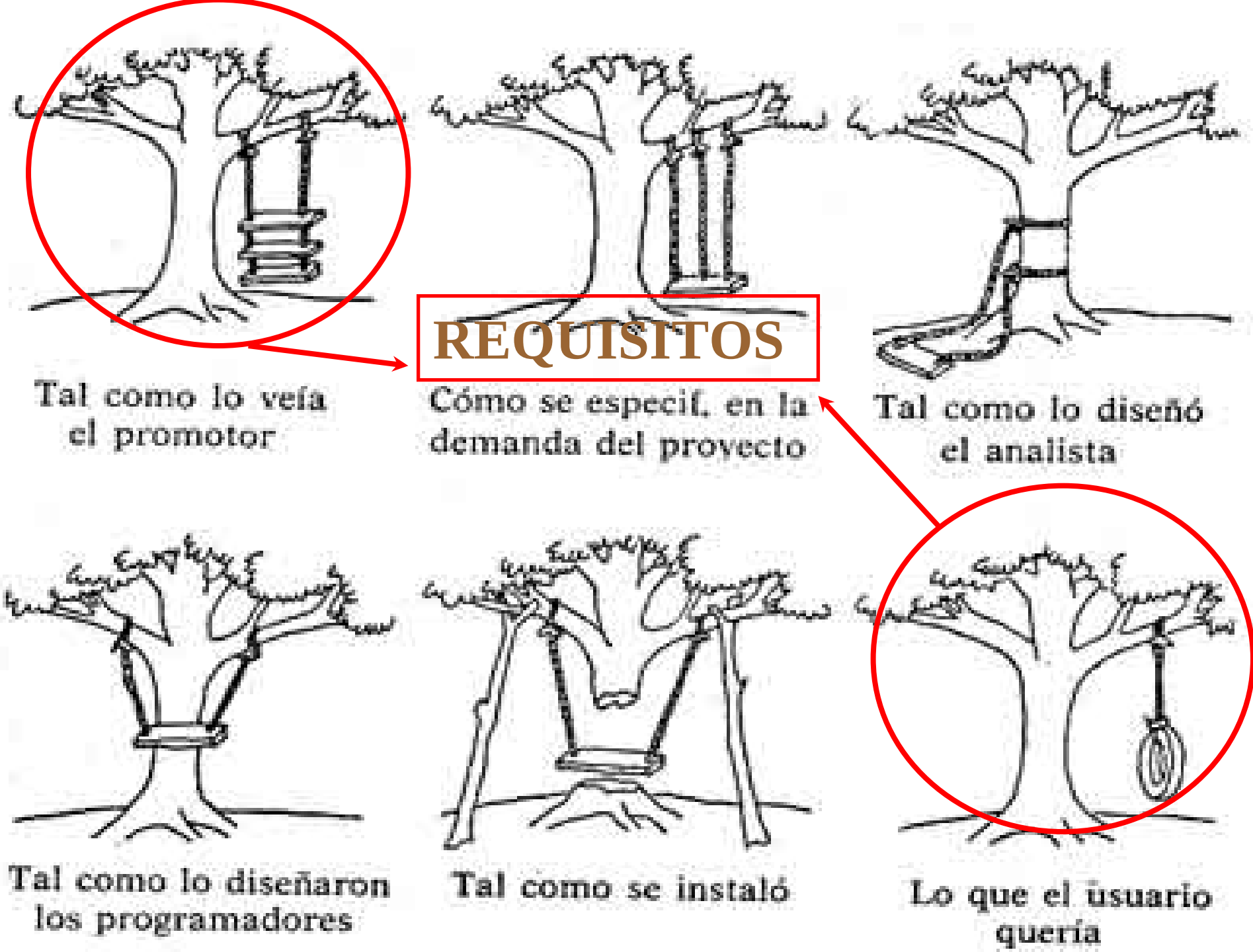
Ingeniería de Requisitos

Curso 2008-2009

Gonzalo Méndez
Dpto. de Ingeniería de Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid


REQUISITOS
Tal como lo veía el promotor
Cómo se especif. en la demanda del proyecto
Tal como lo diseñó el analista
Tal como lo diseñaron los programadores
Tal como se instaló
Lo que el usuario quería

Problemas

- Los usuarios no saben lo que quieren
- Un sistema tiene muchos usuarios y ninguno tiene una visión de conjunto
- No saben cómo hacer más eficiente la operación en su conjunto
- No saben qué partes de su trabajo pueden transformarse en software
- No saben detallar lo que saben de forma precisa

Solución tradicional: analistas

- Labores
  - obtener una lista de requisitos de cada usuario
  - adquirir una visión de conjunto
  - componer una especificación completa, correcta y consistente
- Desventajas
  - listas de requisitos son difíciles de comprender y de hacer bien
  - difíciles de transformar en especificaciones de diseño e implementación

¿Qué buscamos?

