INGENIERÍA DE SOFTWARE I

Tema 6: Flujos de trabajo del Proceso Unificado

2º G.I.I.
Fecha de última modificación: 20-2-2018

Dr. Francisco José García Peñalvo / fgarcia@usal.es
Alicia García Holgado / aliciagh@usal.es

Departamento de Informática y Automática
Universidad de Salamanca

VNiVERSiDAD D SALAMANCA
CAMPUS OF INTERNATIONAL EXCELLENCE

GRIAL Research Group University of Salamanca

Resumen

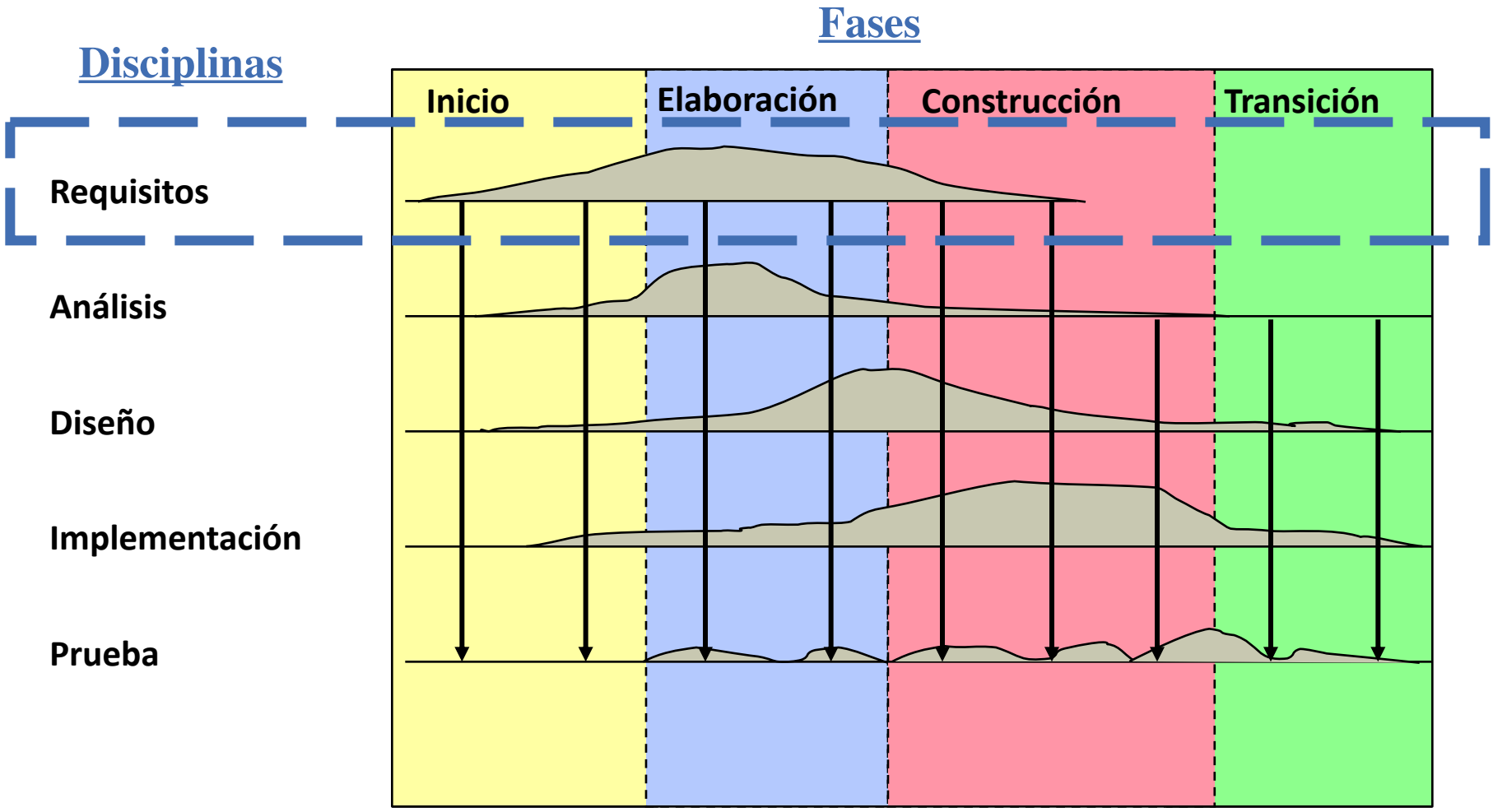
Resumen	Este tema recoge los flujos de trabajo del Proceso Unificado vinculados con los requisitos, el análisis y el diseño. Se pretende que este tema sea una referencia a estos flujos de trabajo, si bien estos mismos se van a desarrollar en los temas que profundizan sobre los conceptos ligados a estas fases del ciclo de vida desde un enfoque orientado a objetos
Descriptores	Proceso; Proceso Unificado; ciclo de vida; requisito; caso de uso; escenario; modelo de dominio; modelo de análisis; modelo de diseño; arquitectura *software*; clase; interfaz
Bibliografía	[Jacobson et al., 2000] Capítulos 6, 7, 8 y 9

Esquema

- Requisitos en el Proceso Unificado
- Análisis en el Proceso Unificado
- Diseño en el Proceso Unificado
- Referencias

