

# Ingeniería de Requisitos

---

## Tema 1. Introducción a la Ingeniería de Requisitos

*Departamento de Lenguajes y Sistemas Informáticos*

*Universidad de Alicante*

*Materiales cedidos para su uso docente por el prof. Emilio Insfrán Pelozo. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia*

# Contenido

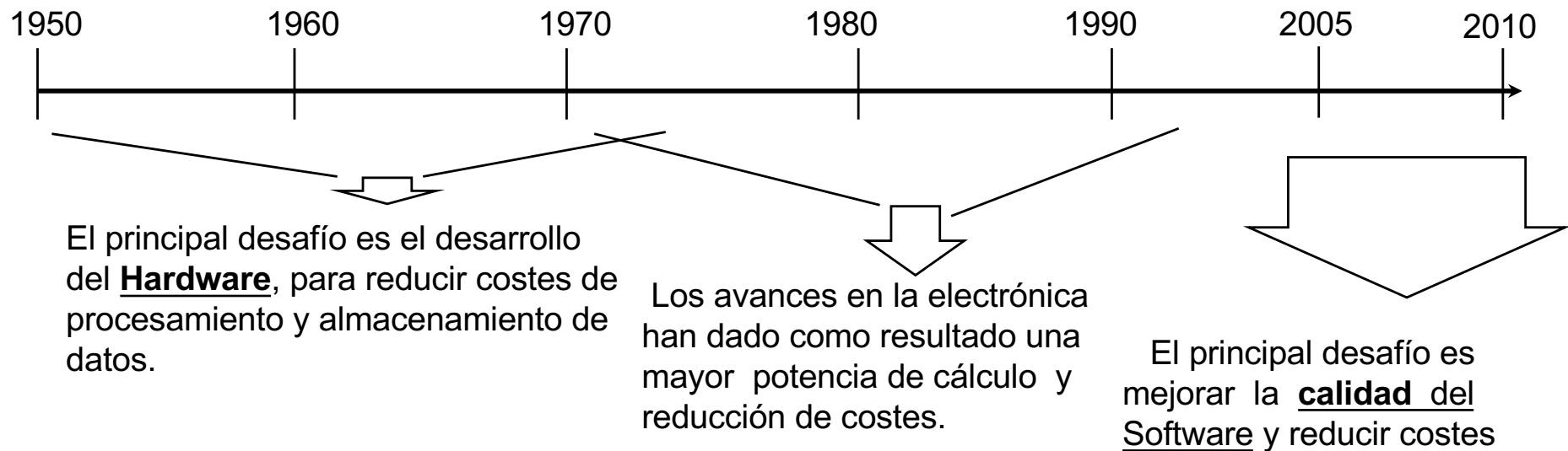
- Introducción
  - Motivación
  - Sistemas
  - Factores de Calidad del Software
  - Ingeniería del Software
    - Ciclos de Vida
- Ingeniería de Requisitos
  - Requisitos
  - Niveles de Requisitos
  - Definiciones
  - Actividades
  - Visión General

# Motivación


**Actualmente en los sistemas basados en ordenador, el software ha superado al hardware como factor decisivo de éxito.**

→ El software es el factor que marca la diferencia

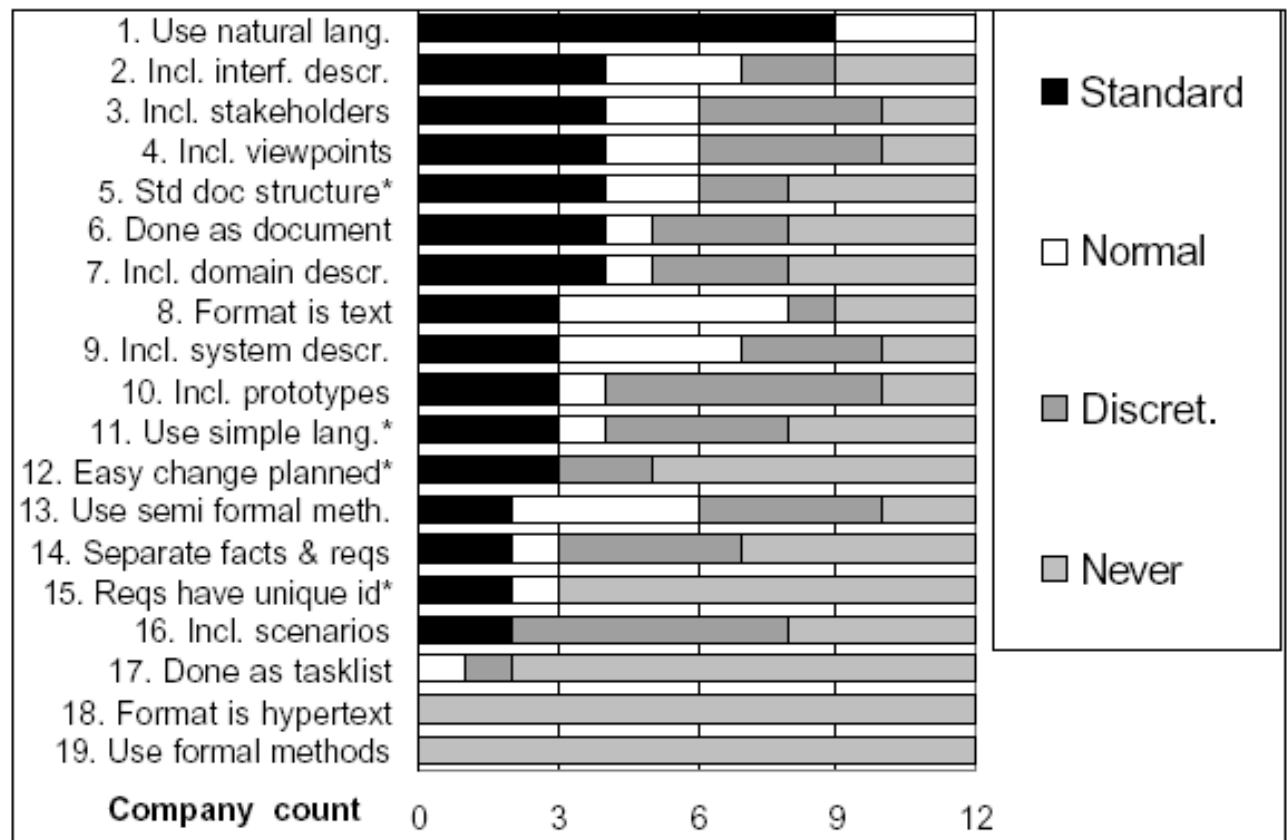
## Evolución



# Motivación

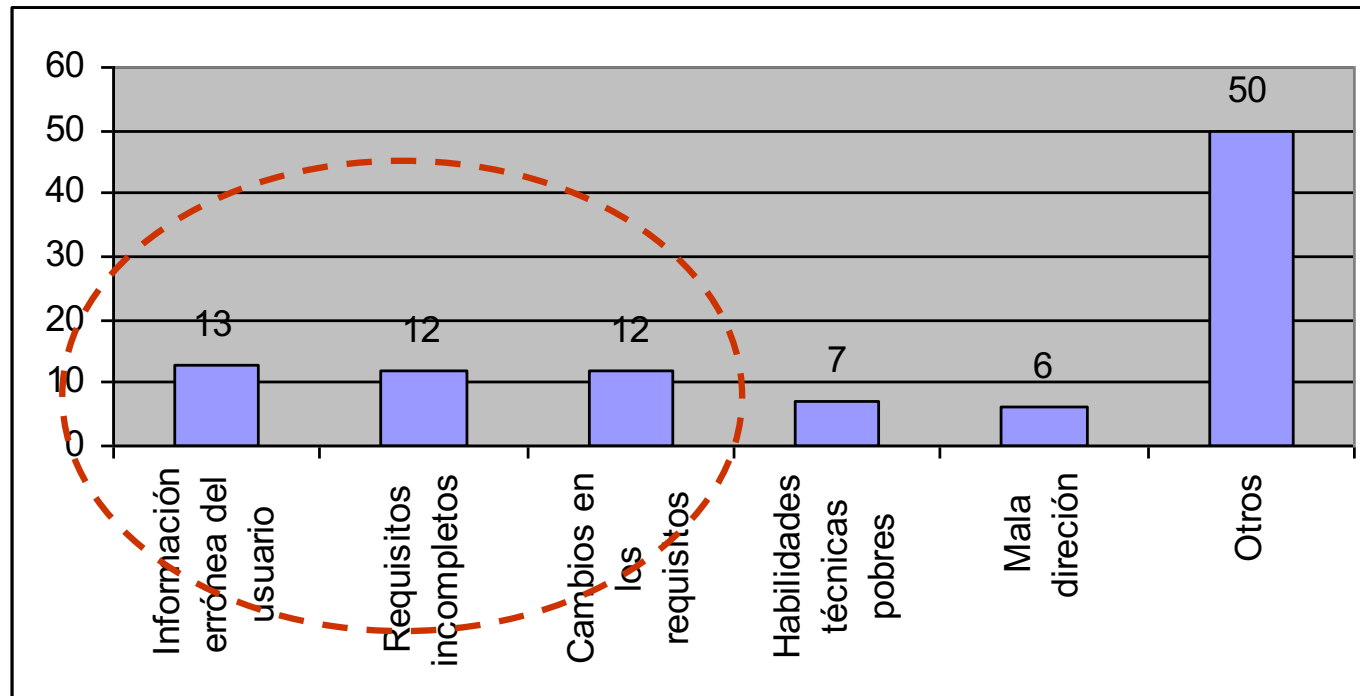
- Un estudio realizado en 100 compañías de desarrollo de software para la industria automovilística en la UE mostró que el 90% de los participantes usaban Lenguaje Natural editado con MS-Word.   
(Alen Dutoit and Barbara Peach, May 2000)

- Otro estudio sobre el estado de la práctica en IDR: (Telecom Business Research Center, Finland, 2001)



# Motivación

- **Standish Group**, encuesta con 8000 proyectos en 350 empresas de USA muestra que los requisitos son la principal fuente de problemas (37%)



- Un estudio similar en Europa (ESI) con 3800 organizaciones en 17 países concluye que la mayoría de los errores están relacionados con IDR (~50%)

Fuente: Standish Group, <http://www.standishgroup.com/chaos/toc.php>

# Motivación

- Costos asociados a la reparación de errores (de acuerdo al momento cuando es detectado).

ETAPA	COSTO DE REPARACIÓN
Requerimientos	1-2
Diseño	5
Codificación	10
Pruebas de Módulo	20
Pruebas de Sistema	50
Mantenimiento	100 - 200

(Estudios realizados en GTE, TRW e IBM midieron y asignaron costos a los errores ocurridos en las distintas fases de proyecto.