- Requisitos
  - Los detalles sobre lo que tendremos que hacer
- Viabilidad
  - Saber si se va a poder hacer o no
- Alcance
  - Cuánto de lo que se podría hacer nos va a dar tiempo a hacer con el tiempo y la gente que tenemos

Ingeniería de Requisitos

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

Ingeniería de Requisitos

Lo más difícil en la construcción de un sistema software es decidir precisamente qué construir

No existe tarea con mayor capacidad de lesionar al sistema, cuando se hace mal

Ninguna otra tarea es tan difícil de rectificar a posteriori

F. P. Brooks, 1987

Hechos

- Boehm, 1975: 45% de los errores tienen su origen en los requisitos y en el diseño preliminar
- DeMarco, 1984: 56% de los errores que tienen lugar en un proyecto software se deben a una mala Especificación de Requisitos
- Hasta hoy no se ha mejorado mucho

Otras historias

- Uno de los estudios más conocidos es el de la General Accounting Office (GAO) de EEUU.
- Este estudio de 1979 reveló que 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

Más historias

- 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.

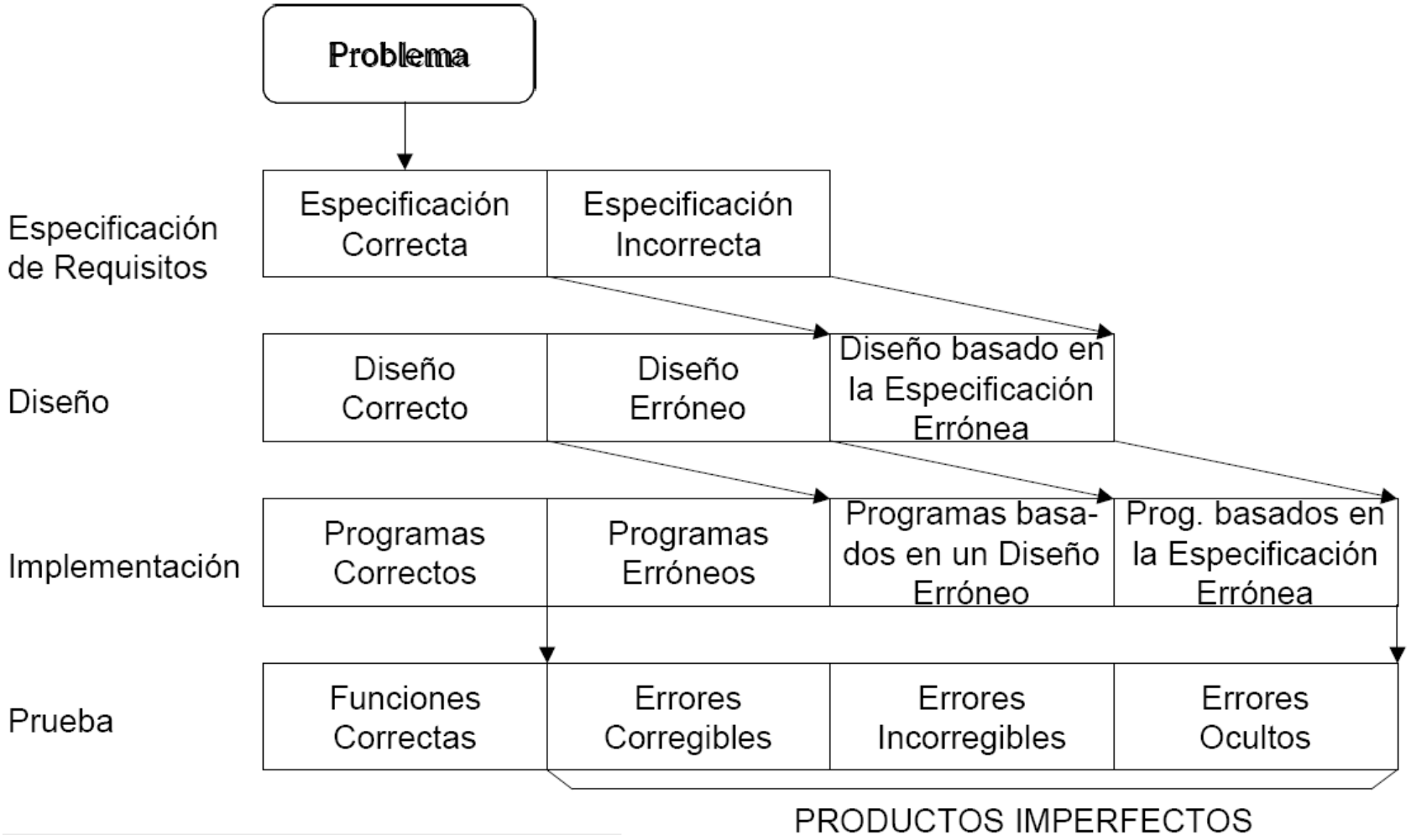
Evidencias empíricas

- 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 incrementará los costes (tiempo, dinero) de forma exponencial

Coste de los errores

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

Acumulación de los errores



Qué son los requisitos

- Todo problema sw. consiste en configurar una máquina M para que ejerza unos efectos R en un dominio D
  - Los efectos R son los requisitos: Necesidades, metas, 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 (comportamiento externo de M, sin detalles internos).
- Por extensión, en IR se denominan “requisitos” a los conjuntos M, R y D, aunque tan sólo R sean propiamente requisitos.

Ejemplo

- Supóngase que hay que desarrollar el software para un sistema de control de una caldera de vapor
- Posibles requisitos
  - El agua entra en ebullición a 100 Grados Centígrados y a 1 atm de presión
  - El sistema evitará que el agua entre en ebullición
  - El sistema leerá la temperatura del agua por medio del sensor
  - El sistema podrá subir la temperatura del agua por medio del regulador

Ejemplo

- 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
