Tema 1: Introducción a la Ingeniería de Requisitos

En este capítulo se introduce la Ingeniería de Requisitos (IR): motivación, definiciones, clasificaciones, etc. con la intención de dar a conocer los términos y conceptos que constituyen la base de esta Ingeniería.

Contenido

1.1 Motivación
1.2 Definiciones
1.3 Clasificación
1.4 Requisitos funcionales
1.5 Requisitos de calidad
1.6 Restricciones
1.7 Ejercicio
1.8 Características requisitos
1.9 Desarrollo de req. VS Gestión de req.

1.1 Motivación

- Conectar dominio del problema con el dominio de la solución
- Garantizar la calidad del desarrollo realizándose en el marco de un proyecto software.

  ETAPAS

  - **Análisis (QUÉ)**
  - Diseño (CÓMO)
  - Implementación
  - Pruebas
  - Mantenimiento

1.1 Motivación

- Frederick Brooks (1987)

"La parte más difícil en la construcción de un sistema software es precisamente decidir **QUÉ** construir. Ninguna otra parte del proyecto es **tan difícil** como establecer de forma detallada los requisitos técnicos. Ninguna otra parte del trabajo realizado **afecta tanto** al resultado final, si se hace de forma incorrecta. En ninguna otra parte es **tan difícil** rectificar los errores cometidos".

1.1 Motivación

- Gran parte de la dificultad estriba en la complejidad de **la comunicación humana**.

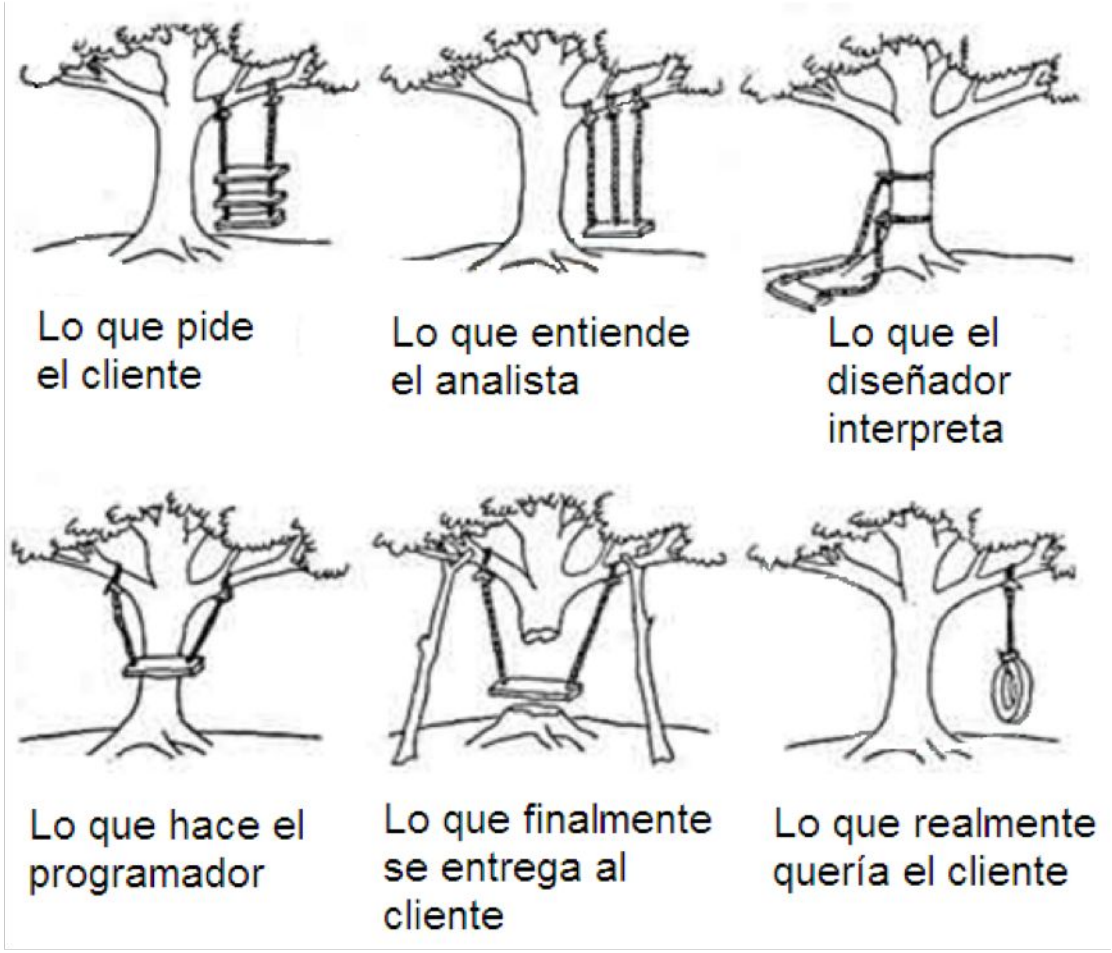
- Hay multitud de variables y parámetros que afectan al entendimiento entre dos partes. La comunicación **no viene dada sólo por el contenido verbal** sino que afectan otros muchos factores: contexto, entonación, comunicación no verbal, etc.

