el uso de la herramienta sea independiente de la plataforma. En el lado servidor, los servicios y programas necesarios están disponibles para las plataformas más populares. Por ejemplo bajo Windows se puede instalar XAMPP, aplicación que contiene el servidor web Apache y PHP incorporado. En el ambiente Linux, la mayoría de las distribuciones vienen con el servidor Apache y PHP incorporados, como por ejemplo el caso de Ubuntu.

- *Extensible:* Posee amplias posibilidades de extensión.

- *Facilita la trazabilidad:* A esta herramienta se le agregaron ayudas gráficas que ayudan a seguir la trazabilidad del proyecto y a detectar posibles anomalías en la especificación.

Entre las tareas pendientes y los desafíos inmediatos que surgen de esta memoria están las siguientes:

- Agregar a la herramienta el soporte multilenguaje.

- Generar una versión de la herramienta que sea capaz de funcionar sobre PDAs.

- Permitir que la descripción de requisitos pueda embeber imágenes, además de texto.

- Agregar la funcionalidad de exportar e importar las especificaciones en un tipo de archivo propio de la herramienta.

# 8. Bibliografía y referencias


[Bec00] K. Beck. “Extreme Programming Explained: Embrace Change”. Addison-Wesley, 2000.

[Bor03] Borland CaliberRM. Borland USA. URL: http://borland.com/caliber/index.html. Última Visita: Marzo, 2007.

[Bro00] A. Brown. Large-Scale Component-Based Development. Object and Component Technology Series. Prentice Hall. 2000.

[Cmm07] CMMI Model Terminology, http://www.sei.cmu.edu/cmmi/faq/termfaq.

html#Q318, Frequently Asked Questions (FAQs), última visita: Mayo de 2007.

[Esa91] European Space Agency. ESA Software Engineering Standards. PSS-05-0 Issue 2. ESA Board for Software Standardization and Control (BSSC). February, 1991. URL: http://www.ess.co.at/ECOSIM/ESA.txt

[Edw91] M. Edwards and S. Howell, A Methodology for System Requirements Specification and Traceability for Large Real-Time Complex Systems, technical report, U.S. Naval Surface Warfare Center Dahlgren Division, Dahlgren, Va., 1991.

[Fow99] M. Fowler and K. Scott. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 2nd Edition. Object Technology Series, Addison-Wesley. 1999.

[Gil88] Gilb, T., Gilb, T. y Finzi, S. Principles of Software Engineering Management. Addison-Wesley. 1988.

[God04] Goda Software. Analyst Pro. http://www.analysttool.com/products.html.Última Visita: Marzo, 2007.

[Hof01] H. F. Hofmann and F. Lehner. Requirements engineering as a success factor in software projects. IEEE Software, July/August, 2001, 58-66.

[Hoo00] I. Hooks. Requirements Engineering: Is it ‘Mission Impossible’? Requirements Engineering. 5(3), 2000, 194-197.

[Ibm04] IBM Rational Rose Requisite Pro.

URL:http://www-306.ibm.com/software/awdtools/reqpro.Última Visita:

Marzo, 2007.

[Ibm05] Supporting Iterative Development Through Requirements Management.

URL: http://www-106.ibm.com/developerworks/rational/library/2830.html. Última Visita : Junio, 2006.

[Ics05] IEEE Computer Science, Requirement Definition, IEEE Std 610.12-1990. URL:

http://www.computer.org/portal/site/ieeecs/menuitem.c5efb9b8ade9096b8a9ca0108bcd45f3/index.jsp?&pName=ieeecs_level1&path=ieeecs/education/certification/guide&file=RequirementDefinition.xml&xsl=generic.xsl.

Última Visita: Junio, 2006.

[Itm05] Information Technology Management Interfacing Requirements Management Tools in the Requirement Management Process – A First Look. URL : http://www.itmweb.com/essay544.htm. Última Visita: Junio, 2006

[Jac92] Ivar Jacobson y otros. Object Oriented Software Engineering. A Use Case Driven Approach. Addison Wesley, 1992.

[Kru00] P.Kruchten. The Rational Unified Process, An Introduction. 2nd Edition. Addison-Wesley. March, 2000.

[Mog01] Freshmeat.net . PHP Spreadsheet_WriteExcel Project , Xavier Moguer

URL:http://www.freshmeat.net/projects/spreadsheet_writeexcel/. Última Visita: Abril 2007

[Pal97] J.D. Palmer, Traceability, Software Requirements Eng., R.H. Thayer and M. Dorfman, eds., pp. 364-374, 1997.

[Pre00] R.S. Pressman. Software Engineering: A Practitioner's Approach. 5 Edition. McGraw Hill. 2000.

