Ingeniería del Software I

Guía de Estudio
Especificación de requisitos

Docentes: Dr. Dario Rodriguez
Mg. Hernán Amatriain
Lic. Santiago Bianco
Lic. Sebastian Martins

Índice

1. INTRODUCCIÓN

"Lo más difícil en la construcción de un sistema software es decidir precisamente qué construir... " F. P. Brooks

Algunas historias relacionadas con los requisitos:

- Uno de los estudios mas conocidos es el de la General Accounting Office (GAO) de EEUU. Este estudio de 1977 reveló que el 47% del dinero empleado en proyectos software se destinó a sistemas que no llegaron a utilizarse. Otro 29% se empleó en proyectos que no llegaron a finalizar. Otro 19% se empleó en software que tuvo que ser profundamente modificado tras su entrega Finalmente tan sólo un 2% del dinero se empleó en proyectos software que sí cumplieron con sus requisitos pero se trataba de proyectos más bien pequeños o de poca envergadura.
- En 1981, Victor Basili encontró cerca de 88 errores en una ERS de 400 páginas para el proyecto A-7E Operational Flight Program. Esta ERS había sido escrita por un grupo de expertos en especificación de requisitos
- Recientemente la NASA ha sufrido dos accidentes espectaculares cuyo origen se atribuye a problemas durante la definición de los requisitos

Por otro lado, la evidencia empírica [Dav93] demuestra que:

- Los requisitos contienen demasiados errores,
- Muchos de estos errores no se detectan al principio,
- Muchos de estos errores podrían ser detectados al principio,
- No detectar estos errores incrementar los costes (tiempo, dinero) de forma exponencial y esto trae como consecuencia que:
    - El sistema resultante no satisfará a los usuarios,
    - Se producirían desacuerdos entre usuarios y desarrolladores,
    - Puede ser imposible demostrar si el software cumple o no los requisitos,
    - Se gastará tiempo y dinero en construir el sistema equivocado.

1.1. LA IMPORTANCIA DE LOS REQUISITOS

La evidencia empírica demuestra que cuanto más tarde se descubran, el coste de la reparación de los errores introducidos en la etapa de requisitos crece exponencialmente. Si asignamos coste al coste de reparar un error de requisitos descubierto en la etapa de requisitos, se obtendrían las siguientes cifras [Dav 93]:

Etapa	Coste de la reparación
Requisitos	1-2
Diseño	5
Codificación	10
Pruebas unitarias	20
Pruebas sistema	50
Explotación/Mtmto.	200

Ante esta situación ¿qué se puede hacer?. En primer lugar, se debe tomar conciencia del problema y estar a la defensiva. Quizá no se conozcan todas las soluciones, pero al menos se conocen los problemas. Quizá no sea posible elaborar los requisitos con absoluta perfección pero se deberá intentar minimizar el impacto de los errores en los requisitos y se podrá tratar de organizar mejor las tareas relacionadas con los requisitos.

¿Hay o habrá soluciones definitivas para el problema de los requisitos?.

Claramente, dado el estado actual del conocimiento, no se han encontrado soluciones universalmente validas e incluso hay serias dudas acerca de si dicha solución existe. Los requisitos se sitúan en la frontera socio técnica de los sistemas, y esta frontera es borrosa, voluble e inconsistente Según M Jackson los requisitos son "donde lo formal se encuentra con lo informal". Los requisitos están vivos: emergen, interactúan, cambian, desaparecen...

Incluso sería recomendable desconfiar de quien pretenda ofrecer una solución definitiva a estos problemas.

1.2. INGENIERÍA DE REQUISITOS

Para remediar en lo posible las situaciones planteadas, surge la Ingeniería de Requisitos (IR) [KS98]:

La IR trata de los principios métodos, técnicas y herramientas que permiten descubrir, documentar y mantener los requisitos para sistemas basados en computadora de forma sistemática y repetible

Hay demasiada confusión a la hora de definir lo que son realmente los requisitos, o qué se entiende por "requisitos" en el campo de la IR. Aún a riesgo de añadir más "ruido", se proporcionará una nueva definición (inspirada en [Jac95] y [Kov99])

En primer lugar se considerará que todo problema software, consiste en configurar una máquina M para que ejerza unos efectos R en un dominio D.

