CAPITULO 2
REQUERIMIENTOS DE SOFTWARE

ACRÓNIMOS

DAG	Grafo Acíclico Dirigido complejo
FSM	Medida Funcional del Tamaño
INCOSE	Consejo Internacional sobre la Ingeniería de Sistemas
SADT	Análisis Estructurados y Técnicas de Diseño

INTRODUCCIÓN

El área del conocimiento de los requisitos del software (KA) se refiere al análisis, a la especificación, y a la validación de los requisitos del software. Está extensamente reconocido dentro de la industria del software que los proyectos de la ingeniería de software son críticamente vulnerables cuando estas actividades se realizan mal.

Los requisitos del software expresan las necesidades y los apremios colocados en un producto de software que contribuye a la solución de un cierto problema del mundo real. [Kot00] El término "ingeniería de requisitos" es ampliamente utilizado en el campo para denotar la dirección sistemática de requisitos. Aunque por razones de consistencia, este término no será utilizado en la guía, pues se ha decidido que el uso del término "ingeniería" para las actividades con excepción de la tecnología de dotación lógica debe ser evitado en esta edición de la guía.

Por la misma razón, tampoco se utilizará "ingeniero de los requisitos," un término que aparece en algo de la literatura. En su lugar se utilizará el término "Ingeniero de Software" o, en algunos casos específicos, "especialista de los requisitos", el último donde el papel en la pregunta es realizado generalmente por un individuo con excepción de una Ingeniería de Software. Esto no implica, sin embargo, que una Ingeniería de Software no podría realizar la función.

La interrupción de KA es ampliamente compatible con secciones de IEEE 12207 que se refieren a actividades de requisitos. (IEEE12207.1-96)

Un riesgo inherente en la interrupción propuesta es que un proceso en cascada puede ser deducido. Para evitar esto, los procesos de requisitos de la subzona 2, se diseñan para proporcionar una descripción de alto nivel del proceso de los requisitos precisando los recursos y los apremios bajo los cuales el proceso funciona y los cuales actúan para configurarlo.

Una descomposición alternativa podía utilizar una estructura basada en producto (requisitos del sistema, requisitos del software, prototipos, casos de uso, y así sucesivamente). Las interrupciones basadas en procesos reflejan el hecho de que el proceso de los requisitos, si es acertado, se debe considerar como proceso que implica actividades complejas, firmemente unidas (ambas secuencial y concurrentemente), más que una única actividad discreta realizada al principio de un proyecto de desarrollo de software.

El KA de los requisitos del software se relaciona de cerca con el Diseño del software, pruebas, mantenimiento del software, gerencia de la configuración del software, tecnología de dotación lógica, proceso de la tecnología de dotación lógica, y KAs de Calidad de software.

INTERRUPCIÓN DE LOS ASUNTOS PARA LOS REQUISITOS DEL SOFTWARE

1. Fundamentos de los requisitos del software

1.1. Definición de un requisito del software

Básicamente, un requisito del software es una característica que se debe exhibir para solucionar un cierto problema en el del mundo real. La guía se refiere a requisitos de "software" porque se refiere a los problemas que se tratarán por el software. Por lo tanto, un requisito del software es una característica que se debe exhibir por el software desarrollado o adaptado para solucionar un problema particular. El problema puede ser automatizar la parte de una tarea de alguien que utilizará el software, para apoyar los procesos del negocio de la organización que ha comisionado el software, a corregir los defectos del software existente, al control de dispositivos, y muchos más. El funcionamiento de los usuarios, los procesos del negocio, y los dispositivos es típicamente complejo. Por extensión, por lo tanto, los requisitos de software son típicamente una combinación compleja de requisitos de diversa gente en diversos niveles de una organización y del ambiente en el cual el software funcionará.

Una característica esencial de todos los requisitos del software es que sean comprobables. Puede ser difícil o costoso verificar ciertos requisitos del software. Por

ejemplo, la verificación del requisito del rendimiento de procesamiento en el centro de la llamada puede hacer necesario el desarrollo del software de la simulación. Los requisitos del software y el personal de la calidad del software deben asegurarse de que los requisitos se puedan verificar dentro de los apremios disponibles del recurso.

Los requisitos tienen otras cualidades además de las características del comportamiento que expresan. Ejemplos comunes incluyen una tasa de prioridad para permitir compensaciones frente a recursos finitos y un valor del estado para permitir que el progreso del proyecto sea supervisado. Típicamente, los requisitos de software se identifican únicamente de modo que puedan estar sujetos al control de configuración del software y manejados sobre el ciclo vital entero del software. [Kot00; Pfl01; Som05; Tha97]

1.2. Producto y requisitos del proceso

Se puede dibujar una distinción entre los parámetros del producto y los parámetros del proceso. Los parámetros del producto son requisitos en software para ser convertido (por ejemplo, "El software verificará que un estudiante resuelva todos los requisitos previos antes de que él o ella se coloque para un curso.").

Un parámetro de proceso es esencialmente un constreñimiento en el desarrollo del software (por ejemplo, "el software será escrito en el Ada."). Éstos se conocen a veces como requisitos de proceso.

Algunos requisitos del software generan requisitos de proceso implícitos. La opción de la técnica de la verificación es un ejemplo. Otro puede ser el uso de técnicas rigurosas de análisis (tales como métodos formales de la especificación) para reducir las averías que pueden conducir a una inadecuada confiabilidad. Los requisitos de proceso pueden también ser impuestos directamente por la organización del desarrollo, su cliente, o terceros tales como un regulador de seguridad [Kot00; Som97].

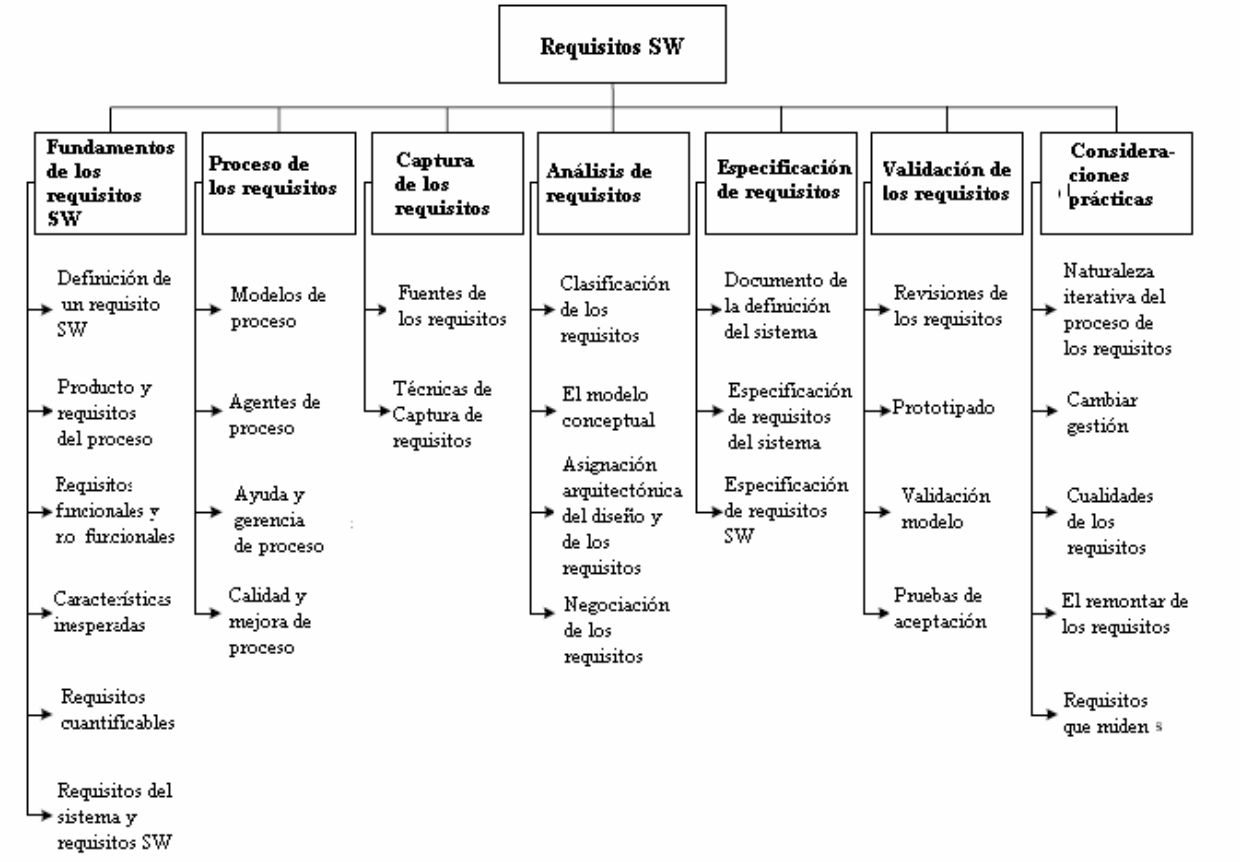

Figura 1: Descomposición de materias para la KA Requisitos del Software

1.3. Requisitos funcionales y no funcionales

Los requisitos funcionales describen las funciones que el software va a ejecutar; por ejemplo, ajustarse a un formato de texto o modular una señal. Se conocen también como capacidades.

