Cimat©

Guía Metodológica para la Ingeniería de Requerimientos

Versión 1.0
02/05/2005

Adaptado de:

Software Requirements, Karl E. Wiegers, Microsoft Press

Control De Versiones

Nombre del Archivo	Versión	Fecha	Autor	Comentarios
GM_Ing_Requerimientos	1.0	02/05/2005	JFCR, APV e IJS	Versión Inicial del Documento

Índice del Documento

Índice de Figuras

Índice de Tablas

1 Introducción

1.1 Propósito

Guiar al alumno en los procesos que se usan para obtener y analizar los requerimientos, escribir y validar especificaciones de requerimientos, y administrar los requerimientos durante todo el ciclo de desarrollo del producto.

1.2 Alcance

Este documento a plica a todos los proyectos generados dentro de la Licenciatura de Sistemas Computacionales plan 04 y la Licenciatura en Ingeniería en Computación plan 03 a partir de la liberación de la línea base.

1.3 Acrónimos

Acrónimo	Descripción
CCB	Change Control Board (Tablero de Control de Cambios)
DER	Diagramas Entidad-Relación
DFD	Diagrama de Flujo de Datos
DTE	Diagramas de Transición de Estado
QFD	Quality Function Deployment (Despliegue de Función de Calidad)
SRS	Especificación de Requerimientos de Software (Software Requirement Specification)
UML	Unified Modeling Language (Lenguaje de Modelado Unificado)

1.4 Referencias

Núm.	Descripción
1	Loucupoulos 1989
2	IEEE Standard Glossary of Software Engineering Terminology (1990)
3	Wiegers Karl E. (2003). Software Requirements (Segunda Edición). Redmond, Washington: Microsoft Press

1.5 Definiciones

A continuación se listarán los conceptos clave de la ingeniería de requerimientos y que ayudarán a entender de mejor el contenido del documento.

Palabra	Definición
Administración de Requerimientos	Es la que establece y mantiene actualizado un acuerdo con el cliente de los requerimientos para el proyecto de software
Administración de Riesgos	Es una manera de reducir al mínimo la probabilidad o el impacto de problemas potenciales.
Analista de Requerimientos	Es el individuo que tiene la responsabilidad principal de recolectar, analizar, documentar, y validar las necesidades de los stakeholders de proyecto.
Casos de Uso	Los casos de uso describen secuencias de interacciones entre un sistema y actores externos.
Checklist	Es una lista que numera actividades u otros puntos que serán observados y verificados.
Cliente	Es un individuo u organización de quien deriva directa o indirectamente un beneficio de un producto.
Desarrollo de Requerimientos	Es el proceso de identificar clases de usuario, extraer necesidades, analizar la información recibida documentándola para lograr el producto que el cliente necesita.
Diagrama de Flujo de Datos (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.
Diagrama de Transición de Estado (DTE)	Es aquel que pproporciona una representación consistente, completa y no ambigua de una maquina de estado-finito.
Diagrama Entidad-Relación (DER)	La relación que existe entre los datos del sistema. 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.
Diagramas de Clase	Son una forma gráfica de representar las clases identificadas durante el análisis orientado a objetos y la relación entre ellas.
Especificación de Requerimientos de Software (SRS)	Describe el comportamiento esperado del sistema de software.
Fuente de los Requerimientos	De donde provienen los requerimientos: Entrevistas Documentos. Observación y Autoaprendizaje.
Ingeniería de Requerimientos	Proceso sistemático de desarrollo de requerimientos a través de un proceso iterativo del análisis de un problema, documentando resultados de observaciones y verificando la exactitud de la comprensión ganada.
Mapas de Dialogo	Es aquel que representa un diseño de interfaz utilizado en un alto nivel de abstracción.
Matriz de Requerimientos	Es la forma más común para representar enlaces entre requerimientos y otros elementos del sistema.

Proceso de Control de Cambios	Es aquel que 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.
Prototipos	Un prototipo de software es solo una porción o modelo del sistema real de un nuevo producto.
Reglas de Negocio	Son aquellas reglas que incluyen políticas corporativas, regulaciones de gobierno, estándares industriales y practicas contables.
Requerimiento	Una oración que identifica una habilidad, característica física, o factor de calidad y que adicionalmente liga un producto o proceso necesario con la solución para la cual es perseguida.
Requerimientos de negocio/Necesidad de Cliente/Necesidades de Mercado	Representan los objetivos de alto nivel de la organización o del cliente que requiere el sistema.
Requerimientos de Software/Diseño/Funcional/Sistema	Especifica la funcionalidad del software que los desarrolladores deben de construir en el producto para posibilitar a los usuarios a completar sus tareas y que a su vez satisfagan los requerimientos de negocio.
Requerimientos de Usuario/Cliente	Describen los objetivos del usuario o tareas que los usuarios deben de ser capaces de ejecutar con el producto.
Requerimientos Funcionales y No Funcionales	Funcionales: Define lo que el sistema debe ser capaz de realizar. Transformaciones que el sistema realiza sobre las entradas para producir las salidas. No Funcionales: Características que pueden limitar el sistema, pero no describe los servicios que el sistema provee.
Stakeholders	Son todos aquellos individuos que tienen relación con el proyecto. Desde clientes que solicitan, pagan por, seleccionan, especifican, usan, o reciben una salida generada por el producto de software, incluyendo incluyen analista de requerimiento, desarrolladores, testers, escritores de documentación, administradores de proyecto, staff de soporte, y staff de mercadotecnia.
Tablas y Árboles de Decisión	Son aquellos que listan varios valores para todos los factores que influyen en el comportamiento e indica las acciones previstas del sistema en respuesta a cada combinación de factores.
Visión y Alcance del Proyecto	La visión describe que es el producto y que eventualmente puede llegar a ser. El alcance del proyecto identifica que porción de la última visión de largo plazo del producto va a tratar el actual proyecto.

2 Requerimientos de Software

2.1 Requerimientos de Software - Qué, Por qué y Quién

2.1.1 El Requerimiento de Software Esencial

2.1.1.1 Definición de Requerimientos de Software

Un problema con la industria del software es la falta de definiciones para los términos que utilizamos al describir aspectos de nuestro trabajo. Diferentes observadores podrían describir la misma sentencia como un requerimiento de usuario, un requerimiento de software, un requerimiento funcional, un requerimiento de sistema, un requerimiento técnico, un requerimiento de negocio, o un requerimiento de producto. La definición de requerimientos de un cliente podría sonar como un concepto de producto de alto nivel para el desarrollador.

Un concepto clave es que los requerimientos deben ser documentados.

2.1.1.1.1 Algunas Interpretaciones de Requerimiento

El consultor Brian Lawrence sugiere que un requerimiento es “cualquier cosa que maneja opciones de diseño” (Lawrence 1997).

La IEEE Standard Glossary of Software Engineering Terminology (1990), define un requerimiento como:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento impuesto formalmente.

3. Una representación documentada de una condición o capacidad.

Wiegers piensa que un requerimiento es una propiedad que un producto debe tener para proveer valor a un stakeholder. La siguiente definición reconoce la diversidad de tipos de requerimientos (Sommerville and Sawyer 1997):

Los requerimientos son… una especificación de lo que debe estar implementado. Son descripciones de cómo el sistema debe comportarse, o una propiedad o atributo del sistema. Puede ser una restricción en el proceso de desarrollo del sistema.

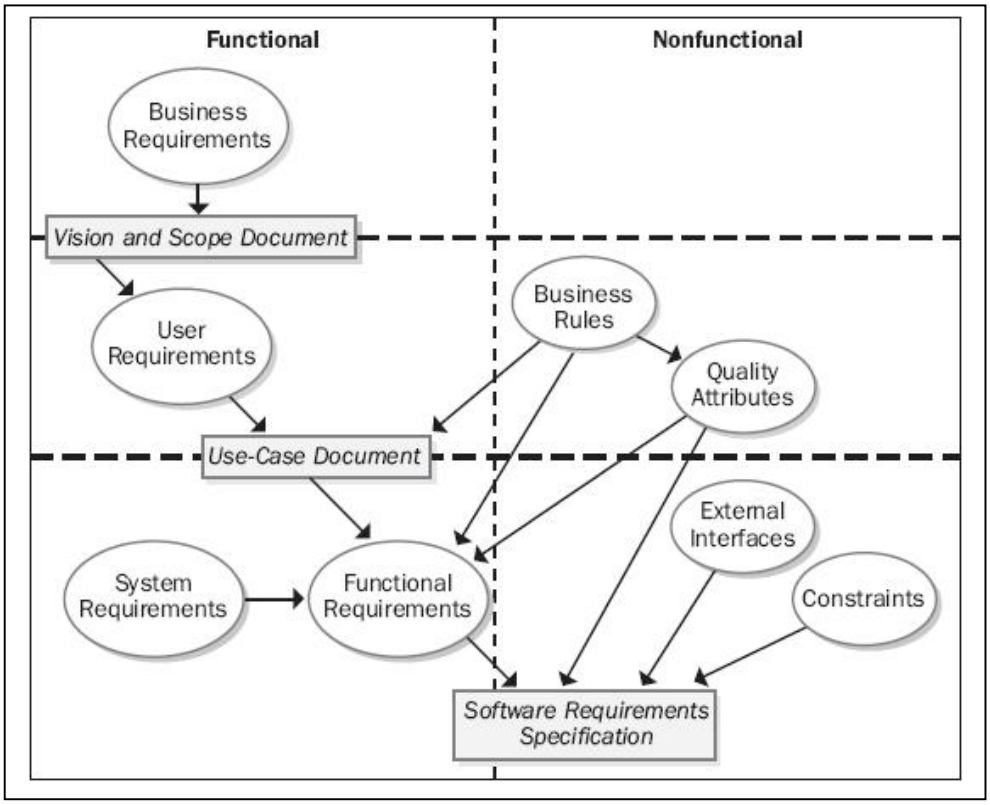
2.1.1.1.2 Niveles de Requerimientos

Los Requerimientos de Software incluyen tres niveles distintos:

1. Requerimientos de Negocio
2. Requerimientos de Usuario
3. Requerimientos Funcionales

Como se puede apreciar en la Figura 2-1.

Figura 2-1 Relaciones de algunos tipos de información de requerimientos



Fuente: Karl E. Wiegers, 2003

La Figura 2-1 ilustra los tres niveles de requerimientos. Los óvalos representan los tipos de requerimientos y los rectángulos indican contenedores (documentos, diagramas o bases de datos) en los cuales se almacena la información.

Requerimientos de Negocio: representan los objetivos de alto nivel de la organización o del cliente que requiere el sistema. Los requerimientos de negocio típicamente provienen del patrocinador principal del proyecto, el cliente, el administrador de los usuarios actual o el departamento de mercadotecnia. Los

requerimientos de negocio describen por qué la organización esta implementando el sistema, es decir, los objetivos que la organización espera alcanzar.

El documento donde se registran los Requerimientos de Negocio es conocido como:

- Visión y alcance
- Proyect charter
- Documento de requerimientos de mercado

Requerimientos de usuario: describen los objetivos del usuario o tareas que los usuarios deben de ser capaces de ejecutar con el producto. Las formas para representar requerimientos de usuario incluyen:

- Casos de uso
- Descripciones de escenario
- Tablas de evento-respuesta

Los requerimientos de usuario describen por lo tanto qué es lo que el usuario es capaz de hacer con el sistema.

Un ejemplo de un caso de uso es “Hacer una reservación” en una línea aérea, en una renta de autos o un hotel a través de un Web site.

Requerimientos funcionales: especifica la funcionalidad del software que los desarrolladores deben de construir en el producto para posibilitar a los usuarios a completar sus tareas y que a su vez satisfagan los requerimientos de negocio. Algunas veces estos requerimientos son llamados de comportamiento, estos se describen con la tradicional sentencia “deberá”.

Un ejemplo de un requerimiento funcional es “El sistema deberá enviar vía e-mail la confirmación de la reservación al usuario”

Reglas de negocio: incluyen políticas corporativas, regulaciones de gobierno, estándares industriales, prácticas contables y algoritmos computacionales. Estas reglas no son en sí requerimientos de software porque estas existen fuera de los límites de cualquier especificación del sistema de software. Sin embargo, restringen quien puede ejecutar ciertos casos de uso.

Algunas veces las reglas de negocio son el origen de atributos específicos de calidad que son implementados en la funcionalidad.

Los requerimientos funcionales son documentados en el SRS (Software Requirement Specification) que describe tanto como sea necesario el comportamiento esperado del sistema de software.

Además de los requerimientos funcionales, el SRS contiene requerimientos no-funcionales.

Un requerimiento de negocio podría leerse “El producto permitirá que los usuarios corrijan eficientemente los errores de ortografía en un documento”. Un requerimiento de usuario podría incluir tareas –casos de uso- como “Encontrar los errores de ortografía”. El corrector ortográfico tiene muchos requerimientos funcionales, operaciones como encontrar y seleccionar una palabra mal escrita, exhibir un cuadro de dialogo con las sustituciones sugeridas y remplazar la palabra mal escrita con la palabra corregida.

Los requerimientos de usuario permiten al analista obtener las partes de la funcionalidad que permitirá a los usuarios llevar acabo sus tareas con el producto. Los desarrolladores usan los requerimientos funcionales y no-funcionales para diseñar las soluciones que implementan la funcionalidad necesaria y consiguen la calidad específica y los objetivos de rendimiento, dentro de los límites que las restricciones los obligan.

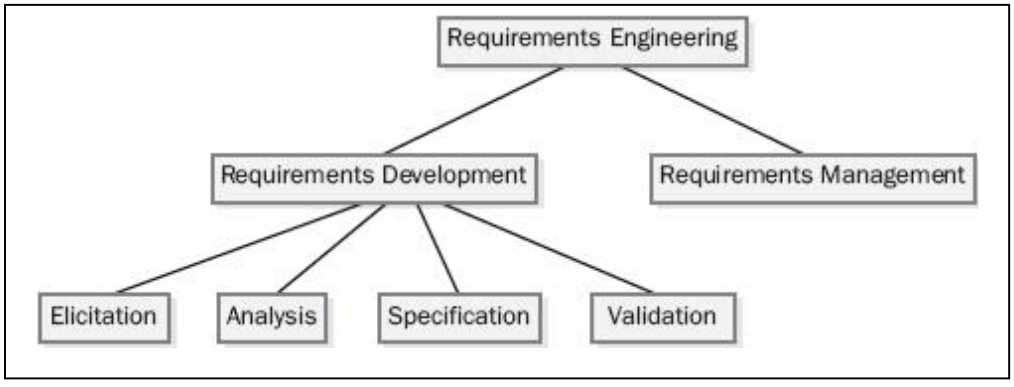
2.1.1.1.3 Lo que los Requerimientos No Son

Las especificaciones de requerimientos no incluyen los detalles de diseño o implementación, información de la planeación del proyecto o información de pruebas (Leffingwell and Widrig, 2000).

2.1.1.2 Desarrollo y Administración de los Requerimientos

Wiegers divide a la Ingeniería de Requerimientos en Desarrollo de Requerimientos y Administración de Requerimientos. Como se puede apreciar en la Figura 2-2.

Figura 2-2 Subdivisiones de la Ingeniería de Requerimientos



Fuente: Karl E. Wiegers, 2003

2.1.1.2.1 Desarrollo de Requerimiento

Podemos subdividir Desarrollo de Requerimientos en Licitación, Análisis, Especificación y Validación. (Abran and Moore, 2001) estas sub disciplinas abarcan todas las actividades relacionadas con la recopilación, evaluación y documentación de los requerimientos de software., incluyendo lo siguiente:

- Identificar las clases de usuario del producto esperado.
- Extraer las necesidades de los individuos que representan cada clase de usuario.
- Comprender las tareas y metas del usuario y los objetivos de negocio con los que esas tareas se alinean.
- Analizar la información recibida de los usuarios para distinguir sus objetivos de tarea de requerimientos funcionales, requerimientos no-funcionales, reglas de negocio, etc.
- Destinar partes de los requerimientos de alto nivel a definir componentes de software en la arquitectura sistema.
- Comprender la importancia de los atributos de calidad.
- Negociar las prioridades de implementación.
- Traducir las necesidades de usuario escritas dentro de las especificaciones y modelos de requerimientos.
- Examinar los requerimientos documentados para asegurar el conocimiento común de los requerimientos presentados por los usuarios y corregir cualquier problema antes de que el grupo de desarrolladores los acepte.

La iteración es una clave para el éxito del desarrollo de los requerimientos.

2.1.1.2.2 Administración de Requerimientos

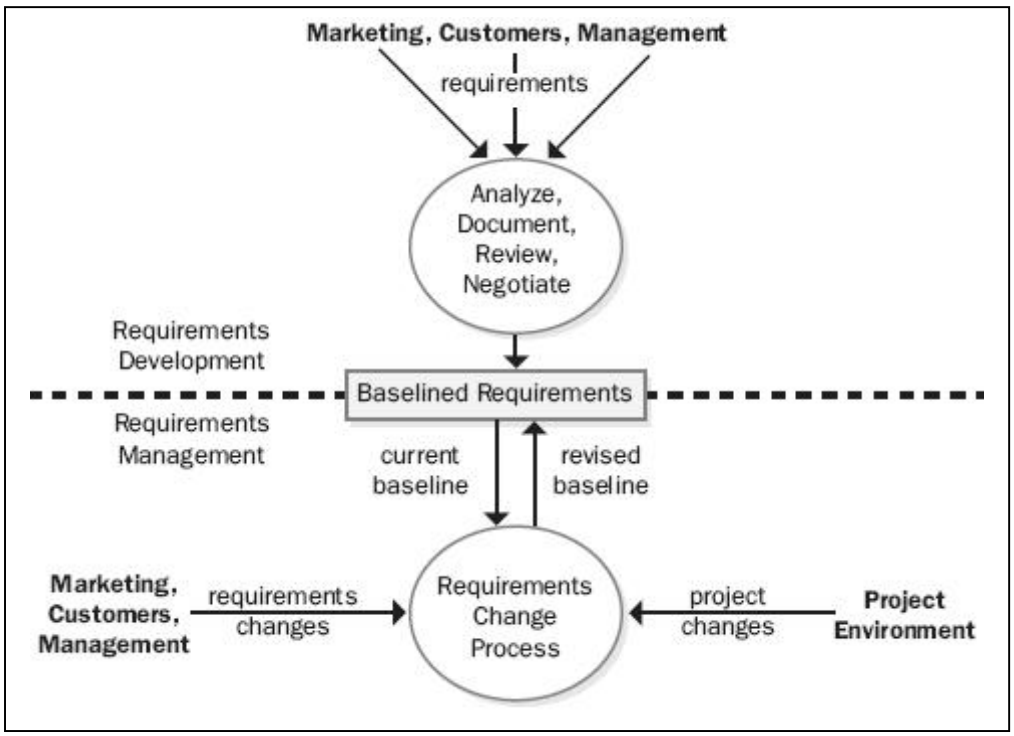
La Administración de Requerimientos implica “establecer y mantener actualizado un acuerdo con el cliente de los requerimientos para el proyecto de software” (Paulk et al, 1995). La Administración de Requerimientos incluye las siguientes actividades:

- Definir el punto de partida de los requerimientos.
- Revisar y evaluar el impacto de cada requerimiento cambiado antes de aprobarlo.

