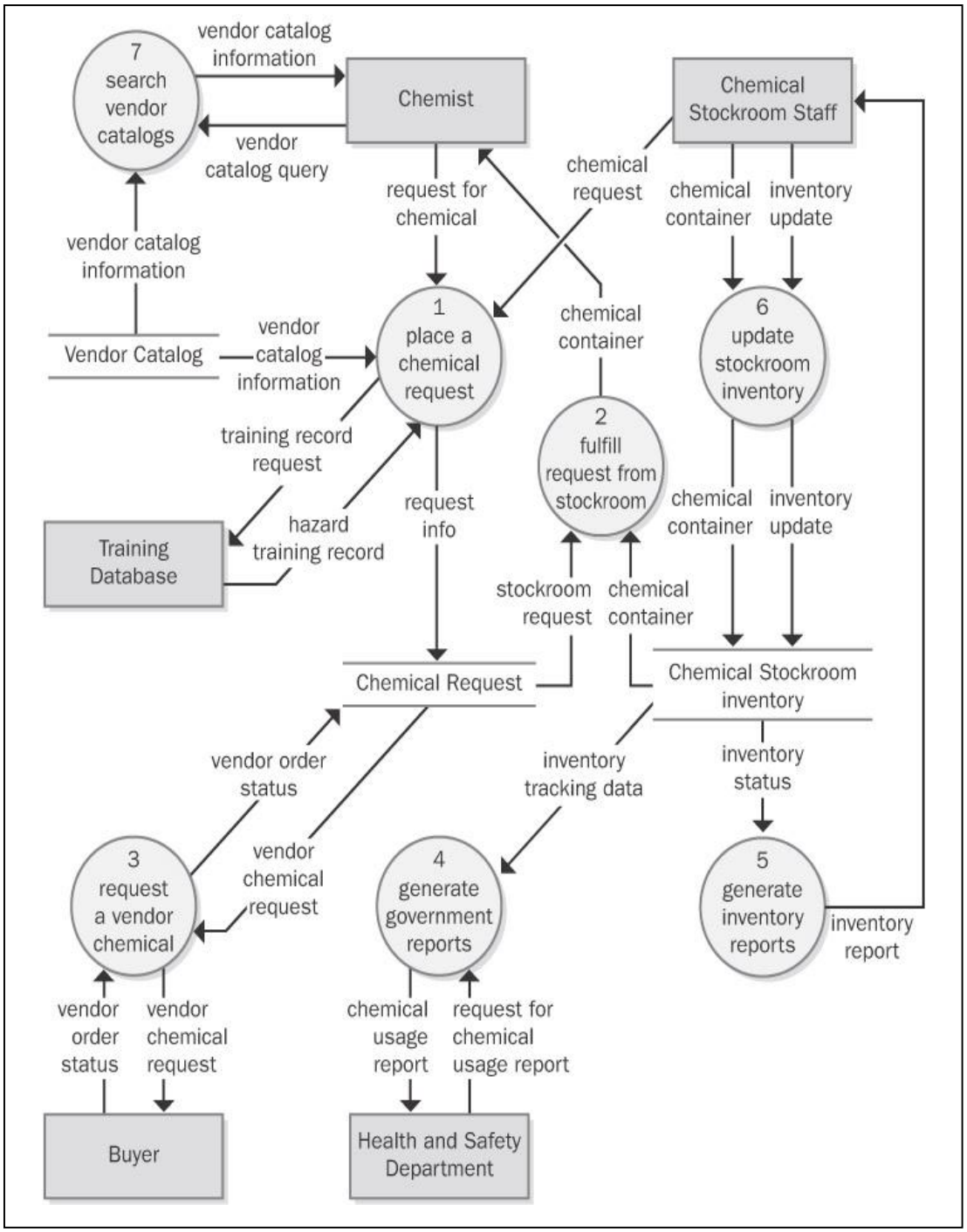
		Entidades o sus Atributos (DER) Clases o sus atributos (Diagramas de clase)
Verbo	Acciones, cosas que usuarios puedan hacer, eventos que puedan ocurrir	Procesos (DFD) Caos de Uso (Diagramas de casos de uso) Relaciones (DER) Transiciones (DTE) Actividades (Diagrama de Actividades)

2.2.7.3 Diagrama de Flujo de Datos

Los diagramas de flujo de datos es una herramienta básica para el análisis estructurado (DeMarco, 1979; Robertson and Robertson, 1994). Un DFD identifica los procesos transformacionales de un sistema, la colección (almacén) de datos o material que el sistema manipula y los flujos de datos o material entre procesos, almacenes y el mundo exterior. El modelado de flujo de datos toma un acercamiento de descomposición funcional al análisis del sistema, rompiendo problemas complejos en niveles progresivos de detalle (DFD 0, DFD 1, etc.). El DFD proporciona una forma de representar los pasos involucrados en un proceso de negocio o la operación de un sistema de software propuesto.La Figura 2-20 muestra el nivel 0 DFD para el sistema Chemical Tracking System.

Figura 2-20 Nivel 0 del Diagrama de Flujo de Datos para Chemical Tracking System



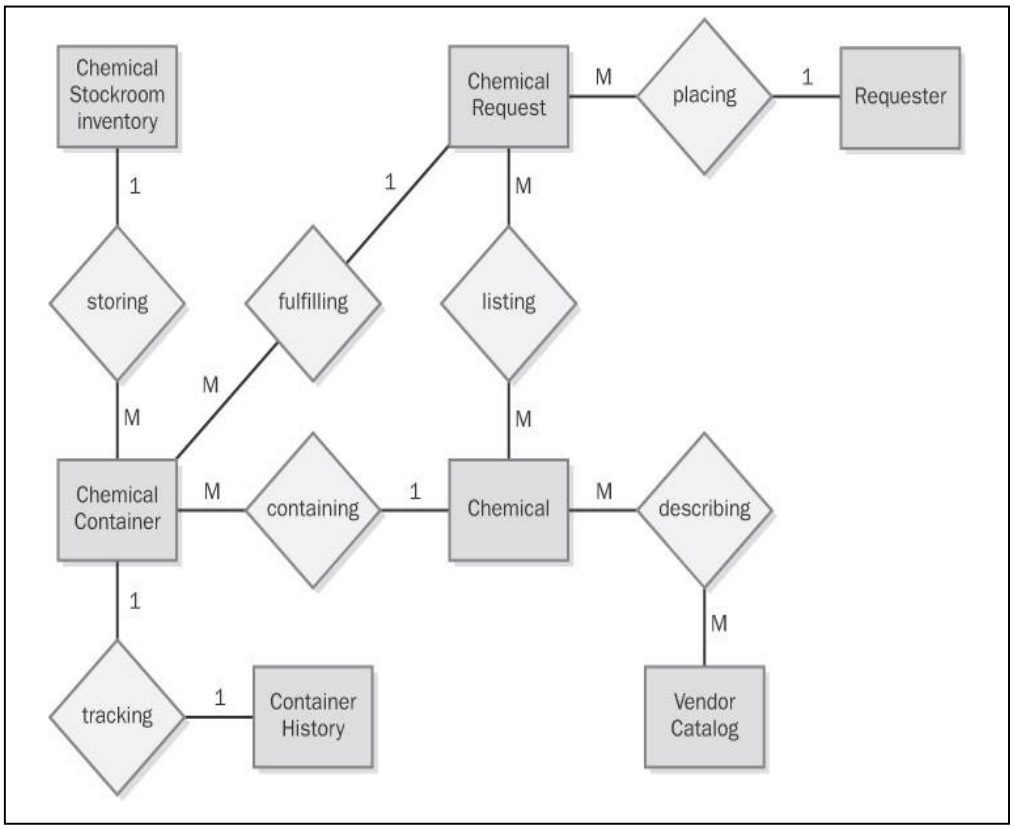
Fuente: Karl E. Wiegers, 2003

2.2.7.4 Diagrama Entidad-Relación

Un modelo de datos representa la relación que existe entre los datos del sistema. Un modelo comúnmente usado para el modelado de datos en el diagrama entidad-relación, o DER (Wieringa, 1996). Si el DER representa grupos lógicos de información del dominio del problema y sus interconexiones, se esta utilizando el DER como herramienta para el análisis de requerimientos. El análisis del DER ayuda a entender y comunicar los componentes de datos del negocio o sistema, sin la implicar que el producto contendrá necesariamente una base de datos. En constaste cuando se crea un DER durante el diseño del sistema, se define la estructura lógica o física de la base de datos del sistema.

Las entidades son conceptos físicos (incluidas personas) o agregaciones de datos que son importantes en el negocio que se esta analizando, o el sistema que se intenta construir. (Robertson and Robertson, 1994). Las entidades son nombradas como sustantivos singulares y se demuestran en un rectángulo. La Figura 2-21 muestra una porción del diagrama entidad-relación para el sistema Chemical Tracking. La cardinalidad o multiplicidad, de cada relación es mostrada con una letra o número sobre las líneas que conecta las relaciones y entidades.

Figura 2-21 Parcial Diagrama Entidad-Relación para Chemical Tracking System



Fuente: Karl E. Wiegers, 2003

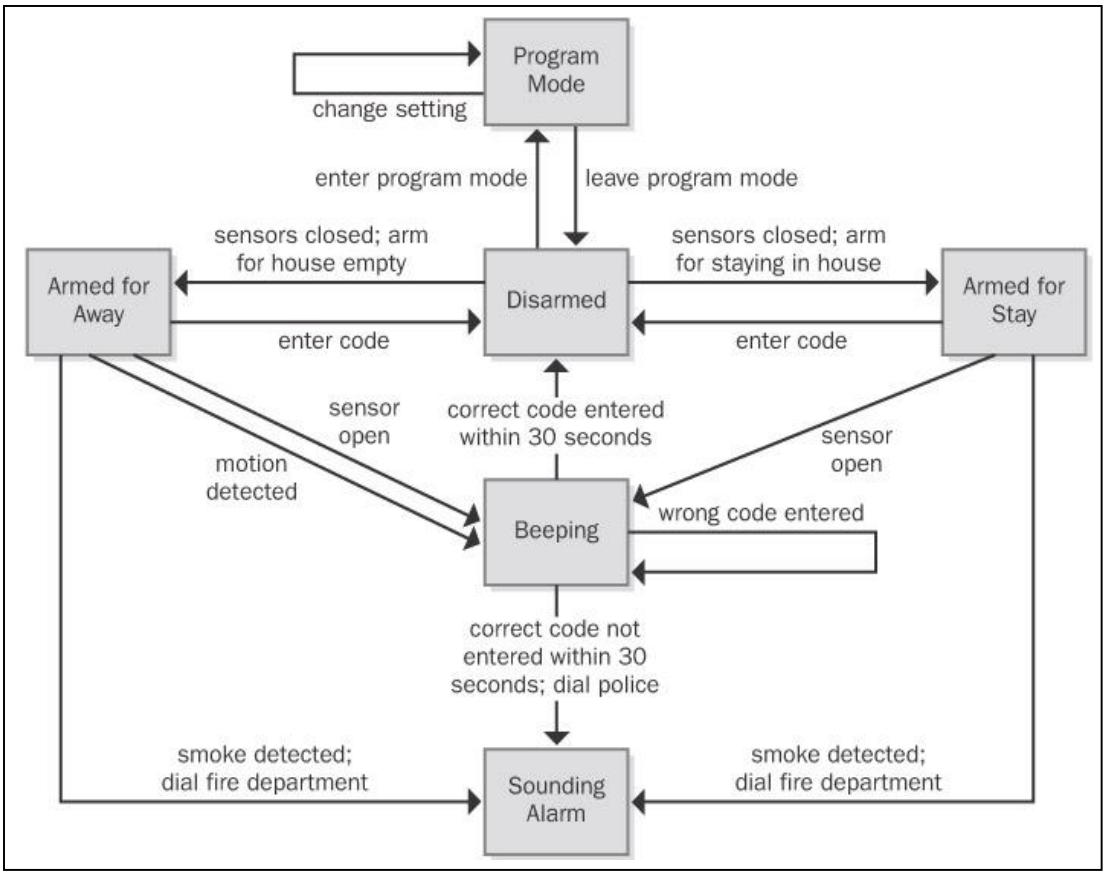
2.2.7.5 Diagrama de Transición de Estado

Todos los sistemas de software implican una combinación de un comportamiento funcional, manipulación de datos y cambios de estado. Un cambio de estado puede tomar lugar cuando los criterios definidos son satisfechos, por ejemplo la recepción de un estimulo especifico de una entrada bajo ciertas condiciones. Una diagrama de transición de estado proporciona una representación consistente, completa y no ambigua de una maquina de estado-finito. Una técnica relacionada es el diagrama de estados incluido en el Lenguaje de Modelado Unificado (UML). El DTE contiene tres tipos de elementos:

- Posible estado del sistema, mostrado por rectángulos.
- Cambios o transiciones permitidas de estados, mostradas como flechas que conectan rectángulos.
- Eventos o condiciones que causan la transición, mostrados como etiquetas de texto o en cada flecha de la transición.

La Figura 2-22 muestra una porción de un DTE para un sistema de seguridad casera.

Figura 2-22 Diagrama de Transición de Estados.

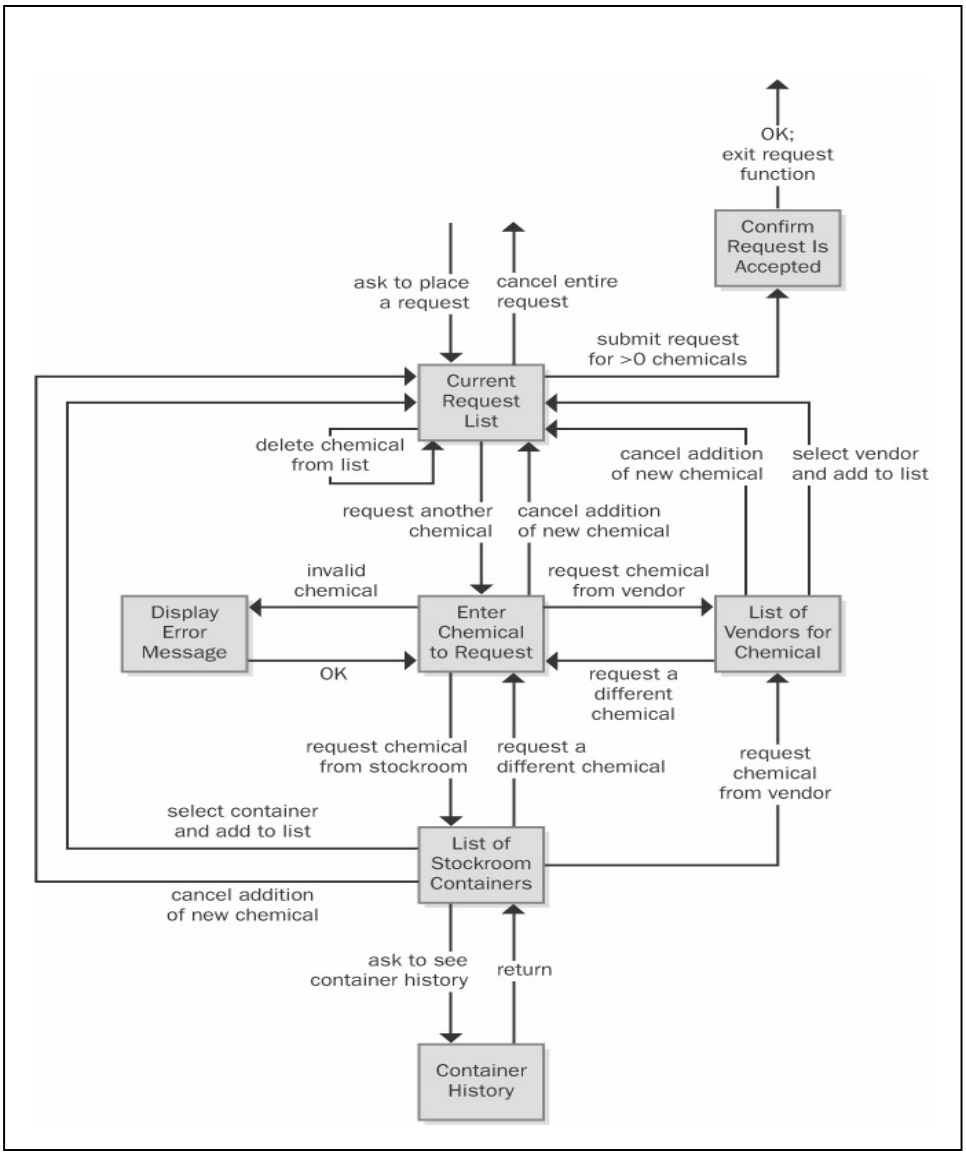


Fuente: Karl E. Wiegers, 2003

2.2.7.6 Mapa de Dialogo

Muchas interfaces de usuarios pueden ser modeladas como un diagrama de transición de estados, llamados mapas de dialogo (Wasserman, 1985, Wiegers, 1996ª). Un mapa de dialogo representa un diseño de interfaz utilizado en un alto nivel de abstracción. Este muestra los elementos de dialogo en el sistema y la ligas de navegación entre ellos, pero no muestra los diseños detallados de la pantalla. La figura Figura 2-23 muestra un mapa de dialogo para el sistema Chemical Tracking.

Figura 2-23 Mapa de Dialogo para el Sistema Chemical Tracking



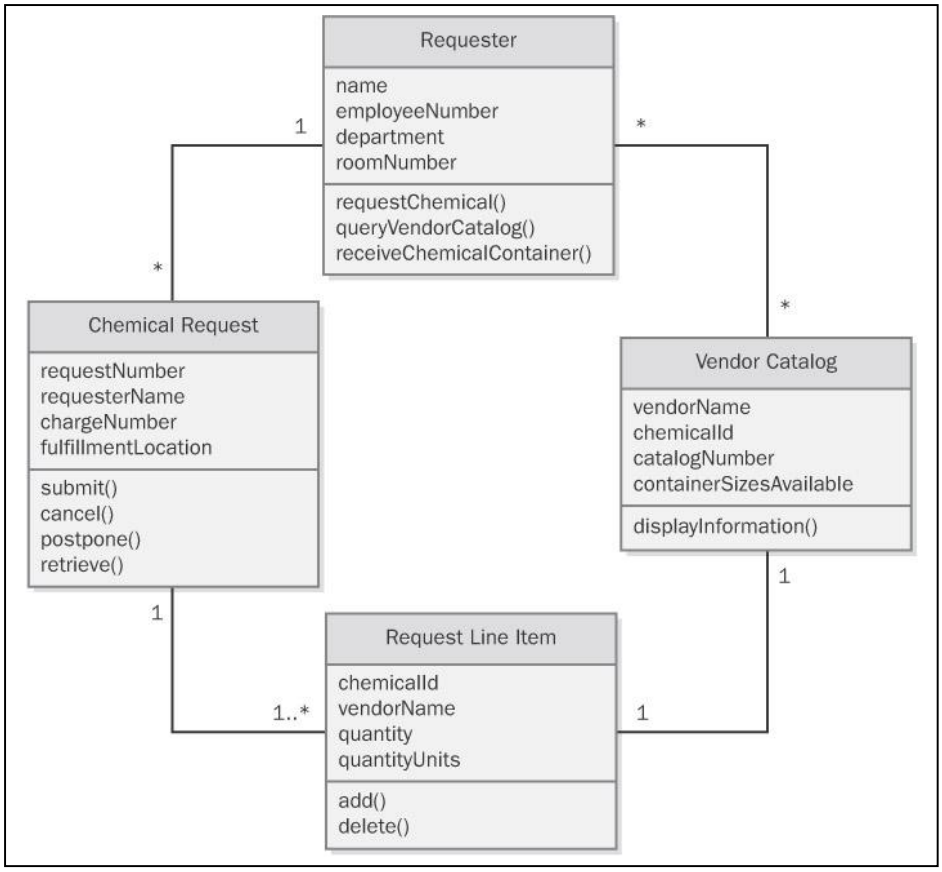
Fuente: Karl E. Wiegers, 2003

2.2.7.7 Diagrama de Clases

Los diagramas de clases son una forma gráfica de representar las clases identificadas durante el análisis orientado a objetos y la relación entre ellas. El estándar del modelado orientado a objetos es el Lenguaje de Modelado Unificado (Booch, Rumbaugh, Jacobson, 1999). Para el nivel de abstracción en el análisis de requerimientos, se pude utilizar la notación de UML para dibujar diagramas de

clases, como lo muestra la Figura 2-23, la cual es una porción del sistema Chemical Tracking.

