Unidad 4

Análisis

Ingeniería del Software

Universidad Autónoma de Madrid

Escuela Politécnica Superior

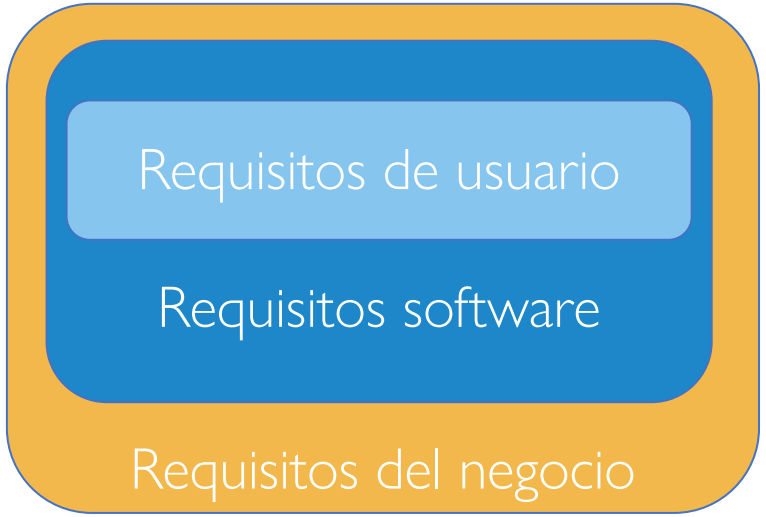
Contenido

- Definiciones, Importancia y Actividades
- Tareas
- Tipos de Requisitos
- Representación de Requisitos
- Validación de Requisitos
- Principales errores del análisis
- Documentación final: ERS

Definición, Importancia y Actividades

Definición: Análisis de requisitos

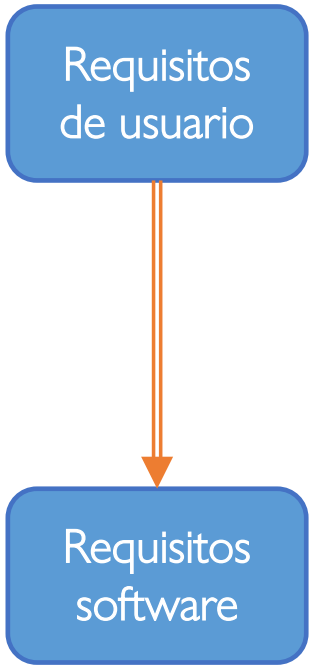
Análisis del problema y especificación completa del comportamiento externo que se espera del sistema software que se va a construir, así como de los flujos de información y control.



Requisitos de usuario y Requisitos software

Requisitos de usuario

- Son declaraciones 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).

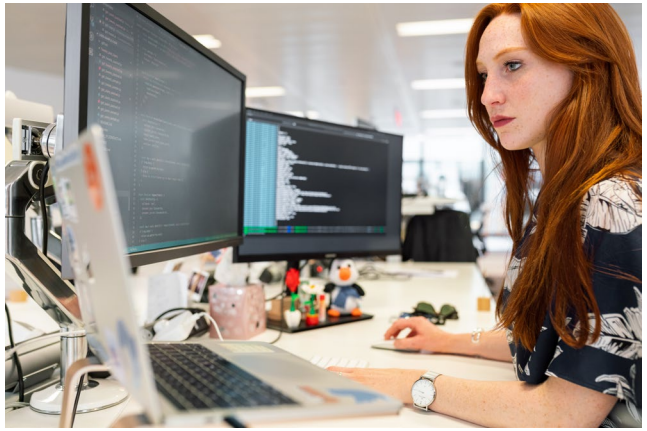


Requisitos software

- 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?

