

Unidad 4


Análisis y Diseño



Parte I - Análisis

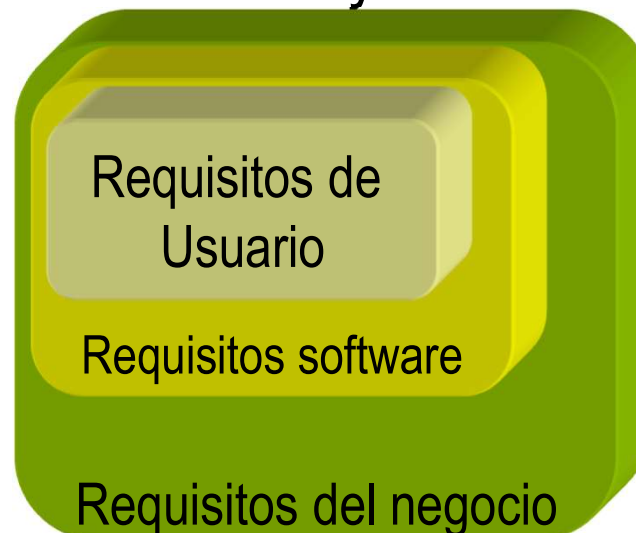


Contenido

- 
- Definiciones, Importancia, Objetivos y Actividades.
 - Educción de Requisitos.
 - Tipos de Requisitos.
 - Análisis de Requisitos.
 - Representación de Requisitos.
 - Validación de Requisitos.
 - Perfil del analista.
 - Principales errores del análisis.
 - Documentación final: ERS.

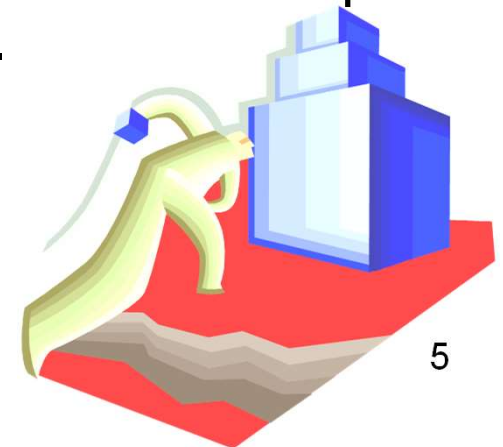
Definición

- Análisis de requisitos:
 - Análisis del problema y especificación completa del comportamiento externo que se espera del sistema software que se va a construir, así como de los flujos de información y control.



Definición de Requisitos

- Para el usuario:
 - ☐ Son las condiciones o capacidades necesarias para que el usuario pueda resolver un problema o alcanzar un objetivo.
- Para el equipo de desarrollo:
 - ☐ Son las condiciones o capacidades que debe reunir un sistema para satisfacer un contrato, estándar o cualquier otro documento impuesto formalmente.



Requisitos de usuario y Requisitos software

■ Requisitos de usuario

- ☐ Son declaraciones en lenguaje natural de las funciones o acciones que los distintos usuarios pueden realizar con la aplicación y bajo qué restricciones.
- ☐ Conforman el Documento de Requisitos de Usuario (DRU/ERU).

■ Requisitos software

- ☐ Especifican de una manera completa, consistente y detallada qué debe hacer y cómo debe comportarse el software para cumplir con los objetivos de la aplicación.
- ☐ Sirven de base a los desarrolladores para diseñar el sistema.
- ☐ Se recogen en la Especificación de requisitos Software (ERS).
- ☐ Deben responder a la pregunta: ¿qué características necesita cumplir el sistema software para permitir alcanzar los requisitos expuestos en el DRU?

Requisitos
de usuario

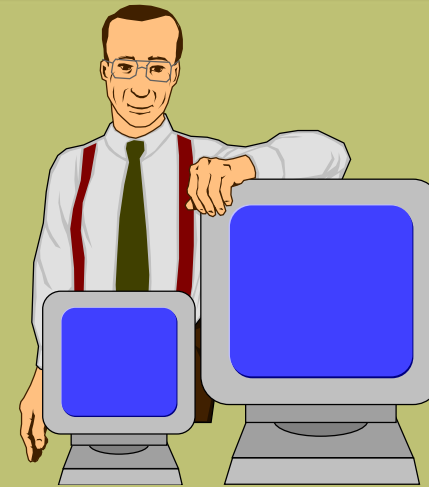


Requisitos
software

Papeles

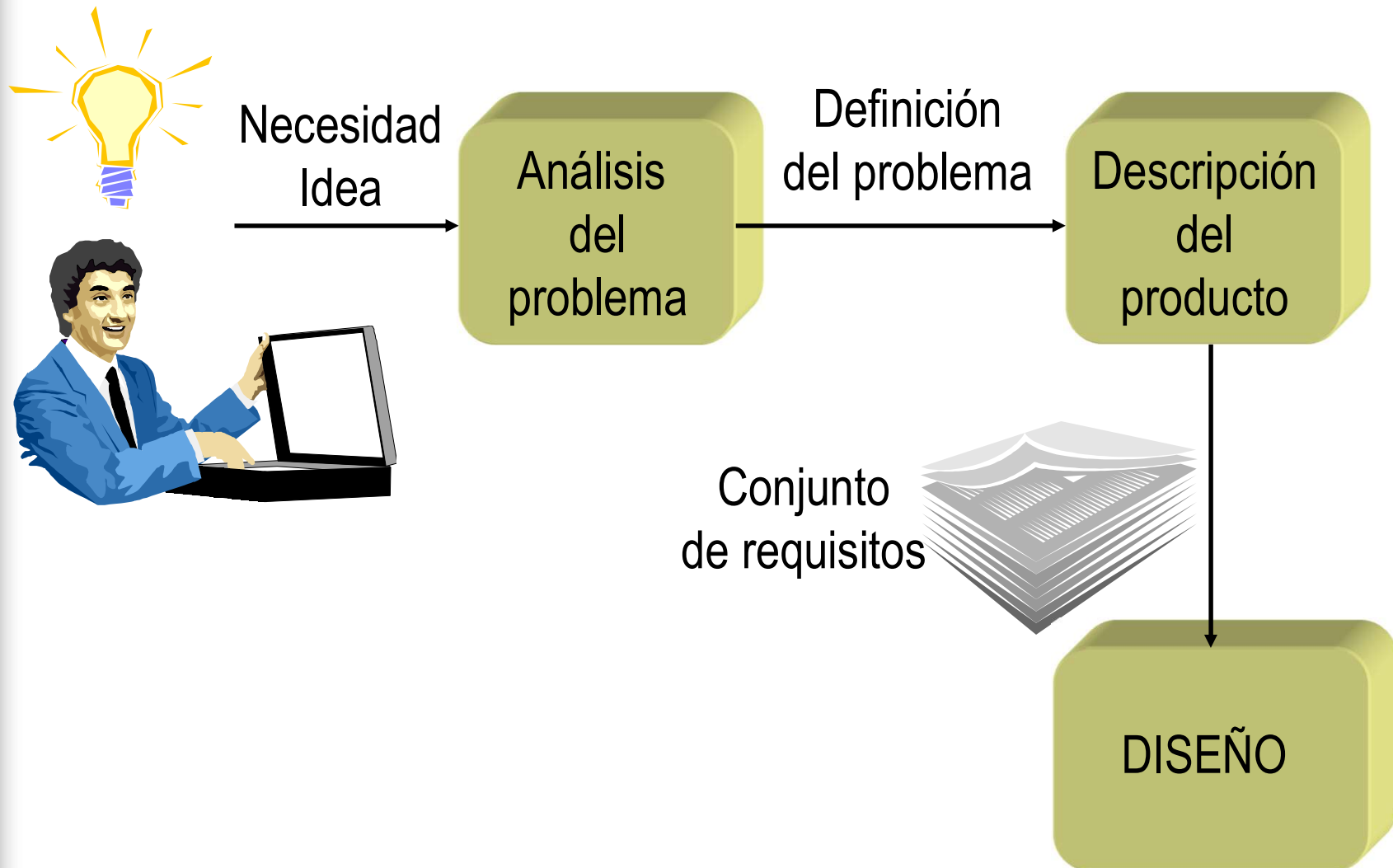


Los clientes y usuarios plantean el problema actual, el resultado que esperan obtener y las condiciones que esperan.