- Los efectos R sería, propiamente hablando, los requisitos: representan necesidades, metas y objetivos.
- El dominio D es el contexto: Los requisitos R, sin contexto, no tienen sentido. Cambiando el contexto D, un requisito de R pierde su sentido.
- La máquina M es la que realizará los requisitos R, gracias a su conexión con D. En la fase de requisitos tan sólo necesitamos describir las conexiones de M con D, es decir, el comportamiento externo de M (M-ex), sin detalles internos (M-int).

Pues bien, por extensión, en IR se denominan "requisitos" a los conjuntos M-ex, R y D, *aunque tan sólo R sean propiamente requisitos.*

Por ejemplo: Supóngase que hay que desarrollar el software para un sistema de control de una caldera de vapor. En este contexto, la afirmación:

- el agua entra en ebullición a 100 Grados Centígrados y a 1 atm. de presión

No es un requisito, ya que no es una meta ni un objetivo. La anterior afirmación (AF1) es parte de la descripción del Dominio (D), y es verdadera independientemente de la existencia o no del sistema. En cambio, la frase

- El sistema evitará que el agua entra en ebullición.

Es un requisito (R). Expresa un deseo u objetivo. Algo que el sistema deberá realizar. Por otro lado, las frases:

- El sistema leerá la temperatura del agua por medio del sensor.

- El sistema podrá subir la temperatura del agua por medio del sensor.

Describen la conexión del software (nuestra máquina M) con el entorno, es decir, describen el comportamiento externo del software (M-ex). No son metas ni objetivos, pero son necesarios para conseguir las metas y los objetivos.

Detallando un poco más lo expuesto anterioriormente, es necesario que un documento de requisitos contenga, en primer lugar:

- Información acerca del problema.
- Propiedades y comportamiento del sistema.
- Restricciones de diseño y fabricación del producto.

Pero además podría contener:

- Descripciones acerca de cómo el futuro sistema ayudará a sus usuarios a realizar mejor sus tareas.
- Restricciones acerca de la tecnología que será utilizada en la construcción del sistema (protocolos, SSOO, COTS, etc).
- Restricciones acerca de las propiedades emergentes del sistema (requisitos no funcionales)

1.3. ¿CÓMO ESCRIBIR REQUISITOS?

Los requisitos constituyen la base de la comunicación entre todas las partes interesadas en el desarrollo del sistema: usuarios, clientes, desarrolladores y el equipo encargado de realizar las pruebas. Todas estas personas forman un conjunto heterogéneo, lo cual provoca, desafortunadamente, que la "mejor forma" de escribir requisitos no exista. Lo que para unos puede ser un lenguaje incomprensible.

Debido a esto, en la práctica, lo más utilizado es el lenguaje natural, a pesar de su inherente ambigüedad. Se acostumbra a especificar cada requisito como una frase corta ("el sistema hará X…", "se facilitará X…", etc). También se utiliza el lenguaje natural completando con diagramas y/o notaciones formales. La notación utilizada depende de quién leerá o quien escribirá los requisitos.

Ejemplos de requisitos podría ser:

1. El sistema mantendrá la temperatura de la caldera entre 70 y 80 grados.

2. El sistema mantendrá un registro de todos los materiales de la biblioteca, incluyendo libros, periódicos, revistas, videos y Cdroms.
3. El sistema permitirá a los usuarios realizar una búsqueda por título, autor o ISBN.
4. La interfaz de usuario se implementará sobre un navegador web.
5. El sistema deberá soportar al menos 20 transacciones por segundo.
6. El sistema permitirá que los nuevos usuarios se familiaricen con su uso en menos de 15 minutos.

Aquí puede verse que los requisitos son de muy distintos tipos:

- Requisitos que definen efectos sobre el entorno (p.ej El 1)
- Requisitos muy generales (p.ej ,el 2)
- Requisitos funcionales (3)
- Requisitos de implementación (4)
- Requisitos de rendimiento (5)
- Requisitos de usabilidad (6)

Debido a que hay tantos tipos distintos de requisitos, no es posible establecer una forma estándar de escribirlos. Tampoco es posible decir cual es "la mejor forma" de especificarlos. Todo depende mucho de quien los escribe, quién los va a leer, el dominio de la aplicación, etc.

