Tema 4

Análisis de Requisitos

Jorge Álvarez Fidalgo

Introducción

Abarca aquellas actividades que nos permiten conseguir un **entendimiento** más detallado y preciso de los requisitos:

- Analizar la información obtenida de los **usuarios**.
- **Detallar** adecuadamente requisitos de alto nivel.
- Estudio de los atributos de **calidad**.
- **Derivar** requisitos funcionales a partir de otros requisitos.
- Definir **prioridades**.
- Identificar requisitos **innecesarios** o **incompletos**.

Refinado de Requisitos

Refinado de Requisitos

El objetivo consiste en **estudiar** y **detallar** los **requisitos de alto nivel** producidos en la Obtención para elaborar requisitos de **software**.

- Requisito de alto nivel:
  - El sistema generará una factura.
- Requisitos de software:
  - El sistema generará una factura por cada pedido realizado.
  - El sistema realizará el proceso de generación de facturas cada hora.
  - Las facturas generadas por el sistema contendrán la siguiente información: […]
  - Por cada factura generada, el sistema cambiará el estado del pedido asociado a "Facturado".
  - Por cada factura generada, el sistema creará un registro de cobro.
    - El estado del nuevo registro de cobro será "Pendiente de pago".

Modelado visual

- Empleamos **modelos visuales** para:
  - Representar el **dominio** del problema.
  - Representar conceptualmente el nuevo **sistema**.
- Representan:
  - Datos y objetos del dominio.
  - Transacciones y transformaciones.
  - Cambios en el estado.
- Extrapolamos modelos a partir de los requisitos textuales...
  - ...o extraemos RF a partir de modelos de alto nivel.

Modelado visual

- El uso de modelos visuales facilita la **detección** de errores y omisiones.
  - Contrastar grandes listados textuales resulta inadecuado para las capacidades humanas.
- La **complejidad** de un sistema no es impedimento para modelar.
  - Si somos incapaces de modelar un sistema complejo, implementarlo será casi imposible.
- Los modelos generados en esta fase podrían usarse como modelos de hombre de paja en futuros proyectos.

Modelos visuales

- Diagramas de casos de uso.
- Diagramas de actividad.
- Diagramas de flujo de datos.
- Diagramas *swimlane*.
- Diagramas de estado.
- Mapas de diálogo.
- Árboles de decisión.
- Diagramas entidad-relación.
- Diagramas de clase.

Casos de uso

- **Caso de uso**: secuencia de interacciones entre un actor externo y un sistema que resulta en la obtención de un valor para dicho actor.

- Siempre siguen la estructura verbo + objeto.
  - *Comprar billete, elegir asientos, facturar equipaje...*

- A partir de un caso de uso, se pueden extrapolar requisitos funcionales y sus correspondientes pruebas.

Casos de uso: actores

- **No confundir** usuarios y actores.
  - Un mismo usuario puede jugar el rol de distintos actores.
  - *Recordad: un actor es una abstracción que define un participante en el caso de uso.*
- Para **identificar** actores, tened presente:
  - ¿A quién se notifica un evento del sistema?
  - ¿Quién provee servicios o información al sistema?
  - ¿Quién ayuda al sistema a completar una tarea?
- Al igual que con los usuarios, no tienen por qué ser humanos.

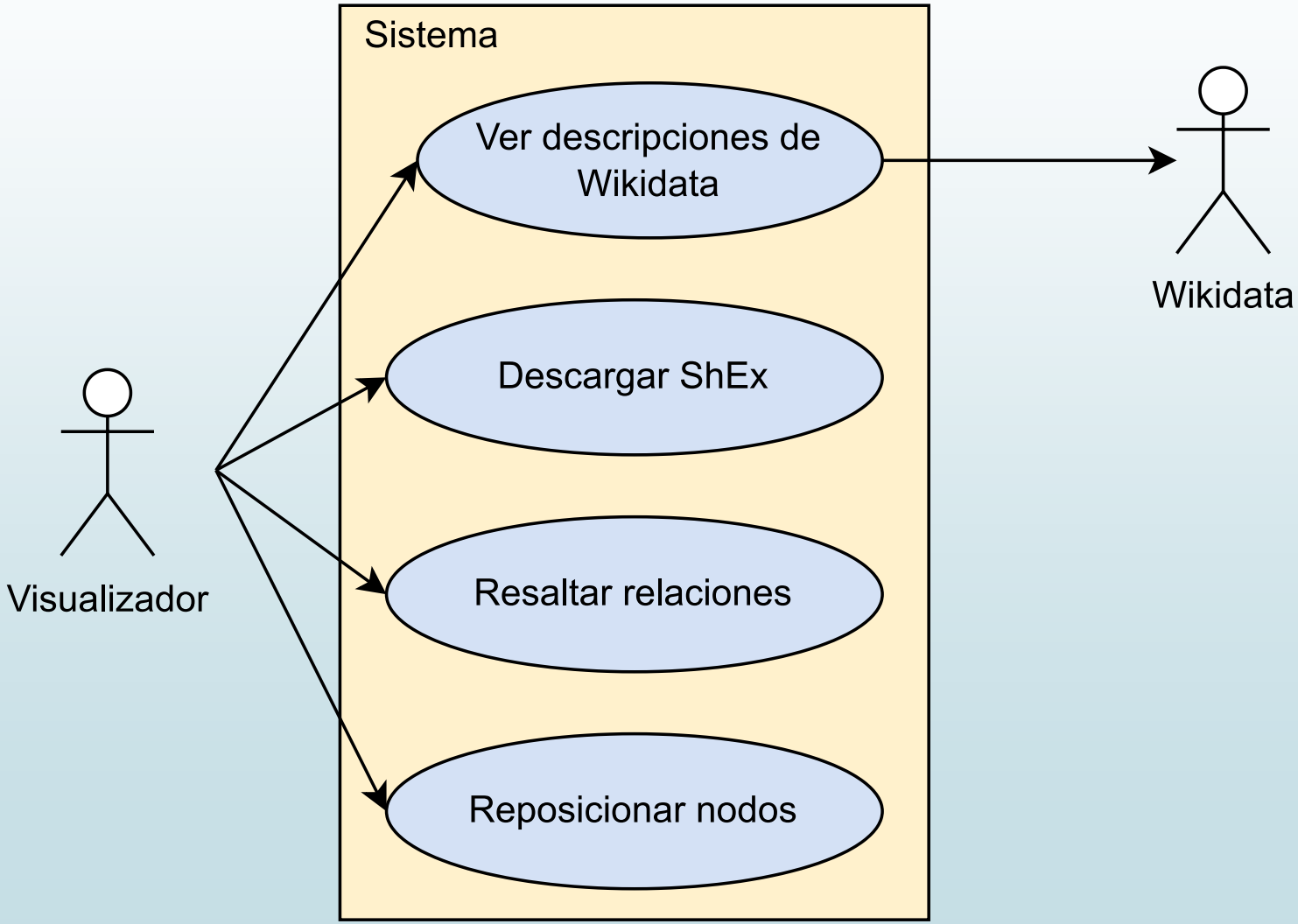
Diagramas de casos de uso

- Se emplean **diagramas** para representar los casos de uso.

- Hacemos uso de una notación basada en **UML**. *
  - Una caja representa los límites del sistema.
  - Los casos de uso se representan mediante óvalos.
  - Las flechas entre actores y casos de uso representan las interacciones.
    - Actor -> caso de uso: actor primario del caso, lo inicia y obtiene valor.
    - Caso de uso -> actor: actor secundario, interviene de algún modo.

[UML for IT Business Analyst: A practical guide, H. Podeswa]

Diagramas de casos de uso



Diagramas de casos de uso

- Los diagramas de casos de uso pueden incluir las siguientes relaciones:
  - **Extensión**: un caso de uso añade pasos a otro caso de uso de primer nivel.
  - **Inclusión**: un caso de uso requiere a otro caso de uso que no puede existir por sí mismo y está duplicado en varios casos de uso.