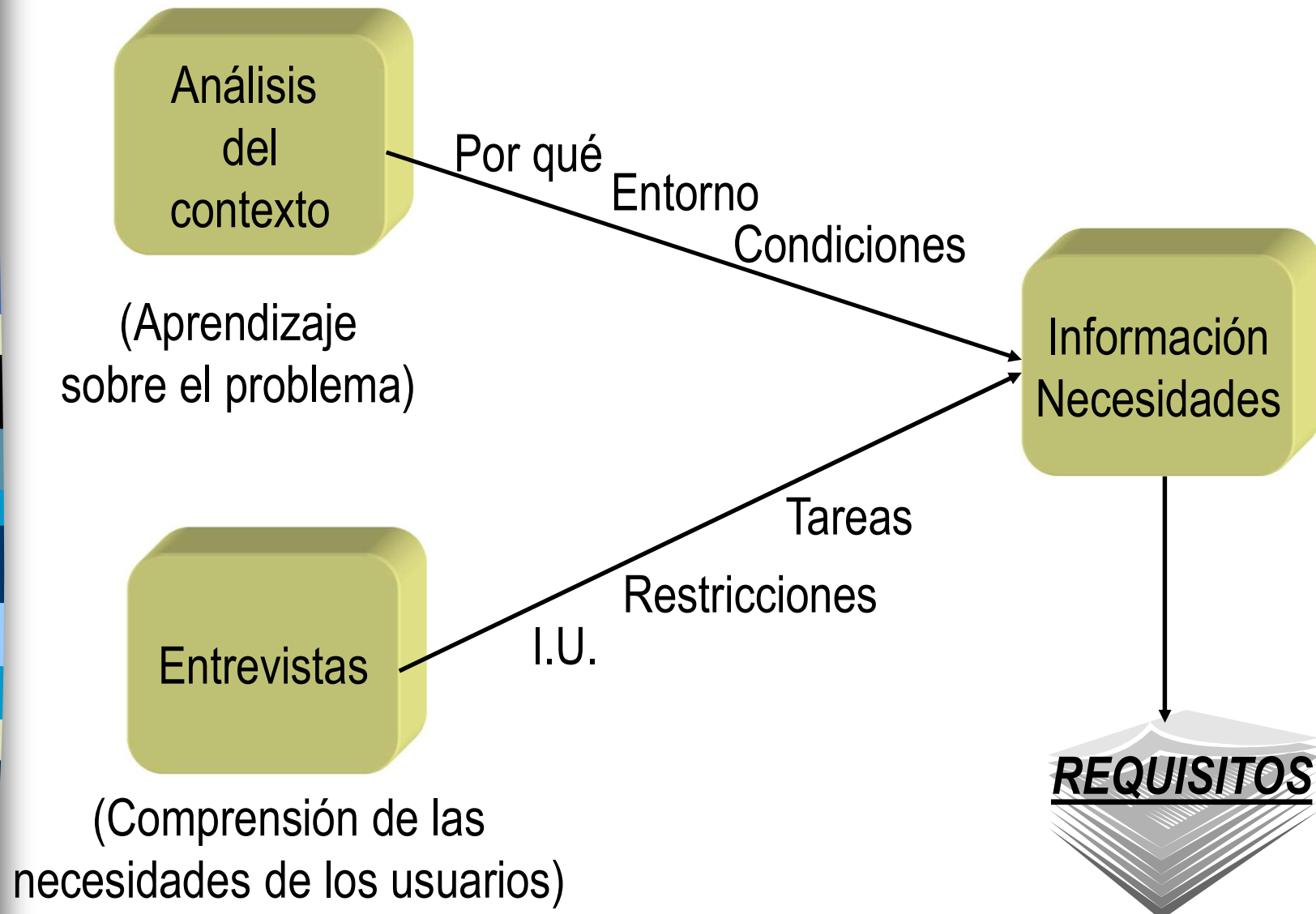


El ingeniero del software pregunta, analiza, asimila y presenta la solución adecuada.

Actividades Principales



Fuentes y Técnicas



Objetivos e Importancia

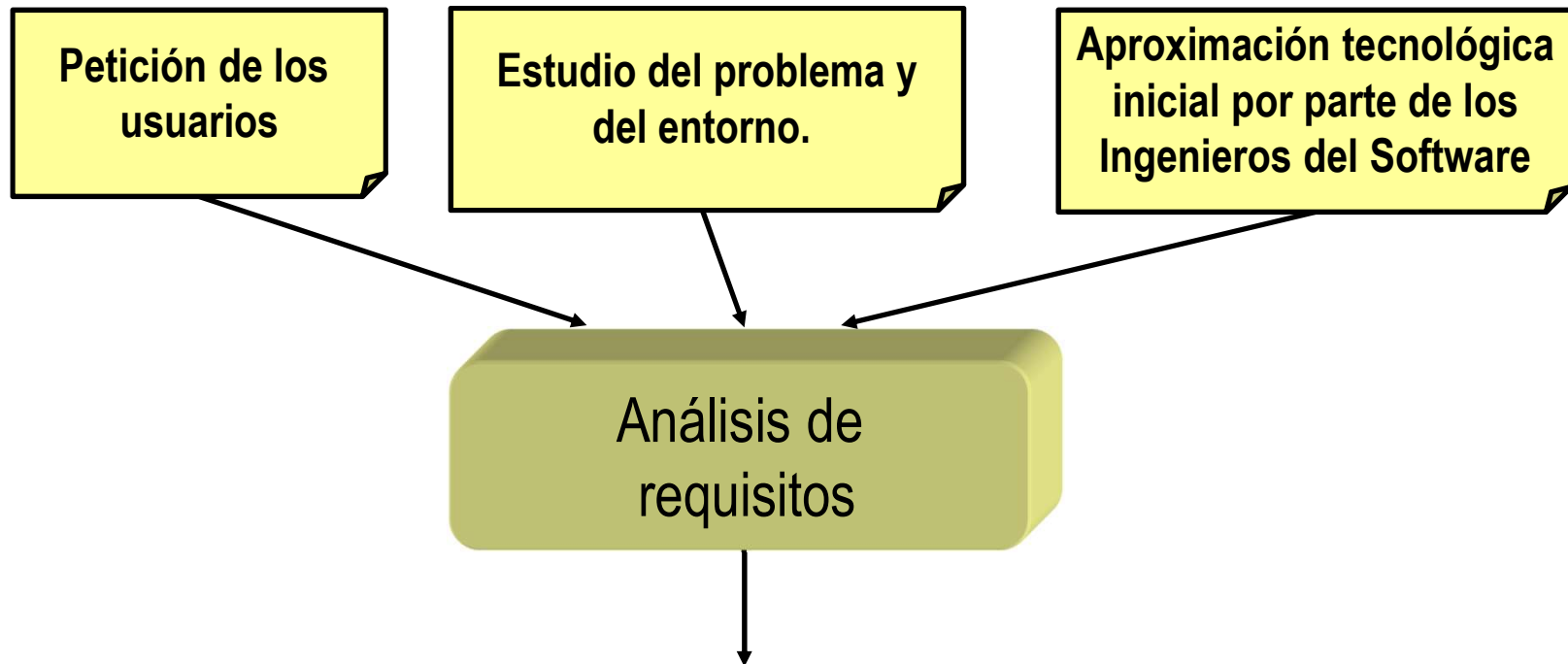
■ Otros objetivos:

- ☐ Proporcionar los medios de comunicación entre todas las partes: clientes, usuarios, jefe de proyecto, analistas y resto del equipo de desarrollo.
- ☐ Servir como base para las actividades de prueba y validación.
- ☐ Ayudar al control de la evolución del software.

■ Importancia:

- ☐ El impacto de cometer errores en la fase de análisis de requisitos puede resultar en que el producto final no satisfaga las necesidades de los usuarios.

Entradas y Salidas



- Especificación de requisitos del software (ERS).
- Definición de lo que los usuarios quieren y lo que se les va a proporcionar.
- Definición de restricciones.


Principios del Análisis

- Se debe comprender el problema y su entorno.
- Los requisitos han de determinarse siguiendo una aproximación descendente, primero se analiza el problema globalmente, para pasar posteriormente al detalle.
- Se debe representar la información, función y comportamiento del sistema.
- Se debe separar el qué del cómo.
- La especificación de requisitos debe poder ser ampliable.