1.3.1. REQUISITOS EN NEGATIVO

Tan importante como decir lo que el sistema debe hacer, lo es el decir lo que el sistema NO debe hacer. Estos requisitos "en negativo" limitan el ámbito del sistema. Las razones son varias:

- En primer lugar, dicen donde NO se deben emplear recursos.
- Por otro lado, especificar lo que el sistema no hará es fundamental para sistemas críticos (es decir, sistemas en los que un fallo puede provocar un accidente). Se trata de especificar cómo el sistema evitará la aparición de situaciones potencialmente peligrosas.

1.3.2. REQUISITOS FUNCIONALES Y NO FUNCIONALES

Los requisitos funcionales describen los servicios (funciones) que se esperan del sistema. Por ejemplo: El sistema aceptará pagos con VISA.

Los requisitos no funcionales son restricciones sobre los requisitos funcionales. Por ejemplo: El sistema aceptará pagos con VISA de forma segura y con un tiempo de respuesta menor a 5 segundos.

Pero esta distinción, muchas veces, resulta arbitraria. Por ejemplo: El sistema aceptará pagos con VISA a través del protocolo SET.

En este caso, lo que aparentemente es un requisito funcional, está determinado, en el fondo, por una serie de características no funcionales (características que hacen que el protocolo SET sea el más adecuado para los objetivos perseguidos).

2. 2. EL PROCESO DE REQUISITOS

El proceso de Ingeniería de Requisitos es un conjunto estructurado de actividades que sirven para derivar, validar y mantener los requisitos de un sistema (hardware, software o hardware+software). Las tareas que conlleva este proceso, a grandes rasgos, se representan en la figura 2.1.