Ejemplo: *nuestro sistema es un cajero automático. Un caso de uso es "Retirar efectivo". El caso de uso "Calcular Comisión" extiende a "Retirar Efectivo" dado un "punto de extensión" (condición): la operación se realiza desde una entidad distinta a la del usuario.*

Por otra parte, "Retirar Efectivo" incluiría el caso de uso "Iniciar sesión", ya que es un prerrequisito para ese y muchos otros casos de uso. Nótese que "Iniciar sesión" no puede existir por sí mismo ya que no aporta valor.

Casos de uso

- Un caso de uso ha de contener, como mínimo, la siguiente información:
  - **Identificador** único y **nombre** descriptivo,
  - **Descripción** textual.
  - Condición que **inicia** el caso de uso.
  - **Precondiciones** a satisfacer para empezar el caso de uso (0+).
  - **Postcondiciones** que describen el estado del sistema tras el caso de uso (1+).
  - **Flujo/s de eventos** que tiene lugar durante el caso de uso.

- Otras recomendaciones:
  - Fecha, autor, actores, prioridades, referencias, supuestos...

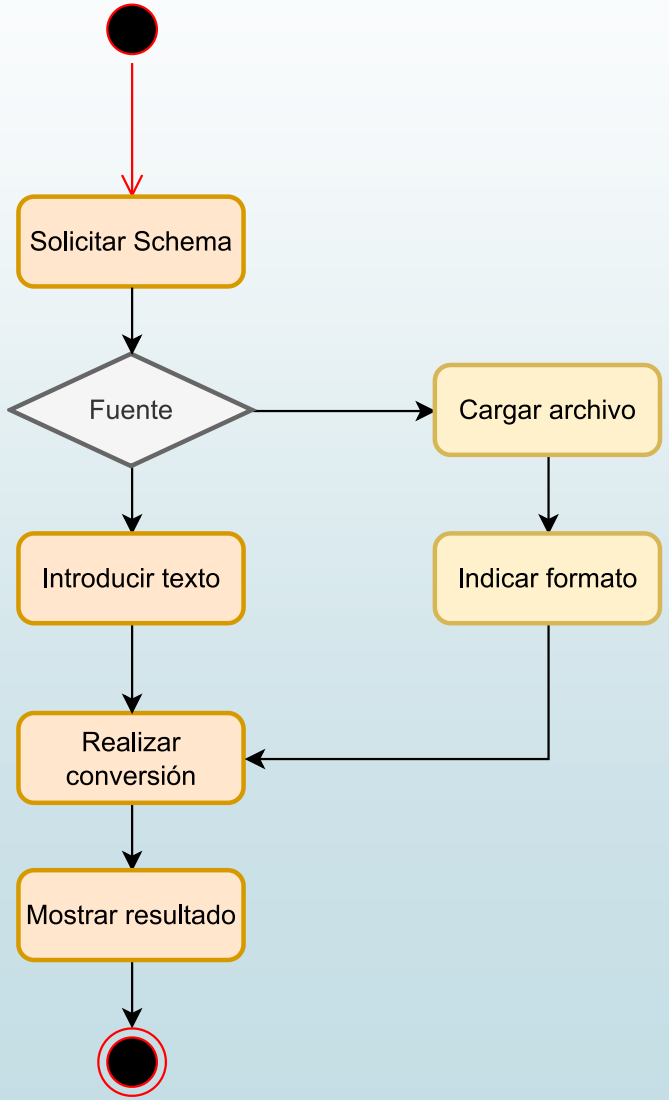
Casos de uso: condiciones

- **Precondiciones**. Requisitos a cumplir por el sistema previa ejecución del caso de uso.
  - La condición de inicio (*trigger*) no es una precondición.
  - *El cajero automático debe disponer de efectivo para realizar una retirada de efectivo.*
- **Postcondiciones**. Estado del sistema tras la ejecución del caso de uso.
  - Cambios observables por el usuario.
  - Salidas físicas.
  - Cambios internos en el sistema.
  - *El cajero automático debe actualizar su registro de efectivo tras una retirada.*
- Es posible que las postcondiciones de un caso de uso sean las precondiciones de otro.
  - Combinando casos de uso para formar una *macro*.

Casos de uso: flujos

- **Normales**. Representan la secuencia de acciones más habitual que produce un resultado satisfactorio. *(Escenario primario)*
- **Alternativos**. Representan secuencias de acciones menos comunes o relevantes que también producen un resultado satisfactorio. *(Esc. secundario)*
- **Excepciones**. Representan condiciones que pueden impedir un resultado satisfactorio.
  - Pueden ser recuperables o no.
  - Si no se especifican en esta fase:
    - El manejo de errores puede ser inconsistente.
    - El sistema puede fallar en determinadas condiciones.
  - Excepciones comunes (errores de conexión, de BD...) deberían extraerse a sus propios RF.

Casos de uso: d. de actividad



Casos de uso: escenarios

- Un **escenario** es una descripción de una **instancia** concreta de **uso** del sistema.
- Un caso de uso consta de **múltiples** escenarios relacionados...
  - ...y un escenario es una instancia específica de un caso de uso.
- De cara al análisis de requisitos, podemos:
  - Extrapolar escenarios específicos a partir de un caso de uso general.
  - Generalizar un escenario concreto para obtener un caso de uso general.

Casos de uso

ID y Nombre:	UC-33 Solicitar una miniatura
Creado por:	Seth
Fecha de creación:	06/09/2020
Actor Primario:	Solicitante
Actores Secundarios:	Impresor, Almacén
Descripción:	El Solicitante indica la miniatura deseada introduciendo su nombre, identificador numérico o seleccionándola del catálogo. El sistema ofrece al Solicitante obtenerla del Almacén si hay stock o permite al Solicitante imprimirla en 3D.
Condición inicio:	Solicitante indica que quiere solicitar una miniatura.
Precondiciones:	PRE-1. La identidad del usuario ha sido autenticada. PRE-2. La BD del inventario del almacén está en línea.
Postcondiciones:	POST-1. La solicitud es almacenada en el sistema. POST-2. La solicitud es enviada al Almacén o al Impresor.
Flujo normal:	**33.0 Solicitar una miniatura del Almacén.** 1. El Solicitante especifica la miniatura deseada. 2. El sistema lista las réplicas de la miniatura deseada que se encuentran en el Almacén. 3. El Solicitante selecciona una miniatura específica o solicita una orden de impresión (ver 4.1). 4. El Solicitante introduce otra información necesaria para completar la solicitud. 5. El sistema almacena la solicitud y notifica al Almacén.
Flujos alternativos:	**33.1 Solicitar una miniatura al Impresor.** 1. El Solicitante busca en los catálogos del Impresor la miniatura (ver 4.1.E1). 2. El sistema lista los tamaños y materiales disponibles para la impresión. 3. El Solicitante selecciona un tamaño, material y nº de miniaturas. 4. El Solicitante introduce otra información necesaria para completar la solicitud. 5. El sistema almacena la solicitud y notifica al Impresor.
Excepciones:	**33.1.E1 Miniatura no disponible.** 1. El sistema informa que no existe una miniatura de esas características en los catálogos. 2. El sistema pregunta al Solicitante si desea solicitar otra miniatura (3a) o salir (4a). 3a. El Solicitante pide solicitar otra miniatura. 3b. El sistema empieza de nuevo el flujo normal. 4a. El Solicitante pide salir. 4b. El sistema termina el caso de uso.
Prioridad:	Alta
Frecuencia de Uso:	Aproximadamente 30 veces a la semana.
Reglas de negocio:	BR-88
Otra información:	N/A
Supuestos:	Las miniaturas a imprimir están libres de derechos de autor.

Casos de uso: ejemplo

- **ID**: UC-33.
- **Nombre**: Solicitar una miniatura.
- **Creado por**: Seth.
- **Fecha de creación**: 06/09/2020.
- **Actor primario**: Solicitante.
- **Actores secundarios**: Impresor, Almacén.
- **Descripción**: El Solicitante indica la miniatura deseada introduciendo su nombre, identificador numérico o seleccionándola del catálogo. El sistema ofrece al Solicitante obtenerla del Almacén si hay stock o permite al Solicitante imprimirla en 3D.