- Seguir cada requerimiento en su diseño, código fuente y pruebas.
- Agrupar los requerimientos según rendimiento y actividad de cambio durante todo el proyecto.

Como se puede apreciar en la Figura 2-3.

Figura 2-3 El límite entre el Desarrollo y la Administración de Requerimientos



Fuente: Karl E. Wiegers, 2003

La Figura 2-3 provee otra vista de la diferencia entre Desarrollo de Requerimientos y Administración de Requerimientos.

2.1.1.3 Cada Proyecto tiene Requerimientos

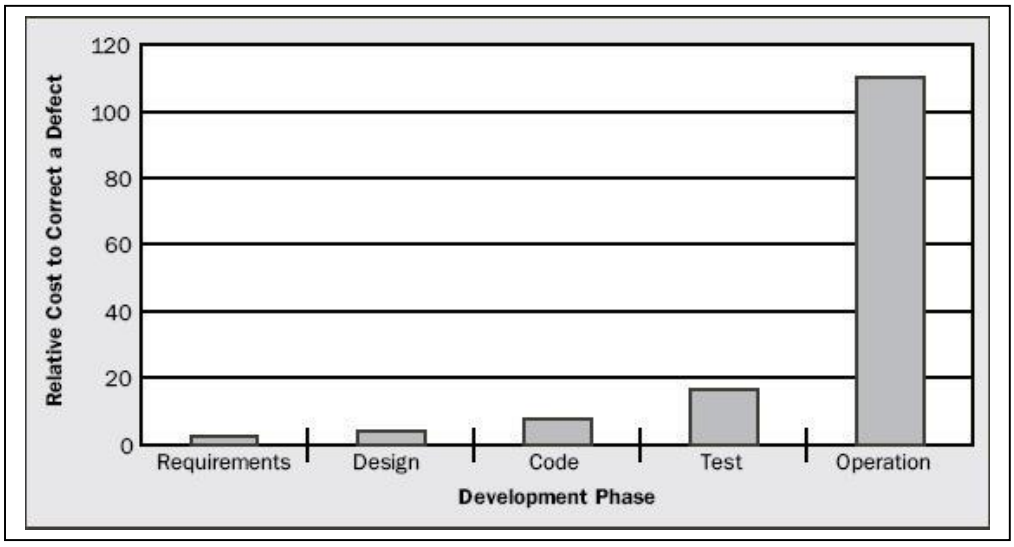
La parte difícil de construir un sistema de software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos, incluyendo todas las interfaces para las personas, las computadoras y otros sistemas de software. Ninguna otra parte del trabajo paraliza los resultados del sistema si se hace mal y ninguna otra parte es más difícil de rectificar más tarde.

2.1.1.4 Cuando los Requerimientos Malos Pasan a otras Personas

Cuesta mas corregir un defecto que se encontró tarde en el proyecto que arreglarlo poco después de su creación (Grady, 1999). Prevenir los errores de requerimientos y atraparlos tempranamente reduce la reelaboración. Una consecuencia importante de hacer mal los requerimientos es la reelaboración.

Como se puede apreciar en la Figura 2-4.

Figura 2-4 Costo Relativo por Corregir un Defecto



Fuente: Karl E. Wiegers, 2003

2.1.1.4.1 Participación del Usuario Insuficiente

Los clientes no comprenden por qué es tan esencial trabajar duro en la obtención de requerimientos y garantizar su calidad. Los desarrolladores no pueden enfatizar la participación del usuario, porque trabajar con los usuarios no es tan divertido como escribir código o porque piensan que ya saben qué necesitan los usuarios.

2.1.1.4.2 Requerimientos de Usuario Progresivos

Cuando los requerimientos se desenvuelven y crecen durante el desarrollo, a menudo los proyectos superan sus calendarios y presupuestos planeados, por lo que continuas modificaciones en los requerimientos hacen peor el problema.

Para evitar problemas se necesita un statement claro de los objetivos del negocio, la visión estratégica, el alcance, las limitaciones, los criterios de éxito y el uso del producto.

2.1.1.4.3 Requerimientos Ambiguos

La ambigüedad es la gran pesadilla de las especificaciones de requerimientos (Lawrence, 1996). Un síntoma de la ambigüedad es que el lector puede interpretar un requerimiento de muchas formas. Otra señal es que múltiples lectores de un requerimiento llegan a diferentes acuerdos del que éste representa.

Una manera de descubrir la ambigüedad es tener personas que representen perspectivas diferentes al inspeccionar los requerimientos. Escribir los casos de prueba contra los requerimientos y desarrollar prototipos son otra forma de descubrir la ambigüedad.

2.1.1.4.4 Requerimientos "Gold Plating"

Gold Plating tiene lugar cuando un desarrollador añade funcionalidad que no estaba en la especificación de requerimientos pero que el desarrollador cree "les encantará a los usuarios". Los desarrolladores deben luchar por conseguir la madurez y sencillez, no para ir más allá de lo que el cliente pide sin la aprobación del mismo.

2.1.1.4.5 Especificación Mínima

Algunas veces el personal o administradores de mercadotecnia crean especificaciones limitadas, quizá solo un concepto del producto esbozado en una servilleta. Esperan que los desarrolladores entren en detalle cuando el proyecto avanza. En la mayoría de los casos, frustran a los desarrolladores (quienes podrían estar operando bajo suposiciones incorrectas y con dirección limitada) y decepcionan a los clientes (quienes no consiguen el producto que previeron).

2.1.1.4.6 Pasar Por Alto Clases de Usuarios

Si no se identifican las clases de usuario importantes para el producto tempranamente, algunas necesidades de los usuarios no serán cubiertas. Después de identificar todas las clases de usuario, asegurar de que cada uno tenga voz.

2.1.1.4.7 Planeación Incorrecta

La mala estimación ocurre por los cambios frecuentes en los requerimientos, comunicación ineficiente con los usuarios, especificación pobre de los requerimientos y análisis insuficiente de los requerimientos.

2.1.1.5 Beneficios de un Proceso de Requerimientos de Buena Calidad

Las organizaciones que implementan efectivamente los procesos de ingeniería de requerimientos pueden cosechar múltiples beneficios. Una gran recompensa viene de reducir la reelaboración innecesaria durante el desarrollo del proyecto. Los posibles beneficios que se podrían disfrutar en cuanto al ahorro del tiempo y dinero, se cuantifican los siguientes:

- Menos defectos en los requerimientos
- Reelaboración del desarrollo reducida
- Disminución de propiedades innecesarias
- Rápido desarrollo
- Disminución de la falta de comunicación
- Caos de proyecto reducido
- Estimaciones más aproximadas
- Satisfacción del cliente y el equipo

2.1.1.6 Características de Requerimientos Excelentes

2.1.1.6.1 Características del Statement de Requerimientos

Completo. Cada requerimiento debe describir completamente la funcionalidad para ser alcanzado. Debe contener toda información necesaria por el desarrollador para diseñar e implementar esa parte de funcionalidad.

Correcto. Cada requerimiento debe describir la funcionalidad con exactitud para ser construido. La referencia para la corrección es el origen del requerimiento, como un usuario actual o un requerimiento de sistema de alto-nivel. Un requerimiento de software que está en contradicción con su requerimiento de sistema padre no es correcto.

Viable. Debe ser posible implementar cada requerimiento dentro de las capacidades y limitaciones conocidas del sistema y su ambiente operativo. Evitar especificar requerimientos inalcanzables, tener un desarrollador trabajando con el de mercadotecnia o el analista de requerimientos durante todo el proceso de licitación.

Necesario. Cada requerimiento debe documentar una capacidad que los clientes realmente necesitan.

Prioritario. Asignar una prioridad de implementación para cada requerimiento funcional. Si todos los requerimientos son considerados equitativamente importantes, es difícil que el administrador de proyecto responda a recortes de presupuesto, perdida de personal, o nuevos requerimientos adicionados durante el desarrollo.

No Ambiguo. Todos los lectores de un statement de requerimientos deben llegar a una sola interpretación consistente, pero el lenguaje natural es muy propenso a la ambigüedad. Escribir requerimientos en un lenguaje simple, conciso y sencillo apropiado para el usuario. Definir todos los términos especializados y términos que puedan confundir a lectores en un glosario.

Verificable. Es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas. Los requerimientos que están incompletos, inconsistentes, o ambiguos son imposibles de demostrar (Drabick, 1999).

2.1.1.6.2 Características de la Especificación de Requerimientos

Un conjunto de requerimientos que son coleccionados en una especificación deben presentar las siguientes características:

Completo. Ningún requerimiento o información necesaria debe estar ausente.

Consistente. Los requerimientos consistentes no están en conflicto con otros requerimientos del mismo tipo o con el negocio de más alto-nivel, el sistema, o los requerimientos de usuario. Los desacuerdos entre requerimientos deben ser resueltos antes de que el desarrollo pueda seguir. Registrar el creador de cada requerimiento permitirá saber a quién hablar si se descubren conflictos.

Modificable. Se debe poder cambiar el SRS cuando sea necesario y mantener una historia de los cambios hechos en cada requerimiento. Cada requerimiento debe aparecer solo una vez en el SRS. Una tabla de contenido o un índice hará el SRS más fácil de modificar. Guardar requerimientos en una base de datos o en una herramienta de administración de requerimientos comercial los hace objetos reutilizables.

Rastreable. Un requerimiento rastreable puede estar ligado desde atrás a su origen y hacia adelante a elementos de diseño y código fuente que lo implementan y pruebas que verifican la implementación correcta.

2.1.2 Requerimientos desde la Perspectiva del Cliente

Los requerimientos resultan un problema de confusión sobre los diferentes niveles de requerimientos: negocio, usuario y funcionales.

La sociedad cliente-desarrollador es crítica para el éxito de un proyecto de software. Para esto se propone una Carta de Requerimientos de Derechos para los Clientes de Software, así como una Carta de Requerimientos de Responsabilidades para los Clientes de software.

2.1.2.1 ¿Quién es el cliente?

En el más amplio sentido de la palabra, un cliente es un individuo u organización de quien deriva directa o indirectamente un beneficio de un producto. Clientes de software incluye stakeholders quienes solicitan, pagan por, seleccionan, especifican, usan, o reciben una salida generada por el producto de software. Otros stakeholders incluyen analista de requerimiento, desarrolladores, testers, escritores de documentación, administradores de proyecto, staff de soporte, y staff de mercadotecnia.

Los requerimientos de negocio describen los objetivos de negocio que el cliente, compañía, y otros stakeholders quieren alcanzar. Los requerimientos de negocio establecen una línea en el marco de trabajo para el resto del proyecto, sin embargo no proporciona suficiente detalle para decirle al desarrollador que es lo que se quiere construir.

El siguiente nivel de requerimientos –requerimientos de usuario- debería venir de gente quien utiliza actualmente el producto, directa o indirectamente. Estos usuarios (algunas veces llamados usuarios finales) por lo tanto constituyen otro tipo de usuario. Los usuarios pueden describir las tareas que ellos necesitan para realizarlas con el producto y las características de calidad que ellos esperan que el producto vaya a exhibir.

Para los sistemas de información, los contratos de desarrollo o las aplicaciones personalizadas de desarrollo, los requerimientos de negocio deben venir de la persona que paga para el desarrollo, mientras que los requerimientos de usuario deben venir de gentes que presionan las teclas para usar el producto.

Desafortunadamente, ambos tipos de clientes no pueden creer que tienen tiempo para trabajar con analistas de requerimientos quienes tienen que analizar y documentar los requerimientos.

2.1.2.2 Sociedad Cliente-Desarrollador

Productos excelentes de software son resultados de una buena ejecución basada en excelentes requerimientos. Los requerimientos de alta calidad en resulta de una comunicación y colaboración eficaz entre una sociedad desarrollador y cliente.

Un esfuerzo de colaboración puede trabajar solamente cuando todas las partes involucradas saben lo que necesitan para tener éxito, cuando se entiende y respeta lo que necesitan sus colaboradores para ser exitoso.

En la Tabla 2-1 podemos ver una Carta de Requerimientos de Derechos para los Clientes de Software, en la que lista 10 expectativas que el cliente puede llevar acabo respecto a su interacción con los analistas y desarrolladores durante las actividades de ingeniería de software. Inversamente, en la Tabla 2-2 una Carta de Requerimientos de Responsabilidades para los Clientes de Software, que lista 10

responsabilidades que el cliente tiene con los analistas y desarrolladores durante el proceso de requerimientos.

Tabla 2-1 Derechos para los Clientes de Software

	El Cliente tiene derecho a:
1.	Esperar que el analista hable su lenguaje.
2.	Esperar que le analista aprenda sobre el negocio y objetivos del sistema.
3.	Esperar que el analista estructure la información presentada durante la licitación de requerimientos en un documento SRS (software requirements specification)
4.	Esperar que el analista explique todos los productos generados durante el proceso de requerimientos.
5.	Esperar que el analista y desarrolladores le traten con respeto y mantenga una actitud de colaboración y profesional a través de sus interacciones.
6.	Que analistas y desarrolladores proporcionen ideas y alternativas para los requerimientos y para la implementación del producto.
7.	Describir las características del producto que lo hará más fácil y agradable de usar.
8.	Dar la oportunidad de ajustar los requerimientos para permitir la reutilización de componentes de software existentes.
9.	Recibir las estimaciones de costos, impacto y cambios (trade-offs) cuando se solicita un cambio en los requerimientos.
10.	Recibir un sistema que resuelva sus necesidades funcionales y de calidad.

Tabla 2-2 Responsabilidades para los Clientes de Software

	El cliente tiene la responsabilidad de:
1.	Educar a los analistas y desarrolladores sobre su negocio y definir la jerga del negocio.
2.	Pasar el tiempo necesario con los analistas de requerimientos, clarificándolos, y teniendo iteraciones.
3.	Ser específico y exacto cuando se proporciona la entrada de los requerimientos del sistema.
4.	Tomar decisiones oportunas sobre los requerimientos cuando se solicita hacerlo.
5.	Respetar el costo y viabilidad de los requerimientos.
6.	En colaboración con los desarrolladores, colocar prioridades en los requerimientos funcionales, características del sistema o caos de uso.
7.	Revisar documentos de requerimientos y evaluar prototipos.
8.	Comunicar los cambios en los requerimientos tan pronto como usted los sepa.
9.	Seguir el proceso organizacional de desarrollo para solicitar cambios en los requerimientos.
10.	Respetar los procesos que los analistas utilizan para desarrollar la ingeniería de requerimientos.

Como parte de la planeación del proyecto, los clientes y desarrolladores participantes deberían revisar estas dos listas y conocer el alcance de su significado. Así como asegurarse que participantes clave en el desarrollo de requerimientos entiendan y acepten sus responsabilidades.

2.1.3 Buenas Prácticas para la Ingeniería de Requerimientos.

La Tabla 2-3 lista cerca de 50 practicas agrupadas dentro de siete categorías, estas prácticas pueden ayudar a los equipos de desarrolladores a hacer mejor su trabajo en las actividades de requerimientos.

Tabla 2-3 Buenas Prácticas para la Ingeniería de Requerimientos

Conocimiento	Administración de Requerimientos	Administración de Proyectos
Análisis del seguimiento de Requerimientos	Definir un proceso de control-cambio en los requerimientos	Seleccionar el ciclo de vida apropiado
Educar a los usuarios representantes y administradores en el proceso de Ingeniería de Requerimientos	Establecer un tablero de control de cambios	Basar planes en los requerimientos
Entrenar a los desarrolladores en el dominio de la aplicación	Realizar un análisis del impacto de los cambios	Renegociar compromisos
Crear un glosario	Establecer una línea base (baselined) y control de las versiones de requerimientos	Administración de riesgos en los requerimientos
	Mantener la historia de los cambios	Seguimiento de esfuerzo de requerimientos
	Seguimiento del estatus de los requerimientos	Revisión de lecciones aprendidas
	Medición de la volatilidad de los requerimientos	
	Usar herramientas de administración de requerimientos	
	Crear una matriz de rastreabilidad requerimientos	

Desarrollo de Requerimientos

Licitación	Análisis	Especificación	Validación
Definir el proceso de desarrollo de requerimientos	Dibujar un diagrama de contexto	Adoptar la plantilla del SRS	Inspeccionar documentos de requerimientos
Definir visión y alcance	Crear prototipos	Identificar la fuente de los requerimientos	Prueba de requerimientos
Identificar clases de usuario	Analizar viabilidad	Etiquetar como único cada requerimiento	Definir criterios de aceptación
Seleccionar los champions del producto	Priorizar Requerimientos	Registrar las reglas de negocio	
Establecer grupos focales	Modelar los requerimientos	Especificar los atributos de la calidad	
Identificar casos de uso	Crear un diccionario de datos		
Identificar sistemas de evento y respuesta	Asignar los requerimientos a los subsistemas		
Mantener sesiones de trabajo para facilitar la licitación.	Aplicar QFD (Quality Function Deployment)		
Observar el desempeño de los usuarios en sus actividades			
Examinar problemas reportados			
Rehusar requerimientos			

2.1.3.1 Conocimiento

Pocos desarrolladores de software reciben un entrenamiento formal en ingeniería de requerimientos. Sin embargo, muchos desarrolladores realizan el rol de analista de requerimientos en cierto punto de su carrera. El proceso de requerimientos es esencial, y todos los participantes en los proyectos de software deberían entender los conceptos y prácticas de la ingeniería de requerimientos.

Entrenamiento del analista de requerimientos. Todos los miembros del equipo quienes van a funcionar como analistas deberían recibir un entrenamiento básico en ingeniería de requerimientos. Las habilidades del analista de requerimientos son; paciencia buena organización, tiene habilidades interpersonales efectivas y habilidades de comunicación, entiende el dominio de aplicación, y tienen un extenso kit de herramientas y técnicas para la ingeniería de requerimientos.

Educar usuarios representativos y administradores sobre requerimientos de software. Usuarios quienes participan en el desarrollo de software deberían recibir uno o dos días de capacitación sobre ingeniería de requerimientos.