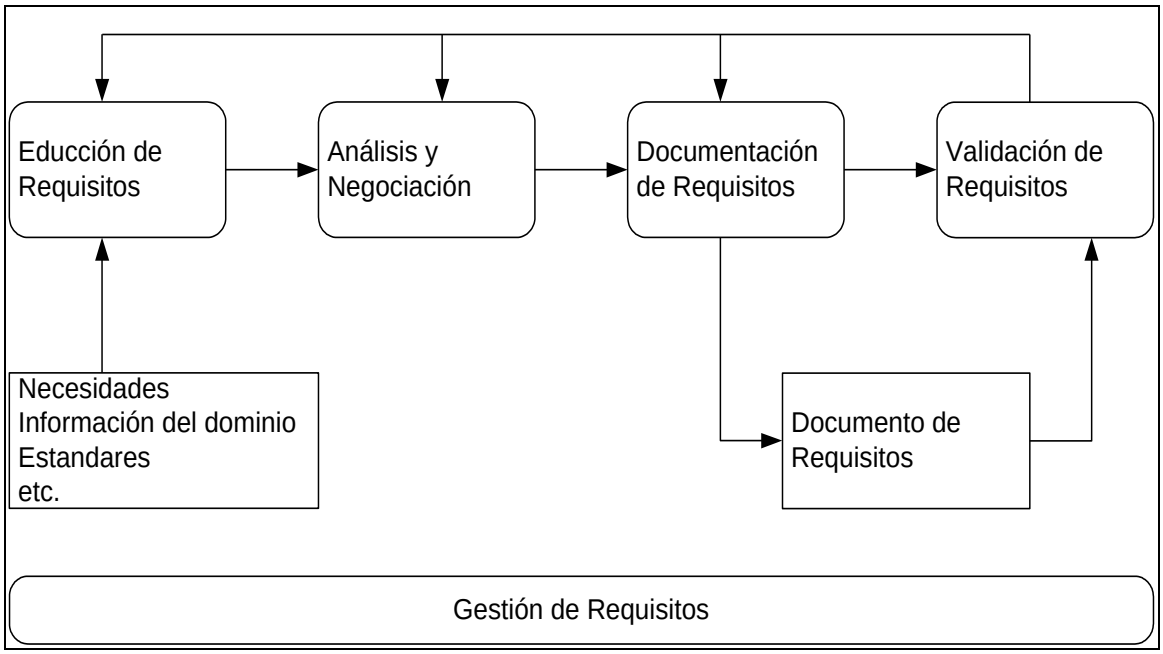

Figura 2.1. Proceso de Ingeniería de Requisitos

En la figura, los cuadros redondeados son tareas. Los cuadrados son productos (imputs o ouptus). En el resto de esta presentación se explicará cada componente de este proceso. Debe destacarse que la separación que aquí se ofrece es más conceptual que real. Las distintas tareas que se ejecutan durante el proceso de requisitos suceden en paralelo y se solapan unas con otras. Por ejemplo, durante un proceso de educción de requisitos empleando prototipado, es inevitable realizar una pequeña validación de los requisitos que se van obteniendo, o incluso una pequeña negociación, si, por ejemplo, estamos tratando con varios usuarios a la vez.

3. EDUCCION DE REQUISITOS

La educción de requisitos se refiere a la captura y descubrimiento de los requisitos. Es una actividad más "humana" que técnica, en la que se identifica a los interesados y se establecen las primeras relaciones entre ellos y el equipo de desarrollo.

3.1. PRINCIPALES FUENTES DE REQUISITOS

Los requisitos pueden proceder de:

- Metas: Factores críticos de éxito.
- Conocimiento del dominio de la aplicación.
- Los interesados. Los afectados por el sistema.
- El entorno físico que rodea al sistema.
- El entorno organizacional. Los procesos de negocio.

3.2. PROBLEMAS DE EDUCCIÓN

Entre los principales problemas que pueden entorpecer la tarea de educción de requisitos se cuentan los siguientes:

- Los usuarios no pueden/saben describir muchas veces sus tareas.
- Mucha información importante no llega a verbalizarse.
- A veces hay que "inventar" los requisitos (sistemas orientados a miles de usuarios).
- La educción se afronta como un proceso pasivo, cuando debería ser un proceso cooperativo.

3.3. TÉCNICAS DE EDUCCIÓN

- Entrevistas: Es el método "tradicional"
- Observaciones y análisis de tareas
- Escenarios: los requisitos se sitúan en el contexto de uso del sistema
- Prototipado: útil cuando la incertidumbre es total acerca del futuro sistema. Hay dos tipos principales:
    - Evolutivo
    - De usar y tirar (prototipazo en papel, mago de Oz, etc)

4. ANÁLISIS DE REQUISITOS

El análisis de Requisitos consiste en detectar y resolver conflictos entre requisitos. Durante la realización de esta tarea, se precisan los límites del sistema y la interacción con su entorno, se trasladan los requisitos de usuario a requisitos del software (implementables) y, ante todo, se realizan tres subtareas fundamentales:

- Clasificación de los requisitos
- Modelización de requisitos
- Negociación

4.1. CLASIFICACIÓN DE LOS REQUISITOS

En el análisis de requisitos, éstos se pueden clasificar de distintas formas y atendiendo a distintos criterios:

- En funcionales vs. No funcionales (Capacidades vs. Restricciones).
- Por prioridades.
- Por coste de implementación.
- Por niveles (alto nivel, bajo nivel).
- Según su volatilidad / estabilidad.
- Si son requisitos sobre el proceso o sobre el producto.

4.2. MODELIZACIÓN DE REQUISITOS

Ciertos aspectos de los requisitos se expresan mediante modelos de datos, de control, de estados, de interacción, de objetos, etc. La meta es entender mejor el problema, más que iniciar el diseño de la solución (idealmente). El tipo de modelo elegido depende de:

- La naturaleza del problema.
- La experiencia del modelizador.
- La disponibilidad de herramientas.
- Por decreto. El cliente impone una notación.

Tradicionalmente se entendía que "el análisis" se reducía a modelizar (DFSs, modelos de objetos, etc), pero el análisis de requisitos NO es exclusivamente un proceso de modelización. Por otro lado, no existe "la mejor" forma de

modelizar requisitos. En realidad, no hay evidencia empírica que demuestre, en general, la superioridad de unas notaciones de modelización frente a otras.

4.3. NEGOCIACIÓN DE REQUISITOS

En todo proceso de IR intervienen distintos individuos y, a veces, enfrentados intereses. Estos conflictos entre requisitos se descubren durante el análisis. Todo conflicto descubierto debería disparar un proceso de (re)negociación, es decir, los conflictos NUNCA se deben resolver "por decreto".

