Calidade e probas en robótica

2022/23

Ingeniería de requisitos

Marcos Boullón Magán

Depto. Electrónica e Computación

Modelos de proceso

Hay muchos procesos propuestos para desarrollar sistemas software, pero todos incluyen las mismas **actividades fundamentales**:

- especificación de funcionalidades
- desarrollo
  - diseño del sistema y de componentes
  - implementación
- validación
- mantenimiento y evolución

Modelos de proceso (2)

Abstracciones de un proceso general, pensados para extenderse y adaptarse por cada compañía; no son excluyentes, y se pueden combinar:

- **modelo en cascada;** las cuatro actividades fundamentales son fases separadas e independientes en el proceso de desarrollo.
- **modelo incremental;** el producto se construye como una sucesión de productos que van añadiendo funcionalidades nuevas sobre las versiones (*incrementos*) anteriores.
- modelo orientado a la reutilización
- modelo en espiral, orientado al riesgo
- proceso unificado racional (RUP)

Modelos de proceso (3)

Grandes categorías de procesos de desarrollo:

- **procesos dirigidos por un plan;** todas las actividades se planean por anticipado y el avance se mide contra dicho plan:

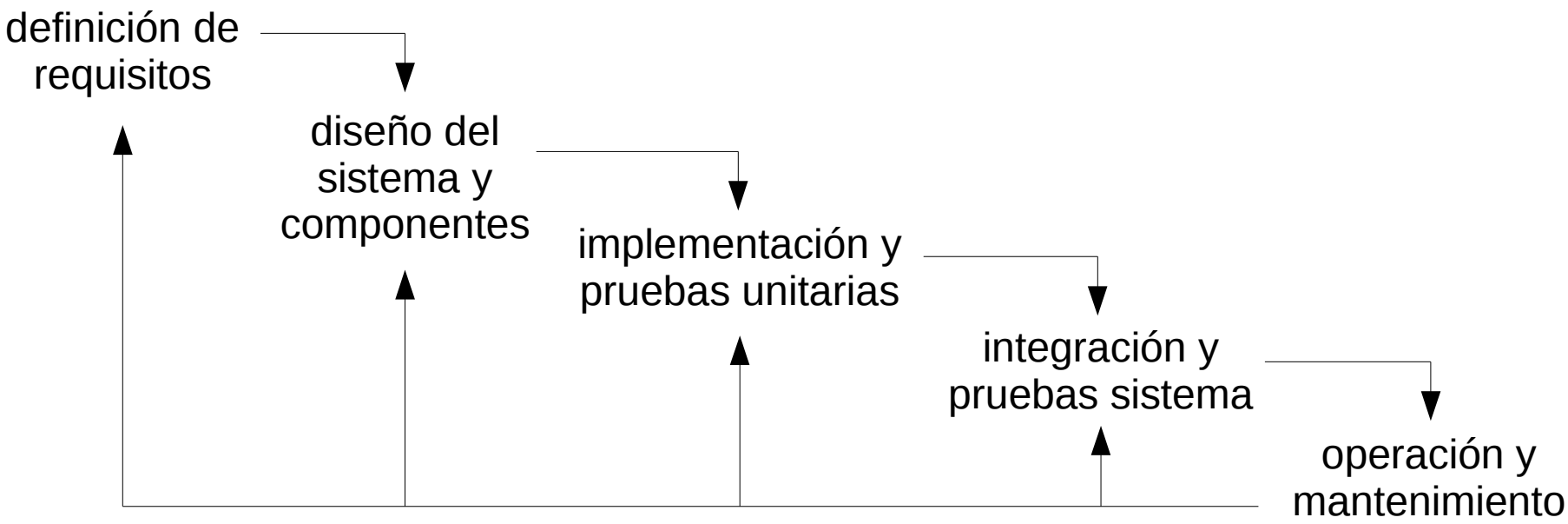
  actividades bien definidas, requisitos bien identificados, entregas planificadas...

- **procesos ágiles;** planificación incremental para permitir gestionar requisitos de usuario o de sistema cambiantes en el tiempo:

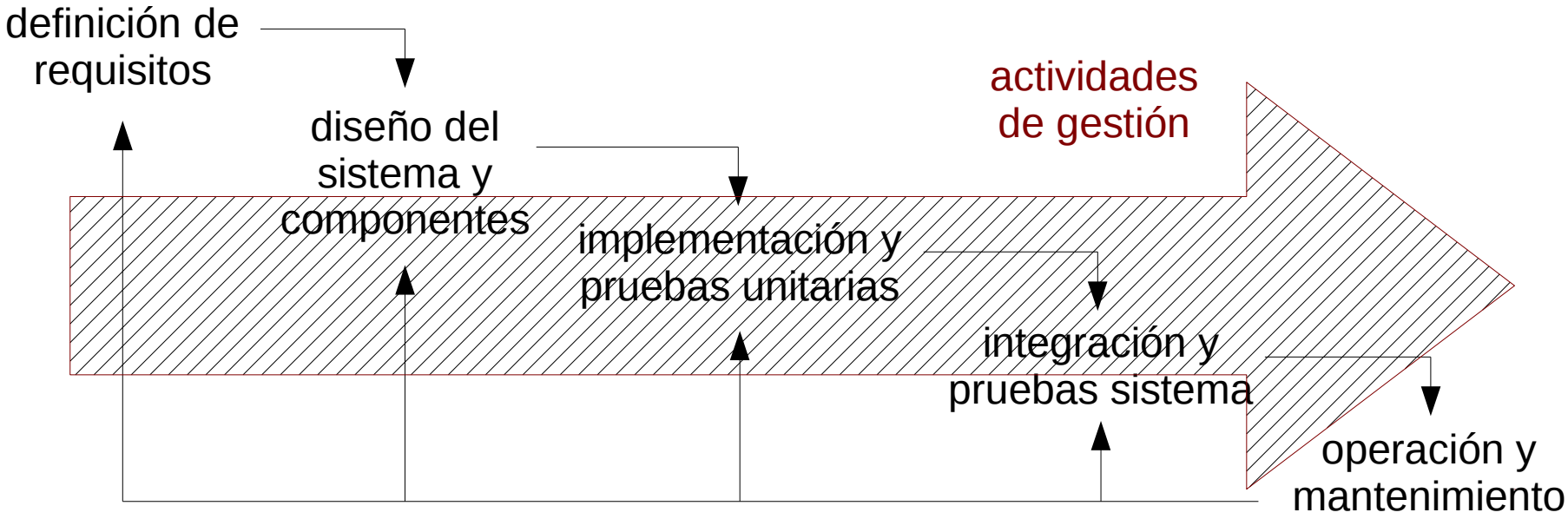
  continua interacción con el cliente, entregas parciales, ciclos iterativos cortos...