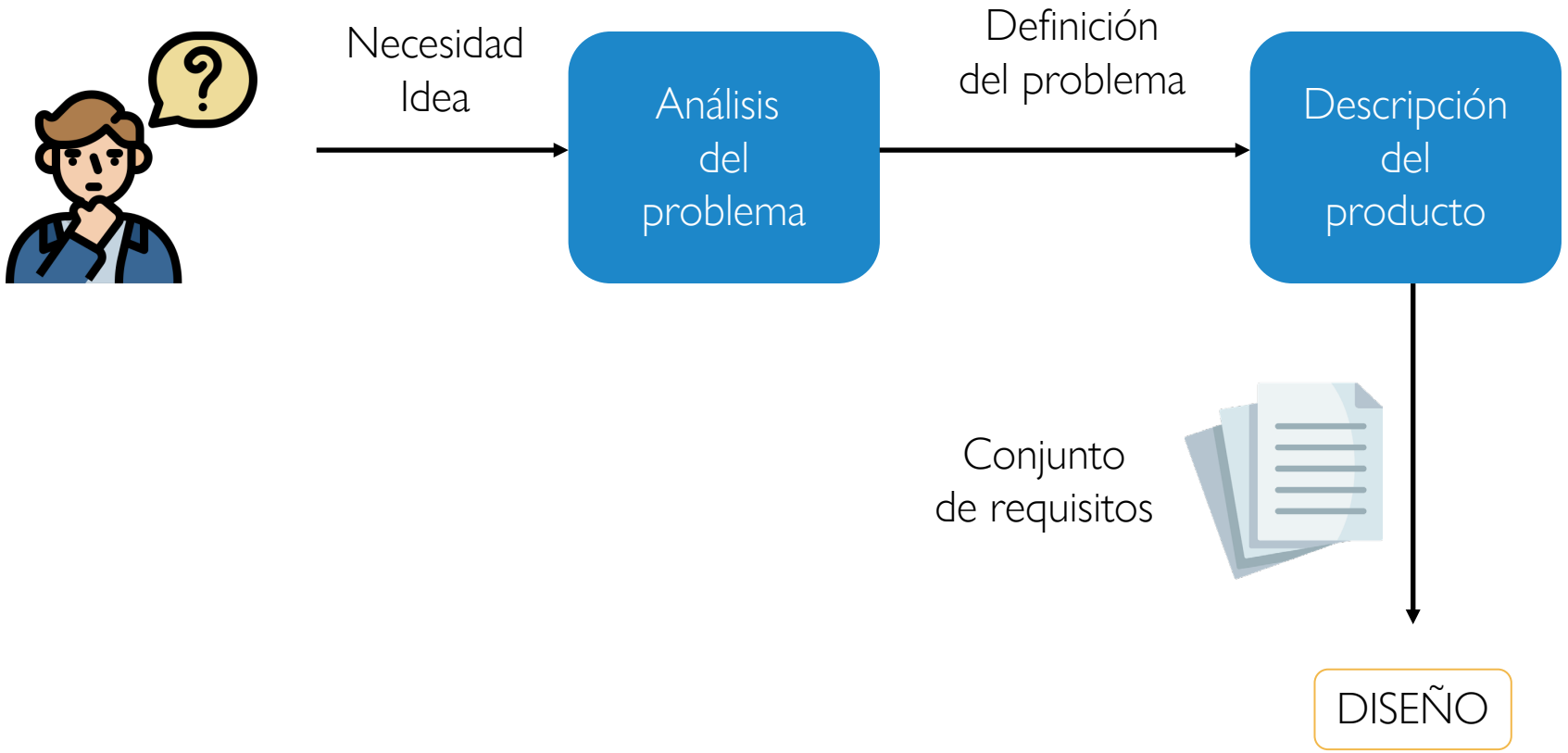
Papeles

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