*Standish Group, <http://www.standishgroup.com/chaos/toc.php>*

# Motivación

Situación del Desarrollo de Sw en USA (58%) y Europa (24%) en los últimos años (Standish Group) (51% de las empresas pertenecen al grupo Fortune-1000)

	1994	1998	2002	2004	2006	2009
Proyectos entregados con éxito	16%	26%	34%	29%	35%	32%
Proyectos entregados con cambios (retrasos, sobrecostos, menos funcionalidad,...)	53%	46%	51%	53%	46%	44%
Proyectos cancelados	31%	28%	15%	18%	19%	24%

## Principales Factores de Éxito

- Participación del usuario
- Apoyo de la dirección ejecutiva
- Requisitos bien definidos

## Principales Factores de Fracaso

- Falta de *input* de los stakeholders
- Requisitos y especificaciones incompletas
- Cambios en los requisitos y en las especificaciones

## Motivación. Fallos de sistemas de acuerdo al tamaño

Resultado del proyecto (%)	Tamaño del proyecto en Puntos de Función*			
	<100	100 - 1000	1000 - 5000	>5000
Cancelado	3	7	13	24
Tarde > 6 meses	9	24	35	37
Tarde > 12 meses	1	10	12	18
Aprox. en tiempo	72	53	37	20
Antes de lo esperado	15	6	3	1

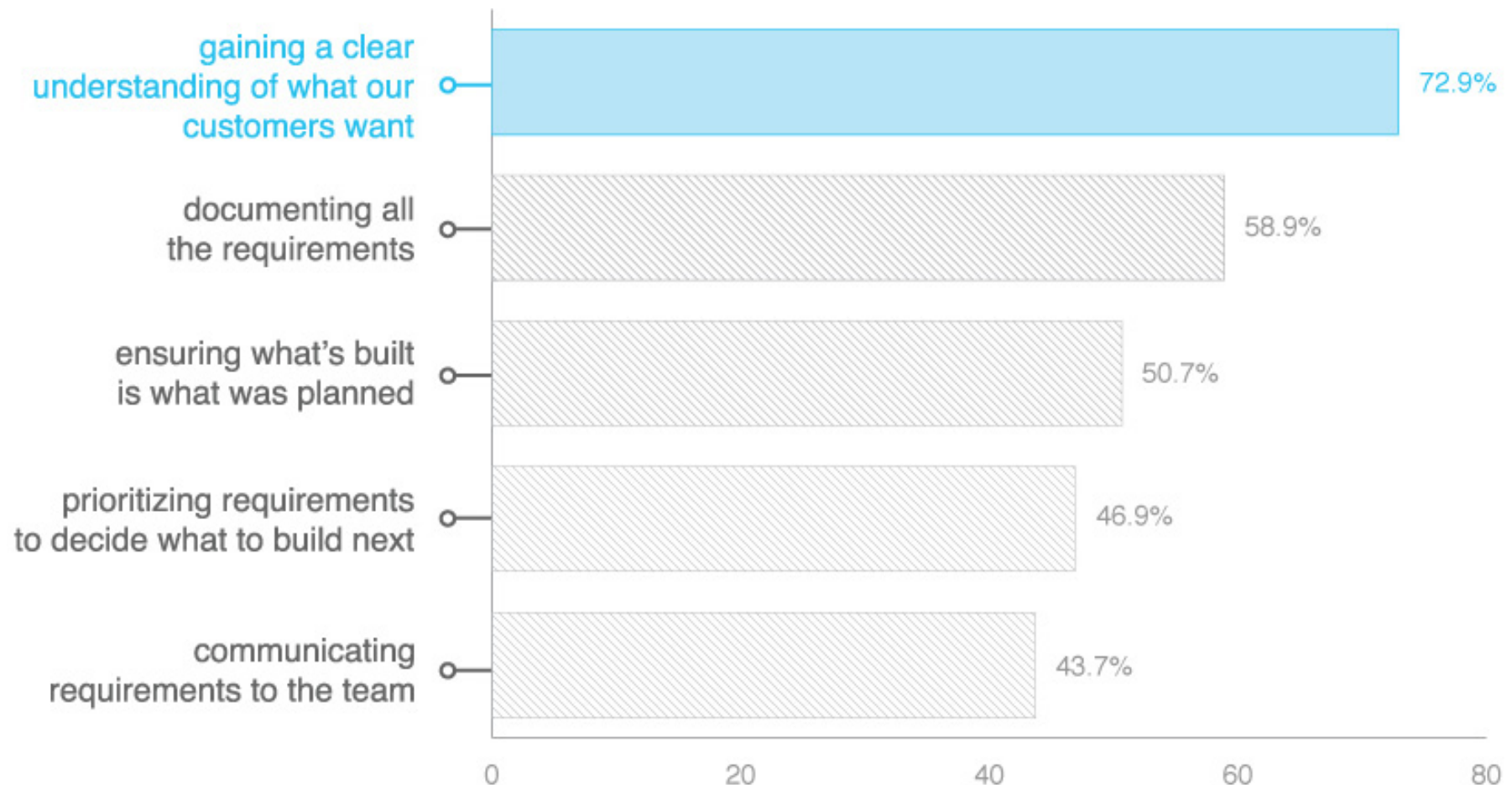
***\* 1 PF es aprox 100 líneas de código fuente en un lenguaje procedural***

**Fuente:** *IEEE Computer 28(3) basado en el estudio de Gartner Group*



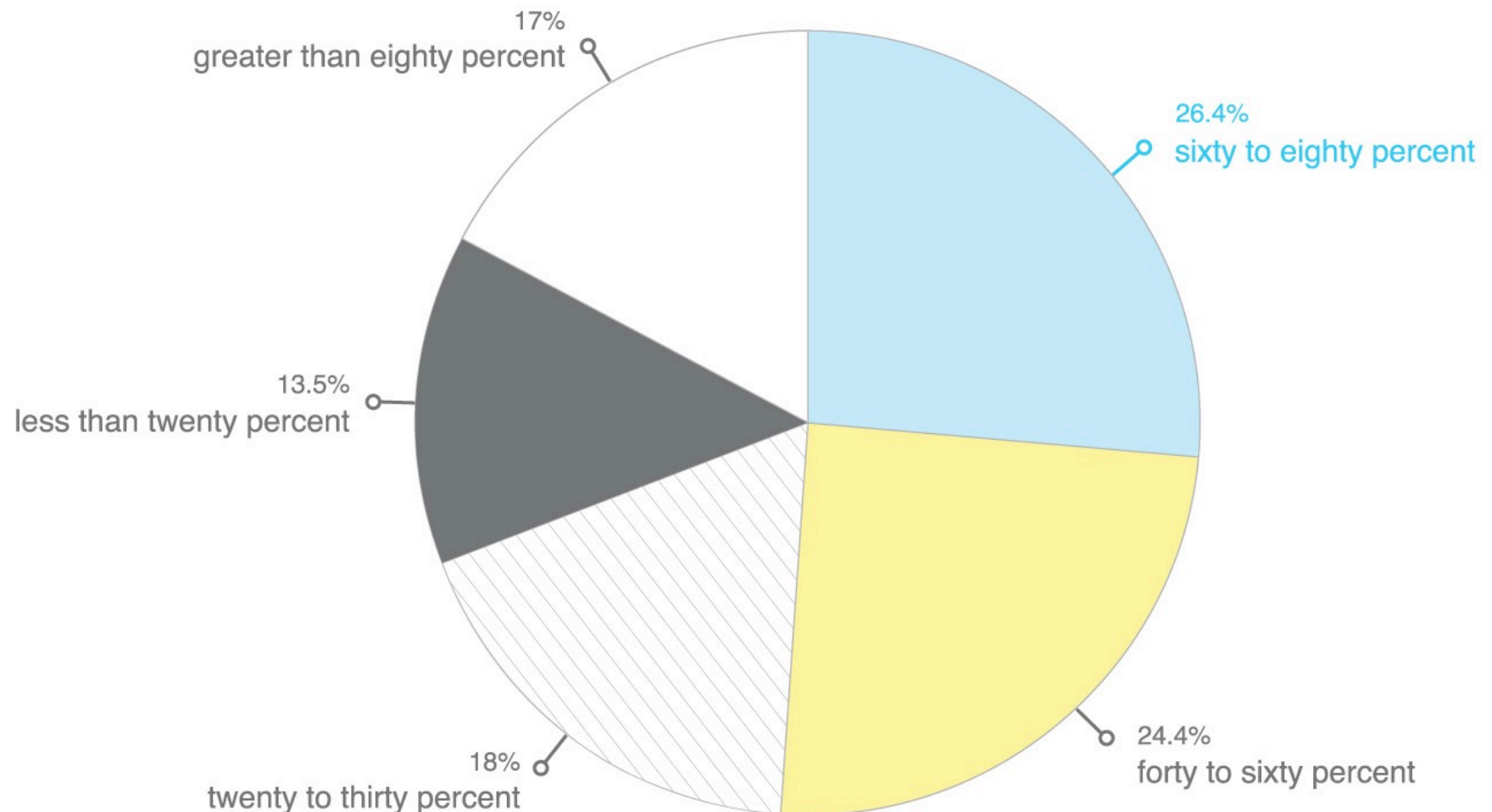
# Motivación – Survey Jama Software (2011) (1/5)

- 808 participantes de **todo el mundo** con diversos perfiles
- Principales desafíos



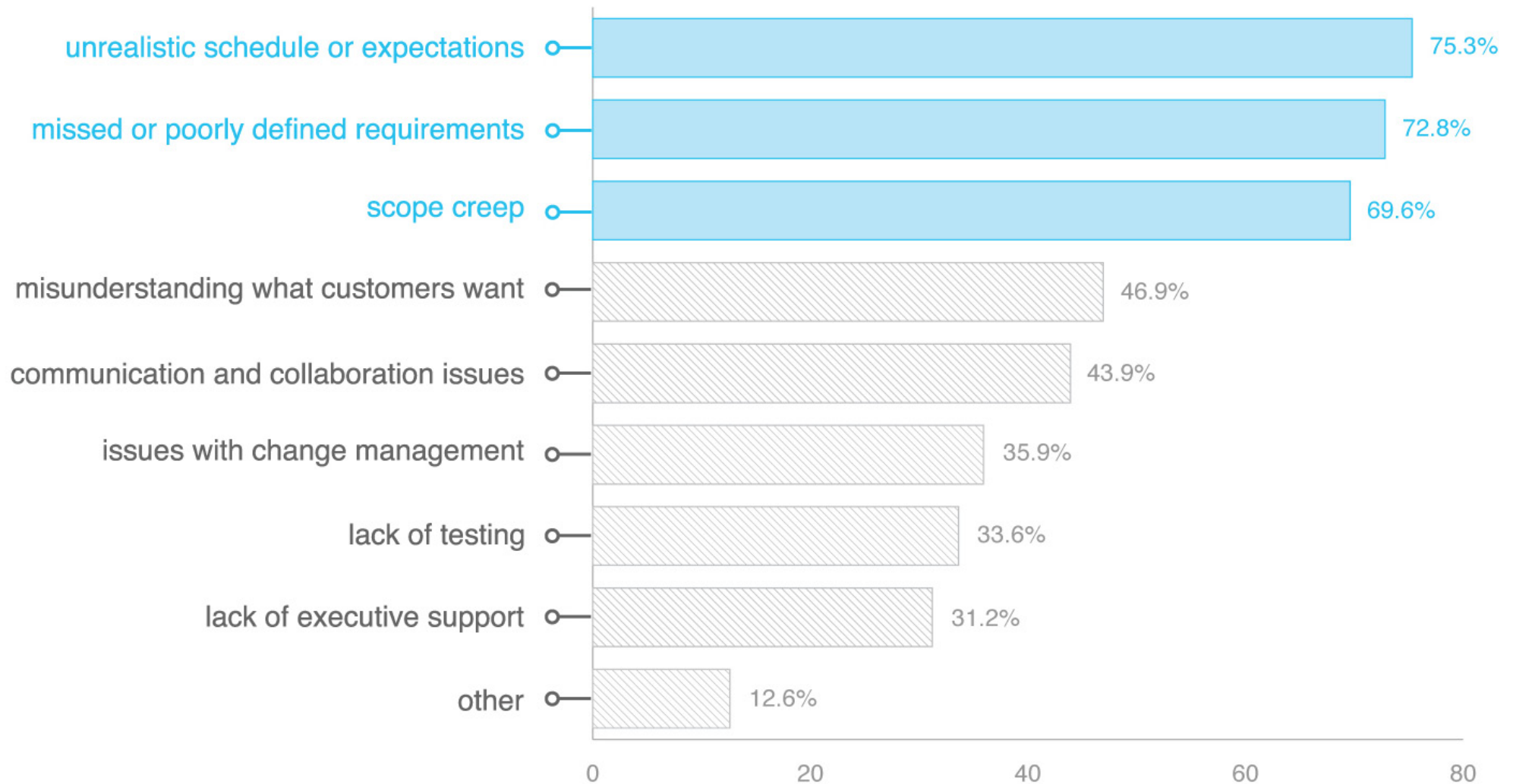
## Motivación – Survey Jama Software (2011) (2/5)