Entrenamiento a los desarrolladores en conceptos del dominio de la aplicación. Ayudar a los desarrolladores a alanzar un entendimiento básico del dominio de aplicación, con un entrenamiento para las actividades de negocio del cliente, terminología y objetivos del producto que se esta empezando a hacer.

Creación de un glosario. Un glosario que defina términos especializados del dominio de la aplicación, reducirá malentendidos. Incluye sinónimos, términos que pueden tener múltiples significados.

2.1.3.2 Licitación de Requerimientos

Anteriormente se habló de los tres niveles de los requerimientos: negocio, usuario y funcionales. Estos vienen de diferentes fuentes y diferentes tiempos durante el proyecto, tienen diferentes propósitos y se necesitan documentar de diversas maneras. Los requerimientos de negocio expresados en el documento de alcance de proyecto no deben excluir ninguno de los requerimientos esenciales, y se debe de ser capaz de rastrear todos los requerimientos específicos de usuario. También se necesita licitar los requerimientos no funcionales, tales como la calidad y expectativas de desempeño desde, fuentes apropiadas.

Definir un proceso de desarrollo de requerimientos. Un documento con los pasos que la organización debe seguir para la licitación, análisis, especificación y validación de los requerimientos.

Escribir un documento de la visión y alcance. Un documento de visión y alcance contiene los productos de los requerimientos de negocio. El statement de visión brinda a los todos los participantes del proyecto un entendimiento común de los objetivos del producto. El alcance define el límite entre que se va a hacer y que es lo que no se va a hacer.

Identificar clases de usuarios y sus características. Para evitar pasar por alto las necesidades de cualquier comunidad de usuario, se requiere identificar los diferentes grupos de usuarios para el producto. Puede ser que difieran en la

frecuencia de uso, características de uso, niveles de privilegio o niveles de habilidad.

Seleccionar un producto champion para cada clase de usuario. Establecer por lo menos una persona quien pueda exactamente servir como la voz del cliente para cada clase de usuario. El producto principal presenta las necesidades de las clases de usuario y toma decisiones a su favor.

Establecer grupos principales de usuarios típicos. Convoca a grupos de usuarios representativos de productos anteriormente realizados o productos similares. Recolecta sus entradas en cuanto a funcionalidad y características de calidad para el producto de desarrollo.

Trabaja con usuarios representativos para identificar casos de uso. Explorar con usuarios representativos tareas que necesiten lograr con el software sus casos de uso. Discutir la interacción entre los usuarios y el sistema que va a permitir que terminen cada tarea.

Identificar sistemas de eventos y respuesta. Lista eventos externos que el sistema puede experimentar y sus respuestas previstas para cada evento.

Mantener talleres fáciles de licitación. Facilitar los talleres requerimientos-licitación que permiten la colaboración entre el analista y el cliente, esto es una manera de explorar las necesidades del usuario y de bosquejar los documentos de requerimientos.

Observar el desempeño de los usuarios en su empleo. Observar el desempeño en sus tareas de negocio estableciendo un contexto para el potencial de la nueva aplicación.

Examinar problemas reportados de actuales sistemas para ideas de requerimientos. Problemas reportados y peticiones de mejora de clientes que proveen una fuente rica de ideas para incluir en el último lanzamiento o en el nuevo producto.

Rehusar requerimientos a través de proyectos. Si los clientes solicitan funcionalidad similar a un proyecto ya existente, analizar si los requerimientos son bastante flexibles para permitir reutilizarlos o adaptarlos a los componentes existentes.

2.1.3.3 Análisis de Requerimientos

El análisis de requerimientos implica el refinamiento de los requerimientos para asegurar que todos los stakeholders o participantes del proyecto los entiendan y examinarlos para verificar si tienen errores, omisiones y otras deficiencias. El análisis incluye requerimientos a alto nivel en detalles, construcción de prototipos,

viabilidad de evaluación y prioridad de negociación. La meta es desarrollar requerimientos de suficiente calidad y detallados para que los administradores puedan construir estimaciones reales del proyecto y el staff técnico pueda diseñar, construir y probar.

Dibujar un diagrama contextual. Un diagrama de contexto es un simple modelo que muestra como el nuevo sistema cabe en su ambiente. Este define el limite y las interfaces entes el sistema a desarrollar y entidades externas al sistema, tales como usuarios, dispositivos de hardware, y cualquier otra información del sistema.

Crear una interfaz de usuario y técnicas de prototipo. Cuando los desarrolladores no están seguros de los requerimientos, se construye un prototipo –uno parcial, posible, o preeliminar a la implementación- haciendo los conceptos más tangibles.

Analizar la viabilidad de los requerimientos. Evaluar si la viabilidad de implementación de cada requerimiento es aceptada en costo y desempeño, en el ambiente de desarrollo previsto.

Priorizar requerimientos. Aplicar un análisis detallado para determinar, la prioridad relativa de implementación de las características del producto, casos de uso, o requerimientos individuales. De acuerdo a la prioridad, determinar que lanzamiento podría contener cada característica o requerimiento.

Modelar los requerimientos. Un análisis gráfico del modelo representa requerimientos de un alto nivel de abstracción. En contraste los detalles son mostrados en el SRS o la vista de interfaz de usuario que los prototipos proveen. Los modelos pueden revelar requerimientos incorrectos, inconsistentes, faltantes y superfluos.

Crear un diccionario de datos. La definición de todos los artículos de datos y estructuras reside en un diccionario de datos. Esto permite a cualquiera que trabaje en el proyecto usar una definición consistente de datos.

Asignar los requerimientos a los subsistemas. Requerimientos para un producto complejo que contiene múltiples subsistemas se debe repartir entre varios componentes, recursos humanos, software y hardware (Nelson, 1990).

Aplicar Quality Function Deployment (QFD). Es una técnica rigurosa para relacionar características y atributos del producto con el valor del cliente (Zultner 1993; Pardee 1996). Proporciona identificar de manera analítica la máxima satisfacción del cliente.

2.1.3.4 Especificación de Requerimientos

No importa como se obtengan los requerimientos, documentarlos de una manera consistente, accesible y analizable es una buena práctica. Puede registrar los requerimientos de negocio en un documento de visión y alcance. Los requerimientos de usuario típicamente son representados en forma de casos de uso o como tablas de evento-respuesta. El SRS contiene los requerimientos funcionales y no funcionales detallados del software.

Adoptar un template SRS. Define un template estándar para documentar requerimientos de software en la organización. El template provee una estructura consistente para registrar la descripción funcional y otra información relacionada con los requerimientos. Muchas organizaciones empiezan con el template del SRS descrita en el estándar de la IEEE 830-1998 (IEEE 1998b).

Identificar fuentes de los requerimientos. Asegurar que todos los stakeholders sepan porqué cada requerimiento se encuentra en el SRS y facilitar la clasificación y seguimiento de cada requerimiento a su origen.

Colocar una sola etiqueta a cada requerimiento. Definir un estándar o convención para que cada requerimiento en el SRS cuente con una etiqueta única de identificación. El estándar o convención debe ser bastante robusto para soportar adiciones, cancelaciones, y cambios hechos a los requerimientos en un cierto plazo.

Registrar las reglas de negocio. Las reglas de negocio incluyen políticas corporativas, regulaciones de gobierno y algoritmos computacionales. Documentar las reglas de negocio separadas del SRS, porque tienen una existencia más allá del alcance del proyecto.

Especificar los atributos de calidad. Ir más allá de la discusión de la funcionalidad, es decir, explorar las características de calidad que ayudaran al producto a satisfacer las necesidades del cliente. Estas características incluyen desempeño, eficiencia, confiabilidad, utilidad, entre otras.

2.1.3.5 Validación de Requerimientos

La validación asegura que el estado de los requerimientos es correcto, demostrando las características deseadas de calidad, y cubrirán las necesidades del cliente. Escribiendo casos de prueba a menudo revelan ambigüedad e imprecisión en los requerimientos.

Inspeccionar documentos de requerimientos. La inspección formal de documentos de requerimientos es una práctica disponible de alto valor para la calidad de software. Montar un equipo pequeño de inspectores que represente diferentes prospectivas (tales como analistas, clientes, desarrolladores y tester),

examinando cuidadosamente el SRS, analizando modelos y la información relacionada para encontrar defectos.

Test de requerimientos. El objetivo es derivar los casos de pruebas funcionales de los requerimientos de usuario, documentando el comportamiento del producto bajo específicas condiciones.

Definir criterio de aceptación. Pedir al usuario que describa cómo se determina si el producto resuelve sus necesidades cumple las expectativas de uso.

2.1.3.6 Administración de Requerimientos

Una vez que se tienen los requerimientos para trabajar, se debe hacer frente a los inevitables cambios de clientes, administradores, mercadotecnia, equipo de desarrollo y otras peticiones durante el desarrollo. La administración eficaz demanda procesos para proponer cambios y evaluar su costo potencial e impacto en el proyecto. El Comité de Control de Cambios (CCB), compuesto por stakeholders clave, deciden cuales cambios propuestos son incorporados.

Definir un proceso de control de cambio en los requerimientos. Se debe establecer un proceso con el cual los cambios propuestos en los requerimientos son analizados y resueltos.

Establecer un tablero de control de cambios. Establecer un pequeño grupo de stakeholders como CCB para recibir propuestas de cambios en los requerimientos, los cuales los evalúan, se decide que rechazar y que aceptar, así como también se colocan prioridades y relanzamientos.

Realizar un análisis del impacto de los cambios. El análisis del impacto ayuda al CCB a tomar decisiones informadas de negocio. Evalúa cada propósito de un cambio en los requerimientos para determinar el efecto que tendrá en el proyecto.

Establecer una línea base (baselined) y control de las versiones de requerimientos. La línea base consiste de requerimientos que han sido comprometidos para la implementación de una liberación específica. Después de que los requerimientos han sido puestos en la línea base, los cambios deberán ser hechos sólo a través de proceso definido de control de cambios.

Mantener la historia de los cambios en los requerimientos. Registrar las fechas de cuando los requerimientos fueron cambiados, los cambios que fueron realizados, quien hizo cada cambio y por qué.

Seguimiento del estatus de cada requerimiento. Establecer una base de datos por cada requerimiento funcional discreto. Almacenar los atributos dominantes de cada requerimiento, incluyendo el estatus (tales como propuesto, aprobado, implementado o verificado).

Medición de la volatilidad de los requerimientos. Registrar el número base de requerimiento y el número de cambios propuestos y aprobados (Adiciones, modificaciones, eliminaciones) por semana.

Usar herramientas de administración de requerimientos. Herramientas comerciales de administración de requerimientos permiten almacenar varios tipos de requerimientos en una base de datos. Se puede definir atributos para cada requerimiento, seguir el estatus de cada requerimiento.

Crear una matriz traceable de requerimientos. Establecer una tabla que ligue cada requerimiento funcional con los elementos del diseño y código que los ponen en ejecución y posteriormente lo verifiquen.

2.1.3.7 Administración de Proyecto

La administración del proyecto de software se relaciona directamente con el proceso de requerimientos. Basar los recursos, calendarios y comisiones del proyecto en los requerimientos debe ser implementado. Los cambios en los requerimientos afectan a los planes del proyecto. Los proyectos deben anticipar que los requerimientos cambian y que el alcance crece.

Selecciona el apropiado ciclo de vida de desarrollo de software. La organización debe definir varios tipos de ciclos de vida para el desarrollo que sean apropiados para varios tipos de proyectos y diferentes grados de incertidumbre en los requerimientos (Wigers, 2002).

Basar los planes en los requerimientos. Desarrollar planes y calendarios para un proyecto de manera iterativa tanto como sea posible hasta que el alcance y los requerimientos detallados sean claros.

Renegociar el proyecto cuando los requerimientos cambian. En la incorporación de nuevos requerimientos en el proyecto, evaluar si se puede alcanzar el calendario actual y si va a alcanzar la calidad deseada con los recursos disponibles.

Documentar y administrar riesgos relacionados con los requerimientos. Identificar y documentar los riesgos relacionados con los requerimientos como parte de las actividades de administración de riesgos.

Seguimiento del esfuerzo realizado en la ingeniería de software. Registrar el esfuerzo realizado en el desarrollo de requerimientos y actividades de administración. Utilizar estos datos para determinar si las actividades se están realizando según lo previsto y mejorar el plan de los recursos requeridos para futuros proyectos.

Revisión de lecciones aprendidas en requerimientos, en otros proyectos. Una organización que aprende, conduce retrospectivas de proyectos también llamados postmortems y revisores post-proyecto para coleccionar lecciones aprendidas.

2.1.3.8 Empezar con Nuevas Prácticas

La Tabla 2-4 agrupa las prácticas descritas anterior mente y su relativo impacto que pueden tener en la mayoría de los proyectos y su relativa dificultad de implementación.

Tabla 2-4 Implementando Nuevas Practicas de Ingeniería de Requerimientos

	Impacto		**Dificultad**
	Alto	***Medio***	***Bajo***
Alto	• Define el proceso de desarrollo de requerimientos • Planes base de requerimientos • Renegociación	• Identificar Casos de Uso • Atributos especiales de calidad • Priorizar requerimientos • Adopción de el SRS Template • Definir el proceso de control de cambios • Establecer CCB • Inspección de documentos de requerimientos • Asignar requerimientos a los subsistemas • Registrar las reglas de negocio	• Entrenamiento a los desarrolladores en el dominio de la aplicación • Definir la visión y alcance • Identificar clases de usuarios • Dibujar un diagrama de contexto • Identificar fuente de los requerimientos. • Control de versiones de los requerimientos
Medio	• Educar a los usuarios sobre los requerimientos • Modelo de requerimientos • Administración	• Selección del champions del producto • Establecer grupos principales • Creación de	• Análisis de viabilidad • Creación de un glosario • Creación de un diccionario de datos

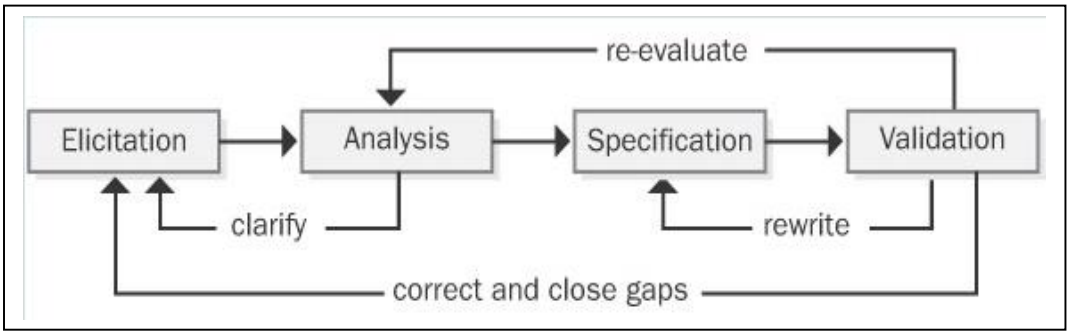
Impacto			Dificultad
	de los riesgos en los requerimientos • Uso de herramientas para la administración de requerimientos • Seguimiento a los requerimientos • Facilidad en los talleres de licitación	prototipos • Definir los criterios de aceptación • Análisis del impacto de los cambios • Seleccionar el apropiado ciclo de vida	• Observar el desempaño de los usuarios en su trabajo • Identificar eventos y respuestas • Etiquetar cada requerimiento con un identificador único • Probar los requerimientos. • Seguir el estado de los requerimientos. • Revisión de lecciones aprendidas en el pasado
Bajo	• Reusar requerimientos • Aplicar funciones de calidad. • Mantener volátil la historia en los requerimientos.	• Medir a los requerimientos • Esfuerzo en los requerimientos	• Examinar reportes de problemas

2.1.3.9 Un Proceso de Desarrollo de Requerimientos

No realizar las actividades del desarrollo de requerimientos como licitación, análisis, especificación y validación en forma lineal, en una secuencia de pasos. En la práctica estas actividades son interpoladas, incrementales e iterativas como nos muestra la Figura 2-5. En el trabajo con los clientes en el rol de analista, se harán preguntas, escuchando lo que dicen los clientes y observando lo que él hace (licitación). Se procesará información para entenderla y se clasificará en varias categorías, y se relacionarán las necesidades del cliente con los requerimientos de software posibles (análisis). Se estructurarán todas las entradas de los clientes derivadas en los requerimientos escritos en documentos y

diagramas (especificación). Después, preguntará al cliente representativo que confirme que es correcto lo que se ha escrito, complete y corrija cualquier error (validación). Este proceso iterativo continúa a través del desarrollo de requerimientos.

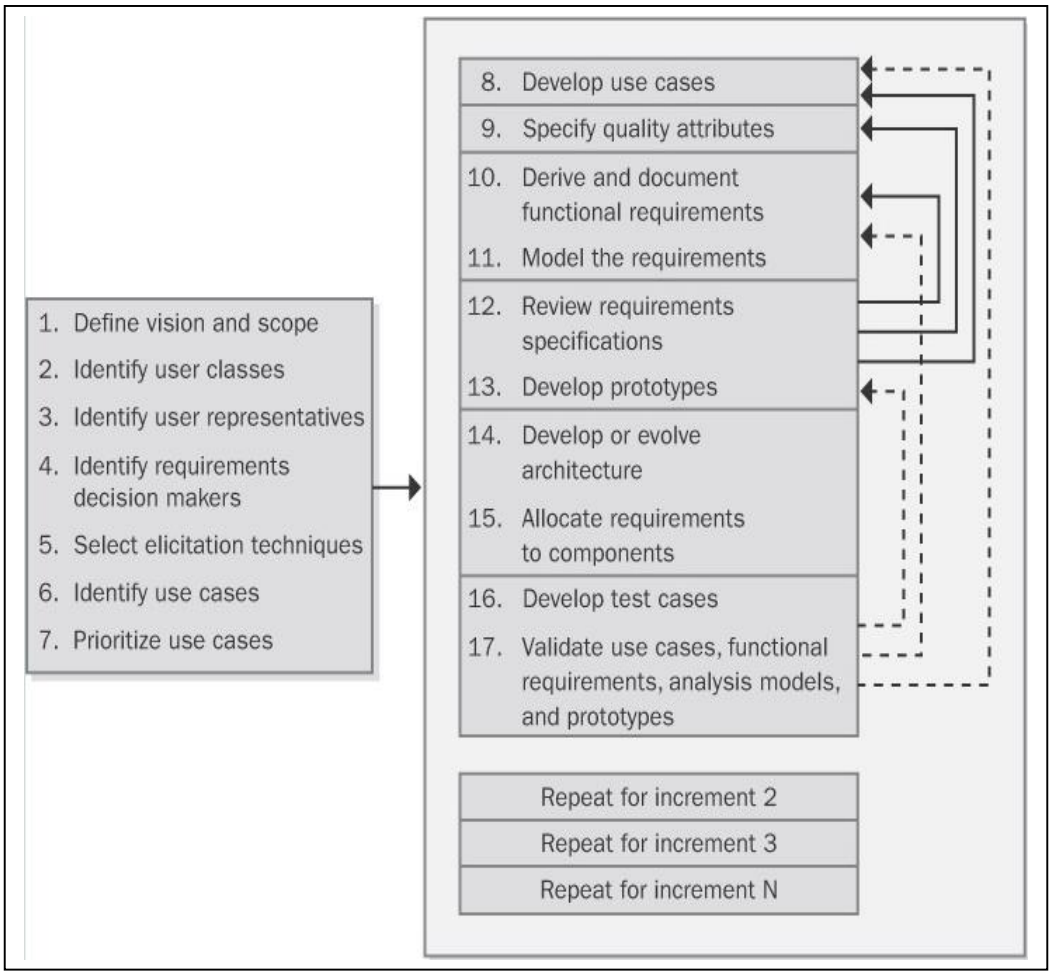
Figura 2-5 El desarrollo de los requerimientos es un proceso Iterativo



Fuente: Karl E. Wiegers, 2003

Debido a la diversidad de proyectos de desarrollo de software y de las culturas de organización, no existe una sola formula para el desarrollo de requerimientos. En la Figura 2-6 sugiere un marco de proceso que trabaja con el desarrollo de requerimientos –ajustable a cualquier organización- para múltiples proyectos.