1. Requisitos en el Proceso Unificado

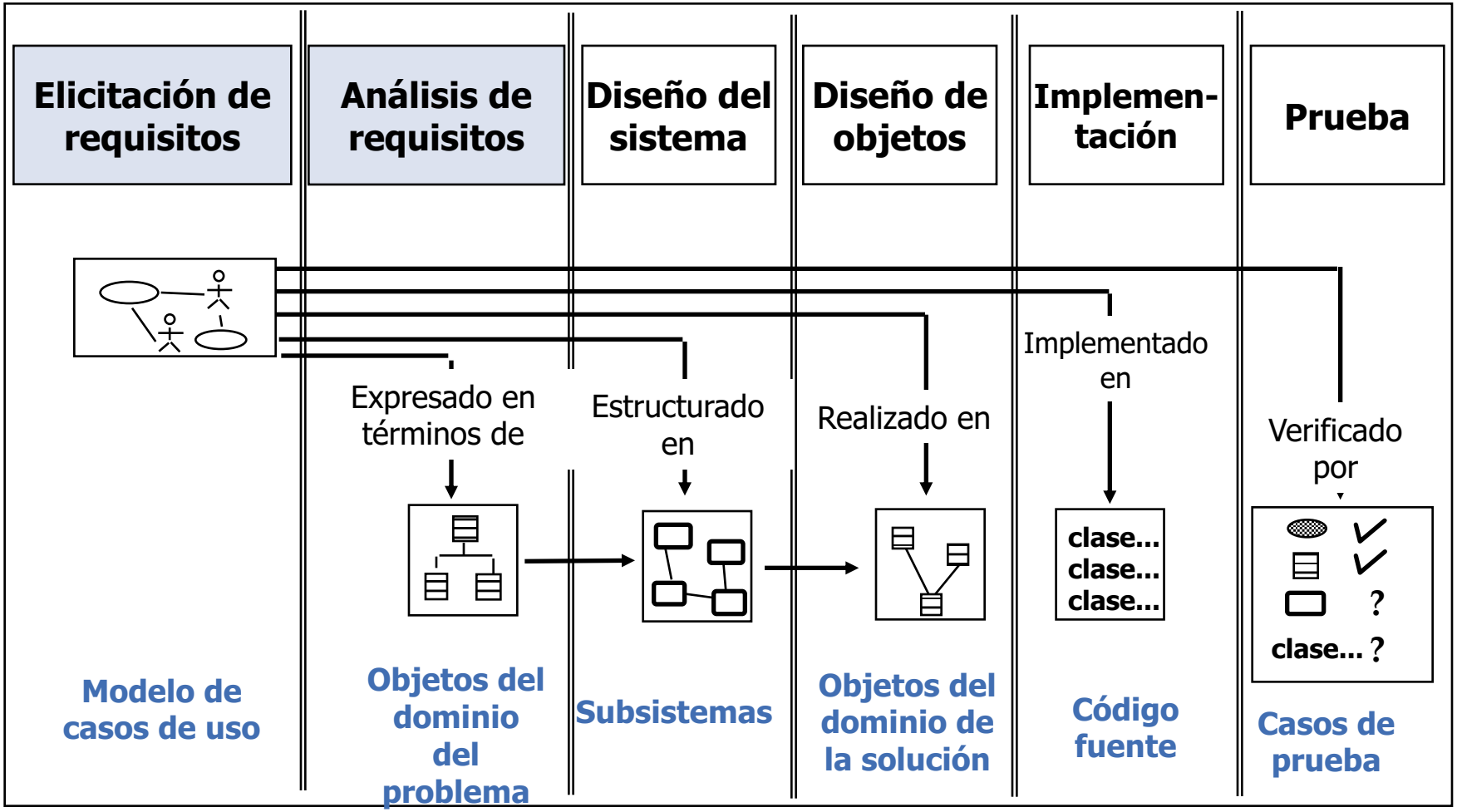
Requisitos en el Proceso Unificado



Requisitos en el Proceso Unificado

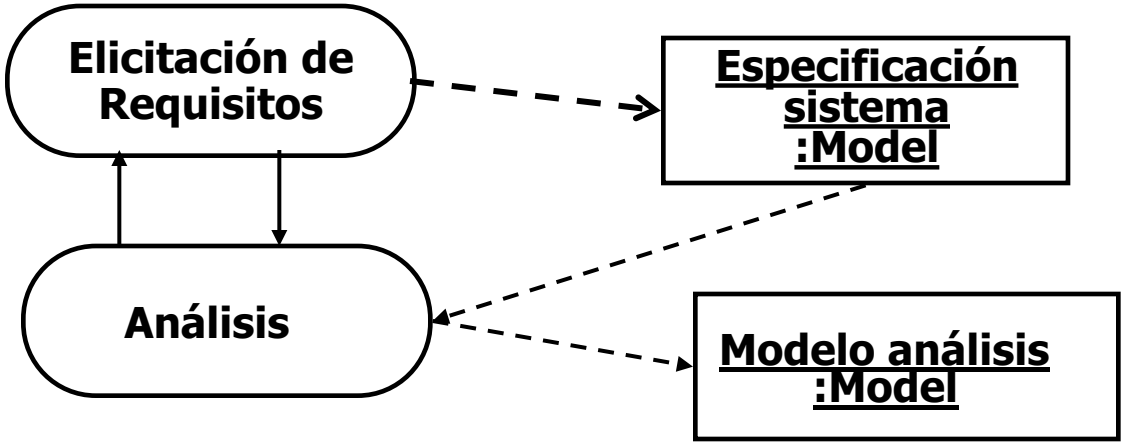
- Construye dos modelos principales
  - **Modelo del dominio**
    - Describe los requisitos de datos estáticos de la aplicación
  - **Modelo de casos de uso**
    - Describe los requisitos de procesamiento dinámico de la aplicación
- Estos modelos se desarrollan de forma incremental durante el desarrollo (principalmente en las fases de inicio y elaboración)
  - **Inicio**: Identifica la mayoría de los elementos del dominio y de los casos de uso para delimitar el sistema y el alcance del proyecto. Se detallan los casos de uso críticos
  - **Elaboración**: Se capturan los restantes requisitos de forma que se pueda estimar el tamaño y el esfuerzo de desarrollo
  - **Construcción**: Se capturan los requisitos residuales
  - **Transición**: No se capturan requisitos salvo que alguno haya cambiado

Proceso dirigido por casos de uso



[Bruegge y Dutoit, 2000]

Captura y recogida de requisitos



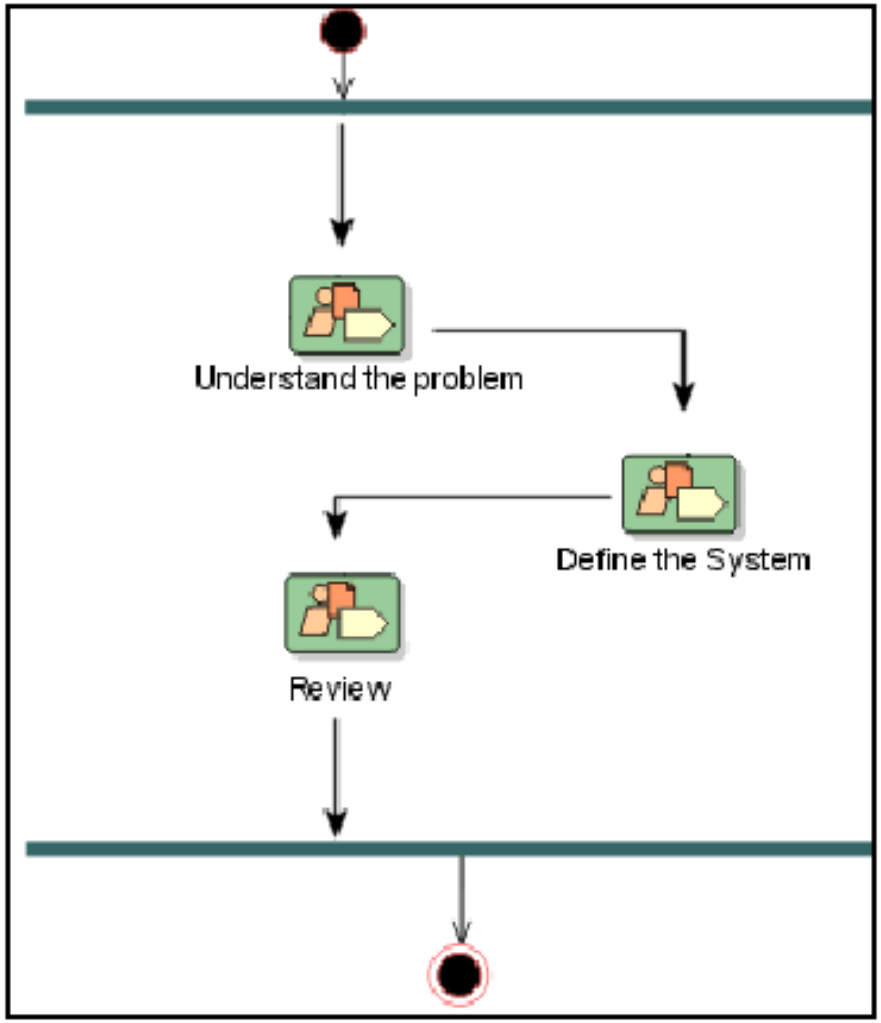
- **Elicitación de requisitos**
  - Especificación del sistema comprensible por el usuario
- **Análisis**
  - Modelo de análisis que el equipo de desarrollo puede interpretar sin ambigüedades
- La especificación del sistema y el modelo de análisis representan la misma información, uno en lenguaje natural y el otro en notación formal o semiformal
- Se centra en la visión que el usuario tiene del sistema

Identificación de las necesidades del usuario

- Recoger información del dominio de la aplicación
  - **Investigación** de la documentación existente
  - **Observación** de cómo se realiza el trabajo diario
  - **Entrevistas** personales y mediante cuestionarios
  - **Prototipado** de las interfaces y las funciones