- Frecuencia con la que los proyectos son entregados a tiempo y dentro del presupuesto



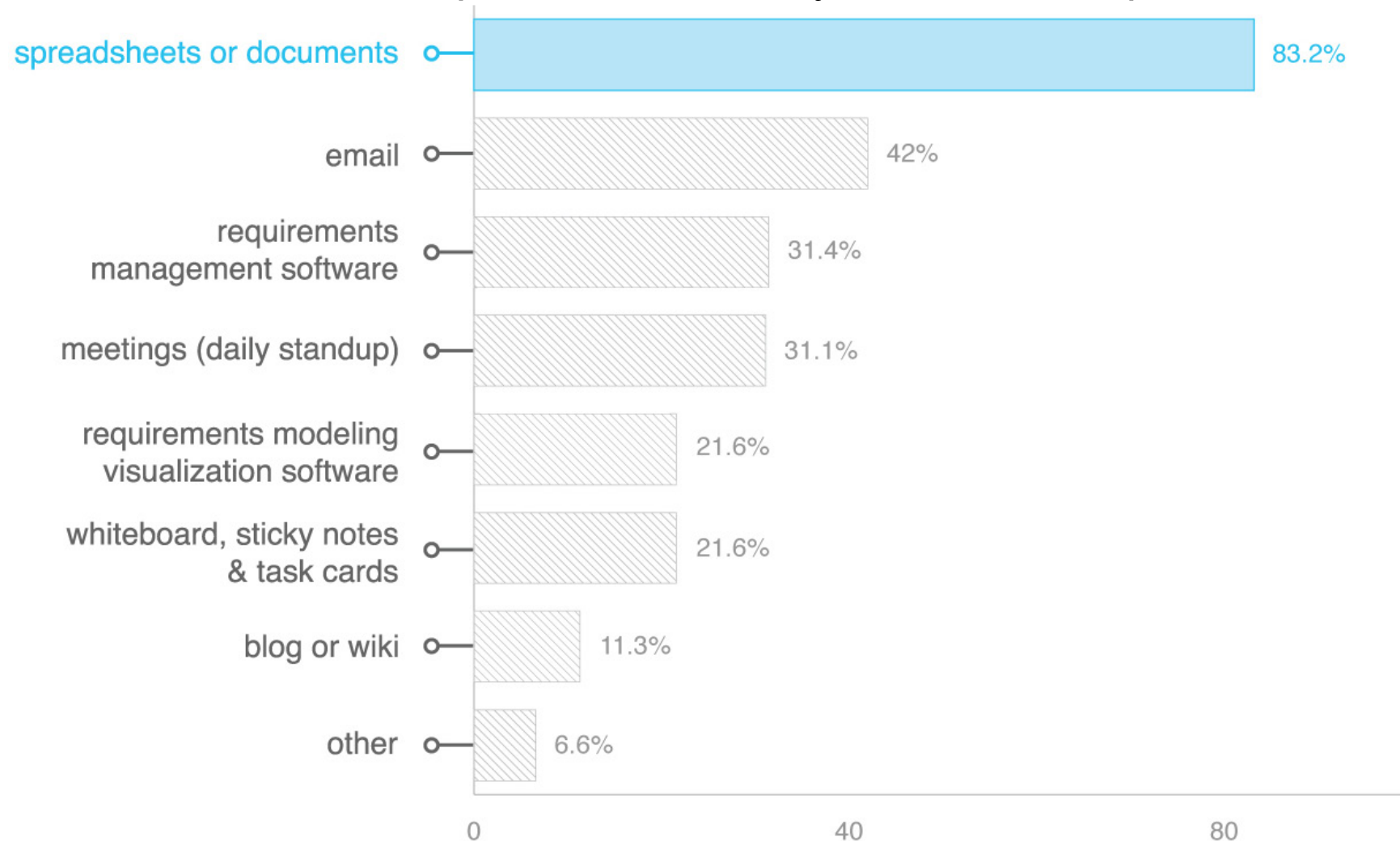
# Motivación – Survey Jama Software (2011) (3/5)

- Típicas causas de “fracaso”



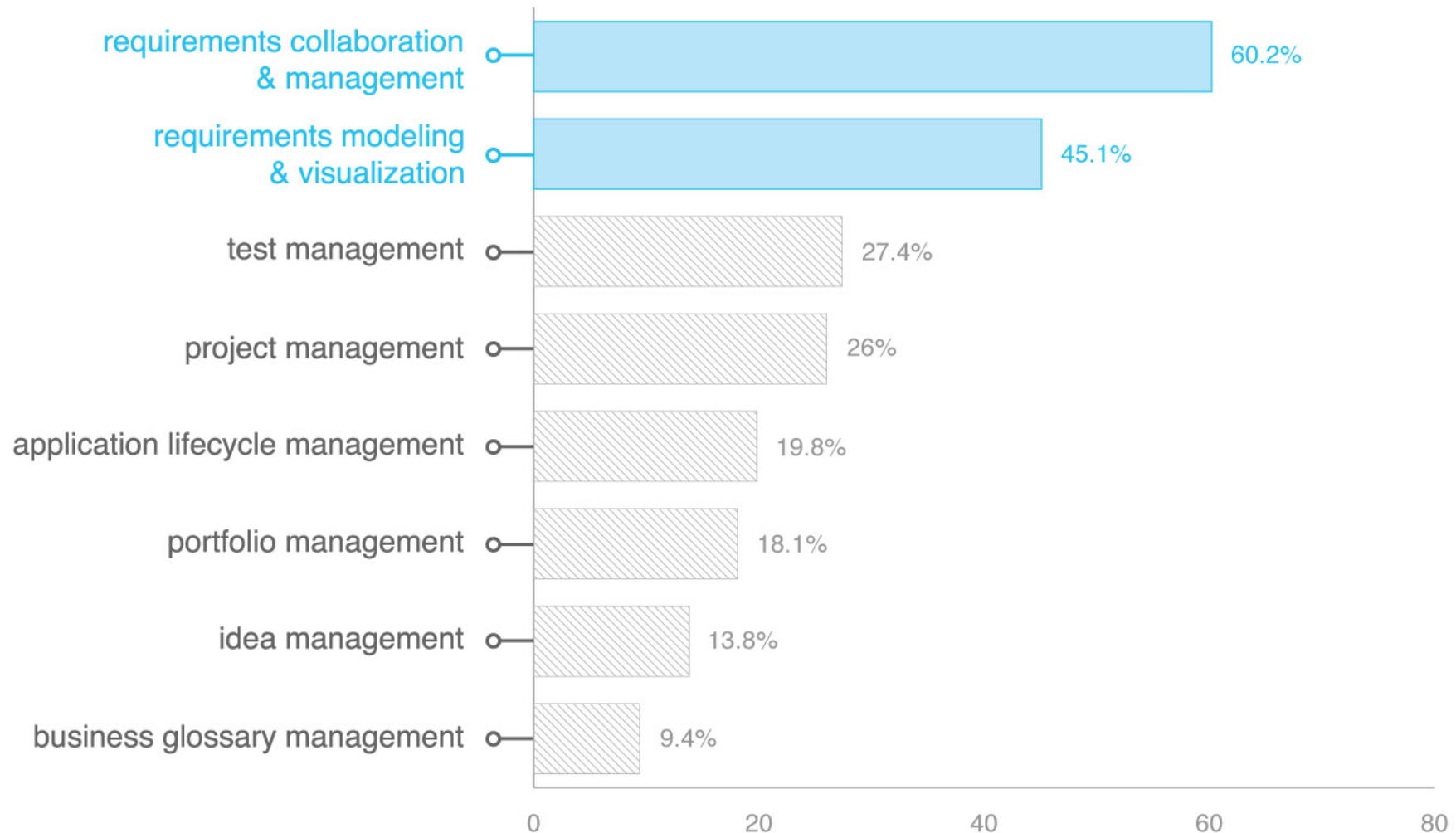
# Motivación – Survey Jama Software (2011) (4/5)

- Técnicas utilizadas para documentar y comunicar requisitos



# Motivación – Survey Jama Software (2011) (5/5)

- ¿Qué tipo de herramientas actualmente tendría más impacto para la mejora de éxito de los proyectos? (Wish list)



# Motivación

- Si el resultado de un sistema software no es satisfactorio, puede ser porque:
  - fue construido sin un entendimiento adecuado de su propósito.
  - fue construido sin seguir adecuadamente su especificación inicial.
  - es usado para un propósito distinto al que fue concebido.
- El *propósito* de un sistema software debe ser buscado más allá del propio sistema software:
  - Ej. el propósito de un sistema bancario no está en la tecnología usada para construir el sistema sino en las **actividades de negocio** bancarios y en las necesidades diarias de sus usuarios.

por tanto... la correcta identificación del **propósito** del sistema para satisfacer las necesidades de sus usuarios es fundamental!

# Motivación. ¿Por qué la IDR es difícil?

- Los usuarios pueden tener *dificultades en expresar sus necesidades* (o solicitar una solución que no satisface sus necesidades reales).
- Distintos usuarios pueden tener *necesidades opuestas*.
- Los usuarios pueden encontrar *difícil imaginar* nuevas formas de hacer cosas.
  - muchas veces sólo al ver el sistema desarrollado entienden que no satisface sus necesidades, aunque el sistema satisface los requisitos!
- Hay muchos tipos de requisitos, con muy distintos niveles de detalle.
- Su número puede ser inmanejable si no se controla adecuadamente.
- Algunas veces *no hay usuarios reales* ya que es un producto totalmente nuevo.
- Las *necesidades cambian* con el tiempo.

