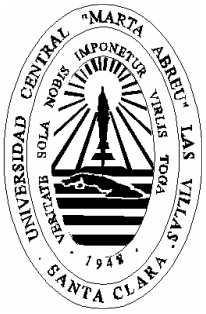
Universidad Central “Marta Abreu” de Las Villas

Facultad de Matemática, Física y Computación



Trabajo para optar por Título de Master en Ciencia de la Computación

Procedimiento para el desarrollo del proceso de ingeniería de requisitos en un proyecto software (PROCIR)

Autora

Ing. Leidy A. Fernández Sánchez

Tutora

Dra. Lourdes García Ávila

Consultante

Dra. Ana María García Pérez

2006

Dedico este trabajo a Lourdes, que fue mi guía.

A mi hijo, que fue mi inspiración.

A mi esposo, por lo que tuvo que esperar.

Agradezco a todas las personas que han contribuido al desarrollo de este trabajo.

A mi tutora, la Dra. Lourdes García Ávila y su familia por el apoyo y cariño que me han brindado.

A Karina, la China y Marthica por estar siempre conmigo en los momentos difíciles.

A Lucy, al Kendy, a Pável, a Cabrera y a mis compañeros de la Filial de Clientes,

todos de la Gerencia de ETECSA Cienfuegos, por todo lo que me dieron.

Al Dr. Sarmiento, Anel y Olamendy por apoyarme.

Resumen

En la actualidad persisten problemas en el desarrollo de software, entre ellos, un inadecuado entendimiento de las necesidades de los usuarios, incapacidad de absorber cambios en los requisitos e insatisfacciones de los clientes por inaceptable o bajo desempeño del software. Las principales causas son la administración insuficiente de requisitos; los problemas que afectan la comunicación; las inconsistencias no detectadas entre requisitos, diseño y programación; las validaciones tardías de requisitos; el enfrentamiento reactivo de riesgos y la propagación de cambios sin control. Los modelos de proceso de Ingeniería de Requisitos (IR), a pesar de su evolución, aún presentan carencias. Por tanto, para obtener un producto de calidad, se requiere una mejora en los procesos de IR. El objetivo general de la investigación es elaborar un procedimiento para efectuar la definición y gestión de los requisitos de un proyecto de software basado en la integración de las mejores prácticas, con un enfoque holístico, proactivo y estratégico que potencie de manera efectiva el desempeño del proceso de gestión de desarrollo del proyecto de software y la satisfacción del cliente en la PyME. Para lograr el objetivo, se realizó el estudio del estado del arte respecto a la ingeniería de requisitos y se validó el procedimiento propuesto en el objeto de estudio práctico, como vía de comprobación y factibilidad de la investigación realizada, con vistas a su posterior generalización a otras organizaciones.

Abstract

Nowadays, the problems still persist in the software development; for example, an inadequate understanding of the users' requirements, incapacity to take up the requirement's changes and customer's satisfaction for an unacceptable or low performance of software. The main causes are the insufficient management of requirements, the problems in the communication, the non detected inconsistencies between the requirements, design and programming; the delayed validations of the requirements, the late confrontation of the risks and the incorrect propagation of changes. The models of the requirements engineering process still exhibit deficiencies, in spite of their evolution. Therefore, it is necessary improving the requirements engineering process in order to obtain a high quality software product. The bottom line of this research is to design and propose a procedure to carry out the definition and requirements management of a software project based on the integration of the best practices, through a holistic, proactive and strategic approach. This procedure allows us an effective way to performance the development and management of software processes and the satisfaction of the client in the PyME. In order to achieve the objective, the study of the state-of-the-art with respect to the requirements engineering was made and the proposed procedure in the object of practical study was validated, like verification way and feasibility of the realized research, in order to be generalized to other organizations thereafter.

Índice

Introducción

Cada día se hacen mayores exigencias a la industria del software; exigencias medidas en términos de productividad, calidad y mantenibilidad de los sistemas. Esto lleva a la organización a la necesidad de reducir el esfuerzo en el desarrollo del proyecto, el tiempo del ciclo de vida y su costo sin afectar la calidad de este, además de satisfacer los requerimientos y los plazos de entrega exigidos por los clientes.

Para que la organización obtenga el software de calidad que demanda hoy la sociedad, se requiere de una sólida base teórica y práctica en los principios, métodos, técnicas y herramientas de la Ingeniería del Software[1], además de que la dirección de la empresa planifique el futuro, implante programas y controle los resultados de la función de la calidad con vistas a su mejora. Esto evitaría los errores de un proceso de desarrollo inmaduro, centrado en la etapa de implementación y no en todo el ciclo de vida, además de un software difícil de mantener, ya sea por su inaccesibilidad o por su alto costo.

Todos los problemas pueden rectificarse, la clave está en ofrecer un enfoque de ingeniería al desarrollo del software, y proporcionar asistencia práctica tanto a la dirección de la empresa como a la persona que lo desarrolla. Cualquier enfoque de ingeniería, incluida la Ingeniería del Software, debe descansar sobre un empeño de organización de calidad.

Sin embargo, en un análisis detallado de lo que sucede en la práctica realizado por García [García, 2000] se plantea entre otras causas la falta de aplicación del despliegue de la función de calidad, tan utilizada en Japón en este tipo de industria, la improvisación de los procesos de software y que no se garantiza la calidad desde el inicio del proyecto.

En la actualidad persisten problemas en el desarrollo de software, entre ellos, un inadecuado entendimiento de las necesidades de los usuarios, incapacidad de absorber cambios en los requerimientos e insatisfacciones de los clientes por inaceptable o bajo desempeño del software. Las principales causas de estos son la administración de requerimientos insuficiente, la comunicación ambigua e imprecisa, inconsistencias no detectadas entre requerimientos, diseño y programación, validaciones tardías de los requisitos, enfrentamiento tardío de riesgos y propagación de cambios sin control [Minasi, 2000], [García, 2000].

Otras investigaciones arrojan que:

- En la fase de requerimientos se introduce más del 55% de los defectos totales detectados en el software.

[1] Ingeniería del Software: La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software; es decir, la aplicación de ingeniería al software.

- De todo el mantenimiento correctivo de software, que demanda el 75% de los recursos totales, más del 80% proviene de la Fase de Requerimientos.
- Un error no detectado en la fase de requerimientos cuya corrección costaría apenas 1 unidad de tiempo o dinero, costará 67 veces más, al detectarlo en la fase de producción.

La parte más difícil de construir de un sistema software es decidir qué construir. [...] Ninguna otra parte del trabajo afecta más negativamente al sistema final si se realiza de manera incorrecta. Ninguna otra parte es más difícil de rectificar después [Brooks 1995]. Por lo tanto, el desarrollo del software sólo puede ser iniciado cuando se tiene bien establecido lo que se quiere producir.

Resultados de estudios y encuestas realizadas a directores de proyectos indicaron que los principales factores de éxito de un proyecto son: implicación de los usuarios, apoyo de los directivos y enunciado claro de los requisitos. Mientras que los principales factores de fracaso de un proyecto son: falta de información por parte de los usuarios, especificaciones y requisitos incompletos y especificaciones y requisitos cambiantes. Por lo tanto, las exigencias a las organizaciones productoras de software de un producto de calidad requieren de una mejora en los procesos de ingeniería de requisitos ya que en este se descubre, analiza, negocia, valida y especifica lo que debe hacerse.

Existen insuficiencias en la gestión de requisitos tanto al nivel de proyecto como al nivel de organización. Esto requiere que el proceso de ingeniería de requisitos tiene que estar bien definido y ser desarrollado de forma disciplinada para repetir los éxitos anteriores en nuevos proyectos.

Los modelos de proceso de ingeniería de requisitos, a pesar de su evolución aún presentan carencias. No todos abordan integralmente el proceso, ya que algunos no incorporan la gestión de los requisitos, ni el manejo de los cambios, o no tiene en cuenta la inevitable interacción con el resto del ciclo de vida del proyecto. Otro aspecto importante es que las mediciones a este proceso aún son insuficientes por falta de datos históricos.

El proceso de ingeniería de requisitos se hace complejo por la presencia del factor humano. En este proceso, la comunicación entre los participantes, es fundamental para obtener una buena especificación del sistema. Sin embargo, para llevar los proyectos a buen término no basta con aplicar ingeniería de software si esta no se hace racionalmente, sin despilfarrar recursos y utilizando en cada momento las técnicas más adecuadas.

En este ámbito se sitúa la ingeniería de requisitos, como un área de investigación que procura atacar un punto fundamental en el proceso de producción, que es la definición de lo que se quiere producir. Cabe a la ingeniería de requisitos, como subproceso de la ingeniería de software, proponer métodos, técnicas y herramientas que faciliten el trabajo de definición de lo que se quiere de un software. Por

lo tanto, la ingeniería de requisitos tiene una interacción muy fuerte con aquéllos que demandan un producto de software.

Jackson afirma que la ingeniería de requisitos se ubica en el punto de encuentro entre lo informal y lo formal del desarrollo de software [Jackson, 2001].

La pequeña y mediana empresa de software (PyME) necesita mejorar el desempeño sus procesos. Sin embargo, existe la tendencia de trasladar los requisitos que imponen los modelos como el CMMI [CMMI-SW, 2002] e ISO/IEC TR 15504-2 [SPICE, 1998] a la empresa, sin que estos respondan a las características de la entidad.

Es necesario adaptar los requisitos de estos modelos en la empresa de acuerdo a los objetivos de negocio y la estructura de la organización. Es por ello, que las micro, pequeña y mediana empresa de software requieren de un instrumento metodológico adecuado y pertinente que les permita accionar de manera ágil, proactiva y estratégica sobre los procesos para administrar su portafolio de proyectos acorde a los objetivos de la organización.

Son varios los esfuerzos que se requieren para dar respuesta a las necesidades de implementar enfoques modernos e integrales de gestión orientados hacia la calidad, la productividad y la competitividad.

Todo lo anterior pone de manifiesto la gran tendencia al cambio de la naciente industria de software y al logro de la madurez de los procesos y la necesidad de un enfoque más disciplinado del software.

Cuba no está exenta de esta problemática, sus procesos son inmaduros y no están definidos. Las empresas cubanas del software deben que realizar profundas transformaciones, que no es solo el uso de la tecnología más novedosa, sino la ejecución de tareas relativas a la gestión de la calidad, al nivel de la organización y también al nivel de los proyectos, como parte del perfeccionamiento empresarial.

En Cuba se realizan diferentes proyectos en este campo por el CRIS (Centro de Referencia de Ingeniería del Software) y por la Comisión Nacional de Calidad de Software del Ministerio de Informática y las Comunicaciones (MIC).

El análisis histórico - lógico en torno a la gestión de la calidad del software, conduce a inferir que continúa siendo una ***situación problemática*** aquella que se deriva de las insuficiencias en la definición y gestión de los requisitos del software en la PyME, por la carencia de enfoques holístico, proactivo y estratégico. Esto justifica el planteamiento del ***problema científico:***

Ausencia de un procedimiento con un fundamento metodológico para la PyME que:

- permita la definición y gestión de los requisitos del software durante el ciclo de vida del proyecto del software,
- permita la definición y gestión de los riesgos asociados a las características de los requisitos del software desde el inicio del proyecto,
- obtener un producto de calidad que satisfaga al cliente cumpliendo el equilibrio de plazo y costo del proyecto.

Siendo el ***objeto de estudio*** los procesos de la ingeniería de requisitos. Se hace indispensable, realizar un profundo proceso de investigación para caracterizar este proceso y mejorarlo.

De acuerdo a lo anteriormente planteado, así como de la revisión de la bibliografía especializada y otras fuentes en la construcción del marco teórico o referencial de la investigación, se formuló la ***hipótesis*** siguiente:

Con la concepción e implementación de un procedimiento metodológico para el desarrollo del proceso de ingeniería de requisitos en la PyME, basado en la integración de las mejores prácticas, se puede realizar una mejor definición y gestión de los requisitos de un proyecto de software, así como trazar un conjunto coherente de políticas, "buenas prácticas" y acciones de mejora con un enfoque holístico, proactivo y estratégico que contribuya a elevar la satisfacción del cliente en estas organizaciones.

La validación de la hipótesis se realiza si se comprueba que:

- El procedimiento desarrollado se caracteriza, tanto en su concepción como en su implementación, por poseer las cualidades que hacen factible su aplicación racional en el objeto de estudio práctico seleccionado, a partir de criterios como: pertinencia, consistencia lógica, parsimonia, flexibilidad, generalidad que permita extender su empleo a otras entidades pequeñas y medianas de la industria del software.
- La aplicación de las herramientas propuestas en el objeto de estudio práctico permite:
  - obtener especificaciones de requisitos de software completas, formales y acordadas entre todos los participantes, que cumplan con las características de calidad definidas.
  - exponer que la integración de las mejores prácticas, los métodos y herramientas facilitan una adecuada realización de las tareas de gestión de requisitos en función de los objetivos, metas y estrategias de la organización, y
  - poner en evidencia la posibilidad de mejora gradual en los niveles actuales de desempeño de la gestión de los requisitos del software y la satisfacción del cliente en el objeto de estudio práctico.

Todo lo anterior conduce a plantear en la investigación, el sistema de objetivos siguiente:

Objetivo general

Elaborar un procedimiento metodológico para el desarrollo del proceso de ingeniería de requisitos de un proyecto de software basado en la integración de las mejores prácticas, con un enfoque holístico, proactivo y estratégico que potencie de manera efectiva el desempeño del proceso de la ingeniería de requisitos del proyecto de software y la satisfacción del cliente en la PyME.

Objetivos específicos

- Construir el marco teórico-referencial de la investigación derivado de la consulta de la literatura nacional e internacional actualizada, y otras fuentes de referencia sobre la temática objeto de estudio.
- Realizar un análisis crítico sobre el estado actual de la ingeniería de requisitos en la PyME, enfatizando en los métodos, técnicas y metodologías que se emplean, de modo que se detecten sus principales deficiencias, así como las posibilidades de su perfeccionamiento.
- Diseñar un procedimiento metodológico para el desarrollo del proceso de ingeniería de requisitos de un proyecto de software con un enfoque holístico, proactivo y estratégico.
- Desarrollar el conjunto de procedimientos, coherentes con los objetivos del negocio, que permita en la PyME: obtener y gestionar especificaciones de requisitos de software completas, formales y acordadas entre todos los participantes, que cumplen con las características calidad definidas; el mejoramiento gradual de los niveles de desempeño actuales del proceso de ingeniería de requisitos, que logre la satisfacción del cliente en estas organizaciones.
- Validar el procedimiento propuesto en el objeto de estudio práctico, como vía de comprobación y factibilidad de la investigación realizada, con vistas a su posterior generalización a otras organizaciones.

Los ***métodos de investigación*** utilizados fueron: modelación, análisis y síntesis, inductivo, observación y el método general de solución de problemas.

La ***novedad científica*** está dada por:

- La concepción y definición de un procedimiento metodológico para el desarrollo del proceso de ingeniería de requisitos en la PyME que explica la problemática objeto de estudio y conduce a su solución.
- La definición del procedimiento general con enfoque holístico, proactivo y estratégico que, soportado por un conjunto de procedimientos específicos, está sustentado en un nuevo enfoque funcional del equipo de trabajo de gestión de requisitos, en la definición de condiciones de inicio

para este proceso, en la gestión de los riesgos asociados a las características de calidad de los requisitos, en la gestión de los requisitos y sus cambios a lo largo de todo el ciclo de vida del proyecto y en ser independiente de la metodología de desarrollo adoptada, permite especificar y gestionar requisitos, el mejoramiento gradual del proceso de desarrollo y la gestión de un proyecto de software que logre la satisfacción del cliente en estas organizaciones.

El Capítulo I, referido al marco teórico y referencial de la investigación; Capítulo II, donde se plantea el procedimiento para el desarrollo del proceso de ingeniería de requisitos en la PyMe de software y el Capítulo III que contiene la aplicación en un caso de estudio práctico y la validación del procedimiento.

Capítulo 1. Marco teórico - referencial de la investigación

La revisión de la literatura especializada, así como de otras fuentes bibliográficas y referenciales consultadas, se estructuró de forma tal que permitiera el análisis del estado del arte y de la práctica sobre la temática objeto de estudio, permitiendo sentar las bases teórico-prácticas de la investigación. El hilo conductor seguido, como estrategia de construcción del marco teórico-referencial de la investigación, se expone en la figura 1.1. En principio, se analizan diferentes aspectos referentes a la problemática general que existe en torno al papel que juega la ingeniería de requisitos, su alcance, objetivos y su relación con el ciclo de vida del desarrollo del software en el marco estratégico-competitivo actual. El cuerpo principal contempla diferentes tópicos sobre el tema de investigación, que van desde los orígenes, definiciones, conceptos y enfoques, hasta los modelos, técnicas y herramientas más empleados en el proceso de ingeniería de requisitos, conjuntamente con la medición del proceso y su implicación en la calidad del software. Los contenidos aquí expuestos, además de representar los enfoques predominantes con mayor aceptación en la literatura académica, gozan también de validez y evidencia empírica. Finalmente, se hace alusión al estado actual de estos contenidos en el marco de las pequeñas y medianas empresas de software en Cuba.

1.1 Ingeniería de requisitos

El software de computadora se ha convertido en un factor clave que diferencia los productos y servicios modernos. Hoy su impacto en la sociedad y en la cultura continúa siendo profundo. Al mismo tiempo que crece su importancia, la comunidad del software trata continuamente de desarrollar tecnologías que hagan más sencilla, rápida y menos costosa la producción de programas de computadora de alta calidad [Pressman, 2002].

El software es un conjunto de programas y su documentación asociada, tales como sus requisitos, diseño, modelos y manuales de usuarios [Sommerville, 2005].

A la vez que el software se ha convertido en el elemento clave de la evolución de los sistemas y productos informáticos, también constituye un factor que limita la evolución de los sistemas informáticos. El intento de la ingeniería de software (disciplina de la ingeniería que concierne a todos los aspectos de la producción de software) es proporcionar un marco de trabajo para construir software con mayor calidad.

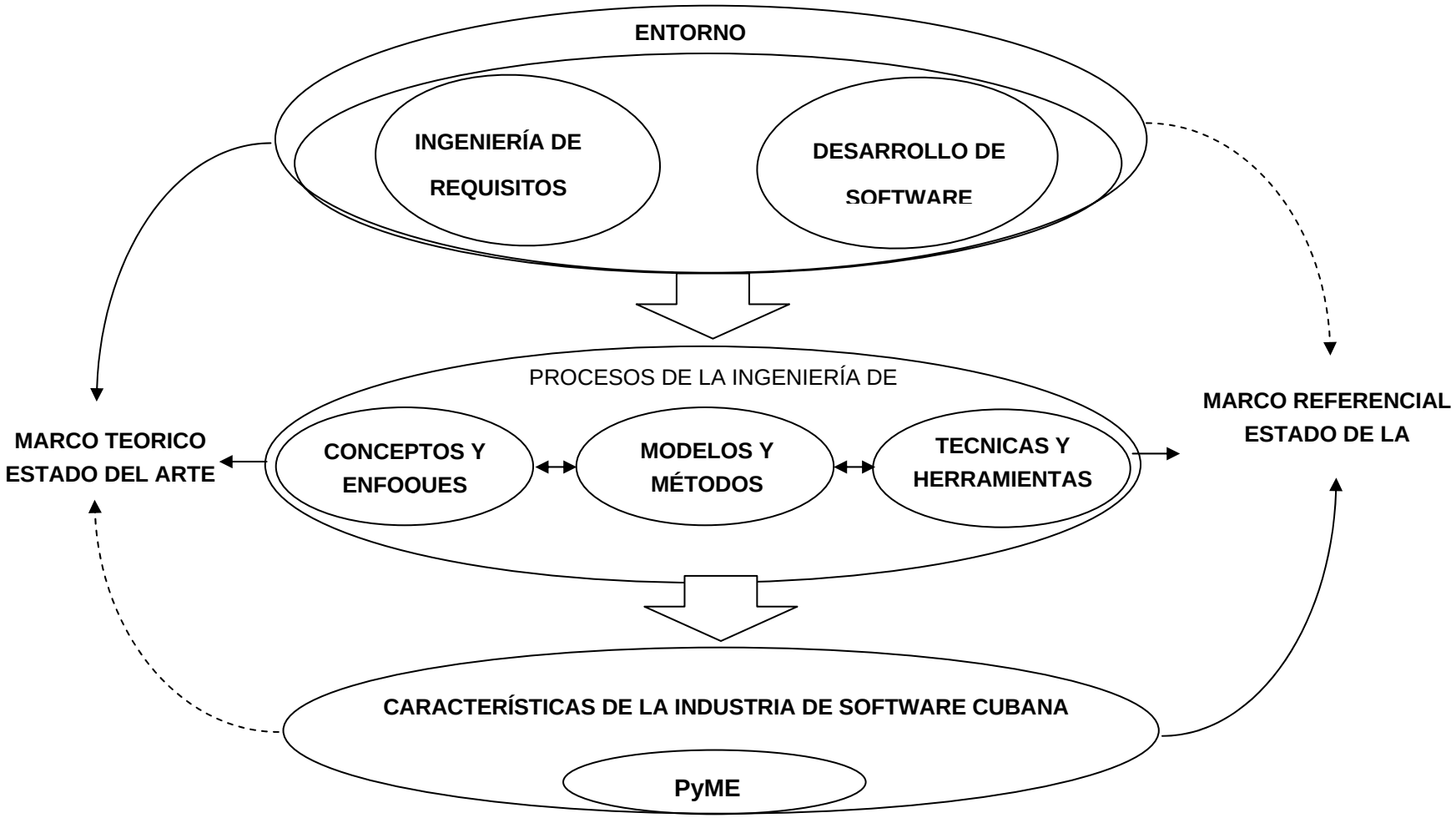


Figura 1.1 Estrategia seguida en el análisis y estudio de la bibliografía para la construcción del marco teórico - referencial de la investigación.

Según Alan Davis en [Thayer y Dorfman, 1997] Ingeniería de Software (IS) es "la aplicación de principios científicos a: 1) la transformación ordenada de un problema en una solución de software, y 2) el mantenimiento subsiguiente de ese software hasta el final de su vida útil".

Pressman establece sus objetivos: "Los objetivos claves de la ingeniería de software son definir, crear y aplicar 1) una metodología bien definida, dirigida a un ciclo de vida de planeamiento, desarrollo, y mantenimiento; 2) un conjunto establecido de componentes de software que documenta cada paso en el ciclo de vida y muestra un seguimiento paso a paso, y 3) un conjunto de hitos predecibles que pueden ser revisados a intervalos regulares a través del ciclo de vida del software" [Pressman, 1997].

La Ingeniería del Software, por tanto, no sólo cubre los aspectos puramente tecnológicos de la producción del software, sino que conlleva además la gestión de sus procesos internos y recursos (los presupuestos, equipos de desarrollo, de proyectos, de la planificación de estos, de los riesgos, de los cambios). Esta disciplina comprende, tres elementos claves: métodos, herramientas y procesos (denominados frecuentemente paradigmas de la Ingeniería del Software).

Los métodos indican "cómo" construir el software. Abarcan la planificación y estimación de proyectos, el análisis de los requisitos del sistema y del software, el diseño de estructuras de datos, la arquitectura de programas y procedimientos algorítmicos, así como la codificación, prueba y mantenimiento.

Las herramientas suministran un soporte automático para los métodos. Existen para soportar cada uno de los métodos mencionados anteriormente y proporcionan un entorno con integración de diferentes herramientas, que es lo que se denomina un sistema CASE (*Computer Aided Software Engineering*).

Los procesos juntan los métodos y las herramientas. Ellos definen la secuencia en la que se aplican los métodos, las entregas (documentos, informes, etc.) que se requieren, los controles que ayudan a asegurar la calidad, coordinar los cambios y las directrices que ayudan a los gestores del software a evaluar el progreso.

A pesar de la creciente participación del software en el mundo actual y de los avances producidos, su proceso no es adecuado. Es función de la ingeniería de software evitar que estos errores ocurran, produciendo software más robustos y proveyendo procesos de producción más confiables [Ridao, 2002].

En la actualidad persisten problemas en el desarrollo de software, entre ellos, un inadecuado entendimiento de las necesidades de los usuarios, incapacidad de absorber cambios en los requerimientos e insatisfacciones de los clientes por inaceptable o bajo desempeño del software.

Las principales causas de estos son la administración de requerimientos insuficiente, comunicación ambigua e imprecisa, inconsistencias no detectadas entre requerimientos, diseño y programación, validaciones tardías de los requisitos, enfrentamiento tardío de riesgos y propagación de cambios sin control [Minasi, 2000], [García, 2000].

El punto de partida del proceso de producción de software es el momento en que se define lo que se quiere. Por lo tanto, el desarrollo del software sólo puede ser iniciado cuando se tiene bien establecido lo que se quiere producir. Cuando se trata de sistemas complejos, esta definición de lo que se quiere no es trivial. Por ello, muchos productos de software no se comportan como sería deseable. Hacer una definición que cubra todas las necesidades de un sistema complejo es una tarea difícil. Y es más difícil aún si no se cuenta con métodos, técnicas y herramientas adecuadas.

En este ámbito se sitúa la ingeniería de requisitos, como un área de investigación que procura atacar un punto fundamental en el proceso de producción, que es la definición de lo que se quiere producir. Cabe a la ingeniería de requisitos, como subproceso de la ingeniería de software, proponer métodos, técnicas y herramientas que faciliten el trabajo de definición de lo que se quiere de un software. Por lo tanto, la ingeniería de requisitos tiene una interacción muy fuerte con aquéllos que demandan un producto de software, sean éstos el mercado o los clientes de una aplicación que será especialmente construida.

Jackson afirma que la ingeniería de requisitos se ubica en el punto de encuentro entre lo informal y lo formal del desarrollo de software. Los programas computacionales son efectivamente construcciones formales, que utilizan métodos matemáticos basados en teoría de tipos, precondiciones y poscondiciones, e invariantes. Pero el mundo de los seres humanos y objetos físicos en el cual los requisitos se ubican es informal, y no puede ser tratado adecuadamente por métodos puramente formales [Jackson, 2001].

Uno de los beneficios de una buena ingeniería de requisitos es la prevención de backtracking (vuelta atrás) en etapas tardías del proceso de desarrollo. Averiguar correctamente las necesidades de los clientes, disminuye costosas tareas de reingeniería después de la codificación [Kovitz, 1998].

La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación [Pressman, 2002]. Tiene por objetivo la obtención de los requisitos2 del sistema que se desea construir con cierto nivel de abstracción y cumpliendo ciertas características, haciendo un puente entre el espacio del problema y el espacio de la solución como puede verse en la figura 1.2.

[2] En el presente trabajo utilizaremos indistintamente las palabras requisitos y requerimientos, asumiéndolas como sinónimas.

Figura 1.2 La IR como puente entre los espacios del problema y la solución.

El proceso de descubrimiento y captura de requisitos de cualquier ciclo de desarrollo de sistemas ha evolucionado en la última década a pasos agigantados al pasar de una concepción reduccionista, en donde se asumía que los requisitos eran proporcionados única y exclusivamente por los clientes, a una serie compleja de actividades de comunicación y descubrimiento, por parte de los ingenieros de requisitos, de las necesidades de los usuarios.

La relativa corta vida de la disciplina de la ingeniería de requisitos y el afán de la comunidad en resolver los problemas atinentes a ella, ha generado un sinnúmero de métodos, metodologías, procesos y herramientas pero que aún tienen carencias que no llegan a resolver de manera satisfactoria todos los inconvenientes que se presentan en esta fase de cualquier proyecto de desarrollo, entre estas, la falla de mecanismos de comunicación entre clientes y analistas; etapas, actividades y documentos sin clara delimitación conceptual, escasa reutilización de requisitos y poca gestión del proceso de la IR

La autora coincide con Durán en que la ingeniería de requisitos es un proceso de descubrimiento y comunicación de las necesidades de clientes y usuarios y la gestión de los cambios de dichas necesidades [Durán, 2002]. La alta presencia del factor humano hace que lo más importantes de esta definición sea la comunicación, característica que hace complejo al proceso. Este aspecto es responsable de la que disciplina incluya aspectos socio-culturales y no sólo de índole técnica [Goguen, 1994]. Además, la IR debe ser considerada como un proceso de construcción de una especificación de requisitos en el que se avanza desde unas especificaciones iniciales, que no poseen las características oportunas, hasta especificaciones finales completas, formales y acordadas entre todas las partes [Pohl, 1997].

1.1.1 Ingeniería de requisitos y ciclo de vida del desarrollo de software

La relación no lineal entre la ingeniería de requisitos y el resto del ciclo de vida del desarrollo del software, mostrada en la figura 1.3, ha sido detectada desde antaño y propuestas metodológicas como el Modelo en Espiral [Boehm, 1988] y el Proceso Unificado de Racional [Jacobson, et al.,

1998], incorporan estrategias iterativas dentro de sus procesos de desarrollo para facilitar la ejecución de actividades propias de la ingeniería de requisitos, una vez iniciado el resto del proceso de desarrollo, al detectarse en éste la necesidad de renegociar algunos requisitos de difícil implementación o porque aparecen nuevos requisitos durante el proceso de desarrollo, entre otros. Debe tenerse en cuenta que la IR continúa durante todo el proceso de desarrollo [Sawyer y Kontoya 1999].

Otro aspecto a considerar es la línea base de requisitos. Esta es una estructura dinámica que se genera durante el proceso de ingeniería de requisitos que evoluciona junto al proceso de desarrollo de software y que acompaña las tareas de mantenimiento. Es importante enfatizar la importancia de la capacidad de referencias cruzadas entre la línea base de requisitos y el proceso de desarrollo y mantenimiento. Esto permite el seguimiento de los requisitos en cualquier punto durante el desarrollo y el mantenimiento hacia su punto de origen [García, 1999].

En la figura 1.4 se observa como la evolución de la línea base va aparejada con la evolución del proceso de software, tanto en lo que se refiere al desarrollo, como al mantenimiento.

1.1.2 Dimensiones de la ingeniería de requisitos

Una posible visión de la ingeniería de requisitos es considerarla como un proceso de construcción de una especificación de requisitos en el que se avanza desde especificaciones iniciales, que no poseen las propiedades oportunas, hasta especificaciones finales completas, formales y acordadas entre todos los participantes [Pohl, 1994, Pohl, 1997].