Es durante el análisis cuando muchos de los conflictos entre requisitos son descubiertos. EL CONTROL NO ES RECHAZABLE y no debe resolverse por decreto, sino mediante un proceso de negociación. Desde este punto de vista, los conflictos son positivos, pues SON FUENTE DE NUEVOS REQUISITOS. Los acuerdos alcanzados deber ser convenientemente anotados, favoreciéndose así la trazabilidad de los requisitos a sus orígenes ("el requisito 987 surge como resultado de la reunión entre x, y, z el día tal de tal...")

5. DOCUMENTO DE REQUISITOS

El llamado "Documento de Requisitos" es el modo habitual de guardar y comunicar los requisitos. Debe tenerse en cuenta que, al decir "documento", nos referimos a cualquier medio electrónico de almacenamiento y distribución de información como, por ejemplo:

- Procesador de textos.
- Base de Datos.
- Herramienta de Gestión de Requerimientos.

En general, es buena práctica utilizar, al menos, dos documentos, a distinto nivel de detalle:

1. DRU = Documento de Requisitos de Usuario (en inglés, URD)
2. ERS = Especificación de Requisitos Software (en ingles, SRS)

La pregunta que surge es ¿en qué se diferencian los requisitos de usuario de los requisitos del software?. A grandes rasgos se puede afirmar que:

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

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

5.1. ESTÁNDARES

Los dos documentos estándares más conocidos de especificación de requisitos son:

- IEEE Std. 830/1983.
- PSS-05 de la Agencia Espacial Europea (ESA) Las herramientas de gestión de requisitos permiten generar documentación en los anteriores formatos, a partir de una base de datos de requisitos.

5.2. CARACTERÍSTICAS DESEABLES DE UNA ERS

Una ERS de calidad debería presentar, dentro de lo posible, las siguientes características:

- No ambigua: la ERS es no ambigua si todo requisito posee una sola interpretación
- Completa: Una ERS es completa si todo lo que se supone que el software debe hacer está incluido en la ERS. Por completitud, deberían describirse todas las posibles respuestas a todas las posibles entradas y en todas las situaciones posibles. Además, la ERS no contendrá secciones de tipo "por determinar "
- Correcta: Todo requisito de la ERS contribuye a satisfacer una necesidad real
- Comprensible: Todo tipo de lectores (clientes, usuarios, desarrolladores, equipo de pruebas, gestores, etc) entienden la ERS
- Verificable: Si para cada requisito expresado en la ERS existe un procedimiento de prueba finito y no costoso para demostrar que el futuro sistema lo satisface
- Internamente Consistente: No existen subconjuntos de requisitos contradictorios
- Externamente Consistente: Ninguno de los requisitos está en contradicción con lo expresado en documentos de nivel superior. Por ejemplo, en un sistema (hardware/software), los requisitos del software no pueden contradecir los requisitos del sistema
- Realizable: Si, dados los actuales recursos, la ERS es implementable
- Concisa: La ERS debe ser lo más breve posible, sin que esto afecte al resto de atributos de calidad
- Independiente del diseño: Existen más de un diseño e implementación que realizan la ERS. Para ello la ERS debería limitarse a describir el comportamiento externo del sistema
- Trazable: Cada requisito se puede referenciar de forma unívoca Es fundamental para precisar qué requisitos son implementados por qué componente del diseño, lo cual es imprescindible a la hora de realizar las pruebas de dicho componente
- Modificable: Los cambios son fáciles de introducir
- Electrónicamente almacenada: Se encuentra en un archivo de texto, en una base de datos o, mejor aun, ha sido creada con una herramienta de gestión de requisitos (RequisitePro, Doors, etc)