Cada enfoque es más adecuado para un cierto tipo de sistema software, pero no son excluyentes, y se pueden combinar.

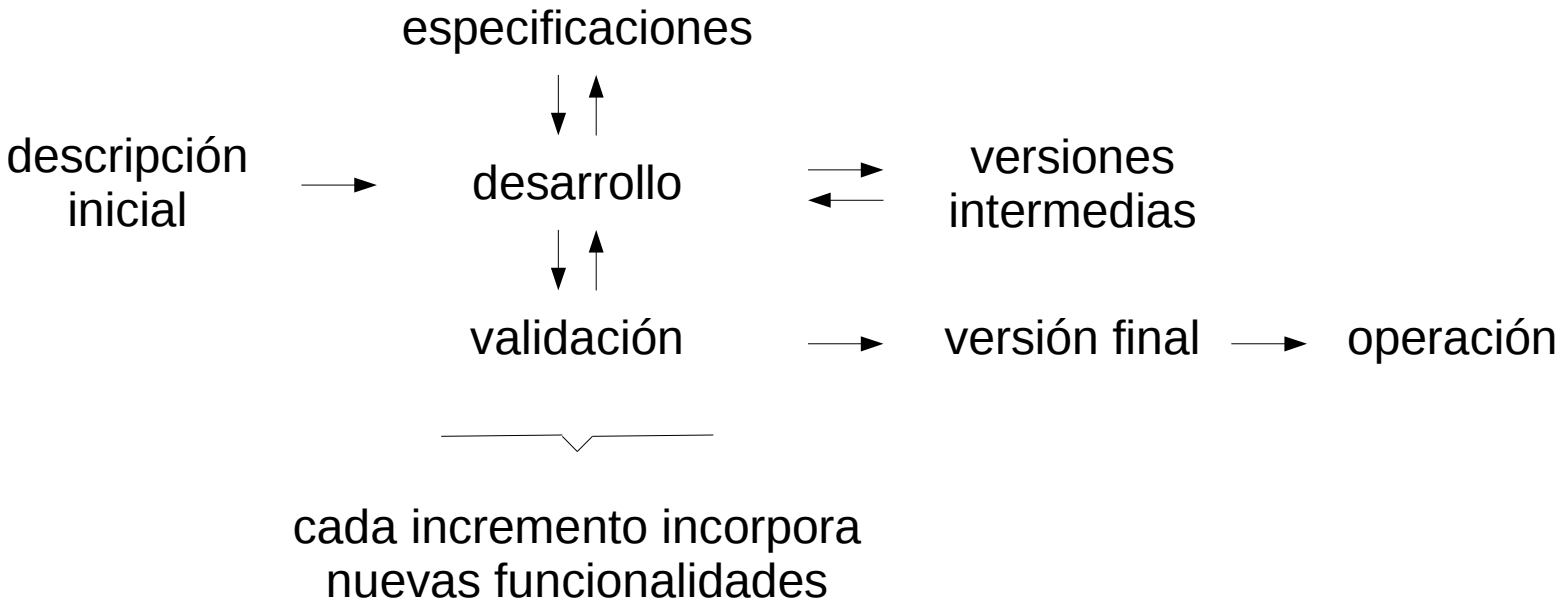
Modelo en cascada



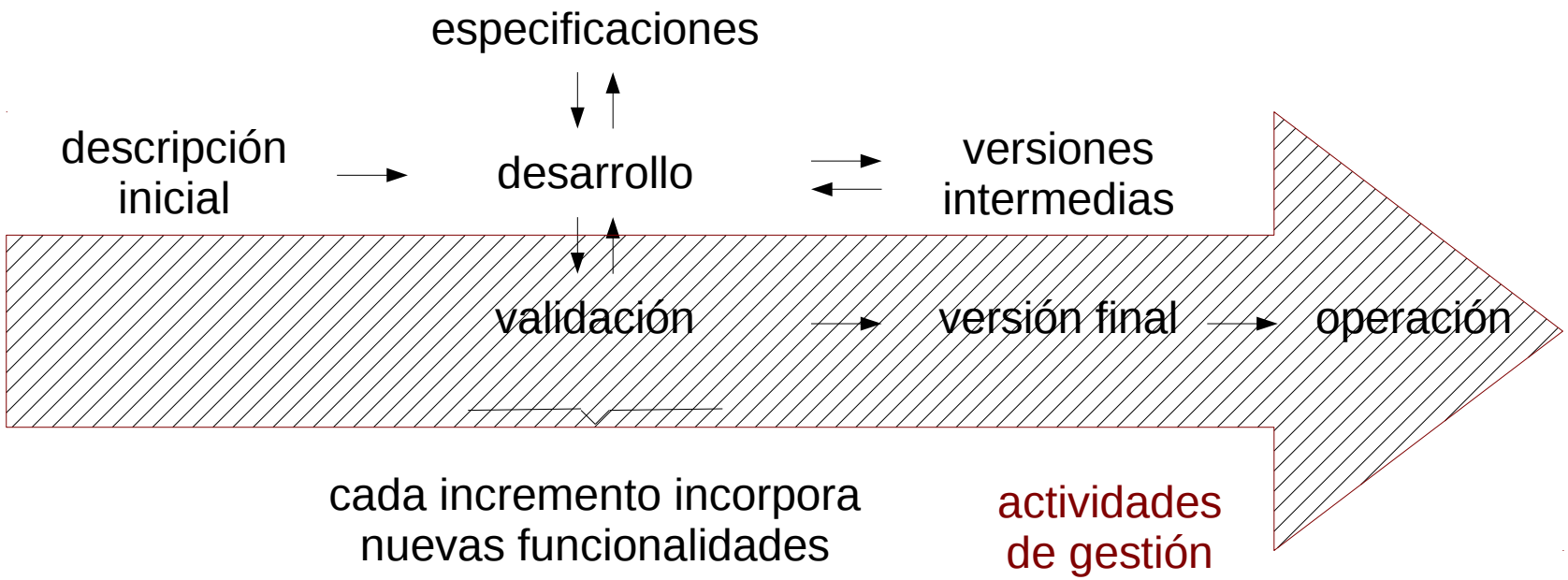
Modelo en cascada (2)