Figura 2-6 Marco de Proceso que Trabaja con el Desarrollo de Requerimientos



Fuente: Karl E. Wiegers, 2003

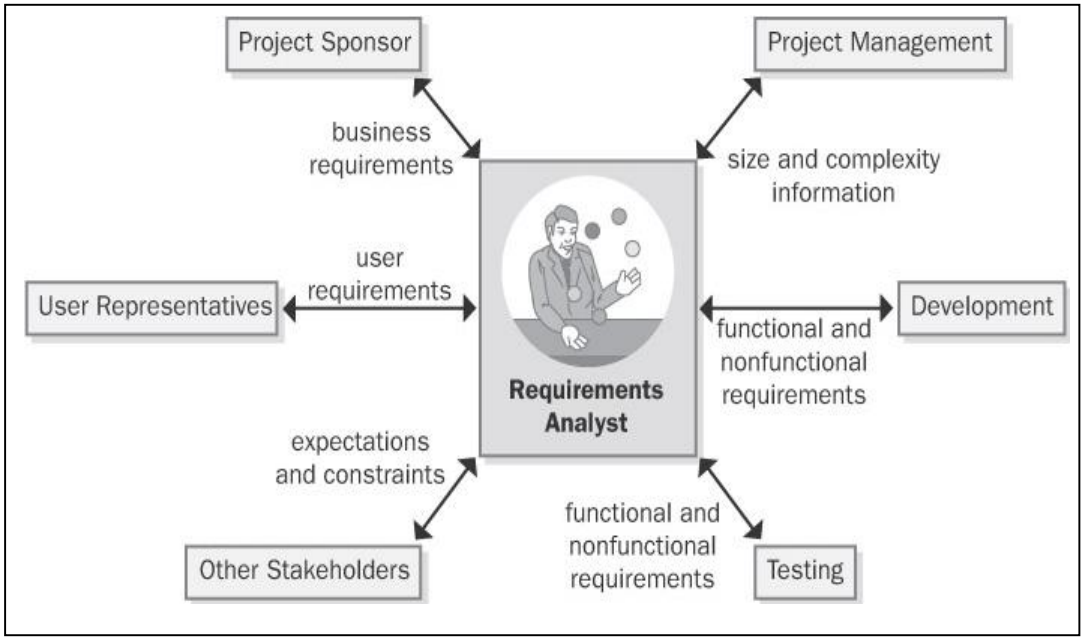
2.1.4 El Analista de Requerimientos

Explícita o implícitamente alguien lleva el rol de analista de requerimientos en cada proyecto de software. Sinónimos para *analista de requerimientos* incluye *analista de sistemas*, *ingeniero de requerimientos*, *administrador de requerimientos* y *analista*.

2.1.4.1 El Rol del Analista de Requerimientos

El analista de requerimientos es el individuo que tiene la responsabilidad principal de recolectar, analizar, documentar, y validar las necesidades de los stakeholders de proyecto. El analista de requerimientos sirve como conducto principal para que los requerimientos circulen entre la comunidad del cliente y el equipo de desarrollo de software. Como se puede apreciar en la Figura 2-7.

Figura 2-7 Comunicación entre el Cliente y los Stakeholders de Desarrollo por medio del Analista



Fuente: Karl E. Wiegers, 2003

El Analista de Requerimientos es un rol de proyecto, no necesariamente un titulo de trabajo. Uno o más especialistas dedicados podrían llevar acabo el rol, o podría ser asignado a miembros del equipo que también tienen otras funciones de trabajo. Estas funciones incluyen administrador de proyecto, experto en la materia (subject matter expert, SME), desarrollador, y usuario ecuánime. Sin considerar sus otras responsabilidades de proyecto, los analistas deben tener la destreza, el conocimiento, y la personalidad para llevar acabo el rol de analista.

2.1.4.1.1 Las Tareas del Analista

Definir los requerimientos de negocio. Su trabajo como analista, comienza cuando ayuda al negocio, administrador de producto, o administrador de marketing a definir los requerimientos de negocio del proyecto. Quizás la primera pregunta es "¿Por qué estamos emprendiendo este proyecto?". Los requerimientos de negocio incluyen un statement de los objetivos de negocio de la organización y la visión final de lo que el sistema será y hará. Podría sugerir el template para el documento de visión y alcance.

Identificar stakeholders del proyecto y clases de usuarios. El documento de visión y alcance podría ayudar a identificar las clases de usuario y otros stakeholders importantes para el producto. Después, trabajar con los

patrocinadores de negocio para seleccionar representantes apropiados para cada clase de usuario, conseguir su participación, y negociar sus responsabilidades

Extraer requerimientos. Un analista pro-activo ayuda a los usuarios a articular las capacidades del sistema que necesitan para cubrir sus objetivos de negocio. Podría emplear las técnicas de obtención de información seleccionadas de la siguiente lista:

- Entrevistas
- Análisis de documento
- Encuestas
- Visitas al cliente
- Análisis de proceso de negocio
- Flujo de trabajo y análisis de tarea
- Listas de evento
- Análisis del producto competitivo
- Ingeniería inversa de sistemas existentes

Analizar los requerimientos. Descubrir palabras vagas y poco sólidas que causan ambigüedad. Indicar requerimientos opuestos y áreas que necesitan más detalle. Especificar los requerimientos funcionales a un nivel de detalle apropiado para el uso por los desarrolladores que los implementarán.

Escribir especificaciones de requerimientos. El analista es responsable de escribir especificaciones bien-organizadas que expresan claramente lo entendido. Usar templates estándar para casos de usos y el SRS acelera el desarrollo de requerimientos recordando al analista los tópicos de los que tiene que hablar con el usuario representante.

Modelar los requerimientos. El analista debe determinar cuando es útil representar requerimientos usando métodos aparte del texto. Estas vistas alternativas incluyen a varios tipos de modelos de análisis gráficos, tablas, ecuaciones matemáticas, y prototipos. Para maximizar la información y claridad, dibujar modelos de análisis de acuerdo con las convenciones de un lenguaje de modelado estándar.

Validar requerimientos. El analista debe asegurar que el statement de requerimientos debe poseer todas las características de un “statement de requerimientos” y que un sistema basado en los requerimientos satisfará las necesidades de usuario. Los analistas son los participantes principales en los peer reviews de los documentos de requerimientos. También deben revisar diseños, código y casos de prueba que fueron derivados de las especificaciones de requerimientos para asegurar que los requerimientos fueron interpretados correctamente.

Facilitar la prioridad de los requerimientos. El analista actúa como intermediario de la colaboración y la negociación entre varias clases de usuario y los desarrolladores se aseguran de que se tomen decisiones sensatas de prioridad.

Administrar requerimientos. Un analista de requerimientos está involucrado durante todo el ciclo de vida de desarrollo de software, así que debe ayudar a crear, revisar, y ejecutar el plan de administración de requerimientos de proyecto. Después de establecer la línea base de los requerimientos, el enfoque del analista cambia para administrar esos requerimientos y verificar su satisfacción en el producto. Almacenar los requerimientos en una herramienta comercial de administración de requerimientos facilita su continua administración.

La administración de requerimientos incluye el seguimiento del estatus de requerimientos funcionales individuales como su progreso desde la inserción hasta la comprobación en el producto integrado.

2.1.4.1.2 Habilidades Esenciales del Analista

Habilidad de escuchar. Llegar a ser muy competente en la comunicación de dos vías. Aprender cómo escuchar eficazmente. Escuchar implica eliminar las distracciones, mantener una pose activa atenta y contacto ocular, y repetir puntos clave para confirmar el conocimiento. Comprender lo que las personas están diciendo y también leer entre líneas para detectar la inseguridad de lo que dicen.

Habilidad de entrevistar e interrogar. Hacer preguntas correctas que recubran información esencial de los requerimientos. Con la experiencia llegar hacer hábil en el arte de hacer preguntas que revelan y aclaran las incertidumbres, desacuerdos, suposiciones, y experiencias malas (Gause and Weinberg, 1989).

Habilidad analítica. Un analista eficaz puede pensar en múltiples niveles de abstracción.

Habilidad de facilitación. La habilidad para facilitar talleres de licitación de requerimientos es una capacidad necesaria del analista (Gottesdiener, 2002).

Habilidad de observación. Un analista observador detectará los comentarios hechos en pasado que pueden resultar ser importantes. La habilidad de observación revela requerimientos adicionales que nadie había mencionado.

Habilidad de escritura. El entregable principal del desarrollo de requerimientos es una especificación escrita que comunica información entre clientes, mercadotecnia, administradores, y personal técnico. El analista necesita una habilidad sólida del lenguaje y una habilidad para expresar las ideas complicadas claramente.

Habilidad organizativa. Los analistas deben trabajar con una selección extensa de información desordenada obtenida durante la licitación y análisis.

Habilidad de modelado. Diagrama de flujo de datos, diagrama entidad-relación, UML, etc., deben ser parte del repertorio de cada analista.

Habilidad interpersonal. Los analistas necesitan la habilidad de conseguir a gente con intereses competentes para trabajar juntos. Un analista debe sentirse cómodo hablando con individuos con funciones de trabajo diversas y de todos los niveles de la organización.

Creatividad. El analista no es simplemente un escritor que registra lo que el cliente dice. Los mejores analistas inventan requerimientos (Robertson, 2002). Son innovadores, imaginan nuevos mercados y oportunidades de negocio, y piensan de manera que sorprenden y "encantan" a sus clientes. Un analista muy valioso encuentra formas creativas de satisfacer las necesidades que los usuarios incluso que no sabían que tenían.

2.1.4.1.3 Conocimientos Esenciales del Analista

Un analista efectivo tiene un conjunto de herramientas de técnicas disponible y sabe cuándo –y cuando no- usar cada una.

Conocimientos de dominio de aplicación son una posesión fuerte para un analista eficaz.

2.1.4.2 El Quehacer de un Analista

2.1.4.2.1 El Usuario Antiguo

Los usuarios que llegan a ser analistas de requerimientos necesitarán aprender más sobre la parte técnica de desarrollo de software con el propósito de que puedan representar la información en las formas más apropiadas para sus múltiples audiencias.

2.1.4.2.2 El Antiguo Desarrollador

Los administradores de proyecto que carecen de un analista de requerimientos dedicado, a menudo esperan que un desarrollador haga el trabajo. Por supuesto, muchos desarrolladores reconocen lo crítico del proceso de requerimientos y son voluntarios para trabajar como analistas cuando es necesario.

2.1.4.2.3 El Experto en la Materia

Ralph Young (2001), recomienda tener un analista de requerimientos que sea un experto en el dominio de aplicación o un experto en la materia.

2.1.4.3 Crear un Ambiente de Colaboración

El analista de requerimientos tiene la responsabilidad para forjar una relación de colaboración con los usuarios representantes y otros stakeholders de proyecto.

2.2 Desarrollo de Requerimientos de Software

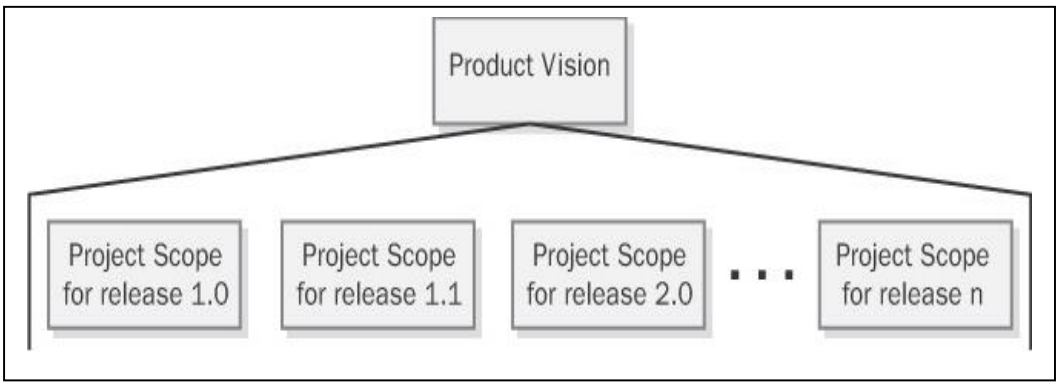
2.2.1 Estableciendo la Visión del Producto y el Alcance del Proyecto

2.2.1.1 Definiendo la visión con los requerimientos de negocio

La visión describe que es el producto y que eventualmente puede llegar a ser. El alcance del proyecto identifica que porción de la visión de largo plazo del producto va a alcanzar el actual proyecto. El alcance también define las limitaciones del proyecto.

La visión se aplica al producto en su totalidad. Cambiará relativamente lento tal y como se vaya colocando la estrategia de producto o como el objetivo de negocio del sistema de información se desarrolle en el tiempo. El alcance pertenece a un específico proyecto o iteraciones que pueden ser implementadas en el siguiente incremento de la funcionalidad del producto, como lo muestra la Figura 2-8. El alcance es más dinámico que la visión porque el administrador del proyecto ajusta el contenido de cada relanzamiento dentro del calendario, presupuesto, recursos y restricciones de calidad.

Figura 2-8 La Visión del Producto abarca el Alcance para cada lanzamiento planeado



Fuente: Karl E. Wiegers, 2003

2.2.1.1.1 Requerimientos de Negocio y Casos de Uso

Los requerimientos de negocio determinan el sistema de tareas de negocio (casos de uso) que las aplicaciones permiten (anchura) y la profundidad de la ejecución de cada caso de uso. Si los requerimientos de negocio permiten ayudar a determinar que particular caso de uso esta fuera del alcance del proyecto, se esta tomando una decisión de anchura.

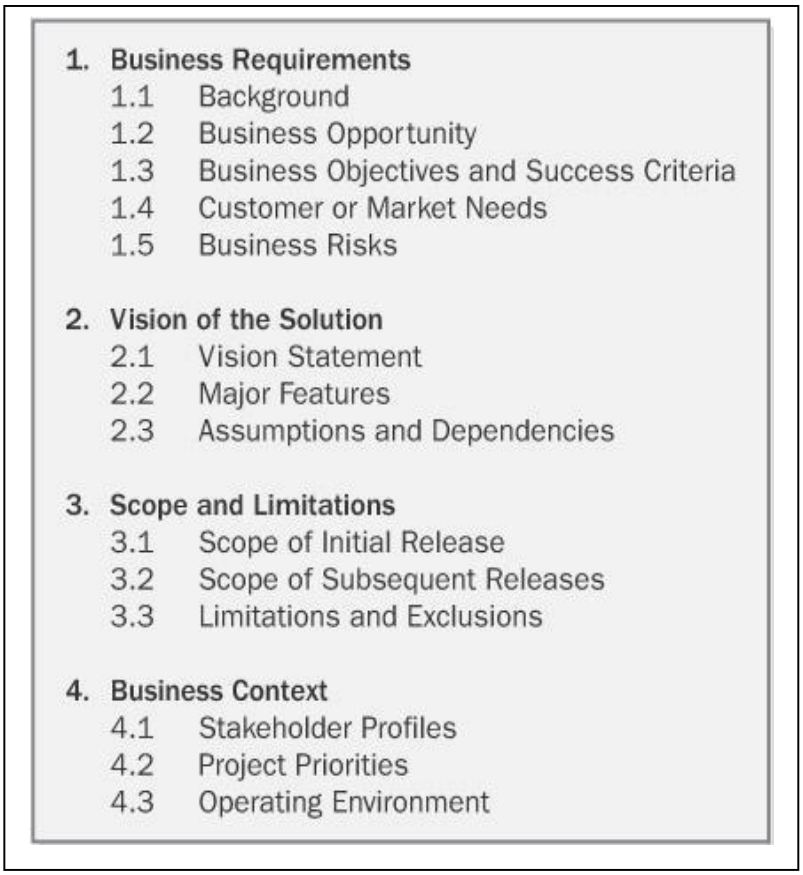
Los requerimientos de negocio implicarán que cada que cada caso de uso demande robustez, comprensión e implementación funcional. Los requerimientos de negocio influencian las prioridades de implementación para los casos se uso y su asociación con los requerimientos funcionales.

2.2.1.2 Documento de Visión y Alcance

El documento de visión y alcance colecciona los requerimientos de negocio dentro de un solo documento que fija la etapa para el desarrollo subsecuente. El dueño del documento de visión y alcance es el ejecutivo de proyecto, personas que financian el proyecto, o alguien similar. El analista de requerimientos puede trabajar con estas personas para escribir el documento de visión y alcance del proyecto. El origen de los requerimientos de negocio debe venir de individuos que tienen un sentido claro del porqué se esta realizando el proyecto.

La Figura 2-9 sugiere un template para el documento de visión y alcance del documento. Como cualquier platilla este se puede adaptarse para resolver las necesidades especificas de diferentes proyectos.

Figura 2-9 Template para el Documento de Visión y Alcance



Fuente: Karl E. Wiegers, 2003

2.2.1.2.1 Requerimientos de Negocio

Los requerimientos de negocio describen los beneficios primarios que el nuevo sistema proporcionará a sus patrocinadores (sponsors), compradores y usuarios.

2.2.1.2.1.1 Background

Resume un análisis razonable y el contexto para el nuevo producto. Provee una descripción general de la historia o situación por la que se tomo la decisión de construir este producto.