Tareas

- Educación de requisitos.
 - ☐ Identificar los requisitos que se obtienen de los usuarios y clientes.
- Análisis de Usuarios y de las Tareas
 - ☐ Identificar potenciales usuarios del sistema, su jerarquía y las tareas a realizar.
- Análisis del problema y de los requisitos.
 - ☐ Razonar sobre los requisitos educidos, combinar requisitos relacionados, establecer prioridades entre ellos, determinar su viabilidad, etc.
- Representación (modelización).
 - ☐ Registrar los requisitos de alguna forma, incluyendo lenguaje natural, lenguajes formales, modelos, prototipos, maquetas, UML, etc.
- Validación.
 - ☐ Examinar inconsistencias entre requisitos, determinar la corrección, ambigüedad, etc. Establecer criterios para asegurar que el software reúna los requisitos cuando se haya producido. El cliente, usuario y desarrollador se deben poner de acuerdo.

Contenido

- 
- Definiciones, Importancia, Objetivos y Actividades.
 - Educación de Requisitos.
 - Tipos de Requisitos.
 - Análisis de Requisitos.
 - Representación de Requisitos.
 - Validación de Requisitos.
 - Perfil del analista.
 - Principales errores del análisis.
 - Documentación final: ERS.

Educción de requisitos: Qué y Cómo

■ Qué


- ☐ Proceso a través del cual, los clientes, compradores o usuarios de un sistema software exponen, formulan, articulan y comprenden sus requisitos.

■ Cómo

- ☐ Reuniones, entrevistas, análisis de las tareas, lectura de documentos o manuales, etc.



Contenido

- 
- Definiciones, Importancia, Objetivos y Actividades.
 - Educción de Requisitos.
 - Tipos de Requisitos.
 - Análisis de Requisitos.
 - Representación de Requisitos.
 - Validación de Requisitos.
 - Perfil del analista.
 - Principales errores del análisis.
 - Documentación final: ERS.

Tipos de requisitos (I)

■ Requisitos funcionales.

Acciones fundamentales que tienen que tener lugar en la ejecución del software.

Son **acciones elementales** necesarias para el correcto comportamiento de nuestro sistema

Ejemplo: “El usuario podrá dar de alta un elemento”



Tipos de requisitos (II)

■ Requisitos no funcionales.

Representan características o cualidades generales que se esperan del software para conseguir su propósito.

☐ Requisitos de interfaz y usabilidad.

- Menús, Ventanas, Mensajes de error, Formatos de Pantalla, etc.

☐ Requisitos operacionales.

- Modos de operación, back-ups, funciones de recuperación, etc.

☐ Requisitos de documentación.

- Idiomas, tipos de usuario, ayuda on-line, web, tutoriales, etc.

☐ Requisitos de seguridad.

- Diferentes niveles de acceso al sistema, protección, mantenimiento de históricos, claves, etc.

Tipos de requisitos (III)

■ Requisitos no funcionales.

☐ Requisitos de mantenibilidad y portabilidad.

- Grado en que debe ser fácil cambiar el software o portarlo.

☐ Requisitos de recursos.

- Tanto hardware como software. Ej: limitaciones sobre memoria, almacenamiento.

☐ Requisitos de rendimiento.

- Tiempo de respuesta, nº de usuarios, terminales soportadas, consumo de memoria, etc.

☐ Requisitos de comportamiento.

Tipos de requisitos (IV)

■ Requisitos no funcionales.

☐ Requisitos de disponibilidad.

- Especialmente para aquellos sistemas informáticos que necesiten estar conectados a la red.

☐ Requisitos de soporte.

- Incluyen la facilidad de instalación, facilidad de mantenimiento, facilidad de actualización y facilidad de portabilidad.

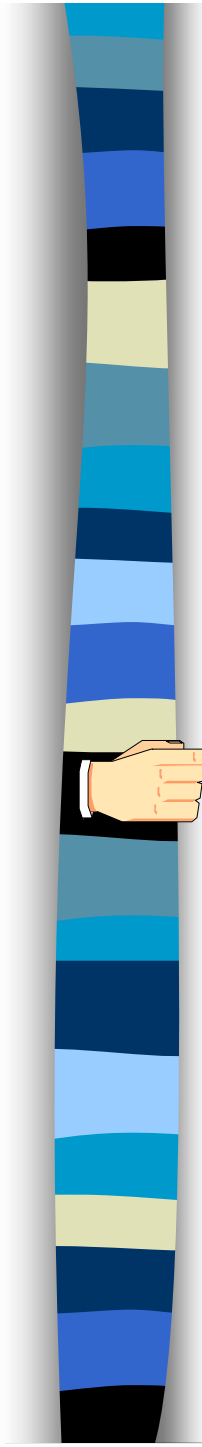
☐ Requisitos de verificación y fiabilidad.

- Recuperación ante situaciones anómalas o de error.

☐ Requisitos legales.

- Características que deben cumplir el sistema para cumplir con la legislación vigente.

Contenido

- 
- Definiciones, Importancia, Objetivos y Actividades.
 - Educción de Requisitos.
 - Tipos de Requisitos.
 - Análisis de Requisitos.
 - Representación de Requisitos.
 - Validación de Requisitos.
 - Perfil del analista.
 - Principales errores del análisis.
 - Documentación final: ERS.

Análisis de requisitos

■ Qué

- ☐ Proceso a través del cual se determina qué requisitos son aceptables y se definen cuáles van a formar parte del producto.

■ Cómo

- ☐ Evaluación de viabilidad técnica y económica.
- ☐ Valoración de riesgos.
- ☐ Clasificación de requisitos en categorías:
 - Obligatorios.
 - Deseables.
 - Accesorios.
 - ...

Contenido

- Definiciones, Importancia, Objetivos y Actividades.
- Educción de Requisitos.
- Tipos de Requisitos.
- Análisis de Requisitos.
- Representación de Requisitos.
- Validación de Requisitos.
- Perfil del analista.
- Principales errores del análisis.
- Documentación final: ERS.

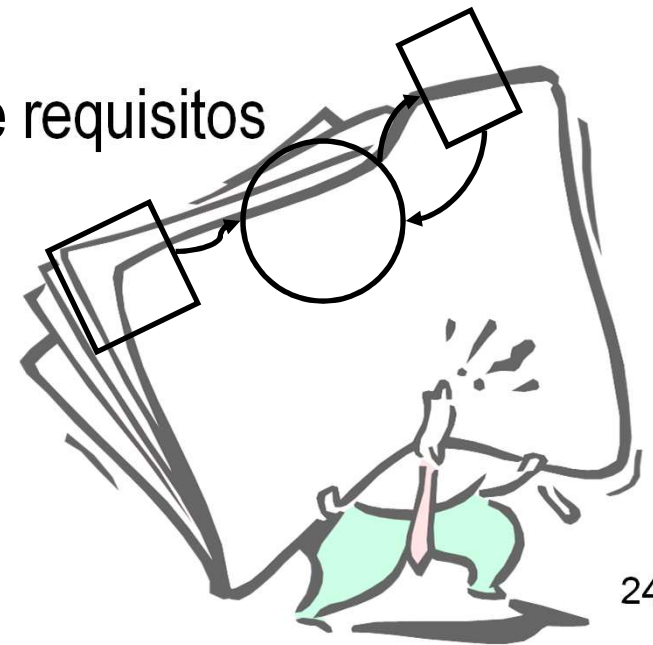
Representación de requisitos

■ Qué

- ☐ Proceso de registrar los requisitos de una o más formas y de especificar aquellos requisitos todavía no educidos.

■ Cómo

- ☐ Lenguaje natural; catálogos de requisitos
- ☐ Lenguaje formal.
- ☐ Modelos.
- ☐ Diagramas.
- ☐ Maquetas.
- ☐ ...



Técnicas de representación de requisitos

- Análisis estructurado.
 - ☐ Técnicas de análisis orientadas a datos.
 - ☐ Técnicas de análisis orientadas a funciones.
 - ☐ Técnicas de análisis orientadas a estados.
- Análisis orientado a objetos.
 - ☐ Diagramas de Comportamiento UML
 - Diagramas de Casos de Uso
 - Diagramas de Interacción
- Escenarios, *Storyboards*, Prototipos y Maquetas
- Lenguajes formales.