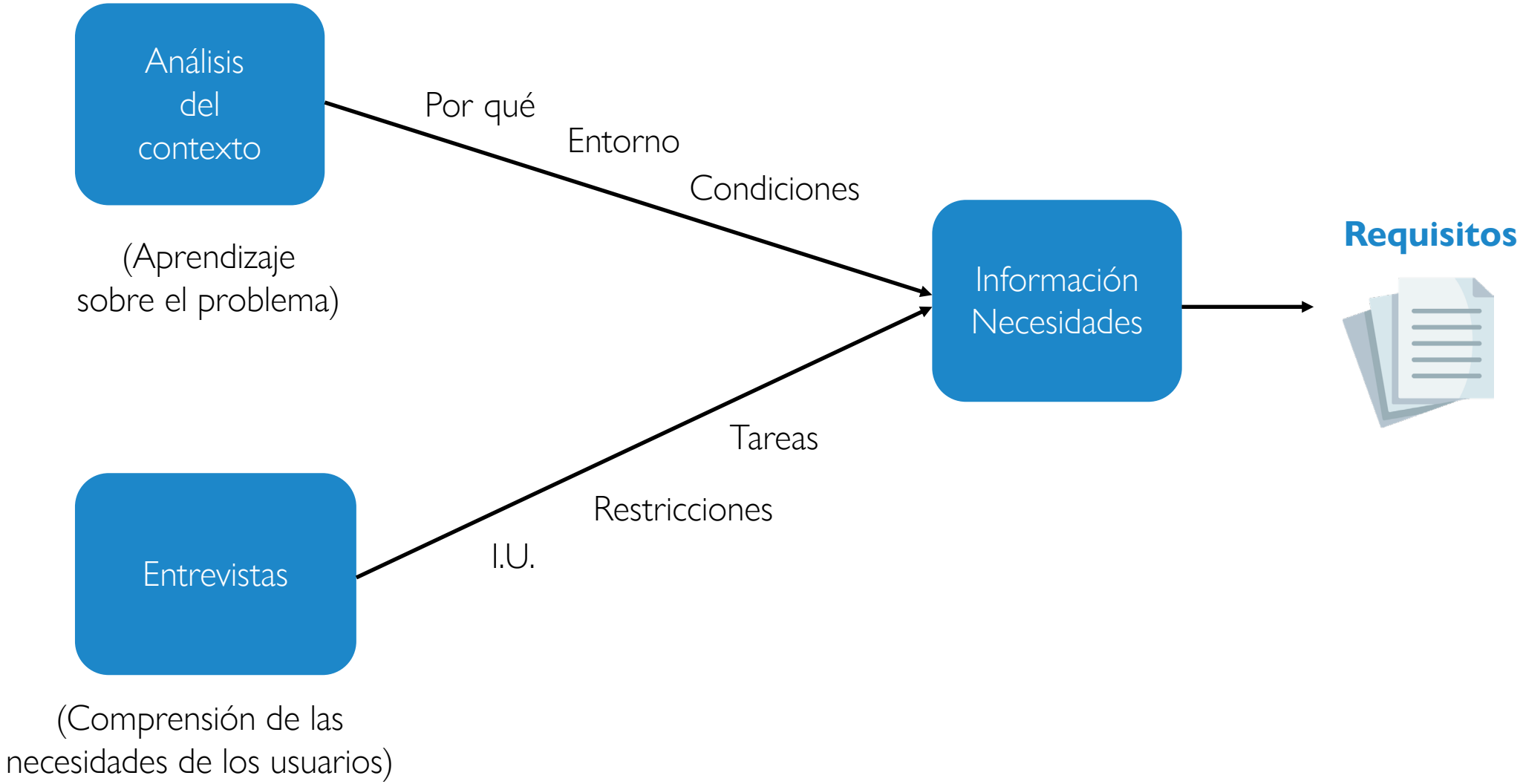


Los **ingenieros del software** preguntan, analizan, asimilan y presentan la solución adecuada.