Casos de uso: ejemplo

- **Condición de inicio**: Solicitante indica que quiere solicitar una miniatura.
- **Precondiciones**:
  - PRE-1. La identidad del usuario ha sido autenticada.
  - PRE-2. La BD del inventario del Almacén está en línea.
- **Postcondiciones**:
  - POST-1. La solicitud es almacenada en el sistema.
  - POST-2. La solicitud es enviada al Almacén o al Impresor.

Casos de uso: ejemplo

- **Flujo normal**: 33.0 Solicitar una miniatura del Almacén.
  1. El Solicitante especifica la miniatura deseada.
  2. El sistema lista las réplicas de la miniatura deseada que se encuentran en el Almacén.
  3. El Solicitante selecciona una miniatura específica o solicita una orden de impresión (ver 4.1).
  4. El Solicitante introduce otra información necesaria para completar la solicitud.
  5. El sistema almacena la solicitud y notifica al Almacén.

Casos de uso: ejemplo

- **Flujos alternativos**: 33.1 Solicitar una miniatura al Impresor.
  1. El Solicitante busca en los catálogos del Impresor la miniatura (ver 4.1.E1).
  2. El sistema lista los tamaños y materiales disponibles para la impresión.
  3. El Solicitante selecciona un tamaño, material y nº de miniaturas.
  4. El Solicitante introduce otra información necesaria para completar la solicitud.
  5. El sistema almacena la solicitud y notifica al Impresor.

Casos de uso: ejemplo

- **Excepciones:** 33.1.E1 Miniatura no disponible.
  1. El sistema informa que no existe una miniatura de esas características en los catálogos.
  2. El sistema pregunta al Solicitante si desea solicitar otra miniatura (3a) o salir (4a).

  3a. El Solicitante pide solicitar otra miniatura.

  3b. El sistema empieza de nuevo el flujo normal.

  4a. El Solicitante pide salir.

  4b. El sistema termina el caso de uso.

Casos de uso: ejemplo

- **Prioridad:** Alta.
- **Frecuencia de Uso**: Aproximadamente 30 veces a la semana.
- **Reglas de negocio**: BR-88.
- **Otra información**: N/A
- **Supuestos**: Las miniaturas a imprimir están libres de derechos de autor.

Casos de uso: nivel de detalle

- El nivel de detalle depende de la **relevancia** de cada caso.
- Podemos **limitarnos** a una descripción del objetivo del usuario y sus interacciones con el sistema, o sólo mencionar los flujos.
- Una especificación **completa** puede ser apropiada si:
  - Se trata de un sistema complejo cuyos fallos ocasionan gran riesgo.
  - Los casos de uso representan requisitos novedosos para los desarrolladores.
  - Los casos de uso son los requisitos más detallados que los desarrolladores dispondrán.
  - Existirá escasa interacción entre los usuarios y el equipo de desarrollo.
  - Se pretende elaborar casos de prueba basados en los requisitos de usuario.

Casos de uso: identificación

- Existen distintos enfoques:
  - 1) Identificar los actores, 2) listar los procesos de negocio, 3) buscar **interacciones** entre ambos para definir casos de uso.
  - Describir un **escenario** concreto para cada proceso de negocio, y generalizar casos de uso a partir de éstos.
  - Estudiar qué **tareas** debe realizar el sistema para completar un proceso de negocio; pueden ser casos de uso.
  - Identificar los **eventos** externos a los que debe responder el sistema y relacionarlos con los actores.
  - Identificar datos que requieran un **CRUD**.
  - Estudiar el rol de los **sistemas externos** que deben interaccionar con el sistema.

Casos de uso: elaboración

Pasos propuestos para la elaboración de casos de uso:

1. Identificar el **actor primario**.
2. Elaborar la **descripción**.
3. Definir **precondiciones** y **postcondiciones**. *Límites del caso*.
4. Definir el **flujo normal** de pasos.
5. Definir el **resto** de flujos.

Requisitos funcionales y casos de uso

- R**equisitos funcionales != casos de uso.**
  - Los casos de uso cubren únicamente el comportamiento "**externo"** del sistema.
- Debemos **especificar** los requisitos funcionales necesarios para implementar cada caso de uso.
  - *P.e.: ¿Qué hacer si una precondición no se satisface?*
- Mantener **trazabilidad** entre casos de uso y requisitos funcionales.

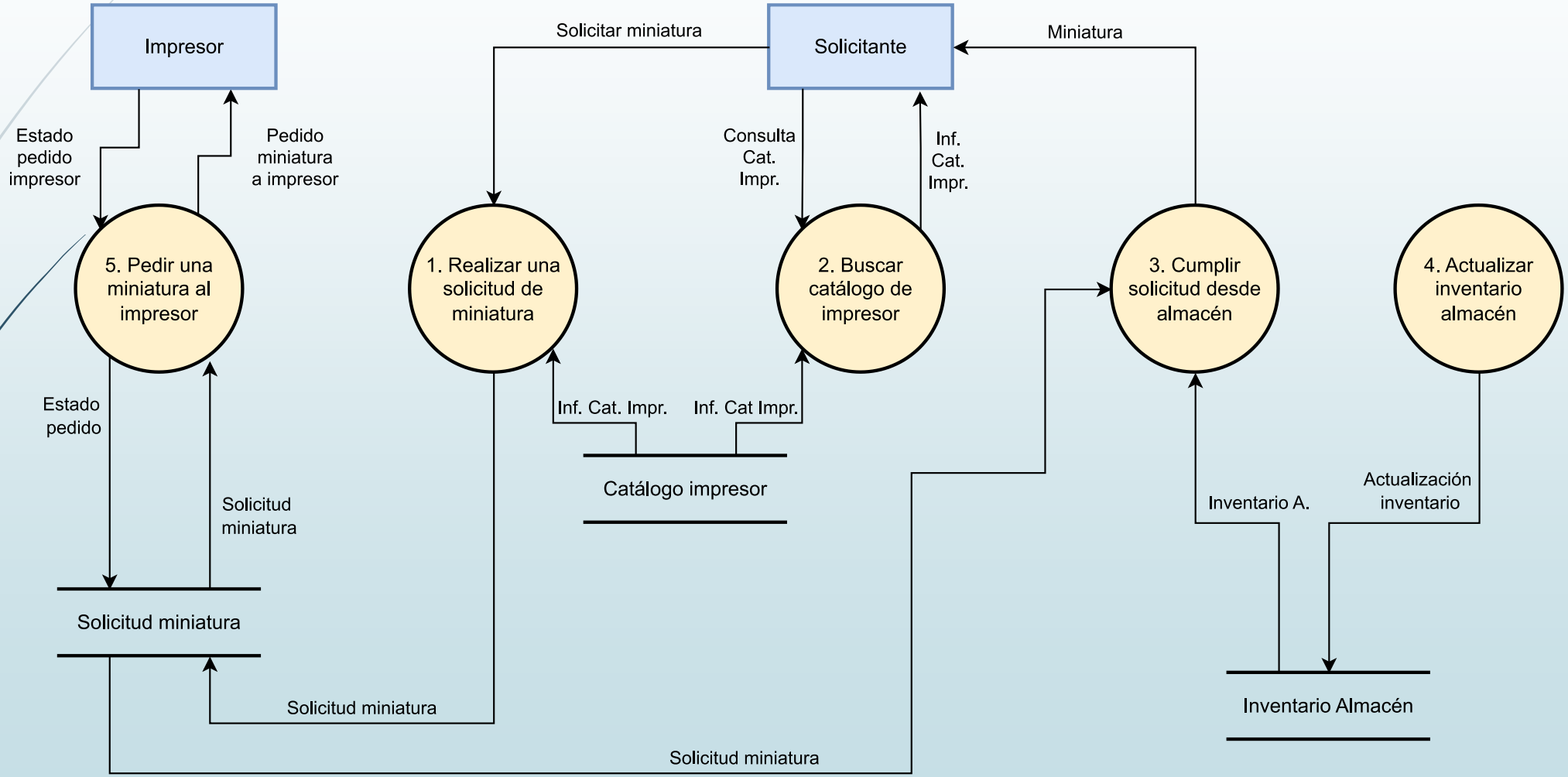
Casos de uso: problemas

- **Demasiados** casos de uso.
  - No es necesario cubrir todos los escenarios posibles.
  - Detallar los más relevantes.
- Casos de uso excesivamente **complejos**.
  - Si un flujo excede 15 pasos, es posible que incluya más de un escenario.
- Dependencia del **diseño**.
  - "El sistema muestra un *drop-down*" -> "El sistema presenta las opciones".
- Casos de usuario que los **usuarios no comprenden**.
  - Si un usuario no relaciona un caso de uso con sus objetivos, estamos perdiendo la perspectiva del usuario.