Desde este punto de vista son tres las dimensiones que comprende la IR que ilustran el avance del proceso desde especificaciones incompletas, informales e individuales hacia unos requisitos especificados de manera completa, formal y acordada entre todos los participantes, como se indica en la figura 1.5.

- El grado de compleción (plenitud) del conocimiento sobre el sistema que se desea desarrollar: vaga, media y completa.
- El grado de formalismo de la representación del conocimiento sobre los requisitos: informal, semiformal y formal.
- El grado de acuerdo al cual llegan los actores del proceso mediante algún tipo de negociación: personal y grupal.

La comunicación como proceso de la IR

En el proceso de elaboración de una propuesta de solución que resuelva las necesidades de los usuarios (incluidas su aceptación y verificación por el cliente) Díez [Díez, 2001] los resume en:

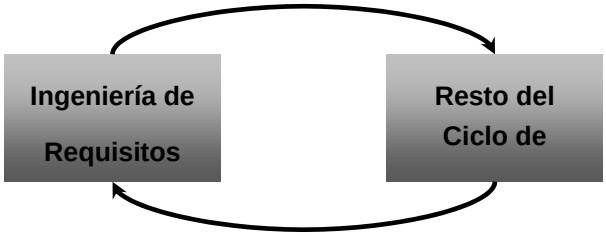


Figura 1.3 IR en el ciclo de vida del software.

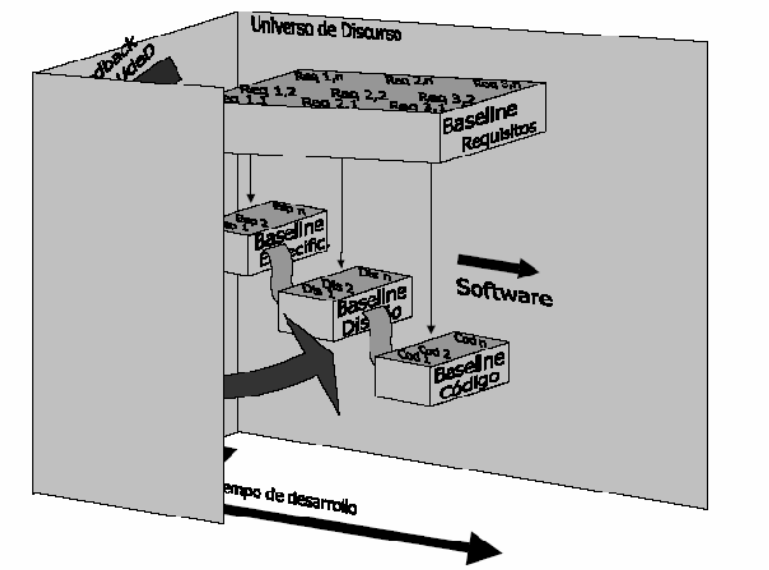


Figura 1.4 Línea base de los requisitos y el proceso de desarrollo.

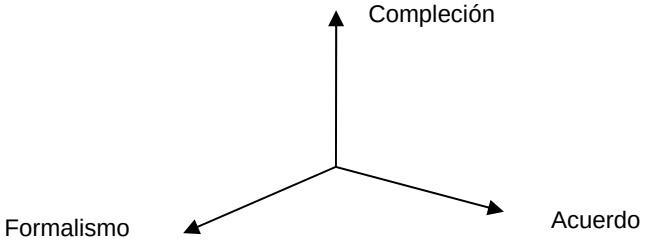


Figura 1.5 Dimensiones de la IR.

- Los usuarios tienen un problema a resolver, los ingenieros informáticos que realizan la elicitación de los requisitos deben entender dicho problema, el contexto en el cual surge y las expectativas de los usuarios.
- Los ingenieros informáticos que realizan la elicitación de los requisitos proponen una solución, los usuarios deben entender y validar la propuesta de preferencia antes de que comience el desarrollo propiamente dicho del sistema.

La comunidad de usuarios potenciales del sistema puede ser muy diversa, con notorias diferencias en la formación y expectativas, con relación a las funciones del sistema a construir y a los escenarios donde deben ejecutarse. Las funcionalidades deben ser especificadas por los ingenieros de requisitos y diseñadas e implementadas por los desarrolladores. La figura 1.6 anterior ilustra una estrategia de comunicación entre los diversos participantes del flujo de trabajo.

Se utiliza el término Requisitos-C para describir los requisitos desde el punto de vista de clientes y

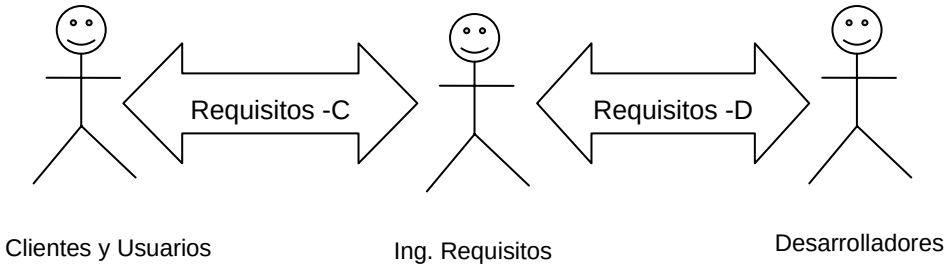


Figura 1.6 Procesos de comunicación en la IR.

usuarios) y el de Requisitos-D para los del punto de vista de los desarrolladores [Durán, 2000], [SWEBOK, 2004].

El estudio realizado sobre ingeniería de requisitos en los modelos y normas de calidad de los procesos de software revela que el modelo CMMI [Chrissis, 2003], el estándar SPICE [SPICE, 1998] y la Guía para el Cuerpo de Conocimiento de Ingeniería de Software [SWEBOK, 2004], concuerdan en las características siguientes:

- La IR es un proceso iterativo, ya que al ser un proceso de descubrimiento y comunicación, difícilmente llegará a realizarse en forma lineal.
- Los requisitos no siempre son entregados en su totalidad por los clientes y usuarios, así que los ingenieros de requisitos también deben saber descubrirlos.
- Los límites de las actividades de IR son difíciles de establecer por la misma naturaleza del proceso.

- Los requisitos pueden evolucionar tan rápidamente que pueden cambiar antes de haber concluido el desarrollo del sistema.

1.1.3 Requisitos del software

Existe falta de uniformidad en la terminología empleada en la ingeniería de requisitos, tanto para los conceptos básicos como para los procesos y los productos [Davis, 1993], Pohl, 1997]. Uno de los conceptos afectados por la falta de uniformidad es el de requisito. Varias son las definiciones de requisitos que aparecen en la literatura científica, las más representativas aparecen a continuación.

La IEEE [IEEE 1990] define requisito como: (a) una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo; (b) una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, una especificación u otro documento formal; (c) una representación en forma de documento de una condición o capacidad como las expresadas en (a) o en (b).

Mientras que en [DoD, 1994] se ofrece una definición más concisa, requisito es la característica del sistema que es una condición para su aceptación.

Otra definición es la dada por Goguen [Goguen, 1994], requisito es la propiedad que un sistema debería tener para tener éxito en el entorno en el que se usará.

La autora coincide con la definición dada en [DoD, 1994] ya que argumenta que requisito es una necesidad del cliente que debe ser satisfecha.

La comunidad académica identifica tres dimensiones en las cuales se pueden clasificar los requisitos según diversos propósitos, como puede observarse en la figura 1.7. Ellas son:

El *ámbito*, que indica si el requisito debe cumplirse al nivel del soporte físico (hardware), soporte lógico (software), organización o sistema, particularidad ésta muy importante en la especificación de sistemas telemáticos, sistemas empotrados y sistemas de control en donde es de vital importancia definir las responsabilidades del soporte físico asociado a una aplicación.

La *audiencia*, que indica el tipo de persona (capaz de entenderlo) a la que está dirigido el requisito. Los tipos de audiencia son clientes, usuarios y los desarrolladores del sistema.

La *característica*, que indica la naturaleza del sistema deseada que se especifica y son de tipo funcional, no funcional y de información.

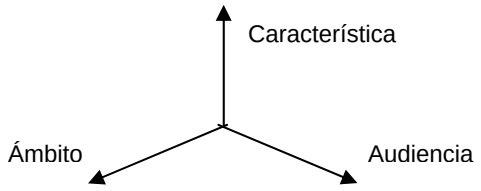


Figura 1.7 Dimensiones de los Requisitos

Propiedades deseables de los requisitos

Sobre las propiedades que deben tener los requisitos y las especificaciones de estos, hay una amplia bibliografía. La autora, después de analizar varias fuentes entre ellas [IEEE, 1996], [IEEE, 1993], [Davis, 1993], [Wieringa, 1996], considera las propiedades más importantes las siguientes:

Comprensible por clientes, usuarios y desarrolladores. La propiedad más importante de una especificación es que sirva como canal de comunicación entre los participantes en el proceso de ingeniería de requisitos. En nuestra opinión, la mejor forma de lograr esta comunicación es pensar en la audiencia a la que van dirigidos los requisitos, de forma que para comunicarse con los clientes y usuarios lo mejor es utilizar requisitos–C y para hacerlo con los desarrolladores lo mejor es usar requisitos–D. Esto puede verse en la figura 1.6.

En [IEEE 1993] se advierte de la necesidad de que la especificación de requisitos sea preparada conjuntamente por clientes, usuarios y desarrolladores, aunque no especifica ningún estilo para su redacción. Esto es considerado por la autora como un elemento importante para eliminar barreras de comunicación entre los *stakeholders*[3].

Correcta: una especificación de requisitos es correcta si y sólo si todo requisito contenido en ella representa alguna propiedad requerida por el sistema a desarrollar [IEEE 1993], [Davis, 1993]. En otras palabras, ningún requisito debe ser innecesario. Deben formar un conjunto único, siguiendo la nomenclatura de [IEEE 1996].

No ambigua: una especificación de requisitos no es ambigua si y sólo si todo requisito contenido en ella tiene una sola interpretación [IEEE 1993], [Davis, 1993]. Para evitar interpretaciones erróneas conviene que aquellos términos con más de una posible interpretación aparezcan en un glosario de términos en el que se especifique claramente su significado en el documento.

Completa: una especificación de requisitos es completa si incluye todos los requisitos identificados por el cliente y todos los requisitos necesarios para la definición del sistema [IEEE 1996].

En [Wieringa, 1996], se considera una especificación como completa si no se omite ningún requisito. Desde un punto de vista práctico, esto significa que el cliente ha validado los requisitos, requisitos, es decir, que está de acuerdo con ellos y que confirma que no conoce más requisitos que

[3] El término inglés habitual para referirse a los participantes en la fase de ingeniería de requisitos es *stakeholder*. Literalmente, *stakeholder* significa *el que tiene una estaca*, aunque el significado real se deduce de la expresión *to have a stake in something*, que significa tener intereses en algo. Por lo tanto, un *stakeholder* es cualquier persona que tenga intereses en juego durante la fase de ingeniería de requisitos: clientes, usuarios, ingenieros de requisitos, desarrolladores, etc. En este trabajo utilizaremos *implicado, participante, involucrado* como sinónimos de este término en español.

sean necesarios. En [Wieringa, 1996] también se considera que los requisitos deben estar priorizados por el cliente para considerar una especificación como completa.

Consistente: una especificación de requisitos es consistente externamente si y sólo si todo requisito contenido en ella no está en conflicto con otros documentos de nivel superior [Davis, 1993]. Es consistente internamente si y sólo si no existen conflictos entre los requisitos que contiene.

Según [IEEE 1993] y [Davis, 1993], los conflictos entre requisitos pueden ser de conducta (dos o más requisitos especifican conductas distintas del sistema para las mismas condiciones y el mismo estímulo externo); de términos (se utilizan términos distintos para referirse al mismo concepto); de característica (dos o más requisitos especifican aspectos contradictorios para la misma característica del sistema) y temporales (dos o más requisitos exigen características temporales contradictorias al sistema).

Verificable: una especificación de requisitos es verificable si y sólo si todo requisito contenido en ella es verificable, es decir, existe un proceso finito y de costo razonable por el que una persona o una máquina pueda comprobar que el sistema final cumple el requisito [Davis, 1993], [IEEE, 1993].

Una condición absolutamente necesaria para que un requisito sea verificable es que no sea ambiguo y que se defina de forma medible, ya que no se puede comprobar algo que no se puede medir ni definir de forma precisa.

Modificable: una especificación de requisitos es modificable si y sólo si su estructura y estilo de redacción permite que los cambios se puedan realizar fácil, completa y consistentemente [Davis, 1993], [IEEE, 1993].

Para conseguir esta propiedad, la especificación debe estar organizada coherentemente y debe contar con los índices y las tablas de referencias cruzadas oportunas, no debe ser redundante (debe estar normalizada según la terminología utilizada en [IEEE 1996] y los requisitos deben expresarse individualmente y no de forma conjunta. En [IEEE 1996] también se considera la necesidad de mantener las distintas versiones de la especificación que se vayan produciendo debido a los cambios, es decir, que la especificación sea configurable.

Rastreable: una especificación de requisitos es rastreable si y sólo si para cada requisito contenido en ella se conoce su origen y puede referenciarse como origen en posteriores documentos durante el desarrollo, es decir, cada requisito puede rastrearse hacia atrás y hacia delante [Davis, 1993], [IEEE, 1993].

Una condición necesaria para que un requisito pueda rastrearse hacia delante o hacia atrás es que pueda referenciarse de manera única, normalmente mediante algún tipo de código. La forma habitual de registrar la rastreabilidad son las matrices de rastreabilidad [DoD, 1994], [Davis, 1993].

Anotada con importancia y estabilidad: una especificación de requisitos está anotada con importancia y estabilidad si y sólo si cada requisito contenido en ella está anotado con la importancia que tiene su cumplimiento para clientes y usuarios y la estabilidad que se espera del requisito, es decir, la probabilidad de que cambie durante el desarrollo [Davis, 1993], [IEEE, 1993].

Conocer la importancia de un requisito permite decidir en qué orden implementarlos en el caso de que se hayan acordado entregar versiones sucesivas del producto. Por otro lado, la estabilidad permite a los diseñadores conocer qué grado de flexibilidad deben introducir en el sistema para soportar futuros cambios.

Independiente del diseño y la implementación: una especificación de requisitos es independiente del diseño y de la implementación si y sólo si no especifica una determinada descomposición del sistema (arquitectura) ni ningún aspecto de su posible implementación [Davis, 1993], [Wieringa, 1996]. Sólo deben admitirse requisitos que limiten la libertad de los diseñadores y programadores en el caso de que el cliente lo solicite explícitamente.

Atributos de los requisitos

- Necesidad (imprescindible, deseable u opcional)
- Prioridad (alta, media y baja)
- Riesgo (relacionado con la dificultad de implementarlo: alto, moderado, bajo).

Para negociar entre funcionalidad, planificación, presupuesto y nivel de calidad, es conveniente que los requisitos funcionales tengan asignado un nivel de necesidad (tres o cuatro categorías: esencial, deseable, opcional), así como un nivel de prioridad temporal (dos o tres categorías: alta, media, baja)

La necesidad de un requisito hace referencia al interés de los usuarios/clientes en que la aplicación lo realice, y hasta qué punto estarían dispuestos a pasarse si él. La prioridad de un requisito hace referencia al orden temporal: indica en qué fase de construcción del sistema se incluirá la funcionalidad que realice el requisito.

La aplicación de la Regla de Pareto implica que el 20% de la aplicación proporciona el 80% de su valor, no más del 20% de los requisitos deberían ser “esenciales”.

La ingeniería de requisitos es el proceso mediante el cual se especifican y validan los servicios que debe proporcionar el sistema, así como las restricciones sobre las que se deberá operar. Consiste en un proceso iterativo y cooperativo de análisis del problema, documentando los resultados en una variedad de formatos y probando la exactitud del conocimiento adquirido [Ferreira, 2001].

Los procesos software necesitan mejorarse continuamente. Para la IR lo mas importante es: mejorar la calidad, o sea, que haya menos errores, lograr documentos de requisitos mas completos, que

reflejen mejor las necesidades del usuario; reducir el tiempo; disminuir los recursos; reutilizar requisitos; implicar al usuario final.

La eficacia de este proceso de IR puede ser afectado en mayor o menor medida (positiva o negativamente) por el nivel de habilidad de los participantes en él, por la estructura del equipo de trabajo, el conocimiento del cliente, la tecnología que va a ser implementada, las herramientas que serán utilizadas y otros aspectos.

La importancia de este proceso es esencial puesto que los errores más comunes y más costosos de reparar, así como los que más tiempo consumen, se deben a una inadecuada IR.

En resumen, la autora plantea que la ingeniería de requisitos establece la definición de requisitos como un proceso en el cual se descubre, analiza, negocia, valida y especifica lo que debe hacerse. En su ejecución se realiza el consenso de los diferentes puntos de vista de los *stakeholders*, usando una combinación de métodos, técnicas y herramientas, de acuerdo al modelo elegido. Su producto es un documento de Especificación de Requisitos del Software (ERS). Este proceso sucede en un contexto previamente definido que llamamos el *Dominio del Problema*, sobre el cual el software deberá ser desarrollado y operado.

Un adecuado proceso de ingeniería de requisitos tiene implicaciones en la calidad del producto final, por ende, en la satisfacción del cliente. Es por ello, que el proceso de IR tiene que estar bien definido y ser desarrollado de forma disciplinada para repetir los éxitos anteriores en nuevos proyectos. Existen en la literatura diferentes modelos de IR que deben ser valorados para su adopción en una organización.

1.2 Modelos de procesos de ingeniería de requisitos

En este epígrafe se realiza un análisis crítico de los modelos más representativos reportados en la literatura científica.

1.2.1 Modelo de Pohl

El modelo de Pohl [Pohl, 1997] es un modelo iterativo que define cuatro actividades, las que aparecen en la figura 1.8, cuyo orden de realización puede ser cualquiera (cualquier tipo de ciclos).

En [Pohl, 1997] se asume una secuencia en la que los requisitos son elicitados, a continuación son negociados entre los participantes, se integran con el resto de la documentación y finalmente se validan y verifican para asegurar que corresponden con las necesidades reales de los clientes y usuarios y que no presentan conflictos con los demás requisitos.

El análisis crítico de las características de estas actividades, realizado por la autora, muestra las debilidades del modelo y se exponen a continuación.

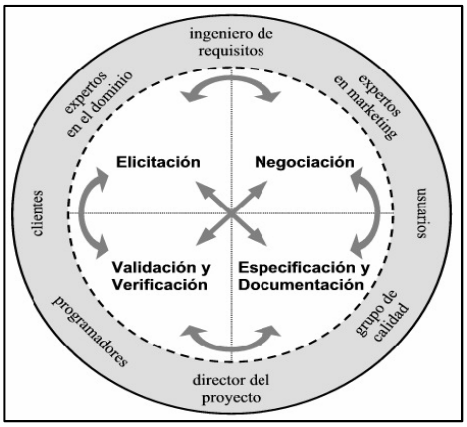


Figura 1.8 Modelo de procesos de ingeniería de requisitos de Pohl.

En la *elicitación de requisitos*, para Pohl, el objetivo es hacer explícito el conocimiento oculto sobre las necesidades de clientes y usuarios y el sistema a desarrollar, de forma que todos los participantes en el problema sean capaces de entenderlo.

Asumiendo las dimensiones de la ingeniería de requisitos propuestas también por Pohl, esta actividad haría avanzar el proceso en la dimensión de compleción, ya que incrementa el conocimiento sobre el sistema a desarrollar. Obsérvese la figura 1.8.

Sin embargo, es una insuficiencia en esta actividad que se asuma sólo de forma implícita la importancia de la realización de tareas tales como la identificación de las fuentes de información, el mejor conocimiento del dominio del problema, la reutilización de especificaciones de requisitos similares y la utilización combinada de las técnicas habituales de elicitación, dejando esto a consideración de las personas que desarrollan el trabajo. Lo anterior puede traer consigo que el nivel de completitud desde la primera iteración sea bajo y por consiguiente, se produzca un incremento de tiempo total de la ejecución del proceso.

En la *negociación de requisitos*, según Pohl, el objetivo es alcanzar acuerdos entre todos los participantes sobre los requisitos elicitados en la actividad anterior, avanzando en la dimensión de acuerdo del proceso. Para ello propone tener en cuenta los factores siguientes:

1. Hacer explícitos los conflictos y evitar los conflictos emocionales entre los participantes, de forma que quede claro qué es lo que se negocia y que dicha negociación no se vea afectada por motivos personales.
2. Hacer explícitos para cada conflicto las alternativas, las argumentaciones y las razones subyacentes que los provocan, de forma que la negociación pueda basarse en las raíces del conflicto.

3. Asegurarse de que se toman las decisiones correctas, de forma que la mayoría de los participantes estén de acuerdo en los resultados de la negociación y no se sientan desplazados del proceso.
4. Asegurarse de involucrar a las personas adecuadas en el momento adecuado, para evitar tener que volver a replantear las negociaciones porque alguno de los participantes afectados no participó en las negociaciones oportunas.

Una carencia de la propuesta de Pohl en esta actividad es que no considera el análisis de los requisitos elicitados anteriormente, la autora considera que la omisión de tareas tan importantes, como la clasificación y la modelación obstaculizan la obtención de requisitos de calidad y entorpecen la negociación, por tanto el avance en la dimensión de de acuerdo del proceso pude prolongarse por más tiempo.

En la actividad *especificación/documentación* deben documentarse los requisitos elicitados y negociados, para lo que Pohl propone que no se utilice una sola notación, sino tantas como sea necesario para que todos los participantes los entiendan, avanzando en la dimensión de formalidad del proceso. En esto coincide con otros autores como [Davis, 1995] o [Sawyer y Kontoya 1999].

Los propósitos de la actividad de *validación/verificación de requisitos* son comprobar que los requisitos documentados corresponden con las necesidades de los clientes y usuarios (validación) y comprobar que la especificación cumple con los requisitos especificados (verificación), haciendo avanzar el proceso en las tres dimensiones descritas en [Pohl, 1994].

Para los aspectos de validación puede recurrirse a prototipos, mientras que para la verificación pueden utilizarse verificaciones formales u otro tipo de técnicas.

Dadas las tendencias actuales de la ingeniería de software en general y de la de requisitos en particular, la autora considera que este modelo tiene tres carencias evidentes. Primeramente, la ausencia de la actividad de gestión de requisitos, cuya importancia para los resultados del proyecto están claramente definidos en [Leffingwell y Widrig, 2003]. Por otro lado, no considera la aparición y el manejo de los cambios, lo que constituye una de las fundamentales causas de fracasos de proyectos de software [TSG, 1995]. Finalmente, no tiene en cuenta la inevitable interacción con el resto del ciclo de vida del proyecto y su repercusión en este proceso, como ha sido planteado por Amador Durán [Durán, 2002].

1.2.2 Modelo en espiral

El modelo espiral [Sawyer, et al., 1997], [Sommerville y Sawyer, 1997], representado gráficamente en la figura 1.9, está basado en el modelo espiral de Boehm para la ingeniería de requisitos [Boehm, et al., 1994] y en el Inquiry–Based Model de Potts [Potts, et al., 1994b], [Potes, et al., 1994a]. En

este modelo se define una naturaleza iterativa del proceso y la dificultad de establecer un punto de terminación del mismo, dado que los requisitos nunca llegarán a ser perfectos. Los ciclos han de tener siempre la forma *elicitación – análisis / validación – negociación.*

Aparte de las tres actividades que se analizan a continuación, el modelo asume que existe una cuarta, la gestión de requisitos, que se realiza durante todo el proceso y que se encarga de gestionar la obtención incremental de los requisitos y los inevitables cambios a los que están sujetos, lo que significa una mejora sustancial con relación al modelo de Pohl, pero, al igual que este, tampoco vincula el proceso de la ingeniería de requisitos con el resto del ciclo de vida del proyecto.

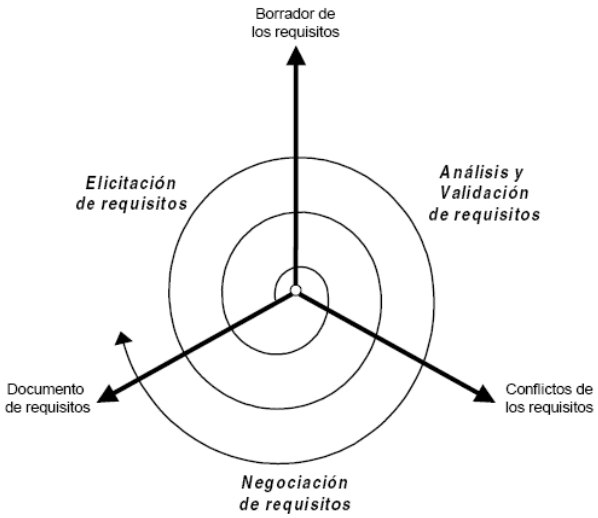


Figura 1.9 Modelo espiral de procesos de ingeniería de requisitos.

En la actividad de *elicitación de requisitos,* distintas fuentes de información como clientes, usuarios, expertos en el dominio, etc. son consultadas para entender el problema y establecer los requisitos del sistema a desarrollar.

En este modelo se presupone que los requisitos elicitados puede que no sean completos y que pueden estar expresados de forma vaga o no estructurada, lo que la autora considera un precedente negativo, ya que el impacto psicológico de estas asunciones sobre las personas involucradas puede conducir al facilismo, la superficialidad y falta de esfuerzo en el logro de mejores resultados, lo que redundaría en el incumplimiento de los plazos de tiempo del proceso y del proyecto.

Durante el *análisis y validación*, los requisitos elicitados se integran y analizan, lo que suele provocar la identificación de requisitos que faltan, inconsistencias y conflictos entre los mismos, pero tampoco se alude explícitamente a tareas como la clasificación y el modelado de los requisitos.

La autora considera que en el modelo espiral hay un aporte al definirse abiertamente el análisis de los requisitos elicitados como actividad fundamental. Sin embargo, aunque en [Sawyer, et al., 1997]

se denomina a esta actividad Análisis y validación de requisitos, no se hace referencia a ningún tipo de tarea de validación, tal como se entiende el término en esta tesis, es decir, la confirmación por parte de clientes y usuarios de que los requisitos reflejan realmente sus necesidades y especifican el sistema que ellos desean.

En la actividad de *negociación de requisitos*, los conflictos identificados durante el análisis deben resolverse llegando a acuerdos entre los participantes en el proceso, para lo que suele ser necesario descubrir nueva información.

Consideramos una debilidad del modelo que la actividad de negociación suceda a la de validación. La negociación no implica en sí misma la validación, por tanto cada cambio acordado debería ser nuevamente validado por los clientes y usuarios, para impedir que se produzcan discrepancias en fases posteriores del proyecto, que conlleven retrasos, gastos adicionales y hasta fracasos.

1.2.3 Modelo Iteración de Actividades propuesto por DURÁN

Esta propuesta [Durán, 2000], conceptualmente basada en el modelo de Pohl y representado en la figura 1.10, consta de tres actividades principales, *elicitación, análisis y validación*, organizadas en tres ciclos: dos internos al proceso y uno de externo, y su principal característica es la iteratividad. Su aporte fundamental es la vinculación del proceso de ingeniería de requisitos con el resto del proceso de desarrollo del software.

El ciclo principal del modelo es el de *elicitación-análisis-validación*, que es un ciclo interno e indica la posibilidad que durante el proceso de validación de los requisitos por parte de los clientes y usuarios aparezcan conflictos o nuevos requisitos que hasta entonces estaban ocultos. En esas circunstancias, es necesario resolver dichos conflictos y consensuar los nuevos requisitos mediante nuevas reuniones de elicitación/negociación, repitiendo a continuación las actividades de análisis y validación.

El segundo ciclo interno, *elicitación-análisis*, es un ciclo que indica la posibilidad que durante la realización del análisis de los requisitos elicitados se descubran conflictos o deficiencias en dichos requisitos, lo que puede provocar la necesidad de nuevas reuniones de elicitación- negociación y el posterior análisis de sus resultados.

El tercer ciclo, entre la ingeniería de requisitos y el resto del desarrollo, muestra la posibilidad de que durante el resto del desarrollo sea necesario volver a alguna de las actividades de ingeniería de requisitos, posiblemente porque se detecte la necesidad de renegociar algunos requisitos de difícil implementación, porque aparezcan nuevos requisitos durante el desarrollo, etc.

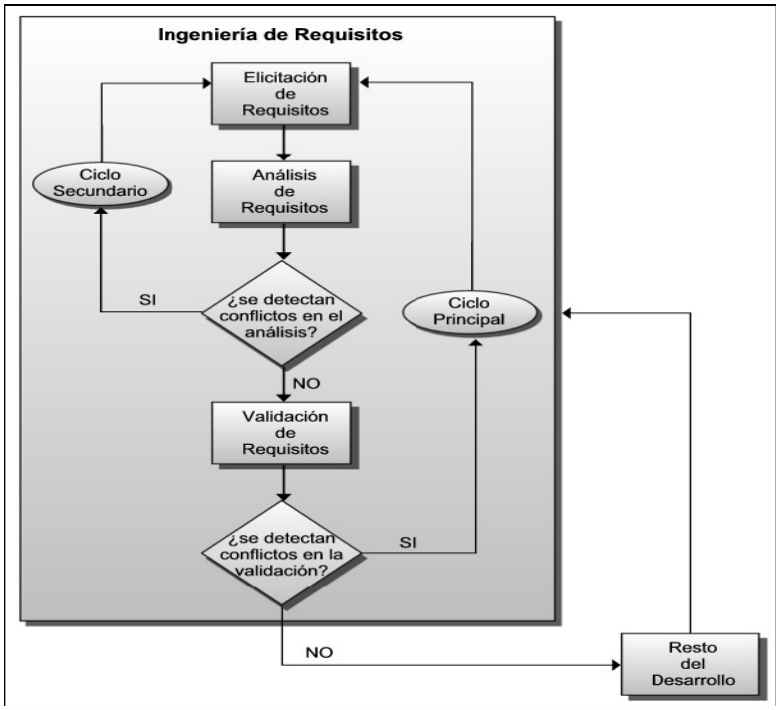


Figura 1.10 Modelo de Iteración de Actividades, propuesto por Durán.