[Ram01] B. Ramesh and M. Jarke. Toward reference models for requirements traceability.

Software Engineering, IEEE Transactions on, 27(1):5893, Jan 2001.

[Roe91] W.H. Roetzheim. Developing Software to Government Standards. New York: Prentice Hall, 1991.

[Som97] I. Sommerville and P. Sawyer. Requirements Engineering. A Good Practice Guide.John Wiley and sons, 1997.

[Som04] Ian. Sommerville. "Ingeniería del Software". Adisson-Wesley, Mayo 2004.

[Spc07] ISO/IEC 1554, SPICE (Software Process Improvement and Capability dEtermination). http://www.sqi.gu.edu.au/spice/ Última Visita: Mayo, 2007

[Sty97] A. C. Stylianou, R. L. Kumar, M.J. Khouja. A Total Quality Management-Based Systems Development Process. The DATA BASE for Advances in Information Systems. (28)3. 1997.

[SST05] Six Sixma Tutorial, What is CMM (Sei Capability Model)

URL: http://sixsigmatutorial.com/CMM/CMM-Intoduction.aspx. Última Visita: Junio, 2006.

[Tha99] R. H. Thayer and M. Dorfman. Software Requirements Engineering. Second Edition. IEEE Computer Society Press and John Wiley & Sons, Inc. 1999.

[USD88] U.S. Departament of Defense. Military Standard 2167A Defense System Software Development, Washington, D.C, 1988.

[Wie03] K. Wiegers. Software Requirements. Second Edition. Microsoft Press, 2003.

[Wie99] K. Wiegers. Automating Requirements Management. Software Development, (7) July, 1999.

[Wkp05] Wikipedia. The Rational Unified Process.

URL: http://en.wikipedia.org/wiki/Rational_Unified_Process. Última Visita: Junio, 2006.

[Wkp07] Wikipedia. Capability Maturity Model Integrator.

URL: http://en.wikipedia.org/wiki/CMMI. Última Visita: Mayo, 2007

# A. Anexos:

## *A.1 Difusión de la herramienta*

Para difundir esta herramienta se creó un proyecto dentro de ChileForge, el cual es un repositorio de proyectos impulsados por CSoL (Centro de Software Libre) que busca aportar a la difusión y desarrollo del software libre en Chile, bajo el modelo Open Source. Se decidió inscribir el proyecto en su primera versión en este repositorio por varias razones, entre las cuales se pueden mencionar el de darle un perfil local al proyecto, resolver las necesidades de los desarrolladores de software a nivel ciudad, país y/o latinoamericano, los cuales en general disponen de menos recursos que desarrolladores norteamericanos y europeos. El proyecto se encuentra alojado en la siguiente URL:[http://chileforge.cl/projects/reqadmin](http://chileforge.cl/projects/reqadmin)


FIGURA A.1. ReqAdmin en ChileForge

## *A.2 Modelo de datos*

El modelo de datos que se muestra a continuación es una muestra parcial del modelo de datos completo de la aplicación y contempla mostrar todas las entidades relacionadas con el manejo de requisitos.