2.2.1.2.1.2 Oportunidad de Negocio

Para productos comerciales, describe la oportunidad de mercado que existe en el mercado en el cual el producto competirá. Para sistemas de información corporativos, describen el problema de negocio que se esta empezando a resolver

o el proceso que se esta empezando a mejorar, así como el ambiente en el cual el sistema será utilizado. Incluir una evaluación comparativa de la existencia de productos y soluciones potenciales, indicando el por qué el producto es atractivo y las ventajas que proveerá. Describir los problemas que no se pueden solucionar sin el producto.

2.2.1.2.1.3 Objetivos de Negocio y Criterios de Éxito

Resume los beneficios importantes de negocio que el producto va a proveer de una manera cuantitativa y medible. Determina como los stakeholders van a definir y medir el proyecto actual (Wiegers, 2002). Incluye factores que proveen el impacto más grande para el éxito del proyecto, incluyendo factores que están fuera de control de la organización. Especificar los criterios medibles para determinar si se han resuelto los objetivos de negocio.

2.2.1.2.1.4 Necesidades del Cliente y de Mercado.

Describe las necesidades típicas de clientes o del segmento del mercado, incluyendo necesidades de productos actuales o necesidades que los productos actuales no resuelven. Presenta problemas que los clientes actualmente encuentran y que el nuevo producto resolverá, y provee ejemplos de cómo el cliente utilizará el producto.

2.2.1.2.1.5 Riesgos de negocio

Resumirá los mayores riesgos de negocio asociados con el desarrollo del producto. Categorías de riesgos incluyen competencia de mercado, problemas de tiempo, aceptación de usuarios, problemas de implementación y posibles impactos negativos en el negocio. Estimar la posible perdida en cada riesgo, la posibilidad de que ocurra y la capacidad de controlarlo.

2.2.1.2.2 Visión de la Solución

Esta sección del documento establece una visión estratégica para que el sistema alcance los objetivos de negocio. Esta visión establece un contexto para tomar decisiones a través del curso de vida del producto.

2.2.1.2.2.1 Statement de Visión

Escribir un statement consistente de la visión que resuma a largo plazo el propósito y intensiones del nuevo producto. El statement de visión deberá reflejar una visión equilibrada que satisfaga las necesidades de los diversos stakeholders. Por ejemplo anticipaciones de mercado, estrategias corporativas, ambiente de

arquitectura y limitación de recursos. Las siguientes palabras son claves para una buena definición del statement de visión:

- **Para** [cliente objetivo]
- **Quién** [declaración de la necesidad u oportunidad]
- **El** [nombre del producto]
- **Es** [categoría del producto]
- **Para que** [beneficios clave, razón por la cual se debe comprar o usar]
- **Diferente** [principal competidor alternativo, sistema actual, o proceso actual de negocio]
- **Nuestro producto** [declaración de diferencias y ventajas del nuevo producto]

2.2.1.2.2.2 Características Importantes

Nombre o enumere cada una de las características del nuevo producto o de las capacidades que hacen diferente al producto, acentuando esas características que lo distinguen de productos anteriores o de productos de la competencia.

2.2.1.2.2.3 Suposiciones y Dependencias

Registrar cualquier suposición que los stakeholders hicieron al concebir el proyecto así como cuando escribieron la visión y alcance del proyecto. Frecuentemente las suposiciones que una parte ha realizado no son compartidas por otros. Si se han escrito y se revisan, se podrán acordar las suposiciones básicas y fundamentales.

2.2.1.2.3 Alance y Limitaciones

El alcance del proyecto define el concepto y rango de soluciones propuestas. Las limitaciones detallan ciertas capacidades que el producto no incluye. El alcance y las limitaciones ayudan a establecer expectativas realistas a los stakeholders. Algunas veces los clientes solicitan características que son muy caras o que estan fuera del alcance del producto. Los requerimientos que estan fuera del alcance deben ser rechazados a menos que sean tan valiosos que el alcance deba ampliarse para acomodar estos requerimientos, con los cambios correspondientes de presupuesto, calendario y recursos. Se deben guardar los requerimientos que fueron rechazados, el porqué fueron rechazados y el por que pueden reaparecer.

2.2.1.2.3.1 Alcance del lanzamiento Inicial

Resumir las características principales que se plantean para el lanzamiento inicial del producto. Describir las características de calidad que permitan que el producto proporcione los beneficios previstos a sus diferentes tipos de usuarios.

2.2.1.2.3.2 Alance de los subsecuentes lanzamientos

Los lanzamientos subsecuentes permiten adicionar casos de uso, características y enriquecer las capacidades de los casos de uso iniciales y características. Se puede mejorar el desempeño del sistema, confiabilidad, y otras características de calidad que hacen madurar al producto.

2.2.1.2.3.3 Limitaciones y Exclusiones

Define los límites entre qué esta dentro y qué esta fuera, para facilitar el manejo de las expectativas del cliente sobre el alcance del proyecto. Enumera cualquiera de las características o capacidades que los stakeholders pueden anticipar pero no son planteados para su inserción en un lanzamiento específico del producto.

2.2.1.2.4 Contexto del Negocio

Esta sección resume algunos problemas de negocio del proyecto, incluyendo perfiles de las principales categorías de stakeholders y prioridades del proyecto para la gerencia.

2.2.1.2.4.1 Perfil del Stakeholder

Los stakeholders son individuos, grupos u organizaciones que están implicadas en un proyecto, afectados por el resultado y pueden influir directamente en el resultado (Proyect Management Institute 200; Smith, 2000). Los perfiles de los stakeholders describen las diferentes categorías de los clientes y otros stakeholders claves para el proyecto. Cada perfil de stakeholder debe incluir la siguiente información:

- Valor o ventaja que el stakeholder recibirá del producto.
- Actitudes probables hacia el producto.
- Principales características de interés.
- Restricciones.

2.2.1.2.4.2 Prioridades del Proyecto

Para permitir la efectiva toma de decisiones, los stakeholders deben colocar prioridades en el proyecto. Una manera de acercarse a esto es considerar las cinco dimensiones de los proyectos de software: características (alcance), calidad, calendario, costo y staff (Wiegers, 1996a).

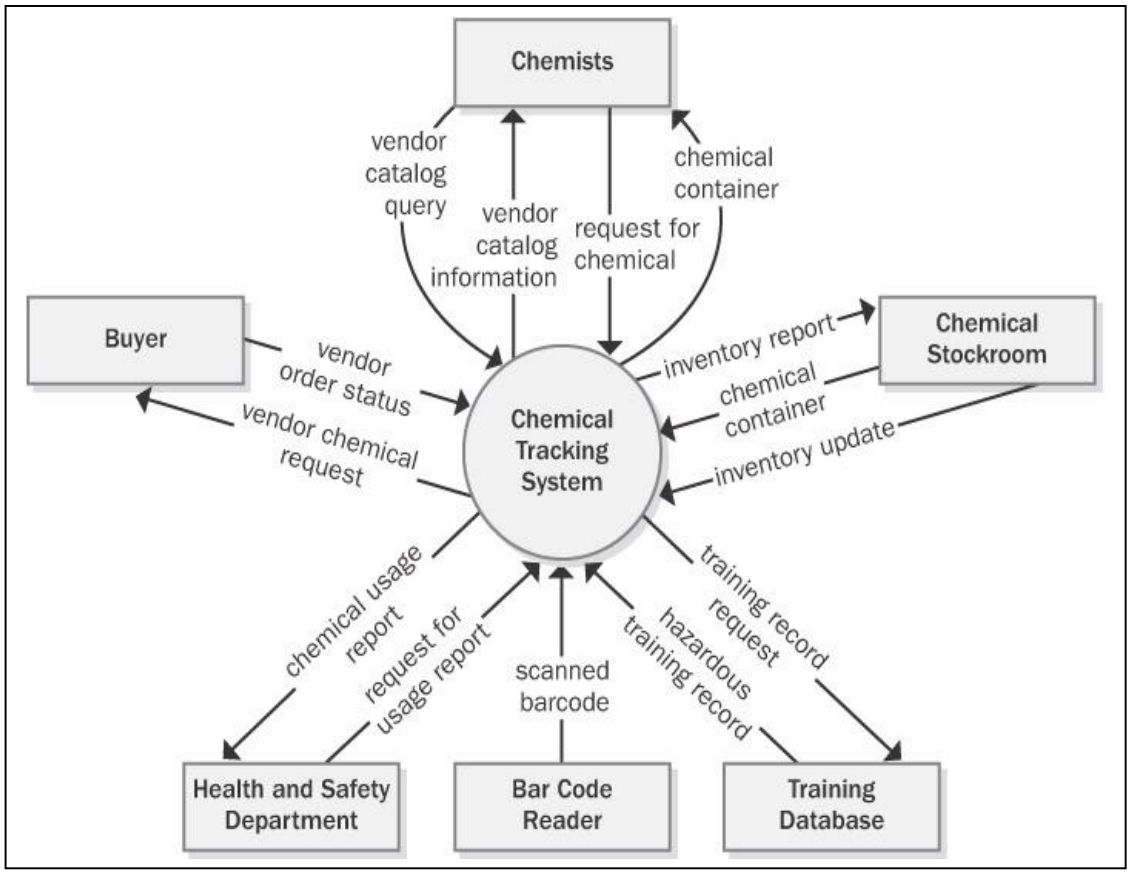
2.2.1.2.4.3 Ambiente de funcionamiento

Describe el ambiente en el cual el sistema va a ser usado y define la disponibilidad, la confiabilidad, funcionamiento y la integridad de los requerimientos. Esta información influenciará significativamente la definición de la arquitectura del sistema.

2.2.1.3 Diagrama de Contexto

La descripción del alcance establece el límite y las conexiones entre el sistema que estamos desarrollando y todo lo que lo rodea. El diagrama de contexto gráficamente ilustra estos límites. Identifica las entidades fuera del sistema que interconectan al él de cierta manera, como datos, control y flujos entre las entidades externas y el sistema. El diagrama de contexto es el nivel superior de la abstracción del diagrama de flujo de datos desarrollado según los principios del análisis estructurado (Robertson y Robertson 1994), pero es un modelo útil para los proyectos que siguen cualquier metodología del desarrollo.¶Usted puede incluir el diagrama de contexto en el documento de visión y alcance, en el SRS, o como parte de un modelo de flujo de datos para el sistema. El propósito de las herramientas tales como el diagrama de contexto es fomentar la comunicación entre y con los stakeholders del proyecto. La Figura 2-10 da una porción del diagrama de contexto para el sistema Chemical Tracking System.

Figura 2-10 Ejemplo de un diagrama de Contexto



Fuente: Karl E. Wiegers, 2003

2.2.1.4 Mantener el Alcance Enfocado

Los requerimientos de negocio y una comprensión de cómo los clientes utilizarán el producto provee de valiosas herramientas para mantener el alcance del proyecto y sus posibles cambios. El cambio del alcance del proyecto no es una cosa mala si ayuda a dirigir el proyecto a una evolución de las necesidades del cliente. La visión y alcance del proyecto permite determinar si las características y requerimientos del proyecto son apropiadas para la inserción del proyecto.

2.2.2 Encontrar la Voz del Cliente

El éxito en los requerimientos de software, y por lo tanto en el desarrollo de software, depende de obtener la voz del cliente cerca del oído del desarrollador. Para encontrar la voz del cliente, seguir los siguientes pasos:

- Identificar las diferentes clases de usuario para el producto.

- Identificar las fuentes de los requerimientos de usuario

- Seleccionar y trabajar con individuos que representan cada clase de usuario y otros grupos de stakeholders.

- Estar de acuerdo sobre quién tomará decisiones de requerimientos para el proyecto

La participación del cliente es la única manera de evitar una brecha de expectativa, una igualdad equivocada entre el producto que los clientes esperan recibir y el producto que los desarrolladores desarrollan.

2.2.2.1 Fuentes de los Requerimientos

Los orígenes de los requerimientos de software dependerán de la naturaleza del producto y su ambiente de desarrollo. Típicas fuentes de los requerimientos de software:

- Entrevistas y discusiones con los usuarios potenciales.
- Documentos que describen productos actuales o competitivos.
- Especificaciones de requerimientos del sistema.
- Informes del problema y petición de mejoramiento para un sistema actual.
- Encuestas de mercadotecnia y cuestionarios.
- Observaciones sobre el trabajo de los usuarios.
- Análisis del escenario de las tareas del usuario.
- Eventos y repuestas.

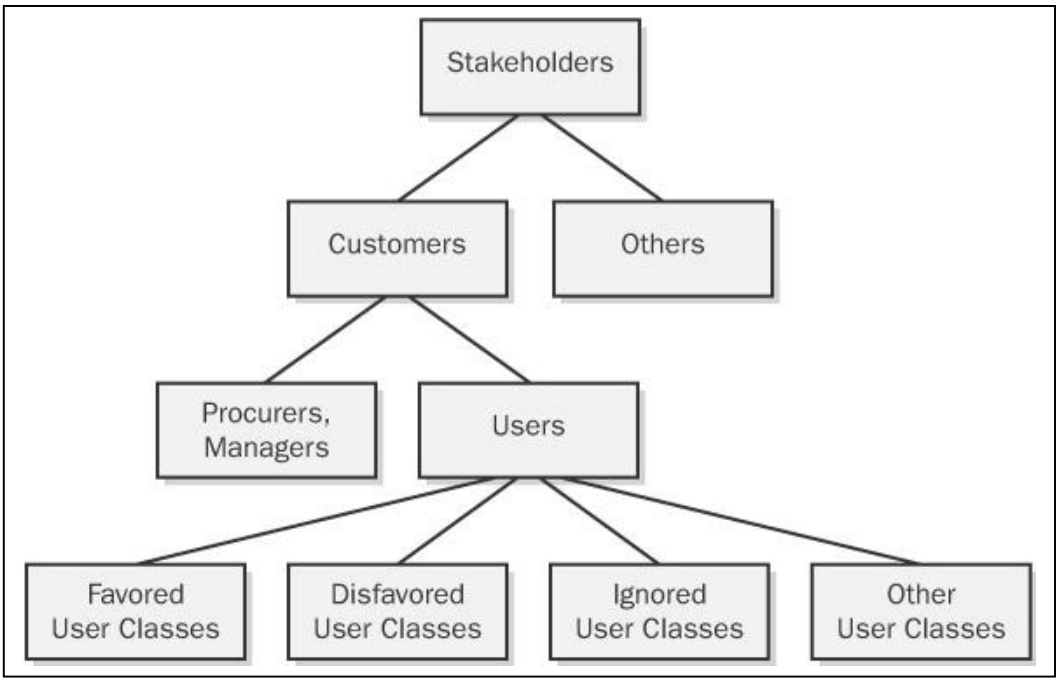
2.2.2.2 Clases de Usuario

Los usuarios de un producto difieren, entre otras cosas, en los siguientes aspectos:

- La frecuencia con la que usan los productos.
- La experiencia de dominio de aplicación y sistemas computacionales.
- Las características que utilizan.
- Las tareas que llevan acabo a favor de sus procesos de negocio.
- Su privilegio de acceso o niveles de seguridad (tales como usuario ordinario, usuario invitado o administrador).

Un individuo puede pertenecer a múltiples clases de usuario. Una clase de usuario es un subconjunto de los usuarios de un producto, que es un subconjunto de los clientes de un producto, que es un subconjunto de sus stakeholders. Como se puede apreciar en la Figura 2-11.

Figura 2-11 Una jerarquía de Stakeholders, Clientes y Usuarios



Fuente: Karl E. Wiegers, 2003

Ciertas clases de usuarios son más importantes que otras. Las clases de usuario predilectas reciben un trato preferencial cuando se están tomando decisiones de prioridad o resolviendo conflictos entre los requerimientos recibidos de diferentes clases de usuario.

Cada clase de usuario tendrá su propio conjunto de requerimientos para las tareas que los miembros de las clases deben llevar a cabo. También podrían tener diferentes requerimientos no-funcionales, como la usabilidad.

Podría parecer extraño, pero las clases de usuario no necesitan ser seres humanos. Se pueden considerar otras aplicaciones o componentes de hardware con los que el sistema interactúa como clases de usuario adicional. Un sistema de inyección de combustible sería una clase de usuario para el software incrustado en el controlador de motor de un automóvil.

Es importante no pasar por alto una clase de usuario y tratar de tener una lista de aproximadamente 15 clases de usuarios.

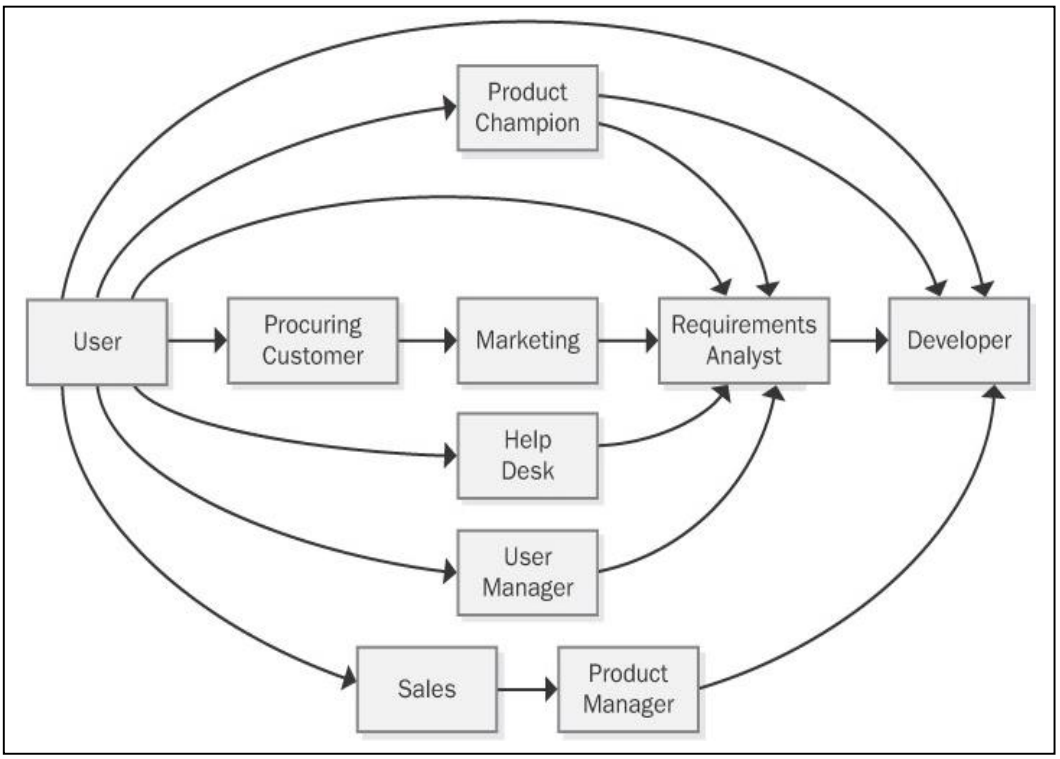
Documentar las clases de usuario y sus características, responsabilidades y ubicaciones físicas en el SRS.