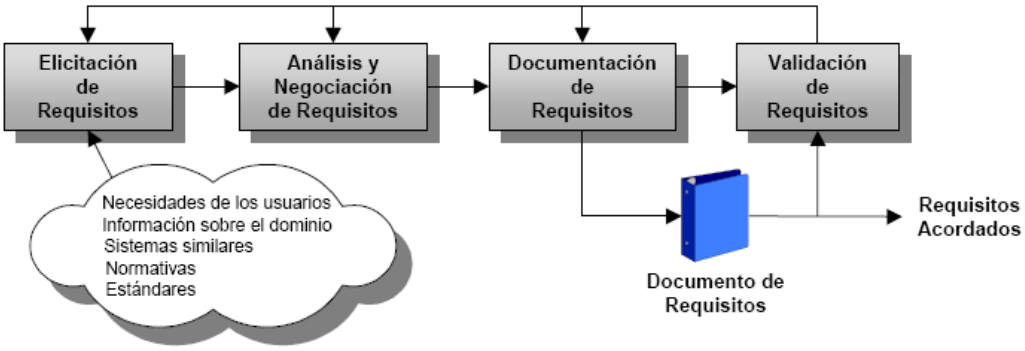


Figura 1.11 Modelo SWEBOK de procesos de ingeniería de requisitos.

La actividad de *elicitación*, considerada por el propio autor [Durán, 2000] como la más destacada, es la que mantiene la interacción entre clientes y usuarios con los ingenieros de requisitos. Sus principales objetivos son conocer el dominio del problema, descubrir las necesidades reales de clientes y usuarios y consensuar los requisitos entre estos.

La actividad de *análisis*, que tradicionalmente ha sido considerada como la más importante, tiene como objetivos principales detectar conflictos en los requisitos orientados al cliente (requisitos–C), profundizar en el conocimiento del dominio del problema y establecer las bases del diseño del sistema. En esta actividad Durán no potencia la modelación de los requisitos, que juega un importante papel en su comprensión y validación por los clientes y usuarios, y que constituyen la base para la próxima etapa del proyecto: el diseño.

En la actividad *validación* se debe confirmar que los requisitos–C, una vez analizados y resueltos los posibles conflictos, corresponden realmente a las necesidades de clientes y usuarios. Asegurarse de que los requisitos describen el producto deseado es el principal objetivo de esta actividad.

A diferencia de los dos modelos anteriormente analizados, en el modelo de Durán se definen como salidas de este requisitos–C, requisitos–D, prototipo y conflictos.

La autora considera que Durán se contradice al plantear los conflictos como una salida, ya que en su modelo se asume que los conflictos aparecerán principalmente durante la actividad de análisis y se resolverán mediante negociación en las actividades de elicitación, que forman parte de los ciclos internos del modelo. Esto no demerita su planteamiento de que es importante registrar los conflictos que vayan surgiendo, para poder acometer su resolución de forma organizada, involucrando a los participantes en el proceso de negociación.

1.2.4 Modelo SWEBOK

El proyecto SWEBOK (Software Engineering Body of Knowledge) es un proyecto conjunto del IEEE y de la ACM para producir un cuerpo de conocimiento sobre ingeniería de software que siente las bases de dicha ingeniería como una profesión [Bourque et al. 1999].En [Sawyer y Kantoya, 1999] se propone el modelo de procesos que puede verse en la figura 1.11.

La actividad de *elicitación de requisitos* es la primera que debe ser realizada para lograr entender los problemas que hay que resolver. Esta es, fundamentalmente, una actividad humana, por lo que, para acometerla con ciertas garantías, es necesario tener en cuenta los puntos siguientes:

- *Los objetivos:* pueden considerarse como los requisitos de alto nivel que deberá cumplir el sistema a desarrollar, por lo que es especialmente crítica su identificación en los comienzos del proceso.

- *Conocimiento del dominio:* es fundamental conocer el dominio del problema para poder inferir el conocimiento tácito que los participantes no suelen hacer explícito, además de para facilitar la comunicación.
- *Implicados (stakeholders):* es preciso identificar a todos los participantes que tengan algún tipo de interés en el desarrollo como clientes, usuarios, desarrolladores, analistas de mercado, proveedores, entre otros.
- *Entorno operacional:* el sistema a desarrollar deberá funcionar en un entorno que es necesario conocer para poder identificar las necesidades de interoperabilidad con otros sistemas ya existentes.
- *Entorno organizacional:* además del entorno operacional, el sistema afectará al entorno organizacional, que es necesario conocer para evitar que surjan problemas con los procesos de negocio.

Para la realización de esta actividad se puede recurrir a las técnicas de elicitación existentes y descritas en [Davis, 1993], [Goguen y Linde 1993], [Kontoya, 2000].

En la actividad de *análisis y negociación de requisitos* se pretende detectar y resolver los conflictos entre los requisitos, determinar los límites del sistema y cómo interactuará con su entorno, además de transformar los requisitos de usuarios en requisitos software.

Las tareas que a continuación se describen, propuestas en el modelo para esta actividad, son una importante contribución para el logro de mayor calidad del proceso.

Clasificación de requisitos

Los autores consideran importante clasificar los requisitos para ayudar en las labores de negociación. Los criterios de clasificación son: funcionales/ no funcionales, capacidad/restricción, prioridad, coste/impacto, volatilidad/estabilidad y requisito de producto/requisito de proceso.

Modelado conceptual

El objetivo del modelado conceptual es ayudar a la comprensión del problema, aunque normalmente no puede evitarse iniciar el diseño de la solución. Existen múltiples técnicas de modelado conceptual, por lo que los autores proponen considerar factores como la naturaleza del problema, la experiencia del ingeniero de requisitos en el uso de una determinada técnica, si el cliente impone como requisito la utilización de una determinada metodología o la disponibilidad de metodologías y herramientas que soporten una determinada notación.

Independientemente de la notación de modelado utilizada, los autores proponen realizar siempre un modelo de los límites del sistema para entender mejor el entorno y las interfaces necesarias.

Negociación de requisitos

La negociación de requisitos, también denominada resolución de conflictos, se ocupa de resolver los problemas que puedan surgir en los requisitos, bien porque haya peticiones por parte de clientes y usuarios que sean incompatibles, bien porque no se disponga de los recursos necesarios para la realización de ciertos aspectos del sistema, etc.

Para resolver estos conflictos es necesario consultar con todos los participantes afectados y registrar las decisiones tomadas y quién las tomó. Es necesario añadir que los autores asumen que los conflictos pueden aparecer no sólo durante el análisis, sino también durante la validación.

Una ventaja de este modelo es que aborda integralmente y con mayor grado de detalle la actividad de análisis y negociación.

En la actividad de *documentación de requisitos,* los documentos de requisitos son el medio habitual para el registro y la comunicación de estos. Los autores consideran deseable que los requisitos–C y los requisitos–D vayan en documentos separados.

Dentro de las tareas relacionadas con la documentación, los autores se remiten a la gestión de requisitos para aspectos como el control de versiones debido a los cambios.

La actividad de *validación de requisitos* debe comprobar los documentos de requisitos para revelar omisiones, conflictos y ambigüedades no detectadas en el análisis. También se debe comprobar que los requisitos siguen las normas de calidad establecidas. Los autores combinan validación y verificación en una sola actividad, mediante el uso de revisiones técnicas de los requisitos basadas en listas de comprobación, el uso de prototipos, verificación formal de especificaciones, etc.

La *gestión de requisitos*, aunque no está reflejada en la figura 1.11, se encuentra implícita, pues el modelo general de SWEBOK considera la gestión en sus diez áreas de conocimiento [SWEBOK, 2004], además de definir algunos procedimientos para controlar los cambios y utilizar técnicas de gestión de configuración.

En el área de Requisitos del Software de SWEBOK, los autores realizan importantes consideraciones prácticas con relación a la gestión de los cambios, el rol de los atributos de los requisitos, la relevancia de la trazabilidad para el análisis de impacto cuando se produce un cambio y, finalmente, acerca de las mediciones en el proceso, que no han sido abordadas en otros modelos.

El tamaño de la guía para la base de conocimiento de la ingeniería de software es una limitación para la utilización de este modelo, por cuanto lo describe en una de las diez áreas que recoge. Por otro lado, la guía, en su concepción, trata la gestión del cambio en los requisitos como una consideración práctica y no como parte del proceso de IR, siendo recogida en la gestión de la configuración (otra de las áreas). Además, los aspectos concretos relacionados con el análisis de

riesgos los desarrolla en el área de gestión de la ingeniería de software. El tratamiento de los riesgos desde las primeras etapas de la fase de ingeniería de requisitos daría un carácter proactivo al modelo, dando mayores posibilidades de éxito al proyecto

Esta dispersión de actividades relacionadas intrínsecamente con un mismo proceso, en una Guía útil, pero voluminosa, trae como consecuencia falta de integración de todas las actividades a desarrollarse en la IR, lo que desestimula su uso y dificulta el manejo por los responsables, sobre todo en pequeñas y medianas empresas.

1.2.5 Otros trabajos sobre procesos de ingeniería de requisitos

Existen otros trabajos cuyo principal objetivo es aportar metodología que permita desarrollar la Ingeniería de los Requisitos, y que no se mencionan de manera tan explícita por su menor influencia en nuestra investigación.

En este grupo destacamos el ***Método KAOS*** *(Knowledge Acquisition in Automated Specification of Software)* [Lamswererde, 1991]. Se trata de un método orientado por objetivos (goal oriented) con el que se elaboran constructivamente los requisitos del sistema a partir de los objetivos de más alto nivel.

El método consiste en la identificación de objetivos para realizar sucesivas re-identificaciones en subobjetivos hasta que cada uno de estos subobjetivos puede asignarse a agentes primarios como personas, dispositivos o software. Los objetos referidos por los objetivos se identifican gradualmente a partir de la propia definición de los mismos.

El método apoya la exploración de los refinamientos alternativos para cada objetivo, así como de las asignaciones a los agentes primarios, obteniendo así alternativas del sistema en las que el límite entre el sistema automatizado y su ambiente puede ser absolutamente diferente. También apoya la identificación y la resolución de conflictos entre objetivos, y la identificación y la resolución de comportamientos excepcionales (obstáculos) de los agentes que vulneran los objetivos y las responsabilidades acordadas durante el proceso del refinamiento de cada objetivo.

Un trabajo interesante directamente relacionado con el ámbito de la ingeniería de los requisitos es la ***Metodología de Definición de Requisitos EasyWinWin*** [Bohem, 2001], que está relacionada con el modelo de desarrollo evolutivo WinWin [Bohem, 1998].

EasyWinWin, que no es considerado como un modelo de proceso, define un conjunto de actividades que guían a los implicados del sistema interactivo a desarrollar a través de un proceso de recolección, elaboración, priorización y negociación de requisitos.

Esta metodología utiliza técnicas participativas de grupos relacionadas con el trabajo colaborativo o cooperativo.

Finalmente, vinculado al modelo CMM [Paulk et al., 1993] de mejoramiento de procesos de software está el ***Modelo de Madurez de Proceso de Ingeniería de Requisitos REAIMS*** [Sawyer, et al., 1997], [Sommerville y Sawyer, 1997]. Este modelo define tres niveles de madurez que se corresponden con los tres primeros niveles del CMM (inicial, repetido y definido). Durante la realización del proyecto REAIMS se constató que la mayor parte de las empresas europeas estudiadas estaban en un nivel de madurez inicial.

En REMAIS también se define una guía para organizaciones sin ningún tipo de proceso de ingeniería de requisitos que incluye:

- Definir una estructura normalizada del documento de requisitos.
- Hacer el documento fácil de cambiar.
- Identificar de manera única cada requisito.
- Definir políticas para la gestión de requisitos.
- Definir plantillas normalizadas para la descripción de requisitos.
- Usar el lenguaje de forma simple, consistente y concisa.
- Organizar revisiones formales de los requisitos.
- Definir listas de comprobación para la validación.
- Usar listas de comprobación para el análisis de los requisitos.
- Planificar los conflictos y su resolución.

Partiendo de los análisis anteriores y otras fuentes consultadas, [Kontoya, 2000], [Pressman, 2002], [Sommerville, 2005], podemos establecer que todos los modelos:

1. Abordan la *ingeniería de requisitos como un proceso iterativo* e incorporan el hecho de que no puede realizarse de forma lineal. La autora coincide con este enunciado y considera que esto ocurre por ser este, básicamente, un proceso humano, de descubrimiento y comunicación.
2. Se proyectan como un *proceso incremental:* como resultado de cada iteración se producen cambios, mejoras, incrementos, que aumentan el ritmo del proceso y que permite realizar liberaciones parciales de versiones de Especificaciones de Requisitos. Esto, además de reducir los plazos, logra mejores niveles de satisfacción de los clientes, al constatar el avance del proyecto con los incrementos sistemáticos.

3. *Tienen dificultades para establecer los límites de las actividades*, aunque disponer de un modelo de procesos es siempre beneficioso. Por ejemplo, es posible comenzar a construir modelos (actividad del análisis) durante las sesiones de elicitación, e incluso si la experiencia en el dominio del problema así lo aconseja, podrían validarse requisitos en dichas sesiones. También es habitual que durante las sesiones de validación, sobre todo usando prototipos, se eliciten nuevos requisitos que hasta entonces estaban ocultos.
4. *No tienen claramente definidos los productos del proceso:* la mayor parte de los modelos de procesos propuestos, excepto los de Iteración de Actividades y SWEBOK, no definen claramente los productos o salidas que se supone que se debe obtener como realización de dichos procesos.
5. *Se definen independiente de la metodología de desarrollo.*

1.3 Técnicas que se utilizan en las actividades de IR

El proceso de descubrimiento de requisitos carece aún de la sistematización formal que permita su planificación. La mayoría de los profesionales utiliza dos o tres técnicas que son las que conoce o que popularmente se aplican en esta etapa. Sin embargo, existe una gran cantidad de técnicas que provienen de otras ciencias, como la sicología o la etnografía, que han sido exitosamente probadas y utilizadas en la captura de requisitos. Esta brecha entre la investigación y la práctica ha merecido especial atención, siendo su eliminación el objetivo principal de diversos eventos y publicaciones creados en los últimos años.

Técnicas para la elicitación de requisitos

El proceso de captura de requisitos puede resultar difícil, principalmente si el entorno de trabajo es desconocido para los analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa. [Goguen y Linde, 1993], [Escalona y Koch, 2003].

A continuación se presentan un grupo de técnicas que de forma clásica han sido utilizadas para esta actividad en el proceso de desarrollo de todo tipo de software.

Entrevista: resulta una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. La entrevista le permite al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Existen muchos tipos de entrevistas y son muchos los autores que han trabajado en definir su estructura y dar guías para su correcta realización [Durán, et al., 1999], [Pan, et al., 2001]]. Sin embargo, la entrevista no es una técnica sencilla de aplicar [Pan,

et al., 2001], requiere de la habilidad y experiencia del entrevistador para obtener toda la información posible en un período de tiempo siempre limitado.

JAD (Joint Application Development/Desarrollo conjunto de aplicaciones): esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes [IBM, 1997]. Está basada en cuatro principios fundamentales dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación WYSIWYG (What You See Is What You Get, lo que ve es lo que obtiene), es decir, durante la entrevista se trabajará sobre lo que se generará. Tras una fase de preparación del JAD al caso concreto, el equipo de trabajo se reúne en varias sesiones. En cada una de ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación. Durante la sesión se discute en grupo sobre estos temas llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema. Esta técnica presenta una serie de ventajas frente a las entrevistas tradicionales ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado. Pero requiere un grupo de participantes bien integrados y organizados.

Brainstorming (Tormenta de ideas): es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre [Raghavan, et al., 1994]. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo debe tener como máximo 10 participantes, una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador. Como técnica de captura de requisitos es sencilla de usar y de aplicar, contrariamente al JAD puesto que no requiere tanto trabajo en grupo como éste. Además suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del sistema, por lo que suele aplicarse en los primeros encuentros.

Concept Mapping: Los mapas conceptuales [Pan, et al., 2001] son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la ingeniería de requisitos pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste. Sin embargo, deben ser usados con cautela porque en algunos casos pueden ser muy sugestivos y pueden llegar a ser ambiguos en casos complejos si no se acompaña de una descripción textual.

Sketches y Storyboards: Está técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno Web. La misma consiste en representar sobre papel en forma muy

esquemática las diferentes interfaces al usuario (sketches). Estos sketches pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación (storyboard).

Casos de Uso: Aunque inicialmente se desarrollaron como técnica para la definición de requisitos [Jacobson, 1995], algunos autores proponen casos de uso como técnica para la captura de requisitos [Pan, et al., 2001], [Liu y Yu, 2001]. Los casos de uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. La ventaja esencial de los casos de uso es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria [Vilain, et al. 2000], [Insfrán, 2002] si no se acompañan con una información textual o detallada con otra técnica como pueden ser diagramas de actividades [UML, 2001].

Cuestionarios y Checklists: Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (*checklist*). Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista.

Comparación de terminología: Uno de los problemas que surge durante la elicitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada en forma complementaria a otras para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos (contraste) [Pan, et al., 2001].

Proceso de Análisis Jerárquico (AHP): Esta técnica tiene por objetivo resolver problemas cuantitativos, para facilitar el pensamiento analítico y las métricas. Consiste en una serie de pasos, a saber: encontrar los requerimientos que van a ser priorizados; combinar los requisitos en las filas y columnas de la matriz *n x n* de AHP; hacer algunas comparaciones de los requerimientos en la matriz; sumar las columnas; normalizar la suma de las filas; calcular los promedios. Estos pasos pueden aplicarse fácilmente a una cantidad pequeña de requerimientos, sin embargo, para un volumen grande, esta técnica no es la más adecuada.

Existen otras técnicas para la captura de requisitos (análisis de otros sistemas, estudio de la documentación, etc.) incluso también es común encontrar alternativas que combinen varias de estas técnicas [Pan, et al., 2001] como:

Entrevistas vs. Casos de Uso: mucha de la información recolectada durante una entrevista, puede ser usada para construir casos de uso. Así, el equipo de desarrollo puede entender mejor el ambiente

de trabajo de los involucrados. Cuando el analista sienta que tiene dificultades para entender una tarea, puede recurrir al uso de un cuestionario y mostrar los detalles recavados en un caso de uso. De hecho, durante las entrevistas cualquier usuario puede utilizar diagramas de casos de uso para explicar su entorno de trabajo.

Entrevistas vs. Brainstorming: Muchas de las ideas planteadas en el grupo, provienen de la información recopilada en entrevistas o cuestionarios previos. Realmente la lluvia de ideas trata de encontrar las dificultades que existen para la comprensión de términos y conceptos por parte de los participantes; de esta forma se llega a un consenso.

Casos de Uso vs. Brainstorming: La lista de ideas proveniente del brainstormg puede ser representada gráficamente mediante casos de uso.

Estudio realizado por [Carrizo y Juristo, 2002] acerca de las técnicas de adquisición de requisitos demuestra que mientras más canales entre cliente y desarrollador, más éxito en el proyecto; se debe utilizar técnicas directas (sin intermediario) para reducir la distorsión o pérdida de información y que es posible seleccionar técnicas más adecuadas para cada entorno.

En la tabla 1.1 aparece un análisis comparativo entre las técnicas de elicitación de requisitos.

Sin embargo, las presentadas ofrecen un conjunto representativo de las más utilizadas y resultan suficientes para el objetivo de este trabajo.

Técnicas para la especificación de requisitos

También para la actividad definición de requisitos en el proceso de ingeniería de requisitos hay un gran número de técnicas propuestas. Describimos brevemente las más relevantes para este trabajo.

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

Glosario y ontologías: La diversidad de personas que forman parte de un proyecto software hace que sea necesario establecer un marco de terminología común. Esta necesidad se vuelve más patente en los sistemas de información web, porque el de desarrollo en ellas suele ser más interdisciplinario [Koch, 2001]. Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recogen y definen los conceptos más relevantes y críticos para el sistema. En esta línea se encuentra también el uso de ontologías, en las que no sólo aparecen los términos, sino también las relaciones entre ellos. Ninguna de las metodologías para el entorno web incluidas en este estudio comparativo por su relevancia en la ingeniería de requisitos propone la definición de una ontología.

Tabla 1.1 Análisis comparativo de las técnicas para la elicitación de requisitos.

Técnica	Ventajas	Desventajas
Entrevistas y Cuestionarios	Mediante ellas se obtiene una gran cantidad de información correcta a través del usuario. Pueden ser usadas para obtener una idea clara del dominio del problema. Son flexibles. Permiten combinarse con otras técnicas.	La información obtenida al principio puede ser redundante o incompleta. Si el volumen de información manejado es alto, requiere mucha organización de parte del analista, así como la habilidad para tratar y comprender el comportamiento de todos los involucrados.
Lluvia de Ideas (Brainstorm)	Los diferentes puntos de vista y las confusiones en cuanto a terminología, son aclarados por expertos. Ayuda a desarrollar ideas unificadas basadas en la experiencia de un experto.	Es necesaria una buena compenetración del grupo participante.
Prototipos	Ayudan a validar y desarrollar nuevos requerimientos. Permiten comprender aquellos requerimientos que no estén muy claros y que sean de alta volatilidad.	El cliente puede llegar a pensar que el prototipo es una versión del software que será desarrollado. A menudo, el desarrollador hace compromisos de implementación con el objetivo de acelerar la puesta en funcionamiento del prototipo
Análisis Jerárquico	Permite determinar el grado de importancia de cada requerimiento. Ayuda a identificar conflictos en los requerimientos. Muestra el orden en que deben ser implementados los requerimientos.	Debe construirse un estándar claro de evaluación, que incluya la participación del cliente.
Casos de Uso	Representan los requerimientos desde el punto de vista del usuario. Permiten representar más de un rol para cada afectado. Identifica requerimientos estancados, dentro de un conjunto de requerimientos.	En sistemas grandes, toma mucho tiempo definir todos los casos de uso. El análisis de calidad depende de la calidad con que se haya hecho la descripción inicial.

Plantillas o patrones: Esta técnica, recomendada por varios autores [Durán, et al., 1999], [Escalona, et al., 2002], tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado detallado, el trabajo de rellenar las plantillas y mantenerlas puede ser demasiado tedioso.

Escenarios: La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos [Liu y Yu, 2001]. La representación del escenario puede variar dependiendo del autor. Esta representación pueden ser casi textuales o ir encaminada hacia representaciones gráficas en forma de diagramas de flujo [Weidenhaupt, et al., 1999]. El análisis de los escenarios, hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales de sistema [Lowe, 1999].

Casos de uso: Es como técnica de definición de requisitos como más ampliamente han sido aceptados los casos de uso. Actualmente se ha propuesto como técnica básica del proceso RUP [Kruchten, 1998]. Sin embargo, son varios los autores que defienden que pueden resultar ambiguos a la hora de definir los requisitos [Díez, 2001], [Vilain, et al., 2000], [Insfrán, et al., 2002], por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.

Lenguajes Formales: Otro grupo de técnicas que merece la pena resaltar como extremo opuesto al lenguaje natural, es la utilización de lenguajes formales para describir los requisitos de un sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción formal, han sido aplicadas en el mundo de la ingeniería de requisitos desde hace años [Peña, 1998]. Sin embargo, resultan muy complejas en su utilización y para ser entendidas por el cliente. El mayor inconveniente es que no favorecen la comunicación entre cliente y analista. Por el contrario, es la representación menos ambigua de los requisitos y la que más se presta a técnicas de verificación automatizadas.

Técnicas para la validación de los requisitos

La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea [Lowe, 1999]. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de

requisitos con el usuario para detectar errores o inconsistencias. Aún así, existen algunas técnicas que pueden aplicarse para ello:

Reviews o Walk-throughs: Está técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar consistencia de la documentación o información faltante.

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

Matrices de trazabilidad: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo [Durán, et al., 1999].Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.

Prototipos: Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario [Olsina, 1999]. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final.

Las técnicas que pueden ser utilizadas en las diferentes actividades de la IR aparecen en la tabla 1.2.

Tabla 1.2 Técnicas utilizadas por actividad en la ingeniería de requisitos.

	Elicitación	Análisis y negociación	Especificación de Requisitos	Validación	Evolución
Entrevistas/Cuestionarios	X				X
Tormenta de Ideas	X	X			X
Prototipos				X	
Análisis Jerárquico		X			X
Casos de Uso	X		X		X

1.4 La medición en el proceso de ingeniería de requisitos

Para gestionar la calidad de los procesos de software, independiente del estándar que se adopte, es necesaria la gestión de los procesos, las mediciones, el análisis y la mejora.

Lo principal de este concepto es que una alta calidad del software, entre otras cosas, puede llegar a ser consistente si se miden constantemente los procesos mejorando su calidad.

La medición es algo común en la ingeniería, sin embargo, no es así en el contexto de la ingeniería de software [García, 2000] y menos aun en el ámbito de la ingeniería de requisitos. Existen problemas al ponerse de acuerdo en qué medir y al evaluar las métricas recopiladas.

La autora, parte de la premisa que la calidad de la especificación de requisitos es un elemento crucial para la buena marcha del proyecto de desarrollo de software y para la calidad del producto final.

La Guía del Cuerpo de Conocimiento para la Ingeniería de Software recomienda realizar mediciones sobre los requisitos, concretamente del volumen de estos en proyectos particulares. Este número podrá ser utilizado para evaluar el "tamaño" de los cambios en los requisitos, estimar los costos en las tareas de desarrollo y mantenimiento [SWEBOK, 2004].

En [Monzón, 2001] se limita el concepto clásico de medida a la contabilización de ciertos parámetros considerados relevantes en el ámbito de una especificación de requisitos, considera necesario extender esta contabilización a una serie de valores estadísticos derivados de algunos de los parámetros propuestos. Este autor define que una "métrica" será todo aquel valor numérico, deducido de la especificación, que permita tomar decisiones acerca de su desarrollo o evolución futura.

Monzón asume como medidas "clásicas" de la calidad de la especificación (como conjunto de requisitos) las propiedades de ésta, como corrección, no ambigüedad, completitud, consistencia, trazabilidad y otras. Además clasifica las métricas en absolutas y relativas.

Las métricas absolutas son valores globales de la especificación que miden el volumen de ésta y las métricas relativas son parámetros que indican las relaciones existentes entre diferentes elementos de especificación. Se asume que las métricas absolutas proporcionan una idea del tamaño de la especificación y las relativas proporcionan una medida del acoplamiento entre elementos.

En definitiva, los conceptos vinculados con la calidad de una especificación tratados en este trabajo proporcionan una buena base teórica para abordar de forma efectiva la medición de la especificación, pero no indican cómo llevarla a cabo.

Bernárdez, Durán y Toro, realizaron una propuesta en [Bernardez, et al., 2004], donde describen un conjunto de heurísticas para verificación de requisitos basadas en métricas, concretamente para requisitos funcionales especificados mediante casos de uso, con el objetivo de identificar aquellos que potencialmente puedan contener defectos, basándose para esto en los valores de ciertas métricas que pueden calcularse de forma automática mediante herramientas de gestión de requisitos como REM [Durán, 2004]. Si el valor de estas métricas está fuera del rango habitual establecido en las

heurísticas correspondientes, un caso de uso se considera como potencialmente defectuoso y debe ser revisado.

En [Kolde, 2004] se presenta un conjunto sencillo de métricas relacionadas con los requerimientos del proyecto software, que pueden ser adoptadas como práctica en la gestión de requisitos, en las diferentes actividades del proceso de IR y a través del resto del ciclo de vida del proyecto. Independientemente del proceso de software seguido, la utilización de métricas relacionadas con los requisitos, ayudaría al jefe del proyecto a manejar adecuadamente el cambio durante el proyecto, evitar retrabajo y atrasos, por ende, a controlar mejor tanto la duración como los costos mientras dure el proyecto. Las medidas se pueden elegir según las necesidades de la gestión de requisitos de los proyectos.

Las métricas identificadas se pueden agrupar en tres categorías. Primero, las métricas que ayudan a evaluar la calidad del proceso de ingeniería de requisitos en si mismo. Segundo, las que proporcionan al jefe del equipo de trabajo información objetiva acerca del estado del proyecto, que lo ayude a conducirlo a una terminación exitosa. Finalmente, las métricas que ayudan a evaluar el impacto de la gestión de los requerimientos a lo largo de todo el proceso de desarrollo, tanto en el costo del proyecto como en la calidad del producto.

La autora considera que la aproximación propuesta por [Kolde, 2004] es sencilla y fácil de implementar.

1.5 Ingeniería de requisitos asistida por computadora (CASE)

En la actualidad, la ingeniería de software cuenta con una serie de herramientas automatizadas destinadas a diferentes propósitos. Dentro de las herramientas CASE que sirven de apoyo a estos procesos, están las especializadas en la administración de requisitos. Estas se concentran en capturar requerimientos, administrarlos y producir una especificación de requisitos.

Las ventajas que nos proporcionan las herramientas automatizadas para IR son:

- Permiten un mayor control de proyectos complejos.
- Permiten reducir costos y retrasos en la liberación de un proyecto.
- Permiten una mayor comunicación en equipos de trabajo.
- Ayudan a determinar la complejidad del proyecto y esfuerzos necesarios.

Dos de las herramientas más utilizadas para este propósito son RequisitePro y DOORS.

RequisitePro(R) es la herramienta que ofrece Rational Software para tener mayor control sobre los requerimientos planteados por el usuario, los técnicos y los cambios que surjan durante el ciclo de vida del proyecto.

Con RequisitePro los requisitos se encuentran documentados bajo un esquema organizado de documentos; estos esquemas cumplen con los estándares exigidos por IEEE, ISO, SEI, CMM y por el Rational Unified Process.

Este CASE se integra con aplicaciones para la administración de cambios y con herramientas de modelado de sistemas y de pruebas. Esta integración asegura que los diseñadores conozcan los requerimientos del usuario, del sistema y del software en el momento de su desarrollo. También permite el desarrollo en equipo, el check-in y check-out de los documentos involucrados en el proyecto. Con esta integración, se pueden conservar juntos todos los requisitos y ser manipulados por todos los miembros del equipo.

Para RequisitePro los atributos de los requisitos son la principal fuente de información para ayudar a planear, comunicar y rastrear las actividades a través del ciclo de vida. Cada proyecto tiene necesidades únicas y se deberán seleccionar las propiedades críticas para asegurar su éxito: prioridad de desarrollo, estado, autor, responsable, relaciones, fecha de registro, fecha última modificación, versión, etc. Esta permite la asignación de prioridades a los requisitos en base a:

Beneficios al cliente: Todos los requerimientos no son desarrollados de igual forma. Se les da prioridad a partir de la importancia relativa del usuario final, basado en un análisis previo de los analistas y desarrolladores del equipo.