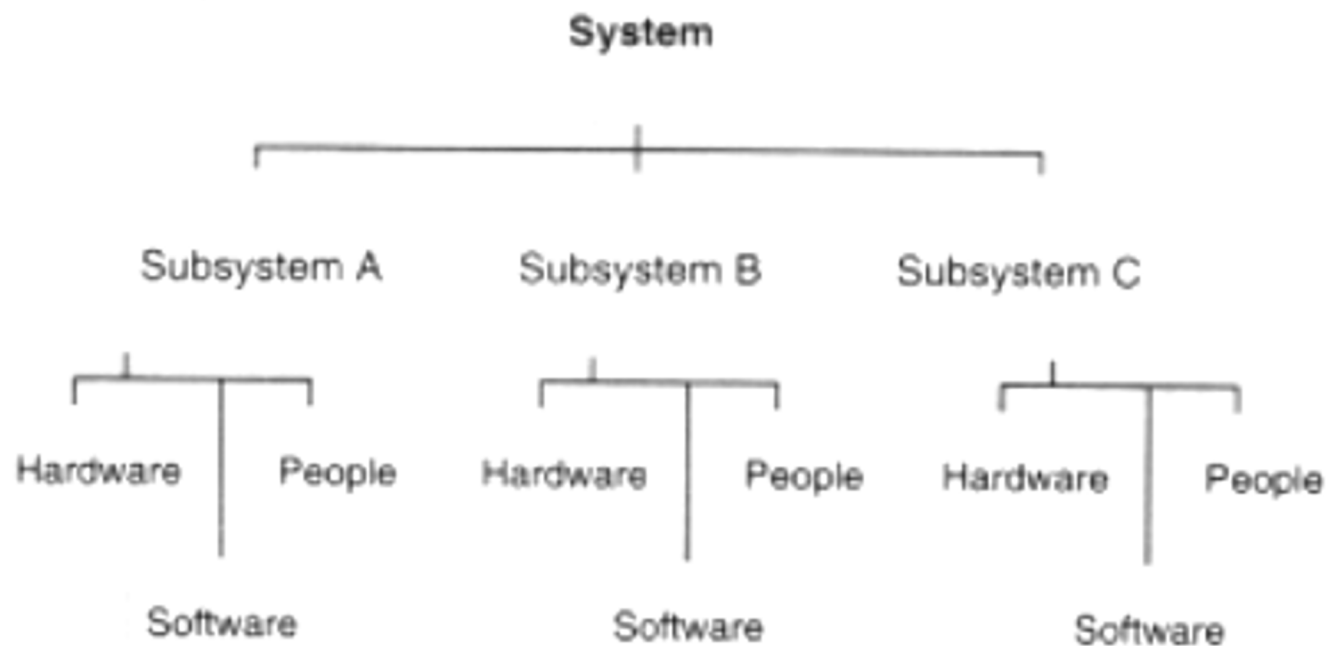
# Motivación. Desafíos para el Ingeniero de Requisitos

- La IDR trata principalmente sobre la “**comunicación**” entre los usuarios y el ingeniero de requisitos.
- Por tanto, los principales desafíos para el ingeniero de requisitos son:
  - **Entender** el dominio del problema.
  - **Capturar** los requisitos del usuario.
  - **Especificar** los requisitos adecuadamente.
  - **Validar** la especificación con el usuario.
  - **Trasmitir** los requisitos al equipo de desarrollo.
  - **Verificar** la correcta implementación de los requisitos.



# Sistemas

- Un sistema es “una colección de elementos interrelacionados que trabajan juntos para alcanzar un objetivo”.
- Algunos productos son **sistemas complejos**, compuestos de partes interrelacionadas o **subsistemas**, cada uno de ellos con sus propias capacidades operativas y conjuntos de requisitos.



# Sistemas

- **Requisitos del sistema:** define los requisitos de más alto nivel y que corresponde a uno o varios subsistemas.
  - A su vez, cada subsistema tiene sus propios requisitos que corresponde al hardware, software o personal que lo compone.
  - **Hardware:** componentes físicos o dispositivos
  - **Software:** controla la operación del procesamiento, los datos y los dispositivos.
  - **Personal:** individuos que operan y mantienen el sistema.
- **Requisitos software:** describe las capacidades del software necesarias para el sistema que se va a construir, mejorar o comprar.

# Sistemas. Tipos de sistemas Sw

Distintas *clasificaciones* para tipos de sistemas software:

- **Sistemas de Información:** orientados a la gestión de información en empresas para tratar sus operaciones o servicios de negocio.
- **Sistemas empotrados:** residentes en memoria para controlar sistemas hardware. Varían desde sistemas simples (ej. Reproductor de CD) hasta sistemas complejos (ej. Maquinaria de una planta química).
- **Software de ingeniería y científico:** orientadas al cálculo.
- **Sistemas expertos:** basados en técnicas de IA.
- **Sistemas de computación personal:** para uso personal o negocio (juegos, procesamiento de texto,...).
- **Sistemas de tiempo real:** monitorizar y analizar eventos.
- **Software de sistema:** software para servir a otros programas (SO, compiladores...).

# Sistemas. Complejidad de sistemas

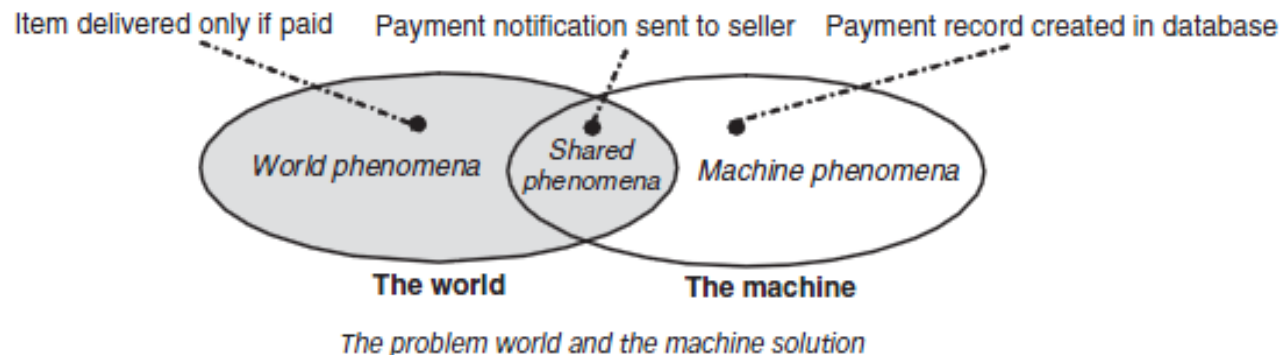
- Habitualmente, un sistema software debe dar soporte a diversas y complejas actividades humanas
  - diferentes tipos de usuarios
  - diferentes intereses,
  - diferentes puntos de vista,
  - diferentes prioridades,
  - ...

# Sistemas. Complejidad de sistemas

- La IDR proporciona técnicas para el tratamiento de la complejidad basada en los principios de:
  - **Abstracción**: se ignoran los detalles. Un conjunto de elementos y actividades persona-ordenador se pueden describir de forma “esencial”. Ejemplo: diagrama de dominio
  - **Descomposición**: un conjunto de fenómenos se descomponen en partes. Ejemplo: diagrama de subsistemas.
  - **Proyección**: adopta una vista o perspectiva particular. Diferente a la *descomposición* porque no particiona el espacio del problema sólo la observa desde una perspectiva. Ejemplo: una vista estática o dinámica.
  - **Modularización**: elige estructuras estables para identificar y representar abstracciones (subsistemas, funciones,...). Ejemplo: entidades, actividades, actores, casos de uso, ...
- El uso sistemático de la *abstracción*, *descomposición*, *proyección* y *modularización*, permite tratar la complejidad haciendo los problemas más simples.

# Sistemas. Construcción de sistemas

- El **propósito** de la construcción de un sistema software es solucionar un problema del mundo real o mejorar sus condiciones.
- **Dominio del problema** (el mundo real): se refiere a un área organizacional, técnico o físico. Ej. Departamento de créditos de un banco.
- **Dominio de la solución** (el ordenador): se refiere a los elementos en el contexto del sistema software. Ej. Los componentes internos del sistema software, SO, redes, etc.

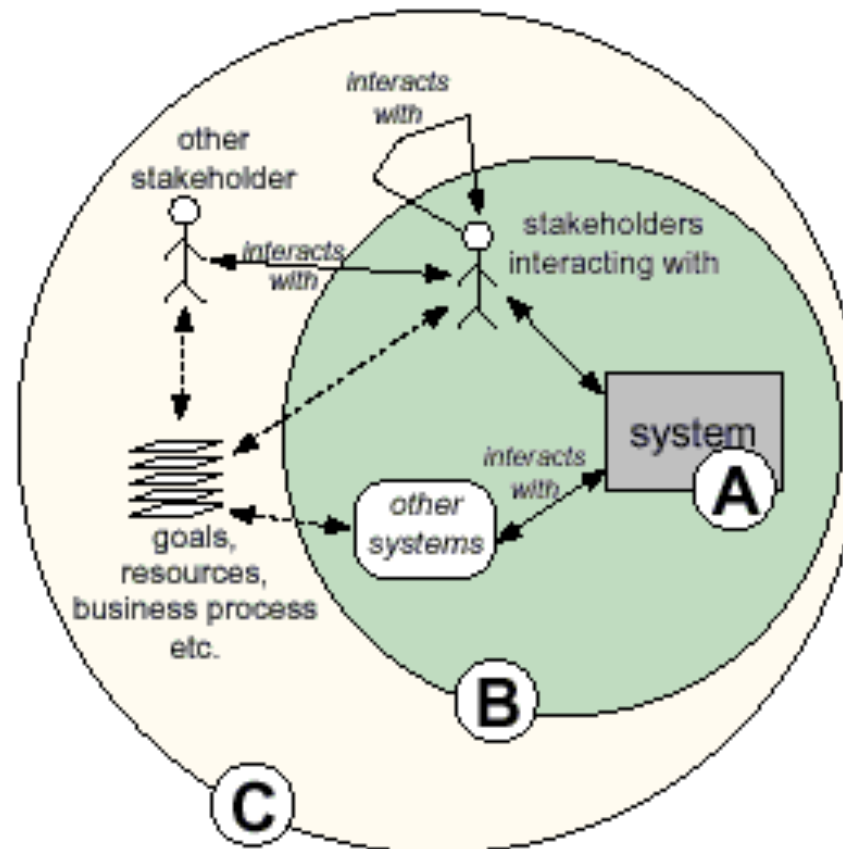


La IDR se refiere a los efectos del sistema sobre el mundo real y las asunciones sobre este mundo.

Por tanto, estudia los fenómenos del mundo real incluyendo los compartidos pero NO las características internas de la solución!.

# Sistemas. Construcción de sistemas

- Enfoques para la IDR



ESPRIT Project CREWS (96-99)

(C. Rolland, N. Meiden, M. Jarke, K. Phol, E. Dubois, B. Achour, A. Sutcliffe,...)

# Sistemas. Construcción de sistemas

Considerar un sistema de **Subasta de Artículos**.

- Este **sistema actual** se compone de componentes, tales como: *vendedores, compradores, empresas de mensajería*, un subsistema independiente de *pago electrónico* y sistemas de *correo electrónico*.
- Un nuevo **sistema a desarrollar** de *Subasta Electrónica en Internet* permitirá la gestión de artículos, publicidad, ofertas, facturación al mejor postor, registro de evaluaciones de vendedores y compradores.
- El **objetivo** es aumentar la *satisfacción* de los compradores al conseguir un mayor acceso a artículos interesantes, la *satisfacción* de los vendedores al obtener un acceso más amplio a compradores potenciales, y *automatizar* las normas que regulan el sistema de subasta.