Figura 2-24 Diagrama de Clases para el Sistema Chemical Tracking



Fuente: Karl E. Wiegers, 2003

Los atributos asociados con cada clase son mostrados en la parte central de cada clase. Las operaciones son servicios que las clases solicitantes pueden realizar. Estas son mostradas en la parte inferior de cada clase. Las líneas representan la asociación entre las clases. Y los números entre las líneas representan la multiplicidad de la asociación.

2.2.7.8 Tabla y Árboles de Decisión

Las tablas de decisión y árboles de decisión son dos técnicas alternativas para representar que es lo que el sistema debe hacer cuado la lógica y decisiones complejas están en juego (Davis, 1993). Las tablas de decisión lista varias valores para todos los factores que influyen en el comportamiento e indica las acciones

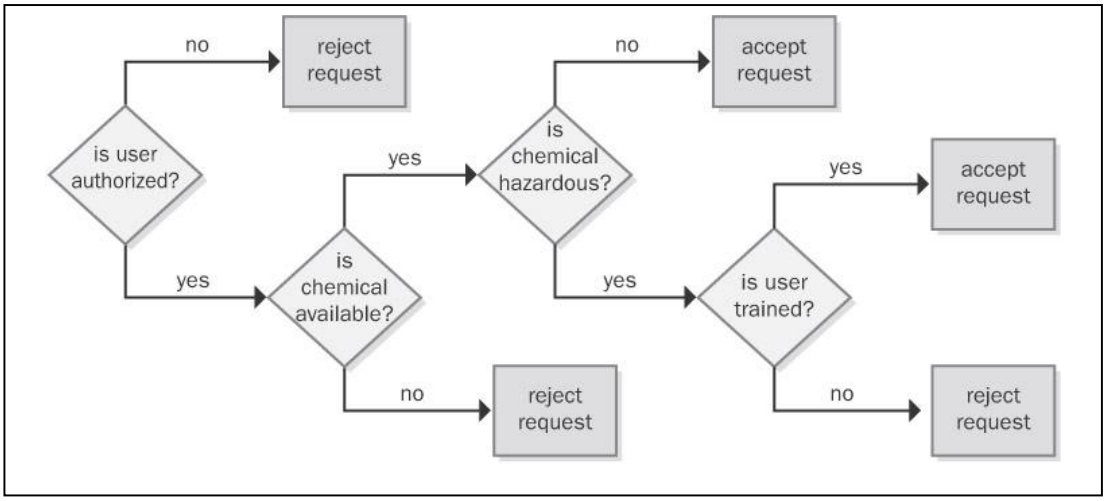
previstas del sistema en respuesta a cada combinación de factores. La Figura 2-25 muestra una tabla de decisión en la que se indica si se debe aceptar o rechazar un nuevo producto químico. La Figura 2-26 muestra un árbol de decisión que representa la misma lógica que la tabla de decisión. Aceptar o rechazar el producto químico.

Figura 2-25 Tabla de Decisión para el Sistema Chemical Tracking

	Requirement Number				
Condition	**1**	**2**	**3**	**4**	**5**
User is authorized	F	T	T	T	T
Chemical is available	—	F	T	T	T
Chemical is hazardous	—	—	F	T	T
Requester is trained	—	—	—	F	T
Action					
Accept request			X		X
Reject request	X	X		X	

Fuente: Karl E. Wiegers, 2003

Figura 2-26 Árbol de Decisión para el Sistema Chemical Tracking



Fuente: Karl E. Wiegers, 2003

2.2.8 Más Allá de la Funcionalidad - Atributos de Calidad de Software

Los clientes generalmente no presentan sus expectativas de calidad explícitamente, aunque la información que proveen durante la licitación proporciona algunas pistas sobre qué tienen en mente. La calidad, en sus muchas dimensiones, debe ser definida tanto para los clientes y aquellos que construirán, harán pruebas, y mantendrn el software actualizado.

2.2.8.1 Atributos de Calidad

Existen algunos atributos de calidad en ambas categorías (clientes y desarrolladores) que cada proyecto debe considerar. Como se puede apreciar en la Tabla 2-8.

Tabla 2-8 Atributos de Calidad de Software

Importante Principalmente a Usuarios	Importante Principalmente a Desarrolladores
Disponibilidad	Mantenimiento
Eficiencia	Portabilidad
Flexibilidad	Re-Usabilidad
Integridad	Propiedad de ser Probado (Testability)
Interoperabilidad	
Confiabilidad	
Robustez	
Usabilidad	

Algunos atributos son críticos para sistemas arraigados (eficiencia y confiabilidad), mientras que otros podrían ser especialmente pertinentes para Internet y aplicaciones de mainframe (disponibilidad, integridad, y mantenimiento) o para sistemas desktop (interoperabilidad y usabilidad).

Entonces definir las metas del usuario y desarrollador en términos de estos atributos esenciales con el propósito de que los diseñadores puedan tomar las opciones apropiadas.

Documentar cualquiera de las metas de calidad global en la sección 5.4 (Atributos de Calidad de Software) del SRS y relacionar metas específicas con las características, casos de uso, o requerimientos funcionales individuales.

2.2.8.2 Definir los Atributos de Calidad

Los analistas luego trabajaban con los usuarios para hacer requerimientos específicos, medibles, y verificables para cada atributo (Robertson y Robertson,

1997). El *Estándar de la IEEE para una Metodología de Métricas de Calidad de Software* presenta una propuesta para definir requerimientos de calidad de software en el contexto de un framework de métricas de calidad global (IEEE, 1992).

2.2.8.2.1 Atributos Importantes para los Usuario

Disponibilidad: es una medida del tiempo planeado durante el cual una computadora o una maquina está propiamente en funcionamiento o disponible para el uso. Los períodos de mantenimiento programados afectan la disponibilidad.

Los requerimientos de disponibilidad son más complicados y más importantes para los sitios Web o aplicaciones globales con usuarios mundiales.

Eficiencia: es una medida de cuán bien utiliza el sistema la capacidad del procesador, el espacio de disco, la memoria, o el ancho de banda de comunicación (Davis1993). La eficiencia está relacionada con el rendimiento. Si un sistema consume demasiados recursos disponibles, los usuarios se tropezarán con un rendimiento degradado, una señal visible de la ineficiencia. El mal rendimiento de una base de datos una molestia para el usuario que está esperando una consulta eficiente. Pero los problemas de rendimiento también pueden representar riesgos serios para la seguridad (safety), como cuando un sistema de control de proceso de tiempo real está sobrecargado.

Flexibilidad: mide qué fácil es añadir las nuevas capacidades al producto. Si los desarrolladores prevén hacer muchas mejoras, pueden escoger propuestas de diseño que maximicen la flexibilidad del software. Este atributo es esencial para los productos que son desarrollados en un modo incremental o iterativo a través de una serie de lanzamientos sucesivos o por la creación de prototipos evolutivos.

Integridad: la integridad - que abarca la seguridad, se las arregla con el acceso no autorizado bloqueando las funciones de sistema, previniendo la pérdida de información, asegurando que el software esté protegido de la infección de virus, y protegiendo la privacidad y la inocuidad (safety) de los datos ingresados en el sistema. La integridad es un asunto muy importante con el software de Internet. Los usuarios de sistemas de comercio electrónico quieren que la información de su tarjeta de crédito sea segura. Los requerimientos de contraseña para el control de acceso son un buen ejemplo.

Interoperabilidad: demuestra cuán fácilmente pueden cambiar los datos o los servicios del sistema con otros sistemas. Para valorar la interoperabilidad, se tiene que saber qué otras aplicaciones emplearán los usuarios conjuntamente con el producto y qué datos esperan cambiar.

Confiabilidad: la probabilidad de que el software se ejecute sin fracaso por un periodo de tiempo específico es conocida como confiabilidad (Musa, Lannino, and Okumoto, 1987). La robustez es a veces considerada un aspecto de la confiabilidad. Las formas de medir la confiabilidad del software incluyen porcentajes de las operaciones que son terminadas correctamente y la longitud promedio del tiempo en que el sistema corre antes de fallar.

Robustez: es el grado en el cuál un sistema continúa funcionando apropiadamente cuando se confronta con entradas inválidas, defectos en el software conectado o los componentes de hardware, o condiciones operativas inesperadas. El software robusto se recupera del problema gentilmente y es indulgente con los errores del usuario.

Usabilidad: facilidad de uso e interacción hombre-computadora, la usabilidad aborda innumerables factores que constituyen lo que los usuarios a menudo describen como usuario - simpatía.

2.2.8.2.2 Atributos Importantes para los Desarrolladores

Mantenimiento: indica qué fácil es corregir un defecto o modificar el software. El mantenimiento depende de cuán fácilmente el software puede ser entendido, cambiado, y probado. Está estrechamente relacionado con la flexibilidad y la propiedad de ser probado.

Portabilidad: el esfuerzo requerido para pasar un fragmento de software de un ambiente operativo a otro, es una medida de la portabilidad.

Re-Usabilidad: indica el relativo esfuerzo involucrado en convertir un componente de software para usarlo en otras aplicaciones.

Testability: hace referencia a la facilidad con la cual los componentes de software o el producto integrado pueden ser puestos a prueba para buscar defectos. Testability es también importante si el producto será modificado a menudo porque pasará por la prueba de regresión frecuente para determinar si los cambios dañaron cualquier funcionalidad existente.

2.2.8.3 Requerimientos de Rendimiento

Los requerimientos de rendimiento definen cuán bien o cuán rápidamente el sistema debe efectuar funciones específicas. Los requerimientos de rendimiento abarcan la velocidad (el número de veces en que responde la base de datos, por ejemplo), el procesamiento (transacciones por segundo), la capacidad (cargas de uso simultáneas), y el cronometraje (las demandas en tiempo real difíciles).

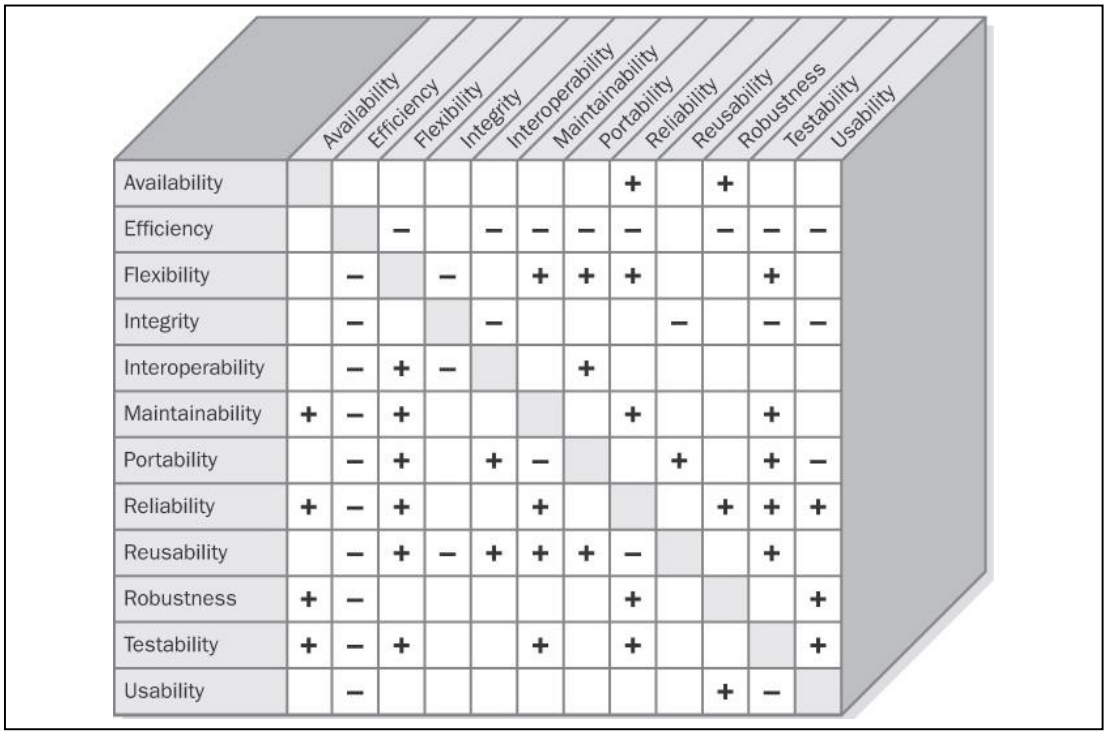
2.2.8.4 Definir Requerimientos No-Funcionales usando en Lenguaje

Para abordar el problema de los requerimientos no-funcionales ambiguos e incompletos, el consultor Tom Gilb (1988; 1997) ha desarrollado Planguage, un lenguaje de planeación con un conjunto abundante de palabras clave que permite declaraciones precisas de los atributos de calidad y otros objetivos del proyecto (Simmons, 2001).

2.2.8.5 Intercambio (Trade-Offs) de Atributo

Ciertas combinaciones de atributos tienen trade-offs inevitables. Los usuarios y desarrolladores deben determinar qué atributos son más importantes que otros, y deben respetar esas prioridades constantemente cuando toman las decisiones. Algunas típicas interrelaciones entre los atributos de calidad. Como se puede apreciar en la Figura 2-27.

Figura 2-27 Relación Entre los Atributos de Calidad



Fuente: Karl E. Wiegers, 2003

Un signo positivo en una celda indica que incrementa el atributo en la fila correspondiente teniendo un efecto positivo sobre el atributo en la columna. Un signo negativo en una celda quiere decir que incrementa el atributo en esa fila afectando adversamente el atributo en la columna.

Para llegar al balance óptimo de las características de producto, se debe identificar, especificar, y priorizar los atributos de calidad pertinentes durante licitación de requerimientos.

2.2.8.6 Implementación de Requerimientos No-Funcionales

Aunque los atributos de calidad son requerimientos no-funcionales, pueden resultar de derivar requerimientos funcionales, guías de diseño, u otra clase de información técnica que causarán características de calidad deseadas. Categorías probables de información técnica que diferentes clases de atributos de calidad generarán. Como se puede apreciar en la Tabla 2-9.

Tabla 2-9 Trasladando Atributos de Calidad en Especificaciones Técnicas

Tipos de Atributos de Calidad	Categoría de Información Técnica Probable
Integridad, interoperabilidad, robustez, usabilidad, safety	Requerimiento funcional
Disponibilidad, eficiencia, flexibilidad, rendimiento, confiabilidad	Arquitectura de sistema
Interoperabilidad, usabilidad	Restricción de diseño
Flexibilidad, mantenimiento, portabilidad, re-usabilidad, testability, usabilidad	Guías de diseño
Portabilidad	Restricción de implementación

2.2.9 Reducción de Riesgos por Prototipo

2.2.9.1 Prototipos: Qué y Porqué

Un prototipo de software es solo una porción o modelo del sistema real de un nuevo producto. Los prototipos tienen tres propósitos importantes:

- Clarificar y completar los requerimientos. Usados como herramienta de requerimientos, el prototipo es una implementación preliminar de una parte del sistema que no ha sido entendida.

- Explorar alternativas de diseño. Usada como herramienta de diseño, el prototipo permite a los stakeholders explorar diferentes técnicas de iteración con el usuario, optimizando la usabilidad del sistema, y evaluando las técnicas potenciales de acercamiento.

- Evolucionando el producto. Usada como una herramienta de construcción, el prototipo es una implementación funcional de un subconjunto inicial del producto.

La primera razón para crear prototipos es resolver incertidumbres rápidamente dentro del proceso de desarrollo. Utilizar estas incertidumbres para decidir qué partes del sistema se van a prototipar y lo que usted espera aprender de las evaluaciones del prototipo. Los prototipos ayudan a revelar y resolver ambigüedades en los requerimientos.

2.2.9.2 Prototipos Horizontales

Cuando la gente dice "*prototipo de software*", se esta hablando generalmente de prototipos horizontales de posibles interfaces de usuarios. Un prototipo horizontal es también llamado un prototipo de comportamiento o maqueta. Es llamado horizontal porque no se introduce en todas las capas de la arquitectura o dentro de los detalles del sistema, pero representa principalmente una porción de la interfaz de usuario. Este tipo de prototipo permite explorar algunos comportamientos específicos del sistema previsto, con la meta de aclarar los requerimientos.

Cuando se trabaja con prototipos horizontales, el usuario se debe enfocar ampliamente sobre los requerimientos y no en los flujos de trabajo de los elementos de la pantalla.

2.2.9.3 Prototipos Verticales

Un prototipo vertical, también conocidos como prototipo estructural o prueba de conceptos, implementa a solo una parte de la funcionalidad de la interfaz de usuario a través de capas de servicio. Un prototipo vertical trabaja semejante como el sistema verdadero se supone que va a trabajar ya que toca todos los niéveles de la implementación del sistema. Se desarrolla un prototipo vertical cuando se esta incierto de que el acercamiento arquitectural propuesto, o cuando se quiere optimizar algoritmos, o se evalúa el propósito de un esquema de base de datos propuesto o se prueban requerimientos críticos. El prototipo vertical es usado para explorar interfaces criticas, requerimientos y reducir riesgos durante el diseño.

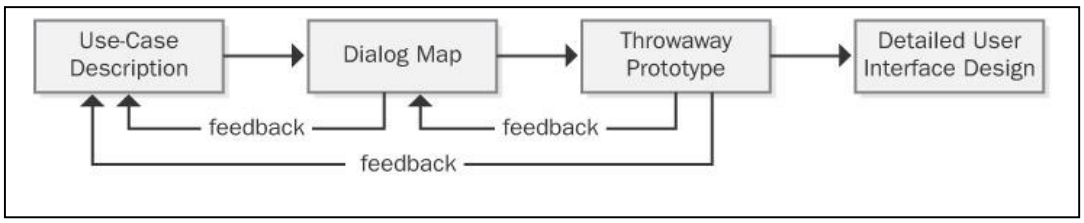
2.2.9.4 Prototipos Desechables

Antes de construir un prototipo, tomar una explicita y bien comunicada decisión, si el prototipo será desechado después de su evaluación o será parte del producto entregado. Se construyen prototipos desechables (o prototipos exploratorios), para contestar preguntas, resolver incertidumbres y mejorar la calidad de requerimientos (Davis, 1993). Cuando de construya un prototipo desechable, este se debe construir lo más rápido y barato que se pueda.

Los prototipos desechables son más apropiados cuando el equipo hace frente a incertidumbres, ambigüedades, estados incompletos, o imprecisiones en los requerimientos.

La Figura 2-28 muestra la secuencia del desarrollo de actividades que va desde el caso de uso hasta el diseño detallado de interfaz de usuario con la ayuda del prototipo desechable. Cada caso de uso incluye una descripción secuencial de las acciones del actor y las respuestas del sistema, en la cual se puede usar un modelo de mapa de dialogo que permite una posible arquitectura de la interfaz de usuario.