Esfuerzo: Claramente, algunos requisitos o cambios a éstos demandan más tiempo y recursos que otros. Estimar esfuerzo necesario para cada requerimiento es la mejor forma de determinar que puede o no ser desarrollado en el tiempo estipulado.

Una característica importante es que la curva de aprendizaje de RequisitePro es pequeña, este puede ser basado en el uso de tutoriales.

Beneficios de RequisitePro

- Permite el trabajo en equipo por medio de un repositorio compartido de información.
- Facilita la clasificación de requisitos, de acuerdo con las necesidades de usuarios, técnicos, comunicación, pruebas.
- Define atributos para todos los tipos de requisitos especificados.
- Ayuda a manipular el alcance del proyecto mediante la asignación de prioridad de desarrollo a cada uno de los requerimientos planteados.

- Características avanzadas de rastreo por medio de matrices, que permiten visualizar las dependencias entre requerimientos dentro de un proyecto o en diferentes proyectos.
- Marcas que automáticamente indican cuándo un requisito es impactado por cambios a otro requerimiento o a atributos asociados.
- Administración de cambios mediante el rastreo y la visualización histórica de los efectuados, cuándo se ejecutaron y quién los realizó.
- Manejo de plantillas creadas por el usuario, o creadas por otras empresas.
- Recolección de requerimientos mediante su importación por medio de Wizards para obtenerlos automáticamente de fuentes externas, incluyendo archivos o bases de datos.
- Interactúa con los demás productos Rational para el ciclo de vida, así como con herramientas de Microsoft Office.
- Ayuda a determinar, en forma automatizada, cuántos requisitos tiene el proyecto, sus responsables y actores.
- RequisitePro, permite organizar los requerimientos, establecer y mantener relaciones padre/hijo entre ellos.

DOORS(R) es la herramienta de administración de requisitos creada por Quality Systems and Software. Esta permite capturar, relacionar, analizar y administrar un rango de información para asegurar el cumplimiento del proyecto en materia de requerimientos.

DOORS permite el acceso de un gran número de usuarios concurrentes en la red, manteniendo en línea un gran número de requerimientos así como su información asociada. Esta ayuda al usuario a procesar las solicitudes de cambios en línea. Permite realizar cualquier modificación vía remota cuando la base de datos está off-line, incorporando sus actualizaciones a la base de datos maestra. Esto hace más fácil la comunicación del equipo con otras organizaciones, subcontratistas y proveedores.

Esta herramienta proporciona rastreabilidad multi-nivel para aquellas relaciones entre requerimientos que poseen gran tamaño. DOORS cuenta con un Wizard que le permite generar enlaces a reportes de muchos niveles, para desplegarlos en la misma vista.

Beneficios de DOORS

- Análisis, comparación y clasificación de requerimientos.
- Interpretación manual de cada requisito.
- Identificación de inconsistencias.

- Operación vía batch.
- Permite compartir requerimientos entre proyectos y crear relaciones entre ellos, mediante la táctica drag-and-drop
- Notifica, vía email, la revisión de los cambios
- Visualización de los cambios pendientes de otros usuarios para anticipar el impacto que ocasionará.
- Reporta gráficamente estadísticas y métricas.
- Los documentos están escritos en lenguaje claro, lo que proporciona una comprensión inmediata de cada requerimiento.
- Permite importar sus documentos a formatos de herramientas de Microsoft Office, RTF, HTML, texto, entre otros.
- Las plantillas presentan la información de manera estandarizada.

La autora considera que independientemente de los beneficios de estas herramientas, es necesario un nuevo instrumento que responda al proceso que se modele.

1.6 Gestión del riesgo

La gestión de riesgos en el ámbito del software procura formalizar conocimiento orientado a la minimización o evitación de riesgos en proyectos de desarrollo de software, mediante la generación de principios y buenas prácticas de aplicación realista [Roppponen, 2000]. Hasta el momento se ha propuesto y utilizado diferentes enfoques de gestión del riesgo desde que Boehm [Boehm, 1988] atrajo a la comunidad de ingeniería del software hacia la gestión del riesgo. Sin embargo, es evidente que pocas organizaciones utilizan todavía de una forma explícita y sistemática métodos específicos para gestionar los riesgos en sus proyectos software [KLCI, 2001].

En [Pressman, 2002] se presenta la definición de riesgo dada por Robert Charette en [Charette, 1989] donde plantea que en primer lugar, el riesgo afecta a los futuros acontecimientos. En segundo lugar, el riesgo implica cambios. En tercer lugar, el riesgo implica elección, y la incertidumbre que entraña esta.

Cuando se considera el riesgo en el contexto de la ingeniería de software, los tres pilares de Charette se hacen continuamente evidentes. Es indiscutible que están presentes permanentemente las características de incertidumbre (acontecimiento que caracteriza al riesgo y que puede o no ocurrir) y de pérdida (si el riesgo se convierte en una realidad ocurrirán consecuencias no deseables o pérdidas)

Están definidas las categorías de riesgos: los riesgos del proyecto, que amenazan el plan; los riesgos técnicos, que amenazan la calidad y la planificación temporal; y los riesgos del negocio, que amenazan la viabilidad del proyecto o del producto. Otra categorización a considerar a partir del conocimiento que se tenga de ellos: los riesgos conocidos (los que se descubren en las evaluaciones); los riesgos predecibles (se extrapolan de la experiencia) y los riesgos impredecibles (pueden ocurrir, pero es muy difícil identificarlos de antemano).

También son claras las estrategias frente al riesgo. Por un lado están las reactivas, cuyo método es evaluar las consecuencias del riesgo cuando este ya se ha producido (ya no es un riesgo) y actuar en consecuencia. Este tipo de estrategias acarrea consecuencias negativas, al poner el proyecto en peligro. Y por el otro las preactivas, que aplican el método de evaluación previa y sistemática de los riesgos y sus posibles consecuencias, a la par que conforman planes de contingencias para de evitar y minimizar las consecuencias. Consecuentemente, este tipo de estrategias permite lograr un menor tiempo de reacción ante la aparición de riesgos impredecibles.

La autora defiende a los partidarios de la aplicación de estrategias preactivas y coincide con [Pressman, 2002] y [Gallagher, 1999] en la necesidad de la realización de los análisis de riesgos de forma temprana, sistemática, formal y profunda.

De acuerdo con [Pressman, 2002], la administración o gestión de riesgos es un proceso iterativo que se aplica durante todo el proyecto y se desarrolla en cuatro etapas:

Identificación de riesgos: se identifican los posibles riesgos para el proyecto, el producto y el negocio y se listan.

Análisis de riesgos: se valoran las probabilidades y las posibles consecuencias de los riesgos identificados. Se ordena la lista de riesgos por la probabilidad de ocurrencia (alta, media, baja) y por el impacto que pueda tener en cuanto a su alcance y duración.

Planificación de riesgos: se crean planes para abordar los riesgos, tanto para evitarlos como para minimizar sus efectos

Supervisión de riesgos: se valora constantemente los riesgos y se revisan los planes para su mitigación tan pronto como la información de los riesgos está disponible.

Los resultados de la administración de riesgos deben ser documentados en un plan de administración de riesgos. La figura 1.12 refleja el procedimiento.

En [Gallagher, 1999] se define el paradigma de la gestión del riesgo del SEI, con un proceso de seis actividades, como se observa en la figura 1.13, que deben ser desarrolladas de forma continua a lo largo de todo el ciclo de vida del proyecto.

Otra diferencia con el proceso de gestión propuesto en [Pressman, 2002], es que en este son incluidas las actividades de vigilancia y comunicación, que mantendrá información y retroalimentación acerca del proceso de gestión de riesgos, del estado de los riesgos atenuados, vigilados y emergentes (nuevos). Considera que las fuentes de información de riesgos pueden ser internas o externas al proyecto.

La Gestión, Monitorización y Mitigación de Riesgos (RMMM) tienen como objetivo marcar las estrategias y formas de actuar del equipo de trabajo frente a los riesgos: cómo evitarlos, monitorizarlos, gestionarlos y plan de contingencia. Escuetamente podemos establecer las tareas principales de cada una

Para evitar el riesgo hay que definir las estrategias necesarias para que el riesgo no se produzca y tomar las medidas encaminadas para que, aún cuando se produzca, se minimicen sus efectos.

Para el monitoreo del riesgo hay que definir los indicadores que influyen en la probabilidad de que el riesgo se produzca y revisar periódicamente dichos factores, además de vigilar la efectividad real de las acciones encaminadas a evitar el riesgo.

La gestión del riesgo y plan de contingencia asume que la evitación y la monitorización han fallado y el riesgo se ha producido. Por ello se definen las estrategias y acciones a tomar para lograr que los efectos se minimicen. Nunca se podrá reducir a cero el coste del plan de contingencia, ya que él puede implicar algunos costes en sí mismo, por lo cual se ha de valorar el beneficio que se espera obtener de éste.

Para la autora queda claro que cuanto antes se comience el proceso de gestión de riesgos en un proyecto mayores serán las probabilidades de éxito del mismo, por tanto está convencida de que es en la etapa de la IR donde debe comenzar, lo que se dificulta por estar poco tratado el tema para las actividades que componen el proceso de IR.

1.7 Caracterización de las pequeñas y medianas empresas de software en Cuba

Las organizaciones de software se caracterizan por ser proveedores externos ya sea de software en paquete o de servicios de desarrollo. Por otro lado, existen organizaciones usuarias que tienen departamentos internos de software que producen software para estas entidades y que no se dedican a comercializarlo.

La PyME de software promedio es una empresa con una decena de clientes, y una veintena de empleados muy especializados, que tienen gran capacidad de reacción y de adaptación y un invalorable conocimiento de la cultura local.

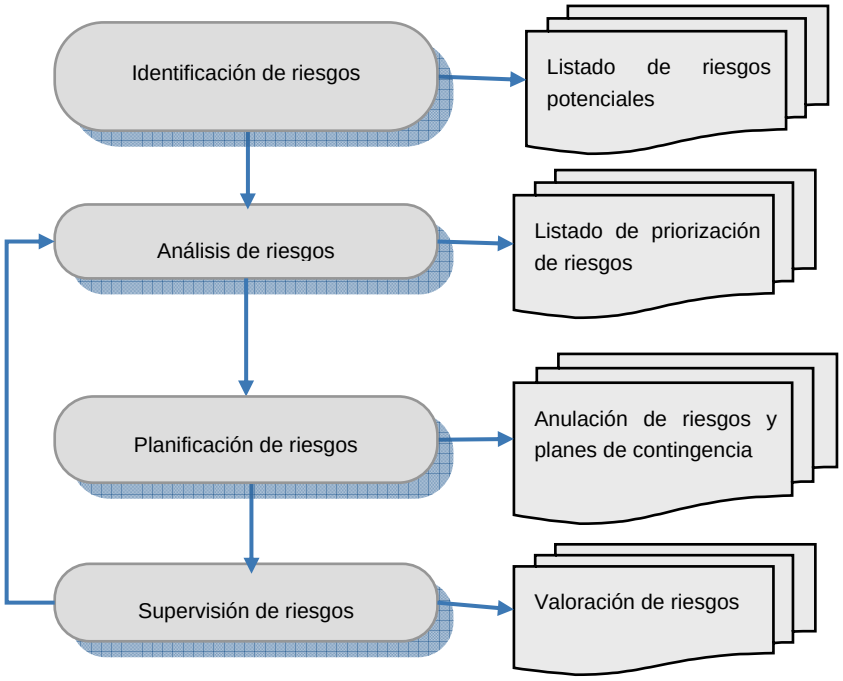


Figura 1.13 Procedimiento de gestión del riesgo.

Figura 1.14 Actividades de la Gestión del riesgo.

La razón de ser de estas empresas es:

- Las medianas, desarrollan software y servicios informáticos casi siempre en el campo de la gestión empresarial.
- Las empresas pequeñas, desarrollan distinto tipo de software o proveen servicios informáticos.

En la "PyME de software", el valor añadido en productos o servicios procede sustancialmente de software original o modificado.

Los problemas más frecuentes de las PyMES de Latinoamérica son: infraestructura, mano de obra disponible y ambiente para inversiones.

Lamentablemente este tipo de problema hace que exista escasez de mano de obra, pese a que los recursos humanos son excelentes y de altísima calidad.

Datos estadísticos reportados por [INEGI, 2003] muestran la cantidad de personal dedicado al desarrollo de software. Este personal se encuentra tanto en departamentos internos como en empresas desarrolladoras de software clasificadas en pequeñas y medianas organizaciones y grandes organizaciones cuyos datos aparecen a continuación.

Empresas con departamentos de software:	**Empresas dedicadas al desarrollo de software:**
Número de organizaciones: 12,521	Número de organizaciones: 1,462
Organizaciones pequeñas y medianas: 79.3 %	Organizaciones pequeñas y medianas: 83 %
Número de empleados de sistemas: máx. 10	Número de empleados de sistemas: de 11 a 25
Organizaciones grandes: 11.9 %	Organizaciones grandes: 27 %
Número de empleados: máximo 20	Número de empleados: de 57 a 100

El principal activo de estas empresas es el personal. Sin embargo, aquellas PyME que desarrollan software tienen menos de 15 personas dedicadas al software donde cada una tiene varios roles y responsabilidades. Esto trae por consecuencia que si los procesos no están bien definidos, gestionados y en mejoramiento continuo se arriben a problemas tales como proyectos con altos costos, entrega tardía de productos a los clientes, no realización de revisiones al software, no aplicación de sistemas de medición y baja calidad de los productos. Ante esta situación el esfuerzo tiene que ser enfocado hacia la mejora de los procesos de software, de tal manera que les permita a estas empresas incrementar su competitividad. Otro aspecto importante es la no certificación en este tipo de empresa debido a su costo y a que los modelos y normas se formulan para empresas grandes.

No obstante, en Latinoamérica hay empresas certificadas en CMMI en Argentina, Colombia (PSL en nivel 5) y Chile.

La certificación de calidad de los procesos de software y de los productos obtenidos es un paso que tarde o temprano las empresas productoras de software deben dar como respuesta a dos situaciones: la primera, por imagen, para incursionar y mantenerse en un mercado global; la segunda, por necesidad, para poder hacer de sus proyectos unidades administrativas eficientes y eficaces.

La PyME necesita alcanzar alto nivel de calidad y transmitir una imagen única y singular que las represente. Nadie compra los productos de software por el origen, lo principal para el cliente es contar con la garantía del servicio postventa.

Las organizaciones de software en Cuba

Dentro de las proyecciones del gobierno cubano para los próximos años se encuentra el fomento de una Industria Cubana del Software (ICSW), que permita diseñar y proveer de equipos electrónicos y sistemas informáticos que beneficien a la sociedad y también con posibilidad de exportarlos para aportar a la base material de todos los programas del país. [SIME, 1997], [Lage, 2000], [González, 2003], [Álvarez, 2003].

Para lograr esto, Cuba tiene un gran potencial humano calificado en el área de la informática, distribuidos en alrededor de 250 entidades que bien exportan software, producen software o brindan algún servicio informático. De estas entidades, solo 46, tienen carácter de empresa de software. A pesar de la potencialidad que poseen, tienen bajo índice de exportación.

A partir de documentos revisados, [García y Álvarez, 1999] [García, 2000] [Febles, 2000] [Febles, 2002], [GTI, 2002], [Álvarez, 2003], [Moreno, 2003], se destacan problemas relacionados a la dimensión organizacional que inciden en la dimensión técnica del software, entre ellos:

- Algunas organizaciones no utilizan un enfoque basado en procesos, sino en funciones.
- Los procesos o no están definidos o no se aseguran y controlan.
- La estructura de la organización no es por procesos.
- Las organizaciones no practican la planificación estratégica orientada hacia el mercado.
- Los trabajadores no han aprendido a mantener satisfechos a sus clientes.
- No existen procedimientos de trabajo formalizados.
- No existen herramientas integradas al control de los procesos.
- Los ejecutivos no cuentan con los datos necesarios para una adecuada toma de decisiones.
- Insuficiente papel motivador y visionario de la alta dirección.

- No se establecen acertados mecanismos de coordinación entre las diferentes áreas.
- Problemas técnicos en el diseño del nuevo producto o en su programación, deficiente calidad y rendimiento del producto, que al final no satisface los requisitos del cliente.
- Incapacidad para convertir a los nuevos clientes en clientes habituales.
- Entrega tardía de los proyectos que se aplican.
- Insuficiente información y comunicación de la cartera de productos que ofrecen las organizaciones, así como la gestión para su comercialización.
- Deficiente conocimiento por parte de los ejecutivos de la cultura organizacional.

Muchos de los problemas anteriores corresponden al proceso de ingeniería de requisitos, es por ello que el mejoramiento continuo de la calidad de los procesos de ingeniería de requisitos es el punto de partida para obtener un producto capaz de satisfacer las necesidades de los clientes.

1.8 Conclusiones parciales

- Las exigencias a las organizaciones productoras de software de un producto de calidad requiere de una mejora en los procesos de ingeniería de requisitos ya que en este se descubre, analiza, negocia, valida y especifica lo que debe hacerse.
- Existen insuficiencias en la gestión de requisitos tanto al nivel de proyecto como al nivel de organización. Esto requiere que el proceso de IR tiene que estar bien definido y ser desarrollado de forma disciplinada para repetir los éxitos anteriores en nuevos proyectos.
- Los modelos de proceso de ingeniería de requisitos, a pesar de su evolución aún presentan carencias. No todos abordan integralmente el proceso, ya que algunos no incorporan la gestión de los requisitos, ni el manejo de los cambios, o no tiene en cuenta la inevitable interacción con el resto del ciclo de vida del proyecto.
- El tratamiento de los riesgos desde las primeras etapas de la fase de ingeniería de requisitos daría un carácter proactivo al modelo, dando mayores posibilidades de éxito al proyecto. La gestión de los riesgos desde las primeras actividades de la ingeniería de requisitos no se reporta en los modelos estudiados.
- Es conveniente encontrar alternativas que combinen varias de las técnicas en la captura de requisitos.
- En la literatura no se reportan muchas métricas para el proceso de ingeniería de requisitos. Una adecuada definición de la línea base ayudará en este objetivo.

- La complejidad del proceso, dada por el factor humano, requiere que la conformación del equipo de trabajo rompa con la práctica comúnmente adoptada de cliente y proveedor como contrapartes y adopte una composición novedosa, lo que facilitaría el comportamiento iterativo e incremental del proceso.
- Se requiere de una herramienta CASE sencilla que provea al jefe del proyecto la información para una mejor gestión de los requisitos durante el ciclo de vida del proyecto.
- Por todo ello, es que constituye una necesidad para la industria de software cubana, definir un procedimiento para la definición y gestión de los requisitos. Este incluirá, un conjunto o de procedimientos, con toda la base teórica que exige y fundamentados científicamente. El procedimiento tendrá un carácter proactivo que permitirá tomar decisiones, para definir las acciones correctivas, para la mejora, en el ámbito de un proceso de software organizado.

Capítulo 2. Procedimiento para el desarrollo del proceso de ingeniería de requisitos en un proyecto software (PROCIR)

En este capítulo se expone una solución al problema científico planteado en la introducción de esta investigación, que toma como base el estado del arte plasmado en el marco teórico referencial y que consiste en un procedimiento para el desarrollo de de la ingeniería de requisitos en un proyecto software, las técnicas y herramientas para obtener beneficios, medidos a través de la satisfacción de los clientes en las organizaciones de software cubanas.

Primero se exponen las premisas generales para la construcción del procedimiento y los principios en que este se sustenta. Luego se describe el procedimiento y cada una de las actividades en que está dividido, incluyendo la propuesta de herramientas metodológicas para implementar el mismo.

2.1 Premisas generales para la construcción del procedimiento

La construcción del procedimiento se realizó sobre las premisas siguientes:

- Motivar a las organizaciones a aplicar un enfoque basado en procesos.
- El procedimiento es independiente de cualquier paradigma de programación, metodología de calidad, modelado y desarrollo. Su diseño permite aplicarlo a cualquiera de los tipos de software existentes.
- Estimular a las organizaciones de software a desarrollar las actividades de la ingeniería de requisitos de forma organizada, sistemática y repetitiva, con el objetivo de crear una base de conocimiento que les permita aplicar las mejores prácticas para obtener mayor grado de satisfacción del cliente.
- Estimular la participación de clientes y usuarios en todas las actividades del proceso, involucrándolos en la toma de decisiones de los cambios solicitados.
- Su implementación práctica de forma repetitiva puede preparar a la organización para futuros procesos de evaluaciones de calidad.
- Las técnicas y herramientas metodológicas para la implementación del procedimiento y la esencia cíclica del mismo garantizan el mejoramiento continuo.

2.2 Principios en los que se sustenta el procedimiento

Los principios que sustentan el procedimiento desarrollado son los siguientes:

Adaptabilidad: puede ser aplicado en cualquier organización de software del país.

Generalidad: puede usarse como herramienta metodológica para realizar estos estudios en otros procesos semejantes.

Mejoramiento continuo: se materializa en la naturaleza iterativa e incremental del proceso, un ciclo seguido de otro que comienza nuevamente con las mismas actividades, de forma que se puedan ir mejorando deficiencias detectadas.

Pertinencia: la posibilidad de ser aplicado integralmente en las condiciones actuales de las organizaciones de software cubanas.

Aprendizaje: a través de las lecciones aprendidas por la aplicación repetitiva del procedimiento en diferentes proyectos.

Consistencia: se definen los pasos lógicos de ejecución de este tipo de proceso.

Suficiencia: se dispone de toda la información que se requiere para su aplicación

2.3 Descripción del Procedimiento PROCIR

La definición del procedimiento está basada en los principios, métodos y técnicas de la ingeniería de requisitos, las mejores prácticas en esta área [Rational, 2001] y en los aspectos novedosos de los modelos CMMI [CMMI-SW, 2002], [SPICE, 1998]. Lo integran un conjunto de actividades que garantizan los procesos que son abarcados por la ingeniería de requisitos, disciplina de la que forma parte el mismo, a nivel de proyecto, ubicándose en las corrientes de mejora continua de los procesos de software. Está demostrado que para lograr un producto software que satisfaga al cliente es necesario partir de una especificación de requisitos de calidad y una gestión eficiente. [Leffingwell y Widrig, 2003], [Sommerville, 2005].

En la figura 2.1 se representan las siete etapas del procedimiento PROCIR en una secuencia lógica de pasos. PROCIR ha sido construido a partir de las mejores prácticas reconocidas para la especificación y la gestión de los requisitos [Rational, 2001], [SWEBOK, 2004].

El procedimiento está enfocado a obtener una especificación de requisitos del software de calidad, a través del desarrollo de la gestión de los requisitos que la conforman a lo largo de todo el ciclo de vida del proyecto software, y en él:

- Se integran clientes, usuarios, desarrolladores y otros implicados, facilitando una mejor y más rápida comprensión del problema.
- Se evalúan las características de calidad de la ERS para disminuir los cambios.

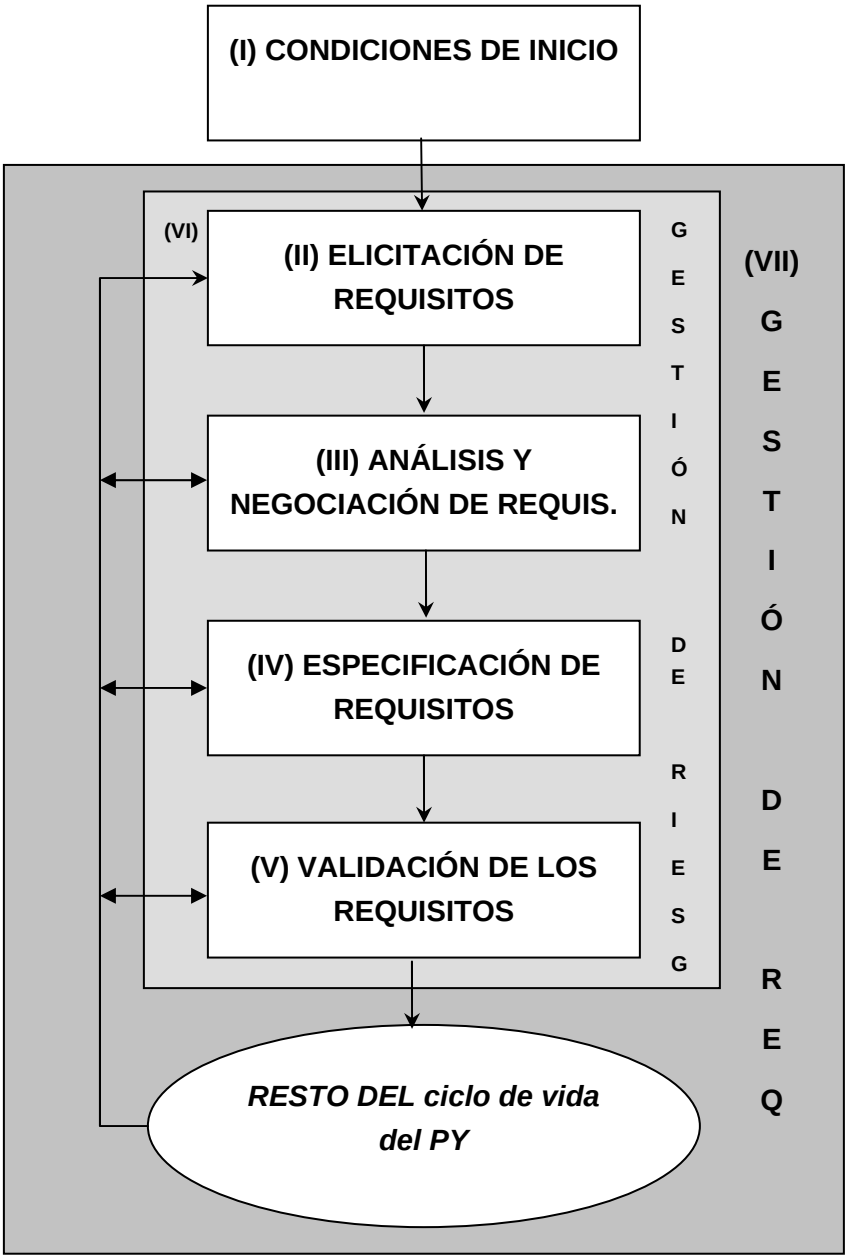


Figura 2.1 Procedimiento para el desarrollo del proceso de ingeniería de requisitos.

- Se establecen mecanismos que garantizan la trazabilidad de los requisitos a lo largo de todo el ciclo de vida, hacia atrás, hacia delante y entre ellos. Esto permite proveer una relación entre requisitos, diseño e implementación del sistema y el reconocimiento temprano de aquellos que no son satisfechos.

De este modo, PROCIR posee:

- Un método que incluye el soporte técnico (tarjetas, tablas, etc.)

- Técnicas de gestión que permiten un esfuerzo sostenido hacia la calidad a lo largo del ciclo de vida del proyecto.
- Un conjunto de procedimientos e instrumentos de control, que aplicados, permiten obtener una ERS de calidad.

PROCIR está diseñado para proporcionar resultados de evaluación repetibles y comparables dentro de contextos similares. La repetibilidad está basada en la guía para conducir el desarrollo del proceso y la evaluación de su desempeño en proyectos similares.

Este procedimiento está enfocado a procesos, facilita las tareas repetitivas, no se depende de un individuo para ejecutar una tarea, ayuda a conocer el desempeño, mejorar las estimaciones y calidad de los productos software y permite una buena administración y control del proceso de IR.

El amplio campo de aplicabilidad del procedimiento está dado por la factibilidad de su utilización para diversos tipos de proyectos durante las fases de la IR, siendo totalmente independiente de cualquier metodología de desarrollo. Su dominio abarca las relaciones cliente - proveedor en todas las áreas de proceso.

2.3.1 Características del procedimiento

Su concepción teórica incluye lo relevante de las mejores prácticas y formas de hacer de la ingeniería de requisitos. Es un procedimiento iterativo e incremental por la propia naturaleza del proceso. No está sujeto a ninguna metodología de desarrollo de software. También es proactivo, ya que en él aparecen actividades que de hecho son preventivas, indicando cómo desarrollar bien, de manera coherente y organizada, cada actividad y tarea desde la primera vez. El procedimiento ha sido diseñado de forma clara y sencilla. Está conformado por un conjunto de actividades y etapas que definen e integran los diferentes procesos de la ingeniería de requisitos de un proyecto software. Los elementos de éste son: objetivos, elementos de evaluación, entradas, salidas y controles.

Procedimiento PROCIR/Objetivos

Definir los requisitos del producto a construir de forma que, la especificación que resulte, permita desarrollar el resto del proceso de software con eficiencia, dentro del plazo y costos previstos, con alto grado de satisfacción del cliente, a partir de una eficaz y planificada gestión de los requisitos y los cambios que sean solicitados por cualquiera de las partes involucradas.

Procedimiento PROCIR/Elementos de Evaluación

Los elementos de evaluación de PROCIR son las propiedades de los requisitos y de la especificación de los requisitos, a través de las listas de chequeo que permiten evaluarlas, los elementos de riesgo y los cambios solicitados.

Procedimiento/Entradas, Salidas y Controles

La entrada al procedimiento es la decisión de la alta dirección de llevar a cabo el proyecto. Su salida principal es la Especificación los Requisitos del Software a desarrollar, este documento constituye un acuerdo entre las partes y puede considerarse contractual, por el compromiso que implica entre el cliente y la entidad desarrolladora. Los cambios que sean aprobados serán asentados en las versiones sucesivas de la ERS, las que, al emitirse, mantendrán su status legal entre las partes.

Los controles están dados por la evaluación de los atributos de calidad de cada requisito de forma individual y de la ERS como conjunto de estos, y por la aplicación de las técnicas de trazabilidad de los requisitos durante todo el proceso de IR. Los requisitos son identificados, definidos y actualizados a través de todas las fases del ciclo de vida del producto.

2.4 Descripción de las actividades del procedimiento PROCIR

2.4.1 Actividad I: CONDICIONES DE INICIO

Para poder iniciar con éxito las actividades específicas del proceso de ingeniería de requisitos, y posteriormente de todo el proyecto, es imprescindible que se desarrollen varias tareas, de las que dependerá el futuro desarrollo del proyecto. Ellas son:

- Decisión de llevar a cabo el Proyecto.
- Constitución del Equipo de Trabajo de Gestión de Requisitos.
- Designación del Jefe del Proyecto.
- Recogida inicial de información.
- Estudio de Viabilidad del Proyecto.