2.2.2.3 Encontrar a Representantes de Usuario

Toda clase de proyecto necesita que representantes de usuario apropiados proporcionen la voz del cliente. Los representantes de usuario deben estar involucrados en el ciclo vida de desarrollo.

Algunos posibles caminos de comunicación entre los usuarios y los desarrolladores. Como se puede apreciar en la Figura 2-12.

Figura 2-12 Posibles caminos de comunicación entre los Usuarios y los Desarrolladores



Fuente: Karl E. Wiegers, 2003

2.2.2.4 El "Mejor Producto" (Product Champion)

Cada "mejor producto" sirve de interfaz principal entre miembros de una sola clase de usuario y el analista de requerimientos del proyecto. El mejor producto colecciona requerimientos de otros miembros de las clases de usuario que representan y concilian contradicciones. El desarrollo de requerimientos es por lo tanto, una responsabilidad compartida de los analistas y los clientes seleccionados, aunque el analista escribe documentos de requerimientos.

2.2.2.4.1 Expectativas del "Mejor Producto"

Algunas actividades que el mejor producto podría llevar acabo. Como se puede apreciar en la Tabla 2-5.

Tabla 2-5 Actividades Posibles del Champion de Producto

Categoría	Actividades
Planeación	Refinar el alcance y las limitaciones del producto Definir interfaces para otros sistemas Evaluar el impacto del nuevo sistema sobre las operaciones de negocio Definir un camino de transición de aplicaciones actuales
Requerimientos	Coleccionar requerimientos de otros usuarios Desarrollar escenarios de uso y casos de uso Resolver conflictos entre los requerimientos propuestos Definir prioridades de implementación Especificar la calidad y el rendimiento de los requerimientos Evaluar los prototipos de interfaz de usuario
Validación y Verificación	Inspeccionar los documentos de requerimientos Definir los criterios de aprobación de los usuarios Desarrollar casos de prueba y escenarios de uso Proveer conjuntos de datos de prueba Llevar acabo una prueba beta
Usuario Ayuda	Escribir porciones de manuales de usuario y texto de ayuda Prepara materiales de entrenamiento para tutoriales Presentar demostraciones de producto a semejantes
Cambio de Administración	Evaluar y priorizar la correcciones de defecto Evaluar el impacto de los cambios de requerimientos propuestos en los usuarios y procesos de negocio Participar en realizar cambio de decisiones

2.2.2.4.2 Múltiples Champions del Producto

Una persona puede difícilmente describir las necesidades de todos los usuarios de una aplicación. El administrador de proyecto puede colocar un equipo de analistas y de champions del producto para coleccionar requerimientos correctos de fuentes correctas.

2.2.2.5 ¿Quién Toma las Decisiones?

Alguien debe resolver los requerimientos en conflicto de los diferentes tipos de usuario, armonizar inconsistencias, y arbitrar las cuestiones de alcance que surgen. A comienzos del proyecto, determinar quién tomará las decisiones para los requerimientos en desacuerdo.

Las decisiones deben ser tomadas desde abajo en la jerarquía organizacional por personas que están cerca de los desacuerdos y bien informadas de estos.

2.2.3 Escuchar la Voz del Cliente

El corazón de la ingeniería de requerimientos es la *licitación*, el proceso de identificar las necesidades y restricciones de los stakeholders para un sistema de software. La licitación se concentra en descubrir los requerimientos de usuario, el nivel medio de la tríada de los requerimientos de software (requerimientos de negocio, requerimientos de usuario y requerimientos funcionales). Los requerimientos de usuario abarcan las tareas que los usuarios tienen que lograr con el sistema y las expectativas de rendimiento, usabilidad, y otros atributos de calidad de los usuarios.

Planear las actividades de licitación de requerimientos del proyecto incrementa la oportunidad de éxito y coloca las expectativas objetivas para los stakeholders. El plan debe abordar los siguientes ítems:

- Objetivos de licitación: como validar datos de mercado, explorar casos de uso, o desarrollar un conjunto detallado de los requerimientos funcionales para el sistema.

- Estrategias y procesos de licitación: por ejemplo, combinaciones de encuestas, talleres, visitas a clientes, entrevistas individuales, y otras técnicas.

- Productos de esfuerzos de licitación: quizás una lista de casos de uso, un SRS detallado, un análisis de los resultados de la encuesta.

- Calendario y recursos estimados: identificar tanto a desarrolladores como a clientes participantes en las actividades de licitación, junto con las estimaciones de esfuerzo y calendario requeridos.

- Riesgos de licitación: identificar los factores que podrían impedir la habilidad de terminar.

2.2.3.1 Licitación de Requerimientos

La licitación de requerimientos es quizá la más difícil, puede dar resultado solamente a través de una sociedad de colaboración entre clientes y el equipo de desarrolladores. El analista debe crear un ambiente propicio para una exploración minuciosa del producto que está especificado.

Tratar de traer a la luz cualquier suposición que los clientes podrían tener y resolver las suposiciones en conflicto. Leer entre líneas para identificar las

funciones o características que esperan los clientes están incluidas sin haberlas dicho explícitamente.

En vez de transcribir lo que los clientes dicen, un analista creativo sugiere ideas y alternativas a los usuarios durante la licitación. A veces los usuarios no se dan cuenta de las capacidades que los desarrolladores pueden proveer y se sorprenden cuado les sugieren la funcionalidad que hará el sistema especialmente útil. Cuando los usuarios no pueden expresar qué necesitan realmente, quizá el analista puede observar como trabajan y sugerir formas de automatizar bloques de trabajo.

Las entrevistas con individuos o grupos de usuarios potenciales son una fuente tradicional de introducir requerimientos.

Después de cada entrevista, documentar los ítems que el grupo discutió y pedir a los entrevistados que revisen la lista para hacer las correcciones. Una revisión temprana es esencial para el desarrollo de los requerimientos exitosos porque solamente esas personas que proporcionaron los requerimientos pueden juzgar si fueron capturados correctamente.

2.2.3.2 Talleres (Workshops) de Licitación

Los analistas de requerimientos frecuentemente promueven los talleres de licitación de requerimientos. Los talleres de grupo de colaboración son una técnica muy efectiva para conectar usuarios y desarrolladores (Keil and Carmel, 1995).

Los siguientes puntos son unos cuantos consejos para dirigir sesiones de licitación eficaces.

Establecer directrices. Los participantes deben coincidir en algunos principios operativos básicos para sus talleres (Gottesdiener 2002). Los ejemplos incluyen lo siguiente:

- Iniciar y terminar reuniones a tiempo.
- Regresar de breaks inmediatamente.
- Mantener solamente una conversación a la vez.
- Esperar que todos participen.
- Enfocar los comentarios y las críticas en los asuntos, no en los individuos.

Permanecer en el alcance. Usar el documento de visión y alcance para confirmar si los requerimientos de usuario propuestos se encuentran dentro del alcance del proyecto actual.

Discusiones de Timebox. El facilitador podría destinar un periodo de tiempo fijo a cada tema de discusión. La discusión podría necesitar ser terminada después,

pero el timeboxing ayuda evitar la trampa de pasar más tiempo de lo previsto en el primer tema y descuidar otros temas planeados.

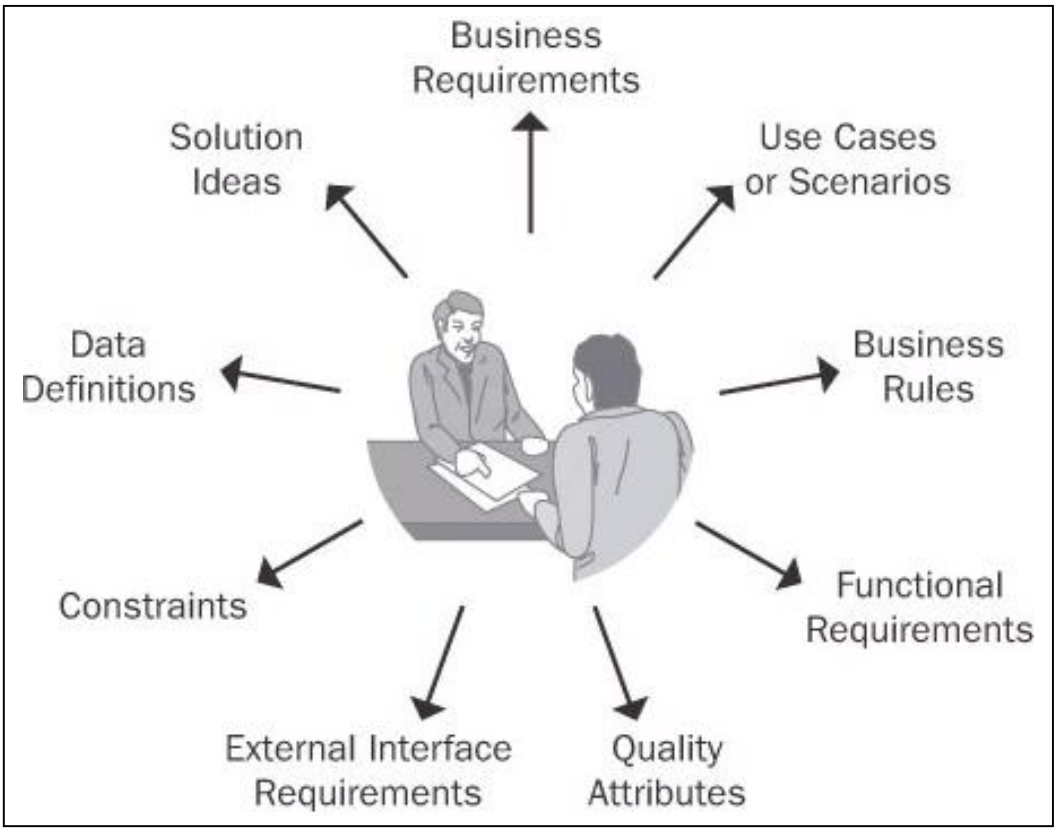
Mantener el equipo pequeño e incluir participantes auténticos. Los grupos pequeños pueden trabajar mucho más rápido que equipos más grandes. Los talleres de licitación con más de cinco o seis participantes activos pueden obstaculizar el trabajo de equipo, deben evitar conversaciones simultáneas, y pelear.

Mantener a todos comprometidos. El individuo podría llevar acabo una expectativa profunda que podría hacer una contribución importante.

2.2.3.3 Clasificando la Aportación del Cliente

Los analistas deben clasificar las partes innumerables de información de requerimientos que escuchan en varias categorías con el propósito de que puedan documentar y usar apropiadamente. Existen nueve categorías de requerimientos. Como se puede apreciar en la Figura 2-13.

Figura 2-13 Clasificar la Voz del Cliente



Fuente: Karl E. Wiegers, 2003

Requerimientos de Negocio. Algo que describe lo financiero, el mercado, u otro beneficio del negocio que cualquiera de los clientes u organización en desarrollo desea adquirir del producto es un requerimiento de negocio. Ejemplo:

- Incrementar la participación en el mercado un X%.

Casos de uso o escenarios. Statements generales de metas de usuario o tareas de negocio que necesitan los usuarios llevar acabo son casos de uso; un solo camino específico a través de un caso de uso es un escenario de uso. Ejemplo:

- Tengo que imprimir una etiqueta de envío para un paquete.

Reglas de negocio. Cuando un cliente dice que solamente ciertas clases de usuario pueden llevar a cabo una actividad bajo condiciones específicas, podría estar describiendo una regla de negocio.

Requerimientos Funcionales. Describen los comportamientos observables que el sistema presentará bajo ciertas condiciones y acciones que el sistema dejará a los usuarios hacer. Los requerimientos funcionales derivaban de requerimientos de sistema, requerimientos de usuario, reglas de negocio, y otras fuentes que hacen la mayor parte del SRS. Ejemplo:

- Si la presión sobrepasa 40.0 psi, la luz de advertencia de alta presión debe encenderse.

Atributos de Calidad. Estar atento a palabras que describen las características del sistema deseable: rápido, fácil, intuitivo, amigable con el usuario, robusto, confiable, seguro, y eficiente. Se tendrá que trabajar con los usuarios para comprender precisamente qué representan estos términos ambiguos y subjetivos y escribir metas de calidad comprobables.

Requerimientos de interface externos. Los requerimientos en esta clase describen las conexiones entre el sistema y el resto del universo. El SRS debe incluir secciones para interfaces de usuarios, hardware, y otros sistemas de software. Ejemplo:

- Debe leer señales de <algún dispositivo>

Restricciones. Restricciones de diseño e implementación que restringen las opciones disponibles para el desarrollador. Las restricciones innecesarias inhiben crear la mejor solución. Las restricciones también reducen la habilidad de usar componentes comercialmente disponibles como parte de la solución. Ciertas restricciones pueden ayudar a conseguir metas de atributo de calidad. Ejemplo:

- Los archivos enviados electrónicamente no podrían exceder los 10 MB en tamaño.

Definiciones de datos. Siempre que los clientes describan el formato, tipo de dato, permitir valores, o valor por default, representan una definición de datos.

Ideas de solución. Gran parte de lo que los usuarios representan como requerimientos cabe en la categoría de ideas de solución. Alguien que describe una manera específica de interactuar con el sistema para llevar a cabo una acción está presentando una solución. Por ejemplo un usuario dice, "Luego selecciono el estado donde quiero enviar el paquete de una lista desplegable." La frase de una lista desplegable indica que esto es una idea de solución.

2.2.3.4 Algunas Advertencias Sobre Licitación

Tratar de fusionar requerimientos ingresados de docenas de usuarios es difícil si no se usa un plan organizado estructurado, como casos de uso. Coleccionar aportaciones de muy pocos representantes o escuchar la voz solamente del cliente más fuerte y más dogmático es también un problema. El mejor balance involucra algunos champions del producto que tienen autoridad de hablar en nombre de sus clases de usuario respectivas.

Durante la licitación de requerimientos, se podría descubrir que el alcance del proyecto está definido incorrectamente, siendo demasiado grande o demasiado pequeño (Christel and Kang, 1992). Extraer requerimientos de usuarios por consiguiente puede llevar a modificar la visión del producto y el alcance del proyecto.

2.2.3.5 Encontrar Requerimientos Faltantes

Las siguientes técnicas ayudarán a detectar requerimientos anteriormente no descubiertos:

- Descomponer a detalle requerimientos de alto nivel para revelar exactamente lo que se está pidiendo.
- Asegurarse de que todas las clases de usuario hayan proveído aportaciones.
- Rastrear los requerimientos de sistema, casos de uso, listas de evento-respuesta, y reglas de negocio dentro los requerimientos funcionales detallados para asegurarse de que el analista ha derivado toda la funcionalidad necesaria.
- Examinar los valores límite en busca de requerimientos faltantes.
- Representar información de requerimientos de formas múltiples (modelos de análisis).

- Usar tablas de decisión o árboles de decisión, que son una opción para asegurar que se cubran todos los problemas de lógica compleja, cubriendo todas las situaciones posibles, esto es; todo el conjunto de requerimientos con operaciones lógicas complejas (ANDs, ORs, y NOTs) que a menudo están incompletos.

Una forma rigurosa de buscar requerimientos faltantes es crear una matriz de CRUD (Create, Read, Update, and Delete), que significa crear, leer, actualizar y eliminar. Una matriz de CRUD correlaciona acciones de sistema con entidades de datos para asegurar que se sabe dónde y cómo se ha creado, leído, actualizado y eliminado cada ítem de dato. Por ejemplo:

- Entidades de datos y eventos de sistema (Robertson y Robertson, 1999)
- Entidades de datos y tareas de usuario o casos de uso (Lauesen, 2002)
- Clases de objeto y eventos de sistema (Ferdinandi, 2002)
- Clases de objeto y casos de uso (Armour y Millar, 2001)

2.2.3.6 ¿Cómo Saber Cuando Se Está Listo?

No hay una señal simple que indique cuando se ha terminado la licitación de requerimientos. Nunca se estará totalmente listo, pero los siguientes ejemplos indican cuando se ha llegado al punto máximo en la licitación de requerimientos:

Si los usuarios no pueden pensar en ningún otro caso de uso.

Si los usuarios proponen nuevos casos de uso pero ya no se obtienen requerimientos funcionales asociados con otros casos de uso.

Si los usuarios repiten los asuntos que ya cubrieron en las discusiones previas.

Si las nuevas características sugeridas de los requerimientos de usuario, o los requerimientos funcionales están fuera del alcance.

Si los nuevos requerimientos propuestos son de baja prioridad.

Si los usuarios están proponiendo capacidades que pueden ser incluidas "Algún día en la vida del producto".

2.2.4 Entendiendo los Requerimientos de Usuario

Por muchos años, los analistas han empleado el uso de escenarios para licitar requerimientos de usuario. (McGraw y Harbison, 1997). Un escenario es una descripción de un solo uso del sistema. Ivar Jacobson y colegas (1992), han formalizado la perspectiva de uso centrada en el acercamiento de casos de uso para la licitación y modelado de requerimientos. Los casos de uso trabajan bien

para desarrollar requerimientos para aplicaciones de negocio, sitios web, servicios que un sistema provee a otro, y sistemas que permiten tener el control de un hardware.

2.2.4.1 Acercamiento a los Casos de Uso

Los casos de uso describen secuencias de interacciones entre un sistema y actores externos. Un actor es una persona, otro sistema de software, dispositivo de hardware que interactúa con el sistema para alcanzar una meta útil. Otro nombre para los actores es rol de usuario, por que los actores son roles que los miembros de una o más clases de usuario pueden realizar con respecto al sistema.

Los casos de uso emergen del desarrollo orientado a objetos. El objetivo de los caos de uso es describir todas las tareas que los usuarios necesiten realizar con el sistema. Los diagramas de casos de uso proporcionan una representación de alto nivel de los requerimientos de usuario.