Los requisitos no funcionales son los que actúan para obligar la solución. Los requisitos no funcionales se conocen a veces como apremios o requisitos de calidad. Pueden ser clasificados más a fondo según si son requisitos de funcionamiento, requisitos de capacidad de mantenimiento, requisitos de seguridad, requisitos de confiabilidad, o uno de muchos otros tipos de requisitos del software. Estos asuntos también se discuten en KA de la calidad del software. [Kot00; Som97]

1.4. Características inesperadas

Algunos requisitos representan características inesperadas del software- esto es, los requisitos que no pueden ser tratados por un solo componente, pero que su satisfacción va a depender de cómo todos los componentes de software ínter operan. El requisito del rendimiento de procesamiento para un centro de llamadas, por ejemplo, dependería de cómo el sistema de teléfono, el sistema de información, y los operadores obraron recíprocamente bajo condiciones de funcionamiento reales. Las características inesperadas son crucialmente dependientes en la arquitectura del sistema. [Som05]

1.5. Requisitos cuantificables

Los requisitos del software se deben indicar tan clara e inequívocamente como sea posible, y cuantitativamente. Es importante evitar requisitos vagos e inverificables que dependen para su interpretación del juicio subjetivo ("el software será confiable"; "el software será de uso fácil"). Esto es particularmente importante para los requisitos no funcionales. Dos ejemplos de requisitos cuantificados son los siguientes: el software para un centro de llamadas debe aumentar el rendimiento del procesamiento del centro un 20%; y un sistema tendrá una probabilidad de generar un error fatal durante cualquier hora de operación menos de 1 * 10−8. El requisito de rendimiento de procesamiento está a un alto nivel y necesitará ser derivado en un número de requisitos detallados. El requisito de la confiabilidad determinará firmemente la arquitectura del sistema. [Dav93; Som05]

1.6. Requisitos del sistema y requisitos del software

En este asunto, el sistema significa "una combinación recíproca de los elementos para lograr un objetivo definido. Éstos incluyen el hardware, software, soporte lógico inalterable, gente, información, técnicas, instalaciones, servicios, y otros elementos de apoyo," según lo definido por el consejo internacional sobre la ingeniería de sistemas (INCOSE00).

Los requisitos del sistema son los requisitos para el sistema en su totalidad. En un sistema que contiene componentes de software, los requisitos del software se derivan de los requisitos del sistema.

La literatura sobre requisitos llama a veces a los requisitos del sistema "exigencias del consumidor." La guía define "exigencias del consumidor" de una manera restricta como requisitos de los clientes o de los usuarios finales del sistema. Los requisitos del sistema, por el contrario, abarcan los requisitos del usuario, requisitos de otros tenedores de apuestas (por ejemplo autoridades reguladoras), y requisitos sin un origen identificable.

2. Proceso de los requisitos

Esta sección introduce el proceso de los requisitos del software, orientando las cinco subzonas restantes y demostrando cómo el proceso de los requisitos encaja con el proceso de ingeniería del software. [Dav93; Som05]

2.1. Modelos de proceso

El objetivo de este asunto es proporcionar una comprensión de que el proceso de los requisitos

- No es una actividad anticipada discreta del ciclo vital del software, sino un proceso iniciado en principio de un proyecto y a continuación refinado a través del ciclo vital

- Identifica los requisitos del software como elementos de configuración, y los maneja usando las mismas prácticas de gerencia de la configuración del software como otros productos de los procesos del ciclo vital del software

- Necesita ser adaptado a la organización y al contexto del proyecto

Particularmente, el asunto se refiere a cómo las actividades de análisis, especificación, y la validación se configuran para diversos tipos de proyectos y apremios. El asunto también incluye las actividades que proporcionan la entrada en el proceso de los requisitos, por ejemplo estudios de la comercialización y de viabilidad. [Kot00; Rob99; Som97; Som05]

2.2. Agentes de proceso

Este asunto introduce los papeles de la gente que participa en el proceso de los requisitos. Este proceso es fundamental interdisciplinario, y el especialista de los requisitos necesita mediar entre el dominio del tenedor de apuestas y el de la tecnología de dotación lógica. Hay mucha gente implicada además del especialista de los requisitos, cada uno de ellos tiene una función en el software. Los tenedores de apuestas variarán según los proyectos, pero incluyen siempre usuarios/operadores y clientes (quiénes no necesitan ser iguales). [Gog93]

Los ejemplos típicos de los tenedores de apuestas del software incluyen (pero no restringen)

- Usuarios: Este grupo abarca a los que utilizan el software. Es a menudo un grupo heterogéneo que abarca a gente con diversos papeles y requisitos.
- Clientes: Este grupo abarca a los que han comisionado el software o quién representa el blanco de mercado del software.
- Analistas de mercado: Un producto del mass-market no tendrá un cliente que

comisiona, de modo que la gente de comercialización a menudo necesita establecer lo que el mercado necesita y actuar como clientes.

- Reguladores: Muchos dominios de aplicación como por ejemplo el transporte público o la banca son regulados. El software en estos dominios debe conformarse con los requisitos de las autoridades reguladoras.
- Ingenieros de software: Estos individuos tienen un interés legítimo en beneficiarse del desarrollo del software, por ejemplo, reutilizando componentes en otros productos. Si, en este escenario, un cliente de un producto particular tiene requisitos específicos que comprometen el potencial para la reutilización de componentes, los ingenieros de software deben pesar cuidadosamente sus propios intereses contra los del cliente.

No será posible satisfacer perfectamente los requisitos de cada stakeholder, y es el trabajo de los ingenieros de software negociar las compensaciones que son aceptables a los stakeholder principales y dentro de presupuestario, técnico, regulador, y otros apremios. Un requisito previo para esto es que identifiquen a todos los stakeholder, la naturaleza de su "interés" analizada, y sus requisitos sacados. [Dav93; Kot00; Rob99; Som97; You01]

2.3. Ayuda y gerencia de proceso

Este asunto introduce los recursos de la gerencia de proyecto requeridos y consumidos por el proceso de los requisitos. Él establece el contexto para la primera subzona (definición de la iniciación y del alcance) de la tecnología de dotación lógica KA de la gerencia. Su propósito principal es hacer el acoplamiento entre las actividades de proceso identificadas en 2.1 y los temas de coste, recursos humanos, entrenamiento, y herramientas. [Rob99; Som97; You01]

2.4. Calidad y mejora de proceso

Este asunto se refiere al gravamen de la calidad y de la mejora del proceso de los requisitos. Su propósito es acentuar el papel dominante que los requisitos procesan en términos de coste y puntualidad de un producto de software, y de la satisfacción del cliente con ello [Som97]. Ayudará a orientar el proceso de los requisitos con estándares de calidad y modelos de mejora de proceso para el software y los sistemas. El proceso de calidad y la mejora se relacionan de cerca con la calidad del software y la tecnología de dotación lógica KA. De interés particular están las aplicaciones de calidad del software, sus cualidades y medida, y la definición de proceso de software. Este punto abarca

- Cobertura de proceso de los requisitos mediante estándares y modelos de la mejora
- Medidas de proceso de los requisitos y benchmarking
- Plan de mejora y puesta en práctica [Kot00 de la mejora del □; Som97; You01]

3. Captura de los requisitos
[Dav93; Gog93; Lou95; Pfl01]

La captura de los requisitos se refiere a de donde vienen los requisitos del software y cómo el ingeniero de software puede recogerlos. Es la primera etapa en la construcción de una comprensión del problema que el software requiere solucionar. Es fundamental una actividad humana, y es donde identifican a los stakeholders y las relaciones se establecen entre el equipo del desarrollo y el cliente. También se conoce como "descubrimiento de los requisitos," y "adquisición de los requisitos."

Uno de los aspectos fundamentales de la buena ingeniería de software es que haya buena comunicación entre los usuarios del software y los ingenieros. Antes de que comience el desarrollo, los especialistas de requisitos pueden formar el conducto para esta comunicación. Deben mediar entre el dominio de los usuarios del software (y otros stakeholders) y el mundo técnico del ingeniero de software.