Modelos de desarrollo de productos software (I)

- El cliente no suele tener una idea clara de lo que quiere, o no sabe explicarlo bien.
- El responsable de desarrollo puede no estar seguro de la eficacia de un algoritmo, del enfoque a tomar en la interacción hombre-máquina, etc.
- Ayudan a comprender y validar los requisitos de usuario.
- Desarrollo de:
 - ☐ Maquetas
 - ☐ Prototipos

Modelos de desarrollo de productos software (II)

■ Maquetas :

- ☐ Cuando los requisitos no están claros.
- ☐ Cuando el alcance del proyecto no está bien definido.
- ☐ Cuando los usuarios no se muestran colaboradores.
- ☐ Cuando las comunicaciones con el entorno real presentan gran complejidad.

Maqueta

Sistema final

Modelos de desarrollo de productos software (III)

- La maqueta permite :
 - ☐ Adaptar el modelo mental del usuario con el del ingeniero del software, permitiendo una mejor educación de requisitos software
 - ☐ Validar los requisitos del usuario.
 - ☐ Comprobar la aceptación del usuario.
 - ☐ Comprobar la conexión e integración con el entorno.
 - ☐ Facilitar el diseño de la Interacción Persona-Ordenador y de las interfaces de usuario de la aplicación final.

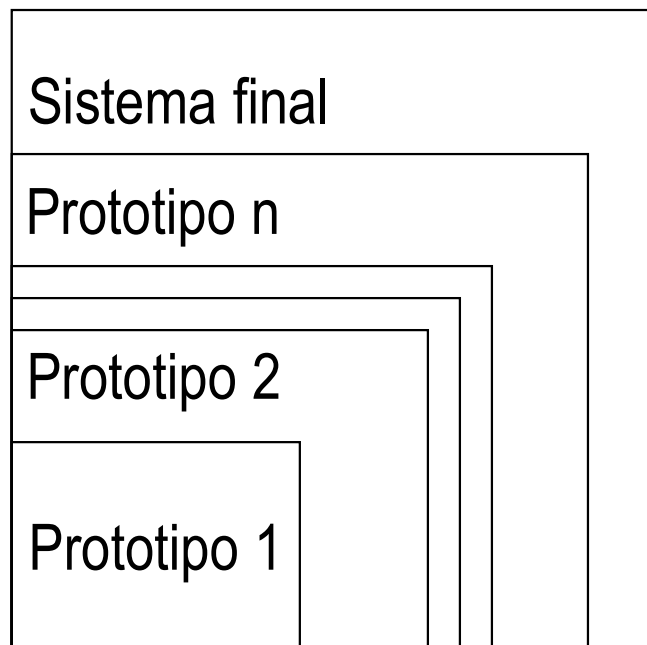
Modelos de desarrollo de productos software (IV)



Modelos de desarrollo de productos software (V)

■ Prototipos:

- ☐ Cuando no se conoce la complejidad total del desarrollo.
- ☐ Cuando el ingeniero del software no tiene muy clara la solución informática más adecuada.
- ☐ Se construye un prototipo de una parte del sistema.



Modelos de desarrollo de productos software (VI)

- El prototipado permite:
 - ☐ Educir y verificar los requisitos principales del usuario.
 - ☐ Verificar la viabilidad del diseño informático del sistema.
 - ☐ Facilitar el diseño de la Interacción Persona-Ordenador y de las Interfaces de Usuario del sistema.
 - ☐ El prototipo evoluciona iterativamente.

Contenido

- Definiciones, Importancia, Objetivos y Actividades.
- Educción de Requisitos.
- Tipos de Requisitos.
- Análisis de Requisitos.
- Representación de Requisitos.
- Validación de Requisitos.
- Perfil del analista.
- Principales errores del análisis.
- Documentación final: ERS.

Validación de requisitos (I)

■ Los requisitos deben ser:

☐ Completos.

Todo lo que el software tiene que hacer está recogido en el conjunto de requisitos.

☐ No ambiguos.

Cada requisito debe tener una sola interpretación.

☐ Relevantes.

Importancia para el sistema software a implementar.

☐ Traceables

Cada acción de diseño debe corresponderse con algún requisito, y debe poder comprobarse.

Validación de requisitos (II)

■ Los requisitos deben ser:

☐ Correctos.

Cada requisito establecido debe representar algo requerido por el usuario para el sistema que se construye.

☐ Consistentes.

Ningún requisito puede estar en conflicto con otro. Tipos de inconsistencias:

- Términos conflictivos: Si dos términos se usan en contextos diferentes para la misma cosa.
- Características en conflicto: Si en dos partes de la ERS se pide que el producto muestre comportamientos contradictorios.
- Inconsistencia temporal: Si dos partes de la ERS piden que el producto obedezca restricciones de tiempo contradictorias.

Validación de requisitos (III)

■ Ejemplos:

1. “... hasta 15 autobuses se dibujarán dentro de la misma ventana. Si excede el número se utilizará otra ventana diferente.”
2. “El sistema tendrá una interfaz de usuario sencilla de utilizar.”
3. “Los usuarios podrán pagar el servicio seleccionado con tarjeta de crédito mediante protocolo seguro siempre que introduzcan los datos solicitados en un tiempo menor o igual a 15 segundos.”
4. “El sistema permitirá a los usuarios realizar una búsqueda.”
5. “El sistema tendrá un tiempo de respuesta aceptable.” ³⁵

Contenido

- Definiciones, Importancia, Objetivos y Actividades.
- Educción de Requisitos.
- Tipos de Requisitos.
- Análisis de Requisitos.
- Representación de Requisitos.
- Validación de Requisitos.
- Perfil del analista.
- Principales errores del análisis.
- Documentación final: ERS.



Perfil del analista

- Personal de desarrollo con más experiencia.
- Dotes de comunicación.
- Capacidad de análisis y de síntesis.



Contenido

- Definiciones, Importancia, Objetivos y Actividades.
- Educción de Requisitos.
- Tipos de Requisitos.
- Análisis de Requisitos.
- Representación de Requisitos.
- Validación de Requisitos.
- Perfil del analista.
- Principales errores del análisis.
- Documentación final: ERS.