  - es parte de la descripción del dominio (D), y es verdadera independientemente de la existencia o no del sistema
- El sistema evitará que el agua entre en ebullición
  - Es un requisito (R). Expresa un deseo u objetivo. Algo que el sistema deberá realizar

Ejemplo

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

Describen la conexión del software (máquina M) con el entorno, es decir, describen el comportamiento externo del software.

No son metas ni objetivos, pero son necesarios para conseguir las metas y los objetivos

Ejemplo

- Aunque propiamente hablando tan sólo el segundo es un requisito, a la Ingeniería de Requisitos le interesan todas las afirmaciones anteriores, ya que todas aportan información relevante para construir el sistema y para que el sistema funcione de la manera deseada

Contenidos relevantes

- Un documento de requisitos contiene, ante todo:
  - Información acerca del problema
  - Propiedades y comportamiento del sistema
  - Restricciones de diseño y fabricación del producto

Otros contenidos relevantes

- Y además:
  - 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)
- Los requisitos NO son análisis top-down, ni son la arquitectura del sistema, ni son el “qué” frente al “cómo”

Cómo escribir requisitos

- La “mejor forma” de escribir requisitos no existe
- Lo más utilizado es el lenguaje natural. Cada requisito expresado en una frases corta (“el sistema hará X ...”, “se facilitará Y ...”, etc)
- O lenguaje natural complementado con diagramas y/o notaciones formales
- La notación utilizada depende de quien lee o quien escribe los requisitos

Ejemplos de requisitos

1. Requisitos que definen efectos sobre el entorno
   - El sistema mantendrá la temperatura de la caldera entre 70º y 80º
2. Requisitos muy generales
   - El sistema mantendrá un registro de todos los materiales de la biblioteca, incluyendo libros, periódicos, revistas, videos y CDRoms
3. Requisitos funcionales
   - El sistema permitirá a los usuarios realizar una búsqueda por título, autor o ISBN
4. Requisitos de implementación
   - La interfaz de usuario se implementará sobre un navegador Web
5. Requisitos de rendimiento
   - El sistema deberá soportar al menos 20 transacciones por segundo
6. Requisitos de usabilidad
   - El sistema permitirá que los nuevos usuario se familiaricen con su uso en menos de 15 minutos.

