Requisitos

— Minino de Cheshire —empezó Alicia tímidamente,… — ¿podrías decirme, por favor, qué camino debo seguir para salir de aquí?
— Esto depende en gran parte del sitio al que quieras llegar —dijo el Gato.
— No me importa mucho el sitio... —dijo Alicia.
— Entonces tampoco importa mucho el camino que tomes —dijo el Gato.
—... siempre que llegue a alguna parte —añadió Alicia como explicación.
— ¡Oh, siempre llegarás a alguna parte —aseguró el Gato—, si caminas lo suficiente!

Lewis Carroll, "Alicia en el país de las maravillas"

Si no sabemos hacia dónde queremos ir, si no establecemos un destino (objetivo), ¿Cómo podremos encontrar los posibles caminos? ¿Cómo podremos elegir el mejor camino? ¿Cómo sabremos cuando hayamos llegado al destino fijado? Es necesario saber hacia dónde vamos y necesitamos los requisitos para llegar a destino. Los requisitos describen hacia dónde queremos ir, no cómo llegar allí.

Requerimiento versus Requisito

En la bibliografía ambos términos se suelen utilizar en forma indistinta como traducción de la palabra inglesa "requirement". En el siguiente texto vamos a utilizar:

- **REQUERIMIENTO** para referirnos a los pedidos que recibimos de los clientes. Entendiendo como cliente a aquel que solicita y define el sistema de software.

  Ej.: *Registrar los clientes del negocio*.

- **REQUISITO** para identificar las especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimiento y limitaciones. Esto es lo que el equipo de desarrollo se compromete a cumplir para construir el sistema de software que necesita el cliente para satisfacer sus requerimientos.

  Ej.: *El sistema debe registrar los clientes con los siguientes datos obligatorios: DNI (que servirá como clave de identificación única), nombres, apellidos, correo electrónico, teléfono, dirección. Si el DNI ya existe, el sistema debe mostrar los datos actualmente registrados con ese número.*

La diferenciación anterior estaría relacionada con la clasificación de Sommerville [Sommerville, 2005]:

- **Requisitos del usuario**: son declaraciones del usuario, en lenguaje natural, de lo que se espera que haga el sistema y de las restricciones bajo las cuales trabajará.
- **Requisitos del sistema**: definen con detalle cada una de las funcionalidades, servicios y restricciones del sistema a construir. Forman la especificación de requisitos del software.

Definiciones de Requisitos

Según Sommerville [Sommerville, 2005]**:**
Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. El término requisito no se utiliza de forma contante en la industria de software. En algunos casos, un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema.

Según el Instituto de Ingenieros en Electricidad y Electrónica (IEEE Institute of Electrical and Electronics Engineers) [IEEE 90] un requisito es:

1. Una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
2. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

Una representación documentada de una condición o capacidad como en 1 o 2.

Requisitos $\Longleftrightarrow$ Qué

Los requisitos representan el **Qué** del sistema, ¿qué se va a construir? No contemplan descripciones específicas referidas a la implementación del sistema, salvo, que el cliente lo pida expresamente.
Los requisitos tienen diferentes niveles de detalle a lo largo del proceso de especificación, siguiendo su descripción una evolución incremental. Comienzan con una *descripción abstracta y general* de una funcionalidad o servicio que debe brindar el sistema, o de una restricción del mismo, hasta aquí son *requisitos de alto nivel*. Al finalizar el proceso se convierten en una *definición detallada* y formal del mismo: *requisito de bajo nivel*.

Ejemplo de la evolución de un requerimiento, hasta convertirse en un requisito de bajo nivel

Requerimiento del cliente: Registrar los clientes del negocio.

Evolución del Requisito (pueden ser más o menos niveles)

- **Alto nivel**:
  El sistema debe registrar los clientes.
- **Nivel intermedio**:
  El sistema debe registrar los clientes guardando datos personales y datos que permitan ubicar al cliente.

- **Bajo Nivel**:

  El sistema debe registrar los clientes con los siguientes datos obligatorios: DNI (que servirá como clave de identificación única), nombres, apellidos, correo electrónico, teléfono de línea, teléfono celular, dirección. Si el DNI ya existe, el sistema debe mostrar los datos actualmente registrados con ese número.

Clasificación de Requisitos

Los requisitos pueden dividirse en **requisitos funcionales** y **requisitos no funcionales**.

> Los **requisitos funcionales** definen las funciones que el sistema será capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. En ocasiones también se documenta lo que el sistema no realizará.