La decisión de llevar a cabo el proyecto es tomada por la alta dirección del cliente, a partir de sus necesidades y estudio preliminar de factibilidad. Es independiente de los elementos que puedan obtenerse, a favor o en contra, en las actividades subsiguientes de este procedimiento. Esta decisión conlleva la solicitud del trabajo que se requiere, por escrito, al proveedor.

La constitución del equipo de trabajo para el desarrollo de las actividades inherentes a la ingeniería de requisitos es una tarea más compleja, por cuanto las características de las relaciones humanas entre los miembros seleccionados puede ser un factor clave del éxito o fracaso del proyecto.

Los autores [Pressman, 2002], [Sommerville, 2005] y otros, coinciden en la necesidad de lograr una comunicación efectiva entre los participantes en el proceso de ingeniería de requisitos para poder cumplir los objetivos del proyecto.

Para superar muchas de las barreras psicológicas del proceso de comunicación, la autora propone que sea creado un ***equipo de trabajo conjunto para la gestión de los requisitos*** (en adelante sólo equipo de trabajo), integrado por todas las partes involucradas en el proyecto, con objetivos, responsabilidades e intereses comunes, y no como es habitual, donde partes separadas (clientes y proveedores) interactúan independientes para "defender" objetivos, intereses, puntos de vistas. Las partes son complemento y no contrapartes, como se observa en la figura 2.2. Esta propuesta está acorde con lo planteado en la Programación Extrema, respecto al valor V1: comunicación continua, y a dos de sus principios, Pr1: trabajo en equipo y Pr2: participación del cliente, [Hurtado, 2005].

Esta conjunción de las partes permite que, si el programador necesita aclarar algo en los requisitos, se reúna con el equipo de trabajo y se resuelvan sus dudas. Un cliente no puede entregar los requisitos, irse y luego volver a mirar los resultados, sino que debe estar disponible para responder a dudas o preguntas que surjan dentro de las iteraciones del desarrollo. Además con esta integración es posible manejar mejor los problemas del cambio de los requisitos.

Para lograr resultados satisfactorios es necesario que todos los miembros del grupo estén comprometidos con el software a desarrollar.

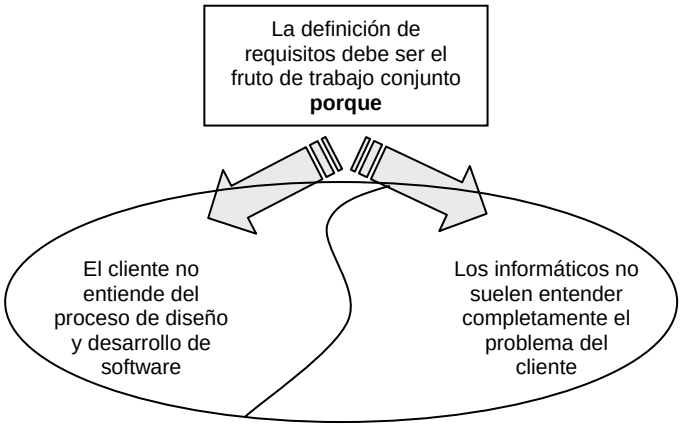


Figura 2.2 Fuentes de conocimientos en el equipo de trabajo.

El equipo de trabajo debe estar integrado por:

- Clientes y patrocinadores
- Experto del negocio
- Clases de usuarios finales (representante de usuarios por clases de usuarios)
- Especialistas informáticos: analista o ingeniero de requisitos, desarrollador
- Otros implicados (técnicos de pruebas, técnicos de sistemas, SQA, terceros)

Es necesario tener en cuenta que:

a. El número de miembros variará en función de las características y condiciones particulares de cada proyecto (alcance) y de las organizaciones implicadas (recursos humanos disponibles). Este puede variar a medida que avanza el proceso y se van descubriendo nuevos implicados, pero es importante que no sea muy elevado, máximo 8 como permanentes [Sommerville, 2000].

b. Los clientes deben tener conocimientos de las necesidades, recursos disponibles y estar envestidos con poder de decisión.

c. Los usuarios deben poseer conocimiento funcional profundo del dominio del problema y del entorno sobre los que incidirá la aplicación informática a desarrollar.

d. Los especialistas informáticos deben ser no menos de dos y es muy deseable que, al menos uno de ellos, tenga experiencia en el dominio de la solución del problema planteado y que posean los conocimientos técnicos especializados imprescindibles para acometer el proyecto.

e. La estructura del equipo debe ser preferiblemente informal (democrática, se toman decisiones por consenso, las tareas son asignadas por habilidad y experiencia, mejor cohesión y rendimiento) para facilitar la comunicación entre los miembros [Sommerville, 2000], [Hurtado, 2005].

Como principio, el equipo de trabajo, integrando ambas partes en un todo, estará integrado por miembros que generen e implementen decisiones colectivas para el producto en desarrollo, sean colectivamente responsables de la liberación del producto y de su calidad. “Todos son responsables”.

Esta propuesta persigue lograr que los miembros tengan, o adquieran en el menor plazo, las habilidades y experticia necesarias para seguir todo el ciclo de vida del producto, colaboren interna y externamente, se entiendan con objetivos y tareas comunes, se conduzcan a sí mismos (independencia), de acuerdo a principios operativos y reglas fundamentales de comunicación entre los miembros del equipo. Esta concepción en el procedimiento es premisa de éxito del proyecto.

Designación del Administrador del Proyecto

La alta dirección debe tener en cuenta que, para que funcione adecuadamente la comunicación y fluya coherentemente el proceso, el designado debe tener preparación técnica (en informática) adecuada al alcance del proyecto que se va a desarrollar y un conocimiento mínimo razonable de la problemática funcional del sistema y su entorno. El seleccionado debe poseer las características, habilidades y cualidades imprescindibles para ser líder de un equipo de trabajo de estas características.

Recogida inicial de información

Al inicio se recaba información de los directivos de nivel alto y medio, porque son los que cuentan con una visión lo suficientemente global del problema como para ayudar a crear una visión lo suficientemente global del proyecto. Después, en el análisis detallado del problema se debe acudir a fuentes más específicas, como por ejemplo los usuarios finales. Las técnicas de recogida de información son las mismas que se utilizan en otras actividades del proceso y que fueron analizadas en el Capítulo I, sólo que en este momento del proyecto no se utilizan a fondo.

Con la información elemental, pero no trivial, recopilada, se elabora el Informe Preliminar de Necesidades, que constituye la base de la siguiente tarea.

Estudio de Viabilidad del Proyecto (Análisis de Factibilidad)

Para iniciar este estudio se deben realizar las preguntas: ¿qué, para qué y cómo? Sus respuestas constituirán alternativas de solución.

Es posible que después de analizar la viabilidad del proyecto, éste se desestime. Para tomar esa decisión, ¡no se debe pensar en el dinero que ya se ha invertido en este análisis!, por cuanto siempre será más costoso emprender un proyecto que se sabe de antemano que fracasará.

El estudio de factibilidad es una actividad compleja que requiere experiencia. Éste permite realizar el análisis y la evaluación de las alternativas que se generan desde los puntos de vista siguientes:

Económico: ¿vale la pena invertir en el proyecto? ¿Los beneficios compensan los costos?

Técnico: ¿se encuentra disponible la tecnología necesaria?

Legal: se debe dilucidar si los requisitos atentan contra alguna ley o reglamento

Operativo: se debe determinar si el sistema se podría implantar de manera efectiva en la empresa. ¿Puede coordinarse con los métodos existentes? ¿Encaja en la filosofía de la empresa?

Posteriormente se selecciona la alternativa que permita resolver mejor el problema planteado, se realiza la argumentación detallada y se define el plan inicial del proyecto.

En esta tarea se debe determinar:

– Contribución del sistema a los objetivos de la organización

– Posibilidad de implementar con la tecnología actual

– Integración con el resto de sistemas

– Integración con el procedimiento de trabajo

– Grado de necesidad de un nuevo sistema

– Problemas actuales que resolverá el sistema

– Contribución del sistema al negocio

– Posibilidad de poblar con los datos actuales

– Nivel de experiencia en la tecnología necesaria

– Nuevas necesidades que genera el sistema

Como mecanismo de evaluación y control en esta actividad se puede utilizar una lista de comprobación (*checklist*) que responda, entre otras, las preguntas siguientes:

¿Existe una petición por escrito del trabajo a realizar?

¿La petición por escrito describe de forma general el problema a resolver, las salidas a producir y las entradas necesarias?

¿Tiene la petición por escrito un único autor?

¿Puede dicho autor tomar decisiones sobre la petición?

¿Está la persona que realiza la extracción de requisitos familiarizada con el área de negocio bajo consideración?

¿Está familiarizada la persona que realiza la extracción de requisitos con la jerga del negocio, sus cuestiones, personal y políticas?

¿Se ha pedido una reunión formal con el cliente para analizar los resultados del Estudio de Factibilidad?

¿Se ha establecido una agenda y las reglas básicas para la reunión? ¿Se ha convocado a todas las personas que necesitan asistir?

En este momento, la descripción del problema es muy preliminar y contiene las necesidades más elementales, expresadas superficialmente y que sirve únicamente como punto de inicio para comprender los requisitos del sistema.

Las ***salidas*** de esta actividad son el informe final del análisis de factibilidad y la solicitud escrita del trabajo a realizar. Ambas constituyen, a su vez, las ***entradas*** a la siguiente actividad. También constituye entrada a las actividades de Gestión de Riesgos y Gestión de Requisitos.

2.4.2 Actividad II. ELICITACIÓN DE LOS REQUISITOS

Elicitación es el proceso mediante el que se identifican los ítems de información que determinan las características deseadas y las restricciones que deberá satisfacer el sistema software, que tendrán efectos satisfactorios para el usuario, en el ambiente donde se encuentra.

Se refiere a la captura y descubrimiento de los requisitos que deberán ser implementados. Es una actividad más "humana" que técnica. Se identifica a los interesados (*stakeholders*) y se establecen las primeras relaciones entre ellos y el equipo de trabajo.

Su *objetivo* es la obtención de una especificación preliminar detallada (informal) de las necesidades de los usuarios del software a desarrollar.

Durante la elicitación de los **requisitos** se debe:

- Comprender el dominio de la aplicación y el problema en cuestión.
- Comprender el contexto del negocio.
- Comprender necesidades y restricciones de los clientes y usuarios.
- Determinar funcionalidades y capacidades.
- Detectar las interfaces internas y externas.
- Determinar el nivel de calidad.
- Fijar las normas de seguridad.
- Establecer las normas de ergonomía,
- Definir el conjunto de datos y relaciones.
- Establecer las condiciones y tipos de pruebas.
- Definir los entornos de operación y ejecución.
- Determinar las necesidades de mantenimiento.
- Definir normas de aceptación del producto.
- Definir requisitos de instalación.
- Fijar la documentación técnica, de usuario, de operación y mantenimiento

Las *fuentes* para la elicitación de los requisitos, entre otras, pueden proceder de:

- Metas de la organización y factores críticos de éxito.
- Conocimiento del dominio de la aplicación.

- Los interesados y afectados por el sistema.
- El entorno físico organizacional que rodea al sistema.
- Los procesos de negocio.

Las t*écnicas* que pueden ser utilizadas en esta actividad están las descritas en [Christel y Kang 1992], [Goguen y Linde 1993], [Carrizo y Juristo, 2002], un resumen de las cuales se recogen en el Capítulo I.

Como instrumento para facilitar la recogida y anotación de los requisitos y de los datos necesarios sobre ellos, que serán imprescindibles para la elicitación y las actividades posteriores del procedimiento, se define una tarjeta o plantilla para uso de todos los miembros del equipo. El llenado de la tarjeta se iniciará en el momento en que es descubierto el requisito. La información será completado en la medida en que avance el proceso y vayan siendo precisados y refinados los requisitos. Esta tarjeta recogerá:

ID del requisito: identificador único del requisito, será asignado al ser analizado en reunión del equipo, no individualmente por quien lo descubrió.

Descripción: una frase que describa el requisito. Preferentemente de la forma “el sistema debe...”.

Clasificación: funcional, no-funcional.

*Área: e*n qué apartado del documento de requisitos debería figurar. En otras palabras, a qué parte del sistema afecta (por ejemplo, pedidos, contabilidad, ventas, etc.).

Razón: ¿por qué es importante este requisito?

Origen: ¿quién lo solicita o qué lo necesita?

Prioridad: hace referencia al orden temporal, indica en qué fase de construcción del sistema se incluirá la funcionalidad que realice el requisito. Puede valorarse, por ejemplo, como alta, media y baja.

Importancia: la necesidad de un requisito hace referencia al interés de los usuarios/clientes en que la aplicación lo realice, y hasta qué punto estarían dispuestos a pasarse si él. Puede evaluarse de esencial (obligatorio), deseable u opcional.

Nivel de riesgo: alto, medio, bajo

Dependencias: otros requisitos de los que depende.

Conflictos: requisitos que, de alguna forma, contradicen a éste.

Referencias: qué otras informaciones son necesarias para comprender el requisito (por ejemplo, documentos).

Historia: historia de los cambios del requisito.

Estado: con relación a la actividad o fase del proyecto en que se encuentra (ejemplo: en análisis, en especificación, o implementado, en diseño, en codificación,…)

Versión: en qué iteración será implementado.

Estabilidad: su probabilidad de cambio puede ser alta, media o baja.

Las tareas que deben realizarse durante la actividad de elicitación son:

- Obtención de información sobre el dominio del problema y del sistema actual.
- Crear un glosario de términos.
- Identificar a los *stakeholders* (implicados) en el sistema.
- Preparar y realizar las reuniones de elicitación/negociación.
- Identificar, revisar y aprobar los objetivos y alcance del sistema

Obtención de la información acerca del dominio del problema y del sistema actual (automatizado o no).

Esta tarea se realiza aplicando diversas técnicas, entre las que se encuentran:

Estudio del dominio del problema, que puede ser opcional sólo si los informáticos tienen experiencias previas en el dominio del problema actual y si conocen la forma de hacer o sistema actual. Esto es válido, sobre todo, cuando se desarrollan sistemas o aplicaciones que están dentro de la cartera de negocios de la organización.

Estudio del sistema actual, sea un proceso manual o una aplicación informática.

Análisis y documentos de sistemas anteriores, para establecer, esclarecer, ampliar, las necesidades de los clientes, usuarios y otros *stakeholders*. Puede ser útil escribir una lista preliminar o informal de ellas, organizada por categorías o tipos.

Obtener y estudiar las regulaciones, estándares, leyes, normas, documentos legales internos, manuales de procedimiento, y similares, que pudieran afectar el desarrollo y explotación del sistema si no se tienen en cuenta.

Investigar y establecer las mejores prácticas que pudieran ser aplicadas en la búsqueda de las soluciones al problema planteado.

Crear un glosario de términos

Debe confeccionarse un glosario en dónde se definan todos los términos que tengan significados comunes (sinónimos), e incluso antónimos, y que serán utilizados durante el proyecto. Esta compilación resulta sumamente beneficiosa, porque reduce los términos ambiguos desde el principio, ahorra tiempo, asegura que todos los participantes de una reunión están en la misma página, además de ser reutilizable en otros proyectos.

Identificar a los *stakeholders* (implicados) en el sistema [Pacheco, 2004]

Identificar a todos los afectados e implicados evita que se produzcan sorpresas a medida que avanza el proyecto. Las necesidades de cada *stakeholder* son discutidas y sometidas a debate durante el proceso de ingeniería de requerimientos, aunque esto no garantiza que vaya a estar disponible toda la información necesaria para especificar un sistema adecuado.

Para saber quiénes son las personas, departamentos, organizaciones internas o externas que se verán afectadas por el sistema, se pueden realizar algunas preguntas que posibiliten conocer la opinión de todo aquel que, de una u otra forma, está involucrado, ya sea directa o indirectamente. La lista puede incluir, entre otras:

¿Quién usará el sistema que se va a construir?

¿Quién desarrollará el sistema?

¿Quién probará el sistema?

¿Quién documentará el sistema?

¿Quién dará soporte al sistema?

¿Quién dará mantenimiento al sistema?

¿Quién se beneficiará por el retorno de inversión del sistema?

¿Quién recibirá los informes que genere?

Preparar y realizar las reuniones de elicitación / negociación

Estas reuniones tienen como objetivos identificar a otros usuarios para el sistema; determinar, esclarecer, necesidades de clientes y usuarios; identificar el problema; proponer soluciones; especificar un conjunto de requisitos de la solución; resolver conflictos ya identificados.

El equipo de trabajo preparará y realizará estas reuniones en lugar neutral, bajo la premisa “acuden todos”. Al inicio se establece una agenda de reuniones. Se nombra un <<coordinador>> que controlará la reunión. Para la primera reunión se preparan preguntas de contexto libre, para el entendimiento del problema y para lograr la eficacia de la reunión.

Identificar, revisar y aprobar los objetivos y alcance del sistema

En esta actividad se deben establecer los objetivos que se espera alcanzar mediante el software a desarrollar. Se identificarán los límites del sistema (visión y alcance), aspecto muy importante, porque se debe definir claramente qué se va y qué no se va a construir, para entender la estrategia del producto a corto y largo plazo. Debe determinarse cualquier restricción ambiental, presupuestaria, de tiempo, técnica y de factibilidad que limite el sistema que se va a desarrollar.

También hay que identificar las interacciones del sistema con el entorno y revisar si existen conflictos con los objetivos previamente identificados.

Como mecanismo de evaluación y control en esta **actividad** se puede utilizar una lista de comprobación que incluya las preguntas siguientes:

¿Tiene el cliente voluntad de acudir a las reuniones?

¿Se han desarrollado escenarios de uso por escrito como parte de la actividad de extracción de requisitos?

¿Se han recorrido los escenarios con el cliente? ¿Se han especificado inexactitudes y modificaciones?

¿Se han definido completamente los objetos básicos de entrada?

¿Se han definido completamente los objetos básicos de salida?

¿Se han definido, descompuesto y asociado a escenarios de uso las funciones a implementar?

¿Se ha definido completamente el comportamiento del sistema (desde el punto de vista del usuario) bajo varias condiciones de operación?

¿Se han discutido y definido las características de la interfase de usuario?

¿Se ha determinado el valor añadido para los requisitos de usuario?

¿Se han identificado las restricciones y limitaciones?

¿Se han clasificado los requisitos por importancia para el usuario?

¿Se han realizado revisiones de toda la información recogida como parte de la extracción de requisitos?

La ***salida*** de esta actividad es una declaración informal de requisitos (escrita), que constituye a su vez la ***entrada*** a la siguiente. También constituye entrada a las actividades de Gestión de Riesgos y Gestión de Requisitos.

2.4.3 Actividad III. ANÁLISIS Y NEGOCIACIÓN

Es la actividad de la IR en la cual *se estudia* la información extraída durante la elicitación, para identificar la presencia de áreas no detectadas, requisitos contradictorios y peticiones que aparecen como vagas e irrelevantes [Báez y Barba, 2001], se clasifican y se negocian con el cliente para verificar los puntos de acuerdo y entendimiento de sus necesidades; es vital precisar los límites del sistema y la interacción con su entorno. Se trasladan los requisitos de usuario a requisitos del software (implementables).

El *objetivo* del análisis es descubrir problemas en la declaración informal de requisitos generados durante la captura de los mismos.

Las tareas de esta actividad son:

- Detección de problemas potenciales.
- Clasificación.
- Modelación.
- Negociación.
- Resolución de conflictos detectados.
- Priorización y determinación de los niveles de riesgos.

La diversa gama de fuentes de las cuales provienen los requerimientos, hace necesaria una evaluación de los mismos antes de definir si son adecuados para el cliente. El término "adecuado" significa que ha sido percibido a un nivel aceptable de riesgo, tomando en cuenta las factibilidades técnica y económica, a la vez que se buscan resultados completos, correctos y sin ambigüedades. Es en el análisis cuando muchos conflictos entre requisitos son descubiertos.

Durante el análisis hay que considerar las propiedades de los requisitos definidas en el Capítulo I de forma individual y en grupo (como un todo). También deben ser asignados los requisitos que brindarán sus funcionalidades a subsistemas específicos.

Descubrir problemas potenciales

En esta tarea se identifican requerimientos ambiguos, incompletos, inconsistentes, se detectan requisitos no especificados etc., es decir, se asegura que todas las características imprescindibles estén presentes en cada uno de los requerimientos. Esta tarea se debe realizar en sesión de trabajo del equipo asignado al proyecto, donde se analizará cada requisito individualmente y su evaluación será por consenso. Es deseable que los miembros del equipo (o parte de ellos), individualmente, hayan realizado un estudio previo de la lista de los requisitos para detectar problemas.

Los problemas que tengan los requisitos serán solucionados en esa propia reunión si fuera posible, o se retornarán a la actividad anterior para la obtención de información que permita resolverlos.

Clasificación de los requisitos elicitados

En el análisis de requisitos, éstos se clasifican en:

Funcionales: describen los servicios o funciones que se esperan del sistema. Entre ellos los requisitos de ejecución, mapeo, información, control, comunicación, construcción.

No funcionales: describen restricciones al sistema. Se refieren a las propiedades emergentes del sistema como la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, la capacidad de los dispositivos de entrada/salida, y la representación de datos que se utiliza en las interfaces del sistema.

En [Dumke, 2005] aparecen definidos formalmente los tipos de requisitos.

También se clasifican los requisitos por costo de implementación, según su volatilidad o estabilidad, de información, de documentación, de calidad, plataforma, proceso.

Analizar y evaluar los riesgos por requisitos

Se valora individualmente cada requisito y se le asigna el nivel de riesgos (alto, medio, bajo). Se tiene en cuenta sus propiedades para determinar el riesgo que se corre para implementar el requisito, cumpliendo las necesidades y expectativas del cliente. Los requisitos ambiguos, incompletos, difíciles de verificar o rastrear, tendrán mayores niveles de riesgos.

Modelación conceptual

La meta de esta tarea es entender mejor el problema, más que iniciar el diseño de la solución (idealmente).

El modelo de requisitos tiene como objetivo delimitar el sistema y capturar la funcionalidad que debe ofrecer desde la perspectiva del usuario. Este modelo puede funcionar como un contrato entre el desarrollador y el cliente del sistema, y por lo tanto proyecta lo que el cliente desea según la percepción del desarrollador. Por lo tanto, es esencial que los clientes puedan comprender este modelo. Por ejemplo, para modelar la frontera del sistema puede ser útil un diagrama de contexto.

El tipo de modelo a elegir depende de la naturaleza del problema, experiencia del modelador, disponibilidad de herramientas o por decreto, cuando el cliente impone una notación.

Si se selecciona desarrollo estructurado, el modelado conceptual se podrá realizar mediante diagramas de flujo de datos (DFD), diagrama de descomposición (DDF), diagramas de entidad relación (DER), diagramas de transición de estados (DTE).

Si se hará desarrollo orientado a objetos, se modelará mediante casos de uso (CU), diagramas de clases (DC), diagramas de actividades (DA), diagramas de comportamiento: secuencia y colaboración.

El propósito del modelo de requisitos es comprender completamente el problema y sus implicaciones. Todos los modelos no solamente se verifican contra el modelo de requisitos, sino que también se desarrollan directamente de él. El modelo de requisitos sirve también como base para el desarrollo de las instrucciones operacionales y los manuales, ya que todo lo que el sistema deba hacer se describe aquí desde la perspectiva del usuario. Este no es un proceso mecánico, el analista debe interactuar constantemente con el cliente para completar la información faltante, y así clarificar ambigüedades e inconsistencias.

Es necesario resaltar que el enfoque del modelado de los requisitos visto de esta manera es limitado, obliga al negocio a acomodarse a la solución que se busca al problema. La generación de los modelos de requisitos del sistema software a partir de los modelos del negocio resolvería este problema. Sería deseable que los modelos de requisitos fueran generados a partir de los modelos de negocios.

La construcción incremental y evolutiva de los modelos de negocios, que cuenten con un nivel de abstracción adecuado para realizar una suave transición a los modelos de requisitos, favorecerá la producción de sistemas software precisos, rigurosos y confiables.

Generalmente los ingenieros que se dedican al desarrollo de software restan importancia a los modelos de negocios, los cuales permiten visualizar la forma de operar de la empresa, así como las necesidades de los usuarios del sistema a desarrollar.

En un proceso de producción de software que no tenga como primera etapa el modelado de procesos de negocios, cualquier esfuerzo para obtener los requisitos del sistema de información estará disminuido por la incapacidad de asegurar, la utilidad real de éste en el contexto de las tareas organizacionales. [Martínez, et al., 2002], [Koubarakis, 2000].

El analista debe separar entre los requisitos verdaderos y las decisiones relacionadas con el diseño e implementación. Se debe indicar cuales aspectos son obligatorios y cuales son opcionales para evitar restringir la flexibilidad de la implementación.

Negociación de Requisitos

En todo proceso de ingeniería de requisitos intervienen distintos individuos con distintos y, a veces, enfrentados intereses. Estos conflictos entre requisitos se descubren durante el análisis. Todo conflicto descubierto debería disparar un proceso de (re)negociación. Los conflictos **nunca** se deben resolver “por decreto”.

Durante la negociación se priorizan los requisitos y se llega a un compromiso sobre el conjunto de requisitos a implementar. Se asignan las prioridades, al identificar la importancia que tiene un requerimiento en términos de implementación y debe ser usada para establecer la secuencia en que ocurrirán las actividades de diseño y prueba de cada requisito. La prioridad de cada requerimiento dependerá de las necesidades que tenga el negocio. En base a la prioridad, cada requerimiento puede ser clasificado como esencial (si afecta una operación crítica del negocio), deseable (inclusiones de mejoras) u opcional (informativo o que puede esperar)

Una vez hecha esta categorización de los requerimientos, se puede tomar como estrategia general el incluir los imprescindibles, discutir los deseables y descartar los innecesarios. Antes de decidir la inclusión de un requerimiento, también debe analizarse su costo, complejidad, y una cantidad de otros factores. Por ejemplo, si un requerimiento tuviera una implementación trivial, puede ser una buena idea incluirlo, por más que éste sea sólo deseable.

Durante el proceso de negociación de los conflictos detectados hay que considerar que el conflicto no es rechazable y no debe resolverse por decreto, sino precisamente, mediante un proceso de negociación, desde este punto de vista, los conflictos son positivos, pues son fuente de nuevos requisitos.

Se recomiendan las técnicas de negociación del tipo WinWin (ganar/ganar) [Boehm et al. 1994], [Boehm, 2001].

En la actividad de análisis y negociación se incrementa la comunicación entre los miembros del equipo de trabajo y los demás afectados. Para que los requerimientos puedan ser comunicados de manera efectiva, hay una serie de consideraciones que deben tenerse en cuenta; entre las principales tenemos:

- Documentar todos los requerimientos a un nivel de detalle apropiado.
- Mostrar todos los requerimientos a todos los involucrados en el sistema.
- Analizar el impacto que causen los cambios a requerimientos antes de aceptarlos.
- Negociar con flexibilidad para que exista un beneficio mutuo.
- Enfocarse en intereses y no en posiciones.
- Establecer las relaciones entre requerimientos que indiquen dependencias (Matriz de seguimiento de dependencias: indica cómo se relacionan los requisitos entre sí).

Para realizar el análisis se pueden utilizar *checklist*, matrices de interacción, matrices de dependencias y otras.

Los acuerdos alcanzados durante las sesiones de negociación deben ser convenientemente anotados, favoreciéndose así la trazabilidad de los requisitos a sus orígenes.

Puede que algunos requisitos deban ser regresados a la Actividad de Elicitación, porque no se llegó a consenso en cuanto a sus características y es necesario profundizar en el dominio del problema, profundizar con las fuentes de origen, recurrir a nuevos usuarios, hasta que se resuelva el problema. Este evento deberá ser anotado en la tarjeta de requisito para que no se pierda la trazabilidad.

La captura, el análisis y la negociación no son actividades independientes. Se retroalimentan hasta tener un documento de requisitos negociados, por esta razón el modelo para este procedimiento asumió la existencia de cualquier tipo de ciclo entre ellas, como fue propuesto por Pohl en su modelo, analizado en el Capítulo I.

La ***salida*** de esta actividad es un acuerdo de requisitos, que constituye a su vez la ***entrada*** a la s actividad de especificación de requisitos. También constituye entrada a las actividades de Gestión de Riesgos y Gestión de Requisitos.

2.4.4 Actividad IV: ESPECIFICACIÓN DE REQUISITOS

La especificación de requisitos, conocida también como definición de requisitos, es el modo habitual de guardar y comunicar requisitos (en composición grupal).

El *objetivo* de esta actividad es obtener un documento de especificación (ERS) que defina, de forma completa, precisa y verificable, los requisitos que debe cumplir el sistema, tanto funcionales como no funcionales, así como las restricciones aplicables al diseño (software y hardware). Debe abordar la descripción de lo que hay que desarrollar, no el cómo ni el cuándo. No debe incluir requisitos innecesarios, no solicitados por el cliente, ni incluir detalles sobre el diseño del sistema

Es aconsejable especificar los requisitos desde dos puntos de vista diferentes:

El DRU (Documento[4] de Requisitos de Usuario), que se escribe desde el punto de vista del usuario/cliente/otros interesados. Normalmente los requisitos de usuario contenidos en el DRU, no poseen demasiado nivel de detalle. Se incluye la descripción del problema actual (razones por las que el sistema de trabajo actual es insatisfactorio) y las metas que se espera lograr con la construcción del nuevo sistema.

[4] Con “Documento” nos referimos a cualquier medio electrónico de almacenamiento y distribución: Procesador de textos, Base de Datos

La especificación de requisitos software, que desarrolla mucho más los contenidos de la DRU. Los requisitos del software contenidos en la ERS son, por tanto, más detallados. Contiene la respuesta a la pregunta ¿Qué características debe poseer el sistema que nos permita alcanzar los objetivos y evitar los problemas contenidos en el DRU?

La especificación de requisitos es, a su vez, una herramienta de la Gestión de Requisitos, por cuanto una versión oficial de una ERS constituye la línea base de requisitos del proyecto concreto.

Para el desarrollo de la ERS deben utilizarse los Estándares. La autora recomienda el IEEE Std. 830, que es de fácil comprensión y el más utilizado en la PYME. También se puede utilizar el PSS-05 de la Agencia Espacial Europea (ESA).

La autora acepta la premisa de que una ERS perfecta no es posible y que su calidad es muy difícil de cuantificar. En general, una ERS de calidad **no garantiza** la ausencia de problemas, pero una ERS pésima garantiza su presencia.