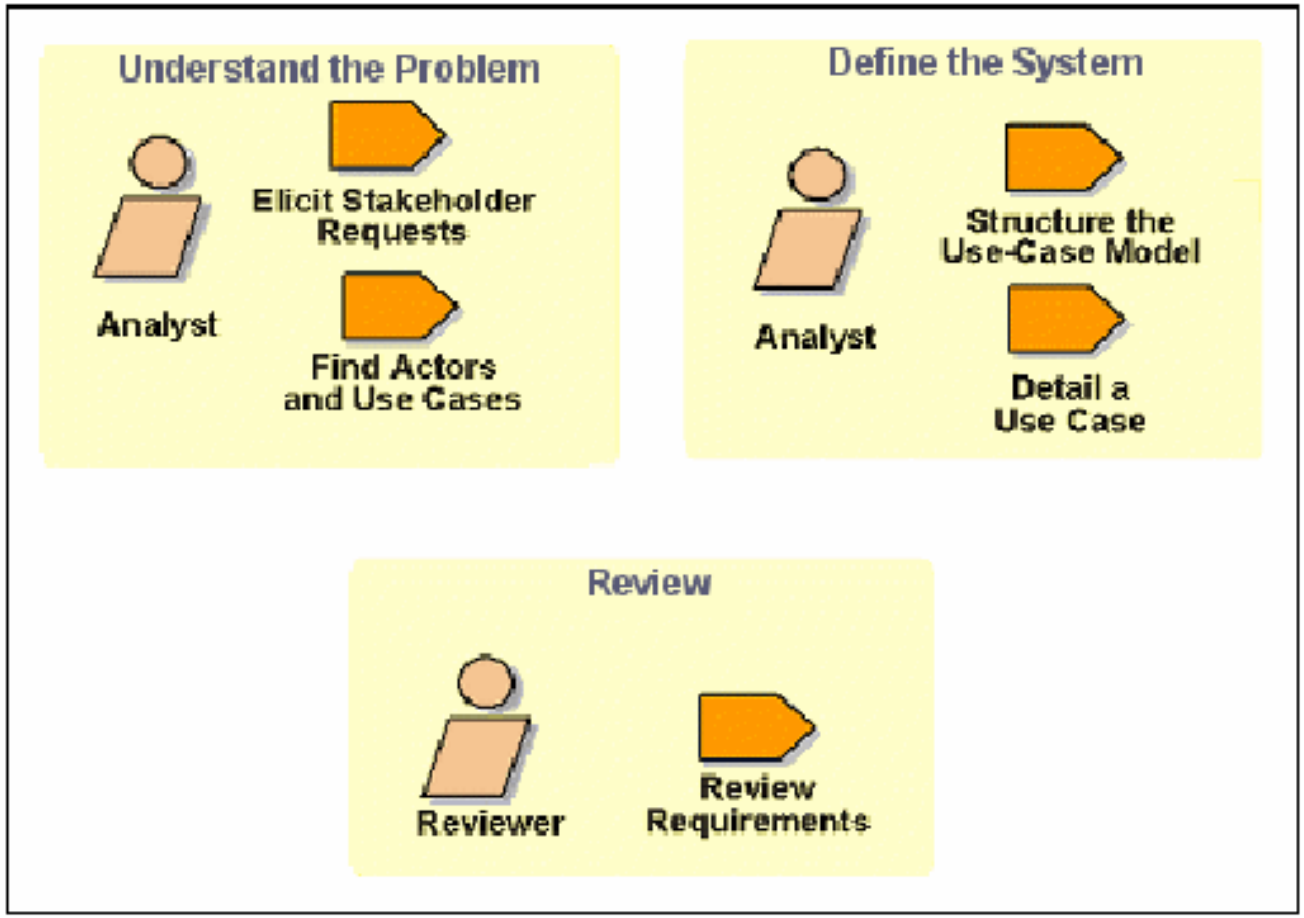
- Distinguir las necesidades de los deseos
  - **Necesidades**
    - Características críticas para el buen funcionamiento del sistema
  - **Deseos**
    - Características deseables pero no esenciales

Flujos de trabajo en la disciplina de requisitos



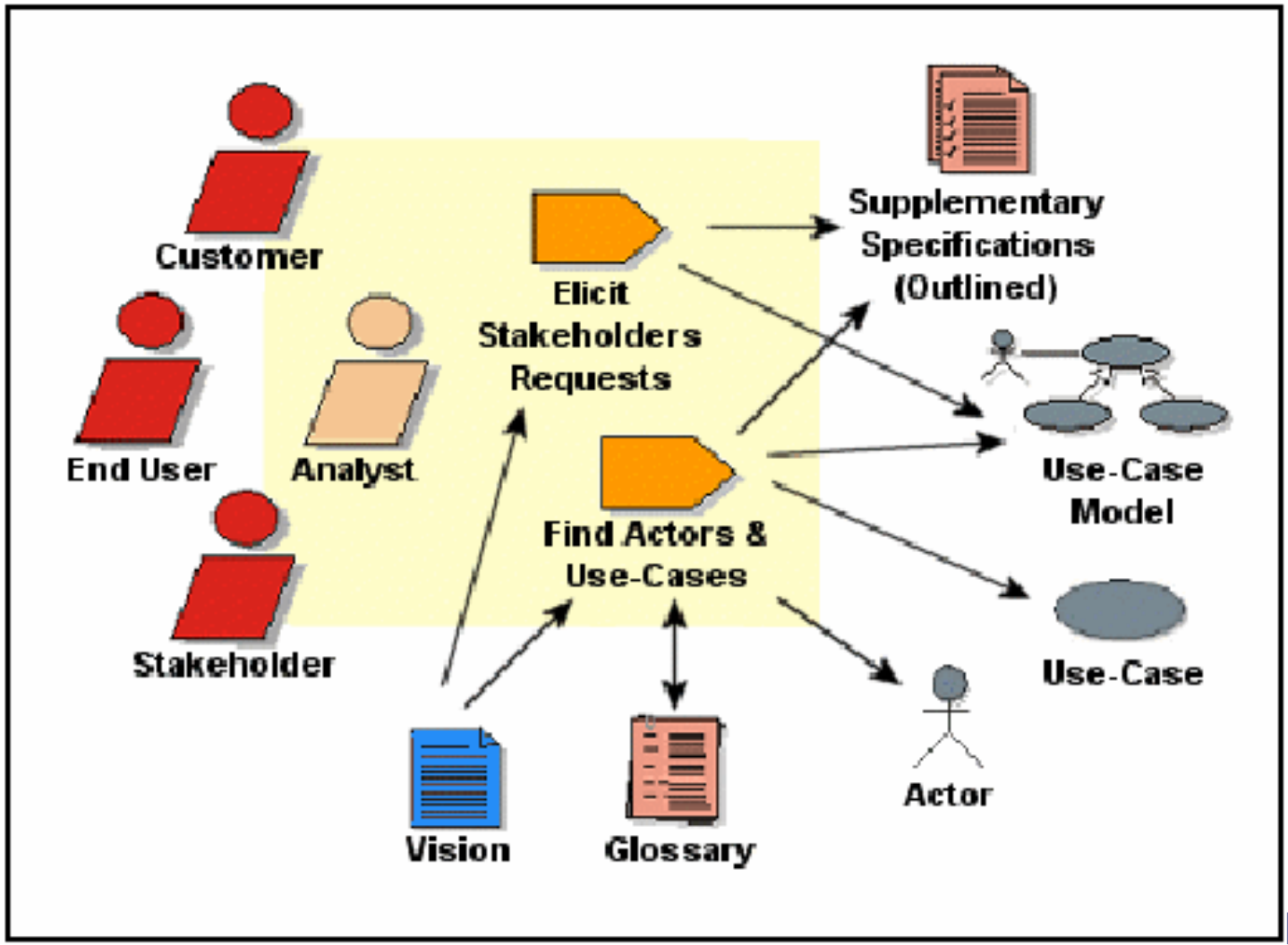
[Jacobson et al., 1999]