Actividades Principales



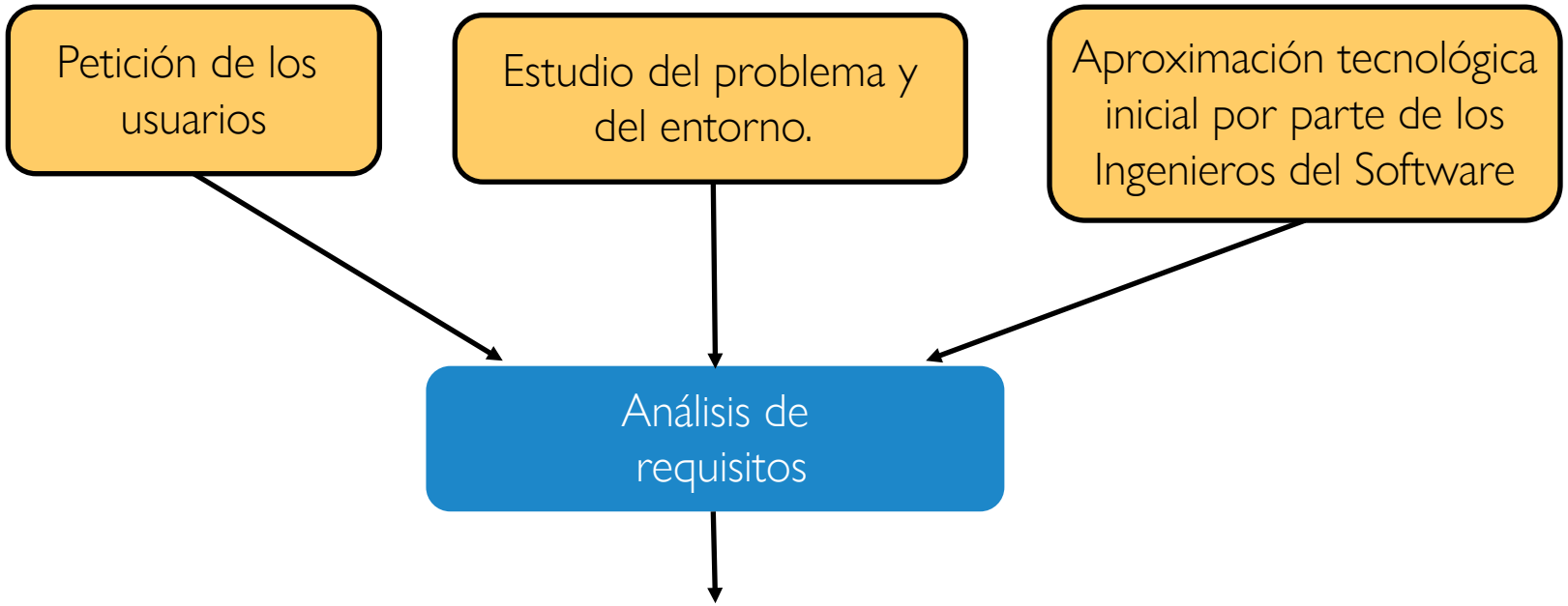
Fuentes y Técnicas



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
- Sirve como base para las actividades de prueba y validación