3.1. *Fuentes de los requisitos* [Dav93; Gog93; Pfl01]

Los requisitos tienen muchas fuentes en software típico, y es esencial que todas las fuentes potenciales estén identificadas y evaluadas para su impacto en él. Este asunto se diseña para promover el conocimiento de las varias fuentes de los requisitos del software y de los armazones para manejarlos. Los puntos principales cubiertos son

- Metas: El término meta (a veces llamada "preocupación de negocio" o "factor crítico del éxito") se refiere a los objetivos totales, de alto nivel del software. Las metas proporcionan la motivación para el software, pero a menudo son formuladas vagamente. Es necesidad que los ingenieros de software presten atención particular a determinar el valor (concerniente a prioridad) y coste de metas. Un estudio de viabilidad es una manera relativamente barata de hacer esto. [Lou95].
- Conocimiento del dominio: Los ingenieros de software necesitan adquirir, o tener disponible, conocimiento sobre el uso del dominio. Esto permite deducir el conocimiento que los stakeholders no articulan, determinar las compensaciones que serán necesarias en requisitos que están en conflicto, y, a veces, para actuar como campeón del "usuario".
- Stakeholders (véase a agentes de proceso del asunto 2.2). Mucho software se ha probado insatisfactorio porque había tensionado los requisitos de un grupo de stakeholders a expensas de los de otros. Por lo tanto, se entrega el software que es difícil de utilizar o que derriba las estructuras culturales o políticas de la organización del cliente. Los ingenieros de software necesitan identificar, representar, y manejar "puntos de vista" de muchos diversos tipos de tenedores de apuestas. [Kot00]
- El entorno operacional. Los requisitos serán derivados del ambiente en el cual el software será ejecutado. Éstos pueden ser, por ejemplo, el medir el tiempo en tiempo real del software o de la interoperabilidad en un ambiente de la oficina. Éstos deben ser buscados activamente, porque pueden afectar grandemente a la viabilidad y el coste del software, y restringen las opciones de diseño. [Tha97]
- El entorno de la organización. El software se requiere a menudo para apoyar un proceso del negocio, la selección del cual se puede condicionar por la estructura, cultura, y política interna de la organización. Los ingenieros de software necesitan ser sensibles a éstos, puesto que, el nuevo software no debe forzar generalmente cambios imprevistos en el proceso del negocio.

3.2. *Técnicas de Captura de requisitos* [Dav93; Kot00; Lou95; Pfl01]

Una vez que se hayan identificado las fuentes de los requisitos, ingenieros de software pueden comenzar a sacar requisitos de ellos. Este asunto se concentra en las técnicas para conseguir que los stakeholders articulen sus requisitos. Es un área muy difícil y ingenieros de software necesitan sensibilizarse al hecho que (por ejemplo) los usuarios pueden tener dificultad para describir sus tareas, puede dejar la información importante sin especificar, o pueden estar poco dispuestos o cooperar. Es particularmente importante entender que la captura no es una actividad pasiva, y que, ingenieros de software tienen que trabajar difícilmente para sacar la información adecuada. Existe un número de técnicas para hacer esto, las principales son [Gog93]

- Entrevistas, medios "tradicionales" de sacar requisitos. Es importante entender las ventajas y limitaciones de las entrevistas y cómo deben ser conducidas.
- Escenarios, valiosos medios para proporcionar contexto a las exigencias del consumidor. Proporcionan un marco para preguntas sobre tareas del usuario permitiendo preguntas "y si" y "cómo se hace esto". El tipo más común de escenario es el caso del uso.
- Prototipos, una herramienta valiosa para clarificar requisitos confusos. Pueden actuar de una manera similar a los escenarios, proveyendo a los usuarios un contexto dentro del cuál pueden entender mejor qué información necesitan proporcionar. Hay una amplia gama de técnicas de prototipado, maquetas de versiones de pruebas beta de los diseños de la pantalla del software , y un traslado fuerte de su uso para captura de los requisitos y el uso de prototipos para la validación de los requisitos (véase el asunto 6.2 Prototipado).
- Reuniones. El propósito de éstas es intentar alcanzar un efecto aditivo por el que un grupo de gente puede obtener más penetración en los requisitos del software que trabajando individualmente. Ellos pueden inspirarse y refinar las ideas que pueden ser difíciles de traer a la superficie usando entrevistas. Otra ventaja es que dejan a los stakeholders reconocer donde hay requisitos en conflicto. Cuando se aplica bien, esta técnica puede resultar en un rico y constante sistema de requisitos que

difícilmente sería realizable de otro modo. Sin embargo, las reuniones necesitan ser dirigidas cuidadosamente (por lo tanto es necesario un facilitador) para evitar que una situación ocurra donde las capacidades críticas del equipo son erosionadas por lealtad del grupo, o los requisitos que reflejan las preocupaciones de algunos favorecen la gente abierta (y quizás mayor) en detrimento de otros.

- Observación. La importancia del contexto del software dentro del ambiente de organización ha conducido a la adaptación de las técnicas de observación para captura de los requisitos. Los ingenieros de software aprenden sobre las tareas del usuario sumergiéndose en la observación de cómo los usuarios obran recíprocamente con su software. Estas técnicas son relativamente costosas, pero son instructivas porque ilustran que muchas tareas del usuario y procesos del negocio son demasiado sutiles y complejos para que sus agentes los describan fácilmente.

4. Análisis de requisitos [Som05]

Este asunto se refiere al proceso de analizar requisitos para

- Detectar y resolver los conflictos entre los requisitos
- Descubrir los límites del software y cómo debe obrar recíprocamente con su ambiente
- Elaborar los requisitos del sistema para derivar requisitos software

La vista tradicional del análisis de requisitos ha sido que esté reducida a modelado conceptual utilizando uno de varios métodos de análisis tales como Análisis Estructurados y Técnicas de Diseño (SADT). Mientras que el modelado conceptual es importante, nosotros incluimos la clasificación de requisitos para ayudar a informar a compensaciones entre los requisitos (clasificación de los requisitos) y el proceso de establecer estas compensaciones (negociación de los requisitos). [Dav93]

El cuidado se debe tomar para describir requisitos con exactitud, suficientemente como para permitir que los requisitos sean validados, su implementación sea verificada, y sus costes estimados.

4.1 Clasificación de los requisitos [Dav93; Kot00; Som05]

Los requisitos se pueden clasificar en un número de dimensiones. Los ejemplos incluyen:

- Si el requisito es funcional o no funcional (véase el asunto 1.3 funcional y requisitos no funcionales).
- Si el requisito está derivado de uno o más requisitos de alto nivel o una propiedad emergente (véase las características inesperadas del asunto 1.4) o está impuesto directamente ante el software por un tenedor de apuestas u otra fuente.
- Si el requisito está en el producto o proceso. Los requisitos en el proceso pueden obligarnos a la opción del contratista, a adaptar el proceso de la ingeniería del software o los estándares que se adherirán.
- La prioridad del requisito. Generalmente cuanta más alta es la prioridad, más esencial es el requisito para satisfacer las metas finales del software. A menudo clasificado en una escala de punto fijo tal como obligatorio, altamente deseable, deseable u opcional, la prioridad a menudo tiene que ser equilibrada con el coste de desarrollo e implementación.
- El alcance del requisito. El alcance se refiere al grado al cual un requisito afecta el software y componentes software. Algunos requisitos, particularmente los no funcionales, tienen un alcance global que no puede ser asignado a un componente discreto. Por lo tanto, el requisito con alcance global puede afectar fuertemente a la arquitectura del software y el diseño de muchos componentes, mientras que uno puede con un alcance estrecho ofrecer un número de opciones del diseño y no afectar demasiado a la satisfacción de otros requisitos.
- Volatilidad/estabilidad. Algunos requisitos cambiarán durante el ciclo de vida del software, y se igualarán durante el proceso de desarrollo. Es útil hacer estimaciones de la probabilidad de que se puedan hacer cambios. Por ejemplo, en actividades bancarias, los requisitos para las funciones para calcular y para acreditar los intereses en las cuentas son probablemente más estables que un requisito para mantener un tipo particular de cuenta exenta de impuestos. Lo anterior refleja una característica fundamental del dominio de las actividades bancarias (esas cuentas pueden ganar intereses), mientras que el último se puede dejar obsoleto por un cambio de gobierno. Señalar requisitos potencialmente volátiles puede ayudar al ingeniero del software a establecer un diseño que sea más tolerante al cambio.

Otras clasificaciones pueden ser apropiadas, dependiendo de la práctica normal y del uso que se le dé.