Figura 2-28 Secuencias de Actividades Usando un Prototipo Desechable



Fuente: Karl E. Wiegers, 2003

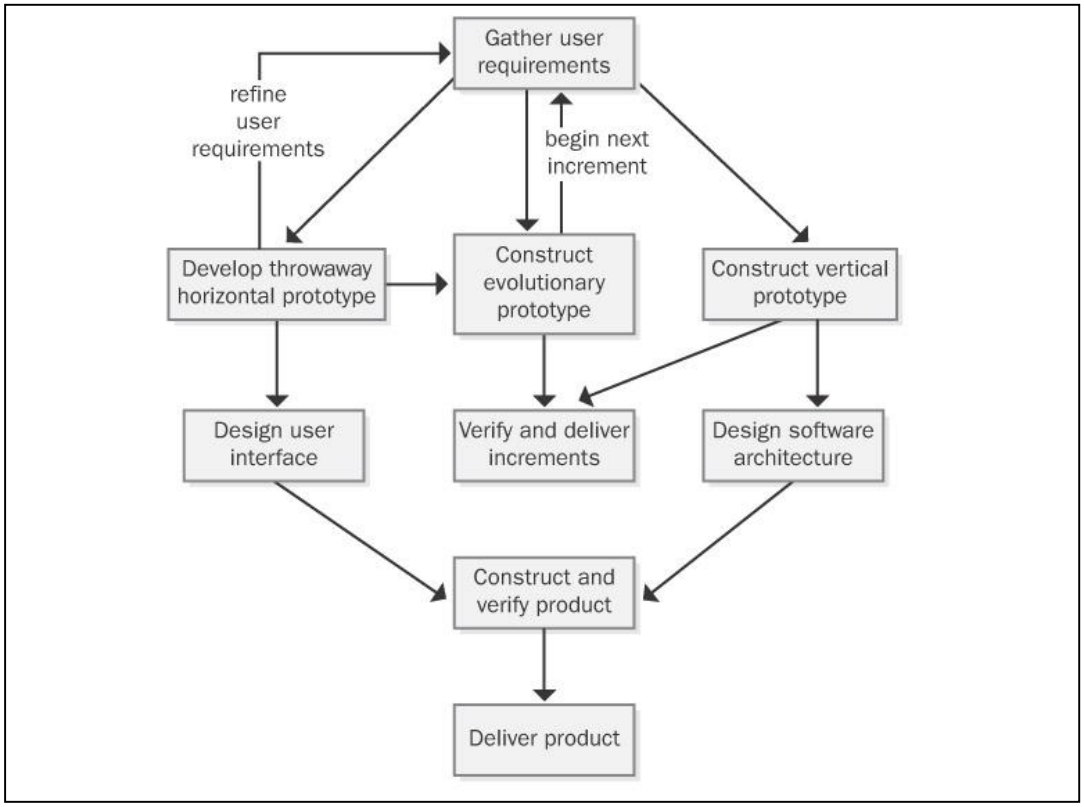
2.2.9.5 Prototipos Evolutivos

En contraste con los prototipos desechables, los prototipos evolutivos proveen una sólida arquitectura para la construcción del producto incremental de acuerdo a como los requerimientos llegan a desarrollarse en el tiempo. Un prototipo evolutivo es un componente del modelo de desarrollo en espiral (Boehm, 1996) y de algunos es proceso de desarrollo orientados a objetos (Kruchten, 1996). Un prototipo evolutivo se debe construir desde un principio con robustez y producir código de calidad.

Pensar en el primer incremento del prototipo evolutivo como un lanzamiento piloto que deje ver si la implementación esta bien entendida y si son estables los requerimientos.

La Figura 2-29 muestra diferentes maneras de combinar varios tipos de prototipos. Por ejemplo se puede usar el conocimiento ganado de una serie de prototipos desechables para clarificar los requerimientos. La Tabla 2-10 resume algunas características típicas de los prototipos desechables, evolutivos, horizontales y verticales.

Figura 2-29 Diferentes Caminos para la Incorporación de Prototipos en el Proceso de Desarrollo de Software



Fuente: Karl E. Wiegers, 2003

Tabla 2-10 Aplicaciones Típicas de Prototipos de Software

	Desechable	Evolutivo
Horizontal	• Clarifican y refinan los requerimientos funcionales y casos de uso. • Identifican funcionalidad faltante. • Exploran la interfaz de usuario.	• Implementan los principales casos de uso. • Implementan casos se uso basados en la prioridad. • Implementan y refinan sitios Web. • Adaptan al sistema rápidamente los cambios en las necesidades de negocio.
Vertical	• Demuestran factibilidad técnica.	• Implementan y hacen crecer la funcionalidad de aplicaciones cliente/servidor y la comunicación entre capas. • Implementan y optimizan los

		principales algoritmos. • Prueban el funcionamiento.

2.2.9.6 Prototipos en Papel y Electrónicos

El prototipo en papel es una manera rápida, barata y de un aprendizaje veloz para explorar lo que puede parecer una porción del sistema puesto en práctica. (Rettig 1994, Omán, 1997). Prototipos en papel ayudan a probar si los usuarios y los desarrolladores llevar acabo una comprensión compartida de los requerimientos.

2.2.9.7 Evaluación de Prototipos

Para la evaluación de prototipos, los scripts de los evaluadores deben alcanzar tareas específicas que se dirijan a las partes del proyecto en las que existe más incertidumbre. Se deben realizar las siguientes preguntas para verificar si el prototipo esta cumpliendo con las tareas específicas:

- ¿La implementación del prototipo tiene la funcionalidad que usted esperaba?
- ¿Qué funcionalidad le hace falta?
- ¿Puede pensar en una condición de error que el prototipo no trata?
- ¿Qué función innecesaria presenta?
- ¿Le parece lógica y completa la navegación?

Se debe cerciorar que la gente que evalúa el prototipo este de acuerdo y que las aproximaciones sean adecuadas. Se deben incluir a los usuarios expertos e inexpertos. Cuando se presente el prototipo a los evaluadores, explicar que se trata solo una parte de la funcionalidad; el resto será ejecutado cuando se desarrolle el sistema real.

2.2.9.8 Factores de Éxito del Prototipo

Para hacer el prototipazo una parte efectiva del proceso de requerimientos, se debe seguir lo siguiente:

- Incluir las tareas del prototipo en el plan de proyecto, calendarizar tiempos y recursos de desarrollo del prototipo, así como también evaluaciones y modificaciones al prototipo.

- Indicar el propósito de cada prototipo antes de que se construya.

- Planear el desarrollo de múltiples prototipos.

- Crear prototipos desechables tan rápido y barato como sea posible.

- No incluir una extensiva entrada de datos en validaciones, técnicas defensivas de código y documentación de código en prototipos desechables.

- No prototipar requerimientos que se entienden, a menos que se exploren alternativas de diseño.

- No esperar que un prototipo sustituya al SRS.

2.2.10 Establecer Prioridades a los Requerimientos

Cada proyecto de software con limitaciones de recurso tiene que definir prioridades con respecto a las capacidades del producto.

2.2.10.1 ¿Por qué Priorizar los Requerimientos?

Priorizar es una manera de arreglárselas con las demandas que compiten por recursos limitados. Establecer la respectiva prioridad de cada capacidad deja planear la construcción para proveer el valor más alto al costo más bajo.

Un administrador de proyecto debe balancear el alcance de proyecto deseado con las restricciones de calendario, presupuesto, personal, y metas de calidad. Una forma de lograr esto es retirar - o postergar para un lanzamiento posterior - requerimientos de baja prioridad cuando nuevos requerimientos más esenciales son aceptados o cuando otras condiciones de proyecto cambian.

2.2.10.2 Los Juegos que la Gente Juega con las Prioridades

En realidad, algunas capacidades del sistema son más esenciales que otras. Pero los clientes establecerán quizás el 85% de los requerimientos como prioridad alta, el 10% como prioridad mediana, y el 5% como prioridad baja. Esto no da mucha flexibilidad al administrador de proyecto. Si casi todos los requerimientos son realmente de máxima prioridad, el proyecto tiene un alto riesgo de no ser completamente exitoso y por consiguiente se tiene que planear. Para alentar a representantes del cliente a identificar requerimientos de baja prioridad, el analista puede hacer las siguientes preguntas:

- ¿Hay alguna otra manera de satisfacer la necesidad que este requerimiento aborda?

- ¿Cuál sería la consecuencia de omitir o postergar este requerimiento?

- ¿Cuál sería el impacto sobre los objetivos de negocio del proyecto si este requerimiento no fuera implementado inmediatamente?

- ¿Por qué sería un usuario desdichado si este requerimiento fuera postergado a un lanzamiento posterior?

2.2.10.3 Escala de Priorización

Una propuesta común para dar prioridad a los requerimientos es agrupándolos en tres categorías: prioridad alta, prioridad mediana y prioridad baja. Tales escalas de prioridad son subjetivas e imprecisas. Los stakeholders deben ponerse de acuerdo sobre qué representa cada nivel en la escala que usan.

Una manera de determinar la prioridad es considerar dos dimensiones: de *importancia* y de *urgencia* (Covey, 1989). Cada requerimiento puede ser considerado como importante o no importante y como urgente o no urgente. Estas alternativas producen cuatro combinaciones posibles, que se pueden usar para definir una escala de prioridad. Como se puede apreciar en la Tabla 2-11.

Tabla 2-11 Prioridad de Requerimientos Basados en Importancia y Urgencia

	Importante	No Importante
Urgente	Prioridad Alta	¡No Hacer Éstos!
No Urgente	Prioridad Media	Prioridad Baja

Los requerimientos de *prioridad alta* son importantes y urgentes. Las obligaciones contractuales o legales podrían ordenar que el requerimiento deba ser incluido, o podría haber razones de negocio obligatorias de implementarlo inmediatamente.

Los requerimientos de *prioridad media* son importantes (el usuario necesita la capacidad) pero no urgentes (pueden esperar un lanzamiento posterior).

Los requerimientos de *prioridad baja* no son importantes (el usuario puede vivir sin la capacidad si es necesario) y no urgentes (el usuario puede esperar, quizás para siempre).

Los requerimientos en el cuarto cuadrante parecen ser urgentes pero no son realmente importantes. No malgastar el tiempo trabajando en éstos. No añaden el valor suficiente al producto.

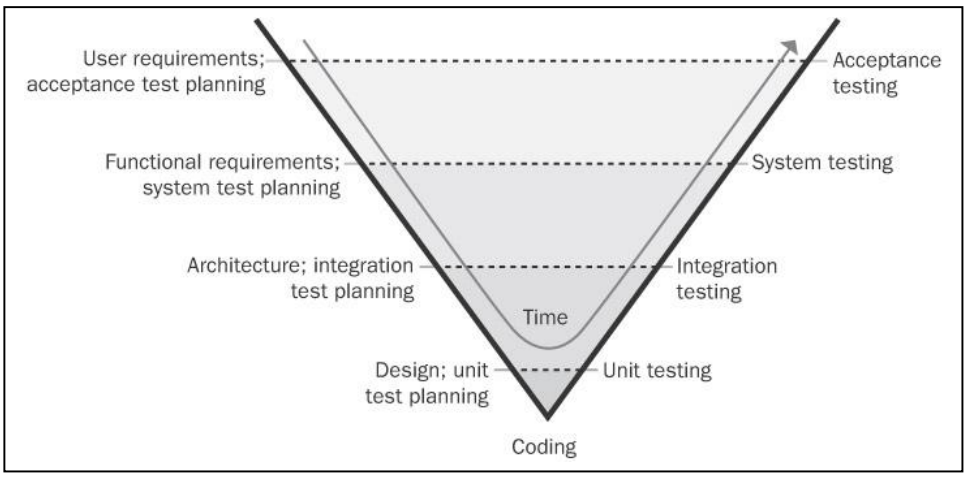
Incluir la prioridad de cada requerimiento en las descripciones de caso de uso, el SRS, o en la base de datos de requerimientos.

2.2.11 Validando los Requerimientos

La mayoría de los desarrolladores han experimentado la frustración de empezar a preguntar sobre la implementación de los requerimientos, que eran ambiguos o incompletos. Si no pueden conseguir la información que se necesita, los desarrolladores tienen que hacer sus propias interpretaciones, que no siempre son las correctas.

Figura 2-30 ilustra el modelo en V del desarrollo de software, la cual muestra que las actividades de prueba empiezan en paralelo con las correspondientes actividades de desarrollo (Davis, 1993). Este modelo indica que las pruebas de aceptación están basadas en los requerimientos de usuario, las pruebas del sistema esta basada en los requerimientos funcionales y las pruebas de integración están basadas en los requerimientos de sistema de arquitectura.

Figura 2-30 Modelo en V de Desarrollo de Software



Fuente: Karl E. Wiegers, 2003

La validación de requerimientos es el cuarto componente del desarrollo de requerimientos junto con la licitación, análisis y especificación. Las actividades de validación de requerimientos deben procurar asegurar que:

- El SRS describe las capacidades previstas del sistema y las características que van a satisfacer las necesidades de varios stakeholders.
- Los requerimientos de software fueron correctamente derivados de los requerimientos del sistema, reglas de negocio, u otras fuentes.
- Los requerimientos son completos y tienen una alta calidad.
- Todas las representaciones de los requerimientos son consistentes unos con otros.
- Los requerimientos proveen una adecuada base para proceder con el diseño y construcción.

La validación asegura que se exhiban las características deseables de un excelente statement de requerimientos (completo, correcto, factible, necesario,

prorizable, no ambiguo y verificable) y de excelentes especificaciones (completo, consistente, modificable y traceable).

2.2.11.1 Revisando Requerimientos

Una revisión es aquella en la que se examina el producto de software por otra persona con excepción del autor, esto es; para encontrar los problemas del producto, conocidos también como peer reviews. Una revisión del documento de requerimientos es una poderosa técnica para identificar requerimientos ambiguos, requerimientos que no fueron definidos claramente y no están listos para que el diseño del producto de software empiece.
Existen diversos tipos de peer reviews con una gran variedad de nombres (Wiegers, 2002a). Las revisiones informales son útiles para educar a las personas sobre el producto y mejorar tomando los defectos que se encontraron. Un enfoque a las revisiones informales incluye:

- Un peer deskcheck, en el cual se pide que un compañero u colega observe el producto de trabajo.
- Un passaround, en el cual se invita a varios compañeros o colegas que examinen el producto de trabajo concurrentemente.
- Un walkthrough, es aquel en el cual el autor describe producto de trabajo y solicita comentarios sobre el producto.

2.2.11.1.1 Proceso de Inspección

La inspección se ha reconocido como una de las mejores prácticas en la industria de software (Brown 1996). Cualquier producto de trabajo de software puede ser inspeccionado, incluyendo los documentos de requerimientos y diseño, así como documentación de código fuente y pruebas, y planes de proyecto.
La inspección es un proceso progresivo bien definido. Esto implica un pequeño grupo de participantes bien entrenados, quienes cuidadosamente examinan los productos de trabajo para detectar defectos y oportunidades de mejora. Las inspecciones proveen un camino hacia la calidad en las cuales los documentos deben ser revisados antes de ser línea base.

2.2.11.1.1.1 Participantes

Los participantes en la inspección deben representar cuatro aproximaciones (Wiegers, 2002a):

- El autor del producto de trabajo.
- El autor de cualquier producto de trabajo predecesor.

- Personas quienes trabajarán basadas en el producto inspeccionado.

- Personas quienes son responsables para los productos de trabajo que interfieren con elementos externos.

2.2.11.1.1.2 Roles en la Inspección

Todos los participantes en una inspección, incluyendo al autor, buscan defectos y una oportunidad de mejora. Algunos miembros del equipo de inspección realizan los siguientes papeles durante la inspección:

Autor. El autor es el que crea o mantiene el producto de trabajo que se inspeccionará. El autor del SRS es usualmente el analista quien recolecto los requerimientos del cliente y escribió las especificaciones. Al autor del documento no se le asigna ningún otro tipo de rol.

Moderador. El moderador o líder de la inspección, planea la inspección con el autor, coordina las actividades, y facilita la reunión para la inspección. El moderador distribuye el material que va a ser inspeccionado a todos los participantes días antes de la reunión para su inspección.

Lector. Un inspector es asignado con el rol de lector. Durante la reunión para la inspección, el lector lee el SRS, mientras que los otros participantes precisan potenciales defectos y problemas.

Secretario. El secretario, utiliza las formas estándar para documentar los problemas y defectos encontrados durante la inspección. El secretario debe revisar que es lo que se escribió junto con los otros participantes en la inspección para conformidad de todos.

2.2.11.1.1.3 Criterios de Entrada

Se esta listo para la inspección de documentos cuando se tienen satisfechos los específicos prerrequisitos. Estos criterios de entrada fijan algunas especificaciones claras para que los autores sigan mientras se preparan para la inspección. Los siguientes puntos son algunos criterios de entrada sugeridos para la inspección de los documentos de requerimientos:

- El documento se conforma del template estándar.

- El documento ha sido verificado en cuanto a sintaxis.

- El autor ha examinado visualmente el documento para cualquier error.

- Cualquier documento anterior o alguna referencia que el inspector requiera estará disponible.

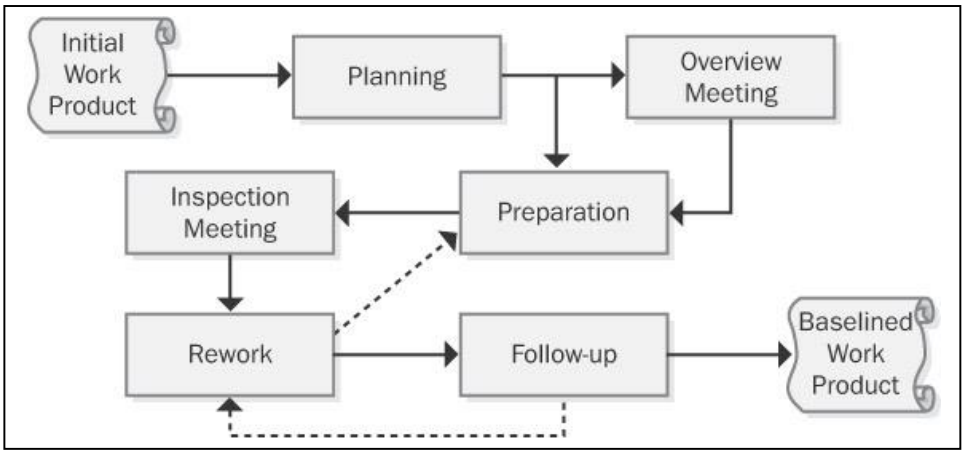
- Los números de línea son impresos en el documento para facilitar referirse a una específica localización durante la inspección.

- El moderador no encontró más de tres defectos en diez minutos de revisión, en una muestra representativa del documento.

2.2.11.1.1.4 Etapas de la Inspección

La inspección es un proceso de muchas etapas, como lo muestra la Figura 2-31.

Figura 2-31 Etapas Inspección



Fuente: Karl E. Wiegers, 2003

Planeación. El autor y el moderador planean juntos la inspección, determinan quienes deben participar, que material deben recibir los inspectores antes de la revisión, y cuantas reuniones serán necesarias para inspeccionar todo el material. Preparación de la reunión de la inspección. Durante los preparativos para la reunión, el autor describe a fondo el material que se inspeccionará y los objetivos específicos de la inspección.

Preparación. Antes de la reunión para la inspección, cada inspector examina los productos para identificar posibles defectos y problemas, usando checklists.

Reunión. Durante la reunión, el lector condice y esta de acuerdo con los otros inspectores a través del SRS, leyendo un requerimiento a la vez. Si los inspectores no están de acuerdo se tendrá que identificar defectos y acciones para

corregirlos. Los inspectores identifican posibles defectos y otros problemas, el secretario captura todos estos posibles defectos.

Retrabajo. Casi toda actividad de control de calidad revela algunos defectos. El autor debe planear volver a trabajar en los requerimientos una vez pasada la reunión para la inspección.

Seguimiento: Es el paso final de la inspección. El moderador o una persona asignada debe trabajar con el autor para asegurar que todos los problemas fueron resueltos y que los errores fueron corregidos correctamente.

2.2.11.1.1.5 Criterios de Salida

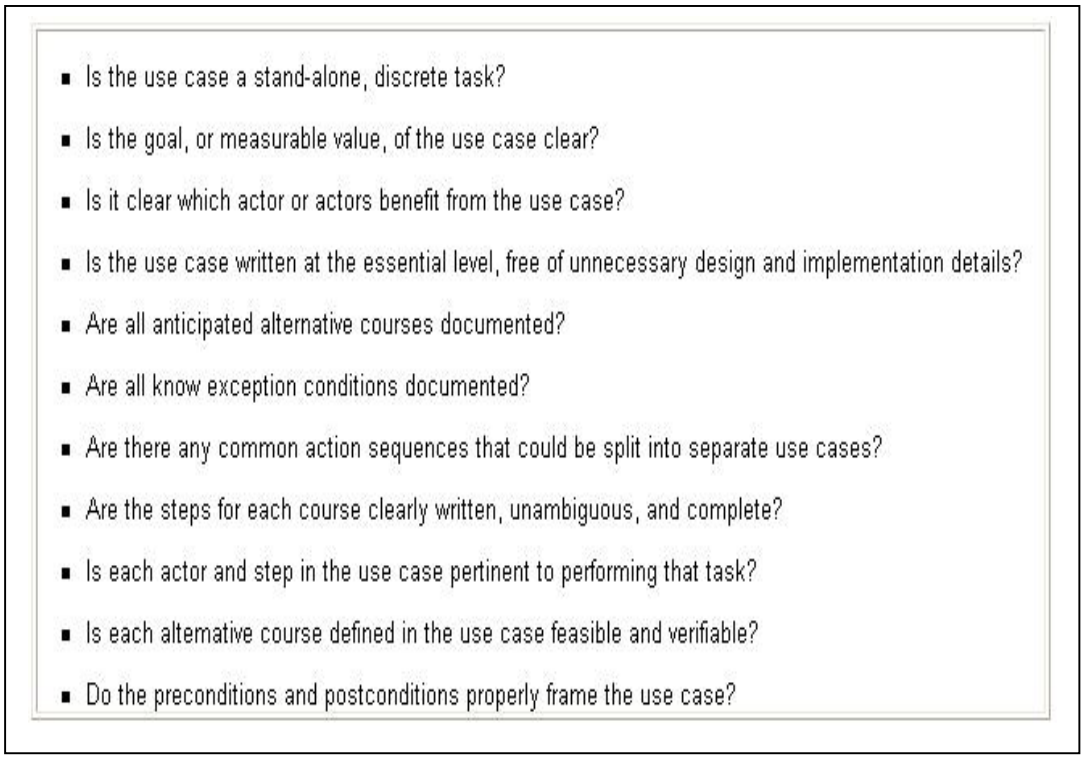
El proceso de inspección debe definir criterios de salida que han sido satisfechos antes de que el moderador declare la inspección terminada. Aquí están algunas posibles sugerencias de criterios de salida para las inspecciones de documentos de requerimientos:

- Se han tratado todos los problemas encontrados durante la inspección.
- Cualquier cambio realizado en los documentos y productos relacionados fueron hechos correctamente.
- El documento revisado ha sido verificado en cuanto a sintaxis.