Diagramas de flujo de datos

- Adecuados para sistemas de procesamiento de **transacciones**.
- Describen el **movimiento de datos** en un sistema.
  - Comunicaciones, transformaciones, colecciones...
  - *Ejemplo: ¿cuántos datos requiere la creación de un pedido?*
- Útiles para extraer **requisitos de datos**.
- A menudo usados en entrevistas.

Diagramas de flujo de datos



Diagramas de flujo de datos

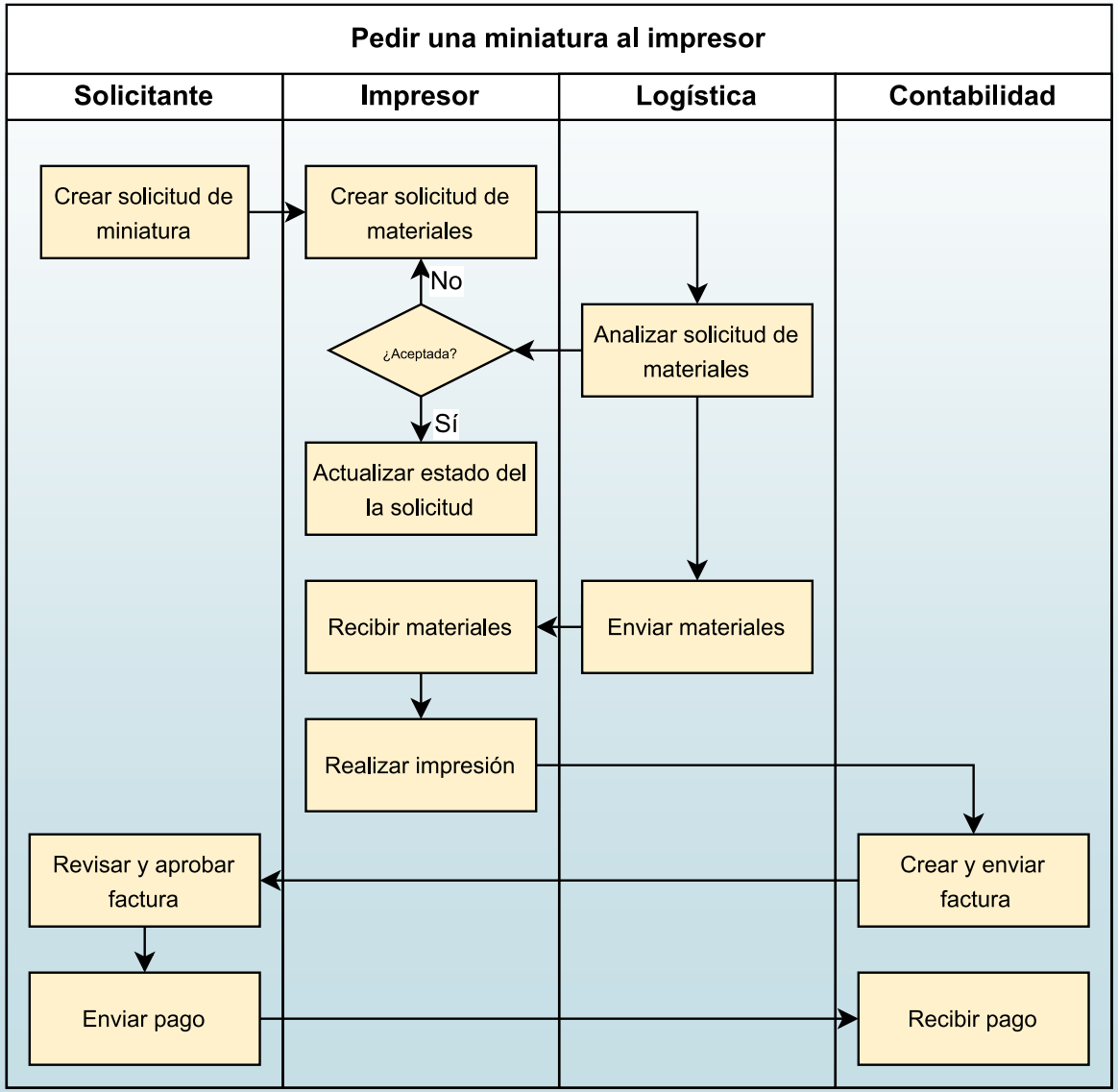
- Algunas **convenciones** a tener en cuenta:
  - Los datos no pueden comunicarse directamente entre entidades del mismo tipo.
  - El DFD no debería tratar de representar la secuencia del proceso.
  - Los procesos se identifican de manera única y dependiendo del nivel de detalle.
  - Los procesos se nombran de forma concisa.
  - Evitar 10+ procesos; abstraer procesos de mayor nivel si es necesario.
  - Los procesos deberían tener tanto entradas como salidas de datos.

Diagramas *swimlane*

- Representan las **operaciones** a realizar para ejecutar un proceso del sistema.

- Similares a los diagramas de actividad, pero subdivididos en **secciones** que representan a los **actores** que realizan una serie de pasos.