1.1 Motivación

- La ambigüedad en el lenguaje dificulta la buena comunicación:
    - **Polisemia**: una misma palabra con varios significados.
    - **Sinonimia**: varias palabras con el mismo significado.
- Debemos evitar:
    - Palabras sinónimas para un mismo concepto.
    - Expresiones polisémicas que continúan siendo ambiguas en un contexto definido. Ejemplo:

      "Ayer vi a Juan paseando"
- Recomendación: realizar un **Glosario**

1.1 Motivación



ENTRE LO QUE PIENSO,

LO QUE QUIERO DECIR,

LO QUE CREO DECIR,

LO QUE DIGO,

LO QUE QUIERES OIR,

LO QUE OYES,

LO QUE CREES ENTENDER,

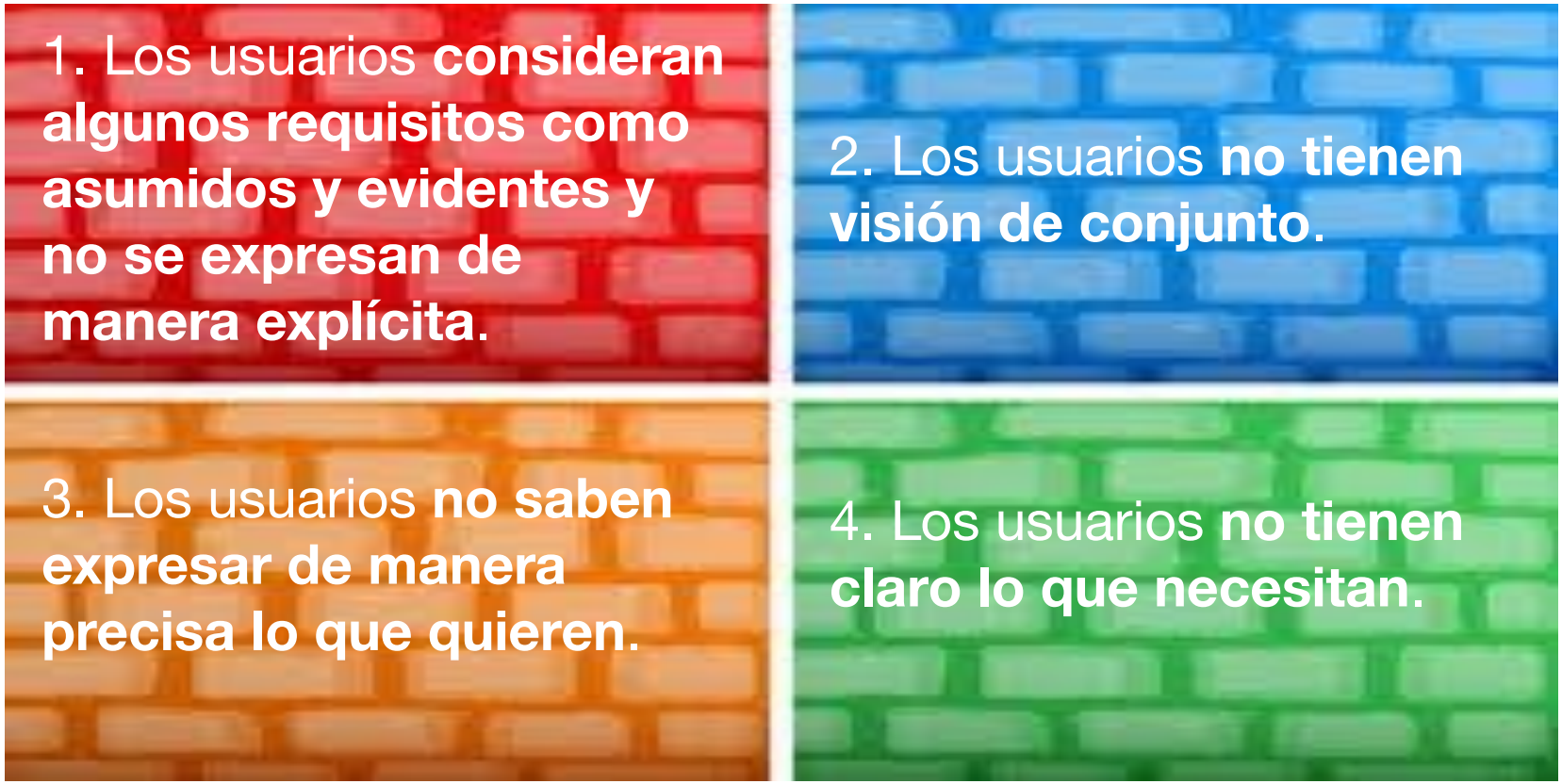
LO QUE QUIERES ENTENDER

Y LO QUE ENTIENDES

Existen 9 posibilidades
de no entenderse.

1.1 Motivación

- La adquisición de requisitos para construir una solución software **no es una tarea trivial**.
- Enfrentamos 4 obstáculos.

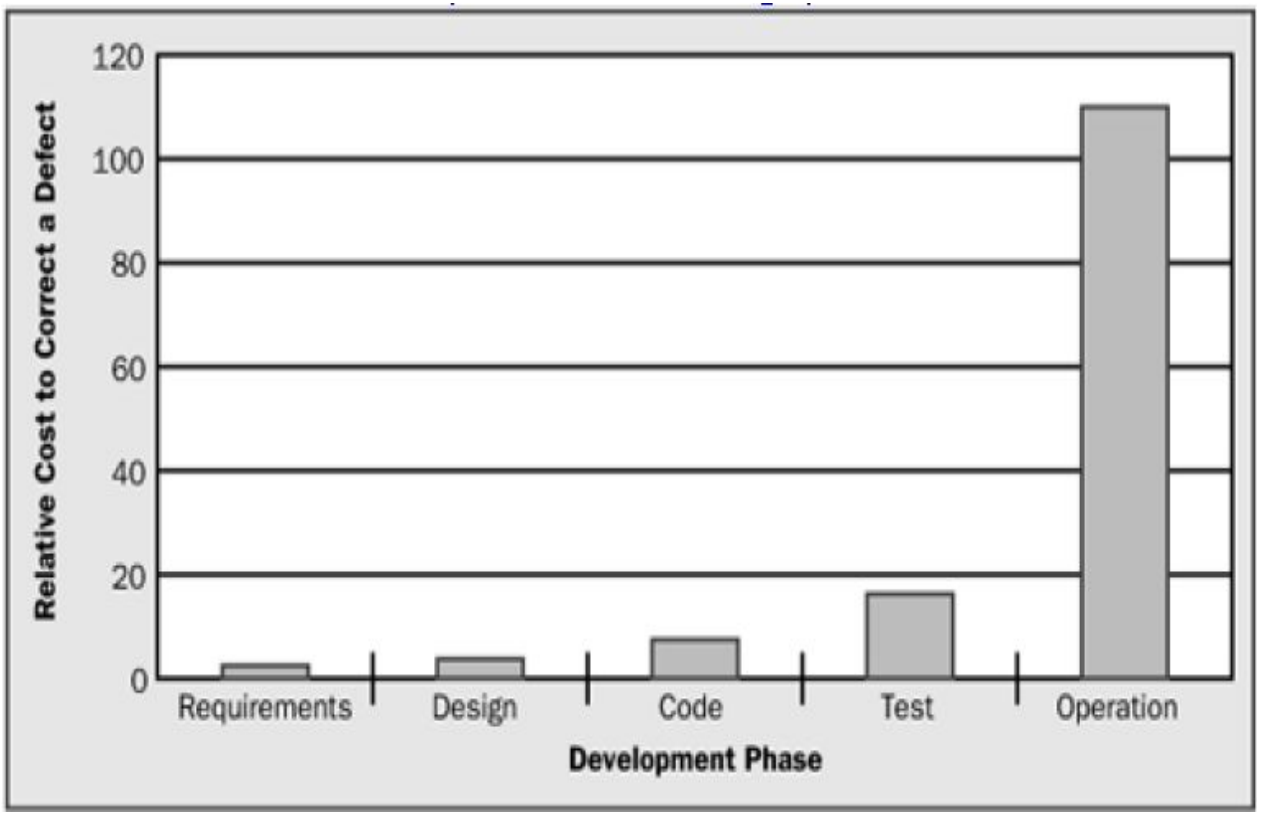


1.1 Motivación

- Datos del Standish Group muestran que los motivos más comunes del fracaso de un proyecto software:
  - Objetivos poco realistas
  - Objetivos ambiguos
  - Implicación insuficiente de los usuarios
  - Requisitos incompletos
  - Mal manejo de los cambios en los requisitos