2.2.11.1.1.6 Checklist de Defectos

Una ayuda para los inspectores al identificar típicos tipos de errores en los productos que se inspeccionan, es desarrollar checklists de defectos para cada documento de requerimientos que la organización crea. Tales checklists se enfocan en problemas históricos frecuentes en requerimientos. La Figura 2-32 muestra un checklist de un caso de uso y la Figura 2-33 presenta un checklist para la inspección del SRS.

Figura 2-32 Checklist de Defectos para los Documentos de Casos de Uso



Fuente: Karl E. Wiegers, 2003

Figura 2-33 Checklist de Defectos para el SRS

Organization and Completeness

- Are all internal cross-references to other requirements correct?
- Are all requirements written at a consistent and appropriate level of detail?
- Do the requirements provide an adequate basis for design?
- Is the implementation priority of each requirement included?
- Are all external hardware, software, and communication interfaces defined?
- Are algorithms intrinsic to the functional requirements defined?
- Does the SRS include all the known customer or system needs?
- Is any necessary information missing from a requirement? If so, is it identified as a TBD?
- Is the expected behavior documented for all anticipated error conditions?

Correctness

- Do any requirements conflict with or duplicate other requirements?
- Is each requirement written in clear, concise, and unambiguous language?
- Is each requirement verifiable by testing, demonstration, review, or analysis?
- Is each requirement in scope for the project?
- Is each requirement free from content and grammatical errors?
- Can all the requirements be implemented within known constraints?
- Are all specified error messages unique and meaningful?

Quality Attributes

- Are all performance objectives properly specified?
- Are all security and fafety considerations properly specified?
- Are other pertinent quality attribute goals explicitly documented and quantified, with the acceptable trade-offs specified?

Traceability

- Is each requirement uniquely and correctly identified?
- Is each software functional requirement traced to a higher-level requirement (for example, system requirement or use case)?

Special Issues

- Are all requirements actually requirements, not design or implementation solutions?
- Are the time-critical functions identified and their timing criteria specified?
- Have internationalization issues been adequately addressed?

Fuente: Karl E. Wiegers, 2003

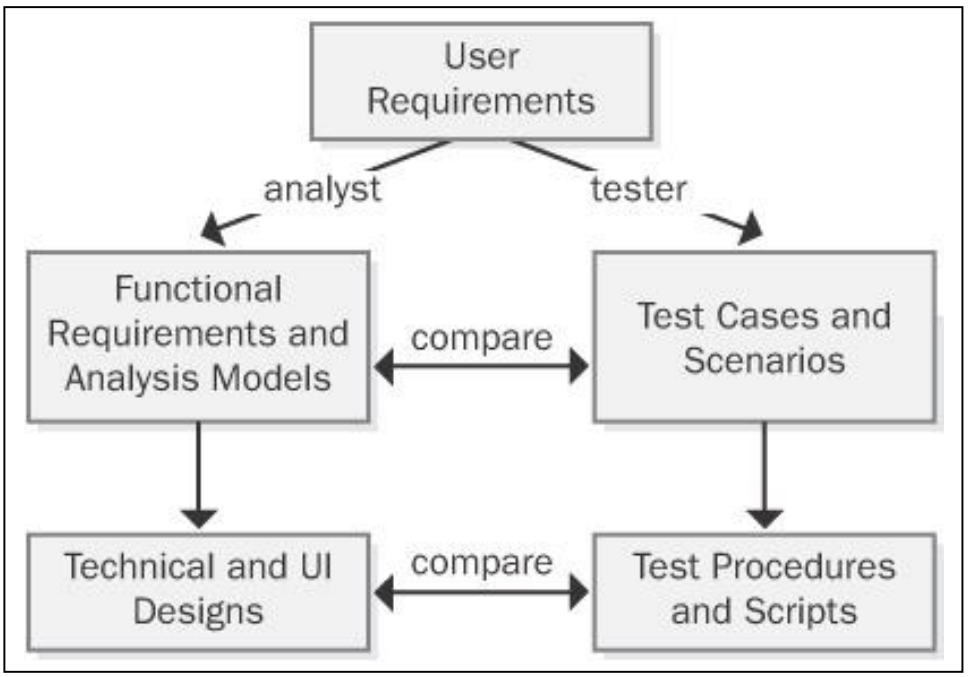
2.2.11.2 Probando los Requerimientos

Es difícil visualizar cómo un sistema funcionará bajo circunstancias específicas únicamente leyendo el SRS. Los casos de prueba se basan en los requerimientos funcionales o derivados de requerimientos de usuario, estos casos de prueba ayudan a todos los participantes en el proyecto a tener un mejor entendimiento del comportamiento del sistema. El simple acto de diseñar casos de prueba revela muchos problemas con los requerimientos incluso si no se han ejecutado las pruebas en un sistema operacional (Beizer, 1990). Si se empieza a desarrollar casos de pruebas tan pronto como los requerimientos se estabilicen, se

encontrarán muchos problemas en los requerimientos los cuales van a poder ser corregidos y no se impactará económicamente al proyecto.

Un analista de requerimientos escribirá los requerimientos funcionales y el tester escribirá los casos de prueba en un punto de partida común, esto es; desde los requerimientos de usuario, tal y como lo muestra la Figura 2-34.

Figura 2-34 Productos de Desarrollo y Pruebas son Derivados de una Misma Fuente



Fuente: Karl E. Wiegers, 2003

2.2.11.3 Definiendo los Criterios de Aceptación

Los desarrolladores de software pueden creer que se ha construido el producto perfecto, pero el cliente es el que tiene la última palabra. Los clientes realizan la prueba de aceptación para determinar si un sistema satisface sus criterios de aceptación (IEEE, 1990). El criterio de aceptación debe evaluar si el producto satisface los requerimientos documentados y esta listo para el funcionamiento en el ambiente previsto (Hsia, Kung, and Sell, 1997).

2.2.12 Desafíos de Desarrollo de Requerimientos Especiales

Se ha descrito el desarrollo de requerimientos como si se empezara con un proyecto de desarrollo de software o sistema nuevo. Sin embargo, muchas organizaciones dedican la mayor parte de su esfuerzo a mantener sistemas

legados existentes o a construir el próximo lanzamiento de un producto comercial establecido. Otras organizaciones construyen pocos sistemas nuevos desde el principio.

2.2.12.1 Requerimientos para Proyectos de Mantenimiento

El mantenimiento consta de modificaciones hechas al software que está actualmente en operación. A veces llamado ingeniería continua o desarrollo continuo, el mantenimiento a menudo consume la mayor parte de los recursos de una organización de software. Los programadores de mantenimiento corrigen los defectos, añaden características o informes nuevos a sistemas existentes, y modifican la funcionalidad para obedecer reglas de negocio reconocidas. Pocos sistemas legados tienen documentación suficiente.

2.2.12.1.1 Empezar a Capturar Información

En ausencia de documentación de requerimientos precisos, el personal de mantenimiento debe aplicar ingeniería inversa, es decir; interpretación de qué hace el sistema desde el código. Acumular la información exacta sobre ciertas partes del sistema actual orientando a que el equipo lleve acabo futuras técnicas más eficientes.

Una técnica útil es dibujar un mapa de diálogo para las nuevas pantallas que se tiene que añadir, mostrando las conexiones de navegación hacia y desde elementos de visualización existentes. Otras técnicas de modelado tal como diagramas de clase e interacción, diagramas de flujo de datos, y diagramas de entidad - relación son también útiles. Un diagrama de contexto o un diagrama de casos de uso describen las entidades externas o los actores que interactúan con el sistema. Otra manera de empezar a llenar el vacío de información es crear las entradas de diccionario de datos cuando se añaden nuevos elementos de datos al sistema y se modifican las definiciones existentes.

2.2.12.1.2 Practicar Nuevas Técnicas de Requerimientos

Todo desarrollo de software está basado en requerimientos. Los proyectos de mantenimiento proporcionan una oportunidad de tratar nuevos métodos a pequeña escala y de poco riesgo de ingeniería de requerimientos. Cuando el próximo proyecto grande llegue, se tendrá un poco de experiencia y confianza en mejores practicas de requerimientos.

Algunas otras técnicas que se pueden probar durante el mantenimiento incluyen:

- Crear un diccionario de datos
- Dibujar modelos de análisis
- Especificar los atributos de calidad y las metas de rendimiento

- Construir prototipos de interfaz de usuario y técnicas
- Inspeccionar especificaciones de requerimientos
- Escribir los casos de prueba de requerimientos
- Definir los criterios de aprobación del cliente

2.2.12.1.3 Haciendo un Seguimiento sobre la Cadena de Rastreabilidad

El rastreo de requerimientos ayudará al programador de mantenimiento determinar qué componentes se tendrían que modificar debido a un cambio en un requerimiento específico.

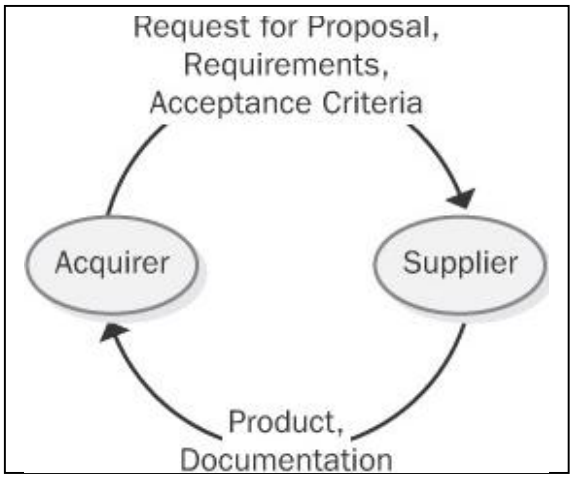
2.2.12.1.4 Actualizar la Documentación

Requerimientos obsoletos y documentos de diseño no son útiles para el futuro mantenimiento. Cualquier representación de requerimientos existentes debe mantenerse actualizada durante el mantenimiento para reflejar las capacidades del sistema modificado.

2.2.12.2 Requerimientos para Proyectos de Outsourcing

La contratación externa para el desarrollo de un producto por una compañía externa requiere requerimientos escritos de buena calidad porque sus interacciones con el equipo de ingeniería son probablemente mínimas. Se estará enviando una solicitud para la propuesta, una especificación de requerimientos, y algunos criterios de aprobación de producto al proveedor, y devolverán tanto el producto terminado y la documentación de soporte. Como se puede apreciar en la Figura 2-35.

Figura 2-35 Requerimientos: Piedra Angular para el Outsourcing



Fuente: Karl E. Wiegers, 2003

No habrá oportunidad para las aclaraciones, toma de decisiones, y cambios diarios que se experimentan cuando los desarrolladores y clientes trabajan en proximidades. Los requerimientos mal definidos y administrados son una causa común de fracaso en proyectos de outsourcing (Wiegers, 2003). Algunas sugerencias cuando se preparan requerimientos para outsourcing son las siguientes:

- Proveer los detalles.
- Evitar la ambigüedad.
- Arreglar puntos de roce con el proveedor.
- Definir un proceso de control - cambio mutuamente aceptable.
- Tiempo de plan para los ciclos múltiples y las revisiones de requerimientos.
- Definir los criterios de aceptación.

2.2.12.3 Requerimientos para Proyectos Emergentes

Los proyectos emergentes son caracterizados por requerimientos inciertos y cambios frecuentes.

Si se está trabajando en un proyecto emergente, considerar usar los siguientes enfoques para los requerimientos:

Especificación de Requerimientos de Usuario Informal
Los requerimientos informalmente documentados son apropiados para los sistemas en evolución siendo desarrollados por equipos pequeños. La metodología ágil llamada Extreme Programming defiende requerimientos registrados sobre tarjetas de índice en forma de historias de usuario simples (Beck 2000; Jeffries, Anderson, and Hendrickson 2001). Este enfoque exige que los clientes comprendan sus requerimientos lo suficientemente bien para describir el comportamiento de sistema en forma de historias.

Cliente en Sitio
Las conversaciones frecuentes entre miembros de equipo y clientes apropiados de proyecto son la manera más efectiva de resolver muchos requerimientos en desacuerdo. La documentación escrita detallada, sin embargo, es un sustituto incompleto para estas comunicaciones continuas. Un dogma fundamental de Extreme Programming es la presencia de un cliente en sitio de tiempo completo, para estas discusiones.

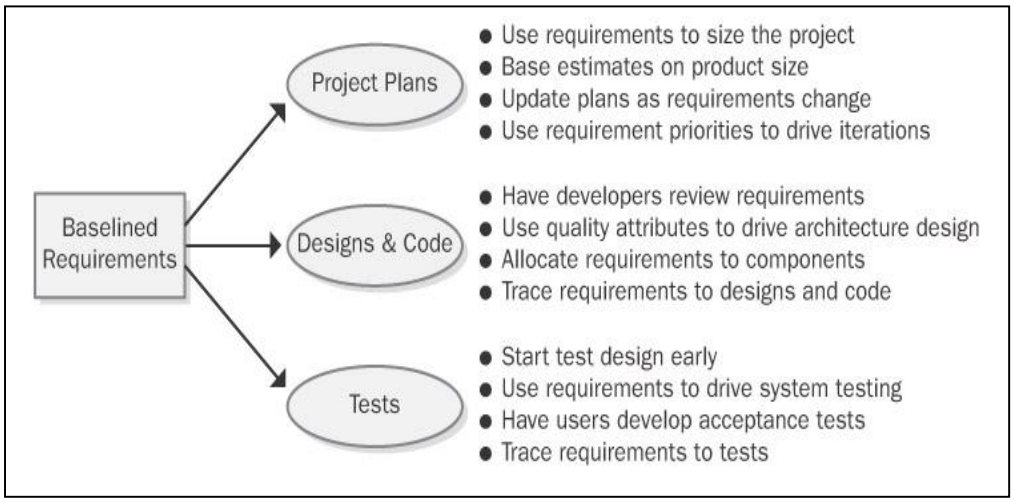
Priorizando Frecuente y Tempranamente
El desarrollo incremental sucede cuando los clientes y desarrolladores colaboran en decidir la secuencia de implementar características. La meta del equipo de desarrollo es conseguir funcionalidad útil y buena calidad en las manos de los usuarios sobre una base regular, así que tienen que saber qué capacidades son planeadas para cada incremento.

Administración de un Cambio Simple
Los procesos de desarrollo de software deben ser tan simples como puedan ser para alcanzar el trabajo bien hecho, pero no más fácil. Un proceso de control - cambio lento no funcionará en proyectos emergentes que demandan modificaciones frecuentes. Racionalizar el proceso de cambio de tal manera que el mínimo número de gente pueda tomar decisiones respecto a solicitudes de cambio tan rápidamente como sea posible.

2.2.13 Más allá del Desarrollo de Requerimientos

Administradores y desarrolladores experimentados entienden el valor de trasladar los requerimientos de software en diseños robustos y planes racionales de proyectos. Los requerimientos como línea base conducen al proyecto a una planeación, un diseño, una codificación y a unas pruebas. Como se puede apreciar en la Figura 2-36.

Figura 2-36 Línea Base de los Requerimientos



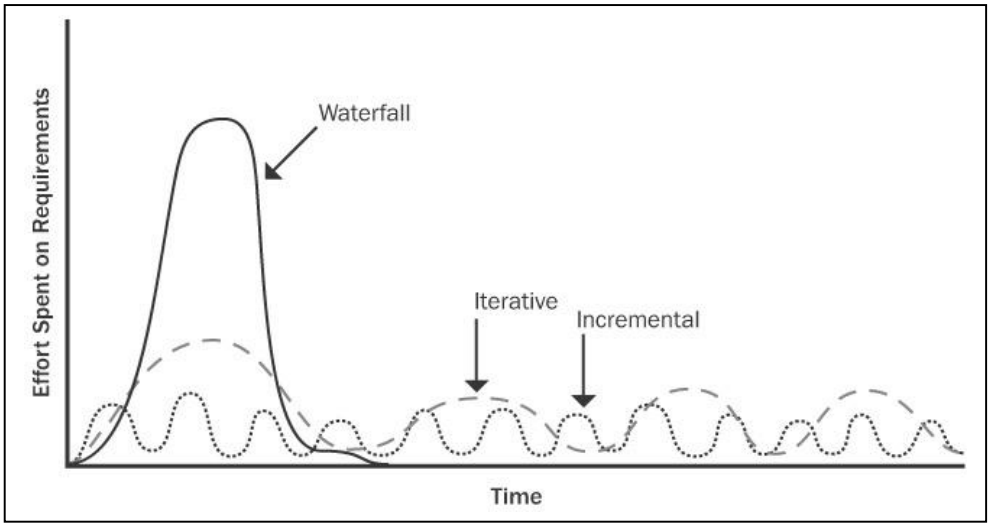
Fuente: Karl E. Wiegers, 2003

2.2.13.1 De los Requerimientos a los Planes del Proyecto

Ya que los requerimientos definen los resultados previstos para el proyecto, se debe basar los planes de proyecto, estimaciones y calendarios en los requerimientos. Recordar, aunque lo más importante del proyecto es que el sistema resuelve sus objetivos de negocio, no todo el esfuerzo en el desarrollo de requerimientos se asigna al principio de proyecto, como lo hacen los modelos del ciclo de vida en cascada o secuencial. Los proyectos que siguen un modelo de ciclo de vida iterativo pasarán tiempo en el desarrollo de requerimientos durante cada iteración del desarrollo del proceso. Los proyectos incrementales de desarrollo lanzan funcionalidades en pocas semanas, así que se tendrán

esfuerzos frecuentes pero pequeños en el desarrollo de requerimientos. La Figura 2-37 muestra como diversos modelos de ciclos de vida asignan esfuerzos a los requerimientos a través de periodos en el desarrollo del producto.

Figura 2-37 Ciclos de Vida



Fuente: Karl E. Wiegers, 2003

2.2.13.1.1 Requerimientos y Estimación

El primer pasó para estimar el proyecto es determinar el tamaño del producto de software. Se pueden basar las estimaciones de tamaño en requerimientos textuales, análisis de modelos, prototipos o interfaz de usuario. Aunque no hay una medida perfecta para el tamaño de software, las siguientes son las métricas más utilizadas:

- Número de requerimientos individualmente testeables (Ilson, 1995).
- Puntos de función y futures pints (Jones, 1996b) o puntos de función 3-D que incorporan datos, funciones y control (Whitmire, 1995).
- El número, el tipo y complejidad de los elementos de las interfaces gráficas de usuarios (GUI).
- Líneas de código estimadas necesarias para implementar requerimientos específicos.
- Una cuenta de clases de objetos u otra métrica orientada a objetos (Whitmire 1997).