Hay un desfase fuerte entre la clasificación de los requisitos y las cualidades de los requisitos (véase las cualidades de los requisitos del punto 7.3).

4.2. El modelo conceptual
[Dav93; Kot00; Som05]

El desarrollo de modelos de un problema del mundo real es clave para el análisis de requisitos del software. Su propósito es ayudar a entender el problema, más que iniciar el diseño de la solución. Por lo tanto, modelos conceptuales abarcan modelos de entidades del dominio del problema, configurados para reflejar sus relaciones y dependencias con el mundo real.

Varias clases de modelos pueden ser desarrollados. Éstos incluyen datos y controlan flujos, modelos de estado, rastros de evento, interacciones de usuario, modelos de objeto, modelos de datos y muchos otros. Los factores que influencian la opción del modelo incluyen

- La naturaleza del problema. Algunos tipos de software exigen que ciertos aspectos estén analizados rigurosamente. Por ejemplo, controlar el flujo y los modelos de estado son probablemente más importantes para el software en tiempo real que para el software de gerencia de información, mientras que es generalmente lo contrario para los modelos de datos.
- La maestría del ingeniero del software. Es a menudo más productivo adoptar una notación que modele o método con el cual el ingeniero del software tiene una experiencia.
- Los requisitos de proceso del cliente. Los clientes pueden imponer su notación o método favorecido, o prohibir cualquiera que le sea desconocido. Este factor puede estar en conflicto con el factor anterior.
- La disponibilidad de métodos y de herramientas. Notaciones o métodos que son mal apoyados por el entrenamiento y las herramientas pueden no alcanzar la aceptación esperada aunque se satisfagan tipos particulares de problemas.

Observar que, en casi todos los casos, es útil comenzar construyendo un modelo del contexto del software. El contexto del software proporciona una conexión entre el software previsto y su ambiente externo. Esto es crucial para entender el contexto del software en su ambiente operacional e identificar sus interfaces con el ambiente.

La aplicación de modelar se junta firmemente con la de los métodos. Para los propósitos prácticos, un método es una notación (o sistema de notaciones) apoyado por un proceso que dirige el uso de las notaciones. Hay escasa evidencia empírica para apoyar las demandas de superioridad de una notación sobre otra. Sin embargo, la extensa aceptación de un método o de una notación particular puede conducir a nivel industrial al beneficio de habilidades y de conocimiento. Ésta es actualmente la situación con el UML (Lenguaje de Modelado Unificado). (UML04)

El modelado formal usando notaciones basadas en matemática discreta, y que son detectables al razonamiento lógico, han tenido impacto en algunos dominios especializados. Éstos puede ser impuesto por los clientes o los estándares y puede ofrecer ventajas que obligan al análisis de ciertas funciones o componentes críticos.

Este asunto no busca "enseñar" un estilo o una notación de modelado particular sino proporcionar algo a la dirección con el propósito de modelar.

Dos estándares proporcionan las notaciones que pueden ser útiles en desarrollo conceptual realizando modelado-IEEE conceptual Std 1320.1, IDEF0 para modelado funcional; e IEEE Std 1320.2, IDEF1X97 (IDEFObject) para modelado de la información.

4.3. Asignación arquitectónica del diseño y de los requisitos
[Dav93; Som05]

En un cierto punto, la arquitectura de la solución debe ser derivada. El diseño arquitectónico es el punto en el cual el proceso de los requisitos se junta con software o diseño de sistemas e ilustra cómo de imposible es desemparejar ambas tareas. [Som01] Este asunto está de cerca relacionado con el capítulo de la estructura y de la arquitectura del software en KA del diseño del software. En muchos casos, el ingeniero del software actúa como arquitecto del software porque el proceso de analizar y de elaborar los requisitos exige que los componentes que serán responsables de satisfacer los requisitos estén identificados. Esto es localización de requisitos-la asignación, a los componentes, de la responsabilidad de satisfacer requisitos.

La asignación es importante para permitir análisis detallado de requisitos. Por lo tanto, por ejemplo, una vez un sistema de requisitos se han asignado a un componente, los requisitos individuales se pueden analizar más a fondo para descubrir otros requisitos de cómo el componente necesita obrar recíprocamente con otros componentes para satisfacer los requisitos asignados. En proyectos grandes, la asignación estimula un nuevo análisis para cada subsistema. Como ejemplo, requisitos para el funcionamiento de los frenos de un coche (distancia que frena, seguridad en condiciones que conducen, suavidad del uso, la presión del pedal requerida, y así sucesivamente) se puede asignar un hardware que frena (montajes mecánicos e hidráulicos) y un sistema de frenos antibloqueo (ABS). Solamente cuando un requisito para un sistema de frenos antibloqueo ha sido identificado, y los requisitos asignados a él, pueden usarse las

capacidades del ABS, el hardware de frenado identificado, y las características indefinidas (tales como el peso del coche) para identificar los requisitos detallados del software del ABS.

El diseño arquitectónico se identifica de cerca con el modelado conceptual. El mapeado de las entidades del dominio del mundo real para componentes de software no siempre tiene un diseño obvio, así que arquitectónicamente se identifica como a asunto separado. Los requisitos de notaciones y los métodos son ampliamente iguales para modelado conceptual y diseño arquitectónico.

IEEE Std 1471-2000, práctica recomendada para la descripción arquitectónica de sistemas orientados al software, sugiere un acercamiento del múltiple punto de vista para describir la arquitectura de sistemas y de sus artículos del software.
(IEEE1471-00)

4.4. Negociación de los requisitos

Otro término comúnmente utilizado para este tema es "resolución del conflicto." Esto se refiere a problemas de resolución de los requisitos donde los conflictos ocurren entre dos stakeholders que requieren características mutuamente incompatibles, entre los requisitos y los recursos, o en entre requisitos funcionales y no funcionales, por ejemplo.

[Kot00, Som97] en la mayoría de los casos, es imprudente para el ingeniero del software tomar una decisión unilateral, y hace necesario consultar con los stakeholders para alcanzar un consenso en una compensación apropiada. Es a menudo importante por esas razones contractuales que tales decisiones sean detectables de nuevo al cliente. Hemos clasificado esto como asunto del análisis de requisitos del software porque los problemas emergen como resultado el análisis. Sin embargo, un caso fuerte se puede también hacer para considerar los requisitos como asunto de la validación.

5. Especificación de requisitos

Para la mayoría de las profesiones de la ingeniería, el término "especificación" se refiere a la asignación de valores o límites numéricos para metas del diseño del producto. (Vin90) Los sistemas físicos típicos tienen un número relativamente pequeño de tales valores. Típicamente el software tiene una gran cantidad de requisitos, y el énfasis se comparte entre la ejecución de la cuantificación numérica y el manejo de la complejidad de la interacción entre el gran número de requisitos. Así pues, en software el término, "especificación de requisitos del software"se refiere típicamente a la producción de un documento, o a su equivalente electrónico, que puede estar sistemáticamente repasado, evaluado, y aprobado. Para los sistemas complejos, particularmente ésos que implican componentes no-software, se elaboran tres tipos de documentos: definición de sistema, sistema requisitos, y requisitos del software. Para sistemas simples, solamente el tercero de éstos es requerido. Los tres documentos se describen aquí, entendiendo que combinados pueden ser apropiados. Una descripción de la ingeniería de sistemas se puede encontrar en el Capítulo 12, disciplinas relacionadas de la tecnología de dotación lógica.

5.1. El documento de la definición de sistema

Este documento (conocido a veces como documento de exigencias del o concepto de operaciones) registra el sistema requisitos. Define los requisitos del sistema de alto nivel desde la perspectiva del dominio. Su número total de lectores incluye los representantes de los usuarios del sistema/de los clientes (la comercialización puede desempeñar estos papeles del mercado software), así que su contenido debe estar en términos de dominio. El documento enumera los requisitos del sistema junto con información de fondo sobre los objetivos totales para el sistema, su ambiente de misión y una declaración de apremios, asunciones, y requisitos no funcionales. Puede incluir los modelos conceptuales diseñados para ilustrar el contexto del sistema, panoramas del uso y las entidades principales del dominio, así como datos, la información, y workflows. IEEE Std 1362, concepto del documento de las operaciones, proporciona consejo sobre la preparación y contenido de tal documento.
(IEEE1362-98)

5.2. Especificación de requisitos del sistema [Dav93; Kot00; Rob99; Tha97]

Desarrolladores de sistemas con los componentes software y nonsoftware, un avión de pasajeros moderno, por ejemplo, separa a menudo la descripción de los requisitos del sistema de la descripción de los requisitos del software. En esto se especifica la visión, requisitos del sistema, los requisitos software se derivan de los requisitos del sistema, y entonces los requisitos para los componentes de software se especifican. En sentido estricto, la especificación de requisitos del sistema es una actividad de la ingeniería de sistemas y esta fuera del alcance de esta guía. IEEE Std 1233 es una guía para los requisitos del sistema que se convierten.
(IEEE1233-98)

5.3. Especificación de requisitos del software [Kot00; Rob99]

La especificación de requisitos del software establece la base para el acuerdo entre los clientes y los contratistas o los proveedores (en proyectos del mercado, estos papeles se pueden desempeñar por las divisiones de comercialización y desarrollo) en los que hay que hacer el producto de software, así como lo que no se espera que haga. Para los lectores no

técnicos, el documento de la especificación de los requisitos es acompañado a menudo por un documento de la definición de los requisitos del software.