- Ejecutable/Interpretable/Prototipable/Animable: Si existe una herramienta software que, recibiendo como entrada la ERS, realice un modelo ejecutable de la misma. Aplicable tan sólo a ciertas notaciones como las notaciones formales o los diagramas de transición de estados
- Anotada por importancia relativa: Si los requisitos se clasifican según su importancia. Como mínimo un requisito puede ser "Obligatorio", "Deseable" u "Opcional". Esto sirve para no asignar demasiados recursos a la implementación de requisitos no esenciales
- Anotada por estabilidad relativa: Los requisitos son, en general, inestables y volátiles. A cada requisito se le asigna una probabilidad de cambio (p. ej. "Alta", "Media" o "Baja"). Esto ayudará a los diseñadores a diferenciar los componentes más flexibles de los más estables
- Anotada por versión: Si un lector de la ERS puede determinar qué requisitos serán satisfechos por qué versión del producto
- No redundante: Cada requisito se expresa en un solo lugar de la ERS. La redundancia de todas formas no es del todo mala si aumenta la legibilidad
- Al nivel adecuado de abstracción: Ni demasiado detallada ni demasiado vaga
- Precisa: Una ERS es precisa si hace uso de valores numéricos para precisar las características del sistema. La precisión es aplicable, ante todo, a los requisitos no funcionales. Por ejemplo, no es útil decir "El tiempo de respuesta será más bien rápido", sino "El tiempo de respuesta será menor que dos segundos". OJO: en la práctica diaria, este atributo es dificilísimo de conseguir pues es fuertemente dependiente de la tecnología disponible, lo cual no siempre se conoce al principio de un proyecto
- Reutilizable: Si ciertas secciones de la ERS se pueden reutilizar
- Trazada: Si está claro el origen de cada requisito (quién o qué lo pide)
- Organizada: Si el lector puede fácilmente encontrar la información buscada
- Con referencias cruzadas: Si se utilizan referencias entre requisitos relacionados (trazabilidad intra-ERS) o entre secciones relacionadas

De todas formas, aumentar una de estas características es posible que disminuya otra (por ejemplo disminuir la ambigüedad puede conducir a un aumento de la ininteligibilidad). La calidad debe perseguirse como algo ideal, a pesar de que una ERS perfecta es imposible. La calidad de la ERS es muy difícil de cuantificar y, en general, una ERS de calidad NO garantiza la ausencia de problemas aunque una ERS pésima garantiza su presencia.

6. VALIDACIÓN DE REQUISITOS

El objetivo de la validación de los requisitos es descubrir problemas en el Documento de Requisitos antes de comprometer recursos a su implementación.

El documento deber revisarse para:

- Descubrir omisiones.
- Conflictos.
- Ambigüedades.
- Comprobar la calidad del documento y su grado de adhesión a estándares.

6.1. REVISIONES (REVIEWS)

Las revisiones del documento de requisitos constituyen la fórmula más empleada en validación. En estas revisiones, un grupo de personas (incluyendo usuarios) se ocupan de revisar el documento de requisitos. Tienen tres fases:

- Búsqueda de problemas
- Reunión
- Establecimiento de acuerdos

Como guía para identificar problemas habituales, se pueden utilizar listas de comprobación ("checklists"). Hay "checklists" adaptadas a distintos tipos de sistemas. Otros métodos de validación son:

- Prototipado: Permite descubrir con rapidez si el usuario se encuentra satisfecho, o no, con los requisitos
- Validación de modelos: Cuando los requisitos se expresan por medio de modelos (de objetos, DFDs, etc)
- Validación de la "testabilidad". El equipo de pruebas debe revisar los requisitos.

El 33% de los errores de requisitos en la especificación del sistema A-7E fueron detectados mediante revisión manual. El resto se descubrieron en posteriores fases, con el consiguiente incremento en el coste.

Curiosamente, las revisiones parecen funcionar también con el código ejecutable: se descubren más errores inspeccionando el código fuente que

ejecutando el programa. Quizá radique aquí el éxito de los desarrollo Open Source o de fuente abierto.

Cada organización, según su experiencia y según el dominio de las aplicaciones que desarrolle, debería desarrollar su lista de comprobación o "checklist" particular. Un ejemplo de cuestiones que deberían figurar en una lista de comprobación podría ser esta:

- ¿Están todos los requisitos convenientemente numerados?
- ¿El mismo servicio es solicitado en distintos requisitos? Existen contradicciones entre ellos?
- ¿Los requisitos son fácilmente comprensibles? Por todo tipo de lectores?