Los **requisitos funcionales (RF)** son sentencias que describen las funciones o servicios que el sistema debería entregar. También describe como debería comportarse el sistema en situaciones particulares (casos de excepción). Es conveniente describir lo que el sistema debe realizar, solo en casos particulares deben utilizarse los requisitos funcionales negativos, para documentar aquellas funciones que quedan excluidas del sistema.
Los requisitos funcionales se pueden expresar de diferentes formas, utilizando diferentes formalismos que depende de las características de la organización. No importa el formalismo que se utilice, siempre debe quedar claramente definida la funcionalidad, las entradas, las salidas y si fuese necesario, las excepciones.
Un requisito funcional, es un requisito que obligatoriamente debe estar presente en el producto final, porque si no, el sistema no cumple su función.

> Los **requisitos no funcionales** tienen que ver con características que de una u otra forma puedan limitar al sistema. Son restricciones que afectan a los servicios o funciones del sistema.

Los **requisitos no funcionales (RNF)** como su nombre lo indica no se refieren directamente a las funcionalidades del sistema, sino que tienen que ver con características que de una u otra forma puedan limitar al sistema, como por ejemplo, el rendimiento, interfaces de usuario, fiabilidad, mantenimiento, seguridad, portabilidad, estándares, etc. Pueden referirse a características que afectan a todo el sistema, que afecten a uno o varios subsistemas, o que sólo se afectan a una función o un servicio del sistema (esto sucede raramente).
Son características que no hacen a la funcionalidad específica del sistema, pero si la pueden afectar. Hay que definirlas e implementarlas en el producto final.

Los requisitos, funcionales o no funcionales, deben poder verificarse que estén presentes una vez que el producto es finalizado y entregado al cliente. Verificar una "función → RF" del sistema es sencillo, se ve si el producto final cumple o no con la misma. Con las "restricciones → RNF" no es tan sencillo, todo depende de cómo definamos los mismos. Por lo tanto siempre debemos comprobar que todos los requisitos no funcionales los definamos de una forma cuantificable, caso contrario será ambigua la comprobación en el sistema entregado.

> Los **RNF** se deben definir de forma que sean **cuantificables**, para que puedan ser validados por el cliente y no tengan interpretaciones ambiguas.

Clasificación de los Requisitos No Funcionales

Diferentes autores clasifican los RNF de formas diversas, nosotros veremos la clasificación que hace de los mismos Sommerville [Sommerville, 2005]:

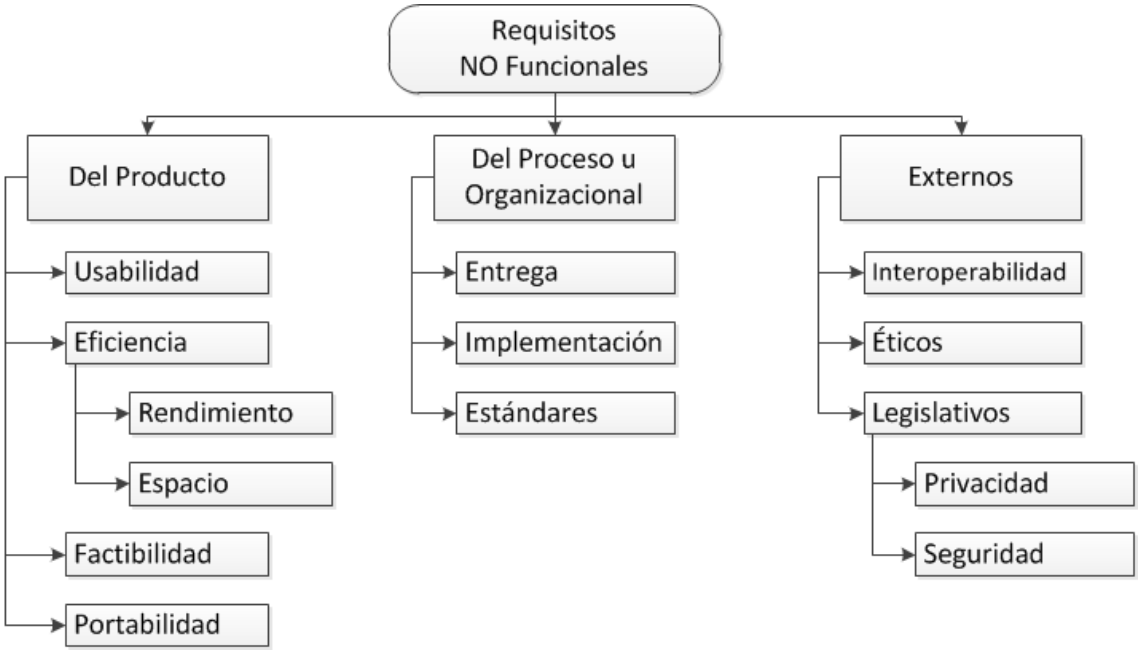

Figura 1 - Clasificación de los Requisitos No Funcionales

1. **Requisitos del Producto**: especifican que el producto entregado debe comportarse de una manera particular.
   a. **Usabilidad**: tienen que ver con características relacionadas con la facilidad de uso del producto. Algunas consideraciones para medir la usabilidad de un producto de software son:
      i. Especificar el tiempo de capacitación requerido para que usuarios normales y expertos para utilizar correctamente el sistema.