La especificación de requisitos del software permite un riguroso gravamen de requisitos antes de que el diseño pueda comenzar y reducir un reajuste final. Debe también proporcionar una base realista para estimar costes, riesgos, y horario del producto.

Las organizaciones pueden también utilizar un documento de especificación de requisitos software para desarrollar su propia validación y que la verificación sea más productiva.

La especificación de requisitos software proporciona una base informada para transferir un producto de software a los nuevos usuarios o a las máquinas nuevas. Finalmente, puede proporcionar una base para el realce de software.

Los requisitos del software se escriben a menudo en lenguaje natural, pero, en la especificación de requisitos del software, ésta se puede suplir por formal o semi-formal. La selección de notaciones apropiadas permite requisitos y los aspectos particulares de la arquitectura del software que se describirá más ajustadamente que en lengua natural. La regla general es que las notaciones deben ser utilizadas para que permitan a los requisitos ser descritos tan exactamente como sea posible. Esto es particularmente crucial para otros tipos seguridad-crítica y casos de software confiable. Sin embargo, la opción de la notación es obligada a menudo por el entrenamiento, las habilidades y las preferencias de los autores y de los lectores del documento.

Se han desarrollado un número de indicadores de la calidad para relacionar la calidad de la especificación de requisitos del software a otras variables del proyecto por ejemplo coste, aceptación, funcionamiento, horario, reproducibilidad, indicadores de la calidad etc. para el individuo las declaraciones de la especificación de requisitos del software incluyen imperativos, directorios, frases débiles, opciones, y continuaciones. Indicadores para el documento de especificación de requisitos software incluye tamaño, legibilidad, especificación, profundidad, y estructura del texto.
[Dav93; Tha97] (Ros98)

IEEE tiene un estándar, IEEE Std 830 [IEEE830-98], para producción y contenido de la especificación de los requisitos del software. También, IEEE 1465 (similar a ISO/IEC 12119) es un estándar que trata requisitos de calidad en paquetes de software. (IEEE1465-98)

6. Validación de los requisitos

[Dav93]

Los documentos de los requisitos pueden estar conformes a la validación y procedimientos de verificación. Los requisitos pueden ser validados para asegurarse de que el ingeniero del software entiende los requisitos, y es también importante para verificar que un documento de requisitos se conforma con la compañía de los estándares, y éste es comprensible, constante, y finito. Las notaciones formales ofrecen la ventaja importante de permitir que las dos características pasadas sean probadas (en un sentido estricto, por lo menos). Diversos stakeholders, incluyendo los representantes del cliente y del revelador, deben también repasar los documentos. Los documentos de los requisitos son conformes a las mismas prácticas de gerencia de la configuración del software como los otros puntos relevantes de los procesos del ciclo de vida del software. (Bry94, Ros98)

Es normal programar explícitamente unos o más puntos en el proceso de los requisitos donde están los requisitos validados. La puntería es tomar cualquier problema antes de que los recursos se destinen a tratar los requisitos. La validación de los requisitos se refiere al proceso de examinar el documento de los requisitos para asegurarse de que este define el software correctamente (es decir, el software que los usuarios esperan). [Kot00]

6.1 Revisiones de los requisitos

[Kot00; Som05; Tha97]

Quizás los medios más comunes de la validación están cerca de inspección o revisiones de los documentos de los requisitos. Asignan un grupo de revisores en un escrito para buscar errores, asunciones confundidas, la carencia de la claridad, y la desviación de la costumbre. La composición del grupo que conduce la revisión es importante (por lo menos un representante del cliente debe ser incluido para un proyecto cliente-conducido, por ejemplo), y puede ayudar a proporcionar la dirección en qué buscar bajo listas de comprobación.

Las revisiones se pueden constituir en el final del documento de definición del sistema, el documento de la especificación de sistema, el documento de la especificación de requisitos del software, especificación de la línea de fondo para un nuevo lanzamiento, o en cualquier otro paso en el proceso. IEEE Std 1028 proporciona la dirección para conducir tales revisiones. (IEEE1028-97) Las revisiones son también cubiertas en KA de la calidad del software, punto 2.3 Revisiones e intervenciones.

6.2. Prototipado

[Dav93; Kot00; Som05; Tha97]

Prototipar es comúnmente el medio para validar la interpretación del ingeniero del software de los requisitos del software, así como para sacar nuevos requisitos. Como con el elicitation, hay una gama de técnicas de prototipado y un número de puntos en el proceso donde la validación del prototipo puede ser apropiada. La ventaja de usar prototipos es que pueden hacer más fácil la interpretación de las asunciones del ingeniero del software y, donde lo necesite, dan la explicación útil de porqué son incorrectas. Por ejemplo, el comportamiento dinámico de un interfaz utilizador se puede entender mejor a través de un prototipo animado que a través de la descripción textual o de modelos gráficos. Hay también desventajas, sin embargo. Éstos incluyen el peligro de la atención de los usuarios que es distraída de la base de la funcionalidad subyacente por las ediciones o los problemas cosméticos de la calidad con el prototipo. Por esta razón, algunas personas dicen que los prototipos que evitan. Los prototipos pueden ser costosos. Sin embargo, si evitan el despilfarro de los recursos causados intentando satisfacer requisitos erróneos, su coste puede ser más fácilmente justificado.

6.3. Validación modelo

[Dav93; Kot00; Tha97]

Es típicamente necesario validar la calidad de los modelos desarrollados durante el análisis. Por ejemplo, en modelos de objeto, es útil para realizar un análisis estático para verificar que las trayectorias de comunicación existen entre los objetos que, en dominio de los stakeholders, intercambian datos. Si las notaciones de especificación formal se utilizan, es posible utilizar el razonamiento formal para probar características de la especificación.

6.4. Pruebas de aceptación

[Dav93]

Una característica esencial de un requisito del software es que debe ser posible validar que el producto final lo satisface. Los requisitos que no pueden ser validados son "deseos realmente justos." Una tarea importante es por lo tanto planear cómo verificar cada requisito. En la mayoría de los casos, el diseño de pruebas de aceptación hace esto. Identificar y diseñar pruebas de aceptación pueden ser difícil para los requisitos no funcionales (véase el asunto los requisitos funcionales y no funcionales de 1.3). Para ser validados, deben primero ser analizados al punto donde pueden ser expresados cuantitativamente.
La información adicional se puede encontrar en el software KA de prueba, conformidad de la prueba en el punto 2.2.4.

7. Consideraciones prácticas

El primer nivel de la descomposición de los puntos presentados en este KA puede parecer que describe una secuencia lineal de actividades. Ésta es una vista simplificada del proceso.
[Dav93]

Los requisitos procesan palmos de todo el ciclo de vida del software. Cambiar la gerencia y el mantenimiento de los requisitos en un estado que refleja exactamente el software que se construirá, o se ha construido, es clave para el éxito del proceso ingeniería del software
[Kot00; Lou95]

No toda organización tiene una cultura de documentación y manejo de requisitos. Es frecuente en las compañías de start-up dinámicas, conducidas por una "visión fuerte del producto" y recursos limitados, para ver la documentación de los requisitos como gastos indirectos innecesarios. Más a menudo, sin embargo, según estas compañías crecen, mientras que su base de cliente crece, y como su producto comienza a desarrollarse, estas descubren que necesitan recuperar los requisitos que las características de producto motivadas para determinar el impacto de cambios propuestos. Por lo tanto, la documentación de los requisitos y la gerencia del cambio son llave al éxito de cualquier proceso de los requisitos.