Requisitos en negativo

- Es importante decir lo que el sistema NO debe hacer
- Estos requisitos “en negativo” limitan el ámbito del sistema. Dicen donde NO se deben emplear recursos
- Fundamental para sistemas críticos
- Se debe mantener la distinción liveness/safety
  - Liveness: dicen lo que el sistema debe hacer
  - Safety: dicen lo que el sistema no debe hacer

Funcionales y no funcionales

- Los requisitos funcionales describen los servicios (funciones) que se esperan del sistema
  - El sistema aceptará pagos con VISA
- Los requisitos no funcionales son restricciones sobre los requisitos funcionales
  - El sistema aceptará pagos con VISA de forma segura y con un tiempo de respuesta menor de 5 segundos
- Pero esta distinción, muchas veces, resulta arbitraria
  - El sistema aceptará pagos con VISA a través del protocolo SET

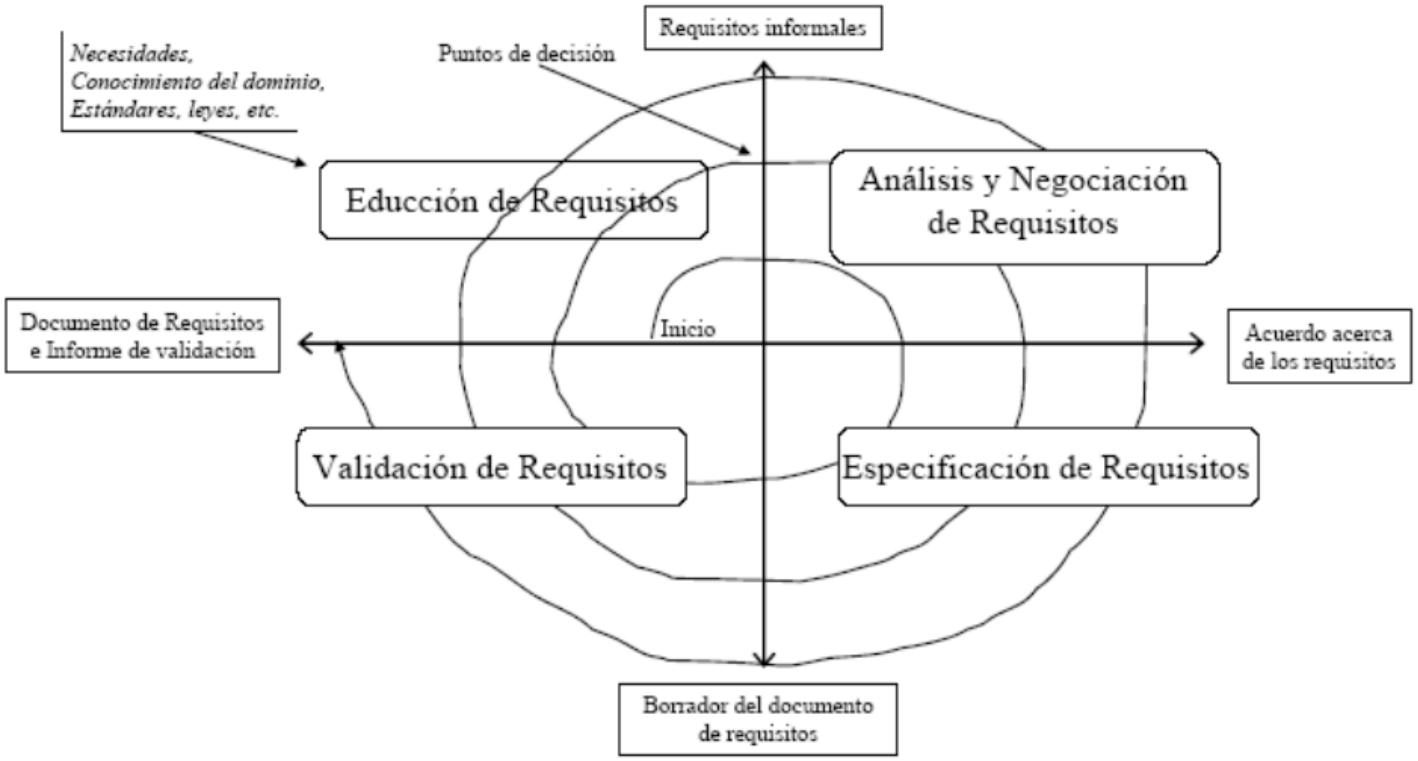
Relación requisitos-arquitectura

- La elección de una determinada arquitectura software debe tener en cuenta los requisitos funcionales pero, sobre todo, los requisitos no funcionales (atributos de calidad del software)
- No hay una regla definitiva para establecer, dados los requisitos, el tipo de arquitectura
- Tan sólo hay una serie de heurísticas para, dados unos requisitos, elegir la arquitectura