Actividades en la disciplina de requisitos



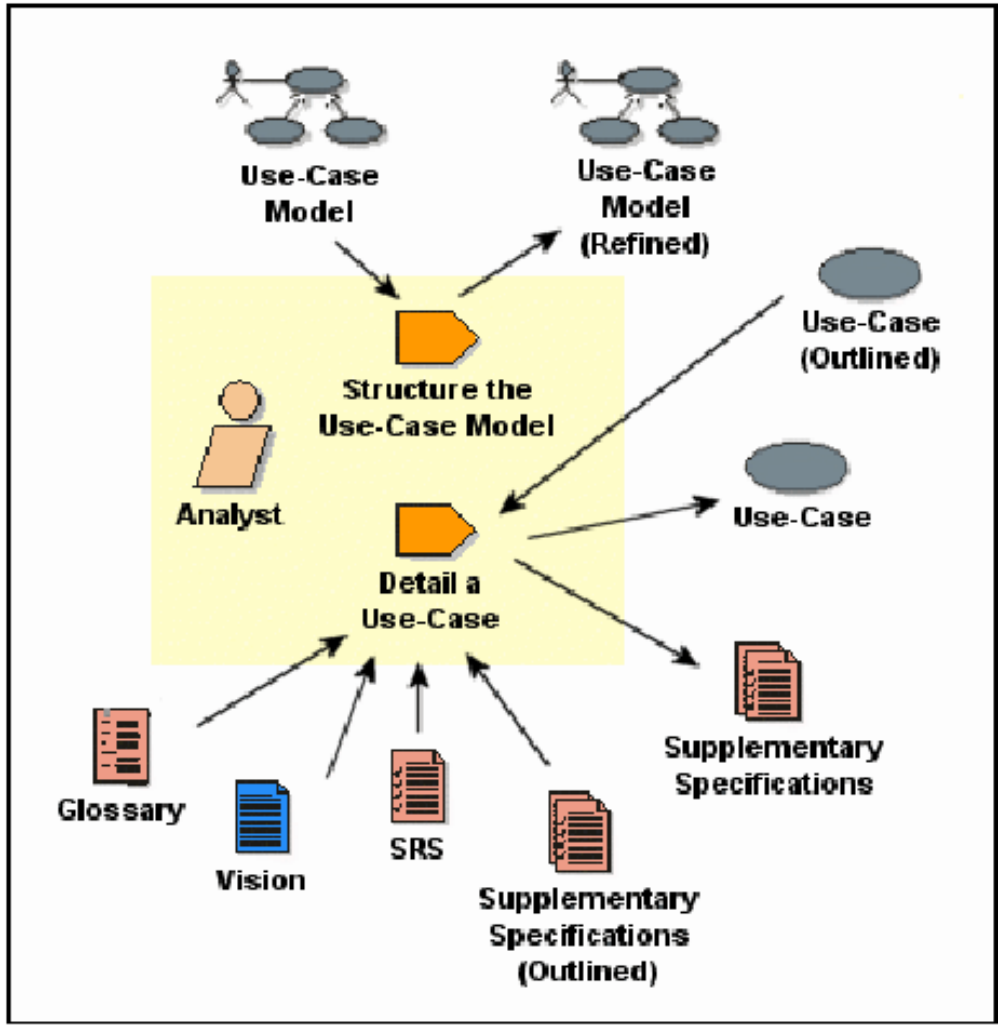
[Jacobson et al., 1999]

Comprensión del problema



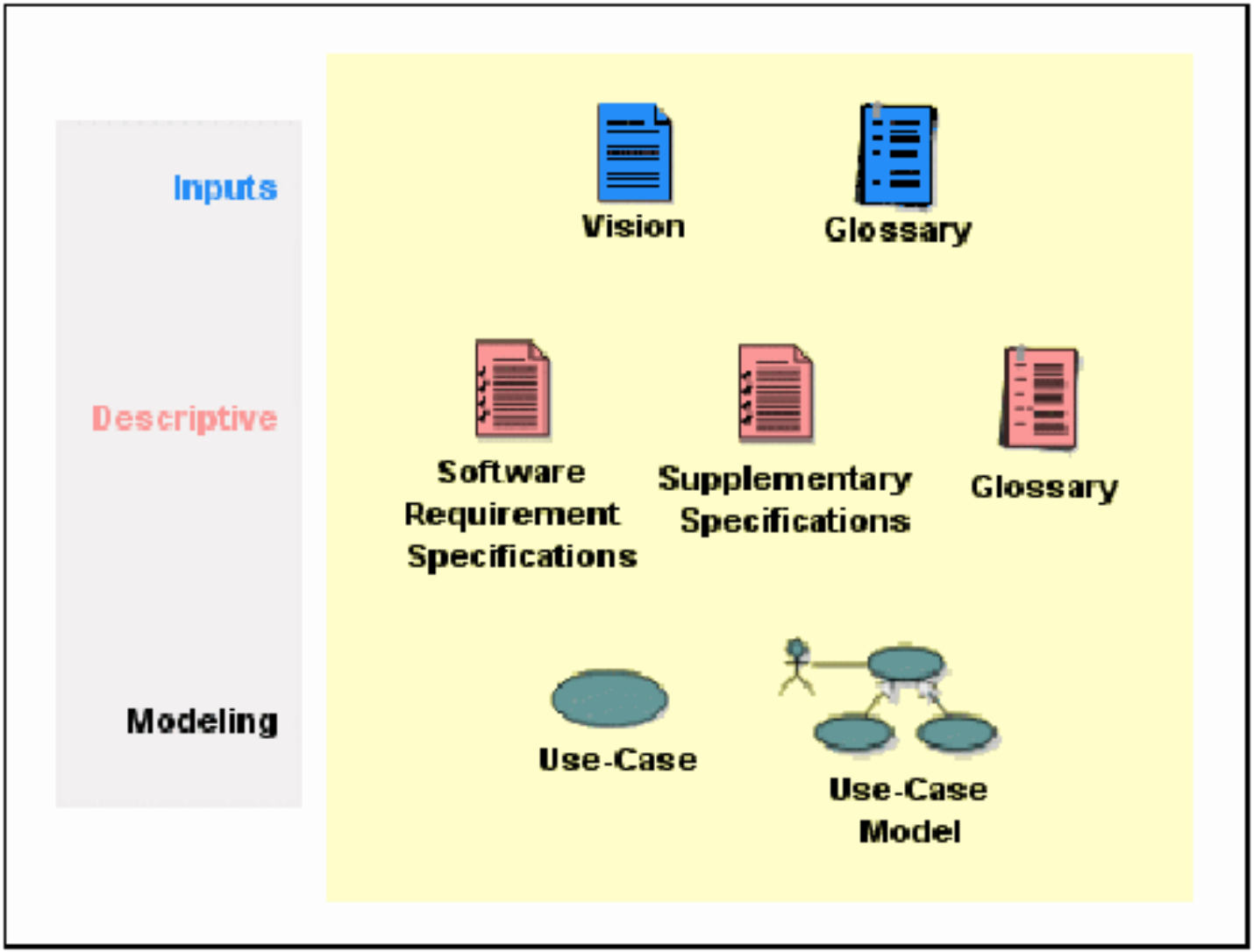
[Jacobson et al., 1999]

Definición del sistema



[Jacobson et al., 1999]

Artefactos en la disciplina de requisitos



[Jacobson et al., 1999]

Modelo de requisitos - Modelo de casos de uso

- El objetivo es **delimitar el sistema** y **definir qué funcionalidad** tiene que afectar al sistema
- El primer modelo a crear del sistema es el modelo de casos de uso
- Son una representación orientada a la funcionalidad del sistema
- Son la base para la identificación de las clases semánticas del sistema en el análisis orientado a objetos
- Describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario
  - Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno