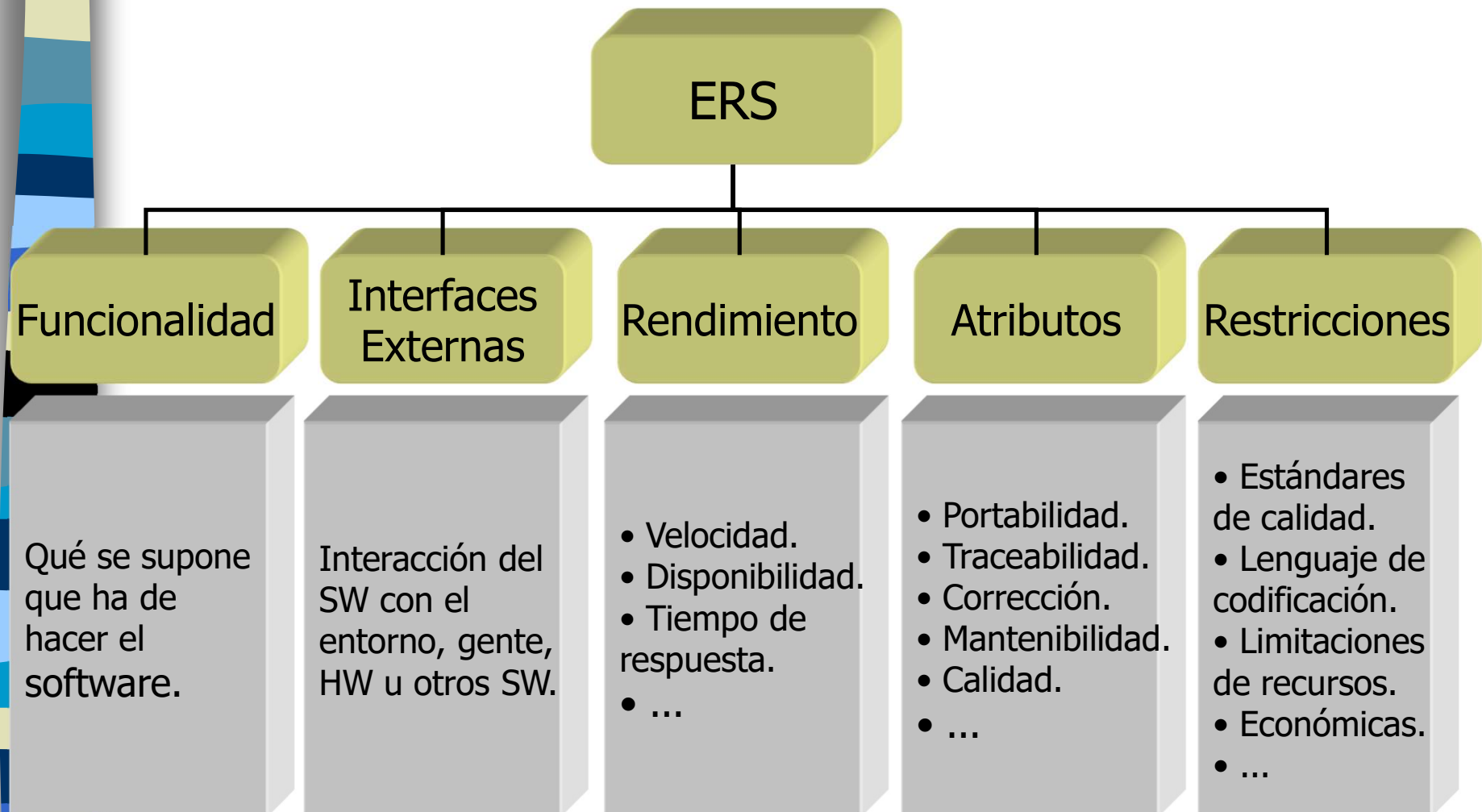
Principales errores del análisis

- Alto riesgo de mal entendimiento entre cliente/usuario e ingeniero del software (Requisitos ambiguos).
- Tendencia a acortar y minimizar la importancia de esta etapa.
- No establecer los criterios de aceptación del Software.
- Pobre estudio del problema: El ingeniero del software debe conocer bien el dominio del problema.
- No revisión de la especificación de requisitos por el cliente/usuario o el ingeniero del software.
- Permitir el continuo cambio de requisitos por parte del usuario.
- Introducir conceptos de implementación o diseño.

Contenido

- Definiciones, Importancia, Objetivos y Actividades.
- Educción de Requisitos.
- Tipos de Requisitos.
- Análisis de Requisitos.
- Representación de Requisitos.
- Validación de Requisitos.
- Perfil del analista.
- Principales errores del análisis.
- Documentación final: ERS.

ERS: Aspectos que considera



Documento final: Especificación de Requisitos del Software (ERS) (I)

■ Propósito.

- ☐ Proporcionar los medios de comunicación entre todas las partes: clientes, usuarios, analistas y diseñadores. Debe ser comprensible para todas las partes.
- ☐ Servir como base para las actividades de diseño, prueba y verificación.
- ☐ Ayudar al control de la evolución del sistema software.

Documento final: ERS (II)

1. Introducción.

1.1. Propósito.

1.2. Definiciones, acrónimos y abreviaturas.

1.3. Referencias.

1.4. Estructura.

2. Descripción general.

2.1. Alcance.

2.2. Funciones del producto.

2.3. Restricciones generales.

2.4. Dependencias.

2.5. Características de usuario.

Documento final: ERS (III)

3. Requisitos específicos.

- 3.1. Requisitos de interfaz.
- 3.2. Requisitos de rendimiento.
- 3.3. Requisitos operacionales.
- 3.4. Requisitos de recursos.
- 3.5. Atributos del sistema software: fiabilidad, disponibilidad, seguridad, etc.

4. Descripción de la información.

- 4.1. Flujo de datos.
- 4.2. Flujo de control.

Documento final: ERS (IV)

5. Descripción funcional.

5.1. Partición funcional.

5.2. Diagramas de soporte.

6. Descripción del comportamiento.

7. Criterios de validación.

7.1. Límites de rendimiento.

7.2. Clases de pruebas.

7.3. Respuesta esperada del producto.

7.4. Consideraciones especiales.

Parte II - Diseño



Contenido

- Introducción.
- Diseño estructurado y Diseño orientado a objetos con UML.
- Métricas de diseño.
- Guías de un buen diseño.
- Principales errores en el diseño.
- Documentación final del diseño.

Contenido



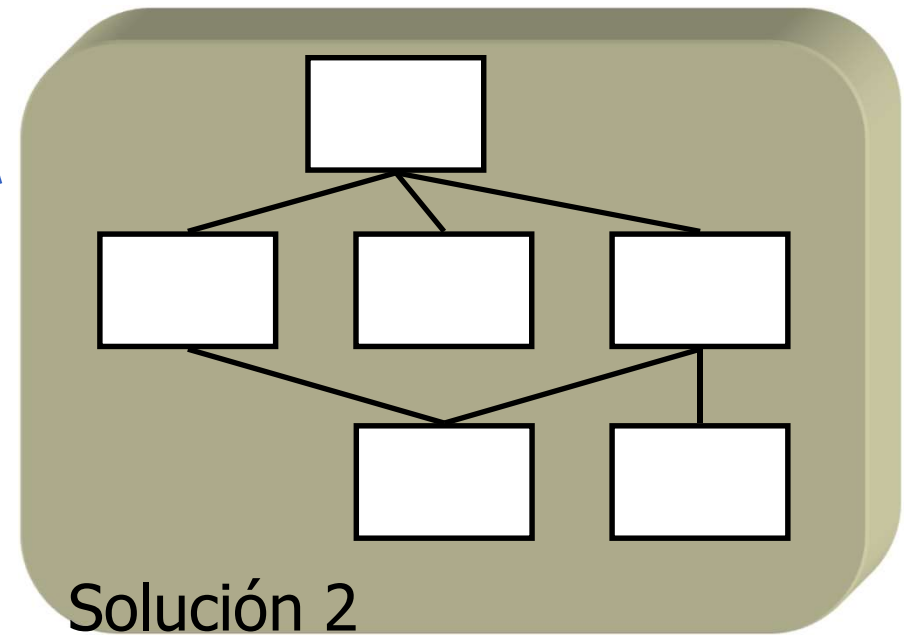
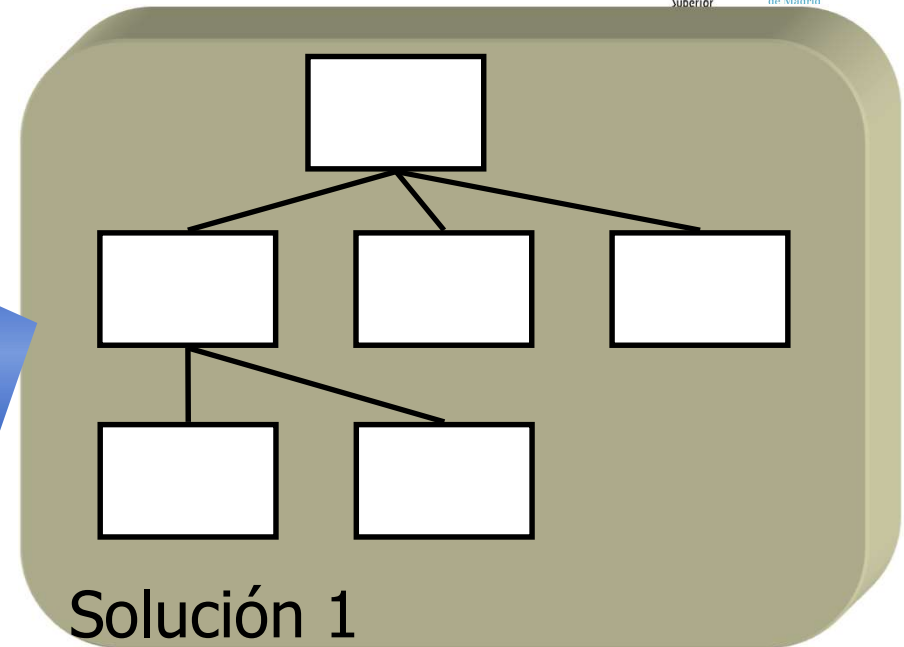
- Introducción.
 - ☐ Transición de análisis a diseño.
 - ☐ Definición y objetivos.
 - ☐ Niveles de diseño.
 - ☐ Tareas.
 - ☐ Principios básicos de diseño.
 - ☐ Métodos de diseño.
- Diseño estructurado y Diseño orientado a objetos con UML.
- Métricas de diseño.
- Guías de un buen diseño.
- Principales errores en el diseño.
- Documentación final del diseño.

Transición de análisis a diseño

- Cambiar la atención del qué al cómo.
 - ☐ ¿Qué hay que hacer? → Especificación de requisitos.
 - ☐ ¿Cómo hay que hacerlo? → Especificaciones de diseño.
- Transformación de la definición del problema a la solución software.