2.2.4.1.1 Casos de Uso y Uso de Escenarios

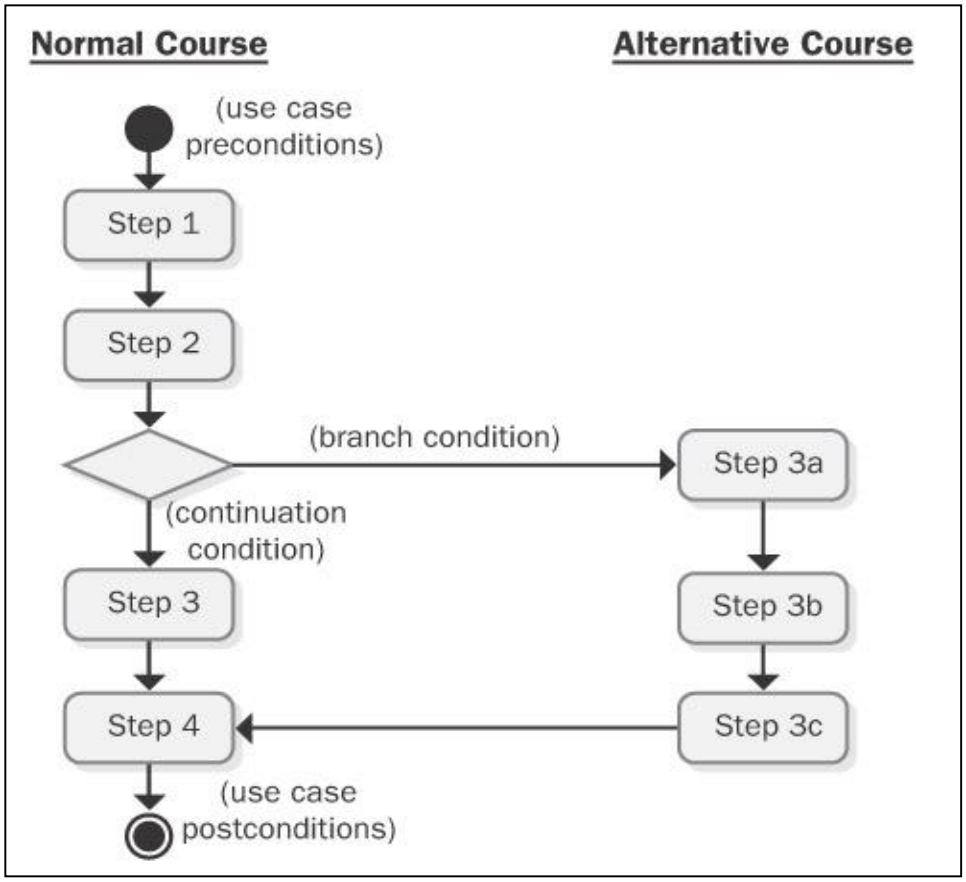
Un caso de uso es discreto a una actividad independiente que un actor puede realizar para alcanzar un cierto valor en el resultado, Un solo caso de uso puede tener un número de tareas similares que tienen una meta en común. Un caso de uso es por lo tanto una colección del uso de escenarios relacionados y un escenario es una instancia específica de un caso de uso. Cuando se exploran requerimientos de usuario, se puede empezar por la abstracción de casos de uso y desarrollo concreto de escenarios. Los elementos esenciales para la descripción de los casos de uso son los siguientes:

- Un identificador único.
- Nombre de la tarea, verbo.
- Una pequeña descripción textual en lenguaje natural.
- Una lista de precondiciones que son hechas antes que el caso de uso pueda comenzar.
- Una lista de poscondiciones que describe el estado del sistema después de que el caso de uso se termine con éxito.
- Una lista de pasos que muestran la secuencia y la interacción entre el actor y el sistema que conduce a las poscondiciones y precondiciones.

Un escenario se identifica como el curso normal de eventos para el caso de uso. Otros escenarios validos dentro de un caso de uso son descritos como cursos alternativos o escenarios secundarios. Cursos alternativos también dan lugar a la terminación aceptada de las tareas y satisfacen las poscondiciones.

Un diagrama de actividad en UML es una manera de visualizar el flujo lógico de complejos casos de uso. La Figura 2-14 muestra un ejemplo de estos diagramas.

Figura 2-14 Diagrama de Actividad en UML



Fuente: Karl E. Wiegers, 2003

2.2.4.1.2 Identificando Casos de Uso

Se puede identificar casos de uso de diferentes formas (Hamm 1998; Larman 1998):

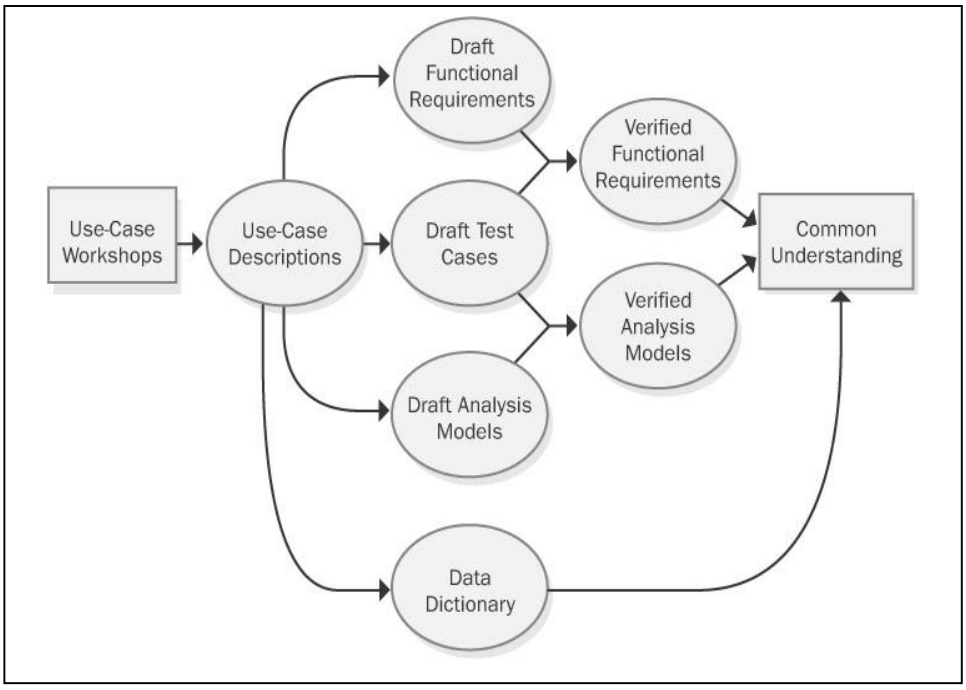
- Primero identificar actores, y en seguida procesos de negocio en los que cada uno participa.
- Identificar eventos externos a los cuales el sistema debe responder, y después relacionar estos acontecimientos con los actores que participan y casos de uso específicos.
- Expresar los procesos de negocio en términos de escenarios específicos, generalizando estos escenarios dentro de casos de uso y identificando los actores involucrados en cada caso de uso.

– Derivar los casos de uso, de existentes requerimientos funcionales.

2.2.4.1.3 Documentando Casos de Uso

Constantine y Lockwood (1999) definen un caso de uso como: "...una descripción simplificada, generalizada, abstracta, de una tecnología libre e implementación independiente de una tarea o una interacción." Es decir, el enfoque debe estar sobre la meta que el usuario esta tratando lograr. La documentación de casos de uso es importante ya que guía paso a paso a todas las variables que intervienen en el caso de uso. La Figura 2-15 muestra la secuencia de eventos para desarrollar casos de uso.

Figura 2-15 Enfoque de la Licitación de Casos de Uso



Fuente: Karl E. Wiegers, 2003

2.2.4.1.4 Casos de Uso y Requerimientos Funcionales

Un requerimiento funcional se puede documentar asociado a los casos de uso de diferentes maneras. El acercamiento que usted elija depende de lo que se espera que se haga, que el equipo realice el diseño, construcción, testing de los documentos de casos de uso, del SRS, o combinación de ambos. Ninguno de los siguientes métodos son perfectos, así que el acercamiento se debe acoplar a

las necesidades de su negocio y como se desee documentar y administrar los proyectos en cuanto a requerimientos de software.

2.2.4.1.4.1 Únicamente Casos de Uso

Una posibilidad es incluir los requerimientos funcionales a la derecha en cada descripción de los casos de uso. Se necesita tener por separado una especificación en la que se describa los requerimientos no funcionales y cualquier requerimiento funcional que no esta asociado con un caso de uso especifico. Varios casos de uso podrían necesitar el mismo requerimiento funcional.

2.2.4.1.4.2 Casos de Uso y SRS

Otra opción es describir de manera simple los casos de uso y documentar los requerimientos funcionales derivados de un caso de uso en un SRS. En esta opción, se necesita establecer un rastro entre los casos de uso y su asociación con los requerimientos funcionales. La mejor manera de administrar el seguimiento de los requerimientos es almacenando todos los casos de uso y requerimientos funcionales en una herramienta de administración de requerimientos.

2.2.4.1.4.3 Únicamente SRS

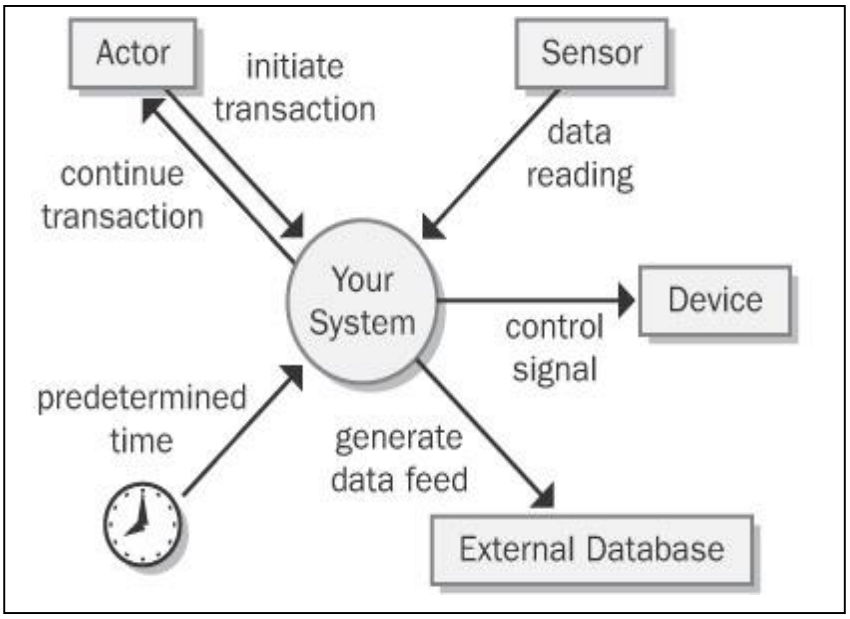
El tercer acercamiento es organizar el SRS por casos de uso o por características incluyendo los casos de uso y los requerimientos funcionales en el SRS. Este esquema no utiliza documentos de casos de uso por separado.

2.2.4.2 Tablas de Evento-Respuesta

Otra forma de organizar y documentar requerimientos de usuario es identificando eventos externos por los cuales el sistema responda. Un evento es un cambio o actividad que toma lugar en un ambiente de usuario que estimula una respuesta al sistema de software. Una tabla de evento-respuesta enumera todos los posibles eventos y el comportamiento que se espera del sistema para cada evento. Hay varios tipos de eventos del sistema, como lo muestra la Figura 2-16 y descritos a continuación.

- Una acción por un usuario humano que estimula un diálogo con el software.
- Una señal de control, una lectura de los datos, o una interrupción recibida de un dispositivo de hardware externo.
- Un acontecimiento tiempo-accionado.

Figura 2-16 Ejemplo de un Sistema Evento-Respuesta



Fuente: Karl E. Wiegers, 2003

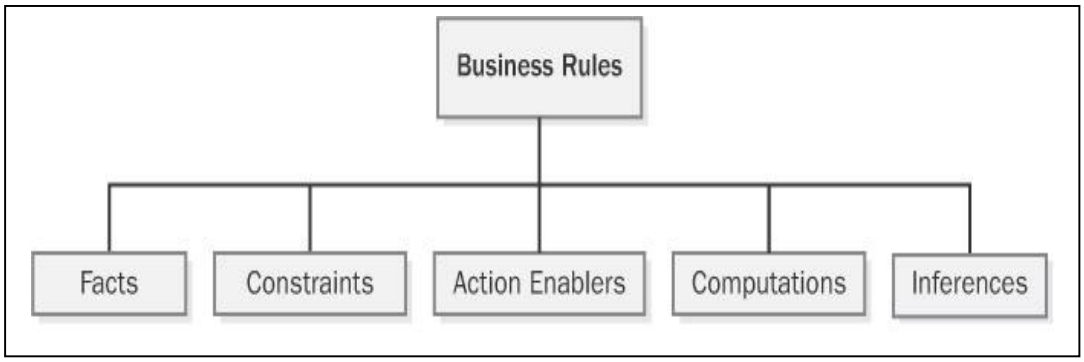
2.2.5 Jugando con las Reglas

2.2.5.1 Reglas de Negocio

Según el Business Rules Grup (1993), "una regla de negocio en una declaración definición u obligación del algunos aspectos del negocio. Estas se realizan para afirmar la estructura del negocio, controlar o influenciar el comportamiento del negocio." Se han desarrollado metodologías para el descubrimiento y documentación de reglas de negocio así como también la implementación automatizada de estas. Identificar las reglas que pertenecen al sistema a desarrollar es importante para ligarlos a los requerimientos funcionales específicos.

Diferentes esquemas de clasificación se han propuesto para las reglas de negocio de una organización. Un esquema simple se muestra en la Figura 2-17, con diferentes tipos de reglas de negocio, que trabajan para diferentes situaciones.

Figura 2-17 Clasificación de las Reglas de Negocio



Fuente: Karl E. Wiegers, 2003

2.2.5.1.1 Hechos

Los hechos son simples declaraciones verdaderas de los negocios. A menudo describen la asociación y relación entre los diferentes tipos de negocio. Muchas veces se traducen directamente en requerimientos funcionales.

2.2.5.1.2 Restricciones

Las restricciones bloquean las acciones que el sistema o los usuarios pueden realizar. Algunas palabras o frases que se sugieren describen reglas de negocio de restricción como, *se necesita, no se necesita, no se puede y únicamente.*

2.2.5.1.3 Acciones Permitidas

Es una regla que acciona cierta actividad bajo específicas condiciones, esta es una acción permitida. La regla conduce a especificar una cierta funcionalidad del software que es expuesta cuando algunas condiciones son verdaderas.

2.2.5.1.4 Interferencias

A veces llamado conocimiento inferido, una inferencia es una regla que establece algunos nuevos conocimientos basados en la verdad de ciertas condiciones. Una inferencia crea un nuevo hecho de otros hechos. Las inferencias son escritas a menudo con el patrón “if/then”.

2.2.5.1.5 Cálculos

Las computadoras calculan, entonces que una clase de reglas de negocio definen los cálculos que se realizan usando formulas o algoritmos matemáticos

específicos. Muchos cálculos siguen reglas que son externas al desarrollo, tales como impuestos.

2.2.5.2 Documentando reglas de negocio

Ya que las reglas de negocio pueden influir en múltiples aplicaciones, las organizaciones deben administrar sus reglas de negocio a nivel de empresa. Un simple catalogo de reglas de negocio es suficiente inicialmente. Organizaciones grandes con operaciones de negocio y sistemas de información pesados establecen sus reglas de negocio en bases de datos.

Como se muestra en la Tabla 2-6, dar a cada regla de negocio un identificador único que le permita dar un seguimiento para los requerimientos funcionales de una específica regla de negocio. En la columna “tipo de regla” identifica que tipo de regla es. La columna “estática o dinámica”, como la regla cambiará en el tiempo. En la columna “fuente” se refiere al origen de la regla, estas pueden ser corporativas, de gobierno, etc.

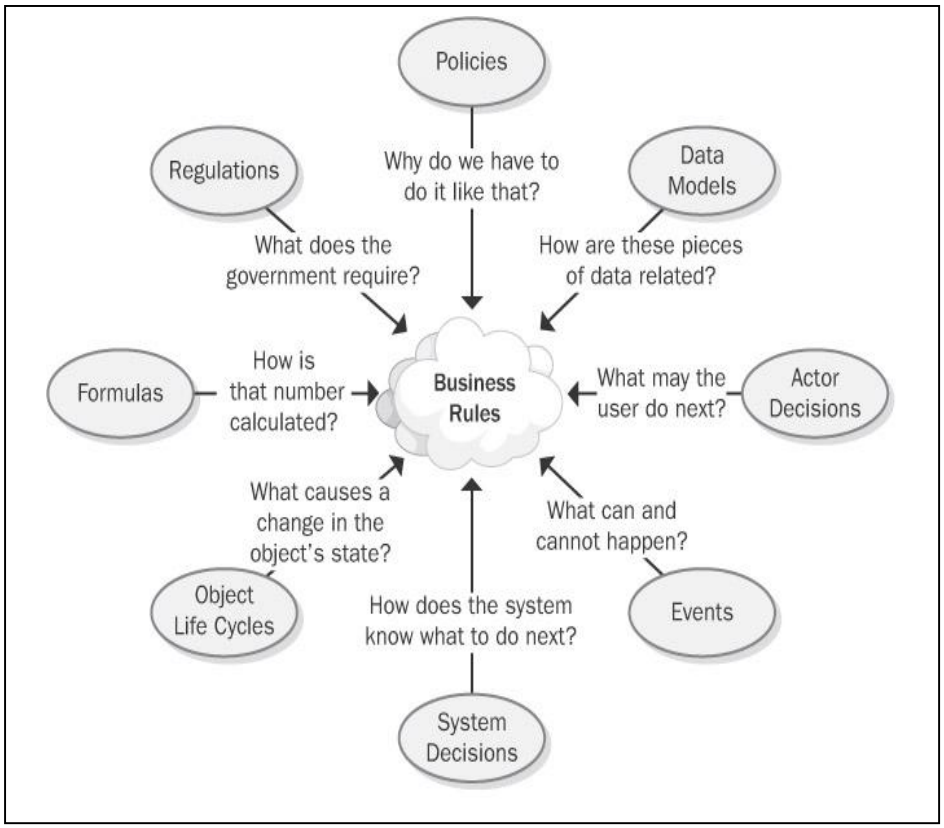
Tabla 2-6 Ejemplo de un Catalogo de Reglas de Negocio

ID	Definición de Regla	Tipo de Regla	Estática o Dinámica	Fuente
ORDER-15	El usuario puede solicitar producto químico de la lista solo cuando a tenido un entrenamiento en los últimos 12 meses.	Restricción	Estática	Política Corporativa
ORDER-13	Se puede tener un descuento en la orden según lo basado en la tabla de descuentos.	Calculo	Dinámica	Política Corporativa

2.2.5.3 Reglas de Negocio y Requerimientos

Durante los workshops de licitación de requerimientos de usuario, los analistas deben hacer `preguntas sobre las restricciones y requerimientos que los usuarios presentan. Antes de identificar y documentar reglas de negocio, determinar cuales se van a implementar en el software. Algunas reglas conducirán a utilizar casos de uso y por lo tanto los requerimientos funcionales probarán que esta regla se cumple. La Figura 2-18 muestra varios orígenes de potenciales reglas de negocio.