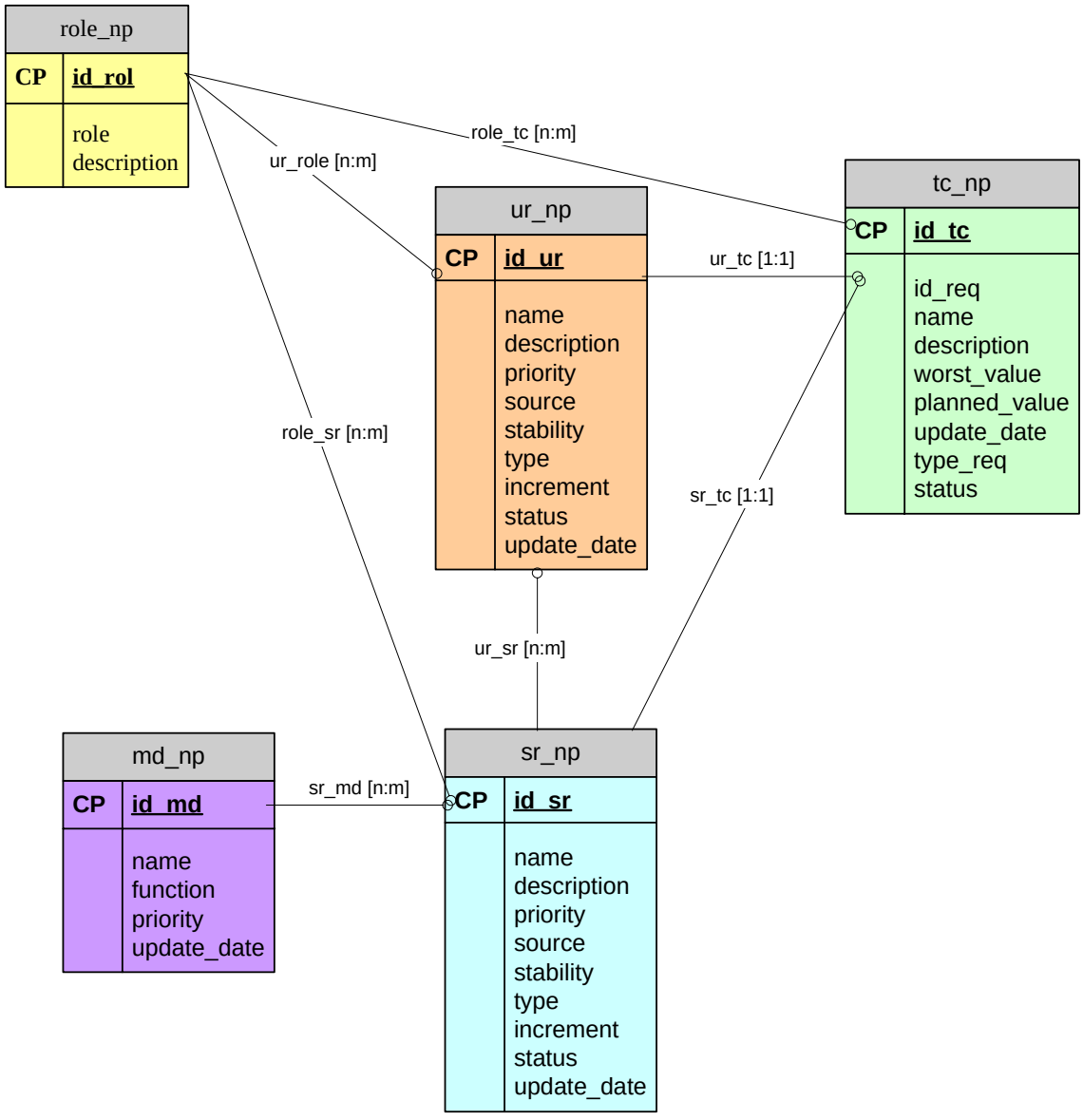


FIGURA A.2. Modelo de datos de la herramienta

El diccionario de datos asociado a las entidades mostradas en la *Figura A.2*, se describe a continuación.

### A.2.1. Entidades

**Tc_np:** En esta tabla se encuentran los casos de prueba. Estos se vinculan a los tipos de usuarios o roles. Para cada proyecto que se ingrese a la herramienta se crea la tabla de casos de pruebas respectiva.

| Campo | Tipo | Observación |
|---|---|---|
| Id_tc | Integer | PRIMARY_KEY, identificador del caso de prueba. |
| Id_req | Varchar(10) | Identificador del requisito asociado, puede ser tanto de usuario como de software. |
| Name | Varchar(50) | Nombre en lenguaje natural del caso de prueba. |
| Description | Text | Descripción del caso de prueba. En qué consiste el test que debe cumplir el software a desarrollar |
| Worst_value | Varchar(50) | Mínimo valor aceptable. |
| Planned_value | Varchar(50) | Valor que se planifica se logre al final del desarrollo. |
| Update_date | Datetime | Fecha de la última actualización del caso de prueba. |
| Type_req | Char(2) | Tipo de requisito (uso interno) |
| Status | Varchar(10) | Estado del caso de prueba: Cumple, No Cumple, Ambiguo |

**Md_np:** En esta tabla se encuentran los módulos. Estos se vinculan a los requisitos de software. Al igual que las otras tablas del modelo de la figura A.2. se crea una tabla de este tipo para cada proyecto.

| Campo | Tipo | Observación |
|---|---|---|
| Id_md | Integer | PRIMARY_KEY, identificador del módulo |
| Name | Varchar(50) | Nombre en lenguaje natural del módulo |
| Description | Text | Descripción del módulo |
| Priority | Varchar(12) | Prioridad del módulo. |
| Update_date | Datetime | Fecha de la última actualización del caso de prueba. |

**Role_np:** Esta tabla almacena los tipos de usuarios que van a estar asociados a algún tipo de requisito funcional dentro del proyecto de desarrollo de software.

| Campo | Tipo | Observación |
|---|---|---|
| Id_role | Int(11) | PRIMARY_KEY, Identificador del tipo de usuario |
| Role | Varchar (50) | Nombre del tipo de usuario |
| Description | Text | Descripción de la labor del rol de usuario |

**RS_np:** En esta tabla se encuentran los requisitos de software del sistema. Estos están vinculados a los requisitos de software. Este vínculo será visible en la matriz de trazado correspondiente.