7.1. Naturaleza iterativa del proceso de los requisitos

[Kot00; You01]

Hay presión general en la industria del software para ciclos de desarrollo más cortos, y esto está particularmente pronunciado en sectores del mercado altamente competitivos. Por otra parte, la mayoría de los proyectos son obligados de cierta manera por su ambiente, y muchas son mejoras, o revisiones, del software existente donde está la arquitectura dada. En la práctica, por lo tanto, es casi siempre impráctico poner el proceso de los requisitos en ejecución como proceso lineal, determinista en el cual los requisitos del software se saquen de los stakeholders, asignado, y entregado al equipo de desarrollo del software. Es ciertamente un mito que los requisitos para los proyectos grandes del software siempre están entendidos perfectamente o especificados perfectamente. [Som97]

En su lugar, los requisitos iteran típicamente hacia un nivel de calidad y detallan que es suficiente permitir las decisiones del diseño y de la consecución que se harán. En algunos proyectos, esto puede dar lugar a requisitos antes de que todas sus características se entiendan completamente. Esto arriesga a una reanudación costosa si los problemas emergen tarde en el proceso de la ingeniería del software. Sin

embargo, los ingenieros del software son obligados necesariamente por planes de la gerencia de proyecto y deben por lo tanto tomar medidas para asegurarse de que la "calidad" de los requisitos es tan alta como sea posible dado los recursos disponibles. Deben, por ejemplo, hacer explícita cualquier asunción que sostengan los requisitos, así como cualesquiera problemas sabidos.

En casi todos los casos, el entendimiento de los requisitos continúa desarrollándose mientras que procede el diseño y el desarrollo. Esto conduce a menudo a la revisión de requisitos tarde en el ciclo de vida. Quizás el punto más crucial de entender la ingeniería de requisitos es que una proporción significativa de los requisitos cambiará. Esto es a veces debido a los errores en el análisis, pero es con frecuencia una consecuencia inevitable del cambio en el "ambiente": por ejemplo, la funcionalidad del cliente o el ambiente del negocio, o el mercado en el cual software se va a vender. Cualquiera que sea la causa, es importante reconocer la inevitabilidad del cambio para atenuar sus efectos. El cambio tiene que ser manejado asegurando esos cambios propuestos mediante una revisión definida y un análisis del proceso de aprobación, y, aplicando los requisitos cuidadosos que remontan, del impacto, y la configuración del software (véase la configuración del software KA de la gerencia). Por lo tanto, el proceso de los requisitos no es simplemente una tarea anticipada en el desarrollo del software, pero atraviesa por completo el ciclo de vida del software. En un proyecto típico, las actividades de los requisitos del software se desarrollan en un cierto.

7.2. Cambiar a gestión

[Kot00]

La gestión del cambio es central en el análisis de requisitos. Este asunto describe el papel de la gestión del cambio, los procedimientos que necesitan estar preparados, y el análisis que se debe aplicar a los cambios propuestos. Tiene acoplamientos fuertes a la configuración del software KA de la gestión.

7.3. Cualidades de los requisitos

[Kot00]

Los requisitos deben consistir no sólo una especificación de qué se requiere, sino también de la información ancilar que las ayudas manejan e interpretan los requisitos. Esto debe incluir varias dimensiones de la clasificación del requisito (véase la clasificación de los requisitos del asunto 4.1) y el método de la verificación o el plan de prueba de aceptación. Puede también incluir la información adicional tal como un resumen-análisis razonado para cada requisito, la fuente de cada requisito, y una historia del cambio. Los requisitos más importantes atribuyen, sin embargo, un identificador que permite requisitos identificados inequívocamente.

7.4 El remontar de los requisitos

[Kot00]

El remontar de los requisitos se refiere a la fuente recuperación de requisitos y de predecir los efectos de esos requisitos. El trazado es fundamental para el análisis de la ejecución del impacto cuando los requisitos cambian. Un requisito debe ser detectable al revés a requisitos y los stakeholders que lo motivaron (de un requisito del software de nuevo a los requisitos del sistema que ayudan a satisfacer, por ejemplo). Inversamente, un requisito debe ser detectable entidades del diseño que lo satisfacen (por ejemplo, de un requisito del sistema en el software requisitos que se han elaborado a partir de él, y encendido en los módulos del código que lo ponen en ejecución).

Los requisitos que remontan para un proyecto típico formarán un gráfico acíclico dirigido complejo (DAG) de requisitos.

7.5 Requisitos que miden

Como cuestión práctica, es típicamente útil tener cierto concepto del "volumen" de los requisitos para un producto de software particular. Este punto es útil evaluando el "tamaño" de un cambio en requisitos, en estimar el coste de una tarea del desarrollo o del mantenimiento, o simplemente para el uso como el denominador en otra medida. La medida funcional del tamaño (FSM) es una técnica para evaluar el tamaño de un cuerpo de requisitos funcionales. IEEE Std 14143.1 define el concepto de FSM. [IEEE14143.1-00] Estándares de ISO/IEC y otras fuentes describen métodos particulares del FSM.

Información adicional sobre la medida del tamaño y los estandares se encuentran en el proceso de la ingenieria del software.

MATRIX OF TOPICS VS. REFERENCE MATERIAL

	[Dav93]	[Gog93]	[IEEE830-98]	[IEEE14143.1-00]	[Kot00]	[Lou95]	[Pfl01]	[Rob99]	[Som97]	[Som05]	[Tha97]	[You01]
1. Software Requirements Fundamentals												
1.1 Definition of a Software Requirement					*		*			c5	c1	
1.2 Product and Process Requirements					*				c1			
1.3 Functional and Non-functional Requirements					*				c1			
1.4 Emergent Properties										c2		
1.5 Quantifiable Requirements	c3s4									c6		
1.6 System Requirements and Software Requirements												
2. Requirements Process	*									c5		
2.1 Process Models					c2s1			*	c2	c3		
2.2 Process Actors	c2	*			c2s2			c3	c2			c3
2.3 Process Support and Management								c3	c2			c2,c7
2.4 Process Quality and Improvement					c2s4				c2			c5
3. Requirements Elicitation	*	*				*	*					
3.1 Requirements Sources	c2	*			c3s1	*	*				c1	
3.2 Elicitation Techniques	c2	*			c3s2	*	*					
4. Requirements Analysis	*									c6		
4.1 Requirements Classification	*				c8s1					c6		
4.2 Conceptual Modeling	*				*					c7		
4.3 Architectural Design and Requirements Allocation	*									c10		
4.4 Requirements Negotiation					c3s4				*			
5. Requirements Specification												
5.1 The System Definition Document												
5.2 The System Requirements Specification	*				*			c9			c3	
5.3 The Software Requirements Specification	*		*		*			c9			c3	
6. Requirements Validation	*				*							
6.1 Requirements Reviews					c4s1					c6	c5	
6.2 Prototyping	c6				c4s2					c8	c6	
6.3 Model Validation	*				c4s3						c5	
6.4 Acceptance Tests	*											
7. Practical Considerations	*				*	*						
7.1 Iterative Nature of the Requirements Process					c5s1				c2			c6
7.2 Change Management					c5s3							
7.3 Requirement Attributes					c5s2							
7.4 Requirements Tracing					c5s4							
7.5 Measuring Requirements				*								

REFERENCIAS RECOMENDADAS PARA REQUSITOS SW

[Dav93] A.M. Davis, *Software Requirements: Objects, Functions and States*, Prentice Hall, 1993.

[Gog93] J. Goguen and C. Linde, "Techniques for Requirements Elicitation," presented at International Symposium on Requirements Engineering, 1993.

[IEEE830-98] IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE, 1998.

(IEEE14143.1-00) IEEE Std 14143.1-2000//ISO/ IEC14143-1:1998, *Information Technology—Software Measurement—Functional Size Measurement—Part 1: Definitions of Concepts*, IEEE, 2000.

[Kot00] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, 2000.

[Lou95] P. Loucopulos and V. Karakostas, *Systems Requirements Engineering*, McGraw-Hill, 1995. [Pfl01] S.L. Pfleeger, "Software Engineering: Theory and Practice," second ed., Prentice Hall, 2001, Chap. 4.

[Rob99] S. Robertson and J. Robertson, *Mastering the Requirements Process*, Addison-Wesley, 1999.

[Som97] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, 1997, Chap. 1-2.

[Som05] I. Sommerville, *Software Engineering*, seventhed., Addison-Wesley, 2005.

[Tha97] R.H. Thayer and M. Dorfman, eds., *Software Requirements Engineering*, IEEE Computer Society Press, 1997, pp. 176-205, 389-404.

[You01] R.R. You, *Effective Requirements Practices*, Addison-Wesley, 2001.

APÉNDICE A. LISTA DE LECTURAS COMPLEMENTARIAS

(Ale02) I. Alexander and R. Stevens, *Writing Better Requirements*, Addison-Wesley, 2002.