2.2.13.1.2 Requerimientos y Calendarios

Los proyectos de software frecuente fallan al resolver sus metas ya que desarrolladores y otros participantes en el proyecto son pobres planeadores, y no porque sean pobres ingenieros de software. Los errores más importantes en la planeación es que pasan por alto tareas comunes como; subestimar en esfuerzo o tiempo, no se hizo un análisis de riesgos y se tuvo un engañoso optimismo infundado. La planeación eficaz de un proyecto requiere de los siguientes elementos:

- Tamaño estimado del producto.
- Productividad del equipo de desarrollo, basado en la historia.
- Una lista de las tareas necesitadas para completar la implementación y verificar las características o casos de uso.
- Requerimientos razonables y estables.
- Experiencia, que ayuda al administrador del proyecto a ajustar para que haya factores intangibles y aspectos únicos en cada proyecto.

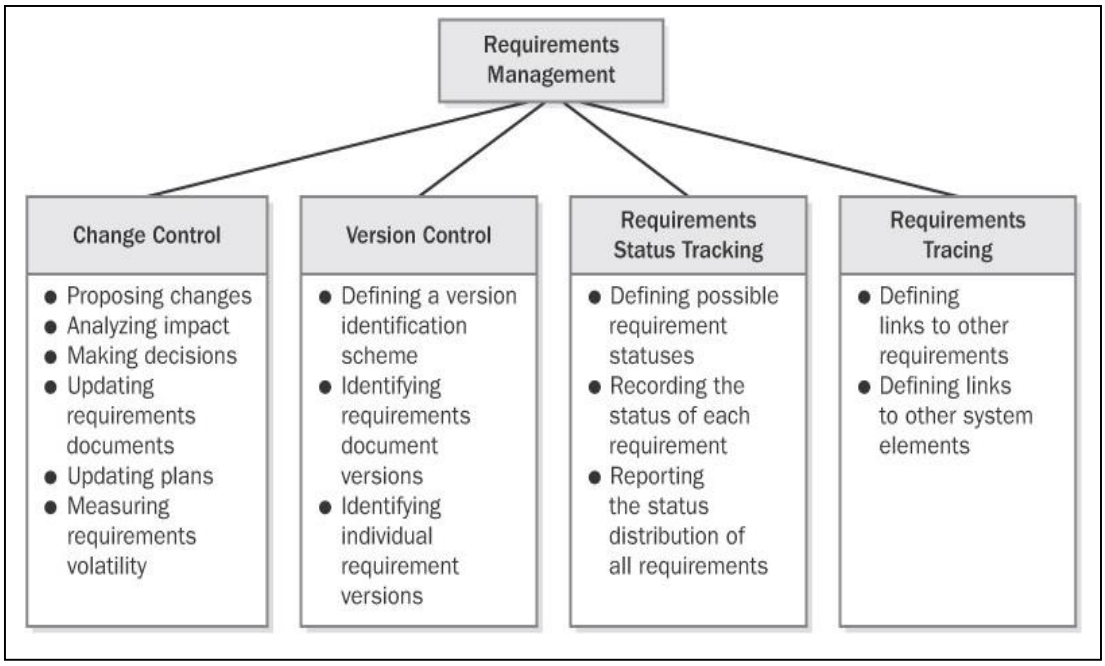
2.3 Administración de Requerimientos de Software

2.3.1 Principios y Prácticas de Administración de Requerimientos

La disciplina de ingeniería de requerimientos se divide en: desarrollo de requerimientos y administración de requerimientos. El desarrollo de requerimientos involucra licitar, analizar, especificar y validar requerimientos de un proyecto de software. El desarrollo de requerimientos incluye un documento de visión y alcance, documentos de casos de uso, una especificación de requerimientos de software, un diccionario de datos y modelos de análisis asociados. Una vez examinado y aprobado, estos artifacts definen la línea base (baseline) de los requerimientos para el esfuerzo de desarrollo, un acuerdo entre el grupo de desarrollo y sus clientes. El proyecto probablemente tendrá contratos adicionales respecto a liberaciones, restricciones, calendarios, presupuestos, o compromisos contractuales.

El acuerdo de requerimientos es el puente entre el desarrollo de requerimientos y la administración de requerimientos. Los miembros de equipo deben tener acceso disponible a los requerimientos actuales durante toda la duración del proyecto. La administración de requerimientos incluye todas las actividades que mantienen la integridad, precisión, y aceptación de los requerimientos de acuerdo como el proyecto avanza. Como se puede apreciar en la Figura 2-38.

Figura 2-38 Principales actividades de la Administración de Requerimientos



Fuente: Karl E. Wiegers, 2003

- Control de cambios en la línea base de requerimientos.
- Mantener los planes del proyecto en curso con los requerimientos.
- Controlar versiones tanto de requerimientos individuales como de documentos de requerimientos.
- Rastrear el estado de los requerimientos en la línea base.
- Mantener los enlaces lógicos entre requerimientos individuales y otros proyectos.

El administrador de proyecto debe negociar cambios en los requerimientos para cumplir con los administradores afectados, los clientes, y otros stakeholders. El proyecto puede responder a requerimientos nuevos o cambiados en varias formas:

- Postergar requerimientos de baja-prioridad.
- Obtener personal adicional.
- Autorizar horas extras de trabajo, preferentemente con sueldo, por un corto tiempo.

- Actualizar actividades del calendario para colocar las nuevas funcionalidades.

- Quitar la calidad para poder cumplir con el calendario.

2.3.1.1 Línea Base (Baseline) de los Requerimientos

La *línea base* de los requerimientos es el conjunto de requerimientos funcionales y no funcionales que el equipo de desarrollo ha prometido implementar en un lanzamiento específico.

Desde una perspectiva práctica, los requerimientos en la línea base deben ser distinguidos de otros que han sido propuestos pero no aceptados. La línea base del documento SRS debe contener solamente esos requerimientos que son planeados para un lanzamiento específico. Debe ser identificado claramente como una versión de línea base para distinguirlo de una serie de versiones previas que no han sido acreditadas todavía.

2.3.1.2 Procedimientos de Administración de Requerimientos

La organización debe definir las actividades que los equipos de proyecto esperan llevar acabo para administrar sus requerimientos. Documentar estas actividades y entrenar a profesionales en la aplicación eficaz permite a los miembros de la organización llevarlas a cabo constantemente y eficazmente. Considerar abordar los siguientes tópicos:

- Herramientas, técnicas, y convenios para controlar versiones de varios documentos de requerimientos y de requerimientos individuales.

- Cómo los requerimientos son *línea base*.

- El estatus de requerimiento que se usará y quien lo puede cambiar.

- Procedimientos de seguimiento de estatus del requerimiento

- Las formas en que los nuevos requerimientos y cambios existentes son propuestos, procesados, negociados, y comunicados a todos los stakeholders afectados.

- Cómo analizar el impacto de un cambio propuesto.

- Cómo los planes y compromisos del proyecto reflejarán cambios en los requerimientos.

Se puede incluir toda ésta información en una sola descripción de proceso de administración de requerimientos. Por otra parte, se podría preferir escribir los procedimientos de control de cambios, análisis de impacto, y seguimiento de estatus independiente. Estos procedimientos deben ser aplicables a través de la organización porque representan las funciones comunes que deben ser llevadas acabo por cada equipo de proyecto.

Alguien debe poseer las actividades de administración de requerimientos, así que las descripciones de proceso también deben identificar el rol del equipo que es responsable de llevar a cabo cada tarea. El analista de requerimientos de proyecto típicamente tiene la responsabilidad principal para administrar los requerimientos. El analista establecerá mecanismos de almacenamiento de requerimientos (como una herramienta de administración de requerimientos), definirá los atributos de los requerimientos, coordinará las actualizaciones de datos de estatus y rastreabilidad de requerimientos, y generará reportes de actividad de cambio.

2.3.1.3 Control de Versión de Requerimientos

El control de versión es un aspecto esencial para administrar especificaciones de requerimientos y otros proyectos documentados. Cada versión de los documentos de requerimientos debe ser identificado excepcionalmente. Cada miembro de equipo debe poder acceder a la versión en curso de los requerimientos, y los cambios deben ser documentados claramente y transmitidos a todos los afectados. Minimizar la confusión, los conflictos, y la falta de comunicación, para permitir que solamente personas designadas actualicen los requerimientos y se aseguren de que el identificador de versión cambie siempre que un requerimiento cambie.

Cada versión que circule de los documentos de requerimientos debe incluir un historial de revisión que identifica los cambios hechos, la fecha de cada cambio, el individuo que hizo el cambio, y la razón de cada cambio. Considerar añadir un número de versión para cada etiqueta de requerimiento individual, que se puede incrementar siempre que el requerimiento es modificado, como Imprimir.ConfirmCopies - 1.

Muchas herramientas comerciales de administración de configuración están disponibles para este propósito.

2.3.1.4 Atributos de Requerimiento

Pensar en cada requerimiento como un objeto con las propiedades que lo distinguen de otros requerimientos. Además de su descripción textual, cada requerimiento funcional debe tener diversas piezas de soporte de información o atributos relacionados con él. Estos atributos establecen un contexto y un background para cada requerimiento que va mucho más allá de la descripción de funcionalidad prevista. Se pueden guardar valores de atributo en una hoja de

cálculo, una base de datos, o, más eficazmente, una herramienta de administración de requerimientos.

Un conjunto abundante de atributos es especialmente importante en los proyectos grandes y complejos. Considerar especificar los atributos como sigue para cada requerimiento:

- La fecha en que el requerimiento fue creado
- Su número de versión actual
- Autor que escribió el requerimiento
- Persona que es responsable de asegurar que el requerimiento es satisfecho
- Propietario del requerimiento o una lista de stakeholders (tomar las decisiones con respecto a los cambios propuestos)
- Status del requerimiento
- Origen o fuente del requerimiento
- La razón fundamental del requerimiento
- El subsistema (o los subsistemas) a que el requerimiento es asignado
- El número de lanzamiento del producto a la que el requerimiento es asignado
- El método de verificación para ser usado o los criterios de prueba de aprobación
- Prioridad de implementación
- La estabilidad (un indicador de cuán probable es que el requerimiento cambie en el futuro; requerimientos inestables podrían reflejar procesos de negocio o reglas de negocio mal definidos o volátiles)

Seleccionar el conjunto más pequeño de atributos que ayudarán a dirigir el proyecto eficazmente.

La línea base de requerimientos es dinámica. El conjunto de requerimientos planificados para un lanzamiento específico cambiará cuando los nuevos requerimientos son añadidos y los existentes son eliminados o postergados a un lanzamiento posterior. El equipo podría estar haciendo malabares con documentos de requerimientos distintos para el proyecto en curso y para futuros lanzamientos.

Definir y actualizar estos valores de atributo es parte del costo de administración de requerimientos, pero esa inversión puede producir una amortización importante.

2.3.1.5 Seguimiento de Estatus de los Requerimientos

Los desarrolladores de software son a veces excesivamente optimistas cuando informan sobre cuánto de una tarea ha sido completada. Estar al día con el estatus de cada requerimiento funcional durante todo el desarrollo provee una medida más exacta del progreso de proyecto.

Lista de algunos estatus de requerimientos. Como se puede apreciar en la Tabla 2-12.

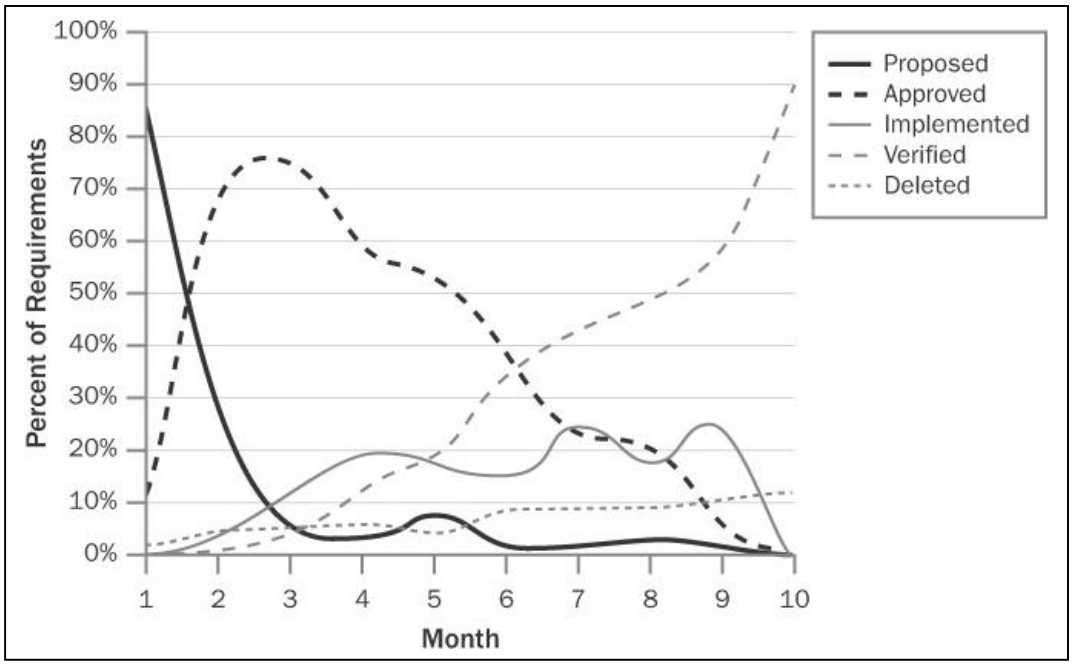
Tabla 2-12 Estatus de Requerimientos

Status	Definición
Propuesto	El requerimiento ha sido solicitado por una fuente autorizada
Aprobado	El requerimiento ha sido analizado, su impacto sobre el proyecto ha sido estimado, y ha sido destinado a la línea base para un lanzamiento específico.
Implementado	El código que implementa el requerimiento ha sido diseñado, escrito, y probado en pruebas de unidad. El requerimiento ha sido rastreado hacia el diseño pertinente y los elementos de código.
Verificado	El correcto funcionamiento del requerimiento implementado ha sido confirmado en el producto integrado. El requerimiento ha sido rastreado a las pruebas pertinentes. El requerimiento es ahora considerado completo.
Eliminado	Un requerimiento aprobado ha sido removido de la línea base. Incluir una explicación de por qué y por quién fue tomada la decisión de eliminarlo.
Rechazado	El requerimiento fue propuesto pero no es planeado para la implementación en cualquier lanzamiento próximo Incluir una explicación de por qué y por quién fue tomada la decisión de rechazarlo.

Es valioso guardar un registro de requerimientos rechazados y las razones por las cuales fueron rechazados, porque los requerimientos desestimados tienen una manera de reaparecer durante el desarrollo. El estatus Rechazar deja guardar un requerimiento propuesto disponible para la futura referencia posible sin desordenar el conjunto de requerimientos comprometidos de un lanzamiento específico.

Reatreo del estatus de los requerimientos durante todo el ciclo de desarrollo de un proyecto. Como se puede apreciar en la Figura 2-39.

Figura 2-39 Rastreo del Estatus de los Requerimientos



Fuente: Karl E. Wiegers, 2003

2.3.1.6 Medir el Esfuerzo de la Administración de Requerimientos

Si se rastrea cuánto esfuerzo se ha gastado en la administración de requerimientos, se puede avaluar si eso es muy poco, o demasiado y ajustar los procesos y la futura planeación.

Medir el esfuerzo de desarrollo y administración de proyecto real requiere un cambio de cultura, y la disciplina individual de registrar las actividades de trabajo diarias.

El seguimiento de esfuerzo también indica si el equipo está llevando a cabo las acciones previstas efectivamente para administrar sus requerimientos. Dejar de administrar requerimientos incrementa el riesgo del proyecto de cambios incontrolados y de requerimientos que son pasados por alto sin querer durante la implementación. Contar el esfuerzo dedicado a las siguientes actividades como esfuerzo de administración de requerimientos:

- Presentar cambios de requerimientos y proponer nuevos requerimientos.
- Evaluar los cambios propuestos, incluyendo el análisis de impacto llevado a cabo.
- Actualizar los documentos de requerimientos o base de datos.

- Comunicar los cambios de requerimientos a grupos y personas individuales afectadas.

- Rastrear y reportar el estatus de los requerimientos.

La administración de requerimientos ayuda a asegurar que el esfuerzo que se invierte en reunir, analizar, y documentar requerimientos no sea despilfarrado.

2.3.2 Qué sucede con los Cambios

Una organización que es seria sobre el manejo de sus proyectos de software debe asegurar que:

- Los cambios propuestos en los requerimientos son evaluados cuidadosamente antes de ser implementados.

- Los individuos apropiados toman decisiones económicas sobre los cambios solicitados.

- Los cambios aprobados se comunican a todos los participantes afectados.

- El proyecto incorpora cambios constantes en los requerimientos de una manera constante.

Los cambios en el software no son malos. Es virtualmente imposible definir todos los requerimientos de un producto sin que tengan un cambio, y el mundo cambia mientras que progresa el desarrollo.

2.3.2.1.1 Proceso de Control de Cambios

Un proceso de control de cambios permite a los líderes de proyecto tomar decisiones económicas informadas que proporcionan valor al cliente y al negocio mientras que se controlan los costos del ciclo de vida del producto. El proceso le deja seguir el estado de todos los cambios propuestos, y ayuda a asegurar que todos los cambios sugeridos no se han perdido. El proceso de cambio debe ser bien documentado, tan simple como sea posible y sobre todo efectivo.

2.3.2.1.1.1 Política Control de Cambios

La administración debe comunicar claramente una política que indique las expectativas de cómo los equipos del proyecto dirigirán los cambios propuestos a los requerimientos. Las siguientes son algunas políticas que se han encontrado:

- Todos los cambios en los requerimientos seguirán el proceso establecido. Si la petición no se somete a este proceso, no será considerado.

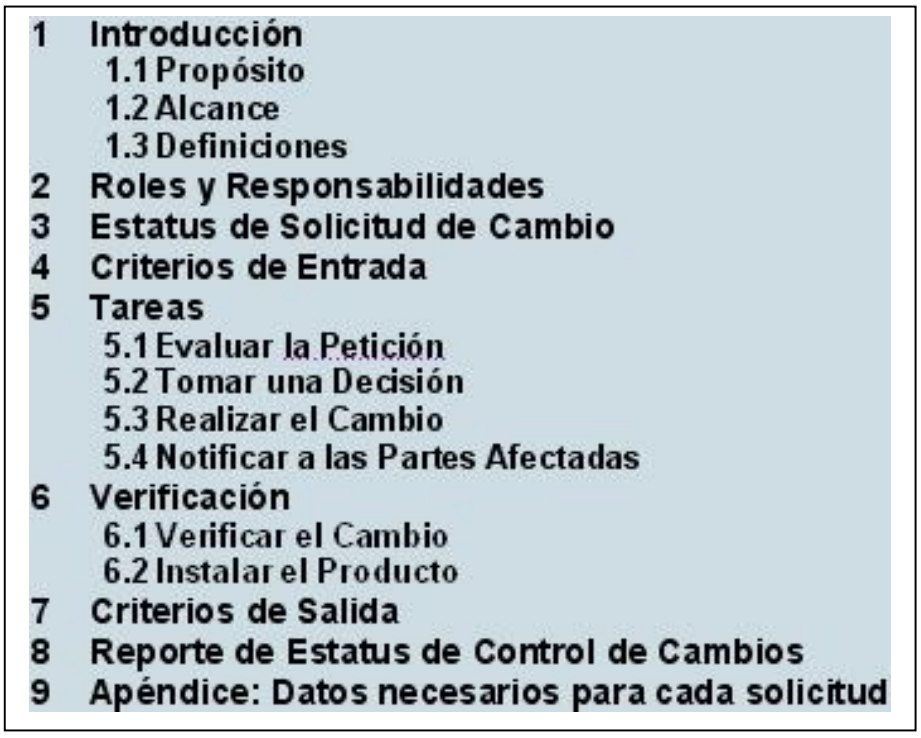
- La petición de un cambio no garantiza que será hecho. El change control borard (CCB), decidirá que cambios serán implementados.

- El contenido de la base de datos de todos los cambios será visible para todos los stakeholders del proyecto.

2.3.2.1.1.2 Descripción del Procesos Control de Cambios

La Figura 2-40 ilustra un template para la descripción del proceso de control de cambios que maneja cambios en los requerimientos y otros cambios en el proyecto.

Figura 2-40 Template para el Proceso de Control de Cambios



Fuente: Karl E. Wiegers, 2003

2.3.2.1.1.2.1 Introducción

La introducción describe el propósito de este proceso e identifica el alcance de la organización a la cual se aplica. Si este proceso cubre cambios solamente en

cierto producto de trabajo, identificarlo aquí. También se define cualquier término necesario para entender el resto del documento.