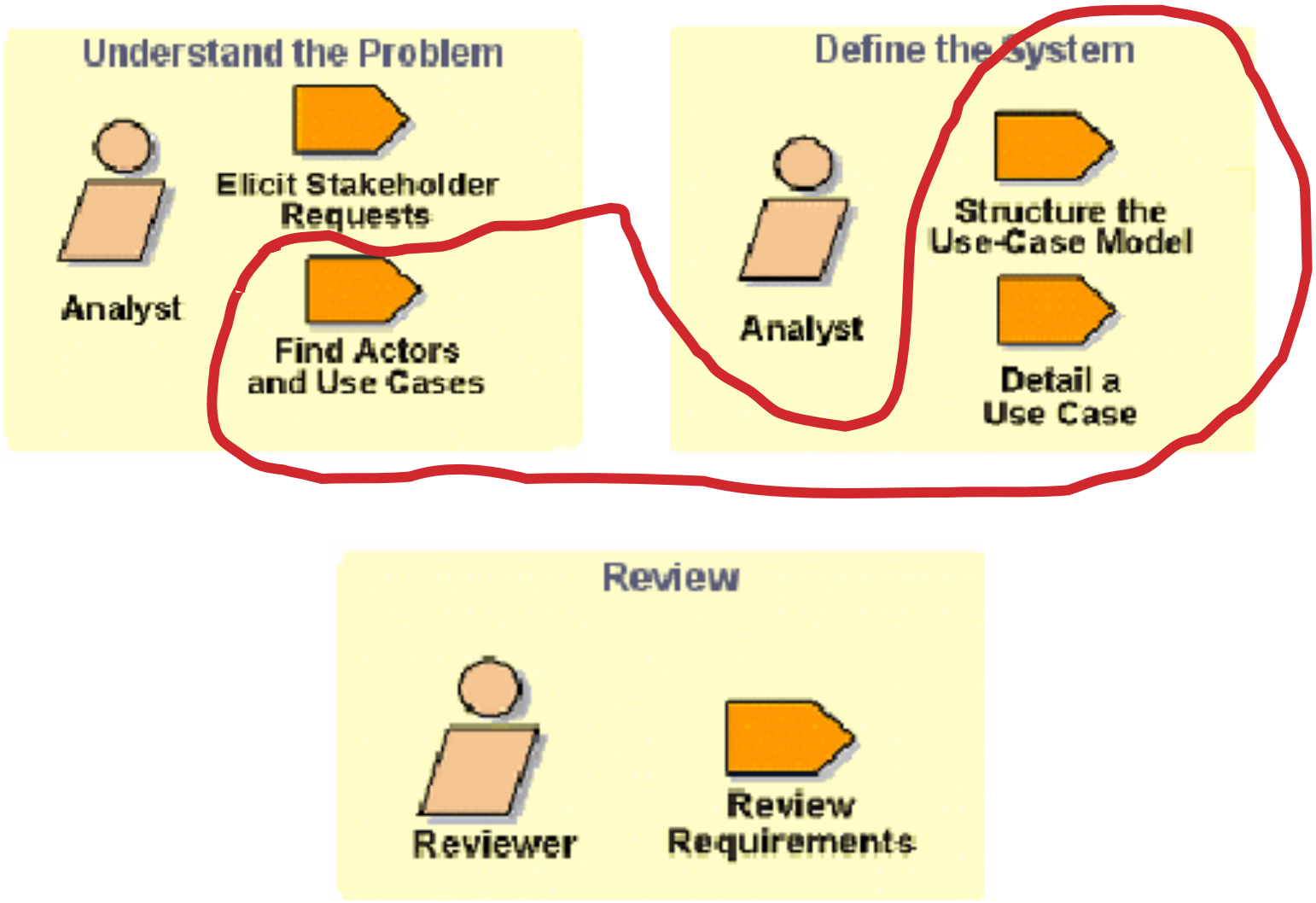
Conceptos a manejar (i)

- El concepto principal a manejar será el de **escenario** que vendrá descrito diagramáticamente por diagramas de interacción y que, en la mayoría de las ocasiones, se denominarán en un abuso del lenguaje como "casos de uso"
- Escenario
  - Iteración estructurada entre entidades en la que se invocan a las responsabilidades de éstas
  - Secuencia de pasos que suceden en ciertas condiciones para conseguir el objetivo primario del actor y con un resultado determinado en relación a este objetivo
  - Hechos que describen un sistema existente y su entorno incluyendo el comportamiento de los agentes con la suficiente información de contexto para permitir el descubrimiento y la validación de los requisitos del sistema
  - Son instancias de la experiencia actual con un sistema según se percibe por sus usuarios
  - Una descripción narrativa de lo que la gente hace y experimenta cuando intenta utilizar sistemas de computación y aplicaciones

Conceptos a manejar (ii)

- Paso de escenario
  - Una interacción con el sistema
  - Una responsabilidad a ser realizada por un actor
  - Un caso de uso referenciado
- Casos de uso
  - Colección de posibles escenarios entre el sistema en estudio y actores externos, caracterizados por el objetivo que el actor primario tiene en relación con las responsabilidades del sistema
- Casos de uso externos
  - Tratan al sistema como una "caja negra" y muestran cómo las entidades externas y su entorno interaccionan con él
  - Muestran cómo las entidades externas "utilizan" el sistema para hacer algo
- Casos de uso internos
  - Muestran cómo interaccionan las entidades internas del sistema
  - Puede interpretarse como una muestra de cómo las entidades "utilizan" a otras entidades para conseguir "hacer cosas"

Ingeniería de requisitos en el Proceso Unificado (i)



Actividades

Proceso Iterativo

- Identificar actores
  - Se identifican los diferentes tipos de usuarios a los que el sistema ha de dar soporte
- Identificar escenarios
  - Se desarrolla el conjunto de escenarios para la funcionalidad proporcionada por el sistema
- Identificar casos de uso
  - Obtención de los casos de uso que representan el sistema
- Refinar casos de uso
  - Garantizar que la especificación del sistema esté completa. Se describe el comportamiento del sistema en presencia de errores o condiciones de excepción
- Identificar relaciones entre casos de uso
  - Se consolida el modelo de casos de uso eliminando redundancias
- Identificar requisitos no funcionales
  - Aspectos relacionados con la funcionalidad: restricciones de rendimiento, documentación, recursos, seguridad, calidad

Actividad: Identificar actores (i)

- Representan entidades externas que interaccionan con el sistema
  - Cualquier cosa que está "enlazada" al sistema: persona, sistema software, dispositivos hardware, almacenes de datos, redes de comunicaciones
- Son abstracciones de rol y no necesariamente se corresponden con personas
- Cada entidad externa puede estar representada por varios actores
  - Si una persona física desempeña diferentes roles respecto al sistema se representa por varios actores
- Sirven para definir los límites del sistema y para buscar todas las perspectivas desde las que los desarrolladores tienen que considerar el sistema

Actividad: Identificar actores (ii)

- Guía
  - ¿Quién usa el sistema?
    - ¿Qué grupo de usuarios están soportados por el sistema para llevar a cabo su trabajo?
    - ¿Qué grupo de usuarios ejecutan las principales funciones del sistema?
  - ¿Qué grupo de usuarios realiza las funciones auxiliares, tales como mantenimiento y administración?
    - ¿Quién inicial el sistema?
    - ¿Quién instala el sistema?
    - ¿Quién apaga el sistema?
  - ¿Interaccionará el sistema con algún sistema software o hardware externo?
    - ¿Existen otros sistemas que utilicen el sistema?
    - ¿Quién obtiene información del sistema?
    - ¿Quién proporciona información a nuestro sistema?
  - ¿Ocurre algo de forma automática en un tiempo predeterminado?