1.1 Motivación

- Coste de corregir errores cometidos en la especificación de requisitos (Bohem y Basili, 2001)

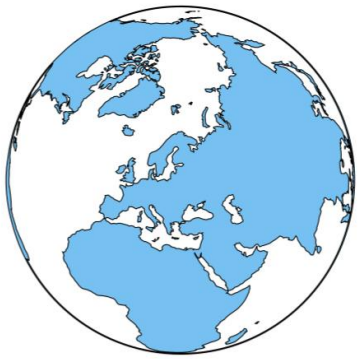


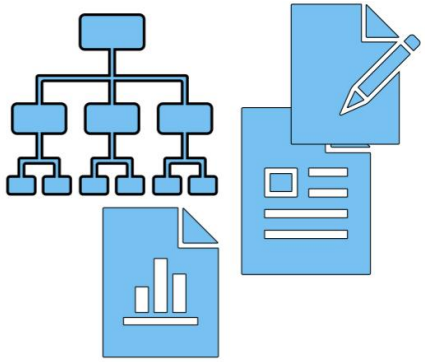
1.1 Motivación

- ¿Qué ofrece la Ingeniería de Requisitos para enfrentar estos problemas?

1. Proporciona técnicas y herramientas que facilitan la **extracción** de los requisitos.
2. Garantiza que la **especificación** de requisitos que sirve de base para el diseño del software, sea Consistente, Correcta y Completa.
3. Proporciona técnicas y herramientas para una correcta **gestión de los cambios** en los requisitos.

1.2 Definiciones





Ingeniería de requisitos

Es la rama de la Ingeniería del Software que trabaja con **objetos del mundo real**, servicios requeridos, restricciones, etc. con el propósito de especificar el comportamiento del sistema y hacer que los requisitos se encuentren perfectamente verificados y validados antes de alcanzar la fase de diseño en el proyecto. Los buenos requisitos deben ser medibles, comprobables, sin ambigüedades o contradicciones, etc.

1.2 Definiciones

Requisito *(según IEEE)*

a) Una condición o capacidad exigida por el usuario para solucionar un problema o alcanzar un objetivo.

b) Una condición o capacidad que debe poseer un sistema, o un componente de un sistema, para satisfacer un contrato, estándar, especificación u otro documento formal.

Un requisito maneja objetos del mundo real.
NO incluye detalles de diseño o implementación.

1.2 Definiciones

Stakeholder

Persona u organización que tiene algún tipo de interés en el sistema a desarrollar y que puede **afectar** o **verse afectado** (de manera directa o indirecta) por el mismo.

Ejemplos:

- Usuarios
- Propietarios o socios
- Clientes
- Proveedores
- Gobiernos/comunidades (legislación)
- Otras organizaciones (estándares)

1.3 Clasificación

- Klaus Pohl, plantea otra clasificación más reducida, que establece tres tipos de requisitos:
  1. **Requisitos funcionales**: *describen cómo debe comportarse el sistema.*
  2. **Requisitos de calidad**: *no describen funcionalidad sino la calidad que se espera de los servicios que va a ofrecer el sistema.*
  3. **Restricciones**: *son limitaciones para el producto (tecnológicas), o para el proceso de desarrollo (tiempo o recursos).*

Ej. "*El usuario introduce el título de un libro y el sistema devuelve información del mismo: autor, editorial y año de publicación [1], tardando menos de 5 segundos en dar la respuesta [2].*"

1.4 Requisitos funcionales

Son declaraciones sobre **servicios** que el sistema debe ofrecer, o cómo el sistema debe **responder** a determinadas entradas o cómo el sistema debería **comportarse** en situaciones particulares.

Ejemplo (sistema antirrobo): "El sistema informará a la empresa de seguridad si un sensor detecta daños o rotura en un panel de cristal".

La necesidad o el problema se describe desde el punto de vista del **USUARIO** y el **requisito** se plantea desde el punto de vista del **SISTEMA**.

1.4 Requisitos funcionales

Hay tres PERSPECTIVAS desde las cuales se deben plasmar los requisitos funcionales:

- **Perspectiva de datos**. Qué conceptos y datos se manejan en el sistema (dará lugar a clases conceptuales, atributos, etc.)
- **Perspectiva de comportamiento**. Cómo se va a comportar el sistema de cara al exterior (interfaz de usuario e interfaces con otros sistemas externos.)
- **Perspectiva funcional**. Qué funciones va a realizar el sistema para desempeñar el comportamiento previsto (métodos y funciones que se implementarán en los objetos.)