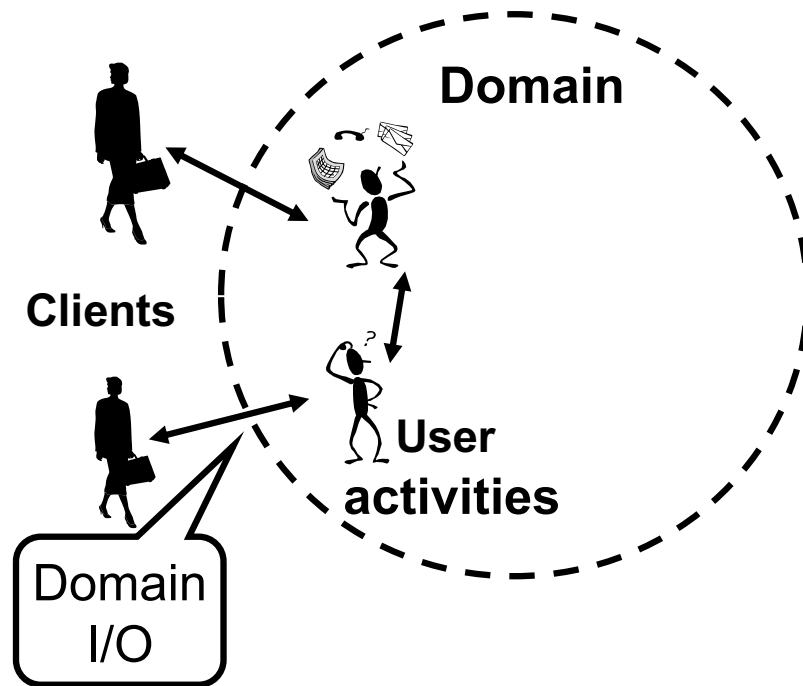


# Sistemas. Construcción de sistemas

Dos versiones del mismo sistema:

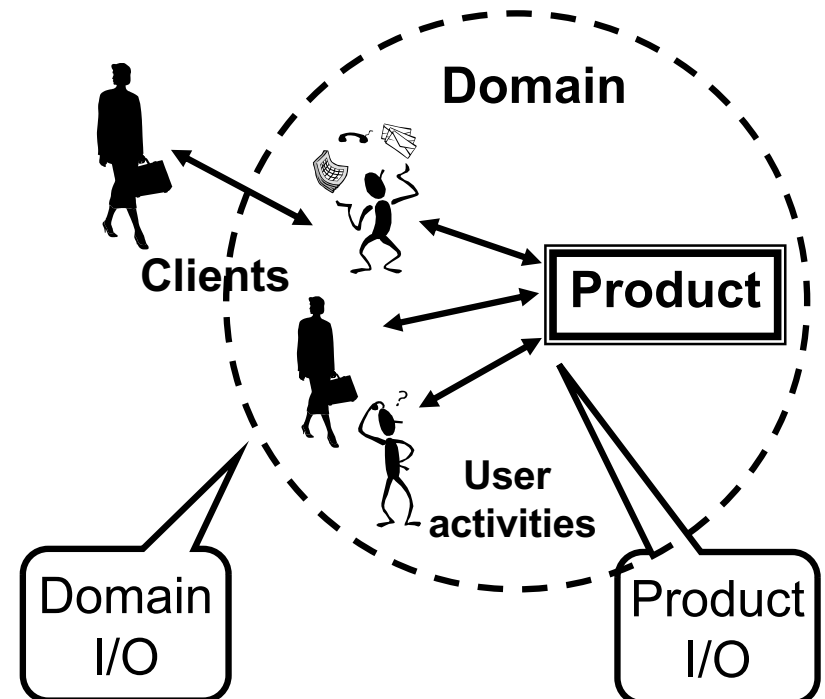
- **Sistema actual**

- Es el sistema como existe *antes* de la incorporación del sistema software.



- **Sistema a desarrollar**

- Se refiere al nuevo sistema que será incorporado en el dominio del problema.



# Sistemas. Construcción de sistemas

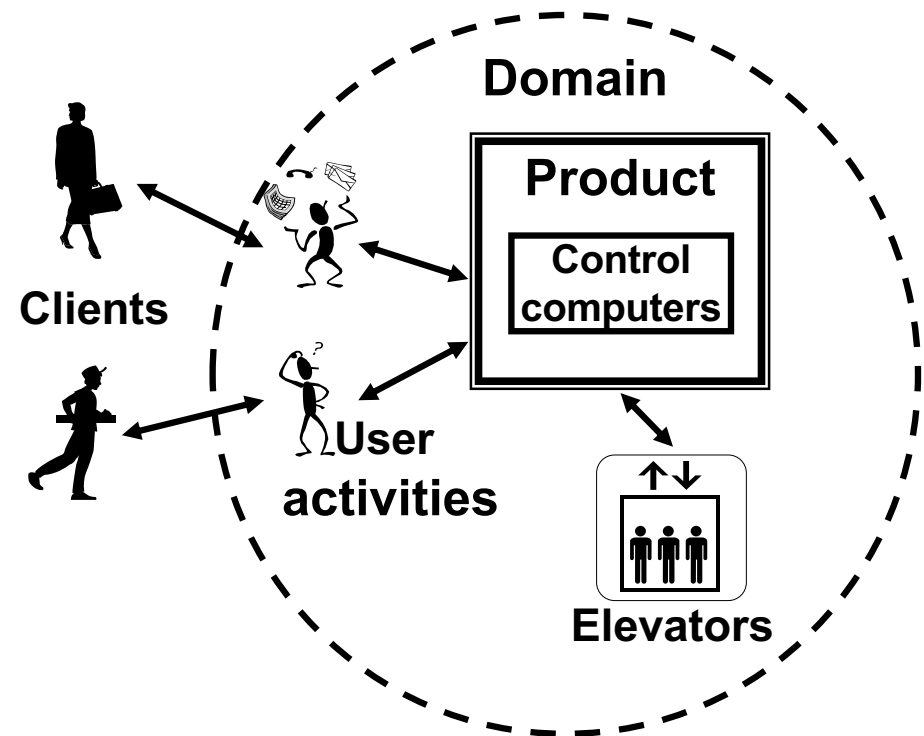
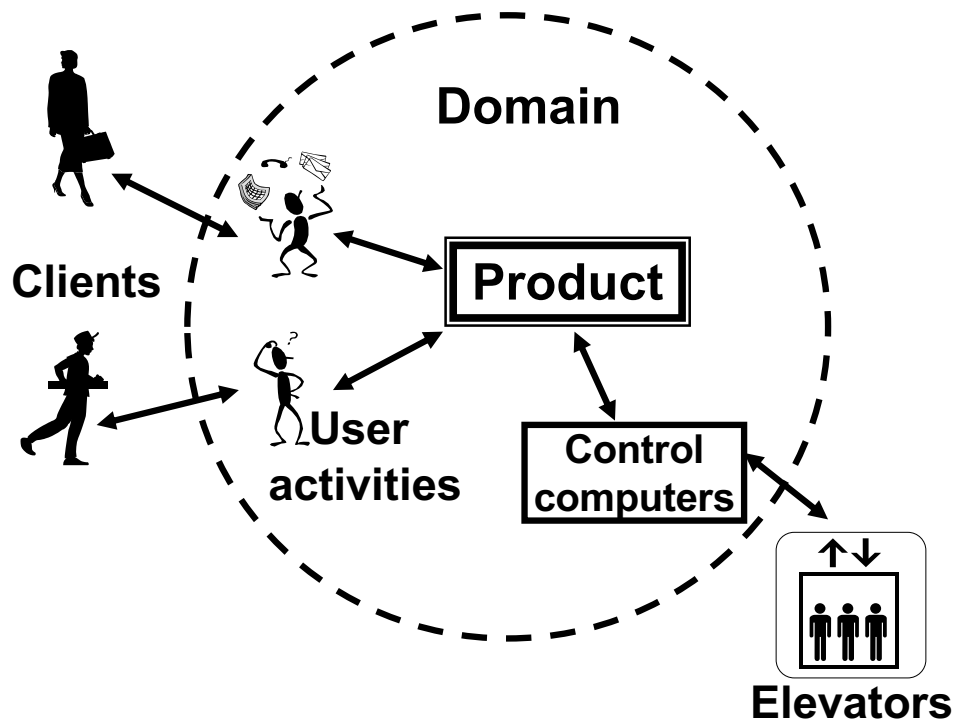
- Además del *sistema actual* y del *sistema a desarrollar*...
- **Sistema futuro** (evolución – preparación para el cambio)
  - Para satisfacer las necesidades de los usuarios, el sistema software debe estar preparado para el cambio.
  - Cambios durante el desarrollo.
  - Cambios después de la implantación del sistema.
- Una parte importante de la IDR (**Gestión de Requisitos**) se encarga de gestionar los cambios en el desarrollo y de visionar las siguientes versiones del sistema software.

# Sistemas. Límites (frontera) del sistema Sw

**Ejemplo:** Un sistema de gestión para ascensores

- **Sistema de información** para registrar el personal de reparación, el plan de mantenimiento, las revisiones, etc.
- Además, comunicación con otro **Sistema de Control** para comprobar el estado de los ascensores, realizar verificaciones rutinarias, monitorizar cámaras de seguridad, etc.

- *Alternativa:* desarrollar el sistema de **información** y de **control**



# Factores de Calidad del Software

- La meta es producir software de calidad.

**¿ Qué es software de calidad ?**

Concordancia con:

- los requisitos funcionales y de rendimiento explícitamente establecidos,
- con los estándares de desarrollo explícitamente documentados y
- con las características implícitas que se espera de todo software desarrollado profesionalmente.

... esta definición puede ser ampliada, puesto que *la calidad del software* es una compleja mezcla de ciertos factores que varían para las diferentes aplicaciones y los clientes que las solicitan.

# Factores de Calidad del Software

- **Factores que determinan la calidad del software.**
  - (Según McCall) La clasificación de los factores de calidad se centra en tres aspectos importantes de un producto software:
    - ( 1 ) Sus características operativas.
    - ( 2 ) Su capacidad de soportar los cambios.
    - ( 3 ) Su adaptabilidad a nuevos entornos.