Actividad: Identificar escenarios (i)

- Descripción informal, concreta y orientada de una característica del sistema desde el punto de vista de un actor
- Los escenarios pueden tener diferentes utilidades a lo largo del ciclo de vida. Existen los siguientes tipos de escenarios
  - ***As-is scenarios:*** Describen la situación actual
  - ***Visionary scenarios:*** Describen un sistema futuro. Pueden utilizarse en la obtención de requisitos y en la representación de diseño. Pueden considerarse como prototipos baratos
  - ***Evaluation scenarios:*** Describen las tareas de usuario contra las que se evaluará el sistema
  - ***Training scenarios:*** Son tutoriales utilizados para introducir a los nuevos usuarios al sistema. Son instrucciones paso a paso diseñados para llevar de la mano al usuario a través de las tareas comunes
- En la elicitación de requisitos los desarrolladores y los usuarios escriben y refinan una serie de escenarios para conseguir un entendimiento común acerca de lo que debe ser el sistema

Actividad: Identificar escenarios (ii)

- Guía
  - ¿Cuáles son las tareas que el actor quiere que realice el sistema?
  - ¿A qué información quiere acceder el actor?
  - ¿Quién crea los datos?
  - ¿Puede ser modificada o eliminada? ¿Por quién?
  - ¿Acerca de que cambios externos tiene el actor que informar al sistema? ¿Con qué frecuencia? ¿Cuándo?
  - ¿Acerca de qué eventos ha de ser informado el actor por parte del sistema? ¿Con qué periodicidad?
  - La respuesta a estas preguntas se obtiene de la documentación del dominio de la aplicación
  - Los escenarios han de escribirse utilizando el lenguaje del dominio
  - Los escenarios se formalizan en los casos de uso

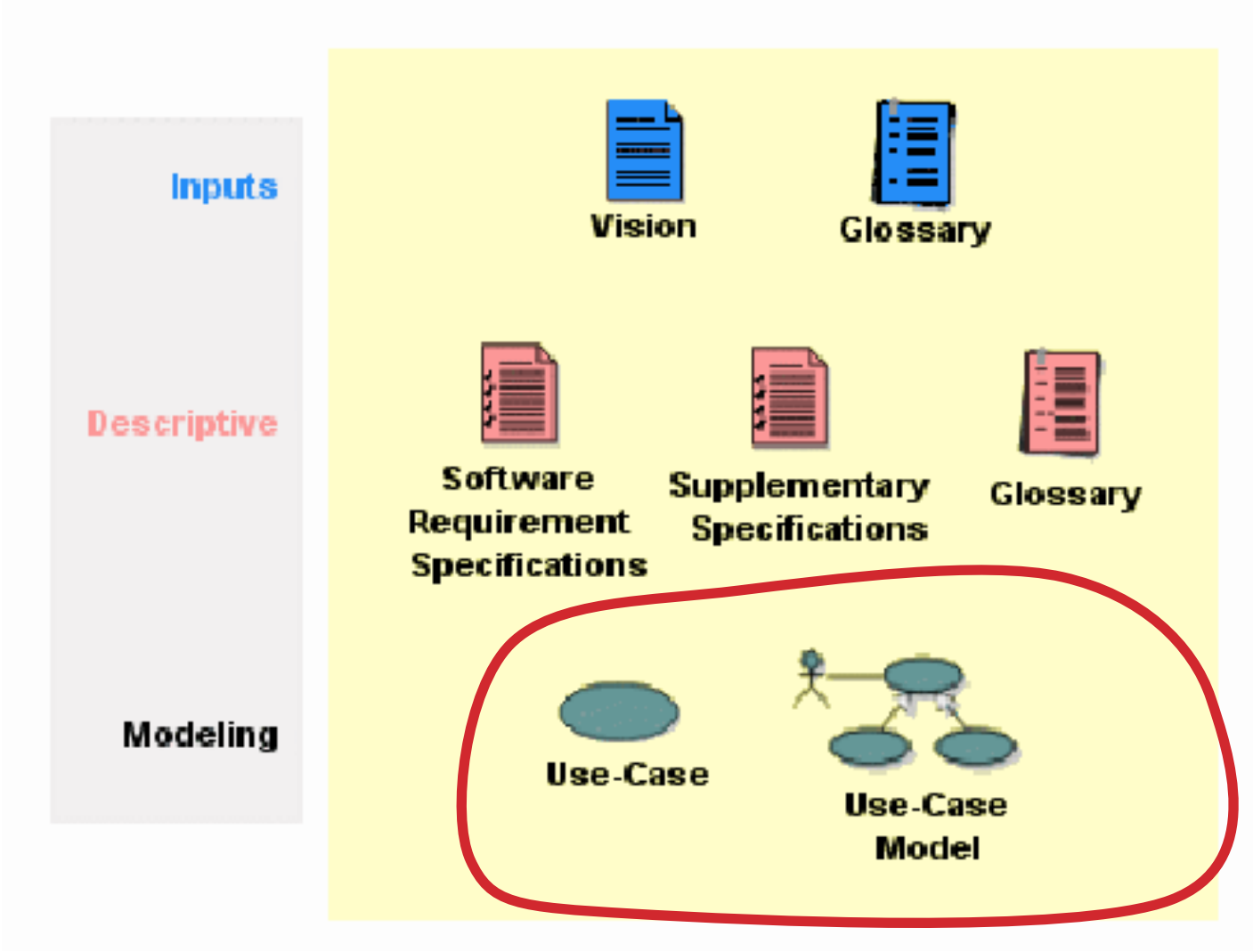
Actividad: Identificar casos de uso (i)

- La identificación de actores y escenarios permiten la comprensión del dominio de la aplicación y la definición del sistema correcto
- El siguiente paso es la formalización de los escenarios en casos de uso
- Un caso de uso especifica todas los escenarios posibles para una funcionalidad dada
- Un caso de uso se inicia por un actor
- Una vez iniciado el caso de uso puede interaccionar con otros actores
- Cada caso de uso es una secuencia completa de eventos iniciada por un actor y especifica la interacción que tiene lugar entre un actor y el sistema
- Un caso de uso es una forma concreta de utilizar un sistema haciendo uso de alguna parte de su funcionalidad
- Agrupa una familia de escenarios de uso según un criterio funcional
- Es un proceso iterativo en el que se debe incluir, refinar, rescribir, eliminar los casos de uso
- No hay que olvidar los casos "raros" o el manejo de excepciones

Actividad: Identificar casos de uso (ii)

- Se identifican los casos de uso de cada actor
  - Un caso de uso es un comportamiento del sistema que produce un resultado mensurable y valioso para un actor
  - Describe las cosas que los actores quieren del sistema
  - Debe ser una tarea completa desde la perspectiva del actor
  - En UML un caso de uso siempre se inicializa por un actor. Pueden existir situaciones en las que se inicie de forma interna al sistema
- Guía para encontrar casos de uso
  - ¿Qué funciones desea el actor del sistema?
  - ¿Almacena el sistema información? ¿Qué actores la crean, leen, actualizan o borran?
  - ¿Necesita el sistema comunicar los cambios en su estado interno a algún actor?
  - ¿Existen eventos externos que el sistema tenga que conocer? ¿Qué actores notifican estos eventos al sistema?
  - ¿Cuáles son las operaciones de mantenimiento del sistema?
- Los casos de uso se determinan observando y precisando, actor por actor, las secuencias de interacción (los escenarios) desde el punto de vista del usuario

Ingeniería de requisitos en el Proceso Unificado (ii)



Actividad: Elaborar el modelo de casos de uso inicial

- Una vez identificados los actores, escenarios y casos de uso se construye el **modelo inicial de casos de uso**
  - Se establecen las relaciones de comunicación entre los actores y los casos de uso
    - Estas relaciones representan el flujo de información del caso de uso
    - El actor que inicia el caso de uso se distingue del resto de actores con los que se puede comunicar el caso de uso
    - Estas relaciones se van identificando a medida que se identifican los casos de uso
  - Se construye el Diagrama de Casos de Uso
  - Se describen los casos de uso
    - Descripción del escenario básico

Actividad: Documentar los casos de uso (i)

- Cada caso de uso tiene que describir **qué** hace un sistema
- Un caso de uso debe ser simple, inteligible, claro y conciso
- Hay que considerar
  - Funcionalidad básica – Flujo de eventos principal (escenario principal)
  - Alternativas – Flujo de eventos excepcional (escenario alternativo)
  - Condiciones de error – Flujo de eventos excepcional
  - Precondiciones y poscondiciones – Qué tiene que ser cierto antes y después del caso de uso
- La descripción caso de uso puede contener condiciones, bifurcaciones e iteraciones
- El flujo de eventos se describe inicialmente de forma textual
- Cuando la comprensión de los requisitos ha avanzado se utilizan diagramas para especificar los escenarios. Diagramas de interacción
- Conviene separar el flujo principal del flujo alternativo
- Se comienza describiendo el flujo principal (escenario básico) al que se le irán añadiendo alternativas y excepciones

Actividad: Documentar los casos de uso (ii)