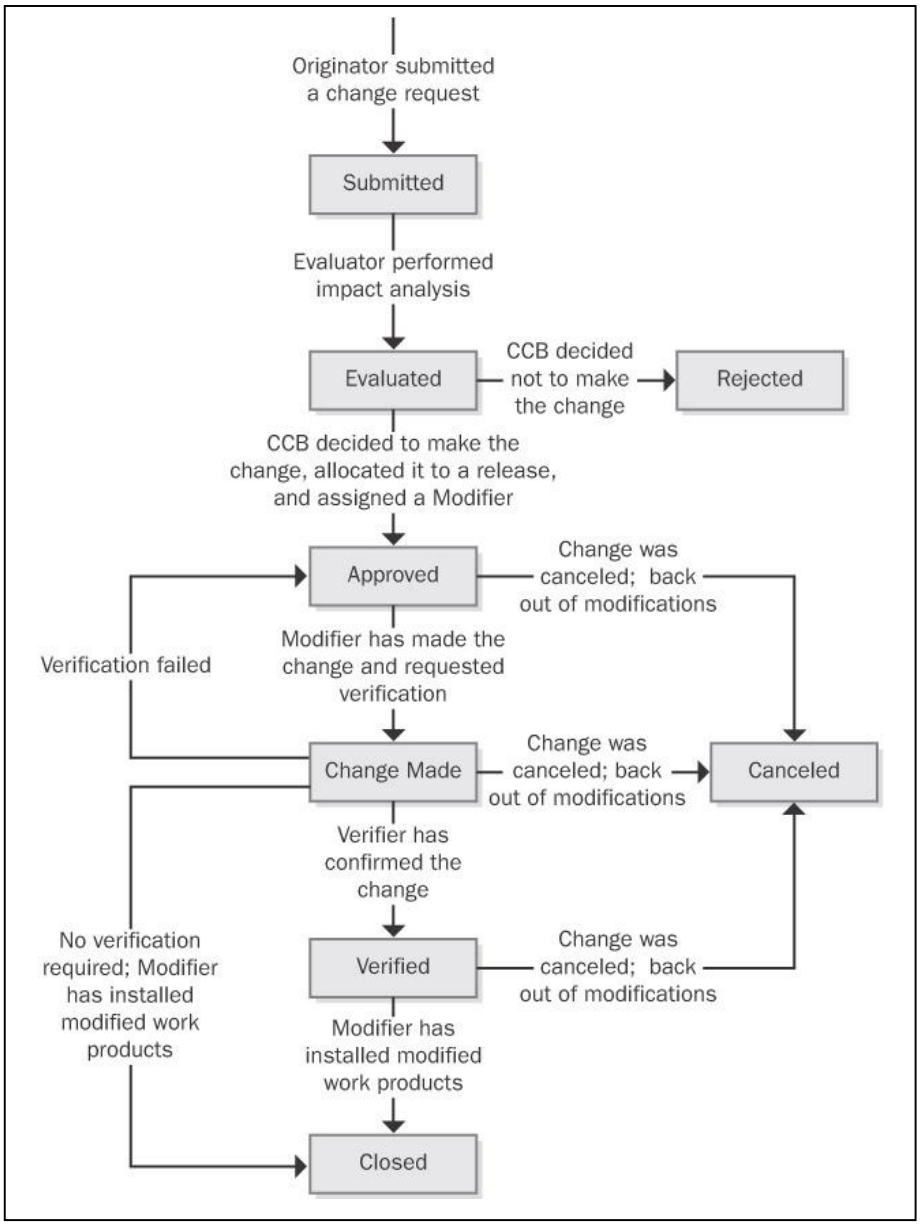
2.3.2.1.1.2.2 Roles y Responsabilidades

Lista a los miembros del equipo del proyecto- por rol, no por nombre- quienes participan en las actividades de control de cambio y describe sus responsabilidades.

2.3.2.1.1.2.3 Estatus de Solicitud de Cambio

Una petición de cambio pasa por un ciclo de vida definido, teniendo un diverso estado en cada etapa de su vida. Se puede representar el estatus del cambio por un diagrama de transición de estado como lo muestra la Figura 2-41.

Figura 2-41 Diagrama de Transición de Estado para una petición de Cambio



Fuente: Karl E. Wiegers, 2003

2.3.2.1.1.2.4 Criterios de Entrada

El criterio básico de entrada para el proceso de control de cambios es:

- Una petición válida del cambio se ha recibido a través de un canal aprobado.

2.3.2.1.1.2.5 Tareas

El siguiente paso es evaluar las solicitudes para la volatilidad técnica, costo y alineaciones con los requerimientos de negocio y restricciones de recursos.

2.3.2.1.1.2.6 Verificación

Los cambios en los requerimientos son típicamente verificados en un peer review para asegurar que las inspecciones modificadas, los casos de uso y los modelos reflejen correctamente el cambio.

2.3.2.1.1.2.7 Criterios de Salida

Todos los siguientes criterios de salida se deben satisfacer para terminar correctamente la ejecución del proceso de control de cambios:

- El estatus de la petición es Rechazado, Cancelado o Cerrado.

2.3.2.1.1.2.8 Reporte de Estatus de Control de Cambios

Identificar todos los reportes que pueden ser usados para resumir el contenido de la base de datos del control de cambios.

2.3.2.1.1.2.9 Apéndice: Datos necesarios para cada solicitud

En esta sección se almacenan todos los puntos por lo cual fue hecha la solicitud de cambio.

2.3.3 Enlaces en la Cadena de Requerimientos

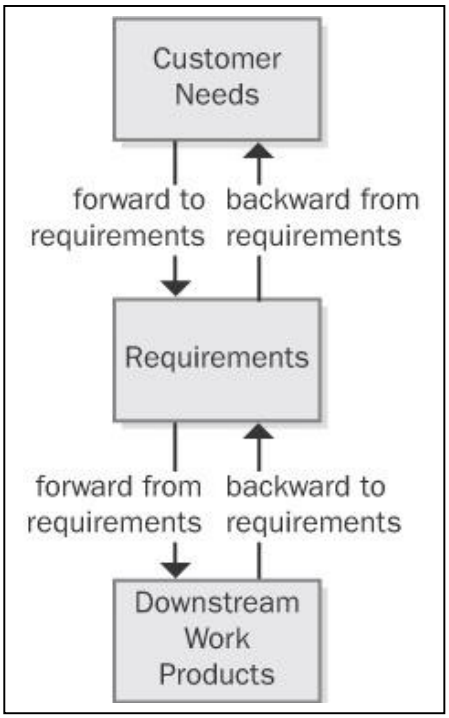
Los cambios de requerimiento simples a menudo tienen impactos trascendentales, hacen necesario que muchas partes del producto sean modificadas. Es difícil encontrar todos los componentes de sistema que puedan ser afectados por una modificación de requerimiento. El análisis de impacto de Cambio es más fácil si se tiene un guía que muestra dónde fue implementado cada requerimiento o regla de negocio en el software.

2.3.3.1 Requerimientos de Rastreo (Tracing)

Los enlaces de rastreabilidad permiten seguir la vida de un requerimiento tanto hacia adelante como hacia atrás, desde el origen hasta la implementación (Gotel y Finkelstein, 1994). Para permitir rastreabilidad, cada requerimiento debe ser excepcionalmente y persistentemente etiquetado con el propósito de que se pueda hacer referencia a éste inequívocamente durante todo el proyecto.

Existen cuatro tipos de ligas de rastreabilidad para los requerimientos. Como se puede apreciar en la Figura 2-42.

Figura 2-42 Rastreabilidad de los Requerimientos



Fuente: Karl E. Wiegers, 2003

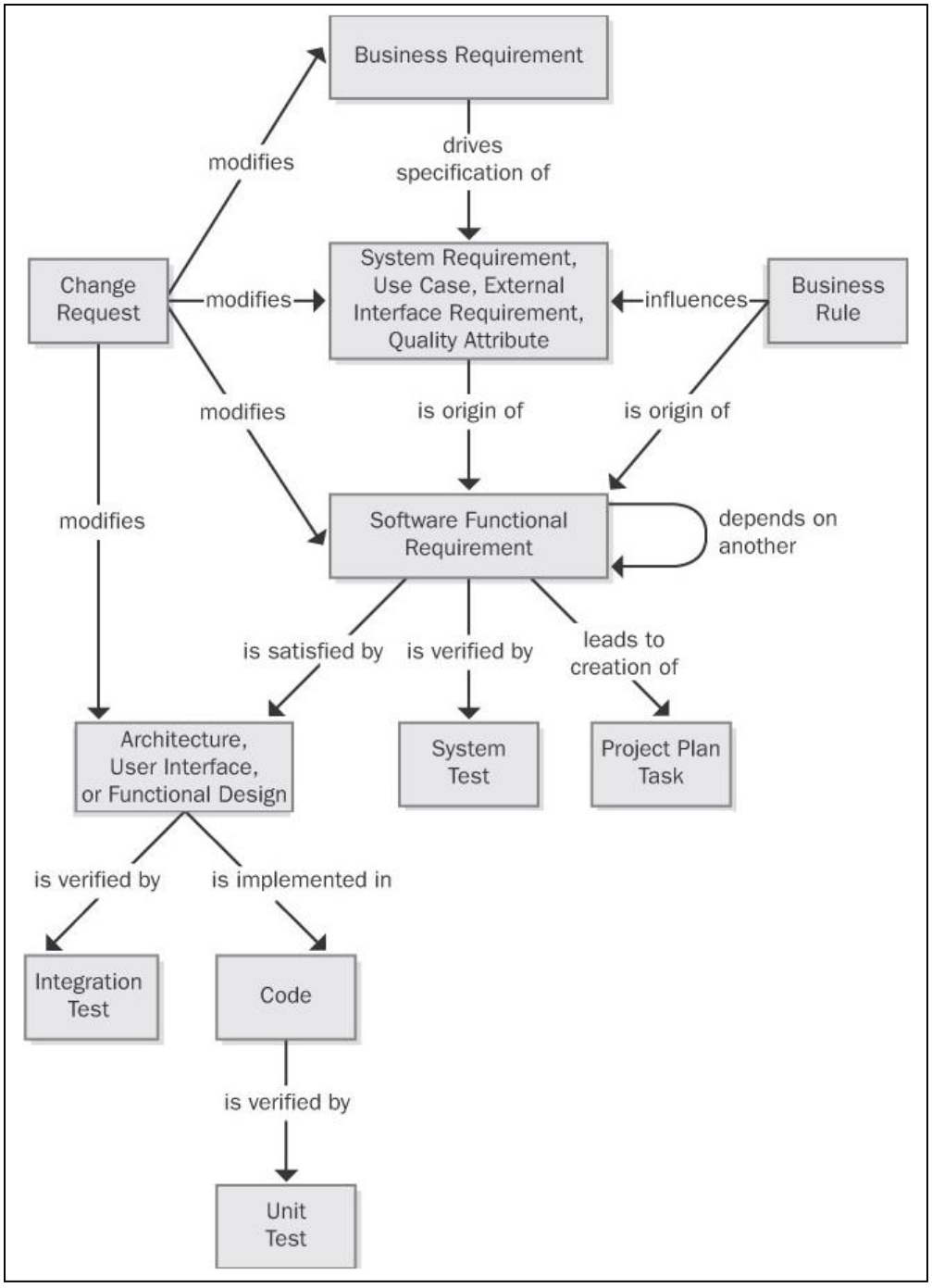
Las necesidades del cliente son rastreadas hacia adelante hacia los requerimientos, así que se puede decir qué requerimientos serían afectados si esas necesidades cambian durante o después del desarrollo. A la inversa, se puede rastrear hacia atrás desde los requerimientos hacia las necesidades del cliente para identificar el origen de cada requerimiento de software.

Cuando los requerimientos circulan en entregables durante el desarrollo, se puede rastrear hacia adelante desde los requerimientos definiendo enlaces entre requerimientos individuales y elementos de producto específicos. Éste tipo de enlace garantiza que se ha satisfecho cada requerimiento porque se sabe qué componentes aborda cada uno. El cuarto tipo de enlace rastrea elementos de producto específicos hacia atrás hacia los requerimientos con el propósito de que se sepa por qué fue creado cada ítem.

Si un tester descubre inesperada funcionalidad sin ningún requerimiento escrito correspondiente, éste código podría indicar que un desarrollador implementó un requerimiento implícito legítimo que el analista podría añadir ahora a la especificación. Por otra parte, podría ser código "huérfano". Los enlaces de rastreabilidad pueden ayudar a ordenar esta clase de situaciones y a desarrollar una fotografía más completa de cómo las piezas del sistema quedan juntas.

Existen muchas clases de relaciones de rastreabilidad directas que pueden ser definidas sobre un proyecto. Como se puede apreciar en la Figura 2-43.

Figura 2-43 Enlaces posibles de rastreabilidad de requerimientos



Fuente: Karl E. Wiegers, 2003

No se requiere definir y administrar todos estos tipos de enlaces de rastreabilidad. En muchos proyectos se puede adquirir el 80 por ciento de los beneficios de rastreabilidad deseados para quizás un 20 por ciento del esfuerzo potencial.

2.3.3.2 Motivaciones para Rastrear Requerimientos

Rastrear requerimientos es una tarea manualmente intensiva que requiere compromiso organizacional.

Beneficios potenciales de implementar requerimientos de rastreabilidad:

Certificación
Se puede usar información de rastreabilidad cuando se certifica un producto crítico de seguridad para demostrar que todos los requerimientos fueron implementados.

Análisis de impacto de Cambio
Sin la información de rastreabilidad, hay una probabilidad alta de pasar por alto un elemento de sistema que sería afectado si se añade, elimina, o modifica un requerimiento especial.

Mantenimiento
La información de rastreabilidad confiable facilita hacer los cambios correctamente y completamente durante el mantenimiento.

Seguimiento del proyecto
Si se registran datos de rastreabilidad ágilmente durante el desarrollo, se tendrá un registro exacto del estado de implementación y de la funcionalidad planeada. Los eslabones perdidos demuestran productos de trabajo que aún no han sido creados.

Reingeniería
Definir enlaces de rastreabilidad es una manera de capturar un poco de lo que se aprende a través de aplicar ingeniería inversa de un sistema existente.

Re-uso
La información de rastreabilidad facilita re-usar componentes de producto identificando paquetes de requerimientos relacionados, diseños, código, y pruebas.

Reducción de riesgo
Documentar las interconexiones de componentes reduce el riesgo si un miembro de equipo clave con conocimientos esenciales sobre el sistema deja el proyecto (Ambler, 1999).

Prueba
Cuando una prueba produce un resultado inesperado, los enlaces entre pruebas, requerimientos, y código apuntan hacia partes probables del código a revisar por un defecto.

2.3.3.3 Matriz de Rastreabilidad de Requerimientos

La forma más común para representar enlaces entre requerimientos y otros elementos del sistema es en una matriz de rastreabilidad de requerimientos (Sommerville and Sawyer, 1997). Como se puede apreciar en la Tabla 2-14.

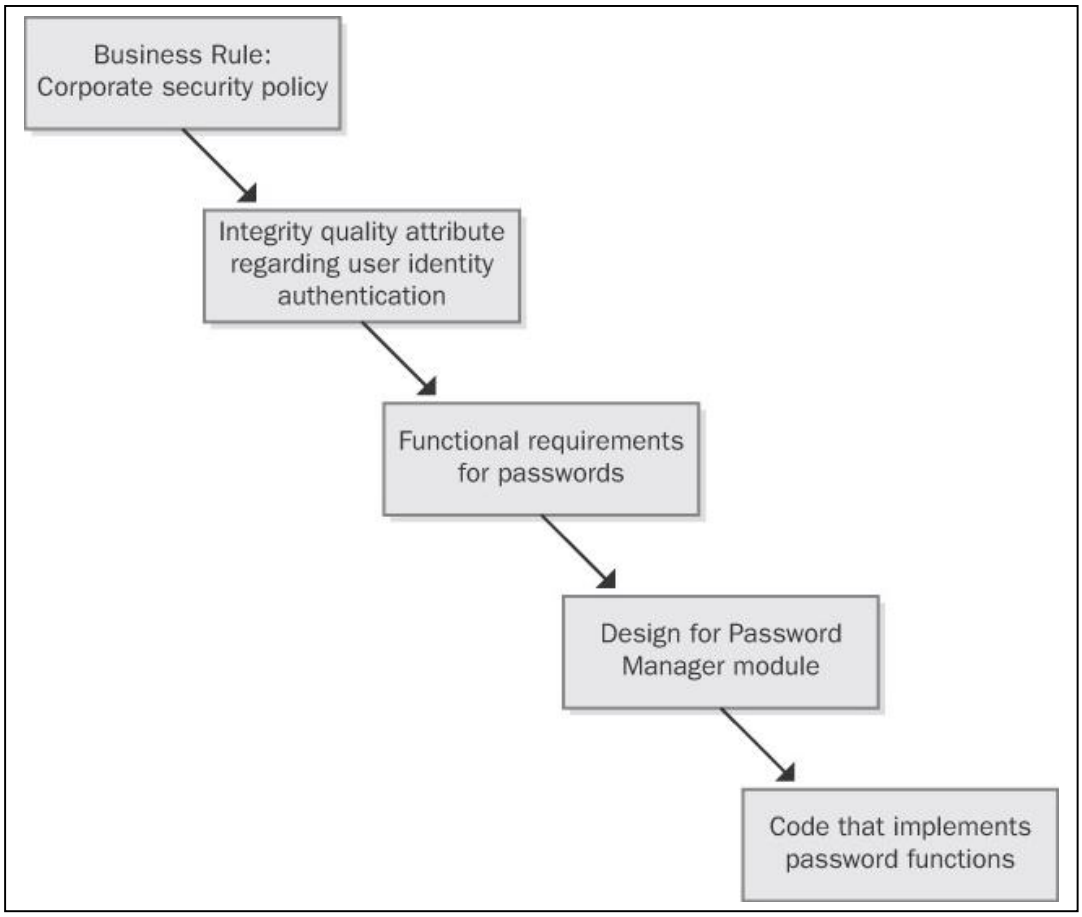
Tabla 2-13 Matriz de rastreabilidad de requerimientos

Requerimiento de Usuario	Requerimiento Funcional	Elemento de Diseño	Módulo de Código	Caso de Prueba
UC-28	catalog.query.sort	Class catalog	catalog.sort()	search.7 search.8
UC-29	catalog.query.import	Class catalog	catalog.import() catalog.validate()	search.12 search.13 search.14

Se pueden añadir más columnas para prolongar los enlaces a otros productos de trabajo, como la documentación de ayuda en línea. Incluyendo mayor detalle de rastreabilidad toma más trabajo, pero da las ubicaciones precisas de los elementos de software relacionados, que puede ahorrar el tiempo durante el análisis de impacto de cambio y mantenimiento.

Los requerimientos no-funcionales como metas de rendimiento y atributos de calidad no siempre se rastrean directamente en código. Como se puede apreciar en la Figura 2-44.

Figura 2-44 Enlace de rastreabilidad de requerimientos no funcionales



Fuente: Karl E. Wiegers, 2003

Los enlaces de rastreabilidad pueden definir relaciones entre elementos de sistema como sigue:

Uno a uno. Un elemento de diseño es implementado en un módulo de código.
Uno a muchos. Un requerimiento funcional es verificado por múltiples pruebas
Muchos a muchos. Cada caso de uso resulta en requerimientos funcionales múltiples, y ciertos requerimientos funcionales son comunes a algunos casos de uso.

2.3.3.4 Herramientas para el Rastreo de Requerimientos

En una herramienta se puede almacenar tanto requerimientos como información en una base de datos, así como definir enlaces entre varios tipos de objetos almacenados.

Algunas herramientas señalan automáticamente un enlace como sospechoso siempre que el objeto en el enlace final es modificado.

2.4 Implementar Ingeniería de Requerimientos

2.4.1 Mejorar el Proceso de Requerimientos

Poner mejores prácticas en acción es la esencia de la mejora de proceso de software. La mejora de proceso consiste en usar mayor cantidad de propuestas que funcionan bien y evitar aquellas que han dado los dolores de cabeza en el pasado.