1.4 Requisitos funcionales

Ejercicio:

Descripción del problema: *Quiero vender música a través de Internet.*

NECESIDADES (perspectiva de usuarios)

- (N1) Los usuarios comprarán créditos para adquirir canciones.
- (N2) Los usuarios buscarán las canciones que deseen y las pagarán con créditos.
- (N3) Los usuarios tendrán algunos días para descargar en su ordenador las canciones que hayan adquirido.

1.4 Requisitos funcionales

Ejercicio:

- (N1) Los usuarios comprarán créditos para adquirir canciones.
  - (R1) El sistema debe registrar la información de los usuarios y los créditos que poseen.
  - (R2) El sistema debe preguntar al usuario la cantidad de créditos deseada y proporcionar un método automatizado para que el usuario efectúe el pago.

Ejercicio propuesto: *Plantear los requisitos funcionales para las otras dos necesidades expuestas.*

1.5 Requisitos de calidad

Requisitos no funcionales relacionados con la calidad del sistema en desarrollo.

Disponibilidad
Eficiencia
Flexibilidad
Integridad
Interoperabilidad
Confiabilidad
Robustez
Usabilidad
Mantenibilidad
Portabilidad
Reusabilidad
Testabilidad

1.5 Requisitos de calidad

Disponibilidad (Availability): Porcentaje de tiempo durante el cual el sistema está plenamente operativo y disponible para usarlo.

Eficiencia (Efficiency): Medida de aprovechamiento de los recursos hardware: tiempo de procesador, memoria, puertos de comunicación, etc.

Flexibilidad (Flexibility): Facilidad de extender el sistema con nuevas capacidades.

Integridad (Integrity): Grado de protección del sistema frente a acceso no autorizado, violación de datos privados, pérdida de información e “infecciones” por causa de software malintencionado.

Interoperabilidad (Interoperability): Facilidad para intercambiar datos o servicios con otros sistemas.

Confiabilidad (Reliability): Probabilidad de que el sistema funcione sin fallos durante un periodo específico de tiempo.

1.5 Requisitos de calidad

Robustez (Robustness): Grado en que un sistema o componente continúa funcionando correctamente cuando es sometido a entradas inválidas, defectos en sistemas o componentes conectados o condiciones operativas no esperadas.

Usabilidad (Usability): Facilidad de uso o grado de esfuerzo por parte del usuario para preparar las entradas o interpretar las salidas del sistema.

Mantenibilidad (Maintainability): Facilidad para corregir defectos o realizar cambios en el sistema una vez que ya está operativo.

Portabilidad (Portability): Esfuerzo requerido para migrar un sistema o componente desde una plataforma operativa a otra (ej. Cambio de sistema operativo, cambio de sistema gestor de bases de datos, …)

Reusabilidad (Reusability): Grado en que los componentes pueden ser usados en otros sistemas distintos de aquél para el cual han sido desarrollados inicialmente.

Testabilidad (Testability): Facilidad de probar o testear los componentes software integrados en el sistema para encontrar posibles defectos.

1.5 Requisitos de calidad

EJEMPLOS

Se dedicará una hora diaria para el mantenimiento del sistema. El resto del tiempo, el sistema estará plenamente operativo.

El periodo de aprendizaje hasta que los usuarios dominen la interfaz del sistema al 100% será como máximo de un mes.

Todos los datos serán públicos por lo que no se requiere ningún tipo de autenticación para la consulta de la información que ofrece el sistema.

El sistema ofrecerá una interfaz perfectamente definida para que programas externos puedan intercambiar datos con el mismo, sin ningún esfuerzo adicional de desarrollo.

Al menos el 90% de los componentes software desarrollados tendrán una alta cohesión y una interfaz perfectamente definida que permita su uso en otros sistemas similares.

La prueba de corrección de todos los componentes software mediante el método de “prueba de caja negra” requerirá un esfuerzo máximo de 1 hombre x mes.

1.5 Requisitos de calidad

EJEMPLOS

Los puertos de comunicación del equipo donde se instale el sistema en desarrollo se ocuparán como máximo en un 30%.

Ante una caída del servidor central, el sistema podrá seguir funcionando de manera autónoma en los equipos cliente. Se habilitarán procesos de sincronización para que, una vez restablecido el servicio, se vuelva a centralizar toda la información.