Modelo incremental



Modelo incremental (2)



Especificación de requisitos

Objetivo: Comprender y definir qué servicios se requieren del sistema, identificar sus capacidades y restricciones.

Genera un **documento de requisitos** con dos niveles de detalles: nivel básico para usuarios y clientes (DRU, documento de requisitos de usuario), nivel detallado para desarrolladores (ERS, especificación de requisitos software).

Especificación de requisitos (2)

En esta fase de análisis de requisitos se **ANALIZA** el problema (con clientes, usuarios, proveedores e ingenieros; entrevistas, análisis de contexto. Simulaciones, prototipos) y se **DEFINE** el producto (una descripción detallada y completa del sistema final).

Principios:

- comprender el problema y su entorno
- funcionalidades y requisitos se determinan con una aproximación descendente
- la especificación de requisitos debe ser ampliable
- sin influencia de factores técnicos: OS, lenguaje, técnica
- para cada subsistema, identificar requisitos funcionales
- para el sistema global, identificar requisitos no funcionales

  documentación, roles, fiabilidad...

Especificación de requisitos (3)

Subactividades:

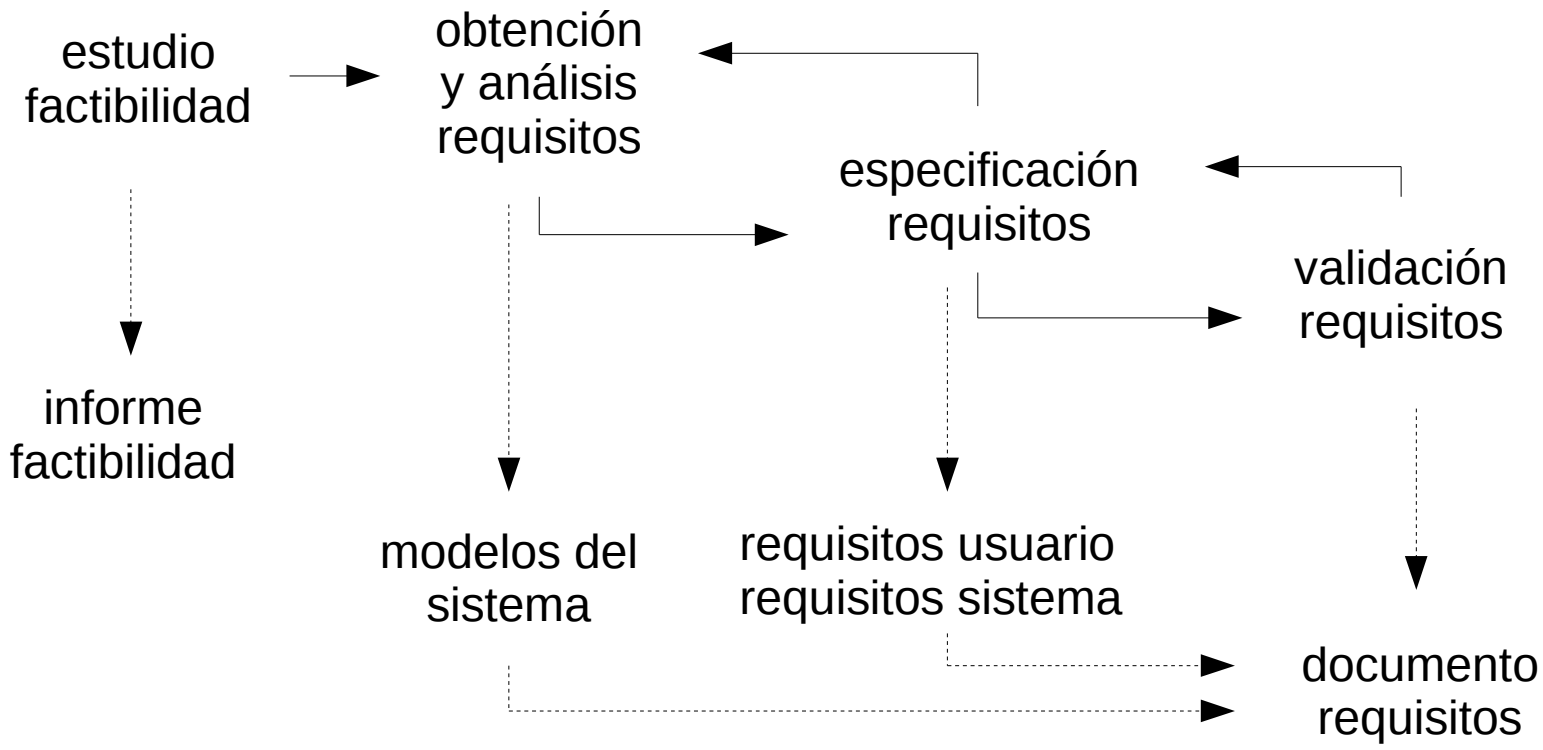
- estudio de factibilidad; rápido y barato; responde a ¿las necesidades del usuario se pueden cumplir? ¿se pueden desarrollar dentro de las restricciones? ¿tiene una buena relación coste/beneficio? ¿seguimos?
- obtención y análisis de requisitos; derivar los requisitos del sistema mediante observación de los sistemas actuales, discusión con los usuarios y proveedores, análisis de la lógica del negocio, usando modelos del sistema y prototipos…