En general, una lista de comprobaciones debería girar alrededor de los atributos de calidad (anteriormente expuestos) que debería poseer una ERS. Para cada atributo de calidad, se pueden plantear una serie de cuestiones que sirven para confeccionar la lista de comprobación.

7. GESTIÓN DE REQUISITOS - REQUIREMENTS MANAGEMENT (RM)

La Gestión de Requisitos consiste, básicamente, en gestionar los *cambios* a los requisitos. Asegura la consistencia entre los requisitos del sistema construido (o en construcción). Esta tarea consume grandes cantidades de tiempo y esfuerzo y abarca todo el ciclo de vida del producto.

Es necesario realizar una adecuada gestión de requisitos por una serie de razones, como:

- Los requisitos son volátiles.
- El entorno físico del sistema cambian.
- Trasladar un sistema de un entorno a otro requiere modificaciones
- El entorno organizacional cambian.
- Las políticas cambian.
- Cambios en las reglas y en los procesos del negocio provocan cambios en el sistema.
- La propia existencia del sistema va a generar nuevos requisitos por parte de los usuarios.

La Gestión de Requisitos implica

- Definir procedimientos de cambios: definen los pasos y los análisis que se realizarán antes de aceptar los cambios propuestos
- Cambiar los atributos de los requisitos afectados
- Mantener la trazabilidad: hacia atrás, hacia delante y entre requisitos
- Control de versiones del documento de requisitos

Para realizar adecuadamente estas tareas, se pueden emplear herramientas de Gestión de Requisitos, que facilitan las tareas relacionadas con la escritura, trazabilidad y gestión de cambios. Estas herramientas organizan los requisitos en bases de datos y permiten añadir atributos a los requisitos. Conocidas herramientas comerciales de Gestión de requisitos sería DOORS, RequistePro, icConcept, etc.

8. LA IR EN LA PRÁCTICA HOY

Hoy el software es parte casi imprescindible de todo tipo de dispositivos. El software se utiliza para proporcionar valor añadido a productos tradicionales (teléfonos, refrigeradores, coches) y para diferenciarse de los productos de la competencia. Basta con decir que, hoy en día, el componente más complejo de un coche es el software.

Industrias tradicionalmente alejadas del software hoy tienen problemas de requisitos. Introducir componentes software es distinto a introducir nuevos componentes físicos. La complejidad que añade el software es de varios ordenes de magnitud respecto a la complejidad que añadiría un nuevo componente físico.

Introducir el software en industrias tradicionales está produciendo choques culturales. Estas industrias poseen tradiciones, normas y procedimientos que no son aplicables al desarrollo de software. El mismo concepto de prototipo posee significados muy distintos en Ingeniería Aeronáutica, por ejemplo, y en Ingeniería del Software.

Actualmente, los requisitos no funcionales (seguridad, fiabilidad, tiempo de respuesta, disponibilidad, etc), ocupan más espacio en el documento de requisitos que los requisitos puramente funcionales.

Los requisitos ya no son un problema exclusivo de la industria del software y, desde luego, no son un problema exclusivo de los Sistemas de Información tradicionales. Pero es el software, precisamente lo que provoca los problemas de requisitos debido al gran potencial que posee para aumentar la complejidad de un sistema

9. REFERENCIAS

[Bro75] F. P. Jr. Brooks. The Mythical Man-Month. Addison Wesley, 1975
[Dav93] A. Davis. Software Requirements: Objects, Functions and States. Prentice-Hall, 1993
[Jac95] M. Jackson. Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices. Addison Wesley, 1999
[Kov99] B. L. Kovitz. Practical Software Requirements. A Manual of Content and Style. Manning,1999
[KS,98] G. Kotonya and I Sommerville. Requirements Engineering Processes and Techniques. Wiley, 1998

10.GUIA DE TRABAJOS PRACTICOS

1. ¿Qué es la Ingenieria de Requisitos?
2. ¿Para que se escriben los requisitos en negativo? De ejemplos, al menos 3.
3. Explique que son los requisitos funcionales y no funciones. De ejemplos, al menos 5 de cada uno.
4. Mencione y explique 7 caracteristicas deseable de una ERS, justifique porque identifico esas 7 y no las otras.