(Ard97) M. Ardis, "Formal Methods for Telecommunication System Requirements: A Survey of Standardized Languages," *Annals of Software Engineering*, vol. 3, 1997.

(Ber97) V. Berzins et al., "A Requirements Evolution Model for Computer Aided Prototyping," presented at Ninth IEEE International Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute, 1997.

(Bey95) H. Beyer and K. Holtzblatt, "Apprenticing with the Customer," *Communications of the ACM*, vol. 38, iss. 5, May 1995, pp. 45-52.

(Bru95) G. Bruno and R. Agarwal, "Validating Software Requirements Using Operational Models," presented at Second Symposium on Software Quality Techniques and Acquisition Criteria, 1995.

(Bry94) E. Bryne, "IEEE Standard 830: Recommended Practice for Software Requirements Specification," presented at IEEE International Conference on Requirements Engineering, 1994.

(Buc94) G. Bucci et al., "An Object-Oriented Dual Language for Specifying Reactive Systems," presented at IEEE International Conference on Requirements Engineering, 1994.

(Bus95) D. Bustard and P. Lundy, "Enhancing Soft Systems Analysis with Formal Modeling," presented at Second International Symposium on Requirements Engineering, 1995.

(Che94) M. Chechik and J. Gannon, "Automated Verification of Requirements Implementation," presented at Proceedings of the International Symposium on Software Testing and Analysis, special issue, 1994.

(Chu95) L. Chung and B. Nixon, "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach," presented at Seventeenth IEEE International Conference on Software Engineering, 1995.

(Cia97) P. Ciancarini et al., "Engineering Formal Requirements: An Analysis and Testing Method for Z Documents," *Annals of Software Engineering*, vol. 3, 1997.

(Cre94) R. Crespo, "We Need to Identify the Requirements of the Statements of Non-Functional Requirements," presented at International Workshop on Requirements Engineering: Foundations of Software Quality, 1994.

(Cur94) P. Curran et al., "BORIS-R Specification of the Requirements of a Large-Scale Software Intensive System," presented at Requirements Elicitation for Software-Based Systems, 1994.

(Dar97) R. Darimont and J. Souquieres, "Reusing Operational Requirements: A Process-Oriented Approach," presented at IEEE International Symposium on Requirements Engineering, 1997.

(Dav94) A. Davis and P. Hsia, "Giving Voice to Requirements Engineering: Guest Editors' Introduction," *IEEE Software*, vol. 11, iss. 2, March 1994, pp. 12-16.

(Def94) J. DeFoe, "Requirements Engineering Technology in Industrial Education," presented at IEEE International Conference on Requirements Engineering,1994.

(Dem97) E. Demirors, "A Blackboard Framework for Supporting Teams in Software Development," presented at Ninth IEEE International Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute, 1997.

(Die95) M. Diepstraten, "Command and Control System Requirements Analysis and System Requirements Specification for a Tactical System," presented at First IEEE International Conference on Engineering of complex Computer Systems, 1995.

(Dob94) J. Dobson and R. Strens, "Organizational Requirements Definition for Information Technology," presented at IEEE International Conference on Requirements Engineering, 1994.

(Duf95) D. Duffy et al., "A Framework for Requirements Analysis Using Automated Reasoning," presented at Seventh International Conference on Advanced Information Systems Engineering, 1995.

(Eas95) S. Easterbrook and B. Nuseibeh, "Managing Inconsistencies in an Evolving Specification," presented at Second International Symposium on Requirements Engineering, 1995.

(Edw95) M. Edwards et al., "RECAP: A Requirements Elicitation, Capture, and Analysis Process Prototype Tool for Large Complex Systems," presented at First IEEE International Conference on Engineering of Complex Computer Systems, 1995.

(ElE95) K. El-Emam and N. Madhavji, "Requirements Engineering Practices in Information Systems Development: A Multiple Case Study," presented at Second International Symposium on Requirements Engineering, 1995.

(Fai97) R. Fairley and R. Thayer, "The Concept of Operations: The Bridge from Operational Requirements to Technical Specifications," *Annals of Software Engineering*, vol. 3, 1997.

(Fic95) S. Fickas and M. Feather, "Requirements Monitoring in Dynamic Environments," presented at Second International Symposium on Requirements Engineering, 1995.

(Fie95) R. Fields et al., "A Task-Centered Approach to Analyzing Human Error Tolerance Requirements," resented at Second International Symposium on Requirements Engineering, 1995.

(Gha94) J. Ghajar-Dowlatshahi and A. Varnekar, "Rapid Prototyping in Requirements Specification Phase of Software Systems," presented at Fourth International Symposium on Systems Engineering, National Council on Systems Engineering, 1994.

(Gib95) M. Gibson, "Domain Knowledge Reuse During Requirements Engineering," presented at Seventh International Conference on Advanced Information Systems Engineering (CAiSE '95), 1995.

(Gol94) L. Goldin and D. Berry, "AbstFinder: A Prototype Abstraction Finder for Natural Language Text for Use in Requirements Elicitation: Design, Methodology and Evaluation," presented at IEEE International Conference on Requirements Engineering, 1994.

(Got97) O. Gotel and A. Finkelstein, "Extending Requirements Traceability: Lessons Learned from an Industrial Case Study," presented at IEEE International Symposium on Requirements Engineering, 1997.

(Hei96) M. Heimdahl, "Errors Introduced during the TACS II Requirements Specification Effort: A Retrospective Case Study," presented at Eighteenth IEEE International Conference on Software Engineering, 1996.

(Hei96a) C. Heitmeyer et al., "Automated Consistency Checking Requirements Specifications," *ACM Transactions on Software Engineering and Methodology*, vol. 5, iss. 3, July 1996, pp. 231-261.

(Hol95) K. Holtzblatt and H. Beyer, "Requirements Gathering: The Human Factor," *Communications of the ACM*, vol. 38, iss. 5, May 1995, pp. 31-32.

(Hud96) E. Hudlicka, "Requirements Elicitation with Indirect Knowledge Elicitation Techniques: Comparison of Three Methods," presented at Second IEEE International Conference on Requirements Engineering, 1996.

(Hug94) K. Hughes et al., "A Taxonomy for Requirements Analysis Techniques," presented at IEEE International Conference on Requirements Engineering, 1994.

(Hug95) J. Hughes et al., "Presenting Ethnography in the Requirements Process," presented at Second IEEE International Symposium on Requirements Engineering, 1995.

(Hut94) A.T.F. Hutt, ed., *Object Analysis and Design - Comparison of Methods. Object Analysis and Design - Description of Methods*, John Wiley & Sons, 1994.(INCOSE00) INCOSE, *How To: Guide for all Engineers, Version 2*, International Council on Systems Engineering, 2000.

(Jac95) M. Jackson, *Software Requirements and Specifications*, Addison-Wesley, 1995.

(Jac97) M. Jackson, "The Meaning of Requirements," *Annals of Software Engineering*, vol. 3, 1997.

(Jon96) S. Jones and C. Britton, "Early Elicitation and Definition of Requirements for an Interactive Multimedia Information System," presented at Second IEEE International Conference on Requirements Engineering, 1996.

(Kir96) T. Kirner and A. Davis, "Nonfunctional Requirements for Real-Time Systems," *Advances in Computers*, 1996.

(Kle97) M. Klein, "Handling Exceptions in Collaborative Requirements Acquisition," presented at IEEE International Symposium on Requirements Engineering,
1997.

(Kos97) R. Kosman, "A Two-Step Methodology to Reduce Requirements Defects," *Annals of Software Engineering*, vol. 3, 1997.

(Kro95) J. Krogstie et al., "Towards a Deeper Understanding of Quality in Requirements Engineering," presented at Seventh International Conference on Advanced Information Systems Engineering (CAiSE '95), 1995.

(Lal95) V. Lalioti and B. Theodoulidis, "Visual Scenarios for Validation of Requirements Specification," presented at Seventh International Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute, 1995.

(Lam95) A. v. Lamsweerde et al., "Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt," presented at Second International Symposium on Requirements Engineering, 1995.

(Lei97) J. Leite et al., "Enhancing a Requirements Baseline with Scenarios," presented at IEEE International Symposium on Requirements Engineering, 1997.

(Ler97) F. Lerch et al.., "Using Simulation-Based Experiments for Software Requirements Engineering," *Annals of Software Engineering*, vol. 3, 1997.

(Lev94) N. Leveson et al., "Requirements Specification or Process-Control Systems," *IEEE Transactions on Software Engineering*, vol. 20, iss. 9, September 1994, pp. 684-707.

(Lut96a) R. Lutz and R. Woodhouse, "Contributions of SFMEA to Requirements Analysis," presented at Second IEEE International Conference on Requirements Engineering, 1996.