  combinar, dividir, determinar viabilidad, priorizar… requisitos
- especificación de requisitos; transcribir la información anterior a un documento; ¿usar diagramas? dos tipos:
  - requisitos de usuario, descripción para usuario/cliente
  - requisitos de sistema: descripción para desarrolladores
- validación; comprobar que los requisitos sean realistas, coherentes y completos, sin ambigüedad y consistentes

Especificación de requisitos (4)

No es necesario seguir estrictamente esa secuencia de subactividades.

Durante todo el proceso pueden aparecer nuevos requisitos. Por supuesto, en metodologías ágiles los requisitos se desarrollan de forma incremental.



Requisitos

Los requisitos del sistema software son descripciones de lo que el sistema **debe hacer**: servicios y funcionalidades que ofrece, y sus limitaciones. Reflejan las necesidades del cliente.

La ingeniería de requisitos es el proceso de descubrir, analizar, documentar y verificar estas funcionalidades y restricciones.

Requisitos (2)

La forma de los requisitos van desde un **enunciado abstracto** de alto nivel sobre una funcionalidad/característica que el sistema tendrá que proporcionar... hasta una descripción detallada en un **lenguaje formal** de la misma.

¡Todas ellas válidas! Diferentes descripciones (diferentes niveles de detalle) de la misma funcionalidad sirven a diferentes roles de usuarios del sistema.

Clasificación de requisitos

En un primer momento identificamos:

- **requisitos de usuario;** enunciados en un lenguaje natural (diagramas) qué servicios esperan los usuarios y bajo qué condiciones va a operar

  gerentes del cliente, usuarios finales del sistema, ingenieros del cliente, gerentes de los contratistas

- **requisitos del sistema;** descripciones detalladas de las funcionalidades y restricciones operaciones del sistema, con la mayor exactitud posible

  usuarios finales del sistema, ingenieros del cliente, arquitectos del sistema, desarrolladores software

Clasificación de requisitos (2)

Pero la clasificación que más útil nos resulta es:

- **requisitos funcionales;** los servicios, funcionalidades y características que el sistema debe proporcionar para cumplir su cometido; definen las entradas y salidas; especifican la respuesta del sistema a situaciones especiales (entrada de datos incorrecta, excepciones...); más los servicios que no debe proporcionar
- **requisitos no funcionales;** limitaciones sobre los servicios y funcionalidades que proporciona; están impuestas por estándares, el proceso de desarrollo o el entorno de uso, entre otras; hacen referencia a atributos y propiedades del sistema como un todo, no a servicios independientes

En realidad, la distinción no es tan obvia (componentes del sistema frente a propiedades globales). Además, los requisitos no funcionales pueden estar enmascarando otros requisitos funcionales. Y es que los requisitos no son completamente independientes.

Requisitos funcionales

Establecen **qué debe hacer el sistema** (expresados en un lenguaje de alto nivel, *requisitos de usuario*, o o un bajo nivel de abstracción, *requisitos de sistema*). Por supuesto, dependen del tipo de sistema software que se está desarrollando.

Desde requisitos muy generales para expresar las funcionalidades básicas hasta otros muy específicos que reflejan particularidades de la organización (sistema, datos, flujo de trabajo...). Pero siempre deben ser completos (indicando todos los servicios necesarios), exactos y consistentes (no contradictorios)... porque vamos a buscar la implementación más sencilla posible.

> *no siempre será fácil cumplir con estas características; en sistemas grandes y complejos existirán un gran número de funcionalidades para servir a múltiples participantes, con necesidades diferentes y contradictorias; situación que se detecta en un análisis posterior*

Requisitos no funcionales

Aquellos que no tienen que ver con servicios específicos sino con propiedades (emergentes) del sistema:

fiabilidad, seguridad, rendimiento...

o con restricciones sobre los componentes del sistema:

estándares, definición de interfaces de datos...

Típicamente restringen o especifican características del sistema como un todo.

La incapacidad de cubrir los requisitos no funcionales **PUEDE** impedir el uso del sistema. Corregirlos puede ser un problema, ya que no es sencillo relacionar componentes concretos con requisitos no funcionales (sí para los requisitos funcionales) sino que su implementación se propaga por todo el sistema.

Requisitos no funcionales (2)

Los requisitos no funcionales surgen de las necesidades del usuario debidas a características requeridas por el software (*requisitos de producto*), la organización que desarrolla y usa el software (*requisitos de la organización*) o fuentes externas (*requisitos externos*).

Requisitos de producto. Especifican o restringen el comportamiento del sistema para cubrir las necesidades del usuario final.

Requisitos de la organización. Derivan de políticas y procedimientos en la organización del desarrollador o del cliente. Incluyen restricciones presupuestarias, políticas organizativas y la necesidad de la interoperabilidad con otros sistemas software o hardware.

Requisitos externos. Se refieren a factores externos al sistema y al proceso de desarrollo, como regulaciones del sector, legislaciones específicas o consideraciones éticas.