Recomendaciones para la redacción de la ERS:

- Prestar atención a las conjunciones aseverativas (e j., ciertamente, por lo tanto, obviamente, s e puede ver que). En estos casos, preguntar porqué se usan.
- Prestar atención a los términos de significado vago (Ej. algunos, algunas veces, a menudo, usualmente, frecuentemente). Pedir aclaraciones.
- Cuando se dan listas, pero no completas, asegurarse de que todos los puntos se han comprendido. Ejemplos a tener en cuenta: "etc.", "…y en adelante ", "y así sucesivamente".
- Asegurarse de que los rangos no contienen asunciones no establecidas (e j., Rangos de código valido desde 10 a 100. ¿Enteros? ¿Binarios? ¿Hexadecimales?).
- Atención a los verbos de significado vago como "Mantenido, descartado, procesado, eliminado". Existen muchas formas en las que pueden ser interpretados.
- Atención a los pronombres ambiguos (Ej., el módulo de entrada/ salida se comunica con el módulo de validación de datos y su flag de control se activa. ¿El flag de qué módulo?)
- Buscar palabras que impliquen certeza (Ej., siempre, cada vez, todos, ningún, nunca), entonces pedir que se pruebe dicha certeza.
- Cuando se define un término en un lugar de forma explícita, probar a sustituir dicha definición por el término en otros lugares en los que aparezca.
- Cuando una estructura esté descrita con palabras, realizar un diagrama que ayude a su comprensión.

- Cuando se especifique algún cálculo, deben aparecer al menos dos ejemplos.

La estructuración de los requisitos en la especificación debe realizarse de forma que la organización de ellos ayude a los lectores (*stakeholders)* a comprender el sistema y a los escritores de requisitos (miembros del equipo de trabajo) a reubicar funciones. Puede hacerse organizado por subsistemas, grupos funcionales, funciones y subfunciones, clase de usuario, clase de objeto, clase de requisito. Recomendamos apoyarse en estándares actuales para la estructuración de los requisitos (p.e., IEEE Std. 830).

La especificación de los requisitos del software tiene, al igual que los requisitos individuales, sus propias características. Para que una ERS posea una característica dada, todos los requisitos individuales que contiene deben poseerla. De acuerdo con esto, las características fundamentales de la ERS son:

- Incluir información veraz y comunicarla de forma eficaz
- Describir correctamente todos los requisitos del software, no describir ningún detalle del diseño del software, de su verificación o de la dirección del proyecto. La ERS debe indicar qué, no cómo ni cuándo
- Ser concisa
- No debe incluir información que no proporcione valor en la comprensión de un requisito
- Prestar especial cuidado en la redacción de los requisitos
- Evitar repetir información que existe en otros documentos
- Incluir referencias a otros documentos productos del ciclo de vida, para lo cual se debe establecer un origen claro para cada uno de los requisitos (hay referencias hacia atrás), posibilitar la referencia de estos requisitos con productos de trabajo futuros dentro del ciclo de vida (hay referencias hacia adelante)
- Cada requisito debe tener un número de referencia único que lo identifique sobre los demás
- Debe ser fácilmente trazable
- Los requisitos deben estar clasificados por prioridades
- No ambigua. Un requisito ambiguo se presta a diferentes interpretaciones. Una ERS es no ambigua si, y solo si, cada requisito tiene una única interpretación.
- Completa: debe comprender la totalidad de los requerimientos: funcionales, no funcionales (seguridad, rendimiento, operación, documentación, fiabilidad, disponibilidad...etc.) así como restricciones de diseño y construcción (software y hardware)

- Debe definir la respuesta del software tanto para entradas válidas como no válidas
- Estar en conformidad con estándares y procedimientos definidos y aprobados en la organización
- Debe ser fácil de verificar (si y solo si para cada requisito existe un procedimiento finito y efectivo en coste para que una persona o máquina compruebe que el software satisface dicho requisito)
- Debe ser consistente. Una ERS es consistente si, y sólo si, ningún conjunto de requisitos son contradictorios o entran en conflicto. Aparecen conflictos si dos o más requisitos describen el mismo objeto real, pero utilizan términos distintos para designarlo. Puede aparecer conflicto en las características especificadas sobre los objetos reales (p.e., un requisito establece que todas las luces han de ser azules y otro verdes), lógico o temporal entre dos acciones determinadas (p.e., un requisito establece la suma de dos entradas y otro la multiplicación).
- Debe ser fácil de modificar, que su estructura y estilo permitan que cualquier cambio se pueda realizar fácil, completa, y consistentemente.
- Debe ser mínimamente redundante. Cada requisito debe aparecer sólo en un lugar.
- Es posible la redundancia con objeto de facilitar la compresión de los requisitos, pero supone problemas de consistencia al modificar. Mejor crear referencias cruzadas entre los requisitos
- Fácil para identificar el origen de cada requisito.
- Fácil de utilizar durante las fases de explotación y mantenimiento.
- El documento ERS establece con precisión las funciones y capacidad del software así como sus restricciones. Especifica la conducta externa del sistema y los límites de la implementación.
- Debe servir como una herramienta de referencia para mantenimiento.

Dentro de la ERS los requisitos deben estar estructurados. Organización por subsistemas, grupos funcionales, funciones y subfunciones. Por clases: de usuario, de estímulo, de objeto, de requisito.

Hay que eliminar de la ERS los requisitos no verificables, o reescribirlo de forma que sea verificable. Para los requisitos incompletos se definen de las causas y acciones para cambiar su estado y seguimiento.

La ERS es la base para toda subsecuente planificación, diseño, y codificación, así como para las pruebas del software y documentación del usuario.

Los desarrolladores y clientes no deben realizar presunción alguna. Si cualquier requerimiento funcional o no funcional no es identificado en la ERS, no es parte del acuerdo y, por lo tanto, nadie debe esperar que aparezca en el producto final.

Técnicas de especificación

Son diversas y se clasifican según la forma de representación en *gráficas (*utilizan elementos gráficos para representar componentes particulares de modelos), *textuales* (especifican con más detalle los componentes definidos en los gráficos mediante una gramática concreta) y *marcos* (o plantillas:<templates>). Según el enfoque de modelado se clasifican en técnicas de información (por la información utiliza el sistema), de función (qué hace el sistema) y de tiempo (cuándo sucede algo en el sistema)

Para la actividad especificación en el proceso de ingeniería de requisitos hay un gran número de técnicas propuestas, las que aparecen en el Anexo 1.

Los requisitos que hayan podido ser considerados en el borrador de la ERS por problemas en sus propiedades que no fueron resueltas hasta este momento se devolverán a las actividades anteriores, con las consecuentes anotaciones en la tarjeta.

La *salida* de esta actividad es el borrador del documento de especificación de requisitos, que constituye a su vez la *entrada* a la actividad de validación de requisitos. También constituye entrada a las actividades de Gestión de Riesgos y Gestión de Requisitos.

2.4.5 Actividad V: VALIDACIÓN DE LOS REQUISITOS

La *validación de los requisitos* es el proceso para comprobar que la ERS se ajusta a las necesidades de clientes/usuarios y otros implicados, o sea, es comprobar con los *stakeholders* que sus necesidades fueron adecuadamente interpretadas. Valida el cliente.

Dentro del proceso de validación también se desarrollan actividades de verificación de requisitos, con el fin de comprobar que la ERS se construye de acuerdo a los criterios y estándares establecidos, o sea, correctamente. La validación no puede hacerse sin la participación y presencia de clientes, usuarios y demás implicados.

Los *objetivos* de esta actividad son la comprobación de la consistencia, completitud, corrección, precisión del documento, así como el descubrimiento de problemas en él antes de comprometer recursos en su implementación.

La ERS debe revisarse para descubrir omisiones, conflictos, ambigüedades, para comprobar la calidad del documento y su grado de adhesión a estándares, normas o procedimientos establecidos. Para este fin pueden utilizarse varias técnicas de revisión (formales- con los documentos completos, o informales), entre ellas acciones correctivas para eliminar defectos y listas de comprobación. Un ejemplo de esta última puede incluir preguntas como las siguientes:

¿Satisfacen los requisitos incluidos en ella las necesidades del Cliente?

¿Se tienen los requisitos adecuados?

¿Están completos?

¿Están bien descritos?

¿Se ajustan a los estándares?

¿El Documento representa una clara descripción del sistema?

Es importante que los implicados en el proceso de IR entiendan por qué es necesario revisar los requisitos cuanto antes. Entre los fundamentos podemos mencionar que el costo de corrección de defectos de requisitos es del orden de 100 a 200 veces inferior en las etapas iniciales. La corrección de requisitos defectuosos implica mucho retrabajo sobre otros productos del proyecto (diseño, construcción, etc.) que se desarrollan a partir de las especificaciones. Una buena comunicación entre desarrolladores, clientes y usuarios puede resolver problemas en las primeras etapas.

Los *métodos* empleados para validar los requisitos son:

- Las revisiones son la fórmula más empleada para validar requisitos.

La autora considera que es recomendable organizar un grupo de personas designadas, incluyendo clientes, usuarios, e informáticos, miembros o no del equipo de trabajo, que realicen una revisión de la ERS (leer, analizar y discutir el documento).

Previamente algunos miembros del equipo deben realizar una pre-revisión para corregir errores ortográficos, cotejar con estándares. Después se desarrolla la búsqueda de problemas, para identificar los más habituales, se pueden utilizar *checklist* que incluyan:

¿Cada requisito tiene un único identificador?

¿Están definidos los términos técnicos en el Glosario?

¿El requisito se comprende sin tener que recurrir a otros?

¿Requisitos diferentes utilizan el mismo término de maneras distintas?

Después se realiza una reunión del grupo de Revisión

Concluidas estas acciones el equipo se reúne y se toman acuerdos respecto a los requisitos aprobados que se escribirán, con sus detalles, en la ERS, de acuerdo a la norma seleccionada.

- Prototipado.

Permite descubrir con rapidez si el usuario se encuentra satisfecho o no con los requisitos. Se puede escribir un manual de usuario para el prototipo desarrollado, que es una forma más de validar requisitos.

Este es un método muy útil pero muy poco utilizado en nuestro entorno. Muchos de los implicados lo consideran una pérdida de tiempo innecesaria, lo que la autora considera una apreciación errónea. Para resolver este problema la autora recomienda comenzar su utilización en proyectos sencillos y pequeños, primeramente con representaciones planas en papel, diseños visuales sin funcionalidad, hasta que se comprenda la utilidad del método.

- Validación de Modelos: cuando los requisitos se expresan por medio de modelos (de objetos, DFDs, etc.)
- Pruebas de requisitos.

Es deseable diseñar pruebas sobre cada requisito individual, si hay dificultades para ello, esto puede implicar que hay ambigüedad o falta información. Esto no es aplicable a requisitos que indiquen propiedades globales del sistema. El personal designado para realizar las pruebas debe revisar los requisitos.

Todos los problemas que se detecten durante las acciones de verificación y validación de los requisitos y que no puedan ser resueltos en breve tiempo, o con facilidad por el equipo de trabajo, se anotarán en *la lista de problemas* en los requisitos no validados. De esta se desprenderá el análisis de las acciones a desarrollar para resolverlos y la toma de decisiones al respecto. Se confeccionará entonces un plan o *lista de acciones* que contendrá las medidas para resolver los problemas detectados, entre ellas:

- Clarificar requisitos mal expresados.
- Identificar y añadir la información faltante en el requisito.
- Negociar los conflictos detectados.
- Eliminación o modificación de requisitos no realistas.

En la tabla 2.1 se resumen las mejores prácticas en el desarrollo de requisitos.

Una *salida* de este proceso de validación de requisitos es la lista de problemas y de acciones para resolverlos, que **implica un retorno** a actividades anteriores del proceso de IR. Obsérvese la figura 2.3.

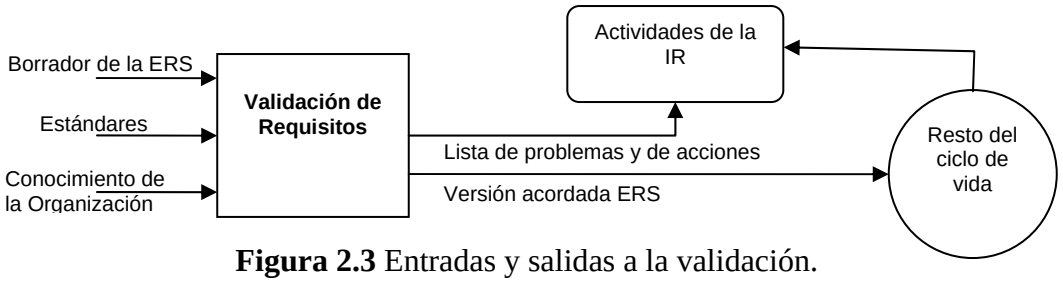


Figura 2.3 Entradas y salidas a la validación.

La ***salida*** principal de esta actividad es el documento de especificación de requisitos (ERS), que es la entrada a la siguiente actividad del proceso de desarrollo del software: el diseño del software. También constituye entrada a las actividades de Gestión de Riesgos y Gestión de Requisitos.

2.4.6 Actividad VI: GESTION DE RIESGOS

Un riesgo es aquel factor que influye negativamente en el éxito del proyecto. El riesgo en un proyecto de desarrollo de software incluye componentes técnicos y de conocimiento del mismo. Los temas de naturaleza organizacional constituyen los factores dominantes de los riesgos del proyecto, a la vez que son los que se tratan satisfactoriamente en menos de la tercera parte de los proyectos de desarrollo, entre ellos los conflictos entre departamentos, entre usuarios, el cambio del responsable ejecutivo del proyecto, volatilidad del personal, número de unidades de la organización implicadas y proyectos que involucran a múltiples proveedores.

Es premisa de esta propuesta que el riesgo se halla, de forma implícita, asociado a toda actividad. El riesgo acompaña a todo cambio porque implica elección e incertidumbre. Si a la vez que se inicia la actividad de elicitación de los requisitos del software a construir, se inicia la identificación de los riesgos asociados a los requisitos individuales y a grupos de ellos, será posible gestionarlos tempranamente para minimizarlos, evadirlos y controlarlos.

El jefe o administrador de proyectos anticipa riesgos que pueden afectar al desarrollo o a la calidad de los requisitos y emprende acciones para evitarlos.

En esta actividad se garantiza que PROCIR permita, desde el inicio del proceso de desarrollo del software, realizar las tareas encaminadas a garantizar la calidad del producto. Basado en el modelo dado por [Pressman, 2002], descrito en el Capítulo I y representado en la figura 1.13, se define el procedimiento siguiente:

Paso 1*: Identificación de riesgos*. Problemas potenciales que pueden ocurrir en el proceso de IR o en los requisitos, o en la ERS, como problemas de presupuesto, problemas de personal, problemas del usuario, problemas de organización, problemas técnicos, problemas de comunicación.

Debe comenzar con el análisis de los riesgos genéricos, que constituyen una amenaza potencial para todos los proyectos de software, que puedan estar presentes en el proyecto en curso. Después se deben identificar los riesgos específicos, que implican un conocimiento profundo del proyecto, y están relacionados con el entorno de desarrollo, la tecnología, la experiencia y el tamaño del equipo.

Para los requisitos y las ERS se debe comenzar esta tarea dando respuesta a la pregunta: ¿qué características especiales tiene este requisito, o documento de ellos, que pueden estar amenazadas?

Tabla 2.1 Resumen de buenas prácticas en el desarrollo de requisitos.

Captura	**Análisis**	**Especificación**	**Validación**
Definir un proceso de desarrollo de requisitos.	Dibujar un diagrama de contexto	Adoptar una plantilla de ERS	Inspeccionar los documentos de requisitos
Definir visión y alcance	Crear prototipos	Identificar los orígenes de los requisitos	Probar los requisitos
Identificar clases de usuarios	Analizar la viabilidad	Etiquetar de modo único cada requisito	Definir criterios de aceptación
Establecer grupos enfocados	Priorizar requisitos	Registrar las reglas de negocio	
Identificar casos de uso	Modelar los requisitos	Especificar atributos de calidad	
Identificar eventos y respuestas del sistema	Crear un diccionario de datos		
Mantener *workshops* de captura	Asignar requisitos a subsistemas		
Observar a los usuarios en la ejecución de su trabajo	Aplicar QFD (Quality Function Deployment)		
Examinar informes de problemas			
Reutilizar requisitos			

Un método probadamente efectivo para identificar riesgos es la creación de una *lista de comprobación de elementos de riesgo*. Esta lista debe centrarse en los riesgos relacionados con: tamaño del producto, impacto en el proyecto y en la organización, características del cliente, definición del proceso, el entorno de desarrollo, la tecnología a construir, el tamaño del equipo y la experiencia del personal.

En esta actividad del procedimiento se establece crear una lista de comprobación para los requisitos individuales y otra para la ERS. Los resultados se vierten en tablas para facilitar el análisis posterior. Los riesgos serán considerados a partir del comportamiento de las características individuales y grupales. En dependencia de la intensidad de ese comportamiento de la característica de calidad del requisito, podrá controlarse de forma efectiva el riesgo en el propio proceso de IR, antes de que pase a la siguiente etapa del ciclo de vida del proyecto de desarrollo del software. De no gestionarse el riesgo en esta etapa inicial, puede hacerse difícil su control efectivo una vez iniciado el proceso de desarrollo. Desequilibrios en el comportamiento de las características de calidad de los requisitos se traducen en que la aparición de riesgos haga vulnerable el proyecto e impráctica la aplicación de cualquier plan de calidad, razón por la cual conviene su detección y tratamiento temprano. En esta situación, se recomienda la redefinición de los requisitos afectados (o la ERS), que es posible por iteraciones en este procedimiento.

Un ejemplo de la aplicación de este instrumento es:

Característica del requisito: ambigüedad.

Pregunta: ¿es ambiguo el requisito?

Aplicada a todos los requisitos especificados, la tabla 2.2 muestra el resultado.

En este procedimiento se establece definir y listar los riesgos que pueden afectar al propio proceso de ingeniería de requisitos como aparece en las tablas 2.3, 2.4 y 2.5.

Tabla 2.2 Identificación de riesgos.

ID requisito	Respuesta	Impacto	Componente	Medida
R1	SI	SI	Costo	Crítica
R2	NO	NO		

Algunos riesgos que pueden afectar el desarrollo del proceso de IR son:

- Sobrepasar los límites de los recursos asignados.
- Finalización fuera de plazos originales (a veces ni se finaliza).
- Pobre o descontrolada gestión de los requisitos.
- Incompatibilidad con el entorno.

- Riesgo más grave: que no se comprendan y no se satisfagan las necesidades de los usuarios.
- Problemas en la comunicación entre clientes y proveedores, entre usuarios, u otros grupos.

***Paso 2**: Evaluación de los riesgos.* Determinar en qué indicador se verá reflejado que un problema se presente, se deben establecer puntos de referencia para cada riesgo, que permita decidir si el riesgo, según su prioridad de atención, se sale del manejo aceptable. Obsérvese en la figura 2.4 un ejemplo de un punto de referencia.

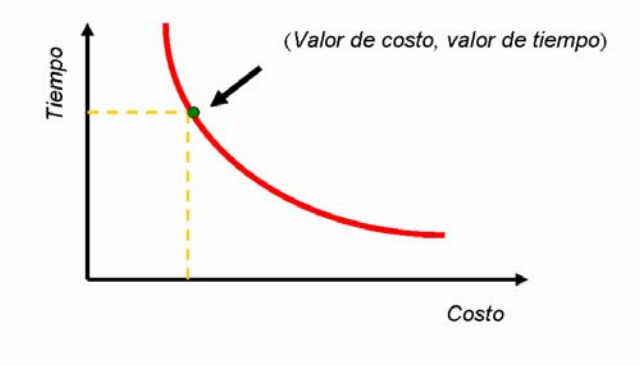


Figura 2.4 Ejemplo de punto de referencia.

Proyección de los riesgos. Consiste en determinar la probabilidad de que un riesgo ocurra y las consecuencias que puede tener, por ejemplo: incremento de costos, cancelación del proyecto, insatisfacción del cliente.

Implica ordenar la lista de riesgos teniendo en cuenta la probabilidad de que ocurra y el impacto de cada riesgo. Se asigna el nivel de probabilidad, que puede ser alta, media o baja. Se valora el impacto (consecuencias) en cuanto al alcance (cuánto se afecta) y la duración (por cuánto tiempo se manifiesta).

En el análisis de riesgos se considera cada riesgo por separado y se valora en intervalos su probabilidad e impacto:

- probabilidad del riesgo valorada como *muy bajo* (<10%), *bajo* (10-25%), *moderado* (25-50%), *alto* (50-75%) o *muy alto* (>75%)
- efectos del riesgo valorados como *catastrófico, serio, tolerable o insignificante*

El resultado se registra en una tabla ordenada por la probabilidad o por el efecto del riesgo. Se decide del total, cuáles son los más importantes, considerados entonces como riesgos claves durante el proyecto -debe ser un número manejable. Por ejemplo, todos los serios o catastróficos con cualquier probabilidad. Obsérvese la tabla 2.6.

Tabla 2.3 Lista de riesgos y clasificación.

Riesgo	Tipo de riesgo	Descripción
Rotación de personal	Proyecto, producto y negocio	Personal con experiencia abandona el proyecto antes de que finalice
Cambios de requisitos	Proyecto y producto	Existencia de más cambios de requerimientos de los previstos inicialmente
Retrasos en la especificación	Proyecto y producto	Retrasos en las especificaciones de interfaces esenciales
Subestimación del tamaño	Proyecto y producto	El tamaño del requisito (la ERS, del proceso de IR) se ha subestimado
Bajo rendimiento de la herramienta CASE	Producto	Las herramientas CASE que ayudan al proyecto no tienen el rendimiento y las funcionalidades esperadas

Tabla 2.4 Riesgos por requisitos.

ID Requisito	Tipo de Riesgo	Riesgos
R1	De personal	El personal no cuenta con los conocimientos requeridos para enfrentar la complejidad del requisito
		Miembros del equipo no disponibles en momentos críticos
	De los requisitos	Cambios de requisitos que precisan modificaciones en el diseño
		Los clientes no comprenden el impacto de los cambios en los requerimientos
	De estimación	El tiempo requerido para desarrollar el proceso de ingeniería de requisitos está subestimado
	De comunicación	El cliente no pueda participar en revisiones y en reuniones

Tabla 2.5 Riesgos por tipos.

Tipo de riesgo	Posibles riesgos
Tecnología	Los componentes de software a reutilizar tienen defectos que limitan su funcionalidad.
Personal	Imposible contratar personal con los conocimientos requeridos.
Organizativos	La organización se reestructura y una nueva administración se responsabiliza del proyecto.
Herramientas	Las distintas herramientas CASE no están disponibles
Requerimientos	Cambios de requerimientos que precisan modificaciones en el diseño.
Estimación	El tamaño del sistema a desarrollar está subestimado.

Tabla 2.6 Riesgos ordenados por efectos.

Riesgo	Probabilidad	Efectos
Los problemas financieros de la organización reducen el presupuesto del proyecto	baja	catastrófico
Imposible contratar personal con los conocimientos requeridos	alta	catastrófico
Personal clave enfermo o no disponible en momentos críticos	moderada	serio
Cambios de requerimientos que precisan modificaciones en la codificación	moderada	serio
El tiempo requerido para desarrollar el proceso de IR está subestimado	alta	serio
Los clientes no comprenden el impacto de los cambios en los requerimientos	moderada	tolerable

Paso 3: *Planificación de riesgos.* Este paso tiene como objetivo desarrollar una estrategia para tratar los riesgos. Si el equipo de trabajo adopta un enfoque proactivo frente al riesgo, evitarlo será siempre la mejor estrategia. Esto se consigue desarrollando los planes de reducción del riesgo y de contingencia.

En la planificación de riesgos se considera cada uno de los riesgos claves identificados y las estrategias para administrarlos, que vendrán dadas por el juicio y la experiencia del administrador del proyecto. Las estrategias de anulación intentan reducir la probabilidad de que surja el riesgo, las estrategias de disminución: intentan reducir el impacto del riesgo. Los planes de contingencia se elaboran para estar preparados por si el riesgo ocurre poder actuar con una estrategia determinada. Obsérvese la tabla 2.7.

Paso 4: *Supervisión de los riesgos*. Consiste en hacer el plan de supervisión, dado el caso de que se acepte continuar con el proyecto. Indicar que acciones y decisiones se tomarán ante un problema que ya ha sido identificado, proyectado y evaluado.

Tabla 2.7 Estrategias por riesgos.

Riesgo	Estrategia
Problemas financieros de la organización	Preparar un documento breve para la dirección de la empresa que muestra que el proyecto hace contribuciones muy importantes a las metas del negocio
Problemas de reclutamiento	Organizar cursos de capacitación para el personal ya existente, investigar la posibilidad de contratar en otras regiones del país
Enfermedad del personal	reorganizar el equipo de tal forma que se solapen el trabajo y los miembros comprendan el trabajo de los demás
Cambios en los requisitos	Rastrear la información para valorar el impacto de los requerimientos, maximizar la información oculta en ellos
Tiempo de la IR subestimado	Alertar al cliente de las dificultades potenciales y las posibilidades de retraso

La supervisión de riesgos valora cada uno de los riesgos identificados para decidir si es más o menos probable y cuándo han cambiado sus posibles efectos. Hay que controlar factores que pueden indicar cambios en la probabilidad y el impacto. Obsérvese la tabla 2.8.

Tabla 2.8 Indicadores potenciales por riesgos.

Tipo de riesgo	**Indicadores potenciales**
Tecnología	Entrega retrasada del hardware o del software. Existencia de informes sobre problemas tecnológicos.
Personal	Baja moral del personal, malas relaciones entre miembros del equipo, plazas vacantes
Organizacional	Rumores en la empresa Falta de iniciativa de la dirección de la empresa
Herramientas	Rechazo de los miembros del equipo a utilizar herramientas Quejas sobre las herramientas CASE Peticiones de estaciones de trabajo más potentes
Requisitos	Peticiones de muchos cambios en los requisitos Quejas del cliente
Estimación	Fracaso en el cumplimiento de los tiempos planificados

Gestión de Riesgos

- La tarea principal del administrador consiste en minimizar riesgos.
- Involucrar a todos los *stakeholders* y al equipo de desarrollo del proyecto en la identificación de los riesgos.
- El riesgo inherente en una actividad se mide en base a la incertidumbre que presenta el resultado de esa actividad.
- Las actividades con alto riesgo elevan los costos.
- El riesgo es proporcional al monto de la calidad de la información disponible. Cuanto menos información, mayor el riesgo.
- Monitorear constantemente los factores que propician la materialización de los riesgos y la efectividad de las acciones definidas encaminadas a su prevención y/o minimización.
- Comunicar los riesgos a todos los niveles de la organización.
- Definir plantillas o tablas de riesgos valorados para diferentes tipos de proyecto que puedan servir como punto de partida para un nuevo proyecto.

Las *salidas* de esta actividad son las listas (tablas) de riesgos en sus diferentes acepciones, el plan de supervisión de riesgos y el plan de contingencia.

2.4.7 Actividad VII: GESTIÓN DE REQUISITOS

Los principios de esta actividad son:

- El acuerdo de los requisitos es el puente entre el desarrollo de requisitos y la gestión de requisitos.
- La gestión de requisitos incluye todas las actividades para mantener la integridad, exactitud y difusión de los acuerdos de los requisitos durante la vida del proyecto.

Los requisitos cambian y esto persiste a lo largo de la vida del sistema, por cambios en las estrategias o prioridades del negocio, por cambios tecnológicos, en leyes y/o regulaciones. Los cambios deben controlarse y documentarse. Hay que convivir con el cambio.

La Gestión de Requisitos es el conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y sus cambios en cualquier momento. O sea, básicamente, consiste en gestionar los cambios a los requisitos acordados, las relaciones entre ellos, las dependencias entre la ERS y otros documentos producidos por el proceso de desarrollo de software. Esta actividad asegura la consistencia entre los requisitos y el sistema construido (o en construcción). Consume grandes cantidades de tiempo y esfuerzo. Abarca todo el ciclo de vida del producto. Es una actividad necesaria porque:

- Los requisitos son volátiles.
  - Mutantes o cambiantes: sufren ligeras variaciones.
  - Emergentes: surgen al ir analizando el sistema en profundidad.
  - Colaterales: surgen como efecto de la inclusión de otros requisitos.
  - Por compatibilidad: se añaden para adaptar el sistema a su entorno, debido a que el entorno físico cambia, o sea, trasladar un sistema de un entorno a otro siempre requiere modificaciones. El entorno organizacional también cambia, las políticas cambian, se producen cambios en las reglas y en los procesos del negocio, provocando cambios en el sistema.
- La propia existencia del sistema va a generar nuevos requisitos por parte de los usuarios.

La gestión de requisitos implica:

- Definir procedimientos que establezcan los pasos y los análisis que se realizarán antes de aceptar los cambios propuestos.
- Cambiar los atributos de los requisitos afectados.
- Mantener la trazabilidad hacia atrás, hacia delante y entre requisitos.

- Controlar las versiones del documento de requisitos.

Como ya se ha expuesto en el capítulo anterior es vital planificar los cambios, de acuerdo a las prioridades que los clientes determinen durante la identificación. Hay que aprobar los mecanismos para la configuración del cambio y las políticas de trazabilidad.

Procedimiento de gestión de cambios en los requisitos

Desde el inicio hay que establecer la conformación de la línea base de requisitos, como un canal simple para el control de cambios, que se podrá usar para "capturar" nuevos cambios.

Línea base de requisitos

Es el conjunto de requisitos funcionales y no-funcionales que el equipo del proyecto se ha comprometido a implementar en una *release* específica. Es una versión aprobada de la especificación de requisitos del software.

Los requisitos antes de entrar en la Línea Base deben ser sometidos a un procedimiento de revisión formal (con su aprobación para ser incluido en la línea base). Una vez entrado el requisito en la línea base cualquier cambio debe someterse al procedimiento de control de cambios.

Es adecuado que el documento de requisitos que se esté elaborando previo a entrar en la línea base esté sometido a un procedimiento de control de versión (para distinguir versiones borradores de versión aprobada).

Canal y control de cambios

La identificación del cambio se inicia desde el análisis de requisitos, nuevas necesidades del cliente, problemas operacionales; después se realiza el análisis del cambio y evaluación de costos, debiendo quedar claro cuántos requisitos se verán afectados y cuánto costará (tiempo y recursos); sólo después se toma la decisión de la implementación del cambio, que será documentado al ser realizado. Obsérvense las figuras 2.5 y 2.6.