Educción de requisitos

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

El proceso


Requisitos informales
Necesidades,
Conocimiento del dominio,
Estándares, leyes, etc.
Puntos de decisión
Educción de Requisitos
Análisis y Negociación de Requisitos
Documento de Requisitos e Informe de validación
Inicio
Acuerdo acerca de los requisitos
Validación de Requisitos
Especificación de Requisitos
Borrador del documento de requisitos

Problemas en el proceso

- El proceso de requisitos es **excesivamente largo y costoso**.
- Los implicados en el proceso se quejan de **falta de tiempo y otros recursos**.
- Se reciben quejas acerca de la **inteligibilidad del documento** de requisitos.
- Los implementadores se quejan del **trabajo perdido por culpa de errores** en los requisitos.
- Los usuarios **no utilizan muchas de las capacidades** del sistema.
- En cuanto el producto se entrega, se recibe una **inmensa cantidad de solicitudes de cambios**.
- Lleva demasiado **tiempo alcanzar un acuerdo cuando se proponen cambios** a los requisitos.

Fuentes de Requisitos

- Metas: Factores críticos de éxito
- Conocimiento del dominio de la aplicación
  - Por ejemplo, un usuario quiere consultar por pantalla todas las pólizas que venzan durante el mes. Para que ello sea posible, el software deberá obligar, cada vez que se crea una póliza, a que se introduzca su fecha de vencimiento. Esto puede resultar obvio para un informático, pero no lo es tanto para un usuario inexperto
- Los interesados. Los afectados por el sistema.
- El entorno físico que rodea al sistema
- El entorno organizacional. Los procesos de negocio

Problemas de la educción

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

Técnicas de educción

- **Preliminares**:
  - Utilizar preguntas libres de contexto.
- **Brainstorming**:
  - Seleccionar un grupo variado de participantes.
  - Eliminar críticas, juicios y evaluaciones mientras los participantes sugieren ideas.
  - Producir **muchas** ideas.
  - Recogerlas todas por escrito.
  - Otro día, en otra sesión, se evalúan las ideas (se puntúan).
- **Prototipado**:
  - Útil cuando la incertidumbre es grande acerca del futuro sistema

Técnicas de educción

- **Entrevistas**:
  - Es el método “tradicional”, pero debe usarse en complemento con otras técnicas, y no debe ser el primer paso de la educción. Es fundamental:
    - Entrevistar a la(s) persona(s) adecuadas.
    - Preparar las preguntas con antelación.
    - Utilizar diagramas, modelos, etc.
- **Observación y análisis de tareas**:
  - Un observador estudia a los futuros usuarios en su entorno de trabajo. A veces se utiliza el video. Anota todo aquello que es susceptible de mejora. Posteriormente, genera una serie de requisitos tentativos.
- **Casos de uso / Escenarios**:
  - Requisitos en contexto de uso.

Entrevistas

- Una situación en que los participantes...
  - no saben qué decir
  - se preocupan de que se les entienda mal
  - piensan a dónde va a llevar
  - tienen expectativas diferentes
  - quieren que se acabe cuanto antes
  - quieren que sea un éxito

Qué se pretende

- Definir objetos observables
- Evaluar el flujo y contenido de la información
- Definir y elaborar funciones del software
- Entender el comportamiento del sistema
- Establecer características de la interfaz
- Descubrir restricciones ocultas

Preguntas sobre el contexto

- Quién solicita este trabajo
- Quién usará el producto
- Cuál es el beneficio económico de una solución satisfactoria
- Hay más fuentes para la solución que se busca

Preguntas sobre el problema

- Describir buenos resultados generados por una solución buena
- Cuál es el problema al que nos enfrentamos
- En qué entorno (describir/mostrar) se va a utilizar
- Restricciones específicas de rendimiento

Preguntas sobre la reunión en sí

- Es usted la persona adecuada para responder a estas preguntas
- Son oficiales sus respuestas
- Le parecen relevantes mis preguntas
- Hago demasiadas preguntas
- Hay alguien más que pueda aportar información
- Hay algo más que debería preguntar

Limitaciones

- Las reuniones generales dan resultados muy pobres.
- Se deben emplear sólo como primer paso, para luego ser sustituidos por reuniones que combinen resolución de problemas, negociación y especificación.