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

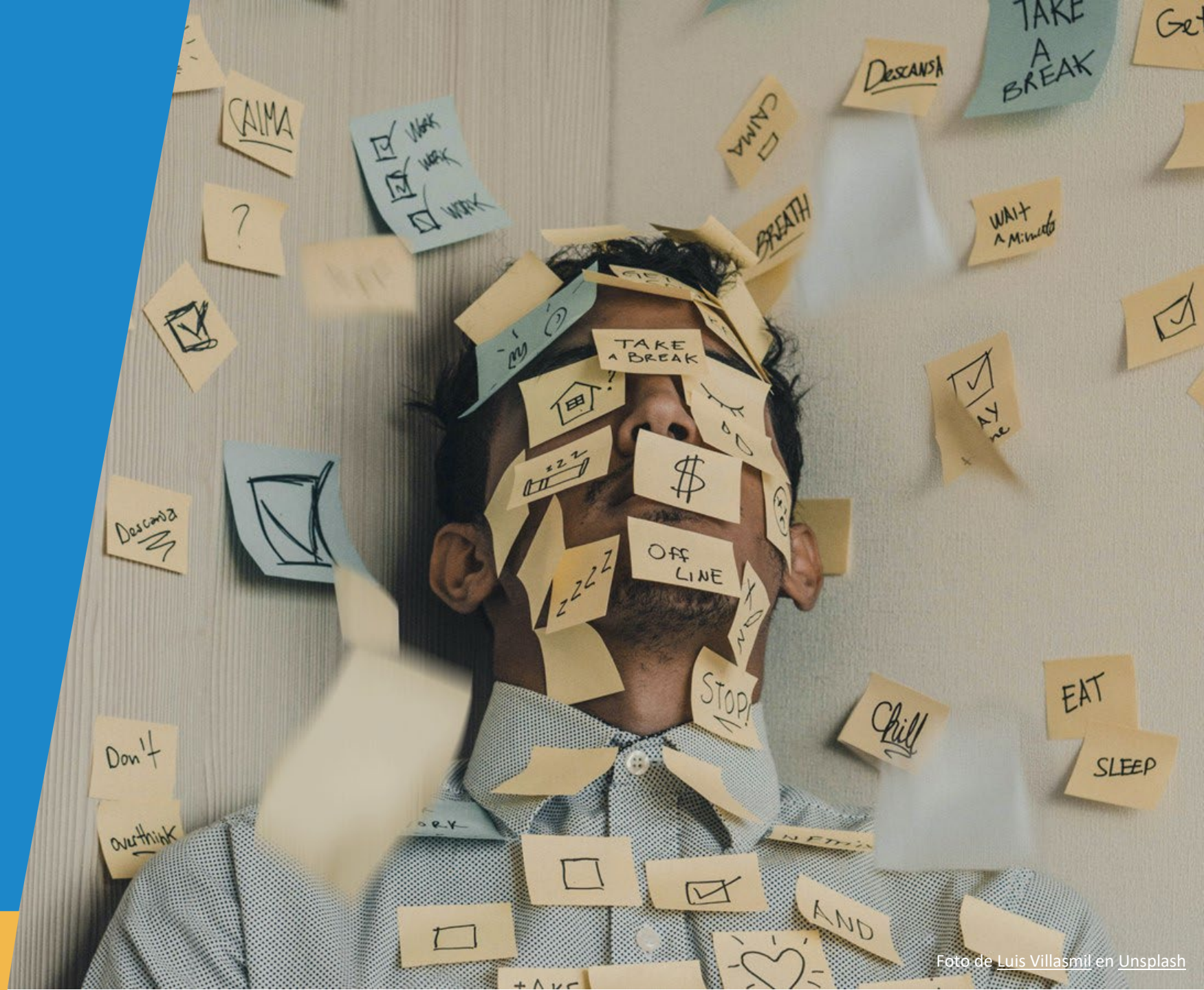


Foto de Luis Villasmil en Unsplash

Tareas

- **Análisis del problema**
- **Análisis de Usuarios y de las Tareas**: Identificar potenciales usuarios del sistema, su jerarquía y las tareas a realizar.
- **Educción de requisitos**: Identificar los requisitos que se obtienen de los usuarios y clientes.
- **Análisis 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.

Educción de requisitos

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

Mediante reuniones, entrevistas, análisis de las tareas, lectura de documentos o manuales, etc.

Tipos de requisitos

- Requisitos funcionales
- Requisitos no funcionales

Foto de Mika Baumeister en Unsplash

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”

Requisitos no funcionales

- Representan **características o cualidades generales** que se esperan del software para conseguir su propósito.
- Distintas clasificaciones

Requisitos no funcionales (I)

- **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.