La gestión de los cambios se realiza de acuerdo a las prioridades, debiéndose identificar problemas y causas por los que se produce el cambio; analizar el impacto y el costo del cambio y finalmente, tomada la decisión, ejecutar (proceder con) el cambio.

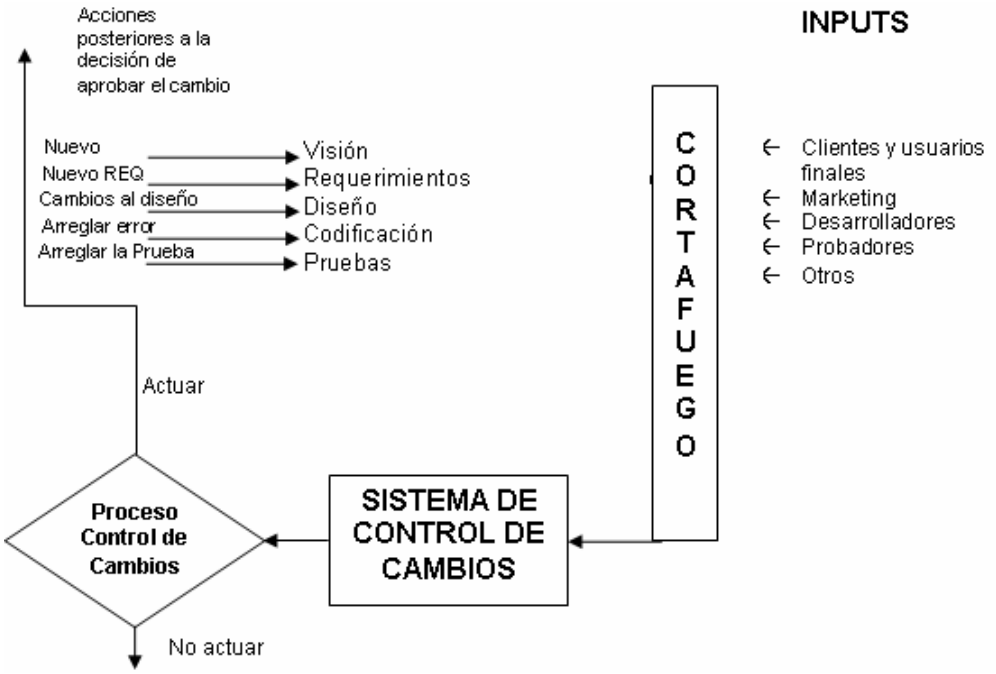


Figura 2.5 Canal de cambios.

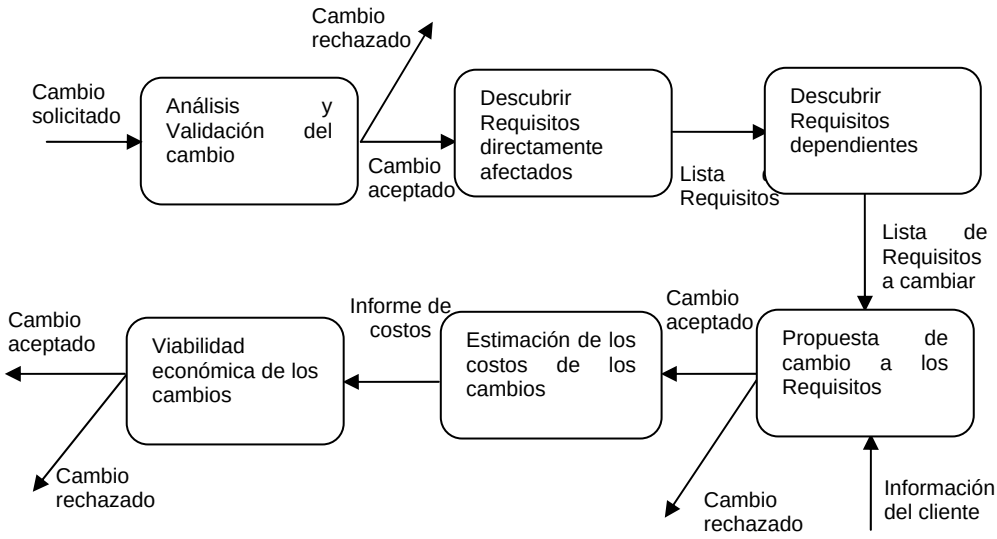


Figura 2.6 Proceso de control de cambios.

Aspectos a considerar para la aprobación de un cambio solicitado:

- Impacto del cambio en el costo y funcionalidades del sistema.
- Impacto para el cliente y *stakeholders* externos.
- Desestabilización potencial del sistema que pudiera ocasionar un cambio.

Las solicitudes de cambios, desde el momento en que son comunicadas hasta su implementación, transitan por diferentes estados en el que intervienen los implicados asumiendo diferentes roles. Obsérvense la figura 2.7 y la tabla 2.8.

Ejecución de los cambios

Aprobado el cambio se procede a su implementación, de acuerdo a la fase del proyecto a que corresponda. En caso de que los cambios aprobados:

- impliquen el desarrollo de un nuevo sistema, entonces será necesario comenzar un nuevo proceso de IR.
- impliquen la implementación de nuevos requisitos, entonces será necesario comenzar por la actividad de elicitación de PROCIR.
- afecten otras fases del proceso de desarrollo del proyecto, entonces se implementan en estas.

Trazabilidad

Los requisitos deben ser "rastreables": por su origen (¿quién lo propuso?); necesidad (¿por qué existe?); por su relación con otros requisitos; por su relación con elementos del diseño y/o la implementación. Esta información es útil para saber qué afecta un cambio en un requisito. Para resolver esta necesidad se usan las matrices para el seguimiento de los requisitos**,** entre ellas:

- Matriz de seguimiento de características (definidas por el cliente).
- Matriz de seguimiento de orígenes: identifica el origen de cada requisito.
- Matriz de seguimiento de dependencias: indica cómo se relacionan los requisitos entre sí.
- Matriz de seguimiento de subsistemas: vincula a los requisitos con los subsistemas (o módulos) que los manejan.
- Matriz de seguimiento de interfases: muestra cómo los requisitos están vinculados a las interfases internas y externas del sistema.

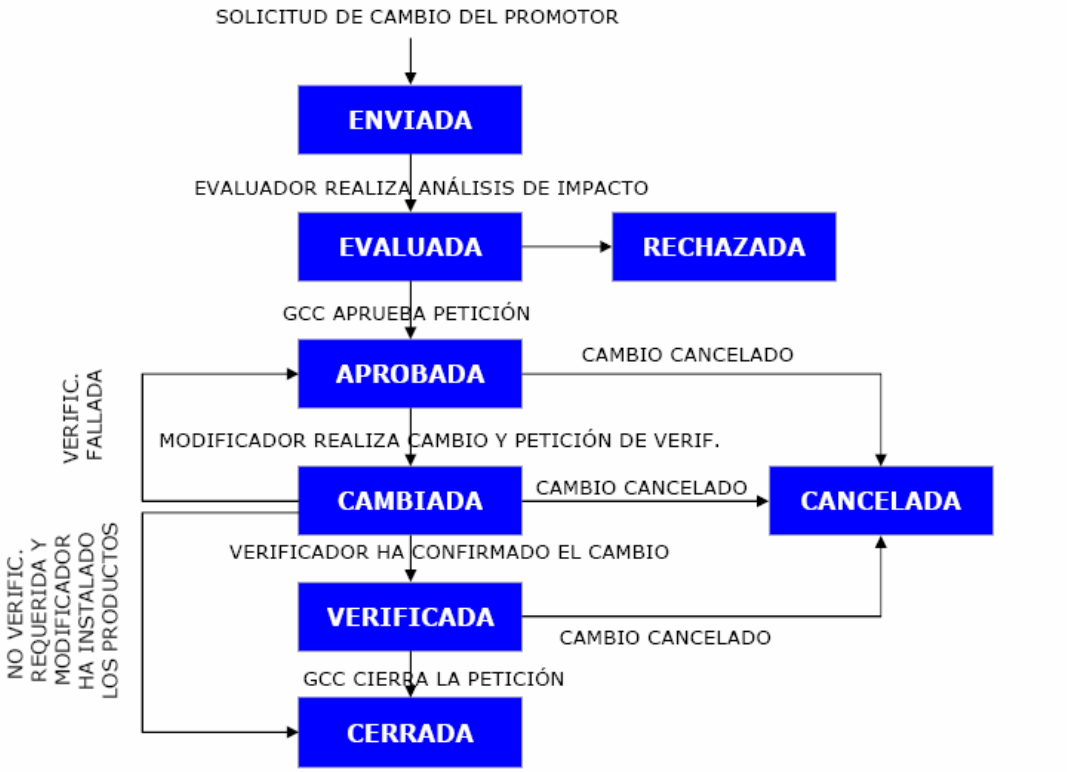


Figura 2.7 Posibles estados de una petición de cambio.

Tabla 2.9 Posibles roles en el procedimiento de control de cambios.

Rol	**Descripción**
Grupo de control de cambios (GCC)	Grupo de personas que deciden aprobar o rechazar las peticiones de cambios para un proyecto específico.
Promotor del cambio	Persona autorizada que solicita la petición de cambio de requisitos.
Evaluador	Persona que analiza el impacto de la petición de cambio (puedes ser técnico, martketing, cliente o combinación).
Modificador	Persona que realiza el cambio como consecuencia de una petición de cambio aprobada.
Verificador	Persona que determina si el cambio se ha realizado correctamente.
Validador	Persona del cliente que valida la implementación del cambio realizado.

En [Edwards, 1991] la trazabilidad está definida como una técnica usada para "proveer una relación entre requisitos, el diseño y la implementación final del sistema". En [Palmer, 1997] se establece que "la trazabilidad da asistencia esencial en el entendimiento de las relaciones que existen entre y a través los requisitos, el diseño y la implementación".

Estas relaciones permiten mostrar que el diseño cumple con los requisitos funcionales y ayudan al reconocimiento temprano de aquellos requisitos que no son satisfechos por el diseño.

Control de versiones (evolución en el tiempo de la ERS)

Habitualmente la ERS necesitará ser modificada a medida que progresa el producto software, como resultado de los cambios aprobados en requisitos individuales o grupos de ellos. En la medida en que la especificación sea más completa, como consecuencia de que sus requisitos fueron especificados lo más completamente posible, menos versiones de la ERS habrá que producir.

Para la trazabilidad y el control de versiones se realiza la identificación y almacenamiento de requisitos. Cada requisito y cada versión de ERS aprobada tendrán un identificador único.

Cuando la Gestión de los Requisitos no se hace bien, de inmediato aparecen síntomas:

- Altos niveles de re-trabajo a lo largo del proyecto.
- Los requisitos se aceptan por el personal desde cualquier fuente que consideren fidedigna.
- Se incrementa de forma desmesurada las peticiones de requisitos.
- Incapacidad para probar que el producto cumple los requisitos aprobados.

¿Por qué se debe tener cuidado? Porque:

- La falta de acuerdo entre los afectados sobre cuáles son los requisitos "reales" incrementa el tiempo y el costo del proyecto.
- Hay altas probabilidades de entregar un producto incompleto o incorrecto.
- Volver a revisar cambios en los requisitos, una y otra vez, es un derroche de recursos muy visibles para el cliente.

Buenas prácticas de la actividad de gestión de requisitos.

- Definir un proceso de control de cambios.
- Establecer un grupo (o comité) de control de cambios.
- Realizar análisis del impacto sobre los cambios.
- Crear líneas base y controlar las versiones de los requisitos.

- Mantener la historia de los cambios.
- Seguir el estado de los requisitos.
- Medir la volatilidad de los requisitos.
- Usar herramientas de gestión de requisitos.
- Crear matrices de trazabilidad de los requisitos.

El proyecto puede responder frente a petición de nuevos requisitos o petición de cambios en los requisitos de diferentes maneras:

- Diferir los requisitos de baja prioridad.
- Obtener recursos adicionales.
- Ordenar realizar más horas de trabajo (preferiblemente durante un periodo corto de tiempo y pagado).
- Retrasar el calendario para acomodarse a la nueva funcionalidad.
- Permitir reducir el nivel de calidad bajo la presión de mantener la fecha original (frecuentemente esta es la reacción por defecto).

Ninguna aproximación es universalmente correcta porque cada uno de los proyectos es diferente (características, presupuesto, calendario, recursos y calidad).

Independientemente a cómo se responda a los cambios de los requisitos, lo que realmente importa es ajustar las expectativas y los compromisos cuando sea necesario.

Las ***salidas*** de esta actividad son los cambios aprobados y, no aprobados, informes acerca del estado del proceso, de actividades o requisitos, resultados de análisis de matrices y otros.

2.5 Conclusiones parciales

- El procedimiento PROCIR tiene características proactivas, porque se prevé la gestión de los requisitos y sus riesgos desde las etapas más temprana del ciclo de vida.
- PROCIR, independientemente de ser un procedimiento para el desarrollo del proceso de ingeniería de requisitos en un proyecto software, contiene aspectos de gestión de la calidad, de gestión de proyectos y aspectos organizacionales como la definición de equipos de trabajo, la definición de un conjunto de procedimientos para esa área del proceso de software y su relación con otras.
- El procedimiento para la gestión del riesgo, está basado en las características de los requisitos y la especificación de ellos. En dependencia de la intensidad del comportamiento de las características

de calidad del requisito o grupo de ellos, podrá controlarse de forma efectiva el riesgo en el propio proceso de IR, antes de que pase a la siguiente etapa del ciclo de vida del proyecto de desarrollo del software. PROCIR gestiona tempranamente los riesgos para minimizarlos, evadirlos y controlarlos.

- PROCIR está sustentado en la definición de condiciones de inicio para este proceso, en un nuevo enfoque funcional del equipo de trabajo, en la gestión de los requisitos y sus cambios a lo largo de todo el ciclo de vida del proyecto y en ser independiente de la metodología de desarrollo adoptada.
- La naturaleza iterativa e incremental de PROCIR garantiza el mejoramiento gradual de los niveles de desempeño actuales del proceso de ingeniería de requisitos, del proceso de desarrollo y la gestión de un proyecto de software, propiciando la satisfacción del cliente en la PyME.
- PROCIR garantiza mayor calidad en la Especificación de los Requisitos del Software, lo que favorece que la fase de planificación y estimación del proyecto sea más objetiva y realista.
- La implementación del procedimiento para el desarrollo del proceso de ingeniería de requisitos permite a la PyME obtener y gestionar especificaciones de requisitos de software completas, formales y acordadas entre todos los participantes, que cumplen con las características calidad definidas.

Capitulo 3. Aplicación práctica del procedimiento para el desarrollo del proceso de ingeniería de requisitos en un proyecto software

El procedimiento PROCIR requiere, por un lado, de su aplicación práctica en la fase de ingeniería de requisitos de un proyecto real, que en este caso será en el Sistema de Combustible de la Gerencia Territorial de ETECSA Cienfuegos. Por otro lado, de una validación de la aplicabilidad de todas las "herramientas" propuestas en el procedimiento, a partir de que este posea un conjunto de características previamente definidas.

3.1 Aplicación de PROCIR al sistema de combustible de ETECSA

El procedimiento se aplica en el contexto de la Subgerencia de Tecnología y Software (TISW) de la gerencia Cienfuegos. ETECSA no es una empresa productora de software, pero tiene una Unidad de Negocios de mediano tamaño dedicada a la producción, adquisición y parametrización de software a la medida, principalmente para uso interno. En Cienfuegos TISW cuenta con un equipo de trabajo de seis profesionales del software.

La decisión de aplicar PROCIR al sistema de combustible se fundamentó en la necesidad de desarrollar una nueva versión de este producto, por los problemas que venía presentando la que ya estaban explotando. Este sistema funcionaba en la Gerencia desde hacía un año, cuando se automatizó el sistema manual del control de las tarjetas y del consumo de combustible. La versión en explotación fue desarrollada por una especialista informática de la propia provincia.

A los tres meses los usuarios comenzaron a solicitar cambios para adicionar y mejorar funcionalidades al sistema. Se desarrollaron dos adaptaciones sucesivas que, lejos de ayudar a resolver las solicitudes, comenzaron a desencadenar algunos reportes de error.

Ante esta problemática se decidió desarrollar una nueva versión del sistema que resolviera los problemas que se estaban presentando y que incluyera los cambios solicitados.

Al iniciarse el proyecto se decidió aplicar PROCIR, comenzando el desarrollo del proceso de ingeniería de requisitos.

3.1.1 Actividad I: CONDICIONES DE INICIO

Dadas las condiciones preexistentes en la Gerencia, relacionadas con este proyecto, no fue necesario aplicar literalmente todas las tareas de esta actividad.

Tarea 1 Decisión de llevar a cabo el Proyecto: esta acción se tomó antes de aprobarse la aplicación de PROCIR en el nuevo proyecto.

Tarea 2 Constitución del Equipo de Trabajo de Gestión de Requisitos.

Se creó un equipo de trabajo conformado por siete miembros: dos especialistas informáticos, dos especialistas económico-financieros (usuarios finales), el especialista de la actividad de combustible (cliente y usuario), un auditor (usuario). Como cliente principal se integró al equipo el subgerente económico, con poder de decisión suficiente para asumir los retos del proyecto.

Tarea 3 Designación del Jefe del Proyecto

Como jefa del proyecto se designó a la analista que desarrolló la primera versión del sistema, cuyos conocimientos especializados, del dominio del problema y características personales para la comunicación, son suficientes para llevar adelante el proyecto y aplicar este procedimiento.

Tarea 4 Recogida inicial de información

Se realizó reunión del equipo y se escribió informe colectivo con las nuevas necesidades que habían sido identificadas (4) y las funcionalidades de la Versión 1 que debían mantenerse (3) y mejorarse (2). Se presentó el documento a los directivos principales y se ratificó.

Tarea 5 Estudio de Viabilidad del Proyecto

Se realizó el estudio y sus resultados arrojaron no sólo que el proyecto era viable económica y técnicamente, sino que el grado de necesidad era superior al valorado inicialmente. Se comprobó que al menos cuatro gerencias más del país tenían esas mismas necesidades y estaban interesadas en el producto. Los posibles *stakeholders* de esas provincias aportaron tres nuevas exigencias para la nueva versión.

Tabla 3.1 Resultados comparativos de la actividad de Condiciones de Inicio.

	Versión 1	Versión 2
Procedimiento	Ad Hoc	PROCIR
Miembros del equipo	1 informático	7 (equipo integrado)
Informe preliminar de necesidades	Elemental, informal e intuitivo	Sistematizada, formalizada en documento. Se usaron técnicas
Estudio de viabilidad	No	Sí
Tiempo consumido	2-3 días	Una semana

3.1.2 Actividad II. ELICITACIÓN DE LOS REQUISITOS

Las tareas de esta actividad se desarrollaron según establece el procedimiento, sólo se obviaron las relacionadas con la obtención de la información acerca del dominio del problema y del sistema actual, por el conocimiento práctico real que poseen del mismo los miembros del equipo.

En esta actividad se evidenció que la composición del equipo propició que los miembros se sintieran responsables por los resultados e hicieran suya la tarea con entusiasmo, las relaciones interpersonales fluyeron adecuadamente a partir del compromiso colectivo con la obtención de un producto de calidad que resolviera sus problemas y que pudiera ser generalizado con éxito.

La utilización de la tarjeta de requisitos, aun cuando se hizo en formato de papel impreso, permitió facilitar el acopio progresivo de todas las características de los requisitos, propició la detección rápida de lagunas informativas que fue posible superar con otras fuentes antes de pasar al análisis.

Tabla 3.2 Resultados comparativos de la actividad de Elicitación de Requisitos.

	Versión 1	Versión 2
Fuentes de información	Cajera y Especialista de combustible	Procesos y objetivos del negocio. *Stakeholders*
Usuarios identificados	4	11 (por gerencia)
Funcionalidades	5	9
Restricciones	1	3
Interfases	2	2
Total de requisitos elicitados	8	14
Normas de seguridad	0	2
Uso de técnicas de elicitación	Entrevistas (dos veces)	Entrevistas, cuestionarios, casos de uso
Medio de recogida de información	Agenda personal	Tarjeta de requisitos propuesta por PROCIR
Glosario de Términos	No	Sí
Otros *steakeholders* detectados	No	Sí, Oficina de recarga de tarjetas de CIMEX
Definidos objetivos y alcance del sistema	Sólo los objetivos	Visión, alcance y objetivos
Tiempo que duró la actividad	1 semana	11 días
Cantidad de requisitos que retornaron para ampliar información	0	2

Los miembros del equipo propusieron agregar un campo que la tarjeta no contemplaba inicialmente: el estado (o fase del proceso de software) en que se encuentra el requisito, que permite darle mejor seguimiento.

3.1.3 Actividad III. ANÁLISIS Y NEGOCIACIÓN DE REQUISITOS

Tarea 1 Detección de problemas potenciales.

Esta tarea resultó de gran utilidad para completar las características deseables de los requisitos. El uso de las tarjetas facilitó el trabajo de análisis. El equipo trabajó en la revisión de los requisitos por dúos y en pleno, aunque algunos hicieron lecturas individuales.

Tarea 2 Clasificación.

Esta tarea resultó un poco compleja, porque la mayor parte de los miembros del equipo no dominan las diferentes clasificaciones de los requisitos, por lo que fue necesario solicitar colaboración de otros informáticos para la revisión del trabajo realizado.

Tarea 3 Modelación.

Se realizó en formato plano, en papel, pero fue de utilidad para la mejor comprensión de los requisitos analizados y permitió resolver algunas ambigüedades detectadas, completar algunos requisitos y eliminar inconsistencias en un caso.

Se evidenció la falta de habilidades y práctica de los informáticos para la modelación de los requisitos, lo que constituye una a superar.

Tarea 4 Negociación.

La negociación fluyó muy positiva. Se llegó a acuerdos importantes relacionados con las prioridades para el desarrollo. Se incluyó un nuevo requerimiento que no había sido descubierto.

Tarea 5 Resolución de conflictos detectados.

Los conflictos detectados entre dos requisitos por problemas de disposiciones legales contradictorias entre las empresas fueron resueltos. Para ello fue necesario recurrir al MAC para esclarecer informaciones acerca de las exigencias de auditabilidad del sistema y al CIMEX para compatibilizar los informes que transitan entre esa entidad y ETECSA sobre la recarga de las tarjetas, el itinerario de cambio o devolución de tarjetas averiadas y bloqueadas, así como el trámite de tarjetas perdidas, todo esto a través del sistema caso de estudio.

Tarea 5 Priorización y determinación de los niveles de riesgos.

Se realizó la asignación preliminar de niveles de riesgos a los requisitos. Se observó que no hay experiencia en el tratamiento de los riesgos en los requisitos. No obstante todo el equipo colaboró para que la tarea se ejecutara.

Tabla 3.3 Resultados comparativos de la actividad de Análisis y Negociación.

	Versión 1	Versión 2
Participantes en el análisis	Desarrolladora y un usuario	Siete miembros del equipo y un invitado
Requisitos detectados con problemas en su redacción	2	4
Requisitos ambiguos	1	5
Requisitos funcionales	5	9
Requisitos no funcionales	1	5
Requisitos de interfaz	2	2
Requisitos con riesgos altos	1	1
Requisitos con riesgos medios	2	3
Requisitos negociados	1	3
Modelado	Sí, plano	Sí, plano
Requisitos dependientes	2	5
Tiempo que duró la fase	4 días	2 semanas

3.1.4 Actividad IV: ESPECIFICACIÓN DE REQUISITOS

En esta actividad fueron especificados 12 requisitos, de 14 que llegaron de la actividad anterior, dos fueron retornados a la elicitación por no poseer el consenso colectivo en cuanto a la claridad con que se encontraban expresados y se consideran difíciles de implementar y probar.

Sobre estos dos requisitos se volvió a trabajar, obteniéndose más información, lo que permitió determinar que uno era redundante, por lo que fue desechado y el otro se decidió posponer para versiones futuras, porque la complejidad de su implementación no tiene correspondencia con el escaso beneficio que podría reportar.

La especificación se realizó en forma textual, en un solo documento, la ERS, pues no se consideró necesario en este caso escribir un documento para los usuarios, DRU, por cuanto un número grande de ellos participó en las actividades desde la elicitación hasta la especificación y lograron un alto nivel de comprensión de los requisitos. Se realizó en un día.

En la Versión 1 no se realizó la especificación.

3.1.5 Actividad V: VALIDACIÓN DE LOS REQUISITOS

En la versión 1 del sistema de combustible la validación no fue validada tal y como se recomienda en la literatura, sino por criterios informales, sin aplicar ninguna técnica.

En la versión 2, la validación se desarrolló sin dificultades. Se revisaron por el equipo de trabajo y los interesados de otras provincias. Se aplicó una lista de chequeo, se efectuó una revisión y todos los requisitos presentados fueron validados. Se observó que se asumió esta actividad con responsabilidad por parte del equipo y demás participantes. Se validó en dos sesiones de trabajo de medio día cada una.

3.1.6 Actividad VI: GESTION DE RIESGOS

En la versión 1 no se realizó esta actividad.

En la versión 2 la Identificación y Evaluación de riesgos arrojó los siguientes riesgos asociados al proyecto:

Tabla 3.4 Identificación y evaluación de los riesgos asociados al proyecto.

Riesgo	**Probabilidad**	**Efectos**
Fallas en los procedimientos de autenticación	baja	serio
Personal clave enfermo o no disponible en momentos críticos (embarazo imprevisto de la Jefa del PY)	moderada	serio
Dificultad para recargar los datos desde el sistema anterior	baja	serio

Planificación de riesgos

Tabla 3.5 Estrategias para enfrentar los riesgos asociados al proyecto.

Riesgo	**Estrategia**
Fallas en los procedimientos de autenticación	Programar tarea que impida acceder a la operación del sistema si fallan los procedimientos de autenticación. Emisión de alarma en caso de falla repetida en la autenticación.
Personal clave enfermo o no disponible en momentos críticos (embarazo imprevisto de la Jefa del PY)	Preparar al segundo desarrollador del equipo de trabajo para que pueda sustituir a la compañera en caso de certificado médico o licencia. Asignar otro informático al equipo para que se familiarice con el proyecto y esté en condiciones de asumir tareas de desarrollo en él si fuera necesario.
Dificultad para recargar los datos desde el sistema anterior	Preparar salvas y una tarea automática para la recarga inicial. Preparar condiciones para cargar los datos manualmente.

El equipo de trabajo, después de analizar, decidió que ninguno de los riesgos identificados pone en riesgo la continuidad del proyecto y que su envergadura no justificaba un plan de supervisión de riesgos.

En el desarrollo de esta actividad se evidenció la inexperiencia y falta de preparación del equipo de trabajo para identificar, evaluar, planificar los riesgos en la fase de ingeniería de requisitos.

3.1.7 Actividad VII: GESTIÓN DE REQUISITOS

En la entidad donde se aplicó PROCIR se confirmó, como fue analizado en el Capítulo 1, que no se recoge información que permita evaluar el desempeño del proceso y definir métricas que puedan ser utilizadas en la toma de decisiones en la fase de ingeniería de requisitos de los proyectos.

Se estableció la línea base para registrar la información de los proyectos. La información a recoger se detalla en la tabla 3.6.

Tabla 3.6 Información para la línea base establecida.

ID y nombre Proyecto	Número consecutivo y único que identifica unívocamente al proyecto al que se refiere
ID Versión PY	Número consecutivo y único que identifica unívocamente a la versión a la que se refiere
Total de REQ	Total de requisitos aprobados
REQ Func	Total de requisitos funcionales
REQ no Func	Total de requisitos no funcionales
C SOL	Total de cambios solicitados
C APROB	Total de cambios aprobados
C EJEC	Total de cambios ejecutados
F sol C	Fecha de solicitud del cambio
F apr C	Fecha de aprobación del cambio
F ejec C	Fecha de ejecución del cambio
Tiempo total C	Tiempo total que se consumió entre la solicitud y la ejecución del cambio
REQ no ambig	Total de requisitos no ambiguos en la especificación
REQ valid	Total de requisitos que fueron validados
REQ probados	Total de requisitos que fueron probados
Total ET	Total de miembros del equipo de trabajo
Total CLIE	Total de clientes en el equipo de trabajo
Total USR	Total de usuarios finales en el equipo de

	trabajo
Riesg Ident	Total de riesgos identificados
Riesg tratad	Total de riesgos tratados
Riesg desech	Total de riesgos desechados
C I	Actividad de Condiciones Iniciales
Elic	Actividad de Elicitación de requisitos
A y N	Actividad de análisis y negociación
Espec	Actividad de especificación de requisitos
Valid	Actividad de validación de requisitos

Para el caso de estudio los datos recogidos aparecen en la tabla 3.7.

Tabla 3.7 Línea base de requisitos.

ID y nombre Proyecto	ID Versión PY	Total de REQ	REQ Func	REQ no Func
1: Sistema Combustible	2	12	9	5

C SOL	C APROB	C EJEC	F sol C	F apr C	F ejec C	Tiempo total C
1	1		12/9/06	15/9/06		

REQ no ambig	REQ valid	REQ probados	Total ET	Total CLIE	Total USR
12	12		7	2 (y users)	3 (+ 2 CL)

T Riesg Ident	Riesg tratad	Riesg desech	Tiempo (días)				
			C I	Elic	A y N	Espec	Valid
3	3	0	7	11	14	1	1

Para seguir la tazabilidad de los requisitos se desarrollaron dos matrices que el equipo consideró les serían de utilidad.

Matriz de Estado: identifica fase en que se encuentra cada requisito.

	Ing. Req	Diseño	Codificación	Pruebas	Liberación	Espera
R1				x		
R2				x		
R3			x			
R4			x			
R5			x			
R6			x			
R7		x				
R8				x		
R9				x		
R10				x		
R11				x		
R12				x		
R13						x

Matriz de seguimiento de dependencias: indica cómo se relacionan los requisitos entre sí.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13
R1		x											
R2													
R3					x								
R4													
R5													
R6													
R7													
R8	x												
R9										x			
R10													
R11													
R12											x		
R13													

El equipo de trabajo aprobó el subgrupo de control de cambios, integrado por tres compañeros, para evaluar las solicitudes y proponer al equipo su aprobación o desestimación.