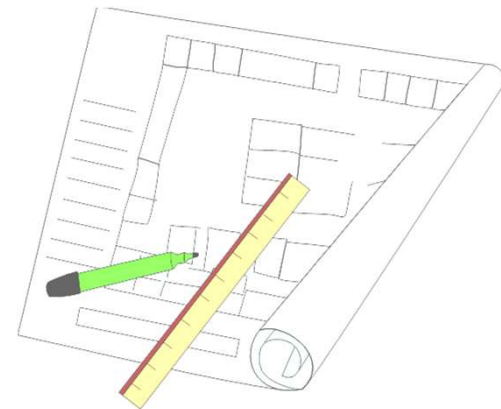


Diseño



Diseño: Definición

- Es el proceso de definición de la arquitectura, componentes, módulos, interfaces, procedimientos de prueba y datos de un sistema software para satisfacer unos requisitos especificados.



Niveles de diseño

■ Diseño de la arquitectura.

- Proceso de definición de la colección de componentes del sistema y sus interfaces.
- Objeto: determinar el marco de referencia que guiará la construcción del sistema.

■ Diseño detallado.

- Proceso de descripción detallada de la lógica de cada uno de los módulos, de las estructuras de datos que utilizan y de los flujos de control.
- Objeto: describir el sistema con el grado de detalle suficiente y necesario para su posterior implementación.⁵²

Tareas

■ Diseño de la arquitectura:

- ✓ Definir los criterios de descomposición.
- ✓ Descomponer el sistema en módulos.
- ✓ Determinar las estructuras de datos.
- ✓ Diseñar las interfaces.
- ✓ Definir los flujos de control.

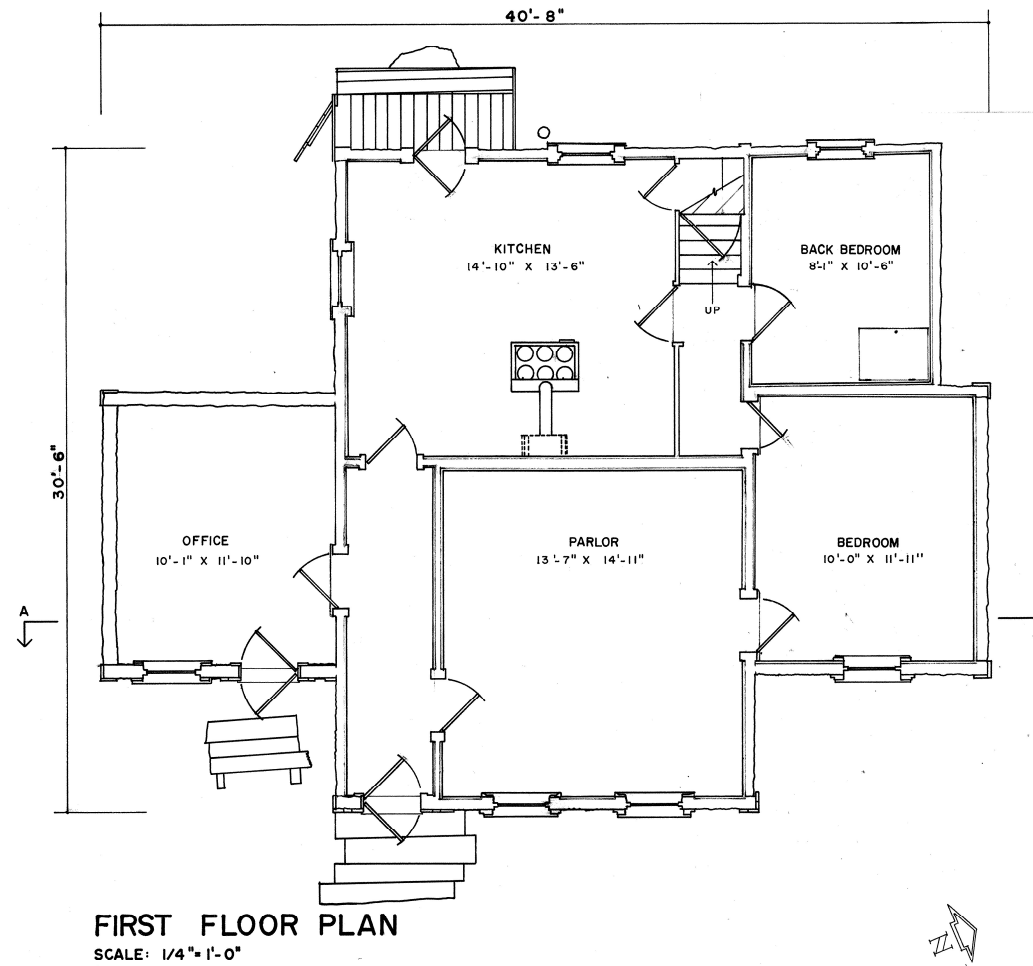


■ Diseño detallado.

- ✓ Descripción detallada de los módulos: estructuras y algoritmos.
- ✓ Descripción detallada de las estructuras físicas de datos.
- ✓ Descripción de los procedimientos de acceso a las estructuras físicas de datos.
- ✓ Descripción detallada de las interfaces.

Revisión

- ¿El plano de una casa sería una tarea de diseño de alto nivel o detallado?
- ¿El color de una puerta de la casa se debe considerar en el diseño de alto nivel?



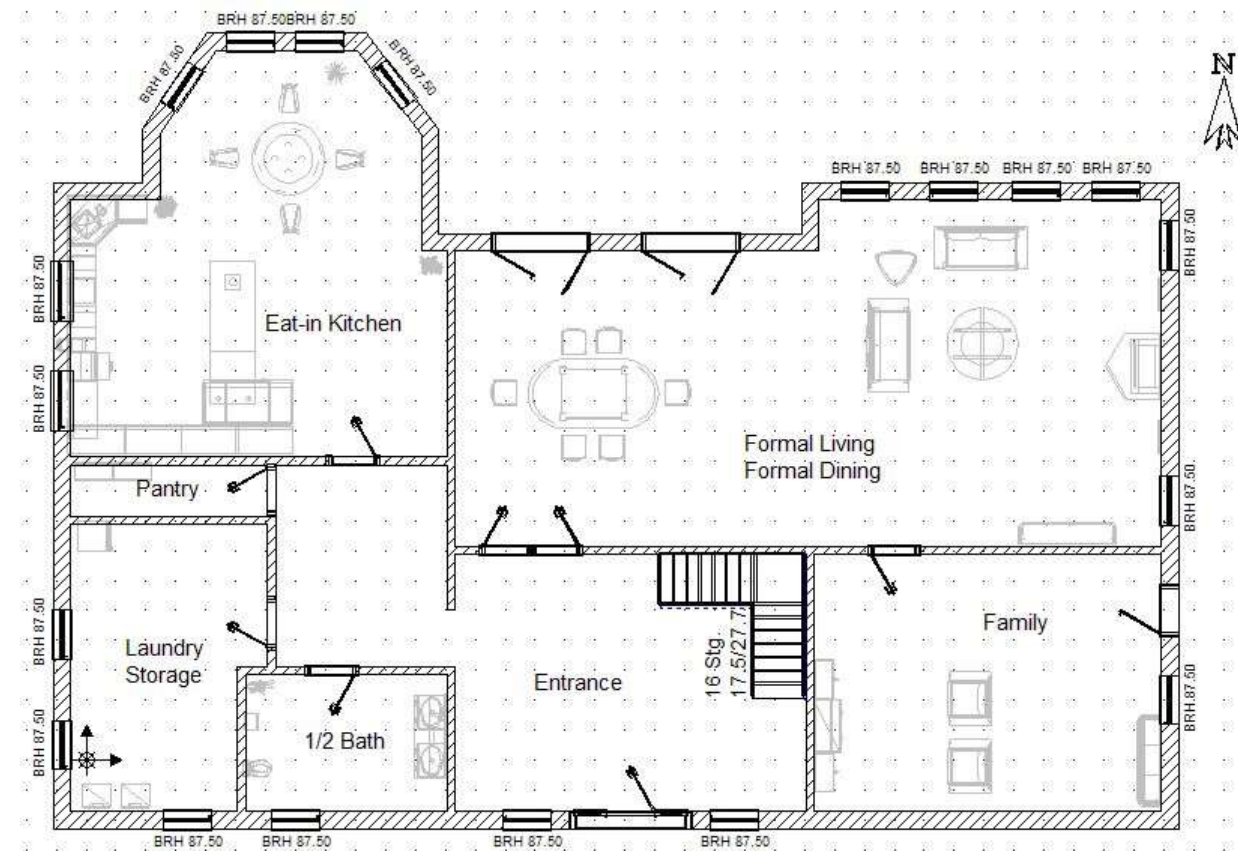
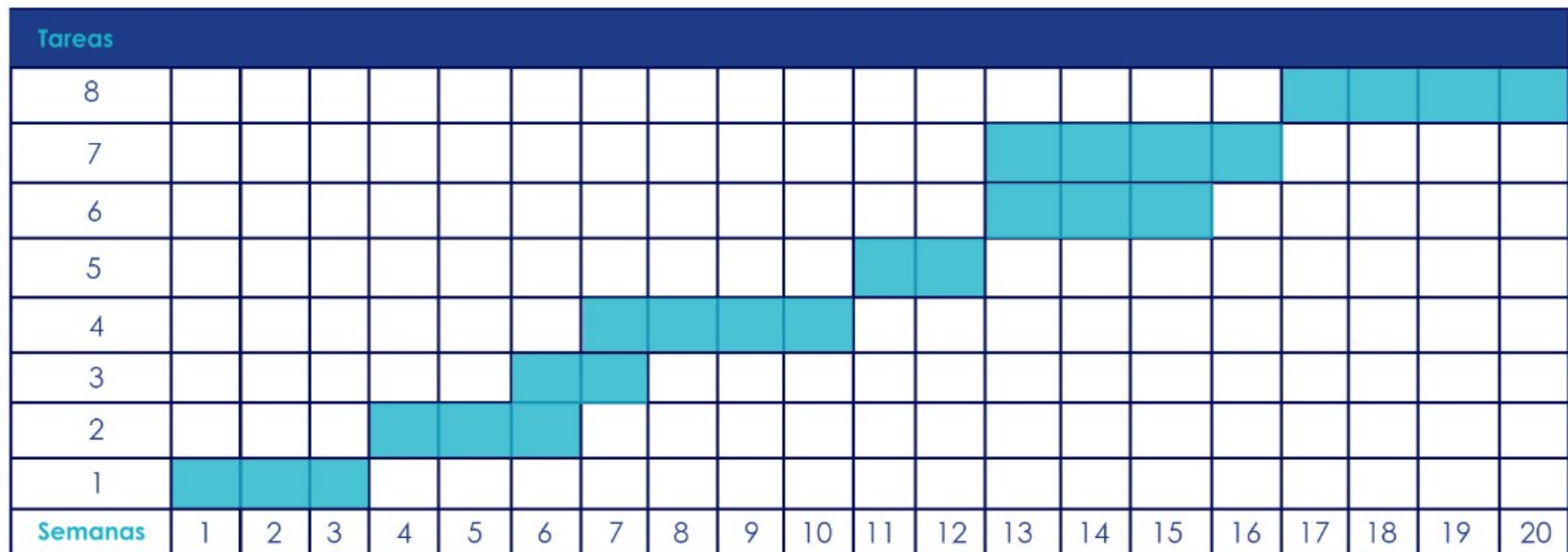