ii. Especificar requerimientos para conformidad con los estándares de Interfaz Grafica de Usuario (GUI).

iii. Especificar necesidades de documentación del producto, incluyendo los requisitos respecto al tipo y forma de presentación de documentación de usuario y sistemas de ayuda para el producto de software a construir (Manual de Usuario, Ayuda en Línea, Manual de Configuración).

iv. Para que sean cuantificables describir requisitos en función de Tiempo de entrenamiento, Número de marcos de ayuda, etc.

Ejemplo:

ERROR no es cuantificable**:**

- *El sistema será amigable***.**

La **forma correcta** de definirlo es cuantificando la característica de amigabilidad utilizada en el ejemplo anterior, por Ej.:

- *De cada 10 usuarios que utilicen el sistema 9 lo harán correctamente luego de un uso asistido por un instructor.*

b. **Eficiencia**: dar valores a los tiempos de respuesta a funciones específicas o medir capacidades del sistema.

i. **de Rendimiento** mide la rapidez de ejecución del sistema, por ej. El tiempo de respuesta para una transacción (promedio, máximo), Capacidad (el número de clientes o transacciones que el sistema puede soportar). Modos de Degradación (modo aceptable de operación cuando el sistema ha sido degradado). Los requisitos deben expresarse en término de:

- Procesamiento de transacciones/segundo
- Tiempo de respuesta usuario/evento
- Tiempo de actualización de la pantalla

Ejemplo:

ERROR no es cuantificable**:**

- *La terminal entrega rápido el dinero.*

La **forma correcta** de definirlo es cuantificando la característica de "rápido":

- *El sistema tarda como máximo 5 segundos en entregar el dinero al cliente, contados a partir del momento que haya finalizado de verificar que hay cambio suficiente en la terminal para satisfacer el pedido del cliente.*

ERROR es ambiguo**:**

- *El sistema debe soportar la carga de usuarios correspondiente al número medio de estudiantes.*

La **forma correcta** de definirlo es eliminando la ambigüedad, por Ej.:

- *El sistema tiene que poder soportar la carga de usuarios correspondiente al número medio de estudiantes que cursan el Ingreso a la universidad cada año.*

ii. **de Espacio** mide la cantidad de recursos utilizados por el sistema, por Ej. memoria requerida, espacio en disco, hardware de comunicaciones. Los requisitos deben expresarse en función de: K bytes, Número de tarjetas de memoria, etc., para que sean cuantificables.

Ejemplo:

ERROR no es cuantificable**:**

- *El sistema funciona con recursos mínimos.*

La **forma correcta** de definirlo es cuantificando la característica de "mínimos", por Ej.:

- *El sistema requiere 1 GB de memoria RAM y al menos 20 MB de espacio en el disco rígido.*

c. **Fiabilidad** fija la tasa de fallos para que el sistema sea aceptable. La confiabilidad podría expresarse en término de alguno de estos aspectos:

i. Disponibilidad: Especificar el porcentaje de disponibilidad de tiempo, horas de uso, acceso de mantenimiento, etc.

ii. Tiempo Mínimo entre fallas: Especificado usualmente en horas, pero también puede especificarse en días, meses y años.

iii. Tiempo Mínimo de Reparación: ¿Cuánto tiempo está permitido que el sistema esté fuera de operación después de una falla?

iv. Certeza: Precisión Específica (resolución) y certeza (sobre un estándar) que es requerida para las salidas del sistema.

v. Errores (bugs) Máximos o ratios de defecto: usualmente expresados en términos de BUGS/KLOC (miles de líneas de código) o bugs por casos de uso

vi. Errores (Bugs) o ratios de defectos: Categorizados en términos de bugs menores, significativos y críticos. Los requerimientos deberán definir lo que quiere decir bug "crítico" (tal como datos completamente perdidos, inhabilitación completa para usar ciertas partes de la funcionalidad del sistema).

vii. Backup: Backup requeridos para asegurar la disponibilidad del sistema.

viii. Estabilidad: capacidad del sistema de disminuir la cantidad de cambios realizados en un período de tiempo.