| Campo | Tipo | Observación |
|---|---|---|
| Id_rs | int(11) | PRIMARY_KEY, identificador del requisito de software |
| Name | Varchar(50) | Nombre del requisito de software |
| Description | Text | Descripción del requisito |
| Priority | Varchar(12) | Prioridad asociada al requisito, esta puede ser crítica, deseable o innecesaria |
| Source | Text | Documento o persona desde la cual surgió el requisito |
| Stability | Varchar(12) | Este campo tiene como propósito señalar si el requisito puede o no puede estar sujeto a cambio durante el ciclo de vida del software. Puede ser transable o intransable |
| Type | Int(11) | Puede ser de varios tipos entre los que destacan Funcional, Calidad y Restricción. Si es de tipo funcional debería estar asociado a por lo menos un rol. |
| Increment | Int(11) | Incremento o iteración del requisito. |
| Fecha_actualizacion | datetime | Fecha de la última actualización del requisito. |
| Status | Varchar(10) | Estado actual del requisito dentro del desarrollo (Cumple, No Cumple o Ambiguo) |
| Update_date | datetime | Fecha de la última actualización del requisito. |

**UR_np:** En esta tabla se encuentran los requisitos de software del sistema. Los requisitos de tipo funcional deberían tener asociado a lo menos un tipo de usuario (rol) asociado.

| Campo | Tipo | Observación |
|---|---|---|
| Id_ur | int(11) | PRIMARY_KEY, identificador del requisito de usuario |
| Name | Varchar(50) | Nombre del requisito de usuario |
| Description | Text | Descripción del requisito |
| Priority | Varchar(12) | Prioridad asociada al requisito, esta puede ser crítica, deseable o innecesaria |
| Source | Text | Documento o persona desde la cual surgió el requisito |
| Stability | Varchar(12) | Este campo tiene como propósito señalar si el requisito puede o no puede estar sujeto a cambio durante el ciclo de vida del software. Puede ser transable o intransable |
| Type | Int(11) | Puede ser de tres tipos (Funcional, Calidad, Restricción). Si es de tipo funcional debe estar asociado a por lo menos un rol. |
| Increment | Int(11) | Incremento o iteración del requisito. |
| Fecha_actualizacion | datetime | Fecha de la última actualización del requisito. |
| Status | Varchar(10) | Estado actual del requisito dentro del desarrollo (Cumple, No Cumple o Ambiguo) |
| Update_date | datetime | Fecha de la última actualización del requisito. |

Además de las tablas anteriores, existe un conjunto de relaciones, que debido a la cardinalidad que exhiben, se representarán a través de las siguientes tablas:

### A.2.2. Relaciones

**Md_rs:** La función de esta tabla es vincular los módulos con los requisitos de software.

| Campo | Tipo | Observación |
|---|---|---|
| Id_md | Int(11) | Identificador del modulo. |
| Id_sr | Int(11) | Identificador del requisito de software. |
| Id_project | Int(11) | Identificador del proyecto al cual pertenece. |

**Role_tc:** Esta tabla sirve para vincular los roles o tipos de usuario a algún caso de prueba en particular.

| Campo | Tipo | Observación |
|---|---|---|
| Id_tc | Int(11) | Identificador del caso de pruebas. |
| Id_rol | Int(11) | Identificador del rol o tipo de usuario vinculado. |
| Id_project | Int(11) | Identificador del proyecto al cual pertenece. |

**Sr_role:** La función de esta tabla es vincular los requisitos de software con los roles.

| Campo | Tipo | Observación |
|---|---|---|
| Id_sr | Int(11) | Identificador del artefacto de software. |
| Id_rol | Int(11) | Identificador del rol. |
| Id_project | Int(11) | Identificador del proyecto al cual pertenece. |

**Ur_role:** La función de esta tabla es vincular los requisitos de usuario con los roles. Hay que recordar que esta relación solo es valida cuando el requisito de usuario es de tipo funcional.

| Campo | Tipo | Observación |
|---|---|---|
| Id_ur | Int(11) | Identificador del artefacto de usuario. |
| Id_rol | Int(11) | Identificador del rol. |
| Id_project | Int(11) | Identificador del proyecto al cual pertenece. |

**Ur_sr:** El propósito de esta tabla es vincular los requisitos de usuario con los requisitos de software. En base a esta tabla se construye la matriz de trazabilidad de RU v/s RS.

| Campo | Tipo | Observación |
|---|---|---|
| Id_ur | Int(11) | Identificador del requisito de usuario |
| Id_sr | Int(11) | Identificador del requisito de software |
| Id_project | Int(11) | Identificador del proyecto al cual pertenece. |