Requisitos no funcionales (3)

Requisitos de producto

- usabilidad
- eficiencia
  - rendimiento
  - espacio
- confiabilidad
- seguridad

Requisitos de la organización

- ambientales
- operacionales
- desarrollo

Requisitos externos

- regulatorios
- éticos
- legales
  - contables
  - protección/seguridad

Requisitos no funcionales (4)

Es posible confundir requisitos no funcionales con metas generales (que describen buenas intenciones y objetivos deseables):

facilidad de uso, minimizando errores, interfaz simple...

Para identificar correctamente requisitos no funcionales se recomienda escribirlos de manera cuantitativa para ponerlos objetivamente a prueba; ¡aunque el coste de verificarlos puede ser excesivo!

- *rapidez;* medida en transacciones por segundo, tiempo de respuesta por evento, tiempo de construcción del documento...
- *tamaño;* bytes, número de servidores...
- *facilidad de uso;* tiempo de capacitación, número de menús de ayuda...
- *fiabilidad;* probabilidad de fallo, tiempo medio entre fallos, porcentaje de disponibilidad...
- *robustez;* porcentaje de eventos que causan un fallo, número reinicios...
- ...

ERS

El ERS (especificación de requisitos software, SRS) es el documento que especifica qué va a implementar los desarrolladores. Incluye tanto los requisitos de usuario como los de sistema.

Será usado por múltiples tipos de usuario. Se escribe entonces como un compromiso: suficiente detalle para desarrolladores y mantenimiento, pero sin agobiar a los usuarios que buscan validar las funcionalidades allí indicadas.

> *clientes del sistema, administradores, ingenieros, ingenieros de pruebas, ingenieros de mantenimiento...*

Por supuesto, el nivel de detalle depende también del sistema concreto.

> *los sistemas críticos requieren establecer requisitos de seguridad y confiabilidad con alto nivel de detalle...*

ERS (2)

En procesos de desarrollo planificados, poco más hay que decir. Pero en procesos no planificados (ágiles) el ERS se vuelve rápidamente obsoleto según se añaden nuevos requisitos para las sucesivas versiones.

En estos procesos se prefieren recopilar también de manera incremental y gestionarlos como *historias de usuario*. En cada incremento, el cliente dará prioridad a algunas de estas historias.

ERS (3)

Contenido básico

- introducción
- documentación general
- descripción de la información
- descripción funcional
- requisitos específicos
- descripción del comportamiento
  - diagramas
- criterios de validación
- conclusiones

Especificación de requisitos

La especificación de requisitos es el proceso de escribir en el ERS los requisitos del usuario y del sistema de forma clara, sin ambigüedades, fáciles de entender, completos y consistentes.

Los **requisitos del usuario** deben describir los requisitos funcionales y no funcionales, de forma que sean comprensibles para los usuarios del sistema que no cuentan con un conocimiento técnico detallado.

Los **requisitos del sistema** son versiones extendidas de los primeros que los ingenieros de software usan como punto de partida para el diseño del sistema. Idealmente, deberían describir de manera simple el comportamiento externo del sistema y sus restricciones operacionales, sin ocuparse de cómo se diseña o implementa. Sin embargo, al nivel de detalle requerido para especificar por completo un sistema de software complejo, es prácticamente imposible excluir toda la información de diseño.

Especificación de requisitos (2)

Preferiremos usar lenguaje natural, con tablas y formas sencillas, así como diagramas intuitivos en el documento de requerimientos. Los requisitos del sistema se escriben también en lenguaje natural, pero de igual modo se utilizan otras notaciones basadas en formas, modelos gráficos del sistema o modelos matemáticos del sistema.

Alternativas:

Enunciados en lenguaje natural. Los requisitos se escriben al usar enunciados numerados en lenguaje natural. Cada enunciado debe expresar un requisito.

Lenguaje natural estructurado. Los requisitos se escriben en lenguaje natural en una forma o plantilla estándar. Cada campo ofrece información de un aspecto del requisito.

Especificación de requisitos (3)

Lenguajes de descripción de diseño. Este enfoque usa un lenguaje como un lenguaje de programación, pero con características más abstractas para especificar los requisitos al definir un modelo operacional del sistema. Poco usado, excepto para especificaciones de interfaz.

Anotaciones gráficas. Los modelos gráficos, complementados con anotaciones de texto, sirven para definir los requerimientos funcionales del sistema; los casos de uso del UML y los diagramas de secuencia se emplean de forma común.

Especificaciones matemáticas. Dichas anotaciones se basan en conceptos matemáticos como máquinas o conjuntos de estado finito. Aunque tales especificaciones sin ambigüedades pueden reducir la imprecisión en un documento de requerimientos, la mayoría de los clientes no comprenden una especificación formal y no pueden comprobar que representa lo que quieren.

Especificación de requisitos: ejemplo

Bomba de insulina/Software de control/SRS/3.3.2

3.2 Si se requiere, cada 10 minutos el sistema medirá el azúcar en la sangre y administrará insulina. (Los cambios de azúcar en la sangre son relativamente lentos, de manera que no son necesarias mediciones más frecuentes; la medición menos periódica podría conducir a niveles de azúcar innecesariamente elevados.)

3.6 Cada minuto, el sistema debe correr una rutina de autoevaluación, con las condiciones a probar y las acciones asociadas definidas en la tabla 1. (Una rutina de autoevaluación puede detectar problemas de hardware y software, y prevenir al usuario sobre el hecho de que la operación normal puede ser imposible.)