Ejemplo:

ERROR es ambiguo**:**

- *El sistema tiene que estar disponible 24 horas.*

La **forma correcta** es eliminar la ambigüedad, por Ej.:

- *El sistema tiene que estar disponible 100% del tiempo. Las 24 horas del día los 7 días de la semana.*

d. **Portabilidad** debe expresar las necesidades de crecimiento del producto hacia otras tecnologías de desarrollo, sistemas operativos y/o plataformas de hardware. Considera la facilidad de reemplazo por nuevas versiones del producto, la facilidad de instalación, la escalabilidad y la adaptabilidad a diferentes plataformas. La portabilidad podría expresarse en término de alguno de estos aspectos:

i. Porcentaje de declaraciones dependientes de entrada

ii. Número de sistemas objetivos

Ejemplo:

ERROR es ambiguo**:**

- *El sistema debe visualizarse y funcionar correctamente en cualquier navegador.*

La **forma correcta** es eliminar la ambigüedad, por Ej.:

- *El sistema se visualizará y funcionará correctamente en Internet Explorer 8, Google Chrome v.23 y Firefox v.17.*

2. **Requerimientos organizacionales** Estos requerimientos se derivan de políticas y procedimientos existentes en la organización del cliente y en la del desarrollador.

a. **de Entrega** especifican cuándo se entregará el producto y su documentación. Si la organización tiene requisitos explícitos respecto a la entrega del producto, entre los cuales podemos mencionar, fechas, épocas del año, días u horas específicos para por hacer el despliegue del producto, etc., deberán especificarse en este apartado.

Ejemplo:

ERROR es ambiguo**:**

- *El sistema debe entregarse en fecha.*

La **forma correcta** es eliminar la ambigüedad, por Ej.:

- *El sistema debe entregarse dentro de los 180 días hábiles contados a partir de la fecha de la firma del contrato.*

b. **de Implementación** hacen referencia a los lenguajes de programación o el método de diseño a utilizar. Este apartado deberá especificar cualquier consideración que impacte en la implementación del producto que sea un requisito planteado por el cliente y que el producto debe respetar. Ejemplos de

esto sería la definición de utilizar cierta base de datos o cierto lenguaje de Programación

Ejemplo:

- *El software se escribirá en lenguaje COBOL.*

c. **Estándares** algunos ejemplos son los estándares en los procesos que deben utilizarse. Si la organización contratante (cliente) desea que el producto respete ciertos estándares asociados al producto en sí mismo o a su proceso de desarrollo, los mismos deberán especificarse en este apartado.

Ejemplo:

- *El proceso de desarrollo debe ser conforme a la norma ISO 9003.*

3. **Requerimientos externos** Este gran apartado incluye todos los requerimientos que se derivan de los factores externos al sistema y de su proceso de desarrollo.

a. **Requerimientos de Interoperabilidad** que definen la manera en que el sistema interactúa con sistemas de otras organizaciones. Este aspecto implica requisitos vinculados con la necesidad de que el producto de software se comunique con otros productos de software del exterior, para intercambiar datos o algún otro aspecto.

Ejemplo:

- *La aplicación utiliza la interfaz definida en el anexo X, para conectarse al sistema XYZ de la casa matriz.*

b. **Requerimientos Éticos**. Estos son puestos en un sistema para asegurar que será aceptado por sus usuarios y por el público en general. Si existen requerimientos que deben considerarse en el contexto del producto que si bien no están legislados, responde a factores morales o pautas de conducta, deberán especificarse aquí.

Ejemplo en un sistema contable**:**

- *El software no realiza doble registro de datos.*

c. **Requerimientos Legislativos** Hacen referencia a los requerimientos legales que protegen la seguridad y confidencialidad del producto y toda documentación asociada. Deben seguirse para asegurar que el sistema funcione dentro de la ley. Debemos identificar si existen legislaciones nacionales, internacionales, provinciales, etc., aplicables y vigentes, que el software deba considerar. También incluye los aspectos relacionados a los derechos de copia (copyright).