Análisis de requisitos

- Consiste en detectar y resolver conflictos entre requisitos
- Se precisan los límites del sistema y la interacción con su entorno
- Se trasladan los requisitos de usuario a requisitos del software (implementables)
- Se realizan tres tareas fundamentales:
  - Clasificación
  - Modelización
  - Negociación

Clasificación de los requisitos

- En el análisis de requisitos, éstos se clasifican
  - 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

Modelización conceptual

- 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 modelador
  - La disponibilidad de herramientas
  - Por decreto. El cliente impone una notación

Negociación de requisitos

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

Documentos de requisitos

- Es el modo habitual de guardar y comunicar requisitos.
- Es buena práctica utilizar, al menos, dos documentos, a distinto nivel de detalle
  - DRU = Documento de Requisitos de Usuario (URD)
  - ERS = Especificación de Requisitos Software (SRS)
- OJO: Con “Documento” nos referimos a cualquier medio electrónico de almacenamiento y distribución:
  - Procesador de textos
  - Base de Datos
  - Herramienta de Gestión de Requisitos

DRU vs. ERS

- El DRU se escribe desde el punto de vista del usuario o cliente o 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 espera 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?

Estándares

- IEEE Std. 830 (versión actual de 1998)
- 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

Esquema de IEEE 830

- Introducción
  - Propósito
  - Alcance
  - Definiciones
  - Referencias
  - Visión General
- Descripción General
  - Perspectiva del producto
  - Funciones del producto
  - Características del usuario
  - Restricciones
  - Suposiciones
- Requisitos específicos
- Apéndices
- Índice

Características de una ERS

- Una ERS *de calidad* debería poseer 24 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...”

Características de una ERS

- 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.

Características de una ERS

- 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

Características de una ERS

- 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.

Características de una ERS

- Electrónicamente almacenada:
  - Se encuentra en un archivo de texto, en una base de datos o, mejor aún, 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

Características de una ERS

- 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.

Características de una ERS

- 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

Características de una ERS

- 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.

Características de una ERS

- 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

Calidad de la ERS

- Una ERS perfecta es imposible.
- La calidad de la ERS es muy difícil de cuantificar.
- En general, una ERS de calidad NO garantiza la ausencia de problemas, pero una ERS pésima garantiza su presencia.

Validación de requisitos: revisiones

- Es la fórmula más empleada para validación
- Un grupo de personas (incluyendo usuarios) se ocupan de revisar el documento de requisitos.
- Tres fases: Búsqueda de problemas, reunión y 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

- 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 su “testeabilidad”. El equipo de pruebas debe revisar los requisitos.

Gestión de requisitos

- Consiste, básicamente, en gestionar *los cambios* a los requisitos.
- Asegura la consistencia ente los requisitos y el sistema construido (o en construcción)
- Consume grandes cantidades de tiempo y esfuerzo
- Abarca todo el ciclo de vida del producto

Necesidad de la gestión de requisitos

- Los requisitos son volátiles
- El entorno físico cambia
  - Trasladar un sistema de un entorno a otro requiere modificaciones
- El entorno organizacional cambia
  - 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

Qué implica la gestión de requisitos

- Definir procedimientos de cambios: definen los pasos y los análisis que se realizarán antes de aceptar los cambios propuestos (hay una Software Change Request (SCR) de la NASA de más de 500 páginas!!!).
- Cambiar los atributos de los requisitos afectados
- Mantener la trazabilidad: hacia atrás, hacia delante y entre requisitos
- Control de versiones del documento de requisitos

Viabilidad

- Tecnología: ¿hay tecnología? ¿se domina? ¿está dentro del estado del arte?
- Financiera: ¿pueden asumir el coste la organización, el cliente, el mercado?
- Tiempo: ¿llegará al mercado antes que la competencia?
- Recursos: ¿qué se va a necesitar? ¿está disponible?

- Relacionado con la experiencia en proyectos similares (si se han hecho muchos es más fácil decidir sobre la viabilidad de la propuesta)

Referencias

- Referencias
  - A. Davis. “Software Requirements: Objects, Functions and States”, Prentice-Hall, 1993
  - G. Kotonya, I. Sommerville. “Requirements Engineering. Processes and Techniques”. Wiley, 1998
  - B. L. Kovitz “Practical Software Requirements: A Manual of Content and Style” Manning 1999
  - I. Sommerville, P. Sawyer “Requirements Engineering. A Good Practice Guide”, Wiley, 1998
  - Pressman, capítulos 10 y 11
  - Sommerville, capítulos 5 y 6
- Material
  - Pablo Gervás
  - Andrés Silva