Los mismos usuarios podrán incorporar al sistema nuevas opciones de obtención de informes, sin la participación del personal informático.

El acceso al sistema estará libre de fallos, o como mucho se admitirá una tasa de fallos de un 1 por mil.

Inicialmente el sistema a desarrollar funcionará en plataformas Windows. No obstante, se contempla la migración a otras plataformas, para lo cual el esfuerzo máximo de adaptación será de 2 hombres x mes.

La inclusión de nuevos campos descriptivos en las tablas o ficheros maestros del sistema, una vez que éste se encuentre operativo, no implicará la participación de personal informático.

1.5 Requisitos de calidad

EJEMPLOS

Disponibilidad: Se dedicará una hora diaria para el mantenimiento del sistema. El resto del tiempo, el sistema estará plenamente operativo.

Eficiencia: Los puertos de comunicación del equipo donde se instale el sistema en desarrollo se ocuparán como máximo en un 30%.

Flexibilidad: Los mismos usuarios podrán incorporar al sistema nuevas opciones de obtención de informes, sin la participación del personal informático.

Integridad: Todos los datos serán públicos por lo que no se requiere ningún tipo de autenticación para la consulta de la información que ofrece el sistema.

Interoperabilidad: El sistema ofrecerá una interfaz perfectamente definida para que programas externos puedan intercambiar datos con el mismo, sin ningún esfuerzo adicional de desarrollo.

Confiabilidad: El acceso al sistema estará libre de fallos, o como mucho se admitirá una tasa de fallos de un 1 por mil.

1.5 Requisitos de calidad

EJEMPLOS

Robustez: Ante una caída del servidor central, el sistema podrá seguir funcionando de manera autónoma en los equipos cliente. Se habilitarán procesos de sincronización para que, una vez restablecido el servicio, se vuelva a centralizar toda la información.

Usabilidad: El periodo de aprendizaje hasta que los usuarios dominen la interfaz del sistema al 100% será como máximo de un mes.

Mantenibilidad: La inclusión de nuevos campos descriptivos en las tablas o ficheros maestros del sistema, una vez que éste se encuentre operativo, no implicará la participación de personal informático.

Portabilidad: Inicialmente el sistema a desarrollar funcionará en plataformas Windows. No obstante, se contempla la migración a otras plataformas, para lo cual el esfuerzo máximo de adaptación será de 2 hombres x mes.

Reusabilidad: Al menos el 90% de los componentes software desarrollados tendrán una alta cohesión y una interfaz perfectamente definida que permita su uso en otros sistemas similares.

Testabilidad: La prueba de corrección de todos los componentes software mediante el método de “prueba de caja negra” requerirá un esfuerzo máximo de 1 hombre x mes.

1.5 Requisitos de calidad

En el proceso iterativo de descomposición hasta llegar a la especificación final de requisitos, un requisito de calidad puede convertirse en un conjunto de requisitos funcionales.

Ejemplo:
R1 El sistema garantiza su propia seguridad.

Es un requisito de calidad, pero está especificado a un alto nivel de abstracción. Al descomponerlo vamos a llegar a requisitos más específicos, que pueden ser funcionales y/o de calidad.

1.5 Requisitos de calidad

Ejemplo:
R1 El sistema garantiza su propia seguridad.

Al leer este requisito, nos surgen preguntas: ¿qué entendemos por seguridad? ¿Qué propiedades debería tener el sistema para garantizar la seguridad? ¿Cómo probar si realmente el sistema construido es o no seguro?

R1 se puede convertir en los siguientes requisitos:

1.5 Requisitos de calidad

R1.1 Cada usuario se identificará en el sistema con su nombre y contraseña antes de poder usar cualquier otra función del sistema (requisito funcional).
R1.2 El sistema recordará al usuario cada cuatro semanas que debe cambiar la contraseña (requisito funcional).
R1.3 Cuando el usuario cambia su contraseña, el sistema validará que la nueva contraseña tiene al menos ocho caracteres y contiene caracteres alfanuméricos distintos (números, letras, puntuación, …) (requisito funcional)
R1.4 La contraseña almacenada debe ser privada y exclusiva del usuario propietario (requisito de calidad: integridad).

1.6 Restricciones

Las restricciones (constraints), como su nombre indican, restringen o limitan el **proceso de desarrollo** o las **propiedades del sistema** que se está desarrollando.

Normalmente vienen impuestas por los gestores de la organización (ej. restricciones de tiempo o recursos) o por el entorno o contexto en el cual el sistema va a funcionar (aspectos tecnológicos, legales, etc.)