Ejemplo:

- *Se implementa lo establecido en la Ley 25.326 para la protección de los datos personales que administra el sistema.*

Propiedades y Atributos de los requisitos [Hadad, 2003]

Las principales propiedades o cualidades que deben presentar los requisitos son:

- **Correctos**: los requisitos deben ser realmente necesarios y cumplir con el propósito del sistema.
- **Conciso**: Un requisito es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- **Consistentes**: los requisitos no deben estar en conflicto entre ellos.
- **Completos**: los requisitos deben abarcar toda posible entrada al sistema y toda posible respuesta del sistema.
- **Entendibles**: los requisitos deben poder ser comprendidos por cualquier involucrado con un mínimo de explicación.
- **No ambiguo**: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector
- **Realizables**: los requisitos deben poder satisfacerse considerando los recursos disponibles y restricciones, es decir, los requisitos deben ser posibles de llevarse a cabo.
- **Rastreables**: los requisitos deben poder relacionarse bidireccionalmente a las fuentes que les dieron origen, y también a los modelos, documentos y componentes del software.
- **Verificables**: los requisitos deben poder comprobarse a través del software.
- **Abstractos**: los requisitos deben ser independientes de la implementación.
- **Referenciables por importancia y/o estabilidad**: los requisitos deben permitir que se les asignen prioridad y grado de estabilidad que presentan en el tiempo.

Especificación de Requisitos de Software (ERS) o Software Requirement Specification (SRS)

Durante la Adquisición de Conocimientos, estudiamos el dominio y comprendimos el problema. Para lograrlo identificamos y priorizamos las fuentes de información, mediante diferentes técnicas extrajimos datos partir de los documentos y educimos información del cliente, de los expertos y de los usuarios del sistema. Con esto adquirimos los conocimientos necesarios para analizar el problema. Analizamos toda la información obtenida.

A partir de los requerimientos de los clientes, definimos los requisitos del sistema. Los requisitos son las necesidades del producto que se debe desarrollar. Por ello, en la fase de análisis se deben identificar claramente estas necesidades y documentarlas. Como resultado de esta fase se debe producir un documento de especificación de requisitos en el que se describa lo que el futuro sistema debe hacer. Por tanto, no se trata simplemente de una actividad de análisis, sino también de síntesis.

La **Especificación de Requisitos de Software** (ERS) es una "colección estructurada de información, que contiene los requisitos del sistema". [IEEE Std 1233, Edición 1998]

La ERS está dirigida a tanto al cliente, que es quien solicitó y definió el sistema, como al desarrollador, que es quien lo debe desarrollar. Por lo tanto, el lenguaje utilizado para la documentación de los requisitos estará en función del nivel que el usuario tenga para entender dichas especificaciones.
Una ERS forma parte de la documentación asociada al software que se está desarrollando, por tanto debe definir correctamente todos los requerimientos, pero no más de los necesarios. Esta documentación no debería describir ningún detalle de diseño, modo de implementación o gestión del proyecto, ya que los requisitos se deben describir de forma que el usuario pueda entenderlos. Al mismo tiempo, se da una mayor flexibilidad a los desarrolladores para la implementación.
La ERS forma parte del contrato entre cliente y desarrolladores. El base para la estimación de costos y planificación. Es un punto de referencia para procesos de verificación y validación, y aceptación del producto final. Es una base para la identificación de posibles mejoras en los procesos analizados.
En diferentes organizaciones y en la literatura misma, este documento recibe variados nombres, desde documento de requisitos (el nombre más informal) hasta especificación de requisitos de software (SRS Software Requirement Specification) (el nombre más formal), y se genera con diversas estructuras aunque existen estándares internacionales que especifican su estructura, tales como [IEEE Std 830-1998], [ESA PSS-05-0] de la agencia espacial europea, [NCC 87] entre otros. [Hadad 2008]

Características de una buena especificación de requerimientos

El estándar IEEE 830-1998 sobre las prácticas recomendadas define ocho características de una buena especificación de requerimientos:

1. **Correcta**: El estándar asegura que no hay herramienta o proceso que asegure la correctitud de la especificación de los requerimientos. Un algoritmo es correcto si la salida de éste se apega a su especificación. Extrapolando, podríamos decir que una especificación es correcta si el producto final se apega a la especificación.
2. **No ambigua**: Una especificación no es ambigua si cada requerimiento declarado tiene una sola interpretación. Como utilizamos el lenguaje del cliente, cuanto menos formal sea el lenguaje, mayores van a ser estos problemas. A la inversa, cuanto más formal sea el lenguaje, más difícil de comprender será para los clientes que deban validarla.
3. **Completa**: Una especificación está completa si no hay requerimientos con elementos a ser definidos, todos los requerimientos significativos están expresados y todas las referencias están definidas.
4. **Consistente**: La especificación de requerimientos debe ser consistente internamente (entre los propios requerimientos) y externa (con otros documentos de especificación).