Se ha gestionado una sola solicitud de cambio, referida al requisito 7, en cuanto al diseño gráfico de una interfaz de usuario, la que fue aprobada por el equipo a propuesta del subgrupo de control, reenrutándose ese requisito a la fase de diseño.

Hasta la fecha el proyecto para la implementación de la versión 2 del sistema de combustible de la Gerencia ETECSA Cienfuegos se desarrolla de acuerdo a las estimaciones de costo y tiempo,

debiendo liberarse el producto el próximo mes, al concluirse las pruebas que se realizan ya en otras provincias.

Durante el proceso de ingeniería de requisitos desarrollado en el caso de estudio práctico, los implicados en él no presentaron quejas por pérdidas de tiempo en ninguna de las actividades, ni por falta de comprensión de procedimiento.

La valoración, tanto del equipo de trabajo como del Grupo de Desarrollo de la Filial de TISW de Cienfuegos, es que la aplicación del procedimiento permitió el desarrollo organizado del proyecto, el control de las actividades a desarrollar, al control de la calidad del proceso. Por lo anterior decidieron asumir PROCIR para su utilización en todos los proyectos que asuman a partir de ahora.

3.2 Aplicabilidad de las "herramientas" propuestas en PROCIR

La aplicabilidad de las "herramientas" propuestas en PROCIR se validará a partir de que esta posea las características siguientes:

3.2.1 Suficiencia informativa

La suficiencia informativa, referida a la disponibilidad de toda la información (y su tratamiento) que se requiere para su aplicación en las empresas cubanas del software.

En la aplicación del procedimiento para la versión 2 del sistema de combustible de la Gerencia ETECSA Cienfuegos quedó demostrada esta característica. La información que se requiere como entrada para obtener las salidas del proceso de desarrollo de requisitos es la solicitud del desarrollo de un proyecto dado.

Para realizar el tratamiento de esta, se dispone del conjunto de procedimientos para las siete actividades definidas en este, salidas, controles y elementos de evaluación.

3.2.2 Consistencia lógica

Consistencia lógica, en función de la ejecución de sus pasos en la secuencia planteada en la correspondencia con la lógica de la ejecución de este tipo de estudio.

La secuencia lógica de pasos para el desarrollo del proceso de ingeniería de requisitos definida en PROCIR, es mantenida independientemente de la empresa donde se aplique. Esto queda demostrado en la aplicación de éste para el desarrollo del sistema de combustible de la Gerencia de ETECSA en su versión 2, en una empresa que no es propiamente de desarrollo de software. No obstante, toda empresa para ejecutar adecuadamente este procedimiento requiere, entre otras cosas, realizar las actividades de condiciones de inicio del proyecto, elicitación de requisitos, análisis y negociación,

especificación, validación, gestión de riesgos y gestión de requisitos y las etapas del ciclo de vida del proyecto.

Actividades lógicamente planteadas como parte del proceso de ingeniería de requisitos al nivel del proyecto.

Esta característica también se puede demostrar en el marco teórico de todo estudio referente a garantizar la calidad del software, porque se incluye desde las etapas más tempranas una actividad protectora, como es la gestión del riesgo.

3.2.3 Flexibilidad

La flexibilidad está dada por la posibilidad de aplicarse en cualquier entidad que desarrolle software y porque es independiente de cualquier metodología que desarrolle software.

Una entidad puede ser incluso un grupo que desarrolle software, caso más particular del contexto en que se puede aplicar el procedimiento y precisamente en este, quedó demostrada esta característica en el ejemplo de aplicación planteado.

3.2.4 Calidad de los resultados desde el punto de vista del producto final

PROCIR presenta resultados similares para el desarrollo del proceso de ingeniería de requisitos a nivel del proyecto, a los alcanzados en otros modelos reconocidos en la literatura, incorporando la manera de realizarlo. PROCIR se basa en los modelos de Pohl, Espiral, Iteración de Actividades y SWEBOK, y en las mejores prácticas de la ingeniería de software. De los principios y valores de la Programación Extrema se define el equipo de trabajo, para superar las barreras de comunicación que afectan las relaciones humanas.

3.2.5 Parsimonia

La parsimonia es referida a su cualidad de ser "simple" dentro de la complejidad inherente que presentan estos estudios.

En PROCIR el proceso de ingeniería de requisitos es transparente. Esto está dado por la sencillez de la aplicación del conjunto de procedimientos definidos, lo cual quedó demostrado específicamente en la evaluación realizada al proyecto de la versión 2 del Sistema de Combustible de la Gerencia ETECSA Cienfuegos.

3.2.6 Racionalidad

La racionalidad es evaluada de acuerdo con la relación gasto - beneficio que se requiere para su aplicación.

Los gastos en la aplicación de PROCIR están dados por el gasto de entrenamiento para su aplicación, ya que no se requiere realizar inversiones en otros recursos.

Existen otros beneficios intangibles de la aplicación del procedimiento que son:

- Contribuye en la definición del proceso de un área clave en una organización desarrolladora de software.
- Permite al equipo de proyectos realizar la autogestión de los riesgos y los cambios desde las etapas más tempranas del proceso de ingeniería de software.
- Contribuye a determinar las necesidades de formación de los recursos humanos.

3.2.7 Pertinencia

La pertinencia está relacionada con la propuesta de un procedimiento para la etapa de ingeniería de requisitos de proyectos informáticos que se adecuan a las condiciones existentes y a las necesidades de las empresas cubanas del software.

Las necesidades del cliente tienen mayor oportunidad de ser satisfechas en proyectos donde se aplique PROCIR por el tratamiento temprano a los riesgos, la gestión de los cambios y las evaluaciones a los atributos de los requisitos de los usuarios desde las primeras etapas del proceso de ingeniería de software.

Debido a las condiciones existentes en la empresa cubana del software, caracterizada por un proceso inmaduro, se necesita comenzar a desarrollar el proceso de ingeniería de requisitos a través de PROCIR, al menos, en los proyectos que se ejecuten. PROCIR contribuye en:

El suministro a la dirección de la organización de información necesaria acerca del estado real de cualquier proyecto de desarrollo a partir del estado de sus requisitos especificados.

Contribuye al mejoramiento del desempeño del proceso de ingeniería de requisitos, al garantizar un eslabón del proceso de la ingeniería de software.

3.2.8 Perspectiva o generalidad

Perspectiva o generalidad, dada por la posibilidad de su extensión como instrumento metodológico para ejecutar estos estudios en otros procesos dentro de la industria del software e incluso para otros sectores.

El soporte técnico de PROCIR, el conjunto de procedimientos, de técnicas de gestión y las interrelaciones entre los elementos, como el cliente, el trabajo en equipo y la mejora continua, ofrecen una plataforma metodológica que permite abordar otros procesos de software.

3.3 Conclusiones parciales

Como conclusiones del capítulo se plantea:

- La aplicación del procedimiento al objeto de estudio práctico seleccionado, demuestra que se puede extender su empleo a otras entidades pequeñas y medianas de la industria del software.
- La aplicación de las herramientas metodológicas propuestas en el caso del proyecto de desarrollo de la versión 2 del sistema de combustible de la Gerencia ETECSA Cienfuegos, permitió obtener una especificación de requisitos de software completa, formal y acordada entre todos los participantes, que cumple con las características de calidad definidas.
- También se demostró que la integración de las mejores prácticas, los métodos y herramientas facilitan una adecuada realización de las tareas de gestión de requisitos en función de los objetivos, metas y estrategias de la organización, y en este caso concreto se logró una mejora en cuanto a los niveles actuales de desempeño de la gestión de los requisitos del software y la satisfacción del cliente en el objeto de estudio práctico.
- Los resultados expuestos en el epígrafe 3.1 demuestran la aplicabilidad de PROCIR en una entidad. La introducción paulatina del procedimiento en una organización a partir de su aplicación en los proyectos debe conducir a la mejora del proceso y calidad del producto final.

Conclusiones y recomendaciones

1. La función de la ingeniería de requisitos está adquiriendo una progresiva connotación en los esfuerzos por alcanzar niveles de competitividad en la industria del software, por su decisiva implicación en el logro de altos niveles de satisfacción del cliente y el cumplimiento del equilibrio de plazo y costo del proyecto. El análisis de esta función, desde su perspectiva más contemporánea, debe centrarse en dos ideas fundamentales: el análisis de las características de requisitos que son el fundamento del desarrollo del proyecto en función de su capacidad para sustentar satisfacer las necesidades del cliente y el análisis con un enfoque proactivo de las políticas y "buenas prácticas" que potencie de manera efectiva el desempeño del proceso.

2. Existe una creciente base teórica conceptual sobre la ingeniería de requisitos, que se ha ido enriqueciendo paulatinamente a partir de contribuciones teóricas, pero sobre todo de estudios e investigaciones empíricas. Sin embargo, en contraste con el desarrollo empírico y teórico-conceptual, en la componente metodológica de la forma de entender, orientar y ejecutar sus actividades, se han apreciado escasos procedimientos que permitan dotar a las empresas de una guía para la definición y gestión de los requisitos y sus riesgos, desde un enfoque proactivo y estratégico para la pequeña y mediana empresa de software cubana. Por lo cual el problema científico formulado para la presente investigación se considera de gran actualidad y pertinencia, tanto en el plano conceptual – metodológico como práctico.

3. El procedimiento general y los procedimientos asociados desarrollados, para la definición y gestión de los requisitos del software en las pequeñas y medianas empresas, conforman un cuerpo de elementos coherentes desde la perspectiva teórico-metodológica desarrollado por la autora, para dar solución al problema científico planteado, constituye también un instrumento estratégico con enfoque proactivo, que permite al gestor adoptar, desarrollar e implementar adecuadamente las actividades de gestión de los requisitos y sus riesgos, de una forma disciplinada, coherente y repetitiva, en función de obtener productos de calidad que satisfagan las necesidades del cliente, manteniendo el equilibrio de plazo y costo del proyecto en virtud de lograr un mejor desempeño del proceso de IR en la pequeña y mediana empresa de software cubana.

4. El procedimiento metodológico general y el conjunto de procedimientos específicos asociados, coherentes con los objetivos del negocio, permite en la PyME: obtener y gestionar especificaciones de requisitos de software completas, formales y acordadas entre todos los participantes, que cumplan con las características calidad definidas; el mejoramiento gradual

del proceso de desarrollo y la gestión de un proyecto de software que logre la satisfacción del cliente en estas organizaciones.

5. La aplicación empírica del procedimiento desarrollado al caso de estudio seleccionado, puso en evidencia las deficiencias e insuficiencias que se generaron durante la especificación y gestión de los requisitos sin seguir un proceso definido y bien estructurado. Además, permitió obtener una especificación de requisitos de software completa, formal y acordada entre todos los participantes, con las características de calidad definidas. Se demostró que la integración de las mejores prácticas, los métodos y herramientas facilitaron la realización de las tareas de gestión de los requisitos y un mejor desempeño y satisfacción del cliente en el objeto de estudio práctico.

Derivadas del estudio realizado, así como de las conclusiones generales emanadas del mismo, se recomienda:

1. Continuar la divulgación de las experiencias y resultado obtenido en este trabajo de investigación, a través de publicaciones científicas en revistas y eventos científicos nacionales e internacionales, todo lo cual contribuirá a la generalización de dichos resultado.
2. Proponer a la Dirección de la Unidad de Negocios de Tecnología y Software de ETECSA la extensión de la experiencia de aplicación del procedimiento general para el desarrollo del proceso de ingeniería de requisitos al desarrollo de nuevos proyectos de software a desarrollar en la empresa, con vistas a su posterior y posible generalización a escala nacional.
3. Diseñar un Programa de Capacitación, dirigido a preparar a los especialistas vinculados a los procesos de ingeniería de requisitos de proyectos de software en los aspectos metodológicos necesarios para aplicar exitosamente PROCIR, potenciando los aspectos relacionados con la modelación de procesos de negocios y de requisitos, casos de uso, identificación y evaluación de riesgos.
4. Desarrollar la herramienta CASE para el procedimiento de gestión de los requisitos de proyectos de software.

Referencias bibliográficas

[Álvarez, 2003] Álvarez, S., (2003): "Un modelo de calidad para una empresa de software", Evento Calidad 2003 Convención Informática 2003, C. Habana (CU).

[Báez y Barba, 2001] Báez, M., Barba, S.: "Metodología DoRCU para la Ingeniería de Requisitos" WER´2001. Buenos Aires, Argentina. 2001.

[Bernardez, et al., 2004] Bernárdez, B., Durán, A. Toro, M.: Una propuesta para la verificación de requisitos basada en métricas. Revista de Procesos y Métricas de las Tecnologías de la Información (RPM) ISSN: 1698-2029 VOL. 1, Nº 2, Agosto 2004, 13-24. Asociación Española de Sistemas de Informáticos (AEMES)

[Boehm, 1988] Boehm, B.: "A Spiral Model of Software Development and Enhancement". IEEE Computer. Vol. 21, # 5. 1988.

[Boehm, et al., 1994] Boehm, B. W., Bose, P., Horowitz, E. y Lee, M.-J.: Software Requirements as Negotiated Win Conditions. En Proceedings of the First International Conference on Requirements Engineering, 1994.

[Boehm, 1998] Boehm, B. W.: Using the WINWIN Spiral Model: A Case Study. Computer, v. 31, No. 7, Julio 1998.

[Boehm, 2001] Bohem, B.; Grünbacher P.; Briggs, R.O. (2001). EasyWinWin: A Groupware-Supported Methodology For Requirements Negotiation. IEEE 2001.

[Bourque et al. 1999] Bourque, P., Dupuis, R. y Abran, A.: "The Guide to the Software Engineering Body of Knowledge". IEEE Software, 16(6), Septiembre/ Diciembre 1999.

[Carrizo y Juristo, 2002] Carrizo, D. y Juristo, N.: Caracterización de las Técnicas de Adquisición de Requisitos. Disponible en:

is.ls.fi.upm.es/doctorado/Trabajos20012002/DCarrizo.doc

[Charette, 1989] Charette, R. N.: Software Engineering Risk Analysis and Management, McGraw–Hill/Intertext, 1989.

[Christel y Kang 1992] Christel, M. G., Kang, K. C.: "Issues in Requirements Elicitation". Technical Report CMU/SEI–92–TR–12, Software Engineering Institute, Carnegie Mellon University, 1992.

[CMMI-SW, 2002] CMMI-SW, 2002. Software Engineering Capability Maturity Model Integration. Disponible en: http://www.sei.cmu.edu/cmmi/models/models.html,

[Chrissis, 2003] Chrissis & Konrad & Shrum, 2003. CMMI. Guidelines for Process Integration and Product Improvement. Editorial Addison-Wesley. ISBN 0321154967.

[Davis, 1993] Davis, A. M.: "Software Requirements: Objects, Functions and States" Development. Prentice Hall. 1993

[Davis, 1995] Davis, A. M.: 201 Principles of Software Development. McGraw–Hill. 1995.

[Díez, 2001] Díez, A.: "IRQA y el Desarrollo de Proyectos: Experiencias Prácticas". Jornadas de Ingeniería de Requisitos Aplicada. Sevilla, España. 2001.

[DoD, 1994] DoD. Military Standard 498: Software Development and Documentation. Departament of Defense of the United States of America, 1994.

[Dumke, 2005] Dumke, R., Schmietendorf, A., Zuse, H.: Formal Descriptions of Software Measurement and Evaluation. *A Short Overview and Evaluation.* Otto-von-Guericke-Universität Magdeburg, Postfach 4120, 39016 Magdeburg, 2005.

[Durán et al. 1999] Durán, A., Bernárdez, B., Ruiz, A. y Toro, M.: A Requirements Elicitation Approach Based in Templates and Patterns. En WER'99 Proceedings, Buenos Aires, 1999.

[Durán, 2000] Durán Toro, A.: Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información. Tesis doctoral. Universidad de Sevilla, 2000.

[Durán, 2002] Durán, A., Bernández, B.: "Metodología para la Elicitación de Requisitos de Software", Universidad de Sevilla. 2002.

[Durán, 2004] Durán, A.: REM. Página web de la herramienta REM. http://rem.lsi.us.es, 2004.

[Edwards, 1991] Edwards M., Howell S.: A methodology for system requirements specification and traceability for large real- time complex systems. Technical Report, Naval Surface Warfare Center, 1991.

[Escalona, et al., 2002] Escalona, M.J., Torres, J., Mejías, M. (2002). Requirements capture workflow in Global Information Systems. Proceedings of OOIS. Springer- Verlag. Montpellier, France.

[Escalona y Koch, 2003] Escalona, M, J, y Koch, N.: Ingeniería de Requisitos en aplicaciones para la web: Un estudio comparativo. VI Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software (IDEAS 2003).

[Febles, 2000] Febles, Ailyn. (2000): "Case Corporativo para el proceso de control de cambios". [Tesis de Maestría]. Ciudad de la Habana (CU), ISPJAE.

[Febles, 2002] Febles, Ailyn (2003): "Guía práctica del modelo de referencia MConfig.pm", Ingeniería Industrial, ISPJAE.

[Ferreira, 2001] Ferreira, M.J., Loucopoulos, P. (2001). Organisation of analysis patterns for effective re-use. Proceedings of the International Conference on Enterprise Information Systems. ICEIS 2001. Setubal, Portugal.

[Gallagher, 1999] Gallagher, Brian P.: Software Acquisition Risk Management Key Process Area (KPA) — A Guidebook, Version 1.02. Carnegie Mellon University. HANDBOOK CMU/SEI-99-HB-001. 1999. Disponible en:

http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99hb001.pdf

[García, 2000] García Ávila, Lourdes. Modelo para la evaluación de la calidad del análisis y diseño orientados a objetos de sistemas informáticos (CADOOSI). Tesis de doctorado. Universidad Central de Las Villas, 2000.

[García y Álvarez, 1999] García & Álvarez, (1999):"Una visión general sobre la certificación de calidad y la evaluación del proceso de desarrollo de software". Revista de Ingeniería Industrial No.4/1.

[Goguen y Linde, 1993] Goguen, J. A. y Linde, C.: Techniques for Requirements Elicitation. En Proceedings of the First International Symposium on Requirements Engineering, 1993. También aparece en [Thayer y Dorfman, 1997].

[Goguen, 1994] Goguen, J. A.: "Requirements Engineering as the Reconciliation of Social and Technical Issues". En Requirements Engineering: Social and Technical Issues, pág 165–199. Academic Press, 1994.

[González, 2003] González P. Ignacio, (2003): Discurso pronunciado en la inauguración de la Convención Informática 2003, Cuba.

[GTI, 2002] GTI, (2002), Reporte de Trabajo No. 2. Comisión de Productos y Servicios-Fuerza de Tareas de la Industria de Software, GTI, MIC, Rev00, septiembre.

[Hurtado, 2005] Hurtado, Julio Ariel: "PROYECTO SIMEP-SW. Investigación: Hacia una Línea de Procesos Ágiles Agile SPsL". Universidad del Cauca, Colombia. 2005.

[IBM, 1997] IBM OOTC (1997). Developing Object Oriented Software. IBM Object Oriented Technology Center. Prentice-Hall.

[IEEE 1990] IEEE. IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard 610.12–1990, Institute of Electrical and Electronics Engineers, 1990.

[IEEE 1993] IEEE. IEEE Standard Collection: Software Engineering. IEEE Standard 610.12–1990, Institute of Electrical and Electronics Engineers, 1993.

[IEEE 1996] IEEE. IEEE Guide for Developing System Requirements Specifications. IEEE/ANSI Standard 1233–1996, Institute of Electrical and Electronics Engineers, 1996.

[INEGI, 2003] INEGI, Durán Rubio, Sergio, Boletín de Política Informática, Num. 6, 2003.

[Insfrán, et al., 2002] Insfrán, E., Pastor, O., Wieringa, R. (2002). Requirements Engineering-Based Conceptual Modeling. Requirements Engineering Journal, Vol 7 (1).

[Jacobson, 1995] Jacobson, I. (1995). Modeling with use cases-Formalizing use-case modelling. Journal of Object -Oriented Programming,

[Jacobson, et al., 1998] Jacobson, I., Booch, G., Rumbaugh, J.: "The Unified Software Process". Adison-Wesley. 1998.

[KLCI, 2001] KLCI (2001). Software Risk management Practices – 2001, KLCI research group report. Disponible en http://www.klci.com (2001).

[Koch, 2001] Koch, N. (2001). Software Engineering for Adaptative Hypermedia Applications. Ph. Thesis, FAST Reihe Softwaretechnik Vol(12), Uni-Druck Publishing Company, Munich. Germany.

[Kolde, 2004] Kolde, Chris: Basic metrics for requirements management. A Borlad White Paper, 2004. Disponible en: http://www.borlan.com

[Kontoya, 2000] Kontoya, G. y Sommerville, I.: "Requirements Engineering: Processes and Techniques". John Wiley & Sons, 2000.

[Koubarakis, 2000] Koubarakis M. and Plexousakis D., A Formal Model for Business Process Modeling and Design, Proceedings of CAiSE*00, Stockholm, Sweden, June 5-9, 2000.

[Kovitz 1998] Kovitz, B. L.: Practical Software Requirements: A Manual of Content & Style. Manning, 1998.

[Kruchten, 1998] Kruchten, P. (1998). The Rational Unified Process. Addison Wesley.

[Lage, 2000] Lage, Carlos (2000): Discurso pronunciado en la Inauguración de la Convención Informática 2000, Cuba.

[Leffingwell y Widrig, 2003] Leffingwell, Dean, Widrig, Don. "Managing Software Requirements". Second Edition. Addison-Wesley 2003.

[Liu y Yu, 2001] Liu, L., Yu, E. (2001). From Requirements to Architectural Design using Goals and Scenarios Proceedings of the 6th Micon Workshop. Canadá.

[Lowe, 1999] Lowe, D., Hall, W. (1999). Hypermedia and the Web. An Engineering approach. John Wiley & Son.

[Martínez, et al., 2002] Martínez, A., Estrada H., Pastor, O. El Modelo de Negocios como origen de especificaciones de requisitos de software: una aproximación metodológica. Disponible en: www.dsic.upv.es/~alimartin/Publications_archivos/Ciicc2002.pdf

[Minasi, 2000] Minasi, Mark: The Software Conspiracy: Why Sftware Companies Put Out Faulty Products, How They Can Hurt You, and What You Can Do. McGraw-Hill, 2000

[Monzón, 2001] Monzón, A.: Calidad en la especificación: ¿Se pueden medir los requisitos? Disponible en:

http://download.irqaonline.com/docs/papers/Calidad_de_la_Especificacion.pdf

[Moreno, 2003] Moreno, Boris (2003): Conferencia magistral, Evento de Calidad, Convención Informática 2003, Cuba.

[Olsina, 1999] Olsina, L. (1999). Metodología cualitativa para la evaluación y comparación de la calidad de sitios web. Ph. Tesis. Facultad de Ciencias Exactas. Universidad de la Pampa. Argentina.

[Pacheco, 2004] Pacheco Agüero, Carla L.: La Identificación de Stakeholders en la Ingeniería de Requisitos. Trabajo de investigación tutelado. Disponible en:

http://www.is.ls.fi.upm.es/doctorado/Trabajos20042005/Pacheco_Aguero.pdf

[Palmer, 1997] Palmer J. D.: Traceability in Software Requirements Engineering, R.H. Thayer and M. Dorfman (Eds), IEEE Computer Society Press, 1997. (364:374).

[Pan, et al., 2001] Pan, D., Zhu, D., Johnson, K. (2001). Requirements Engineering Techniques. Internal Report. Department of Computer Science. University of Calgary. Canadá.

[Paulk et al., 1993] Paulk, M. C., Curtis, B., Chrissis, M. B. y Weber, C. V.: Capability Maturity Model for Software, Version 1.1. Technical Report CMU/SEI–93–TR–024, Software Engineering Institute, Carnegie Mellon University, 1993. Disponible en http://www.sei.cmu.edu.

[Peña, 1998] Peña, R. (1998). Diseño de programas. Formalismo y abstracción. Prentice Hall.

[Pohl, 1994] Pohl, K.: The Three Dimensions of Requirements Engineering: A Framework and its Application. Information Systems, 3(19), 1994.

[Pohl, 1997] Pohl, K.: "Requeriments Engineering: An Overview". Encyclopedia of Computer Science and Technology, 36. 1997. Disponible en http://sunsite.informatik.rwth-aachen.de/CREWS/reports96.htm.

[Potts, et al., 1994a] Potts, C., Takahashi, K. y Anton, A.: Inquiry–Based Requirements Analysis. IEEE Software, 11(2), 1994.

[Potts, et al., 1994b] Potts, C., Takahashi, K. y Anton, A.: Inquiry–Based Scenario Analysis of System Requirements. Informe Técnico GIT–CC–94/14, Georgia Institute of Technology, 1994. Disponible en http://www.cc.gatech.edu/computing/SW Eng/reqts.html.

[Pressman, 1997] Pressman, Roger S.: "Ingeniería de Software. Un enfoque práctico." Cuarta edición. McGraw-Hill. Madrid. 1997.

[Pressman, 2002] Pressman, Roger S.: "Ingeniería de Software. Un enfoque práctico." Quinta edición. McGraw-Hill. Madrid. 2002.

[Raghavan, et al., 1994] Raghavan, S., Zelesnik, G. y Ford, G.: Lectures Notes on Requirements Elicitation. Educational Materials CMU/SEI–94–EM–10, Software Engineering Institute, Carnegie Mellon University, 1994. Disponible en http://www.sei.cmu.edu

[Rational, 2001] Rational Unified Process. Rational Software Corporation. "Rational Unified Process". Version 2001A.04.00, Copyright 1987-2001.

[Ridao, 2002] Ridao, Marcela: Uso de patrones en el proceso de construcción de escenarios. Tesis de maestría. Disponible en www-di.inf.puc-rio.br/~julio/Tesis-marcela.pdf

[Roppponen, 2000] Roppponen, J., Lyytinen, K. (2000). Components of Software

Development Risk: Hot to address Them? IEEE transactions on software engineering, 26(2), pp. 98-111.

[Sawyer, et al., 1997] Sawyer, P., Sommerville, I. y Viller, S.: Requirements Process Improvement through The Phased Introduction of Good Practice. Software Process – Improvement and Practice, 3(1), 1997.

[Sawyer y Kontoya, 1999] Sawyer, P. y Kontoya, G.: SWEBOK: "Software Requirements Engineering Knowledge Area Description". Informe Técnico Versión 0.5, SWEBOK Project, 1999. Disponible en http://www.swebok.org.

[SIME, 1997] SIME, (1997). SIME: Lineamientos estratégicos para la informatización de la sociedad cubana, La Habana (CU).

[Sommerville y Sawyer, 1997] Sommerville I. y Sawyer, P.: Requirements Engineering: A Good Practice Guide. Wiley, 1997.

[Sommerville, 2000] Sommerville, Ian: "Software Engineering". 6th. Edition. Addison-Wesley 2000 (chapter 22).

[Sommerville, 2005] Sommerville, Ian: "Software Engineering". 7th. Edition. Addison-Wesley 2005.

[SPICE, 1998] ISO/IEC TR 15504-2:1998 http://www-sqi.cit.gu.edu.au/spice/

[SWEBOK, 2004] Guide to the Software Engineering Body of Knowledge. IEEE Computer Society Order Number C2330, ISBN 0-7695-2330-7, Library of Congress Number 2005921729. 2004.

[Thayer y Dorfman, 1997] Thayer, R. H. y Dorfman, M., editores. Software Requirements Engineering. IEEE Computer Society Press, 2a edición, 1997. Es la 2a edición de [Thayer y Dorfman 1990].

[TSG, 1995] TSG. "The CHAOS Report." The Standish Group, 1995. Disponible en http://www.standishgroup.com/chaos.html.

[UML, 2001] Unified Modeling Language. Version 1.4. www.omg.org

[Vilain, et al., 2000] Vilain, P., Schwabe, D., Sieckenius, C. (2000). A diagrammatic Tool for Representing User Interaction in UML. Lecture Notes in Computer Science. Proc. UML'2000. York, England.

[Weidenhaupt, et al., 1999] Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P. (1999). Scenarios in Systems Development: Current Practice. IEEE Software. 2, 34-45.

[Wieringa, 1996] Wieringa, R. J.: Requirements Engineering: Frameworks for Understanding. JohnWiley & Sons, 1996.

Anexo 1. Técnicas de especificación de requisitos

- *Lenguaje natural*: Resulta una técnica muy ambigua para la definición de los requisitos. Consiste en definir los requisitos en lenguaje natural sin usar reglas para ello. Pero, a pesar de que son muchos los trabajos que critican su uso, es cierto que a nivel práctico se sigue utilizando.
- *Glosario y ontologías*: La diversidad de personas que forman parte de un proyecto software hace que sea necesario establecer un marco de terminología común. Esta necesidad se vuelve más patente en los sistemas de información web puesto que el equipo de desarrollo en ellas suele ser más interdisciplinario [Koch, 2001]. Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recogen y definen los conceptos más relevantes y críticos para el sistema. En esta línea se encuentra también el uso de ontologías, en las que no sólo aparecen los términos, sino también las relaciones entre ellos.
- *Plantillas o patrones*: Esta técnica, recomendada por varios autores [Durán, et al., 1999], [Escalona, et al., 2002], tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado detallado, el trabajo de rellenar las plantillas y mantenerlas puede ser demasiado tedioso.
- *Escenarios*: La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos [Liu, 2001]. La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia representaciones gráficas en forma de diagramas de flujo [Weidenhaupt, et al., 1999]]. El análisis de los escenarios, hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales de sistema [Lowe, 1999].
- *Casos de uso*: Es como técnica de definición de requisitos como más ampliamente han sido aceptados los casos de uso. Actualmente se ha propuesto como técnica básica del proceso RUP [Kruchten, 1998]. Sin embargo, son varios los autores que defienden que pueden resultar ambiguos a la hora de definir los requisitos [Díez, 2001], [Vilain, et al., 2000], [Insfrán, et al., 2002] por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.
- *Lenguajes Formales*: Otro grupo de técnicas que merece la pena resaltar como extremo opuesto al lenguaje natural, es la utilización de lenguajes formales para describir los requisitos de un

sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción formal, han sido aplicadas en el mundo de la ingeniería de requisitos desde hace años [Peña, 1998]. Sin embargo, resultan muy complejas en su utilización y para ser entendidas por el cliente. El mayor inconveniente es que no favorecen la comunicación entre cliente y analista. Por el contrario, es la representación menos ambigua de los requisitos y la que más se presta a técnicas de verificación automatizadas. (JRDL – CRD, RML, CML – TELOS)