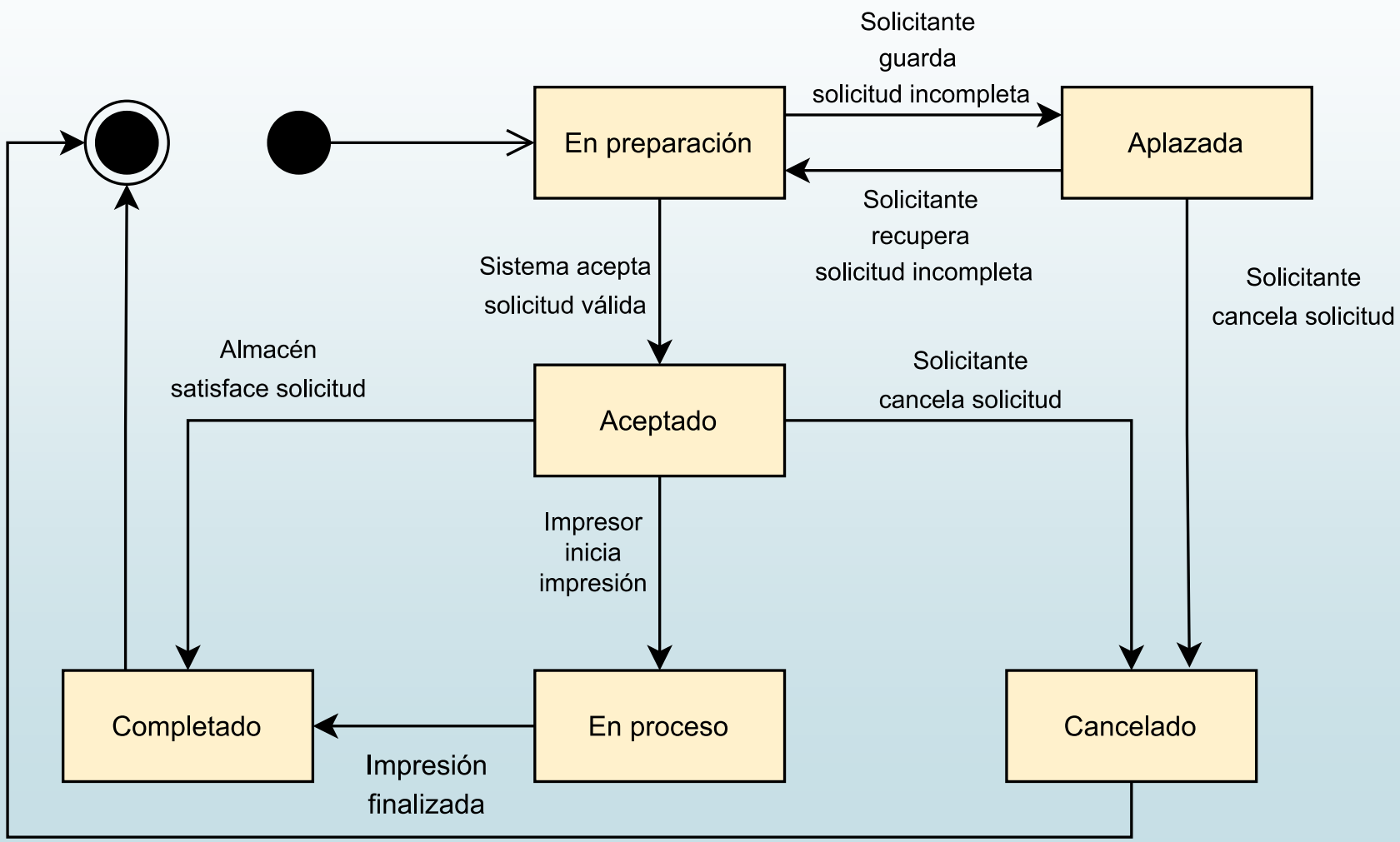
Diagramas *swimlane*



Diagramas de estado

- Describen los **cambios de estado** de un sistema bajo unas condiciones determinadas.
- Uso típico en sistemas de tiempo real y en la gestión de pedidos, inventarios y similares.
- Los cambios de estado se representan mediante flechas entre estados; los eventos que ocasionan dichos cambios se indican mediante etiquetas.
- Se suele representar alternativamente mediante una **tabla de estados**.

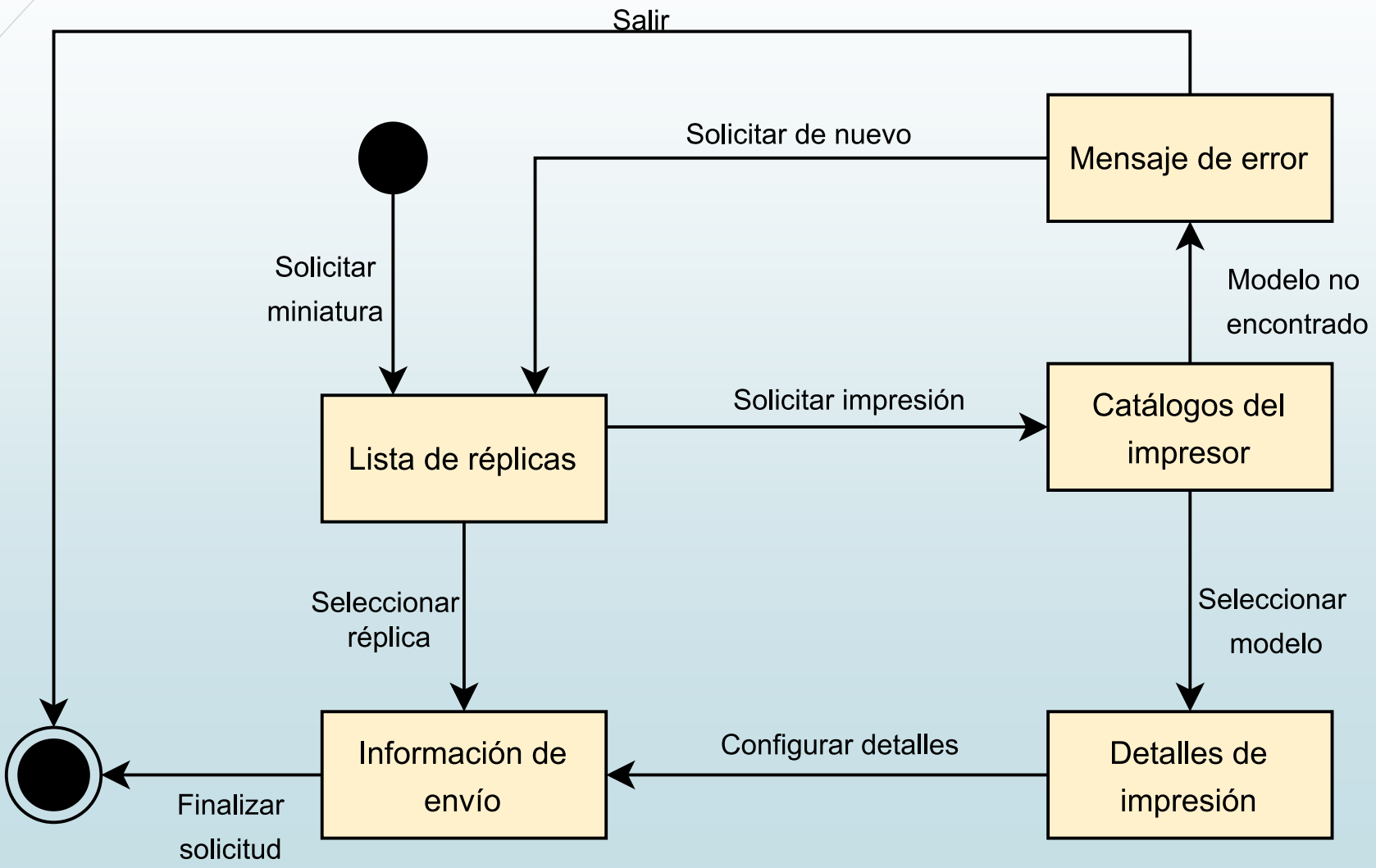
Diagramas de estado



Mapas de diálogo

- Describen la **interfaz de usuario** a un nivel de abstracción alto.
  - Diálogos y sus relaciones, sin entrar en cuestiones de diseño.
  - Puede considerarse un **diagrama de estados** de la interfaz de usuario.
- Permite explorar el flujo de **interacciones** entre usuarios y sistema sin entrar en detalles de los procesos o cuestiones de diseño de IU.
  - Son un buen complemento para los casos de uso.
- Las transiciones entre diálogos pueden deberse a:
  - Acciones del usuario.
  - Valores de datos.
  - Estados del sistema.