5. **Priorizable**: La priorización ayuda a las partes interesadas a establecer cuáles son los requerimientos claves para el negocio. Dentro de una clasificación previa, se puede dividir los requerimientos entre esenciales, condicionales y opcionales, y en cada lista ordenarlos por prioridad.
6. Verificable: Un requerimiento es verificable si existe algún método finito y sin costo excesivo para probarlo.
7. **Modificable**: Si bien todo requerimiento es potencialmente modificable, con esta propiedad nos referimos a que debe ser sencillo modificarlos, a la vez que se mantienen los demás atributos (como completitud y correctitud). Deberán tener referencias cuando se relacionen con otros requerimientos, no ser redundantes, y ser atómicos con respecto a la funcionalidad que se especifica. Evaluar que un requerimiento sea modificable debería depender de la naturaleza del dominio tratado.
8. **Trazable**: La trazabilidad es la habilidad de verificar la historia, ubicación o aplicación de un ítem a través de su identificación almacenada y documentada. El documento habla sobre trazabilidad para atrás (hacia etapas más tempranas de desarrollo) y hacia delante (documentos derivados de la SRS, como por ejemplo, casos de uso).

En 1993 Davis [Davis 93] tomando como punto de partida la lista anterior, generó una lista de 24 propiedades que debe presentar un SRS de calidad:

1. no ambiguo,
2. completo,
3. correcto,
4. entendible,
5. verificable,
6. consistente internamente,
7. consistente externamente,
8. alcanzable (realizable),
9. conciso,
10. independiente de diseño,
11. rastreable,
12. modificable,
13. almacenable electrónicamente,
14. ejecutable / interpretable,
15. registrable por importancia relativa,
16. registrable por estabilidad relativa,
17. registrable por versión,
18. no redundante,
19. en el nivel de detalle adecuado,
20. preciso,
21. reusable,
22. remontado a su origen,
23. organizado y
24. referenciado cruzadamente.

Anexo [IEEE Std 830-1998]

Especificación de Requisitos según el estándar de IEEE 830

IEEE Std. 830-1998
22 de Octubre de 2008

Resumen

Este documento presenta, en castellano, el formato de Especificación de Requisitos Software (ERS) según la última versión del estándar IEEE 830. Según IEEE, un buen Documento de Requisitos, pese a no ser obligatorio que siga estrictamente la organización y el formato dados en el estándar 830, sí debería incluir, de una forma o de otra, toda la información presentada en dicho estándar. El estándar de IEEE 830 no está libre de defectos ni de prejuicios, y por ello ha sido justamente criticado por múltiples autores y desde múltiples puntos de vista, llegándose a cuestionar incluso si es realmente un estándar en el sentido habitual que tiene el término en otras ingenieras. El presente documento no pretende pronunciarse ni a favor ni en contra de unos u otros: tan sólo reproduce, con propósitos fundamentalmente docentes, cómo se organizara un Documento de Requisitos según el estándar IEEE 830.

ÍNDICE

1 INTRODUCCIÓN

En esta sección se proporcionaría una introducción a todo el documento de Especificación de Requisitos Software (ERS). Consta de varias subsecciones: propósito, ámbito del sistema, definiciones, referencias y visión general del documento.

1.1. Propósito

En esta subsección se definirá el propósito del documento ERS y se especificará a quién va dirigido el documento.

1.2. Ámbito del Sistema

- En esta subsección:
- Se podrá dar un nombre al futuro sistema (por Ej. MiSistema)
- Se explicará lo que el sistema hará y lo que no hará.
- Se describirán los beneficios, objetivos y metas que se espera alcanzar con el futuro sistema.
- Se referenciarán todos aquellos documentos de nivel superior (por Ej. en Ingeniería de Sistemas, que incluyen Hardware y Software, deberá mantenerse la consistencia con el documento de especificación de requisitos globales del sistema, si existe).

1.3. Definiciones, Acrónimos y Abreviaturas

En esta subsección se definirán todos los términos, acrónimos y abreviaturas utilizadas en la ERS.

1.4. Referencias

En esta subsección se mostrará una lista completa de todos los documentos referenciados en la ERS.

1.5. Visión General del Documento

Esta subsección describe brevemente los contenidos y la organización del resto de la ERS.

2 DESCRIPCIÓN GENERAL

En esta sección se describen todos aquellos factores que afectan al producto y a sus requisitos. No se describen los requisitos, sino su contexto. Esto permitirá definir con detalle los requisitos en la sección 3, haciendo que sean más fáciles de entender.
Normalmente, esta sección consta de las siguientes subsecciones: Perspectiva del producto, funciones del producto, características de los usuarios, restricciones, actores que se asumen y futuros requisitos.

2.1. Perspectiva del Producto

Esta subsección debe relacionar el futuro sistema (producto software) con otros productos. Si el producto es totalmente independiente de otros productos, también debe especificarse aquí. Si la ERS define un producto que es parte de un sistema

mayor, esta subsección relacionará los requisitos del sistema mayor con la funcionalidad del producto descrito en la ERS, y se identificarán las interfaces entre el producto mayor y el producto aquí descrito. Se recomienda utilizar diagramas de bloques.