1.6 Restricciones

Una restricción está caracterizada por:

- El **origen** de la restricción, que puede ser cultural, legal, impuesto por la organización, restricción de tipo físico o relativa a la gestión del proyecto.

- El **objeto** de la restricción, que puede ser el sistema en sí mismo (ej. restricciones sobre una determinada función del sistema) o el proceso de desarrollo (ej. duración del proyecto).

1.6 Restricciones

Ejemplo:

C1 Las tiendas tendrán acceso a Internet con un ancho de banda de 20 Mbps.

C2 El esfuerzo dedicado al desarrollo del sistema no debe exceder las 480 personas mes.

C1 presenta una restricción sobre el sistema mientras que C2 es una restricción sobre el proceso de desarrollo.

1.7 Ejercicio

Clasificar los siguientes requisitos (Funcional, Calidad o Restricción. Si fuese Calidad, especificar aspecto.)

Gestión de acceso a un club VIP :

R1 El sistema ofrecerá una interfaz fácil de aprender: un curso de formación de dos semanas será suficiente para la puesta en marcha.

R2 El sistema de información generará mensualmente un informe con las solicitudes aceptadas y denegadas.

R3 Si el PIN que el usuario introduce es correcto, el sistema abrirá la puerta y registra el acceso autorizado: fecha, hora y nombre del propietario del PIN.

R4 El sistema debe estar disponible el 1 de Mayo de 2013

R5 Los datos se almacenarán utilizando MySQL, versión 7.3 o posterior.

1.8 Características requisitos

- Completo
- Correcto
- Factible
- Preciso
- Priorizado
- Verificable

1.8 Características requisitos

- Completo

Cada requisito debe describir completamente la funcionalidad a la que se refiere, antes de ser entregado para su diseño e implementación. Si se sospecha que pueda faltar alguna información, se debe dejar indicado (por ejemplo, con las siglas “TBD”, “to be determined”, o “PD”, “pendiente de determinar”; así, antes de entregar los requisitos para su desarrollo, se buscarán las apariciones de las siglas TBD y se resolverán.)

1.8 Características requisitos

- Completo
- Correcto

La corrección de un requisito viene determinada por la fuente del mismo. Por ejemplo, un requisito de usuario es correcto porque expresa una necesidad real de un usuario. También puede corresponderse con una necesidad impuesta por un sistema externo, un estándar,… o cualquier fuente con autoridad para expresar requisitos. Por otro lado, un requisito será incorrecto si entra en conflicto con un requisito correcto de “orden superior”.

1.8 Características requisitos

- Completo
- Correcto
- Factible

Cada requisito debe ser realizable dentro de las capacidades y limitaciones conocidas del sistema y de su entorno operativo. Para evitar requisitos no factibles, durante las tareas de adquisición de requisitos, el analista debe informar a los clientes qué requisitos serían técnicamente inalcanzables y cuáles se podrían conseguir pero a un costo muy elevado.

1.8 Características requisitos

- Completo
- Correcto
- Factible
- Preciso

Se trata de que los requisitos sean comprensibles, con ausencia de ambigüedad. Los lectores deben ser capaces de entender lo que un requisito está diciendo. Es aconsejable añadir un **glosario** con términos que puedan resultar desconocidos o novedosos para el lector. Evitar sinónimos y términos ambiguos (con dos o más significados).

1.8 Características requisitos

- Completo
- Correcto
- Factible
- Preciso
- Priorizado

Se debe asignar prioridades de implementación a cada requisito de manera que sea de utilidad para resolver posibles conflictos entre requisitos y valorar la corrección de los mismos.