Requisitos no funcionales (II)

- **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 no funcionales (III)

- **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.

Ejemplos de requisitos

- “El usuario podrá dar de alta una película.”
- “El sistema debe permitir a los usuarios consultar la información sobre las películas.”
- “El sistema no debe tardar más de 5 segundos en mostrar los resultados de una búsqueda.”
- “El sistema podrá ser utilizado en los sistemas operativos Windows y Linux.”
- “El sistema debe cumplir las disposiciones recogidas en el Reglamento General de Protección de Datos.”
- “El usuario puede modificar los datos de su perfil.”
- “El sistema debe soportar al menos 30 transacciones por segundo.”
- “El sistema asegurará que los datos son protegidos de accesos no autorizados.”
- “El sistema incluirá un procedimiento de autorización de usuario en el que los usuarios se identifican mediante un nombre de usuario y una contraseña.”

Representación de requisitos

Foto de Chinh Le Duc en Unsplash

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 (casos de uso, interacción)

Escenarios, Storyboards, Prototipos y Maquetas

Lenguajes formales

Modelos de desarrollo de productos software

- 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

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.

Modelos de desarrollo de productos software

La maqueta permite:

- Adaptar el modelo mental del usuario con el del ingeniero del software, permitiendo una mejor educción 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

Construir/revisar maqueta

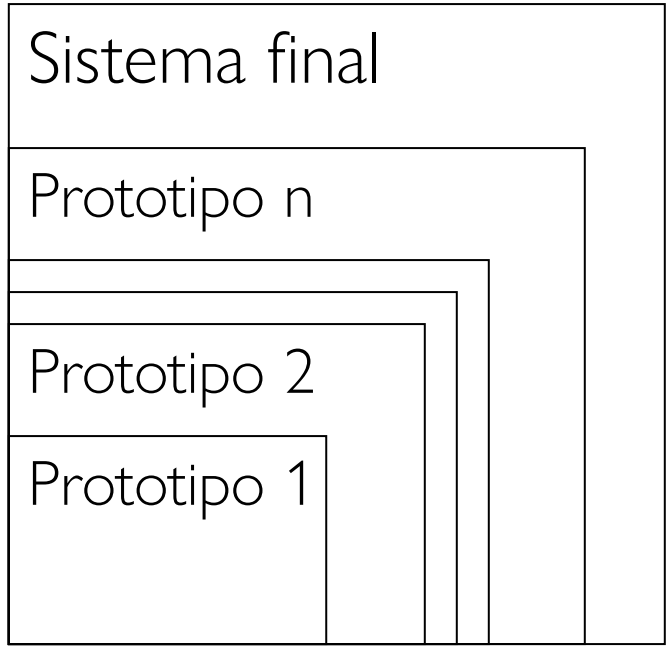
El cliente prueba la maqueta

Escuchar al cliente

Modelos de desarrollo de productos software

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

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.

Validación de requisitos

Foto de Nicola Styles en Unsplash

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.

Principales errores del análisis

Foto de Kenny Eliason en Unsplash

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.

Documentación final de análisis: ERS

Foto de ron dyar en Unsplash

Documento final: 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. Funcionalidades del producto
    2.3. Restricciones generales
    2.4. Dependencias
    2.5. Características de usuario

Documento final: ERS (III)

3. **Requisitos funcionales**

4. **Requisitos no funcionales**
   - 4.1. Requisitos de interfaz.
   - 4.2. Requisitos de rendimiento.
   - 4.3. Requisitos operacionales.
   - 4.4. Requisitos de recursos.
   - 4.5. Atributos del sistema software: fiabilidad, disponibilidad, seguridad, etc.

Documento final: ERS (IV)

5. **Descripción funcional**
    - 5.1. Flujo de datos
    - 5.2. Flujo de control
    - 5.3. Partición funcional
    - 5.4. 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