- Escenario básico
  - Se escribe como si todo transcurriese de forma correcta
  - Debe existir un escenario básico para cada caso de uso
  - Serie de sentencias declarativas sin ramificaciones ni alternativas
- Escenarios alternativos
  - Describen secuencias distintas a la que fue utilizada en el camino básico
  - Documentan las alternativas, situaciones de error o cancelación
  - Dos métodos de localización de caminos alternativos
    - Revisión de cada sentencia del escenario básico
      - ¿Hay alguna acción adicional que pueda hacerse en este punto?
      - ¿Hay algo que pueda fallar en este punto?
      - ¿Hay algún comportamiento que pueda suceder en cualquier momento?
    - Utilización de categorías
      - Un actor [aborta la aplicación / cancela una operación particular / solicita ayuda / proporciona mal los datos]
      - El sistema [falla / no está disponible]
  - Cada camino alternativo necesita un nombre y/o una descripción breve
  - Se documenta en una sección de caminos alternativos o en documentación aparte

Actividad: Documentar los casos de uso (iii)

- Flujo de eventos
  - Es una serie de sentencias declarativas que "relatan" los pasos de un escenario desde la perspectiva del actor
  - Ha de indicar cómo se inicia
    - Sentencia del tipo "El caso de uso comienza cuando..."
  - Ha de indicar cómo finaliza
    - Sentencia del tipo " El caso de uso finaliza con..."
  - Las alternativas se muestran con una sentencia `if`
  - Se pueden indicar repeticiones
    - Hay que indicar claramente donde empieza y finaliza la repetición
    - Hay que indicar la condición de finalización
    - Utilizar `for` o `while`
  - Cada paso tiene que ser una sentencia declarativa simple
  - Por defecto, los pasos deberán estar ordenados temporalmente. En caso contrario ha de especificarse
  - No debe incluirse demasiado detalle. Se están recogiendo los requisitos, no realizando análisis y diseño
- Los requisitos de tipo no funcional o se ponen en un documento aparte o se añaden como coletilla al final de la descripción

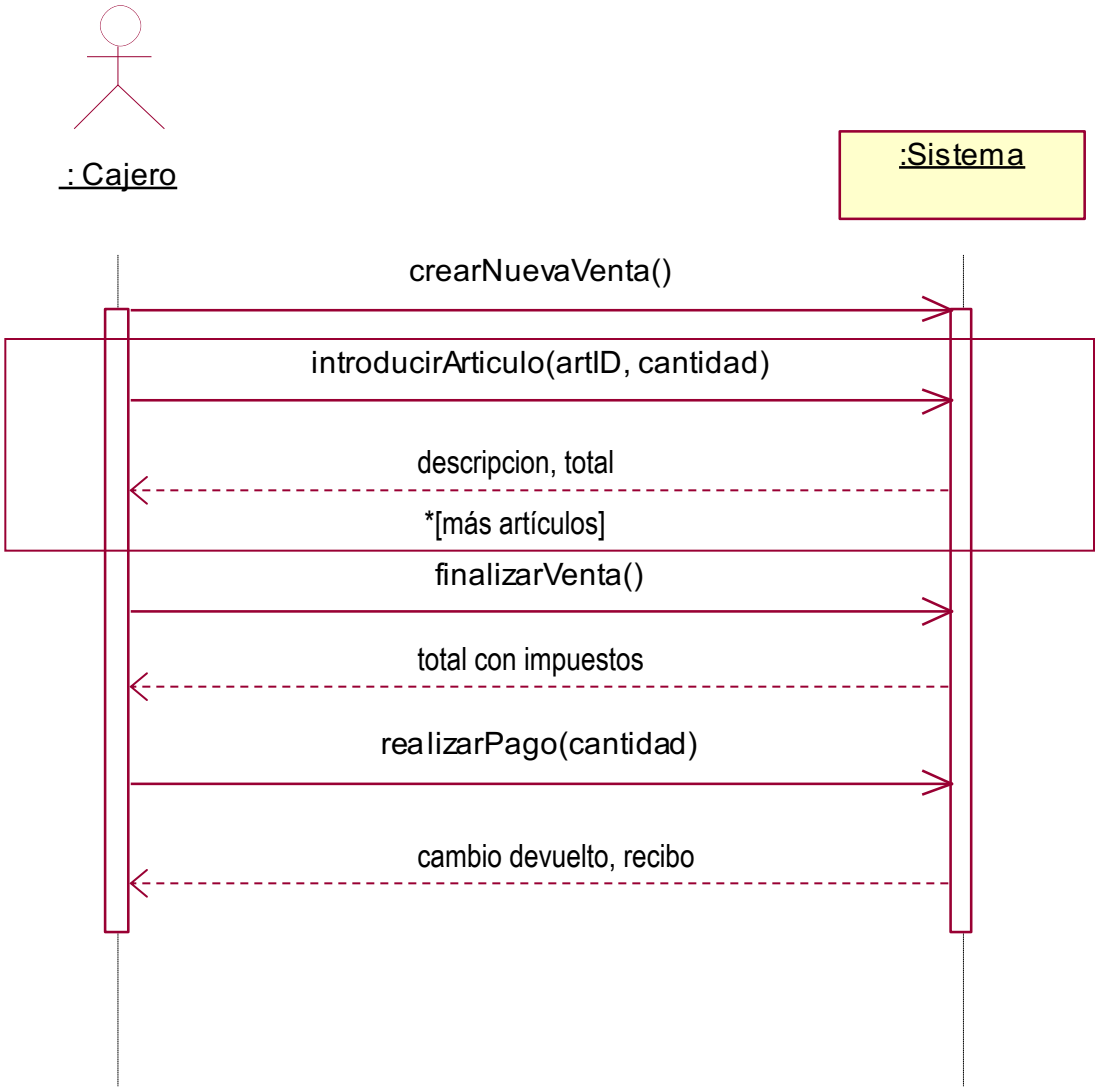
Actividad: Documentar los casos de uso (iv)

- Existen múltiples propuestas para la utilización concreta de los casos de uso como técnica tanto de obtención como de especificación de los requisitos funcionales del sistema
- Para la descripción concreta de los casos de uso se proponen plantillas, en las que las interacciones se numeran siguiendo diversas propuestas y se describen usando lenguaje natural
  - Algunas de estas propuestas son
    - [Schneider y Winters, 2001]
    - [Cockburn, 2000]
    - [Durán y Bernárdez, 2002]

Actividad: Documentar los casos de uso (v)

- Los casos de uso describen cómo interactúan los actores externos con el sistema software a crear
- Sería deseable aislar e ilustrar las operaciones que un actor externo solicita a un sistema
- Un **diagrama de secuencia del sistema** (DSS) muestra, para un escenario específico de un caso de uso, los eventos que generan los actores externos, el orden y los eventos entre los sistemas
- Todos los sistemas se tratan como cajas negras
- Los diagramas destacan los eventos que cruzan los límites del sistema desde los actores a los sistemas
- Debería hacerse un DSS para el escenario principal del caso de uso, así como para los escenarios alternativos complejos o frecuentes [Larman, 2002]

Actividad: Documentar los casos de uso (vi)



[Larman, 2002]

Actividad: Identificar relaciones entre casos de uso (i)

- Incluso en sistemas de tamaño medio pueden aparecer un número elevado de casos de uso
- Una vez elaborado el modelo inicial de casos de uso hay que organizarlos
- Pueden existir similitudes en varios casos de uso que se pueden abstraer en un caso de uso común
  - Se utilizará la relación **«include»** para reducir la redundancia entre casos de uso
- Se puede desear extender un caso de uso sin cambiar la descripción original
  - Se utilizará la relación **«extend»** para separar los flujos de eventos comunes de los excepcionales
- También se pueden encontrar similitudes en actores
- En resumen, se utilizan las relaciones de generalización, inclusión y extensión para factorizar el comportamiento común y las variantes

Actividad: Identificar relaciones entre casos de uso (ii)