2.2. Funciones del Producto

En esta subsección de la ERS se mostrará un resumen, a grandes rasgos, de las funciones del futuro sistema. Por ejemplo, en una ERS para un programa de contabilidad, esta subsección mostrará que el sistema soportará el mantenimiento de cuentas, mostrará el estado de las cuentas y facilitará la facturación, sin mencionar el enorme detalle que cada una de estas funciones requiere.
Las funciones deberán mostrarse de forma organizada, y pueden utilizarse gráficos, siempre y cuando dichos gráficos reflejen las relaciones entre funciones y no el diseño del sistema.

2.3. Características de los Usuarios

Esta subsección describirá las características generales de los usuarios del producto, incluyendo nivel educacional, experiencia y experiencia técnica.

2.4. Restricciones

Esta subsección describirá aquellas limitaciones que se imponen sobre los desarrolladores del producto:

- Políticas de la empresa
- Limitaciones del hardware
- Interfaces con otras aplicaciones
- Operaciones paralelas
- Funciones de auditora
- Funciones de control
- Lenguaje(s) de programación
- Protocolos de comunicación
- Requisitos de habilidad
- Criticidad de la aplicación
- Consideraciones acerca de la seguridad

2.5. Suposiciones y Dependencias

Esta subsección de la ERS describirá aquellos factores que, si cambian, pueden afectar a los requisitos. Por ejemplo, los requisitos pueden presuponer una cierta organización de ciertas unidades de la empresa, o pueden presuponer que el sistema correrá sobre cierto sistema operativo. Si cambian dichos detalles en la organización de la empresa, o si cambian ciertos detalles técnicos, como el sistema operativo, puede ser necesario revisar y cambiar los requisitos.

2.6. Requisitos Futuros

Esta subsección esbozará futuras mejoras al sistema, que podrán analizarse e implementarse en un futuro.

3 REQUISITOS ESPECÍFICOS

Esta sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores diseñar un sistema que satisfaga estos requisitos, y que permita al equipo de pruebas planificar y realizar las pruebas que demuestren si el sistema satisface, o no, los requisitos. Todo requisito aquí especificado describirá comportamientos externos del sistema, perceptibles por parte de los usuarios, operadores y otros sistemas. Esta es la sección más larga e importante de la ERS. Deberán aplicarse los siguientes principios:

- El documento deberá ser perfectamente legible por personas de muy distintas formaciones e intereses.
- Deberán referenciarse aquellos documentos relevantes que poseen alguna influencia sobre los requisitos.
- Todo requisito deberá ser unívocamente identificable mediante algún código o sistema de numeración adecuado.
- Lo ideal, aunque en la práctica no siempre realizable, es que los requisitos posean las siguientes características:
    - Corrección: La ERS es correcta si y sólo si todo requisito que figura aquí (y que será implementado en el sistema) refleja alguna necesidad real. La corrección de la ERS implica que el sistema implementado será el sistema deseado.
    - No ambiguos: Cada requisito tiene una sola interpretación. Para eliminar la ambigüedad inherente a los requisitos expresados en lenguaje natural, se deberán utilizar gráficos o notaciones formales. En el caso de utilizar términos que, habitualmente, poseen más de una interpretación, se definirán con precisión en el glosario.
    - Completos: Todos los requisitos relevantes han sido incluidos en la ERS. Conviene incluir todas las posibles respuestas del sistema a los datos de entrada, tanto válidos como no válidos.
    - Consistentes: Los requisitos no pueden ser contradictorios. Un conjunto de requisitos contradictorio no es implementable.
    - Clasificados: Normalmente, no todos los requisitos son igual de importantes. Los requisitos pueden clasificarse por importancia (esenciales, condicionales u opcionales) o por estabilidad (cambios que se espera que afecten al requisito). Esto sirve, ante todo, para no emplear excesivos recursos en implementar requisitos no esenciales.
    - Verificables: La ERS es verificable si y sólo si todos sus requisitos son verificables. Un requisito es verificable (testeable) si existe un proceso finito y no costoso para demostrar que el sistema cumple con el requisito. Un requisito ambiguo no es, en general, verificable. Expresiones como: “a veces”, “bien”, “adecuado”, etc. Introducen ambigüedad en los requisitos. Requisitos como “en caso de accidente la

nube toxica no se extenderá más allá de 25Km" no es verificable por el alto costo que conlleva.