Especificación de requisitos: ejemplo (2)

Bomba de insulina/Software de control/SRS/3.3.2

función	Calcula dosis de insulina: nivel seguro de azúcar.
descripción	Calcula la dosis de insulina que se va a suministrar cuando la medición del nivel de azúcar actual esté en zona segura entre 3 y 7 unidades.
entradas	Lectura del azúcar actual (r2), las dos lecturas previas (r0 y r1).
fuente	Lectura del azúcar actual del sensor. Otras lecturas de la memoria.
salidas	CompDose: la dosis de insulina a administrar.
destino	Ciclo de control principal.
acción	CompDose es cero si es estable el nivel de azúcar, o cae o si aumenta el nivel pero disminuye la tasa de aumento. Si el nivel se eleva y la tasa de aumento crece, CompDose se calcula entonces al dividir la diferencia entre el nivel de azúcar actual y el nivel previo entre 4 y redondear el resultado. Si la suma se redondea a cero, en tal caso CompDose se establece en la dosis mínima que puede entregarse.
requisitos	Dos lecturas previas, de modo que puede calcularse la tasa de cambio del nivel de azúcar.
precondición	El depósito de insulina contiene al menos la dosis individual de insulina máxima permitida.
postcondición	r0 se sustituye con r1, luego r1 se sustituye con r2.
efectos colaterales	Ninguno.

Especificación de requisitos: ejemplo (3)

Bomba de insulina/Software de control/SRS/3.3.2

Condición	Acción
Nivel de azúcar en descenso (r2 < r1)	CompDose = 0
Nivel de azúcar estable (r2 = r1)	CompDose = 0
Nivel de azúcar creciente y tasa de incremento decreciente ((r2 - r1) < (r1 - r0))	CompDose = 0
Nivel de azúcar creciente y tasa de incremento estable o creciente ((r2 - r1) >= (r1 - r0))	CompDose = round ((r2 - r1)/4) CompDose = MinimumDose SI resultado redondeado = 0

Para escribir requisitos sin ambigüedades, en particular al especificar cálculos complejos, se puede agregar información extra a los requisitos con el uso de tablas o modelos gráficos del sistema.

Actividades del proceso

El proceso de ingeniería de requisitos incluye cuatro actividades de alto nivel:

- valorar si el sistema es útil para la empresa (**estudio de factibilidad**)
- descubrir requisitos (**adquisición y análisis**)
- convertir dichos requisitos en alguna forma estándar (**especificación**)
- comprobar que los requisitos definan realmente el sistema que quiere el cliente (**validación**).

Actividades del proceso (2)

El proceso no es secuencial, sino un proceso iterativo donde las actividades están entrelazadas. Al inicio del proceso, se empleará más esfuerzo para comprender los requisitos empresariales de alto nivel y los no funcionales, así como los requisitos del usuario para el sistema. Más adelante, se le dedicará más esfuerzo a la adquisición y comprensión de los requisitos detallados del sistema.

> *prácticamente en todos los sistemas cambian los requisitos; las personas implicadas desarrollan una mejor comprensión de qué quieren que haga el software; la organización que compra el sistema cambia; se hacen modificaciones al hardware, al software y al entorno organizacional del sistema; al proceso de administrar tales requisitos cambiantes se le llama administración de requisitos*

Actividad: estudio de factibilidad

Estudio inicial de factibilidad.

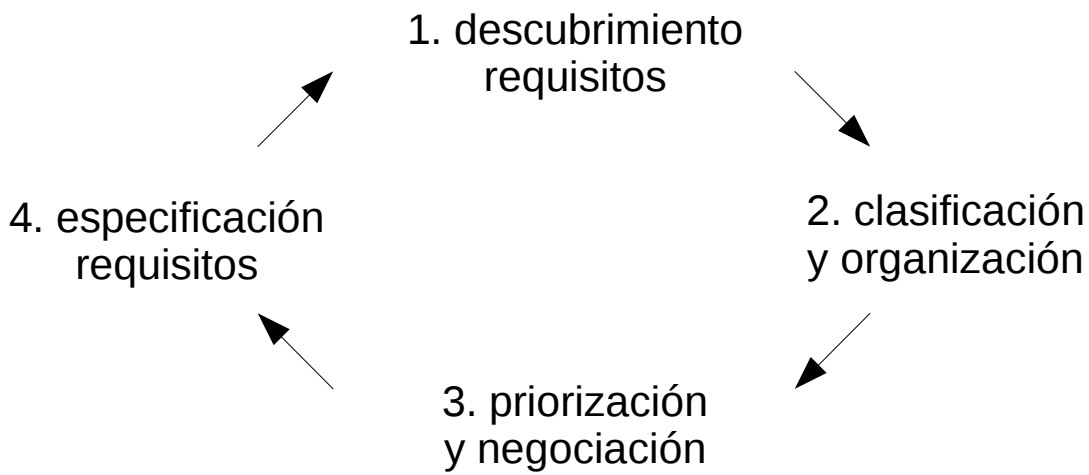
Debe ser rápido y barato. Responde a ¿las necesidades del usuario se pueden cumplir? ¿se pueden desarrollar dentro de las restricciones? ¿tiene una buena relación coste/beneficio? ¿seguimos adelante?

Actividad: adquisición y análisis de requisitos

Los ingenieros de software trabajan con clientes y usuarios finales del sistema para descubrir el dominio de aplicación, qué servicios debe proporcionar el sistema, el rendimiento necesario, las restricciones de hardware... y así derivar los requisitos mediante observación de los sistemas actuales, la discusión con los usuarios y proveedores, el análisis de la lógica del negocio, o usando modelos del sistema y prototipos.

Actividad: adquisición y análisis (2)

Un ejemplo del proceso de adquisición y análisis de requisitos. Es un proceso iterativo, que concluye cuando está completo el documento de requisitos.



Actividad: adquisición y análisis (3)

Descubrimiento de requisitos. Se interactúa con los participantes del sistema para descubrir necesidades. También los requisitos de dominio de los participantes y la documentación se descubren durante esta fase.

Clasificación y organización de requisitos. En la compilación no estructurada de requisitos, se agrupan los requisitos relacionados y se organizan en grupos coherentes. La forma más común de agruparlos es apoyándose en la arquitectura del sistema, identificando subsistemas para asociarles los requisitos a cada uno (requisitos y diseño no pueden separarse completamente).