Diagrama de Gantt





Principios básicos (I)

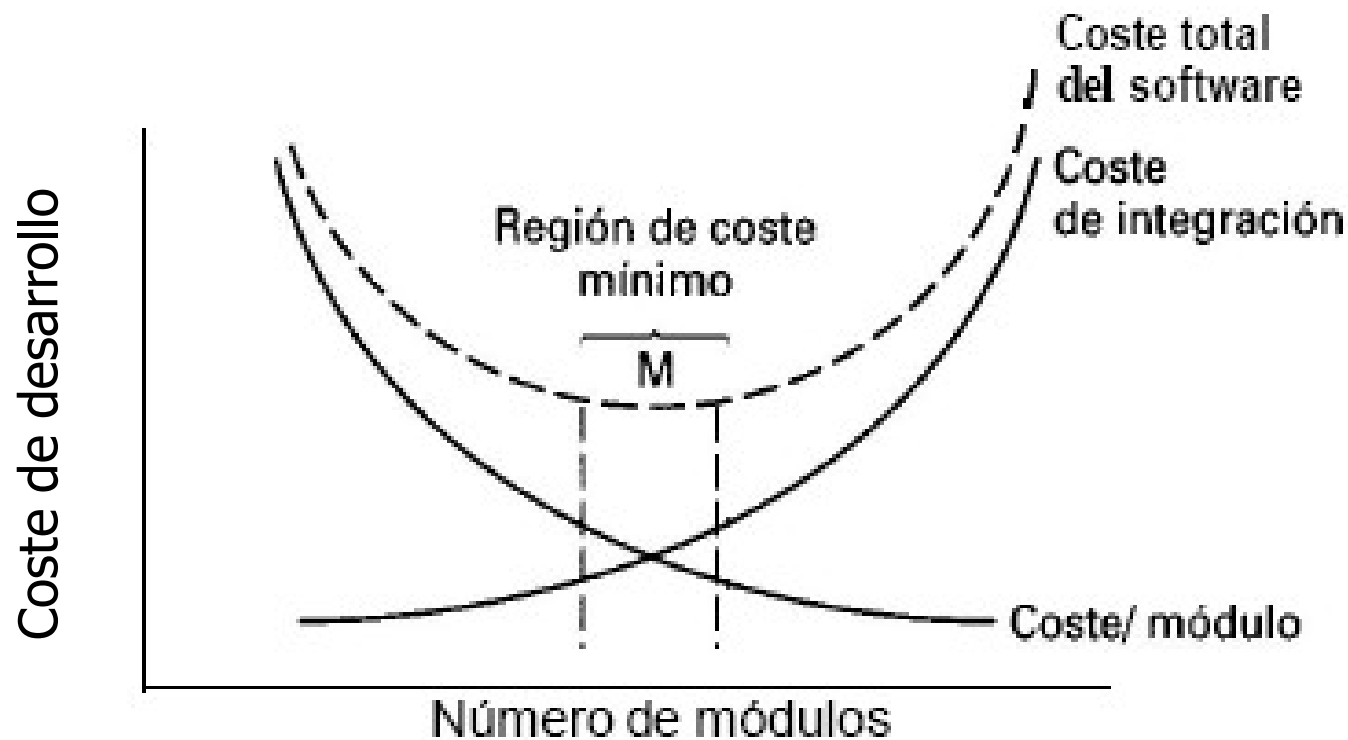
- **Abstracción.**
Manejo de conceptos generales y no de instancias particulares.
- **Refinamiento.**
Seguir una estrategia de diseño descendente.
- **Modularidad.**
División del software en unidades con entidad propia tales como funciones o subrutinas.
- **Ocultación de la información.**
Los detalles internos de cada módulo han de ser transparentes a los demás. Cada módulo debe formarse por especificaciones de diseño que sean independientes del resto de los módulos.

Principios básicos (II)

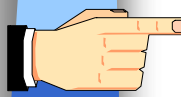
- La complejidad puede reducirse descomponiendo el problema en piezas cada vez más pequeñas, siempre que éstas sean relativamente independientes.
- A partir de un momento dado, la complejidad aumenta a causa de la interdependencia de las piezas.

Principios básicos (III)

¿Hasta dónde debe llegar la descomposición de un sistema?



Contenido

- 
- Introducción.
 - Diseño estructurado y Diseño orientado a objetos con UML.
 - Métricas de diseño.
 - Guías de un buen diseño.
 - Principales errores en el diseño.
 - Documentación final del diseño.

Métodos de diseño

- Diseño estructurado.

Se considera que el sistema es un conjunto de módulos o unidades, cada uno con una función bien definida, organizados de forma jerárquica.

- Diseño orientado a objetos.

Se considera que el sistema es una colección de objetos que interactúan a través de mensajes. Cada objeto tiene su propio estado y conjunto de operaciones asociadas.