# Factores de Calidad del Software

- **Factores que determinan la calidad del software** (McCall)

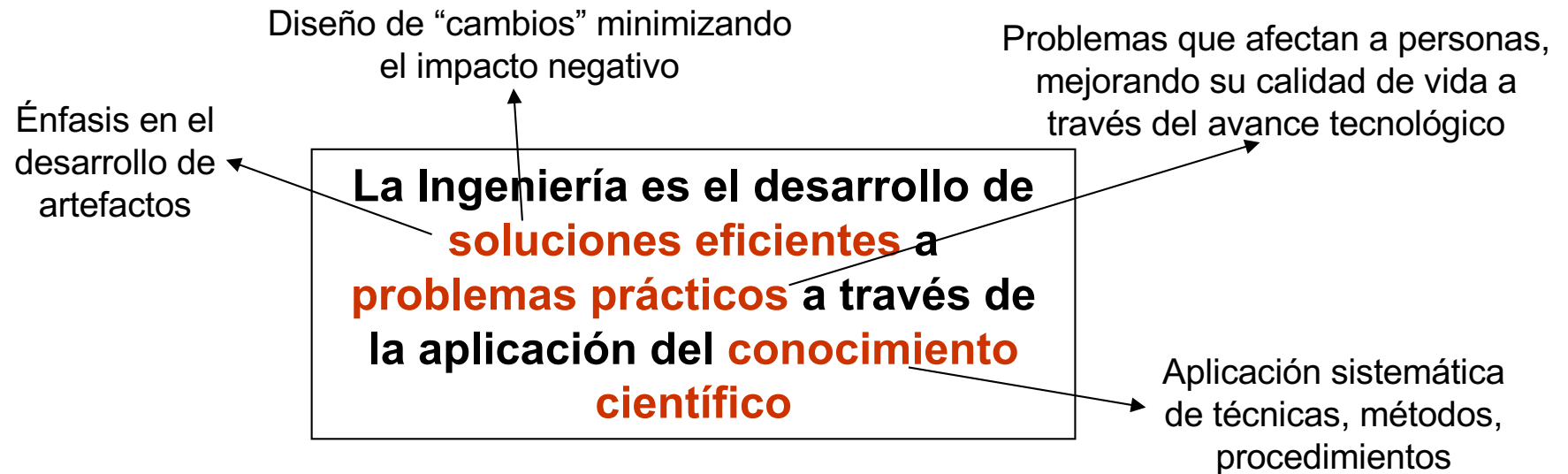
- **Corrección** (¿Hace lo que quiero?)
  - **Fiabilidad** (¿Lo hace de forma fiable todo el tiempo?)
  - **Eficiencia** (¿Se ejecutará en el Hw lo mejor que pueda?)
  - **Integridad** (¿Es seguro?)
  - **Facilidad de uso** (¿Está diseñado para ser usado?)
- (1)

- **Facilidad de mantenimiento** (¿Puedo corregirlo?)
  - **Flexibilidad** (¿Puedo cambiarlo?)
  - **Facilidad de prueba** (¿Puedo probarlo?)
- (2)

- **Portabilidad** (¿Podré usarlo en otra máquina?)
  - **Reusabilidad** (¿Podré reusar alguna parte del Sw?)
  - **Facilidad de interoperación** (¿Podré hacerlo interactuar con otro sistema?)
- (3)

# Ingeniería del Software

- **Definición de Ingeniería:**



- **Definiciones para la Ingeniería del Software:**

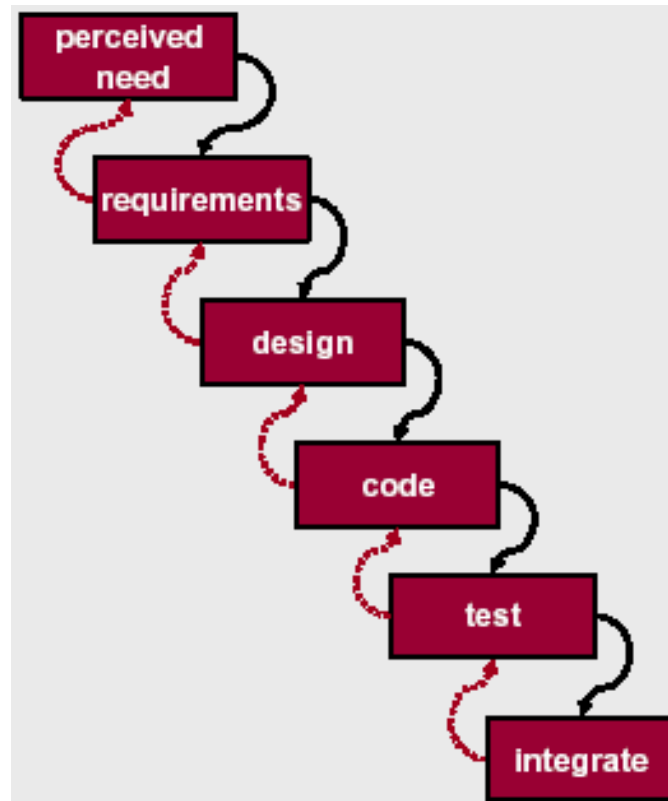
- *Boehm*, “La IS supone la aplicación del conocimiento científico al diseño y construcción de Sw y la documentación asociada requerida para su desarrollo, operación y mantenimiento”
- *Pressman*, “La IS es una disciplina que integra **métodos**, **herramientas** y **procedimientos** para el desarrollo de Sw ”

# Ingeniería del Software

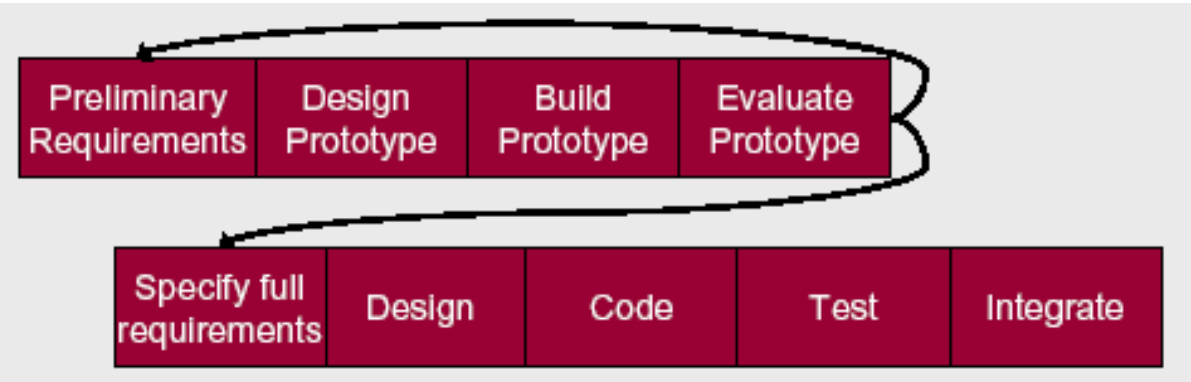
- La IS es algo más que programar.
- El proceso de la IS comienza bastante antes de escribir líneas de código y continúa después de que la primera versión del producto haya sido completada.
- Las etapas por las que atraviesa un producto SW a lo largo de su existencia se denomina Ciclo de Vida.
- Se han propuesto varios paradigmas diferentes:
  - Paradigma clásico o en cascada
  - Paradigma clásico con prototipado.
  - Paradigma de programación automática
  - Paradigma en espiral.
  - ...
- A pesar de las ventajas e inconvenientes de cada uno de ellos, todos tienen una serie de etapas/actividades en común: análisis, diseño, implementación, pruebas, ...



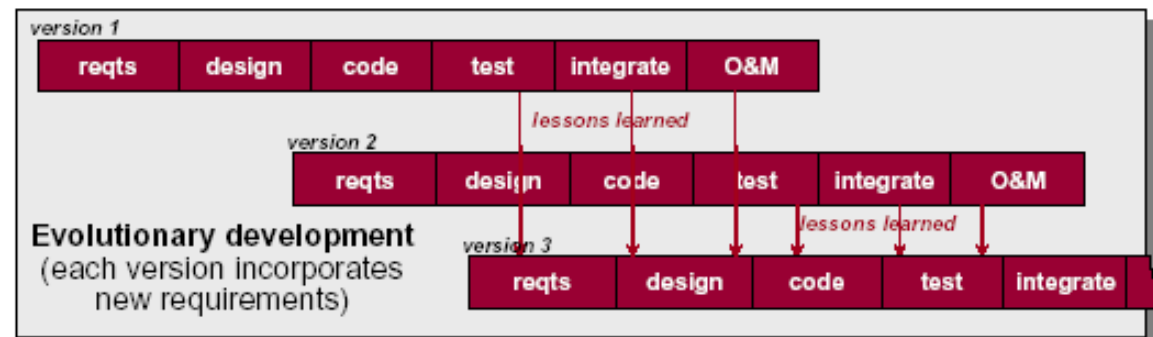
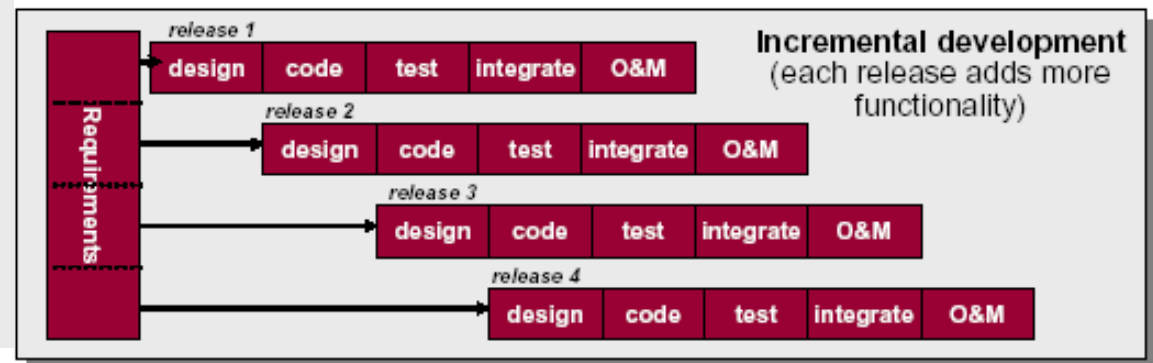
# Ingeniería del Software. Ciclos de Vida