Priorización y negociación de requisitos. Inevitablemente, cuando intervienen diversos participantes, los requisitos entrarán en conflicto. Es necesario priorizar los requisitos, así como por encontrar y resolver conflictos de requisitos mediante la negociación. Por lo general, los participantes tienen que reunirse para resolver las diferencias y ponerse de acuerdo.

Especificación de requisitos. Los requisitos así obtenidos se documentan de una manera más o menos formal.

Técnicas: entrevistas

Las entrevistas formales o informales con participantes del sistema son una parte de la mayoría de los procesos de ingeniería de requisitos.

En estas entrevistas, el equipo formula preguntas a los participantes sobre el sistema que actualmente usan y el sistema que se va a desarrollar. Los requisitos se derivan de las respuestas a dichas preguntas.

Posiblilidades:

- entrevistas cerradas, donde los participantes responden a un conjunto de preguntas preestablecidas
- entrevistas abiertas, en las cuales no hay agenda predefinida; el equipo explora un rango de conflictos con los participantes del sistema y, como resultado, desarrolla una mejor comprensión de sus necesidades

Técnicas: escenarios

Los escenarios son particularmente útiles para detallar un bosquejo de descripción de requerimientos. Se trata de ejemplos de sesiones interactivas.

Cada escenario abarca comúnmente una interacción o un número pequeño de interacciones posibles. Se desarrollan diferentes formas de escenarios y se ofrecen varios tipos de información con diversos niveles de detalle acerca del sistema. La adquisición basada en escenario implica trabajar con los participantes para identificar escenarios y captar detalles a incluir en dichos escenarios.

Técnicas: escenarios (2)

Un escenario utiliza:

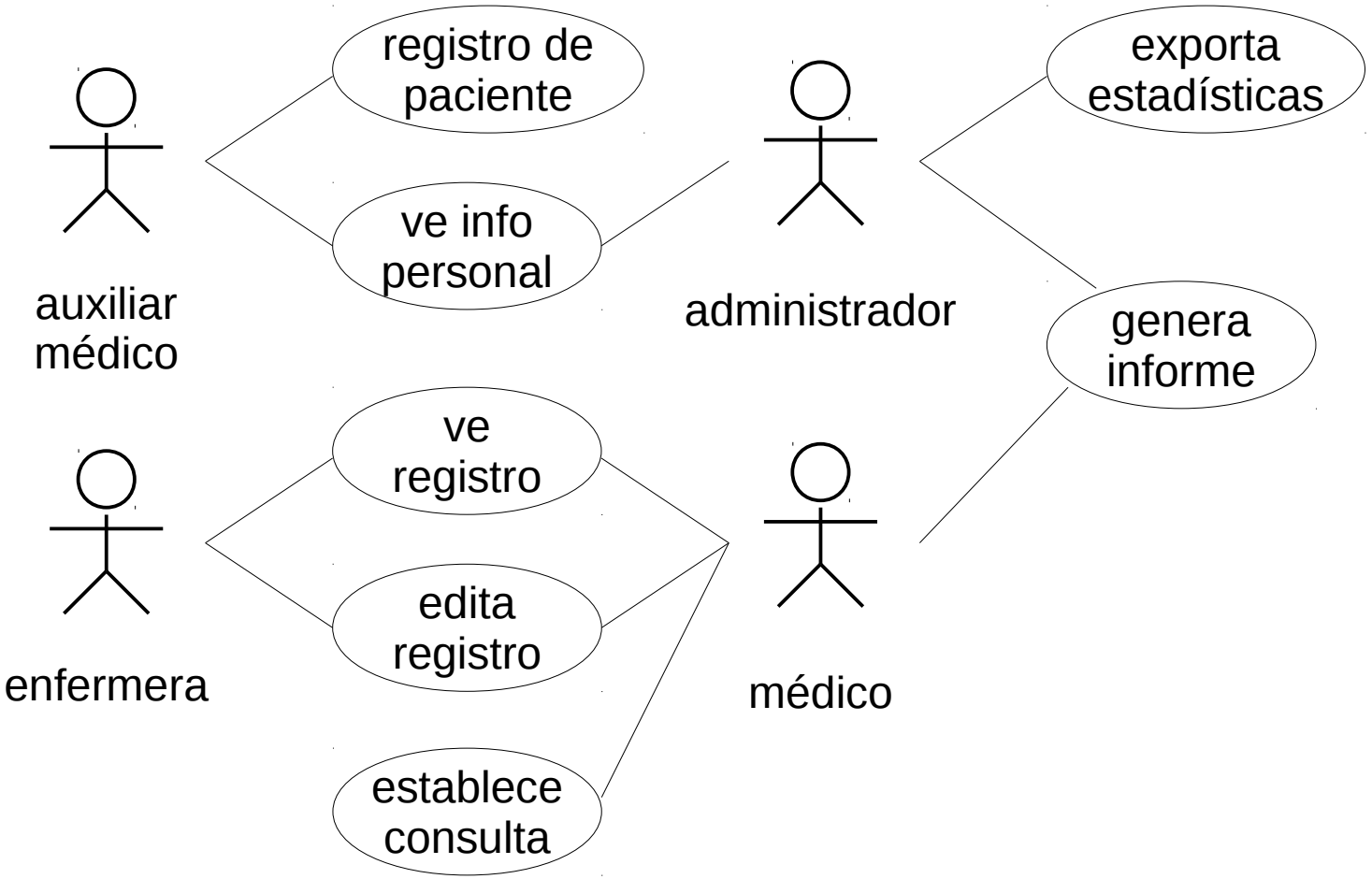
1. Suposición inicial; una descripción de qué esperan el sistema y los usuarios cuando inicia el escenario.
2. Caso normal; una descripción en el escenario del flujo normal de los eventos.
3. Casos anómalos; una descripción de qué puede salir mal y cómo se manejaría.
4. Información de otras actividades que estén en marcha al mismo tiempo, por otros usuarios.
5. Una descripción del estado del sistema cuando termina el escenario.