1.8 Características requisitos

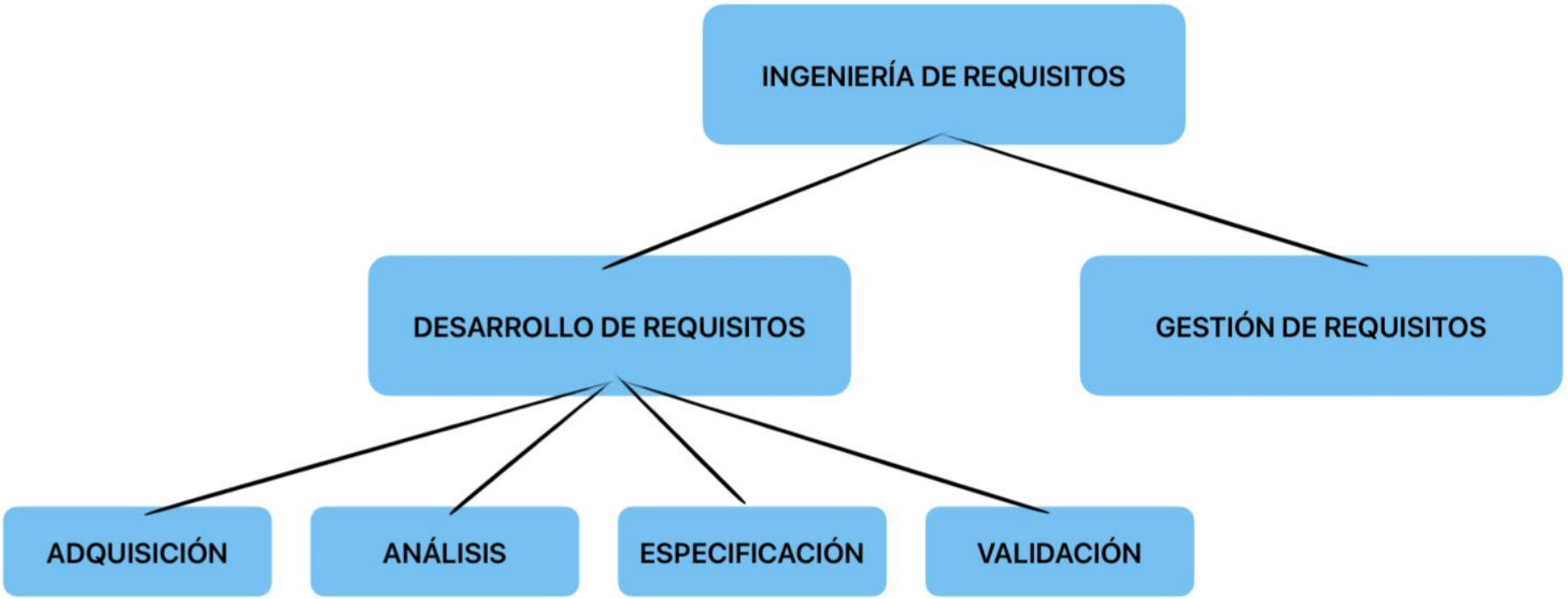
- Completo
- Correcto
- Factible
- Preciso
- Priorizado
- Verificable

  Existe alguna forma de probar o demostrar, de manera objetiva, la correcta implementación del requisito.

1.9 Desarrollo *vs* Gestión

- Hemos de distinguir las tareas que corresponden al desarrollo de requisitos de aquellas otras que corresponden a la gestión. Normalmente, el perfil de la persona o personas encargadas de dichas tareas será diferente.
- El perfil del responsable del desarrollo de requisitos implica conocimientos y experiencia en la ingeniería de software. Deberá conocer y dominar las técnicas de adquisición, especificación y análisis de requisitos.
- Por el contrario, el responsable de la gestión de requisitos tiene un perfil menos técnico, requiriendo mayormente habilidades de gestión, organizativas y de negociación.

1.9 Desarrollo *vs* Gestión



Tareas de **Desarrollo** de Requisitos

Tareas relacionadas con la adquisición, especificación, negociación y validación de requisitos. Se realizan en los inicios del proyecto. Ejemplos:

- Identificar los conceptos (clases conceptuales) que van a formar parte del producto software.
- Obtener las necesidades individuales de cada usuario
- Comprender las tareas y objetivos de usuario
- Analizar la información recibida de los usuarios y distinguir requisitos funcionales, no funcionales, reglas de negocio, etc.
- Negociar prioridades de implementación
- Trasladar las necesidades de usuario a documentos de especificación de requisitos y modelos formales.
- Revisar los documentos de requisitos para asegurar una perfecta comprensión de las necesidades de los usuarios y corregir posibles problemas antes de pasarlos al equipo de desarrollo.

Tareas de **Gestión** de Requisitos

Tareas relacionadas con la trazabilidad y los cambios que se pueden producir en los requisitos. Se realiza a lo largo de todo el proyecto. Ejemplos:

- Llevar un control de versiones de requisitos. Una "línea base" (baseline) constituye una especificación de requisitos aprobada y no modificable. Cualquier cambio implica una nueva versión.
- Revisar los cambios de requisitos propuestos y evaluar el impacto antes de que sea aprobado.
- Incorporar cambios de requisitos aprobados en el proyecto, de una manera controlada.
- Supervisar la trazabilidad de los requisitos: correspondencia entre un requisito y su correspondiente diseño, código fuente, casos de prueba, etc.
- Seguimiento del estado de los requisitos y cambios en los mismos, durante todo el proyecto.