Análisis y Diseño Orientado a Objetos con UML

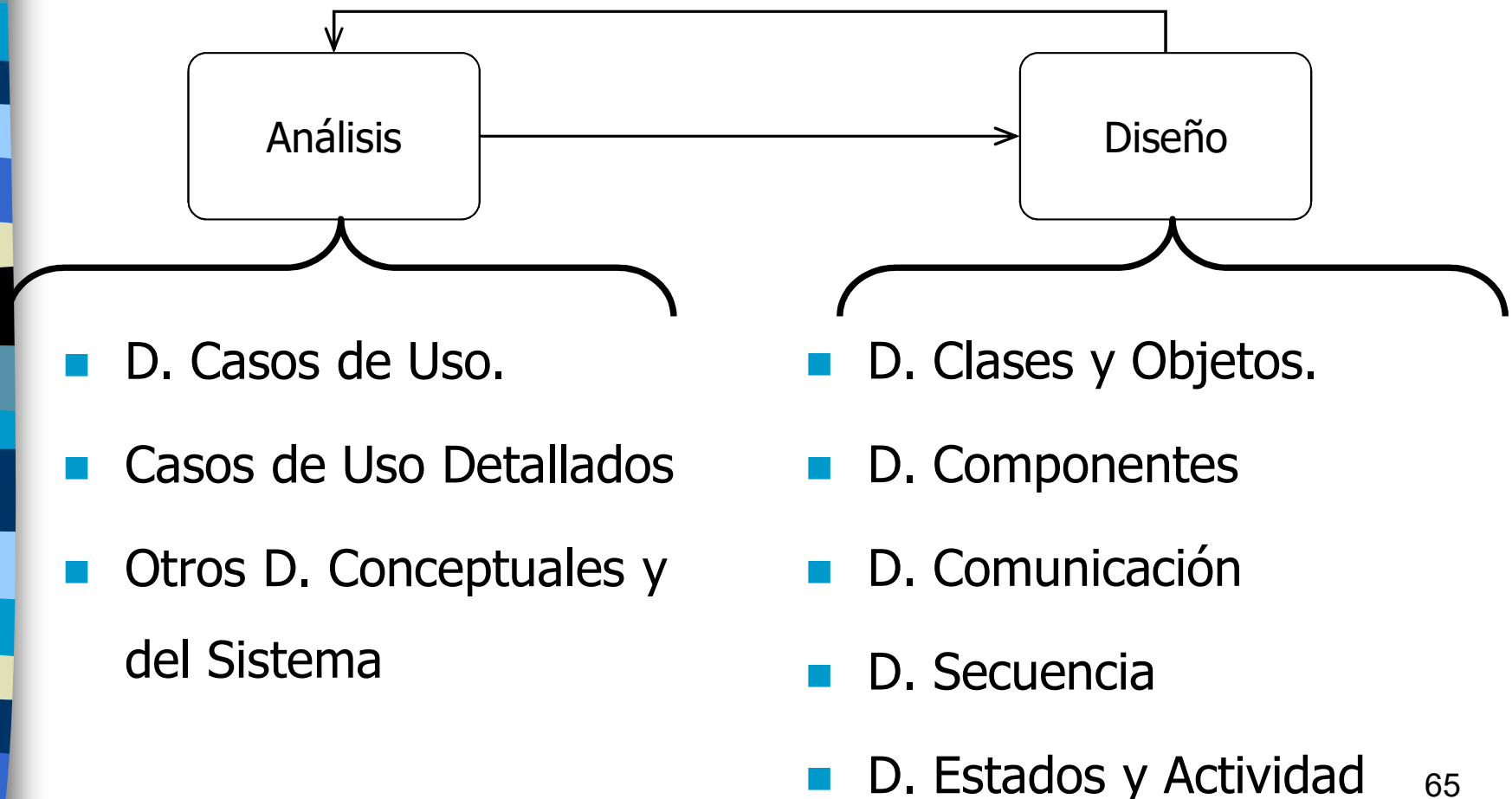
Diagramas Estructurales

- D. de Clases
- D. de Objetos
- D. de Componentes
- D. de Despliegue
- D. de Paquetes

Diagramas de Comportamiento

- D. Casos de Uso
- D. de Secuencia
- D. de Comunicación
- D. de Estados
- D. de Actividad

Análisis y Diseño Orientado a Objetos con UML



Contenido

- Introducción.
- Diseño estructurado y Diseño orientado a objetos con UML.
- Métricas de diseño.
 - ☐ Cohesión.
 - ☐ Acoplamiento.
- Guías de un buen diseño.
- Principales errores en el diseño.
- Documentación final del diseño.

Métricas de diseño (I)

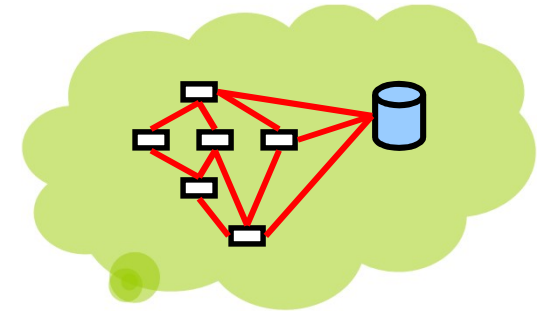
■ Cohesión.

- Es una medida de la relación (funcional) de los elementos de un módulo.
- Es mejor un alto grado de cohesión:
 - Menor coste de programación.
 - Mayor calidad del producto.



Métricas de diseño (II)

■ Acoplamiento.



- Es una medida de la interconexión entre los módulos de un programa.
- Es mejor un bajo acoplamiento:
 - Minimizar el efecto onda (propagación de errores).
 - Minimizar el riesgo al cambiar un módulo por otro.
 - Facilitar el entendimiento.

Métricas de diseño: Cohesión y acoplamiento (III)

- Cohesión:

“Un módulo con un alto grado de cohesión hace (idealmente) una sola cosa”

- Acoplamiento:

“Hacer los módulos tan independientes como sea posible”

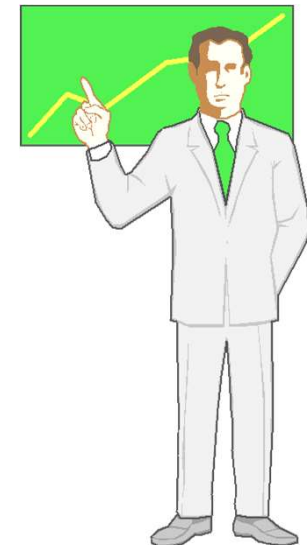
Contenido

- Introducción.
- Diseño estructurado y Diseño orientado a objetos con UML.
- Métricas de diseño.
- Guías de un buen diseño.
- Principales errores en el diseño.
- Documentación final del diseño.



Guías de un buen diseño (I)

“Dividir los módulos de forma que éstos sean tan independientes como sea posible (mínimo acoplamiento) y que cada módulo lleve a cabo una sola función (máxima cohesión)”



Guías de un buen diseño (II)

- **Estructura flexible.**

Facilita cambios para acomodar nuevas necesidades.

- **Extensible.**

Estructura abierta.

- **Portable.**

- **Reusable.**



Contenido

- Introducción.
- Diseño estructurado y Diseño orientado a objetos con UML.
- Métricas de diseño.
- Guías de un buen diseño.
- Principales errores en el diseño.
- Documentación final del diseño.

Principales errores en el diseño

- Prisa por comenzar a programar.

“Cuanto antes se comience a escribir código, más tarde se acabará el programa”.
- Falta de detalle en las especificaciones de diseño.
- No documentar el diseño.
- No tener en cuenta el entorno físico.

Contenido

- Introducción.
- Diseño estructurado y Diseño orientado a objetos con UML.
- Métricas de diseño.
- Guías de un buen diseño.
- Principales errores en el diseño.
- Documentación final del diseño.

Documento final: Diseño del Software (I)

1. Introducción.

- 1.1. Propósito del documento.
- 1.2. Entorno hardware y software.
- 1.3. Principales funciones del software.
- 1.4. Bases de datos externas.
- 1.5. Restricciones y limitaciones.
- 1.6. Referencias.

2. Descripción del diseño.

- 2.1. Diagramas.
- 2.2. Descripción de datos.
- 2.3. Descripción de interfaces.
- 2.4. Descripción de comunicaciones.

Documento final: Diseño del Software (II)

3. Descripción de los módulos (para cada módulo).

- 3.1. Descripción.
- 3.2. Descripción de la interfaz.
- 3.3. Módulos relacionados.
- 3.4. Organización de los datos.

4. Descripción de archivos externos y datos globales.

- 4.1. Descripción.
- 4.2. Métodos de acceso.

Documento final: Diseño del Software (III)

5. Especificaciones de programas.

6. Referencias cruzadas con los requisitos.

7. Plan de pruebas.

7.1. Estrategia de integración.

7.2. Estrategia de pruebas.

7.3. Consideraciones especiales.