Figura 2-18 Descubriendo Reglas de Negocio desde Diferentes Perspectivas



Fuente: Karl E. Wiegers, 2003

Para definir una liga entre los requerimientos funcionales y los patrones de reglas de negocio existen dos formas:

- Usar un atributo de requerimiento llamado “Origen” e indica las reglas como el origen del requerimiento funcional.

- Define las ligas entre los requerimientos funcionales y los patrones de reglas de negocio, en una matriz de requerimientos rastreable.

2.2.6 Documentar los Requerimientos

El resultado del desarrollo de los requerimientos es un documento de conformidad entre clientes y desarrolladores sobre el producto que se está construyendo. El documento de visión y alcance contienen los requerimientos de negocio, y los requerimientos de usuario a menudo son capturados en forma de casos de uso. Los requerimientos funcionales y no-funcionales detallados residen en la especificación de requerimientos de software (SRS).

2.2.6.1 Especificación de Requerimientos de Software

La Especificación de Requerimientos de Software es llamada a veces *especificación funcional*, *especificación de producto*, *documento de requerimientos*, o *especificación de sistema*, aunque organizaciones no usan estos términos del mismo modo. El SRS dice precisamente las funciones y las capacidades que un sistema de software debe proveer y las restricciones que debe respetar. El SRS es la base para toda la planeación del proyecto, diseño, y codificación subsiguiente, tanto como el cimiento para las pruebas del sistema y documentación de usuario. Debe describir los comportamientos del sistema tan completamente como sea necesario bajo varias condiciones. No debe contener el diseño, la construcción, las pruebas, o los detalles de administración del proyecto aparte del diseño y las restricciones de implementación.

Algunas audiencias dependen del SRS:

- Los clientes, el departamento de marketing, y el personal de ventas necesitan saber que producto pueden esperar para ser distribuido.
- Los administradores de proyecto basan sus estimaciones de calendario, esfuerzo, y recursos en la descripción del producto.
- El SRS dice al equipo de desarrollo de software qué construir.
- El grupo de pruebas usa el SRS para desarrollar planes de pruebas, casos de prueba, y procedimientos.
- El SRS dice al personal de mantenimiento y soporte lo que supuestamente hace cada parte del producto.
- Los autores de documentación basan los manuales de usuario y pantallas de ayuda en el SRS y el diseño de interfaz de usuario.
- El personal de entrenamiento usa el SRS y la documentación de usuario para elaborar materiales educativos.
- El personal legal asegura que los requerimientos obedecen leyes y reglas.
- Los subcontratistas basan su trabajo en el SRS, y pueden ser defendidos legalmente con el SRS.

Los desarrolladores y clientes deben no hacer suposiciones. Si cualquier capacidad o cualidad no aparece en alguna parte en el acuerdo de requerimientos, nadie debe esperar que ésta salga en el producto.

Organizar y escribir el SRS con el propósito de que los stakeholders puedan entenderlo.

2.2.6.1.1 Etiquetar los Requerimientos

Para satisfacer el criterio de calidad de rastreabilidad y modificación, cada requerimiento funcional debe ser excepcionalmente y persistentemente identificado. Esto permite hacer referencia a requerimientos específicos en una solicitud de cambio, historia de modificación, referencia cruzada, o matriz de rastreabilidad de requerimientos. También permite re-usar los requerimientos en proyectos múltiples.

Número de Secuencia
La propuesta más simple es dar a cada requerimiento un número de secuencia único, como UR-9 o SRS-43. Las herramientas comerciales de administración de requerimientos asignan tal identificador cuando un usuario añade un nuevo requerimiento a la base de datos de la herramienta. El prefijo muestra el tipo de requerimiento, como UR para el requerimiento de usuario. Un número no es re-usado si un requerimiento es eliminado.

Numeración Jerárquica
Si los requerimientos funcionales aparecen en la sección 3.2 del SRS, tendrán todos ellos etiquetas que inicien con 3.2 (por ejemplo 3.2.4.3). Más dígitos indican un requerimiento más detallado y de menor intensidad. Este método es simple y conciso. Además las etiquetas numéricas no le dicen nada sobre el propósito de cada requerimiento.

Etiquetas de Texto Jerárquicas
El consultor Tom Gilb sugiere un esquema de etiquetando jerárquico basado en texto para etiquetar requerimientos individuales (Gilb, 1988). Por ejemplo: “El sistema pedirá al usuario que confirme cualquier pedido al imprimir más de 10 copias”. Este requisito podría ser etiquetado Imprimir.ConfirmCopies. Las etiquetas de texto jerárquicas son estructuradas, significativas, e indiferentes a la adición, eliminación, o cambio de lugar de otros requerimientos. Su desventaja es que son más voluminosas que las etiquetas numéricas, pero ése es un precio pequeño a pagar por etiquetas estables.

2.2.6.1.2 Interfaces de Usuario y el SRS

Incluir los diseños de interfaz en el SRS tiene tanto desventajas como ventajas. En el lado negativo, las imágenes de pantalla y arquitecturas de interfaz de usuario describen soluciones (diseños), no requerimientos.

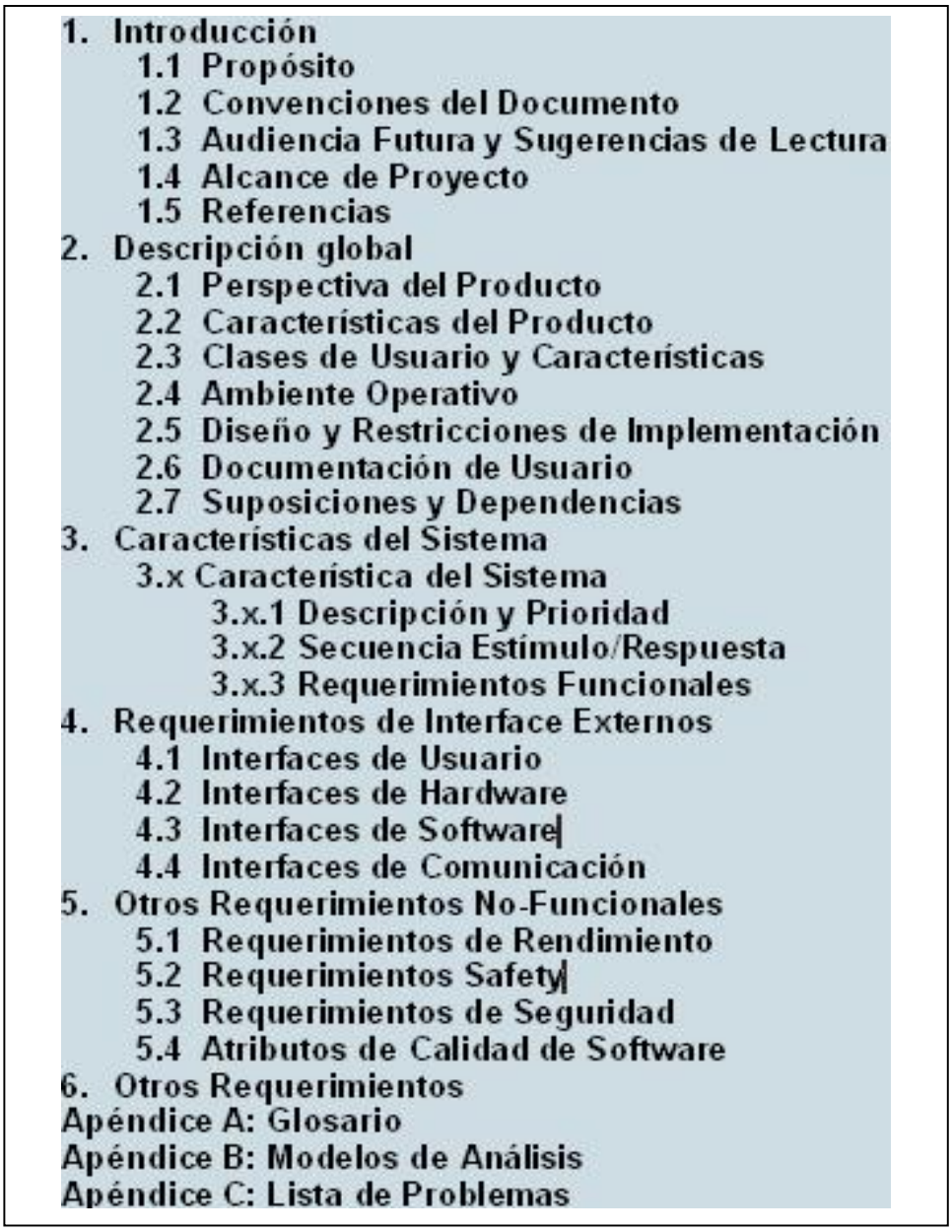
En el lado positivo, analizar las posibles interfaces de usuario (tal como con un prototipo básico) hace tangibles los requerimientos tanto a usuarios como a desarrolladores.

2.2.6.2 Template de Especificación de Requerimientos de Software

Cada organización de desarrollo de software debe adoptar uno o más templates del SRS estándar para sus proyectos.

El template del SRS fue adaptado del estándar de la IEEE 830. Como se puede apreciar en la Figura 2-19.

Figura 2-19 Template para la Especificación de Requerimientos de Software



Fuente: Karl E. Wiegers, 2003

1. Introducción

La introducción representa una visión general para ayudar al lector a entender cómo esta organizado y cómo se usa el SRS.

1.1 Propósito

Identifica el producto o la aplicación cuyos requerimientos son especificados en este documento, incluyendo la versión. Si el SRS está relacionado únicamente con una parte o con el sistema entero, identificar esa parte o subsistema.

1.2 Convenciones del Documento

Describir cualquier estándar o tipografía, incluyendo estilos de texto, resaltos, o notas importantes.

1.3 Audiencia Futura y Sugerencias de Lectura

Hacer una lista de los diferentes lectores a quién esta dirigido el SRS. Describir lo que el resto del SRS contiene y cómo está organizado. Indicar una secuencia de lectura para cada tipo de lector del documento.

1.4 Alcance de Proyecto

Explicar que hará y que no hará el producto de software. Describir la aplicación de software, así como, especificar los beneficios pertinentes, objetivos y metas.

1.5 Referencias

Proporcionar una lista de los documentos y recursos a los que el SRS hace referencia, incluyendo hipervínculos si es posible. Proveer la información suficiente con el propósito de que el lector pueda acceder a cada referencia, incluyendo titulo, autor, número de versión, fecha, y fuente o ubicación.

2. Descripción global

Esta sección presenta una visión general de alto nivel del producto y el ambiente en el que será usado, los usuarios de producto esperados, y restricciones conocidas, suposiciones, y dependencias.

2.1 Perspectiva del Producto

Describir el contexto y el origen del producto. Si el SRS define un componente de un sistema más grande, decir cómo este software se relaciona con el sistema global e identificar interfaces muy importantes entre lo dos.

2.2 Características del Producto

Proporcionar una lista de las características más importantes que el producto contiene o las funciones importantes que lleva acabo. Los detalles serán proveídos en la Sección 3 del SRS, así que solamente se necesita un resumen de alto nivel aquí. Una imagen de los grupos de requerimientos importantes y cómo están relacionados, tal como un diagrama de flujo de datos, un diagrama de caso de uso, o un diagrama de clase, podría ser útil.

2.3 Clases de Usuario y Características
Identificar las clases de usuario para el producto y describir las características pertinentes.

2.4 Ambiente Operativo
Describir el ambiente en el que el software funcionará, incluyendo la plataforma de hardware, los sistemas operativos y las versiones, y las ubicaciones geográficas de usuarios, servidores, y bases de datos. Proporcionar una lista de los componentes de software o aplicaciones con los que el sistema debe coexistir pacíficamente.

2.5 Diseño y Restricciones de Implementación
Describir cualquier factor que restringiría las opciones disponibles para los desarrolladores y la razón fundamental para cada restricción.

2.6 Documentación de Usuario
Proporcionar una lista de la documentación de usuario que serán repartidos al mismo tiempo que el software ejecutable. Podría incluir manuales de usuario, ayuda en línea, y tutoriales.

2.7 Suposiciones y Dependencias
Una suposición es una afirmación que es creída verdadera a falta de pruebas o conocimientos definitivos. Los problemas pueden surgir si las suposiciones son incorrectas, no son compartidas, o cambian, así que las suposiciones se traducirán a riesgos de proyecto.

Identificar cualquier dependencia que el proyecto tenga sobre factores externos fuera de su control. Si estas dependencias ya están documentadas en otro lugar, tal como en el plan de proyecto, referir esos documentos aquí.

3. Características del Sistema
El template en la Figura 2-19 esta organizado por la característica del sistema, que es sólo una manera posible de organizar los requerimientos funcionales.

3.x Característica del Sistema
Decir el nombre de la característica en sólo unas pocas palabras, como “3.1 Pasar el corrector ortográfico”.

3.x.1 Descripción y Prioridad
Proveer una descripción breve de las características y demostrar si es de prioridad alta, mediana, o baja.

3.x.2 Secuencia Estímulo/Respuesta
Listar las secuencias de estímulo de entrada (acciones de usuario, señales de dispositivos externos, u otros disparadores) y las respuestas de sistema que definen los comportamientos para estas características.

3.x.3 Requerimientos Funcionales
Hacer una lista de los requerimientos funcionales detallados relacionados con esta característica. Describir cómo el producto debe responder a las condiciones de error previas y a entradas y acciones inválidas.

4. Requerimientos de Interface Externos
De acuerdo con Richard Thayer (2002) “Los requerimientos de interface externos especifican el hardware, software, o la base de datos con que un sistema o componente deben comunicarse…".

Colocar las descripciones detalladas de los datos y los componentes de control de las interfaces en el diccionario de datos.

4.1 Interfaces de Usuario
Describir las características lógicas de cada interfaz de usuario que el sistema necesita. Esto incluye características de configuración necesarias para lograr los requerimientos de software y todos los aspectos para perfeccionar la interfaz con la persona que debe usar el sistema.

4.2 Interfaces de Hardware
Describir las características de cada interface entre el software y componentes de hardware del sistema. Esta descripción podría incluir los tipos de dispositivo soportados, los datos y las interacciones de control entre el software y hardware, y los protocolos de comunicación.

4.3 Interfaces de Software
Describir las conexiones entre este producto y otros componentes de software (identificado por el nombre y la versión), incluyendo bases de datos, sistemas operativos, herramientas, etc. Decir el propósito de los mensajes, datos.

4.4 Interfaces de Comunicación
Decir los requerimientos para cualquiera de las funciones de comunicación que el producto usará, incluyendo formas electrónicas. Definir cualquier formateado de mensaje pertinente. Especificar la seguridad de comunicación o asuntos de encriptación, velocidades de transferencia de datos, y mecanismos de sincronización.

5. Otros Requerimientos No-Funcionales
Esta sección específica requerimientos no-funcionales aparte de los requerimientos de interface externos, de la sección 4, y las restricciones, de la sección 2.5.

5.1 Requerimientos de Rendimiento

Decir requerimientos de rendimiento específicos para las varias operaciones de sistema. Explicar su razón fundamental para guiar a los desarrolladores a tomar las elecciones de diseño apropiadas.

5.2 Requerimientos Safety
Especificar los requerimientos que están involucrados con posible pérdida, o daño. Definir cualquier garantía o acción que debe ser tomada, tanto como acciones potencialmente peligrosas que deben ser prevenidas.

5.3 Requerimientos de Seguridad
Especificar cualquier requerimiento respecto a la seguridad, integridad, o asuntos de privacidad que afectan el acceso al producto, el uso del producto, y protección de los datos que el producto usa o crea.

5.4 Atributos de Calidad de Software
Decir cualquier característica de calidad de producto adicional que será importante para los clientes o desarrolladores. Estas características deben ser específicas, cuantitativas, y comprobables.

6. Otros Requerimientos
Definir cualquier otro requerimiento que no está cubierto en otro lugar del SRS. Los ejemplos incluyen requerimientos de internacionalización y requerimiento legales.

Apéndice A: Glosario
Definir cualquier término especializado que un lector tiene que saber para interpretar el SRS apropiadamente, incluyendo siglas y abreviaturas. Deletrear cada sigla y proveer su definición.

Apéndice B: Modelos de Análisis
Esta sección opcional incluye o señala modelos de análisis pertinentes como diagramas de flujo de datos, diagramas de clase, diagramas de transición de estado, o diagramas de entidad - relación.

Apéndice C: Lista de Problemas
Esta es una lista dinámica de los requerimientos que quedan para ser resueltos.

2.2.6.3 Pautas para Escribir Requerimientos

Kovitz (1999) presenta muchas recomendaciones y ejemplos para escribir buenos requerimientos:

- Escribir oraciones completas que tengan gramática, ortografía, y pronunciación correcta. Mantener oraciones y párrafos cortos y directos.

- Usar la voz activa.
- Usar constantemente términos que han sido definidos en el glosario.
- Descomponer un requerimiento vago de alto nivel a detalle suficiente para aclararlo y quitar la ambigüedad.
- Expresar los requerimientos de un modo consistente, como “El sistema sí” o “El usuario sí”, seguido por un verbo de acción, seguido por un resultado observable.
- Usar listas, dibujos, gráficos, y tablas para presentar la información visualmente.
- Enfatizar las partes más importantes de la información.
- El lenguaje ambiguo conduce a requerimientos imposibles de demostrar, así que hay que evitar usar términos imprecisos y subjetivos.

Los requerimientos precisamente dichos incrementan la oportunidad de que las personas reciban lo que esperan.

2.2.7 Un Diagrama vale más que 1024 Palabras

2.2.7.1 Modelando los Requerimientos

El modelado visual de requerimientos incluye diagrama de flujo de datos (DFD), diagramas entidad-relación (DER), diagramas de transición de estado (DTE), mapas de dialogo, casos de uso, diagramas de clases y diagramas de actividad de UML. Estos modelos son usados para elaborar y explorar los requerimientos, así como para diseñar soluciones de software.

2.2.7.2 De la Voz del Cliente al Análisis del Modelado

Escuchar cuidadosamente como el cliente presenta sus requerimientos, el analista puede identificar palabras claves que se pueden traducir a elementos en modelos visuales específicos. La Tabla 2-7 sugiere el mapeo de los sustantivos y verbos que el cliente dice, a los diferentes modelos visuales.

Tabla 2-7 Relación entre la Voz del Cliente y el Análisis del Modelado

Tipo de Palabra	Ejemplo	Componentes Análisis del Modelo
Sustantivo	Gente, Organización, sistemas de software, datos u objetos que existen	Almacenes de datos (DFD) Actores (Diagrama de Casos de Uso)