Técnicas: casos de uso

En su forma más sencilla, un caso de uso identifica a los actores implicados en una interacción y los tipos de interacción.

Los casos de uso se documentan con el empleo de un diagrama de caso de uso de alto nivel. El conjunto de casos de uso representa todas las interacciones posibles que se describirán en los requerimientos del sistema. Los actores en el proceso, que pueden ser individuos u otros sistemas, se representan como figuras sencillas. Cada clase de interacción se constituye como una elipse con etiqueta. Líneas vinculan a los actores con la interacción. De manera opcional, se agregan puntas de flecha a las líneas para mostrar cómo se inicia la interacción.

Técnicas: casos de uso (2)



Técnicas: etnografía

La etnografía es una técnica de observación que se usa para entender los procesos operacionales y ayudar a derivar requerimientos de apoyo para dichos procesos. Un analista se adentra en el ambiente laboral donde se usará el sistema. Observa el trabajo diario y toma notas acerca de las tareas existentes en que intervienen los participantes.

El valor de la etnografía es que ayuda a descubrir requerimientos implícitos del sistema que reflejan las formas actuales en que trabaja la gente, en vez de los procesos formales definidos por la organización.

Técnicas: prototipos

Modelos del sistema y prototipos, para comprobar casos límite de la lógica de negocio.

Cuado hay dudas sobre algunos requisitos software, se pueden construir subsistemas con funcionalidad completa; se van refinando iterativamente e implementando nuevas funcionalidades… a veces hasta convertirse en parte del producto.

Actividad: especificación de requisitos

Transcribir la información recogida a un documento (ERS).

Actividad: validación de requisitos

Comprobar que los requisitos recogidos sean realistas, coherentes, expresados sin ambigüedad... y que además definan el sistema que el usuario quiere.

Es un análisis que se hace sobre el documento ERS, antes de tener un sistema en funcionamiento, así que es necesariamente insuficiente y exigirá revisitar esta fase de especificación y análisis más adelante según vayan surgiendo inconsistencias o nuevos requisitos.

Actividad: validación de requisitos (2)

Distintos tipos de comprobación:

- validez; el sistema tiene que funcionar para lo que el usuario necesita; múltiples usuarios tienen necesidades diferentes; se deben cubrir
- totalidad; se recogen todas las funciones y restricciones que el usuario necesita
- simplicidad; los requisitos no están duplicados, ni las mismas funcionalidades se especifican de múltiples formas
- consistencia; los requisitos no entran en conflicto
- completitud; el sistema no tiene estados indefinidos
- realismo; las funcionalidades pretendidas deben ser implementadas: tecnología disponible, experiencia, recursos, presupuesto, tiempo...
- verificabilidad; los requisistos deberían escribirse de manera que fueran verificables (expresados como un conjunto de pruebas ejecutables)

Actividad: validación de requisitos (3)

Técnicas habituales:

Revisiones de requerimientos. Un equipo de revisores verifica el listado buscando problemas.

Prototipos. Se diseña un modelo ejecutable de manera que el usuario pueda experimentar y comprobar que se cubren sus necesidades.

Casos de prueba. El propio diseño de los casos de prueba puede revelar problemas con requisitos difíciles de implementar (si los casos de prueba son difíciles de definir, el requisito podría estar mal expresado), que pueden ser reconsiderados.

Gestión de requisitos

Los requisitos de grandes sistemas cambian en el tiempo:

- usamos requisitos incompletos por ser un problema complejo
- tenemos una mejor comprensión del problema según avanza el desarrollo
- aparecen incompatibilidades o problemas de interacción con otros sistemas que no fueron anticipados
- aparecen nuevos usuarios con nuevas necesidades
- nuevos flujos de trabajo
- se aplican nuevas restricciones organizativas y presupuestarias
- evolucionan las necesidades del negocio
- ...

Gestión de requisitos (2)

Necesitamos **controlar los cambios** de requisitos: seguir la pista de cada requisito individualmente y los vínculos entre requisitos dependientes (para poder evaluar los efectos de cada cambio):

- asignar un identificador único a cada requisito
- definir un proceso formal para hacer cambios sobre un requisito; un conjunto de actividades para valorar el efecto y coste del cambio
- políticas de seguimiento, para mantener esta información registrada y de los requisitos dependientes
- herramientas, para una gestión automatizada; se requiere procesar grandes cantidades de información:
  - herramientas para registrar los requisitos
  - herramientas para administrar el cambio
  - herramientas para apoyo al seguimiento e identificación de relaciones entre requisitos

Gestión de requisitos (3)

Etapas:

Análisis del problemas y especificación del cambio. El proceso comienza con la identificación de un problema en los requisitos o, en ocasiones, con una propuesta de cambio específica. El problema o la propuesta de cambio se analizan para comprobar que es válida.

Análisis del cambio y estimación del costo. El efecto del cambio propuesto se valora usando información de seguimiento y conocimiento general de los requisitos del sistema. El costo por realizar el cambio se estima en términos de modificaciones al documento de requisitos y, si es adecuado, al diseño y la implementación del sistema. Una vez completado este análisis, se toma una decisión acerca de si se procede con el cambio.

Implementación del cambio. Se modifican el documento de requisitos y, donde sea necesario, el diseño y la implementación actual del sistema.

Gestión de requisitos (4)

Recuerda que los *procesos ágiles se diseñaron específicamente* para enfrentar los requisitos que cambian durante el desarrollo.

En dichos procesos, cuando se propone un cambio de requisitos, éste no pasa por un proceso de administración del cambio formal. En su lugar, el usuario puede integrarlo directamente en la siguiente iteración (si no hay historias de usuario de mayor prioridad).