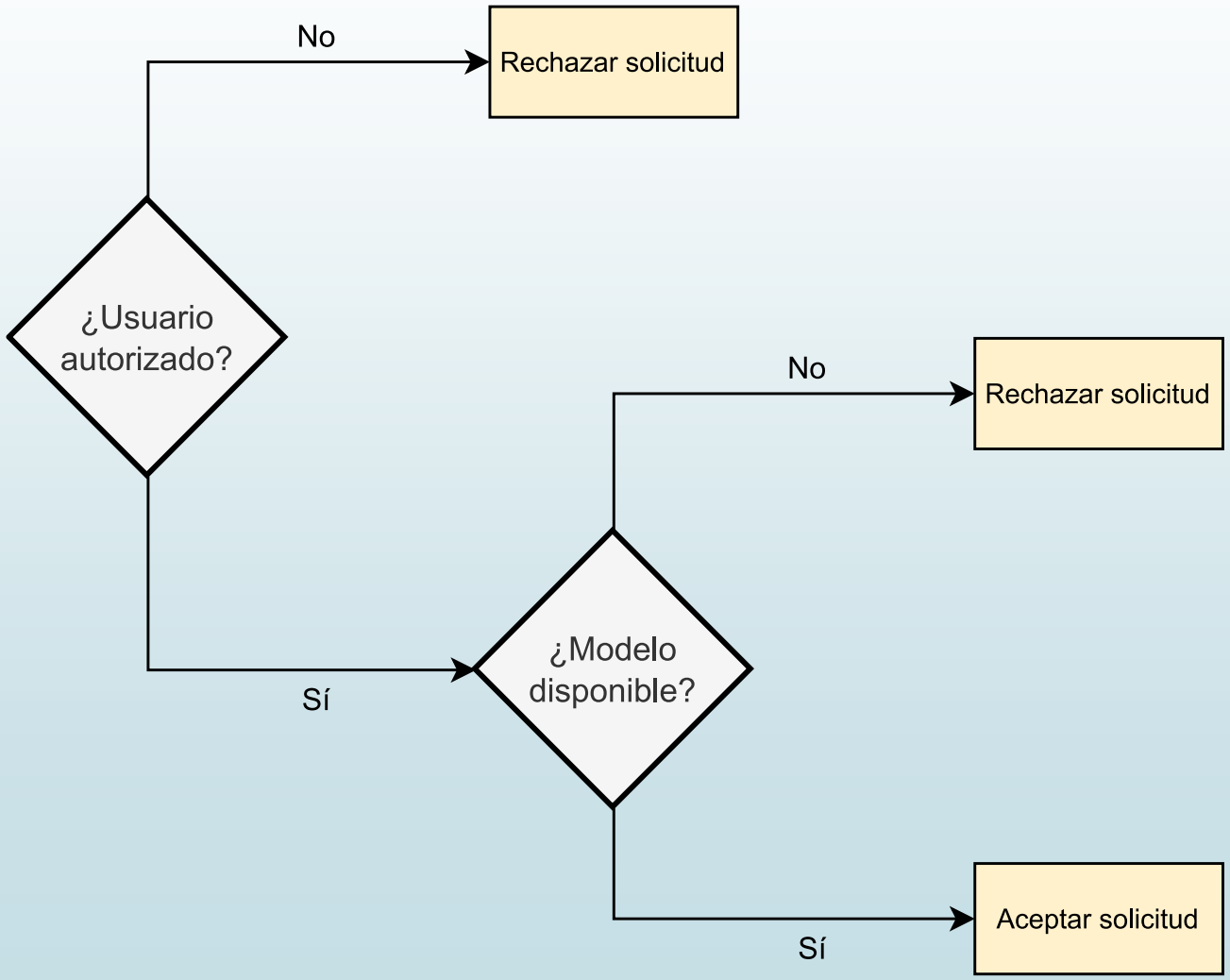
Mapas de diálogo



Árboles de decisión

- Representan la **secuencia lógica** de acciones a tomar por el sistema bajo unas **condiciones** determinadas.
  - *Toma de decisiones.*
- Adecuados para documentar **restricciones** de negocio.

Árboles de decisión



Diagramas E-R

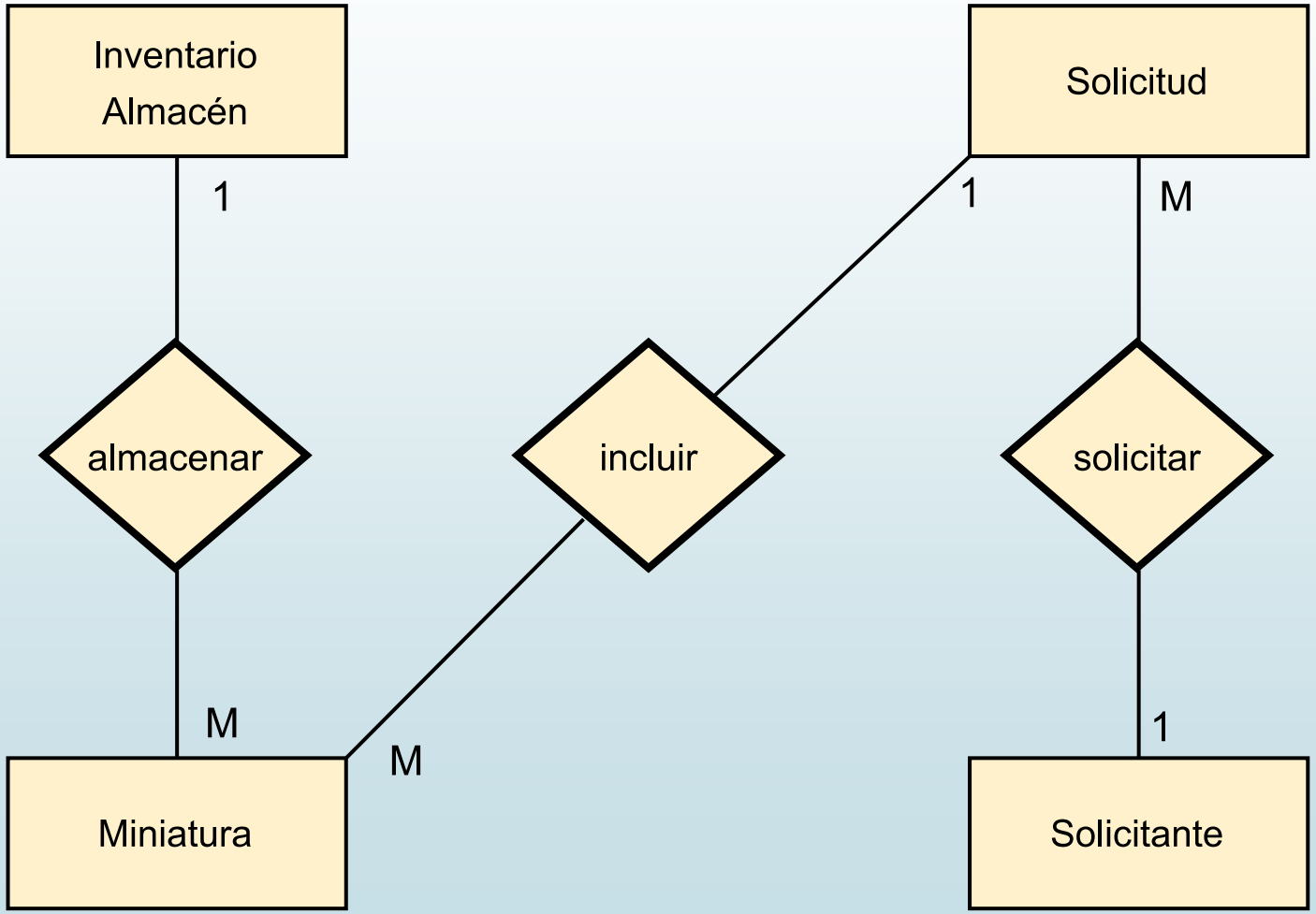
- Un diagrama E-R representa **entidades** y las **relaciones** entre ellas.
- En el Análisis de Requisitos, se emplean para representar entidades del **dominio** y sus interrelaciones.
- Se emplean en el Diseño de un proyecto para definir BD...
  - … pero su uso en el Análisis **no** implica la existencia de una BD. (Mayor nivel de abstracción).
  - No obstante, puede ser un buen punto de partida.

Diagramas E-R

- **Entidades**. Representan elementos **materiales** o conjuntos de **datos** relevantes para el negocio/sistema.
  - Tomando como referencia el DFD, tanto los actores como los almacenes de datos se convierten en entidades en el E-R.
  - Se definen en base a una serie de atributos que deberían especificarse aparte.
  - En el diseño de BD, se suelen convertir en tablas.

- **Relaciones**. Conexiones lógicas entre **pares** de entidades.
  - Recomendable etiquetarlos con un infinitivo.
  - Es posible expresar la cardinalidad (1-1, 1-M, M-M).