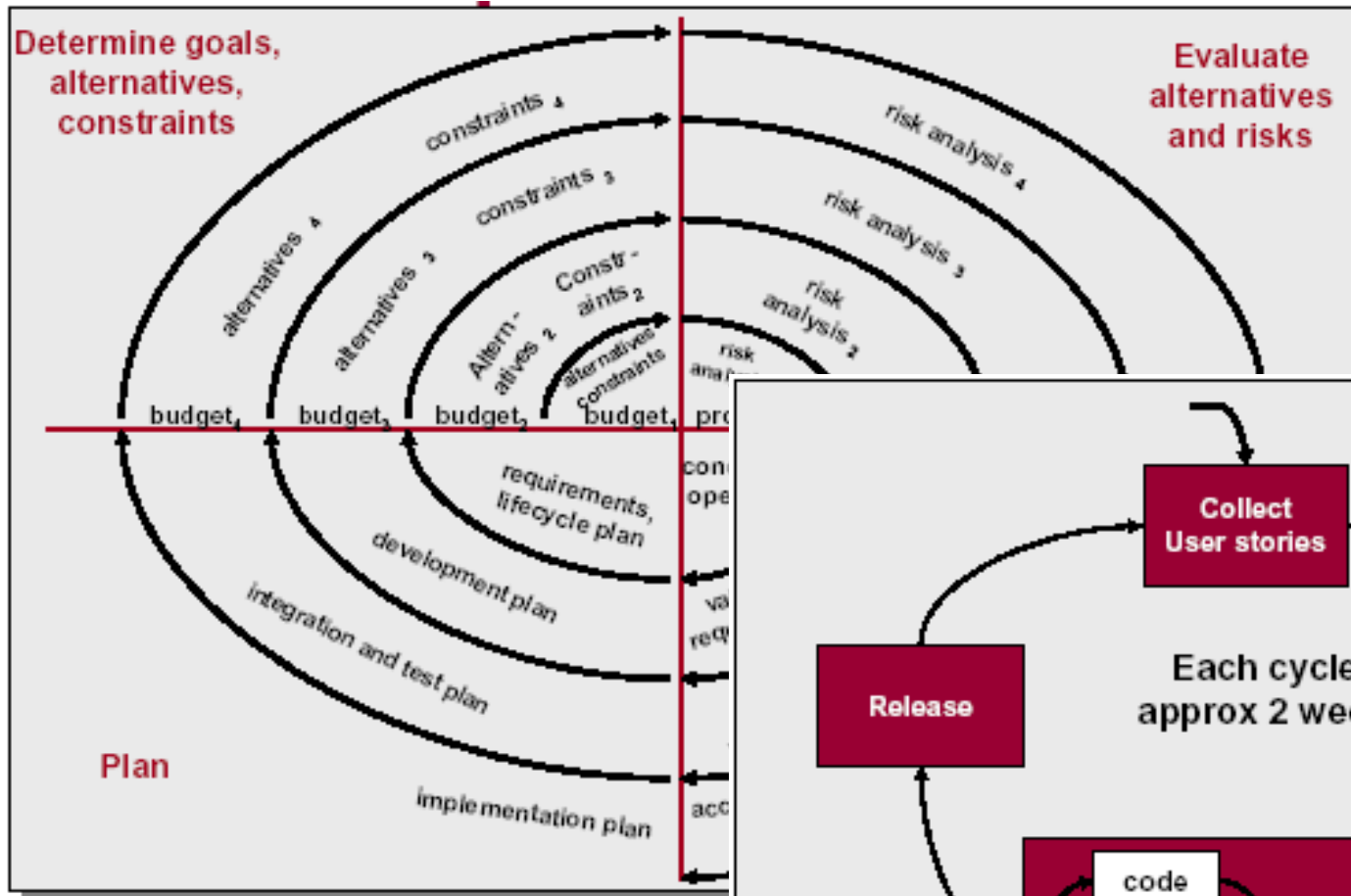
Cascada



Con Prototipos

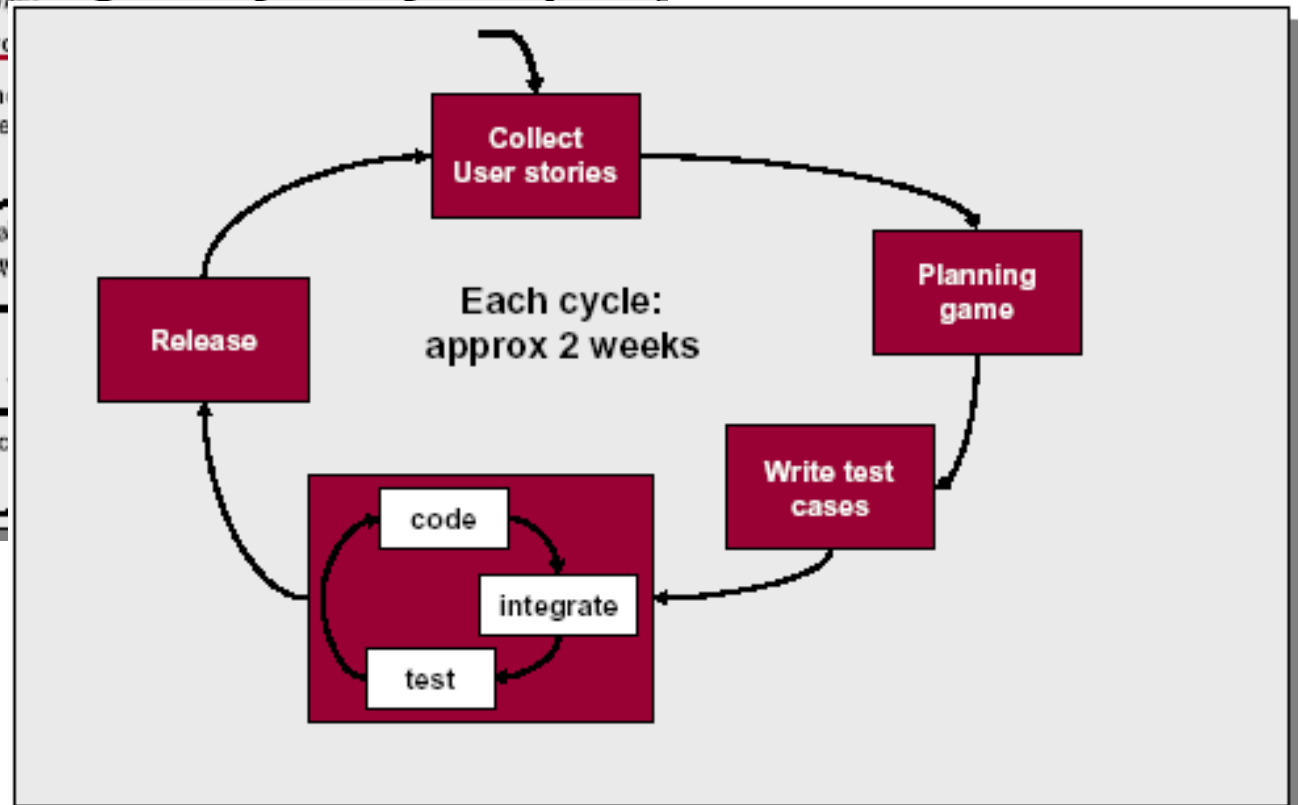


# Ingeniería del Software. Ciclos de Vida



En Espiral

eXtreme Programming

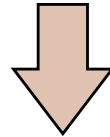


# Contenido

- Introducción
  - Motivación
  - Sistemas
  - Factores de Calidad del Software
  - Ingeniería del Software
    - Ciclos de Vida
- Ingeniería de Requisitos
  - Requisitos
  - Niveles de Requisitos
  - Definiciones
  - Actividades
  - Visión General

# Ingeniería de Requisitos

- Dentro del proceso de construcción de Sw, las primeras actividades consisten en:
  - *Identificar, analizar, definir y especificar* de los requisitos del sistema.



**Ingeniería de Requisitos**

# Ingeniería de Requisitos

## REQUISITOS

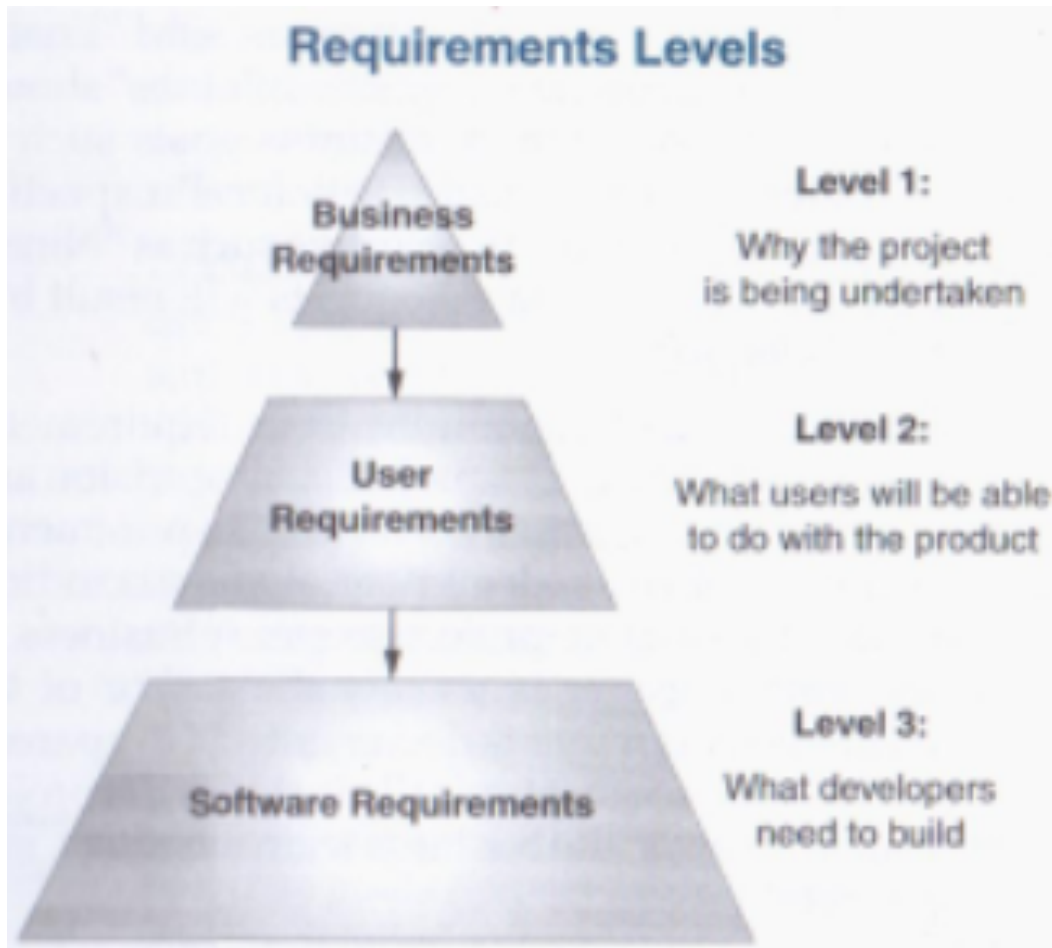
- Diccionario: “Condición necesaria para una cosa”.
- Según Std.610.12-1990, IEEE Standard Glossary of Software Engineering Terminology,
  - (1) “Una condición o capacidad necesaria para que un usuario resuelva un problema o alcance un objetivo”
  - (2) “Una condición o capacidad que debe reunir o poseer un sistema o un componente de un sistema para satisfacer un contrato, un estándar, una especificación u otros documentos impuestos formalmente”.
  - (3) Una representación documentada de una condición o capacidad como en (1) o (2).

# Ingeniería de Requisitos. Ejemplos de requisitos

- Los requisitos (software), por tanto, definen qué tiene que hacer el sistema y bajo qué circunstancias debe operar.
- **Ejemplos de requisitos** para una Biblioteca:
  1. El sistema debe mantener un registro de los materiales de la biblioteca incluyendo libros, revistas, periódicos, video, audio, informes y CDs.
  2. El sistema debe permitir a los usuarios buscar cualquier material por título, autor o ISBN.
  3. El interfaz de usuario debe ser un navegador Web.
  4. El sistema debe soportar al menos 20 transacciones por segundo.
  5. El resultado de las búsquedas debe tener mecanismos de paginado.
- Los requisitos pueden ser de distintos **tipos**:
  - (1) Requisito de *muy alto nivel*.
  - (2) Requisito de *funcionalidad*.
  - (3) Requisito de *implementación*.
  - (4) Requisito de *rendimiento*.
  - (5) Requisito de *usabilidad*.

# Ingeniería de Requisitos. Niveles de requisitos

Los requisitos del software pueden estar en 3 niveles:



- **Requisitos del negocio:** describe el propósito de alto nivel y las necesidades del producto para aumentar ganancias, reducir gastos,...
- **Requisitos de usuario:** describe las tareas que los usuarios necesitan realizar con el software.
- **Requisitos del software:** son descripciones de las necesidades *funcionales* y *no funcionales* que le software debe realizar para satisfacer los requisitos de usuario y del negocio.

# Ingeniería de Requisitos. Niveles de requisitos

- Considerando el sistema de **Subasta de Electrónica de Artículos en Internet:**.