(Lut97) R. Lutz and R. Woodhouse, "Requirements Analysis Using Forward and Backward Search," *Annals of Software Engineering*, vol. 3, 1997.

(Mac96) L. Macaulay, *Requirements Engineering*, Springer-Verlag, 1996.

(Mai95) N. Maiden et al., "Computational Mechanisms for Distributed Requirements Engineering," presented at Seventh International Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute, 1995.

(Mar94) B. Mar, "Requirements for Development of Software Requirements," presented at Fourth International Symposium on Systems Engineering, 1994.

(Mas97) P. Massonet and A. v. Lamsweerde, "Analogical Reuse of Requirements Frameworks," presented at IEEE International Symposium on Requirements Engineering, 1997.

(McF95) I. McFarland and I. Reilly, "Requirements Traceability in an Integrated Development Environment," presented at Second International Symposium on Requirements Engineering, 1995.

(Mea94) N. Mead, "The Role of Software Architecture in Requirements Engineering," presented at IEEE International Conference on Requirements Engineering, 1994.

(Mos95) D. Mostert and S. v. Solms, "A Technique to Include Computer Security, Safety, and Resilience Requirements as Part of the Requirements Specification," *Journal of Systems and Software*, vol. 31, iss. 1, October 1995, pp. 45-53.

(Myl95) J. Mylopoulos et al., "Multiple Viewpoints Analysis of Software Specification Process," *IEEE Transactions on Software Engineering*, 1995.

(Nis92) K. Nishimura and S. Honiden, "Representing and Using Non-Functional Requirements: A Process-

Oriented Approach," *IEEE Transactions on Software Engineering*, December 1992.

(Nis97) H. Nissen et al., "View-Directed Requirements Engineering: A Framework and Metamodel," presented at Ninth IEEE International Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute, 1997.

(OBr96) L. O'Brien, "From Use Case to Database: Implementing a Requirements Tracking System," *Software Development*, vol. 4, iss. 2, February 1996, pp. 43-47.

(UML04) Object Management Group, *Unified Modeling Language*, www.uml.org, 2004. (Opd94) A. Opdahl, "Requirements Engineering for Software Performance," presented at International Workshop on Requirements Engineering: Foundations of Software Quality, 1994.

(Pin96) F. Pinheiro and J. Goguen, "An Object-Oriented Tool for Tracing Requirements," *IEEE Software*, vol. 13, iss. 2, March 1996, pp. 52-64.

(Pla96) G. Playle and C. Schroeder, "Software Requirements Elicitation: Problems, Tools, and Techniques," *Crosstalk: The Journal of Defense Software Engineering*, vol. 9, iss. 12, December 1996, pp. 19-24.

(Poh94) K. Pohl et al., "Applying AI Techniques to Requirements Engineering: The NATURE Prototype," presented at IEEE Workshop on Research Issues in the Intersection Between Software Engineering and Artificial Intelligence, 1994.

(Por95) A. Porter et al., "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment," *IEEE Transactions on Software Engineering*, vol. 21, iss. 6, June 1995, pp. 563-575.

(Pot95) C. Potts et al., "An Evaluation of Inquiry-Based Requirements Analysis for an Internet Server," presented at Second International Symposium on Requirements Engineering, 1995.

(Pot97) C. Potts and I. Hsi, "Abstraction and Context in Requirements Engineering: Toward a Synthesis," *Annals of Software Engineering*, vol. 3, 1997.

(Pot97a) C. Potts and W. Newstetter, "Naturalistic Inquiry and Requirements Engineering: Reconciling Their Theoretical Foundations," presented at IEEE International Symposium on Requirements Engineering, 1997.

(Ram95) B. Ramesh et al., "Implementing Requirements Traceability: A Case Study," presented at Second International Symposium on Requirements Engineering, 1995.

(Reg95) B. Regnell et al., "Improving the Use Case Driven Approach to Requirements Engineering," presented at Second IEEE International Symposium on Requirements Engineering, 1995.

(Reu94) H. Reubenstein, "The Role of Software Architecture in Software Requirements Engineering," presented at IEEE International Conference on Requirements Engineering, 1994.

(Rob94) J. Robertson and S. Robertson, *Complete Systems Analysis*, Vol. 1 and 2, Prentice Hall, 1994.

(Rob94a) W. Robinson and S. Fickas, "Supporting Multi- Perspective Requirements Engineering," presented at IEEE International Conference on Requirements Engineering, 1994.

(Ros98) L. Rosenberg, T.F. Hammer, and L.L. Huffman, "Requirements, testing and metrics," presented at 15th Annual Pacific Northwest Software Quality Conference, 1998.

(Sch94) W. Schoening, "The Next Big Step in Systems Engineering Tools: Integrating Automated Requirements Tools with Computer Simulated Synthesis and Test," presented at Fourth International Symposium on Systems Engineering, 1994.

(She94) M. Shekaran, "The Role of Software Architecture in Requirements Engineering," presented at IEEE International Conference on Requirements Engineering, 1994.

(Sid97) J. Siddiqi et al., "Towards Quality Requirements Via Animated Formal Specifications," *Annals of Software Engineering*, vol. 3, 1997.

(Span97) G. Spanoudakis and A. Finkelstein, "Reconciling Requirements: A Method for Managing Interference, Inconsistency, and Conflict," *Annals of Software Engineering*, vol. 3, 1997.

(Ste94) R. Stevens, "Structured Requirements," presented at Fourth International Symposium on Systems Engineering, 1994.

(Vin90) W.G. Vincenti, *What Engineers Know and How They Know It - Analytical Studies form Aeronautical History*, John Hopkins University Press, 1990.

(Wei03) K. Weigers, *Software Requirements*, second ed., Microsoft Press, 2003.

(Whi95) S. White and M. Edwards, "A Requirements Taxonomy for Specifying Complex Systems," presented at First IEEE International Conference on Engineering of Complex Computer Systems, 1995.

(Wil99) B. Wiley, *Essential System Requirements: A Practical Guide to Event-Driven Methods*, Addison-Wesley, 1999.

(Wyd96) T. Wyder, "Capturing Requirements With Use Cases," *Software Development*, vol. 4, iss. 2, February 1996, pp. 36-40.

(Yen97) J. Yen and W. Tiao, "A Systematic Tradeoff Analysis for Conflicting Imprecise Requirements," presented at IEEE International Symposium on Requirements Engineering, 1997.

(Yu97) E. Yu, "Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering," presented at IEEE International Symposium on Requirements Engineering, 1997.

(Zav96) P. Zave and M. Jackson, "Where Do Operations Come From? A Multiparadigm Specification Technique," *IEEE Transactions on Software Engineering,*, vol. 22, iss. 7, July 1996, pp. 508-528.

APÉNDICE B. LISTA DE ESTANDARS

(IEEE830-98) IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE, 1998.

(IEEE1028-97) IEEE Std 1028-1997 (R2002), *IEEE Standard for Software Reviews*, IEEE, 1997.

(IEEE1233-98) IEEE Std 1233-1998, *IEEE Guide for Developing System Requirements Specifications*, 1998.

(IEEE1320.1-98) IEEE Std 1320.1-1998, *IEEE Standard for Functional Modeling Language-Syntax and Semantics for IDEF0*, IEEE, 1998.

(IEEE1320.2-98) IEEE Std 1320.2-1998, *IEEE Standard for Conceptual Modeling Language-Syntax and Semantics for IDEFIX97 (IDEFObjetct)*, IEEE, 1998.

(IEEE1362-98) IEEE Std 1362-1998, *IEEE Guide for Information Technology-System Definition-Concept of Operations (ConOps) Document*, IEEE, 1998.

(IEEE1465-98) IEEE Std 1465-1998//ISO/ IEC12119:1994, *IEEE Standard Adoption of International Standard ISO/IEC12119:1994(E), Information Technology-Software Packages-Quality requirements and testing*, IEEE, 1998.

(IEEEP1471-00) IEEE Std 1471-2000, *IEEE Recommended Practice for Architectural Descriptionos Software Intensive Systems*, Architecture Working Group of the Software Engineering Standards Committee, 2000.

(IEEE12207.0-96) IEEE/EIA 12207.0-1996//ISO/ IEC12207:1995, *Industry Implementation of Int. Std. ISO/IEC 12207:95, Standard for Information Technology- Software Life Cycle Processes*, IEEE, 1996.

(IEEE14143.1-00) IEEE Std 14143.1-2000//ISO/ IEC14143-1:1998, *Information Technology-Software Measurement-Functional Size Measurement-Part 1: Definitions of Concepts*, IEEE, 2000.