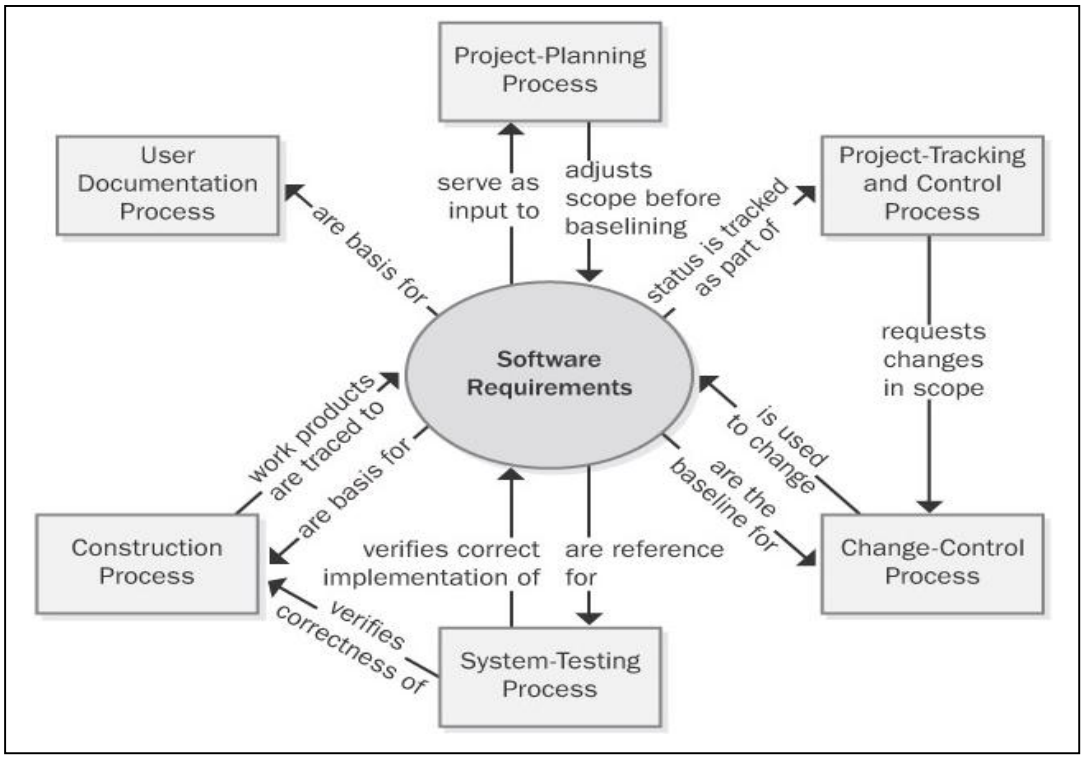
El objetivo último de la mejora de proceso de software es reducir el costo de crear y mantener actualizado el software. Hay algunas maneras de lograr esto:

- Corregir los problemas encontrados en proyectos previos o actuales que surgieron de los defectos de proceso.
- Anticipar y prevenir los problemas con los que se podría tropezar en futuros proyectos
- Adoptar prácticas que son más eficientes que las prácticas actuales que están siendo usadas.

2.4.1.1 Cómo los Requerimientos se Relacionan con otros Procesos del Proyecto

Los cambios que se hacen en el desarrollo de requerimientos y en las estrategias de administración afectan otros procesos, y viceversa. Algunas conexiones entre requerimientos y otros procesos. Como se puede apreciar en la Figura 2-45.

Figura 2-45 Relación de los requerimientos con otros procesos del proyecto



Fuente: Karl E. Wiegers, 2003

Planeación de Proyecto: los requerimientos son el fundamento del proceso de Planeación de Proyecto. Las personas de planeación seleccionan un ciclo de vida de desarrollo de software apropiado y desarrollan estimaciones de recurso y calendario basados en los requerimientos. La planeación de proyecto podría indicar que no es posible repartir el conjunto de característica deseado dentro de los límites disponibles de recursos y de tiempo.

Seguimiento y control de proyecto: el seguimiento de proyecto incluye monitorear el estatus de cada requerimiento con el propósito de que el administrador de proyecto pueda ver si la construcción y la verificación está siguiendo tal planeación. Si no, la administración podría necesitar pedir una reducción de alcance a través del proceso de Control de Cambios.

Control de Cambios: después de que un conjunto de requerimientos ha sido puesto en línea base, todos los cambios siguientes deben ser hechos a través de un proceso de Control de Cambios definido. Este proceso ayuda a asegurar:

- El impacto de un cambio propuesto sea comprendido.

- Las personas apropiadas toman las decisiones bien fundadas para aceptar los cambios.

- Todas las personas que son afectadas por un cambio estan conscientes de eso.

- Los recursos y compromisos son establecidos como necesarios.

- La documentación de los requerimientos es mantenida en curso y es exacta.

Prueba de sistema: los requerimientos de usuario y los requerimientos funcionales son contribuciones esenciales para la prueba de sistema. Si el comportamiento esperado del software bajo varias condiciones no es especificado claramente, los testers serán presionados duramente para identificar los defectos y verificar que toda funcionalidad planeada haya sido implementada como se deseó.

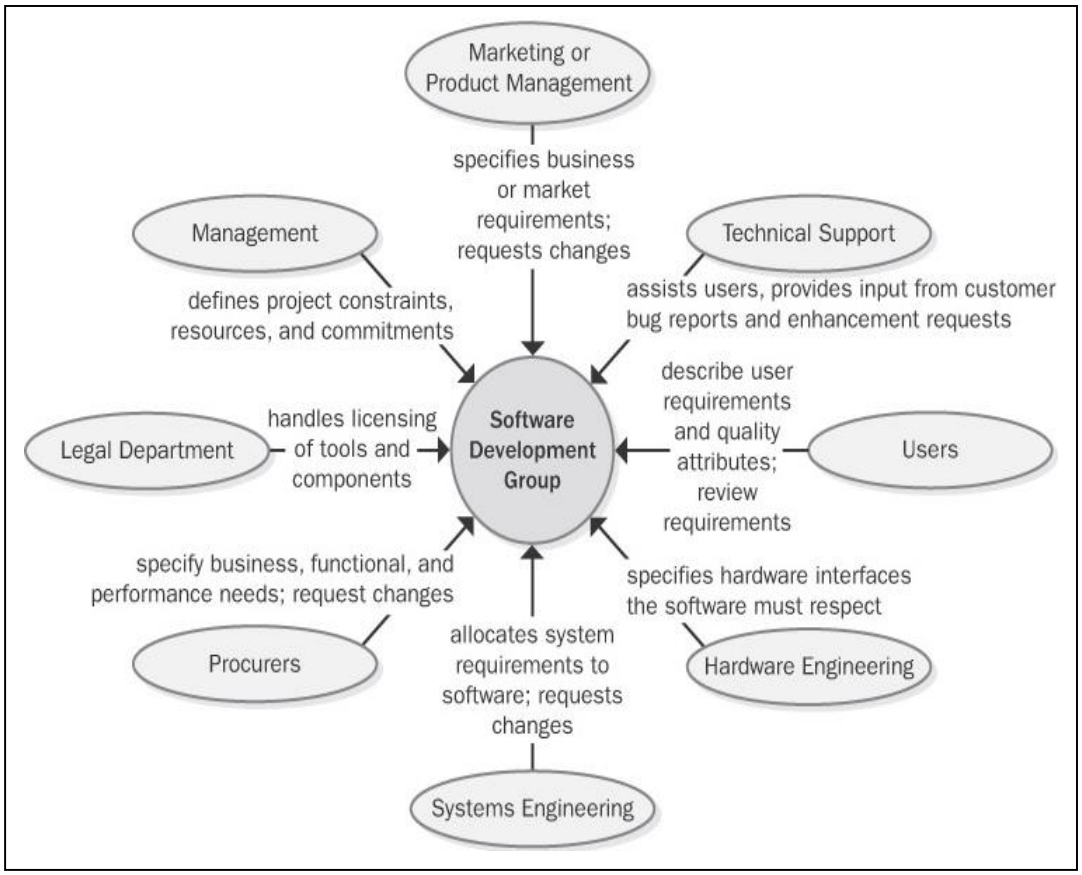
Construcción: aunque el software ejecutable es el último paso de un proyecto de software, los requerimientos forman los cimientos para el diseño y el trabajo de implementación. Usar revisiones de diseño para asegurar que los diseños abordan correctamente todos los requerimientos. La prueba de unidad puede determinar si el código satisface las especificaciones de diseño y los requerimientos pertinentes.

Documentación de usuario: los requerimientos de producto proveen la entrada para el proceso de documentación de usuario así que los requerimientos mal escritos o atrasados resultarán problemáticos para la documentación.

2.4.1.2 Requerimientos y Grupos de Stakeholders Variados

Algunos de los stakeholders de proyecto que pueden conectarse con el grupo de desarrollo de software y algunas contribuciones de los requerimientos que toman para las actividades de ingeniería de requerimientos de un proyecto. Como se puede apreciar en la Figura 2-46.

Figura 2-46 Interfaces entre el desarrollo de software y los stakeholders



Fuente: Karl E. Wiegers, 2003

Luchar por construir relaciones de colaboración entre el desarrollador y otros stakeholders de los procesos de requerimientos.

Cuando el grupo de desarrollo de software cambia sus procesos de requerimientos, las interfaces que presentan a otras comunidades de stakeholders de proyecto también cambian.

Cada cambio de proceso debe brindar la posibilidad de beneficios claros al equipo de proyecto, a la organización de desarrollo, a la compañía, al cliente, o al universo.

2.4.1.3 Fundamentos de la Mejora de Proceso de Software

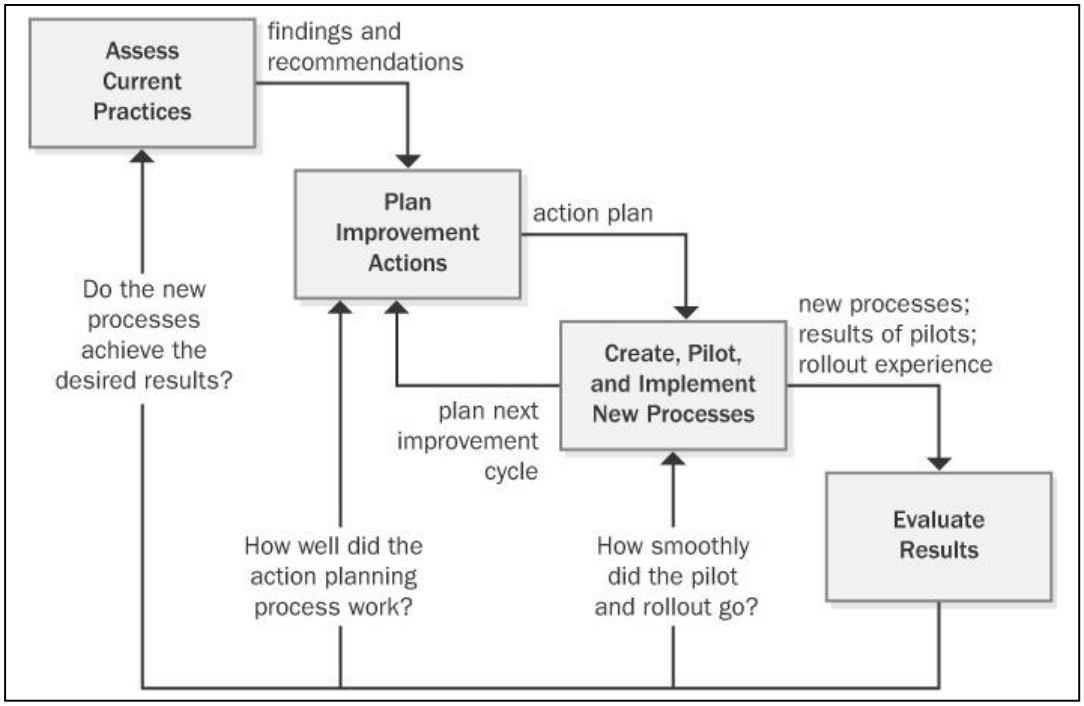
Cuando se empieza la búsqueda de requerimientos excelentes, llevar a cabo los siguientes cuatro principios de mejora de proceso de software (Wiegers, 1996a):

- La mejora de proceso debe ser evolutiva, ininterrumpida, y cíclica.
- Personas y organizaciones cambian solamente cuando tienen un incentivo por hacer.
- Los cambios de proceso deben ser metas -orientadas.
- Tratar actividades de mejora como mini-proyectos.

2.4.1.4 El Ciclo de Mejora de Proceso

Ciclo de mejora de proceso de software para ser eficaz. Como se puede apreciar en la Figura 2-47.

Figura 2-47 Ciclo de Mejora de Proceso



Fuente: Karl E. Wiegers, 2003

Evaluar Prácticas Actuales

El primer paso de cualquier mejora es evaluar las prácticas que se están usando en una organización actualmente e identificar sus fuerzas y defectos. La evaluación ofrece el fundamento para hacer las elecciones correctas sobre los cambios que se deben hacer.

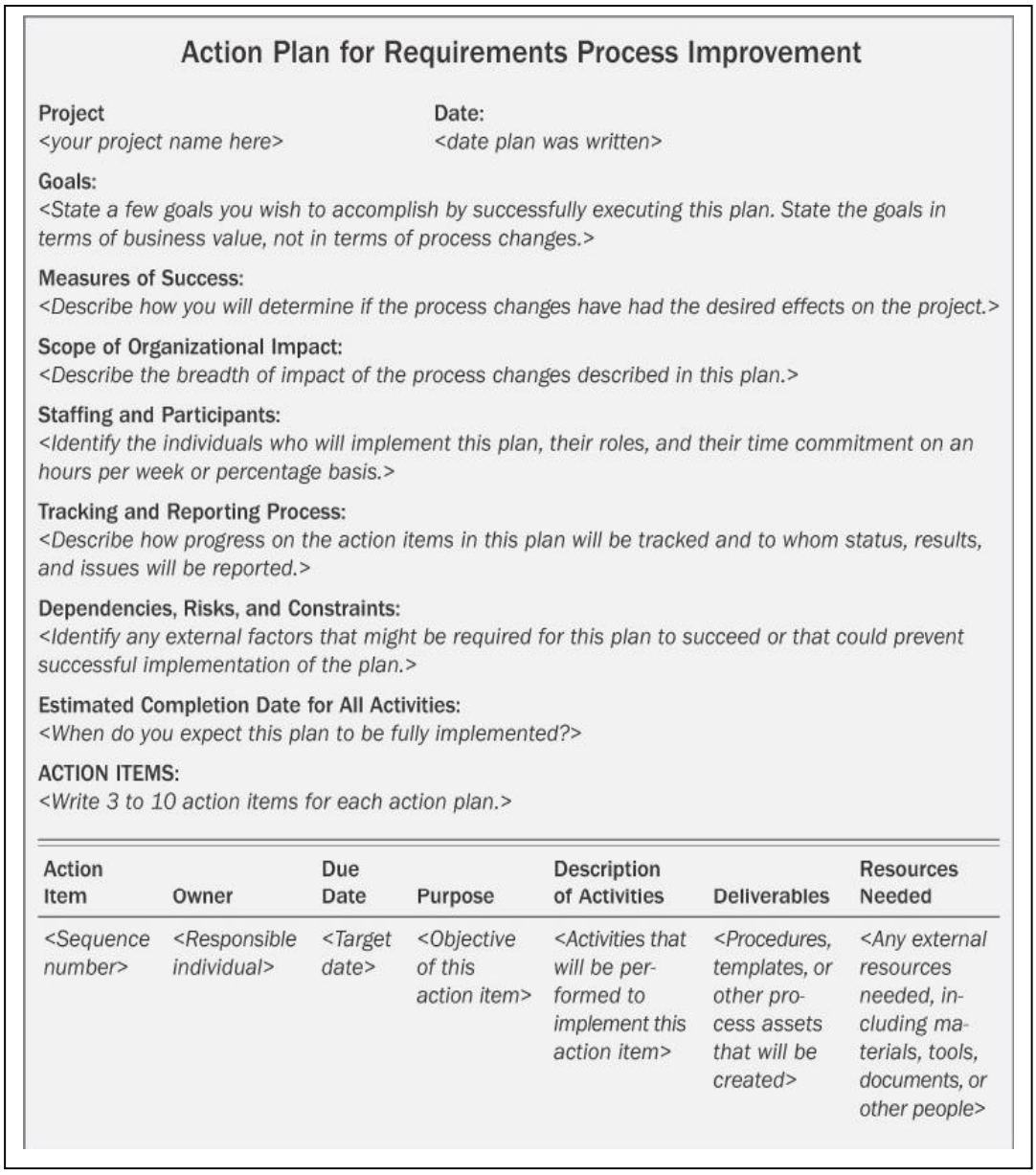
Se pueden evaluar los procesos actuales de muchas formas. Los cuestionarios estructurados proveen un enfoque más sistemático, que puede revelar la perspicacia sobre los procesos actuales a un bajo costo. Las entrevistas y las discusiones con miembros de equipo proveen un conocimiento más exacto y exhaustivo.

Un enfoque más minucioso es tener un consultor externo para valorar los procesos de software actuales. Las evaluaciones de procesos más exhaustivas están basadas en un framework de mejora de procesos establecido, como el Modelo de Madurez de Capacidad para Software (Capability Maturity Model for Software, SW-CMM) desarrollado por el Instituto de Ingeniería de Software (Software Engineering Institute). Los evaluadores externos examinan el desarrollo de software y los procesos de administración, liberando un statement formal que incluye una lista de conclusiones y recomendaciones para abordar las oportunidades de mejora.

Plan de Acciones de Mejora
Un plan estratégico describe la iniciativa de mejora de proceso de software global de la organización. Los planes de acción tácticos se centran en áreas de mejora específicos. Cada plan de acción debe fijar los objetivos de las actividades de mejora, los participantes, y los ítems de acción individuales que deben ser terminados para implementar el plan.

Template de Plan de Acción para la Mejora de Procesos de Software. Como se puede apreciar en la Figura 2-48.

Figura 2-48 Template del Plan de acción para la Mejora de Proceso de Software



Action Plan for Requirements Process Improvement

Project
<your project name here>

Date:
<date plan was written>

Goals:
<State a few goals you wish to accomplish by successfully executing this plan. State the goals in terms of business value, not in terms of process changes.>

Measures of Success:
<Describe how you will determine if the process changes have had the desired effects on the project.>

Scope of Organizational Impact:
<Describe the breadth of impact of the process changes described in this plan.>

Staffing and Participants:
<Identify the individuals who will implement this plan, their roles, and their time commitment on an hours per week or percentage basis.>

Tracking and Reporting Process:
<Describe how progress on the action items in this plan will be tracked and to whom status, results, and issues will be reported.>

Dependencies, Risks, and Constraints:
<Identify any external factors that might be required for this plan to succeed or that could prevent successful implementation of the plan.>

Estimated Completion Date for All Activities:
<When do you expect this plan to be fully implemented?>

ACTION ITEMS:
<Write 3 to 10 action items for each action plan.>

Action Item	Owner	Due Date	Purpose	Description of Activities	Deliverables	Resources Needed
<Sequence number>	*<Responsible individual>*	*<Target date>*	*<Objective of this action item>*	*<Activities that will be performed to implement this action item>*	*<Procedures, templates, or other process assets that will be created>*	*<Any external resources needed, including materials, tools, documents, or other people>*

Fuente: Karl E. Wiegers, 2003

Crear, Probar, e Implementar Nuevos Procesos

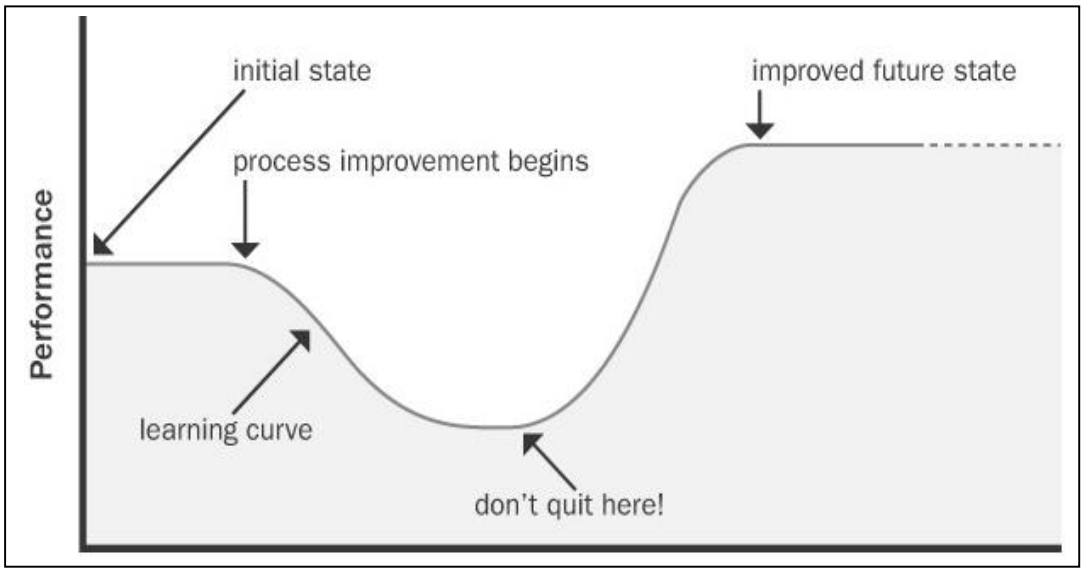
Implementar un plan de acción representa nuevos procesos en desarrollo que se creé que producirán mejores resultados que esas maneras actuales de hacer el trabajo.