Diagramas E-R

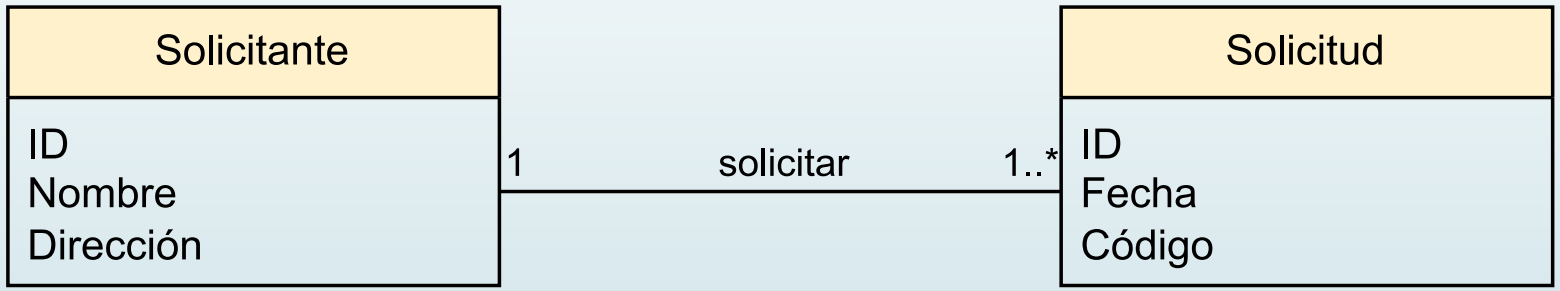


Este diagrama representa parcialmente el caso de uso previo (he ignorado las impresiones, por sencillez)

Diagramas de clase

- **Alternativa** a los diagramas E-R habitual en los desarrollos orientados a **objetos**.
- Las clases se corresponden con las entidades. Relaciones y cardinalidades se expresan con líneas simples.
- Los **atributos** que en un E-R se obviaban aquí se especifican.
  - No así las operaciones, ya que son cuestiones de implementación.

Diagramas de clase



Refinado de Requisitos II

Otras técnicas de interés

Matrices CRUD

- Técnica para el análisis de requisitos de **datos**.
- Consiste en relacionar acciones del sistema y datos para estudiar cuando éstos se crean, leen, actualizan o eliminan (**CRUD**).
- Algunas combinaciones posibles:
  - Datos y eventos.
  - Datos y casos de uso.
  - Clases y casos de uso.

Matrices CRUD

	Solicitud	Miniatura	Solicitante
Crear solicitud	C	R	R
Cambiar solicitud	U, D		R
Ver solicitudes	R		R
Gestionar inventario almacén		C, U, D	
Registro			C
Gestionar cuenta			U

¿Se realizan todas las operaciones para una entidad? ¿Por qué?

Atributos de calidad

Análisis de atributos de calidad

Según la clase de sistema, algunos atributos son más importantes:

- **Aplicaciones web y corporativas**: disponibilidad, escalabilidad, integridad (de datos), interoperabilidad, rendimiento, seguridad, usabilidad.
- **Sistemas integrados**: eficiencia, fiabilidad, rendimiento, robustez, seguridad, usabilidad.
- **Aplicaciones de escritorio y móviles**: rendimiento, seguridad, usabilidad.

Esto no significa que se limite a ellos; depende de las características del sistema y del dominio. (Requisitos emocionales de Callele, Neufeld y Schneider)

Análisis de atributos de calidad

El análisis de atributos de calidad consta de:

1. Partir de un **listado completo** de atributos.
2. **Eliminar** atributos según la información extraída en la Obtención.
3. **Priorizar** atributos.
4. **Concretar** expectativas y necesidades.

Lista de atributos de calidad

Una posible clasificación de los atributos de calidad es la que sigue:

- **Externos**. Características observadas durante la ejecución del sistema, que influencian al usuario y su experiencia.
  - *Disponibilidad, instalabilidad, integridad, interoperabilidad, rendimiento, fiabilidad, robustez, seguridad, usabilidad...*
- **Internos**. Propiedades percibidas por el equipo técnico durante su interacción con el sistema, que pueden afectar indirectamente a los usuarios.
  - *Eficiencia, modificabilidad, portabilidad, reusabilidad, escalabilidad, verificabilidad...*

A continuación, se definen y detallan los más relevantes.

Disponibilidad

- Medida del **tiempo** durante el que el sistema es **completamente operacional**.
  - Tiempo disponible/Tiempo total.
  - Tiempo medio entre fallos/Tiempo medio entre fallos + Tiempo medio de reparación.
- Relevante en aplicaciones web y, en general, sistemas de uso global.
- Estrechamente relacionada con la fiabilidad y la mantenibilidad.

Disponibilidad

- Se debe examinar si los **objetivos de negocio** exigen la disponibilidad del sistema, o parte de éste, durante determinados períodos de tiempo.
- Recomendable acordar cuándo se considera un sistema **disponible**.
  - ¿Qué ocurre con el mantenimiento rutinario? ¿Se considera disponible si es accesible en un modo limitado?
- Una disponibilidad extremadamente alta implica **costes** muy altos (redundancia del hardware, arquitectura compleja).
  - Sólo deberíamos aproximarnos al 100% en sistemas particularmente críticos.

Disponibilidad

- A considerar: [Miller 2009]
  - ¿Qué partes del sistema requieren de una mayor disponibilidad?
  - ¿Qué consecuencias tiene para los objetivos de negocio la no disponibilidad?
  - ¿Debe realizarse un mantenimiento periódico? ¿Cómo afecta a la disponibilidad?
  - ¿Cómo se notifica al usuario la no disponibilidad del sistema?
  - ¿Existen dependencias de disponibilidad entre funcionalidades?

El sistema estará disponible un 99% del tiempo entre las 6.00 y las 24:00.

Fiabilidad

- Probabilidad de que el sistema se ejecute **sin fallos** durante un período de tiempo.
- **Métricas** habituales:
  - % de operaciones completadas sin fallo.
  - Tiempo medio entre fallos (MTBF).
- Como es habitual con los atributos de calidad, es difícil asegurar si un requisito de fiabilidad se cumple...
  - ... pero sí podemos detectar si no se cumple.

Fiabilidad

- Aspectos a considerar:
  - ¿Qué ocurre si tiene lugar un fallo al realizar una operación concreta?
  - ¿Cuáles son las consecuencias de un fallo para los objetivos del sistema?
  - No todos los fallos son iguales: distinguir entre fallos críticos y simples molestias.
  - ¿Existen elementos del sistema cuya fiabilidad es de absoluta importancia?

El tiempo medio entre fallos del subsistema X será de 42 días.

Instalabilidad

- Medida de la facilidad de realización de las operaciones relacionadas con la **instalación** del sistema.
  - Instalación, recuperación, actualización, regresión, desinstalación...
- Una métrica típica es el **tiempo medio de instalación**.
- Posibles requisitos: pasos de la instalación, requisitos mínimos, variaciones por plataforma y región, niveles de acceso, configuraciones varias...

La instalación del sistema requerirá de privilegios de administrador.

Integridad

- Preservación de la **corrección** y la **completitud** de los **datos** del sistema.

- El objetivo es evitar:
  - Pérdida o corrupción de los datos.
  - Borrado o sobreescritura de los datos por parte de los usuarios.
  - Robo o corrupción intencionado de los datos.
  - Imprecisiones, inconsistencias o errores en el formato.
    - Veáse "Redondeo de los Misiles Patriot".

Integridad

- **Recomendaciones** acerca de la integridad del sistema: *[Withall 2007]*
  - Los cambios en los datos deben realizarse completamente o deshacerse si el proceso no se finaliza.
  - Los cambios en los datos deben ser persistentes.
  - Coordinar cambios realizados en distintos almacenes de datos.
  - Realizar y recuperar copias de seguridad.
    - Hay mucho que sopesar: frecuencia, objetivos, automatización, cifrado...
  - Archivado de datos.
    - Cuáles y durante cuánto tiempo...

El sistema no redondeará el tiempo transcurrido desde el arranque del sistema al registrar el valor medido.

Interoperabilidad

- Facilidad de **intercambio** de datos y servicios con **otros sistemas**, así como de integración con dispositivos externos.
- Típicamente, consiste en definir **estándares** a emplear para el intercambio de datos entre sistemas.
- A considerar:
  - ¿Con qué sistemas hemos de conectarnos y qué datos se intercambian?
  - ¿Qué estándares de datos se emplean en estos sistemas?
  - ¿Qué mensajes recibe el sistema de otros sistemas o dispositivos?
  - ¿Existen restricciones externas acerca de la interoperabilidad?