**R1.** Aumentar la *satisfacción* de los compradores al conseguir un mayor acceso a artículos interesantes.

**Nivel de negocio**

**R2.** Gestionar artículos a subastar.

**Nivel usuario/dominio**

**R3.** Registrar un artículo a subastar indicando sus datos identificativos, descripción y precio de salida y mínimo.

**Nivel software (funcional)**

**R4.** Las transacciones de venta deben ser seguras usando el protocolo SSL.

**Nivel software (no funcional)**



# Ingeniería de Requisitos. Niveles de requisitos

## Requisito a nivel de negocio

**R1.** Aumentar la *satisfacción* de los compradores al conseguir un mayor acceso a artículos interesantes.

- Este requisito establece un **objetivo del negocio**, que es bueno porque es lo que la empresa realmente quiere!
- El requisito puede ser verificado, pero sólo después de algún periodo de uso.
- Este requisito es un objetivo y no dice **qué** tiene que hacer el sistema para lograrlo.

# Ingeniería de Requisitos. Niveles de requisitos

## Requisito a nivel dominio / usuario

### R2. Gestionar artículos a subastar.

- Es un requisito a nivel de usuario / dominio. Describe una *actividad realizada en el dominio por un usuario* y que se espera el sistema dé soporte.
- El analista debe reconocerlo como un requisito de alto nivel. Su experiencia le dice que esta función NO se realiza explícitamente en el dominio sino que habrá funciones más específicas que satisfarán este requisito (dar de alta artículo, cambiar sus datos, subastarlo,...)
- Se puede contratar con este tipo de requisitos? Depende de la experiencia de la empresa de Sw con el dominio...
- Se puede verificar el requisito? Sí, incluso antes de la entrega del producto.

# Ingeniería de Requisitos. Niveles de requisitos

## Requisito a nivel producto (funcional)

**R3.** Registrar un artículo a subastar indicando sus datos identificativos, descripción y precio de salida y mínimo.

- Es un **requisito funcional del producto** software. Describe entradas y salidas.
- Básicamente identifica las funciones/aspectos/características que el sistema debe proveer sin dar todos los detalles.
- Se puede contratar con este tipo de requisitos? Sí (si la empresa de Sw no conoce suficientemente el dominio se deben dar más detalles)
- Se puede verificar el requisito? Sí, antes de la entrega del producto.

# Ingeniería de Requisitos. Niveles de requisitos

## Requisito a nivel producto (no funcional)

**R4.** Las transacciones de venta deben ser seguras utilizando el protocolo SSL.

- Es un **requisito de diseño del producto**.
- Aunque indica exactamente el protocolo deseado, no dice cómo utilizarlo
- Puede haber mejores soluciones (negociación)
- Se puede contratar con este tipo de requisito? Sí, aunque no se garantiza que sea el protocolo más seguro.
- Se puede verificar el requisito? Sí.

# Ingeniería de Requisitos. Definiciones

- **(IEEE)** La IDR es la rama de la Ingeniería del Software relacionada con los *objetivos, servicios y restricciones* de los sistemas software.
- **(Davis)** La IDR es el conjunto de actividades que incluyen la búsqueda o aprendizaje sobre el problema que necesita una solución y la especificación del comportamiento externo de un sistema que puede resolver dicho problema.

El producto final de la IDR es la especificación de requisitos software (ERS).

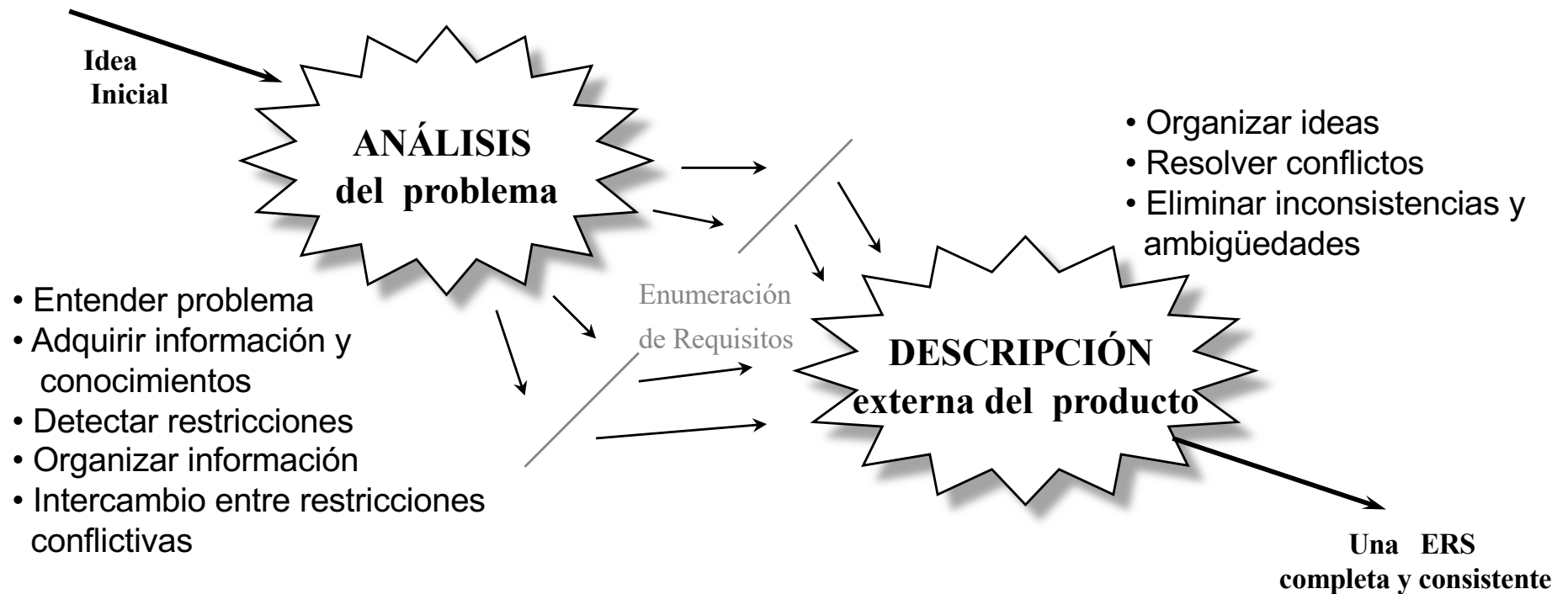
# Ingeniería de Requisitos. Definición

[Steve Easterbrook 2004]

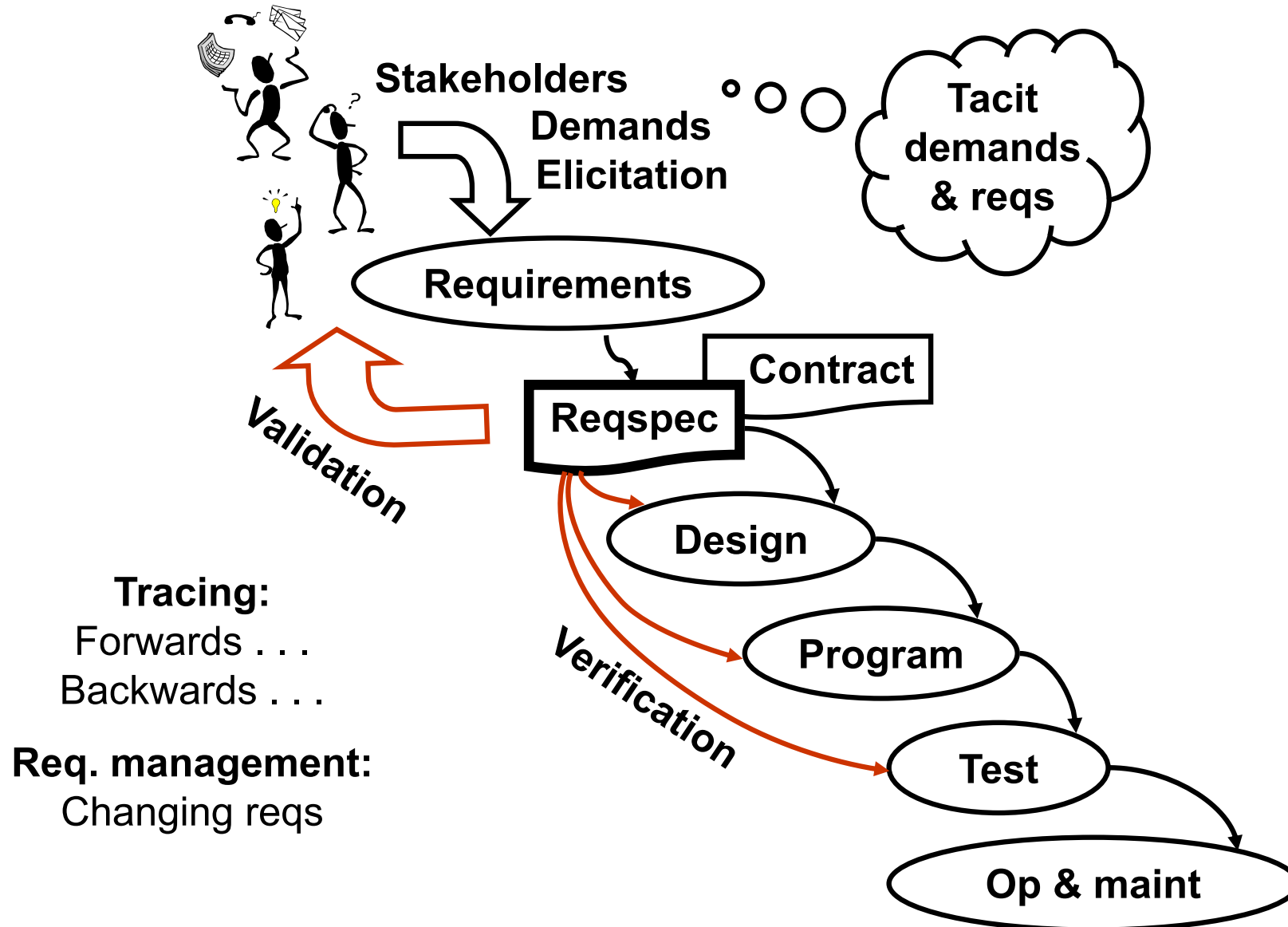


# Ingeniería de Requisitos. Actividades

- **Principales actividades que conlleva** (Davis93)



# Ingeniería de Requisitos. Visión general



From: Soren Lauesen: Software Requirements © Pearson / Addison-Wesley 2002



# Ingeniería de Requisitos. Preguntas frecuentes

- ¿Cuánto puede costar la IDR en un proyecto de desarrollo de software?
  - Aproximadamente un 15% del desarrollo.
- ¿Qué es un proceso de IDR?
  - Conjunto estructurado de actividades para el desarrollo de los requisitos del sistema.
- ¿Qué ocurre cuando hay errores en los requisitos?
  - retrasos, errores, fallos, sobrecostos, no se cumplen las expectativas,...
- ¿Existe un proceso de IDR “ideal”?
  - No. Los procesos se deben ajustar a las necesidades de cada organización.
- ¿Qué es son los “stakeholders”?
  - Cualquiera “afectado” por el sistema (en su concepción, desarrollo u operación).
- ¿Cuál es la relación entre el proceso de requisitos y el diseño?
  - Son actividades entrelazadas. Deberían “idealmente” ser actividades separadas pero en la práctica es imposible.
- ¿Son sinónimos “validar” y “verificar” requisitos?
  - No. La validación se realiza con el usuario y la verificación con los artefactos de diseño/programación.