- Modificables: La ERS es modificable si y sólo si se encuentra estructurada de forma que los cambios a los requisitos pueden realizarse de forma fácil, completa y consistente. La utilización de herramientas automáticas de gestión de requisitos (por ejemplo RequisitePro o Doors) facilitan enormemente esta tarea.
- Trazables: La ERS es trazable si se conoce el origen de cada requisito y se facilita la referencia de cada requisito a los componentes del diseño y de la implementación. La trazabilidad hacia atrás indica el origen (documento, persona, etc.) de cada requisito. La trazabilidad hacia delante de un requisito R indica qué componentes del sistema son los que realizan el requisito R.

3.1. Interfaces Externas

Se describirán los requisitos que afecten a la interfaz de usuario, interfaz con otros sistemas (hardware y software) e interfaces de comunicaciones.

3.2. Funciones

Esta subsección (quizá la más larga del documento) deberá especificar todas aquellas acciones (funciones) que deberá llevar a cabo el software. Normalmente (aunque no siempre), son aquellas acciones expresables como “el sistema deberá. . . ". Si se considera necesario, podrán utilizarse notaciones graficas y tablas, pero siempre supeditadas al lenguaje natural, y no al revés.

Es importante tener en cuenta que, en 1983, el Estándar de IEEE 830 establecía que las funciones deberían expresarse como una jerarquía funcional (en paralelo con los DFDs propuestos por el análisis estructurado). Pero el Estándar de IEEE 830, en sus últimas versiones, ya permite organizar esta subsección de múltiples formas, y sugiere, entre otras, las siguientes:

- Por tipos de usuario: Distintos usuarios poseen distintos requisitos. Para cada clase de usuario que exista en la organización, se especificarán los requisitos funcionales que le afecten o tengan mayor relación con sus tareas.
- Por objetos: Los objetos son entidades del mundo real que serán reflejadas en el sistema. Para cada objeto, se detallarán sus atributos y sus funciones. Los objetos pueden agruparse en clases. Esta organización de la ERS no quiere decir que el diseño del sistema siga el paradigma de Orientación a Objetos.
- Por objetivos: Un objetivo es un servicio que se desea que ofrezca el sistema y que requiere una determinada entrada para obtener su resultado. Para cada objetivo o sub-objetivo que se persiga con el sistema, se detallarán las funciones que permitan llevarlo a cabo.
- Por estímulos: Se especificarán los posibles estímulos que recibe el sistema y las funciones relacionadas con dicho estímulo.
- Por jerarquía funcional: Si ninguna de las anteriores alternativas resulta de ayuda, la funcionalidad del sistema se especificará como una jerarquía de funciones que comparten entradas, salidas o datos internos. Se detallarán las funciones (entrada, proceso, salida) y las subfunciones del sistema. Esto no

implica que el diseño del sistema deba realizarse según el paradigma de Diseño Estructurado.

Para organizar esta subsección de la ERS se elegirá alguna de las anteriores alternativas, o incluso alguna otra que se considere más conveniente. Deberá, eso sí, justificarse el porqué de tal elección.

3.3. Requisitos de Rendimiento

Se detallarán los requisitos relacionados con la carga que se espera tenga que soportar el sistema. Por ejemplo, el número de terminales, el número esperado de usuarios simultáneamente conectados, número de transacciones por segundo que deberá soportar el sistema, etc.
También, si es necesario, se especificarán los requisitos de datos, es decir, aquellos requisitos que afecten a la información que se guardará en la base de datos. Por ejemplo, la frecuencia de uso, las capacidades de acceso y la cantidad de registros que se espera almacenar (decenas, cientos, miles o millones).

3.4. Restricciones de Diseño

Todo aquello que restrinja las decisiones relativas al diseño de la aplicación: Restricciones de otros estándares, limitaciones del hardware, etc.

3.5. Atributos del Sistema

Se detallarán los atributos de calidad del sistema: Fiabilidad, mantenibilidad, portabilidad, y, muy importante, la seguridad. Deberá especificarse qué tipos de usuario están autorizados, o no, a realizar ciertas tareas, y cómo se implementarán los mecanismos de seguridad (por ejemplo, por medio de un login y una password).

3.6. Otros Requisitos

Cualquier otro requisito que no encaje en otra sección.

4. APÉNDICES

Pueden contener todo tipo de información relevante para la ERS pero que, propiamente, no forme parte de la ERS. Por ejemplo:

1. Formatos de entrada/salida de datos, por pantalla o en listados.
2. Resultados de análisis de costes.
3. Restricciones acerca del lenguaje de programación.

NOTA:

REFERENCIAS – GLOSARIO Ver Apunte: “Adquisición de Conocimientos”