Interoperabilidad

El sistema podrá importar cualquier modelo 3D exportado de la herramienta Autodesk 3ds Max (versión 2021 o anterior).

Rendimiento

- Medida de la **velocidad** o **éxito** con el que el **sistema** actúa.
  - Abarca múltiples aspectos: tiempo de respuesta, transacciones por segundo, nº de usuarios concurrentes, latencia, capacidad de la BD...
- Afecta notablemente a la **percepción** del usuario del sistema...
  - ... y también a otros atributos, especialmente en sistemas críticos.
- Estrechamente relacionado con la eficiencia.

El sistema situará la placa giratoria en la posición indicada por el usuario en un máximo de TIEMPO_MAXIMO_GIRO segundos.

Robustez

- Capacidad del sistema para soportar **condiciones** operativas **no ideales**: defectos de los sistemas externos, errores de los usuarios o ataques maliciosos.

- Reflexionar sobre **posibles errores** y cómo el sistema debería **recuperarse** sin afectar negativamente al usuario.

Si el editor de imágenes se cierra antes de que el usuario guarde el archivo, la próxima vez que el usuario inicie el editor el sistema recuperará los contenidos del archivo de, como máximo, TIEMPO_MAXIMO_RECUPERACIÓN antes del fallo.

Seguridad (*safety*)

- **Prevención** de **daños** personales o materiales por parte del sistema.
- Típicamente, se expresan como acciones o estados que el sistema **no** debe permitir.
  - Según el dominio, habrá restricciones externas más o menos detalladas.
  - Habituales en sistemas que controlan dispositivos físicos (Therac-25).
- A considerar:
  - ¿Cómo podría este sistema dañar a una persona? ¿Cómo puede el sistema detectar estos casos y cómo debería responder?
  - ¿Qué acciones del usuario podrían inadvertidamente causar daños?
  - ¿Existen modos de operación que suponen riesgos específicos?

Seguridad (*safety*)

El sistema finalizará inmediatamente el proceso de radiación ionizante si detecta niveles de radiación anómalos.

Seguridad (*security*)

- **Protección** de ataques y accesos no autorizados a las capacidades e información del sistema.

- Aspectos a tratar:
  - Autenticación de los usuarios y niveles de acceso.
  - Privacidad de los datos.
  - Protección contra malware.
  - Encriptado de datos.
  - Logs de operaciones y accesos.

Seguridad (*security*)

- En **SSI**, se describe cómo elaborar requisitos de seguridad (gestión de datos, autenticación y otros) y se proveen algunos ejemplos:
  - Tema 6: Introducción a la seguridad de aplicaciones.
  - Seminario 5 (ejemplos).

El sistema no permitirá a los usuarios reutilizar contraseñas que haya empleado previamente.

Usabilidad

- **Esfuerzo** necesario por parte del **usuario** para operar con el sistema e interpretar sus salidas.
- Abarca muchos temas, en ocasiones **contradictorios**:
  - Facilidad de uso, facilidad de aprendizaje, accesibilidad, gestión de errores...
- **Métricas**:
  - Tiempo medio para realizar una tarea.
  - % de tareas realizadas sin ayuda/errores.
  - Nº de errores durante una tarea.
  - Nº de interacciones requerido para realizar una tarea.

Usabilidad

Un usuario con, al menos, tres meses de experiencia será capaz de solicitar una miniatura en un tiempo medio de 5 minutos y un máximo de 10 minutos.

Eficiencia

- **Aprovechamiento** del sistema de los **recursos** de procesamiento.
  - Directamente relacionada con el rendimiento.
- A considerar:
  - Definir requisitos mínimos de hardware.
  - Definir márgenes para condiciones inesperadas y futuras.
  - Lo anterior se basará en el nº de operaciones y usuarios concurrentes previstos para el sistema.

El 33% de la memoria RAM disponible para el sistema se dejará libre durante los períodos de máxima carga previstos.

Escalabilidad

- Capacidad del sistema para **acomodar más** servicios o usuarios **sin comprometer** otros atributos.
- Puede implicar ampliaciones de hardware u optimizaciones de software.
- A considerar:
  - ¿Cómo podría evolucionar la demanda del sistema en los próximos años?
  - ¿Cuál es el rendimiento mínimo aceptable independientemente del nº de usuarios?

La capacidad del sistema debe ser capaz de incrementarse de 420 transacciones al día a 1500 transacciones al día en un máximo de 24 horas.

Modificabilidad

- Facilidad para **comprender**, **cambiar** y **extender** el diseño y código del sistema.
- Varias dimensiones dependientes del tipo de mantenimiento:
  - Si **corregimos** defectos, nos centramos en mantenibilidad y comprensibilidad.
  - Si **mejoramos** y extendemos la funcionalidad: extensibilidad, flexibilidad.
  - Si debemos **adaptarnos** a nuevos entornos: mantenibilidad.
- **Métricas**:
  - Tiempo medio para añadir nueva característica/solucionar un defecto.
  - % de modificaciones realizadas exitosamente.

Modificabilidad

Un programador de mantenimiento con al menos un año de experiencia podrá modificar los algoritmos de cálculo de impuestos para adaptarlos a cambios en la normativa tributaria en un máximo de 10 horas de trabajo.

Portabilidad

- Esfuerzo necesario para **migrar** el sistema de un **entorno** a otro.
  - Algunos consideran la internacionalización como parte de la portabilidad.
- Es común que una aplicación deba funcionar en distintos sistemas operativos, navegadores o dispositivos.
- A considerar:
  - ¿En qué plataformas debe funcionar este sistema?
  - ¿Qué elementos del sistema necesitan ser portables?
  - ¿Se comprometen otros atributos de calidad al incrementar la portabilidad?

El sistema permitirá importar las características del perfil entre IoS y Android.

Reusabilidad

- Esfuerzo requerido para **emplear parte** del sistema en otros sistemas.
- Algunos **prerrequisitos** de la reusabilidad:
  - Modularidad, independencia del entorno, documentación precisa, cierta genericidad.
- A considerar:
  - ¿Qué requisitos o datos podrían ser reusados en el futuro?
  - ¿Qué funcionalidades de este sistema podrían emplearse en otros?

Los algoritmos de cálculo de impuestos serán reusables en futuras aplicaciones de compra-venta.

Verificabilidad

- Capacidad de **evaluación** de las funcionalidades del sistema.
- Imprescindible en sistemas críticos o sometidos a numerosos cambios.
- A considerar:
  - ¿Cómo comprobamos que las operaciones producen los resultados deseados?
  - ¿Qué salidas del sistema podemos examinar para comprobar su validez?
  - ¿Existen funciones del sistema indeterministas?

La máxima complejidad ciclomática de un módulo del sistema será de 21.

Priorización de atributos de calidad

- Una técnica útil para la priorización es la **comparación por pares**.
- Tras contrastar todas las combinaciones, se **ordenan** en función de las comparaciones ganadas.
- La priorización nos permite **resolver conflictos** en fases posteriores.

Concreción de atributos de calidad

- Por sí solos, los atributos de calidad no nos proporcionan suficiente información.
  - *"El rendimiento es importante para nuestro sistema".*
- Debemos concretar las **expectativas** y **necesidades** de los stakeholders relativas a un atributo de calidad.
  - ¿Cuál es un tiempo de respuesta aceptable para el usuario?
  - ¿Cuál es un tiempo de respuesta inaceptable para el usuario?
  - ¿Cuál sería el nº de usuarios concurrentes de media?
  - ¿Cuál sería el máximo nº de usuarios concurrentes en un momento dado?
  - *Ver las preguntas listadas en "A considerar" en los atributos anteriores.*

Concreción de atributos de calidad

- Resulta útil **definir** los **límites** de lo aceptable.
  - ¿Cuál es un tiempo de respuesta inaceptable para el usuario?
- Nos permiten definir **pruebas** de cumplimiento.
  - Si no es posible forzar al sistema a una situación definida como inaceptable, probablemente hemos cumplido con los objetivos de calidad.
  - Los usuarios no autorizados no podrán eliminar archivos.