Evaluar Resultados
El paso final de un ciclo de mejora de proceso es evaluar las actividades llevadas a cabo y los resultados logrados.

Un paso crítico es valorar si los procesos recién implementados están produciendo los resultados deseados.

Aceptar la realidad de la curva de aprendizaje - eso es, la poca productividad que tiene lugar cuando los profesionales toman tiempo para asimilar las nuevas maneras de trabajar. Como se puede apreciar en la Figura 2-49.

Figura 2-49 La curva de aprendizaje es una parte inevitable de mejora de proceso



Fuente: Karl E. Wiegers, 2003

Esta pizca de productividad a corto plazo - llamada a veces el "Valle de la desesperación" - es parte de la inversión que la organización hace en la mejora de proceso.

2.4.1.5 Elementos Valiosos Activos del Proceso de Ingeniería de Requerimientos

Los proyectos de gran rendimiento tienen procesos eficaces para todos los componentes de ingeniería requerimientos: licitación, análisis, especificación, validación, y administración. Para facilitar el rendimiento de estos procesos, cada organización necesita una colección de elementos valiosos de proceso (Wiegers, 1998). Estos elementos valiosos de proceso ayudarán aquellos involucrados en el proyecto a comprender los pasos que deben seguir y crear los productos de

trabajo que son esperados. Los elementos valiosos de proceso incluyen clases de documentos. Como se puede apreciar en la Tabla 2-14.

Tabla 2-14 Documentos de Proceso Activo

Estatus	Definición
Checklist	Una lista que enumera las actividades, entregables, u otros ítems para ser observados o verificados. Los checklist ayudan a asegurar que personas ocupadas no pasen por alto los detalles importantes.
Ejemplo	Acumular buenos ejemplos cuando los equipos de proyecto los crean.
Plan	Una idea general de cómo se realiza el objetivo y qué necesito para lograrlo.
Política	Un principio guiando que fija una expectativa de administración de los comportamientos, las acciones, y entregables. Los procesos deben permitir la satisfacción de las políticas.
Procedimiento	Una descripción paso a paso de la secuencia de las tareas que logra una actividad. Describir las tareas llevadas a cabo e identificar los roles de proyecto que las efectúan.
Descripción de Proceso	Una definición documentada de un conjunto de actividades llevadas a cabo para algún propósito.
Template	Un patrón para ser usado como guía para producir un producto de trabajo completo. Un template bien- estructurado provee muchos "Huecos" para capturar y organizar la información. El texto guiado arraigado en el template ayudará al autor de documento a usarlo eficazmente.

Algunos elementos valiosos de proceso para la ingeniería de requerimientos. Como se puede apreciar en la Tabla 2-15.

Activos de proceso dominantes para el desarrollo de los requisitos y la gerencia de los requisitos **Assets**

Tabla 2-15 Activos de Proceso Importantes

Elementos Activos del Proceso de Desarrollo de Requerimientos	Elementos Activos del Proceso de Administración de Requerimientos
Proceso de Desarrollo de Requerimientos	Proceso de Administración de Requerimientos
Procedimiento de Asignación de Requerimientos	Proceso de Control de Cambios
Procedimiento para Priorizar Requerimientos	Procedimiento de Seguimiento de Estatus de los Requerimientos
Template de Visión y Alcance	Procedimiento de Rastreabilidad de

	Requerimientos
Template de Caso de Uso	Estatutos de Control de Cambios (Change Control Board Charter)
Template de Especificación de Requerimientos de Software	Template y Checklist de Análisis del Impacto de Cambios a los Requerimientos
Checklists de Defectos de SRS y Casos de Uso	

2.4.1.5.1 Elementos Valiosos del Proceso de Desarrollo de Requerimientos

Proceso de Desarrollo de Requerimientos. Este proceso describe cómo identificar stakeholders, clases de usuario, y champions del producto en su dominio. El proceso debe dirigir cómo planear las actividades de licitación, incluyendo seleccionar técnicas de licitación apropiadas, identificar a participantes, y estimar el esfuerzo y el tiempo de calendario requerido para la licitación. El proceso también describe varios documentos de requerimientos y modelos que el proyecto espera crear y apunta al lector hacia templates apropiados. El proceso de desarrollo de requerimientos también debe identificar los pasos que el proyecto debe llevar a cabo para el análisis y la validación de requerimientos.

Procedimiento de Asignación de Requerimientos. Asignar requerimientos de producto de alto nivel a subsistemas específicos, incluyendo personas, es necesario cuándo sistemas de desarrollo incluyen tanto componentes de hardware como de software o productos complejos que contienen subsistemas de software múltiples (Nelson, 1990). Este procedimiento describe cómo realizar estas asignaciones para asegurarse de que la funcionalidad esta asignada a los componentes apropiados. También describe cómo los requerimientos serán rastreados.

Procedimiento para Priorizar Requerimientos. Para reducir alcance prudentemente o complacer los requerimientos adicionales dentro de un calendario establecido, se tiene que saber qué capacidad del sistema planeado tiene la prioridad más baja y más baja.

Template de Visión y Alcance. El documento de visión y alcance es una descripción concisa y de alto nivel de los requerimientos de negocio del nuevo producto. Suministra una referencia para tomar las decisiones acerca de las prioridades y los cambios de los requerimientos.

Template de Casos de Uso. El template de casos de uso provee un formato estándar para describir las tareas que los usuarios tienen que llevar a cabo con un sistema de software. Una definición de caso de uso incluye una descripción breve de la tarea, las descripciones de las conductas alternativas, las excepciones conocidas que deben ser manejadas, e información adicional sobre la tarea.

Template de Especificación de Requerimientos de Software. El template SRS provee una forma estructurada y consecuente de organizar los requerimientos funcionales y no-funcionales del producto. Considerar asumir más de un template para complacer los diferentes tipos o tamaños de proyectos que la organización emprende. Esto puede reducir la frustración que aparece cuando un procedimiento no es apropiado para el proyecto.

Checklists de Defectos del SRS y Casos de Uso. La inspección formal de documentos de requerimientos es una técnica de calidad de software fuerte. Un checklists de defectos de inspección identifica muchos de los errores comúnmente encontrados en los documentos de requerimientos. Usar un checklists durante la etapa previa a la inspección para enfocar la atención en áreas problemáticas comunes.

2.4.1.5.2 Elementos Valiosos del Proceso de Administración de Requerimientos

Proceso de Administración de Requerimientos. Este proceso describe las acciones que un equipo de proyecto toma para distribuir los cambios, distinguir versiones de la documentación de requerimientos, rastrear e informar sobre el estatus de los requerimientos, y acumular la información de rastreabilidad. El proceso debe listar los atributos para incluirlos en cada requerimiento, tal como la prioridad, la estabilidad pronosticada, y el número de lanzamiento planeado. También debe describir los pasos requeridos para aprobar el SRS y establecer una línea base de requerimientos

Proceso de Control de Cambios. Un proceso práctico de control de cambios puede reducir el caos derivado por interminables, e incontrolados cambios de requerimientos. El proceso de control de cambios define la manera en que un nuevo requerimiento o una modificación para un requerimiento existente es propuesto, comunicado, valorado, y resuelto. Una herramienta de seguimiento del problema facilita el control de cambios, pero es importante recordar que una herramienta no es un sustituto para un proceso.

Procedimiento de Seguimiento de Estatus de los Requerimientos. La administración de requerimientos incluye monitorear e informar sobre el estatus de cada requerimiento funcional. Se necesitará usar una base de datos o una herramienta de administración de requerimientos comercial para estar al día con el estatus de los muchos requerimientos en un sistema grande. Este procedimiento también describe los informes que se puede generar para ver el estatus de los requerimientos coleccionados en cualquier momento.

Estatutos des Control de Cambios. El change control board (CCB) es el cuerpo de stakeholders que determina que cambios en los requerimientos aprobar, cuáles

rechazar, y en qué lanzamiento del producto serán incluidos los cambios aprobados.

Template y Checklist de Análisis de Impacto de Cambios a los Requerimientos. Estimar el costo y otros impactos de un cambio de requerimiento propuesto es un paso clave para determinar si aprobar o no el cambio. El análisis de impacto ayuda al CCB a tomar decisiones inteligentes.

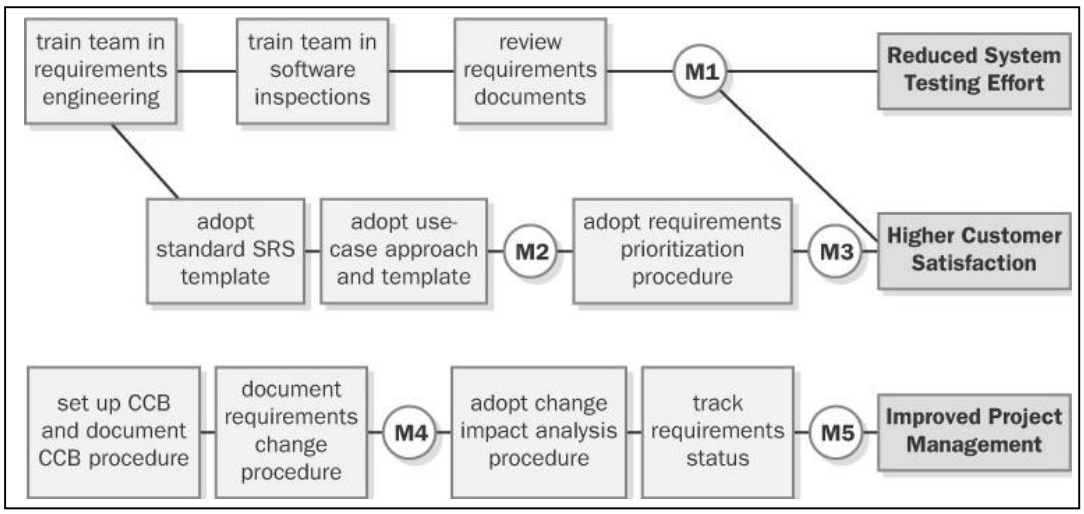
Procedimiento de Rastreabilidad de Requerimientos. La matriz de rastreabilidad de requerimientos lista todos los requerimientos funcionales, los componentes de diseño y los módulos de código que abordan cada requerimiento, y los casos de pruebas que verifican su implementación correcta. La matriz de rastreabilidad también debe identificar el requerimiento de sistema padre, el caso de uso, regla de negocio, u otra fuente de la que cada requerimiento funcional fue obtenido. Este procedimiento describe quién proveer los datos de rastreabilidad, quién los colecciona y administra, y dónde son guardados.

2.4.1.6 Guía de Mejora de Proceso de Requerimientos

La guía de mejora de proceso ordena en serie acciones de mejora de una manera que producirá los beneficios más grandes con la inversión más baja.

Las metas de negocio deseadas son mostradas en cajas en el lado derecho de la Figura 2-50, y las mejores actividades de mejora son mostradas en las otras cajas. Los círculos indican acontecimientos intermedios a lo largo de la trayectoria para conseguir las metas de negocio. Como se puede apreciar en la Figura 2-50.

Figura 2-50 Ejemplo de una Guía de Mejora de Procesos de Requerimientos



Fuente: Karl E. Wiegers, 2003

2.4.2 Requerimientos de Software y Administración de Riesgos

Un riesgo es una condición que puede causar cierta perdida o amenaza al éxito del proyecto. Estos problemas potenciales pudieran tener un impacto adverso en el costo del proyecto, calendario, éxito técnico, calidad del producto o eficacia del equipo. La administración de riesgos es un proceso de identificación, evaluación y control de riesgos antes de que dañen el proyecto. Si algún inconveniente ha sucedido en el proyecto este ya es un problema y no un riesgo.

Porque nadie puede predecir el futuro con certeza, la administración de riesgos es una manera de reducir al mínimo la probabilidad o el impacto de problemas potenciales. Los requerimientos desempeñan un papel central en los proyectos de software, los prudentes administradores de proyectos identificarán riesgos relacionados con los requerimientos tempranamente y los controlarán agresivamente. Algunos riesgos típicos en los requerimientos son, entender mal los requerimientos, implicación inadecuada del usuario, alcance y objetivos inciertos o cambios en estos y continuamente cambios en los requerimientos. Los administradores de proyectos pueden controlar riesgos en los requerimientos solamente con la colaboración del cliente o sus representantes.

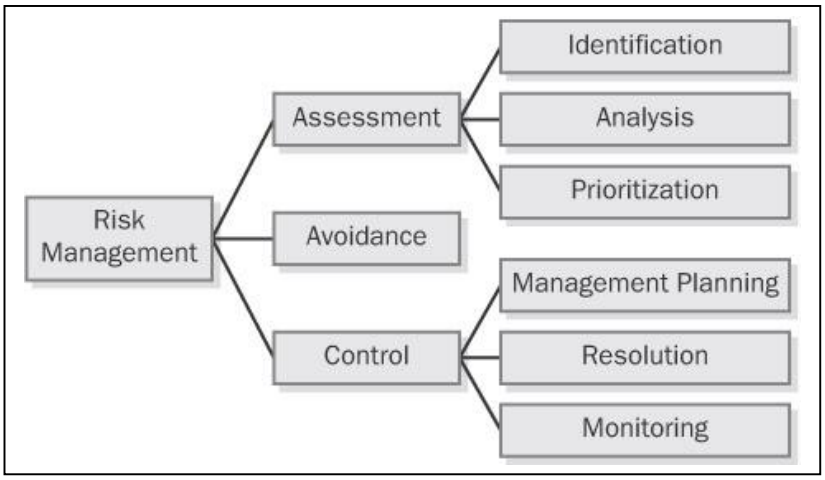
2.4.2.1 Fundamentos para la Administración de Riesgos de Software

Los proyectos hacen frente a muchos tipos de riesgos relacionados con el alcance y los requerimientos del proyecto. Las dependencias a una unidad externa, tal como un subcontratista u otro proyecto que proporciona componentes que se reutilizarán, es una fuente común de riesgos. Los riesgos tecnológicos en la aplicación del dominio, la falta de conocimiento, y los cambios declarados por los gobiernos son una fuente de riesgos comunes.

2.4.2.1.1 Elementos para la Administración de Riesgos

La administración: de riesgos es la aplicación de herramientas y procedimientos para contener riesgos dentro de los límites aceptables. La administración de riesgos provee un estándar para identificar y documentar factores de riesgo, para evaluar su potencial severidad, y proponer estrategias para mitigar estos riesgos (Williams, Walter, and Dorofee 1997). La administración de riesgos incluye las actividades mostradas en la Figura 2-51.

Figura 2-51 Elementos de Administración de Riesgos



Fuente: Karl E. Wiegers, 2003

Evaluación de riesgos es el proceso de identificar áreas del proyecto potenciales de riesgos. Facilitar la identificación de riesgos con listas de comunes factores de riesgos para proyectos de software. Durante el análisis de riesgos, se debe examinar las consecuencias potenciales de riesgos específicos en el proyecto. La prioridad de riesgos ayuda a centrarse en los riesgos más severos.

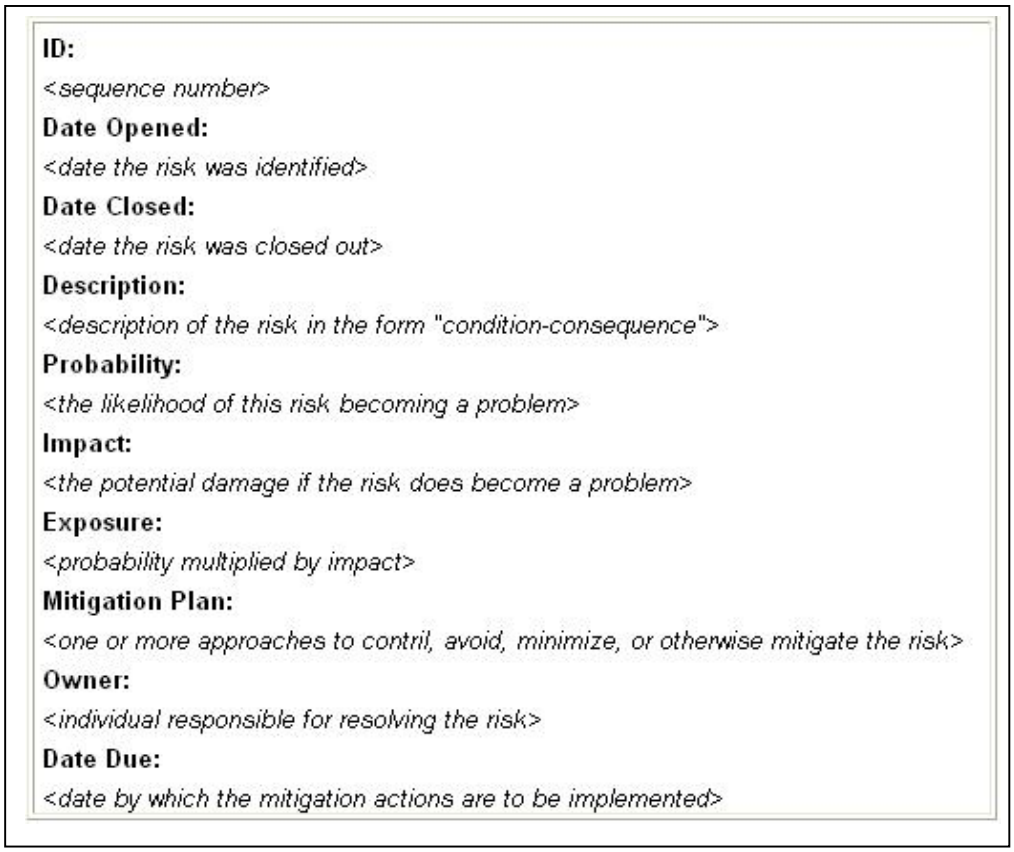
Evasión de riesgos es una manera de no hacer las cosas aventuradamente. Se pueden evitar riesgos no emprendiendo ciertas partes del proyecto en las cuales sus características serán difíciles poner en práctica.

Las actividades de control de riesgos son para manejar los riesgos que se identificaron en etapas posteriores. La administración de riesgos en la planeación produce un plan, ocuparse de cada riesgo significativo, incluyendo mitigación, planes de contingencia, etc. Los riesgos no se controlarán, así que la resolución de riesgo implica el ejecutar los planes de contingencia para atenuar cada riesgo. Finalmente, el monitoreo de riesgos implica el seguimiento de estos para saber el estado del proyecto e implicaciones que se han tenido por los riesgos.

2.4.2.1.2 Documentando Riesgos del Proyecto

No basta con simplemente reconocer los riesgos del proyecto, se necesita manejarlos de una manera que le deje comunicar todos los riesgos y el estado del proyecto durante toda su duración. La Figura 2-52muestra un template de comunicación para documentar un statement de riesgos individuales.

Figura 2-52 Template de Riesgos del Proyecto



Fuente: Karl E. Wiegers, 2003

2.4.2.1.3 Planeación para la Administración de Riesgos

Una lista de riesgos no es igual a un plan de administración de riesgos. Para proyectos pequeños, se pueden incluir el plan para administrar riesgos, en el plan de administración de proyectos de software. En proyectos largos, se debe escribir por separado el plan de administración de riesgos en la que se explique y proponga como identificar, documentar, evaluar y dar seguimiento a los riesgos.