- Si se repiten bloques de comportamiento entre casos de uso, puede indicar que existe algo genérico que puede reutilizarse
- Se puede abstraer el comportamiento común en una relación de inclusión
- Factorizar el comportamiento común de los casos de uso presenta grandes ventajas, incluyendo descripciones más cortas y menor redundancia
- El comportamiento **únicamente** puede ser factorizado en un caso de uso separado si es compartido por dos o más casos de uso
- 

La fragmentación excesiva de la especificación del sistema puede conducir a una especificación confusa
- Procedimiento
  - Identificar los pasos de los escenarios que se quieren utilizar en diferentes sitios
  - Agruparlos en un caso de uso y darle nombre
- 

El caso de uso incluido **no puede** tener dependencias con respecto a ningún caso de uso que lo incluya

Actividad: Identificar relaciones entre casos de uso (iii)

- Un caso de uso extiende a otro caso de uso si el caso de uso extendido puede incluir el comportamiento de la extensión bajo ciertas condiciones
- Modela la parte de un caso de uso que el usuario puede ver como un comportamiento opcional del sistema
- Se utiliza para extender de **forma condicionada** el comportamiento de un caso de uso existente
- Es una forma de añadir comportamiento a un caso de uso sin modificarlo
- Se utiliza cuando se trabajo con una versión posterior de un producto existente o para indicar los sitios donde un producto puede adaptarse o personalizarse
- El caso de uso se actualiza **añadiendo** puntos de extensión
- Las extensiones se describen al igual que los casos de uso. Debe existir una condición de entrada (inicio) y de salida
- El caso de uso extendido **no cambia**

Actividad: Identificar relaciones entre casos de uso (iv)

- La generalización indica que un caso de uso es una versión especializada de otro caso de uso
- El caso de uso general puede ser únicamente una descripción, los flujos se especifican en los casos de uso especializados
- Es posible agregar comportamiento al caso de uso hijo añadiendo pasos a la secuencia de comportamiento heredada del padre, así como declarando relaciones de extensión y de inclusión para el hijo
- Si el padre es abstracto, su secuencia de comportamiento puede tener secciones que sean explícitamente incompletas en el padre y que debe proporcionar el hijo
- El hijo puede redefinir pasos heredados del padre
- En la secuencia heredada del padre se pueden intercalar pasos adicionales
- El uso de la generalización múltiple en casos de uso requiere la especificación explícita de la forma en la que se intercalan las secuencias de comportamiento de los padres para crear la secuencia correspondiente al hijo
- El caso de uso base (**A**) es autosuficiente. El caso de uso hijo (**B**) necesita al caso de uso base (obtiene la funcionalidad base de **A**) y controla qué es ejecutado desde **A** y qué cambia

Construcción del Modelo de casos de uso – Resumen (i)

- Cada caso de uso debe representar un comportamiento distinto e identificable del sistema o de una parte del mismo
- Un caso de uso bien estructurado cumple que [Booch et al., 1999]
  - Nombra un comportamiento simple, identificable y razonablemente atómico del sistema o parte del sistema
  - Factoriza el comportamiento común, incorporando ese comportamiento desde otros casos de uso que incluye
  - Factoriza variantes, colocando ese comportamiento en otros casos de uso que lo extienden o lo especializan
  - Describe el flujo de eventos de forma suficientemente clara para que alguien externo al sistema lo entienda fácilmente
  - Se describe por un conjunto mínimo de escenarios que especifican la semántica normal y las variantes del caso de uso

Construcción del Modelo de casos de uso – Resumen (ii)

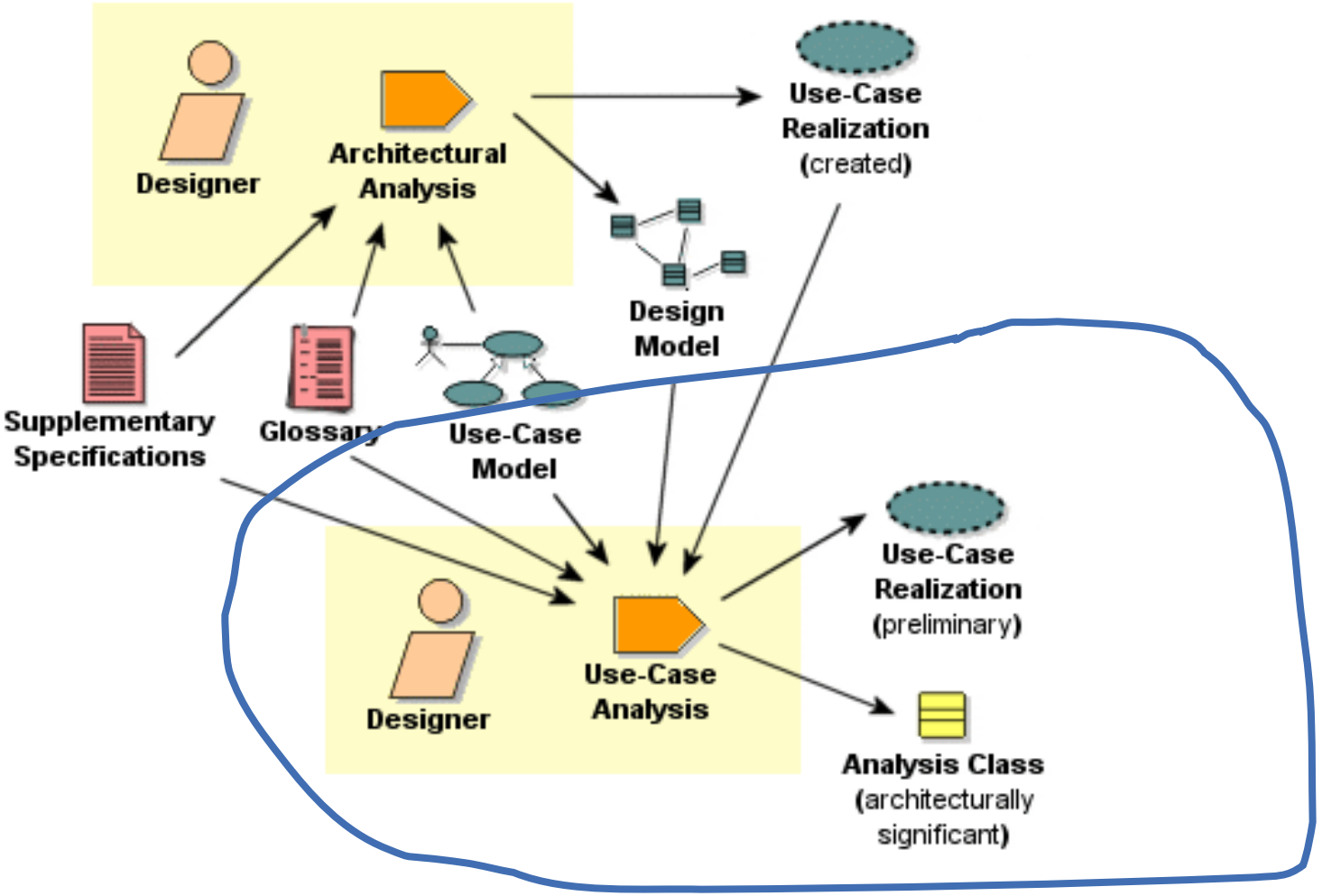
- A la hora de realizar los casos de usos más que preguntarse si un caso de uso es válido o no, habría que preguntarse cuál es el nivel útil para expresar los casos de uso en el análisis de requisitos de una aplicación
- Para el análisis de requisitos de una aplicación informática, hay que centrarse en los casos de uso al nivel de **procesos del negocio elementales** o **EBPs** (*Elementary Business Processes*)
  - Un EBP es un término que procede del campo de la ingeniería de procesos del negocio
    - Es una tarea realizada por una persona en un lugar, en un instante, como respuesta a un evento de negocio, que añade un valor cuantificable para el negocio y deja los datos en un estado consistente [Larman, 2002]
- Un caso de uso de nivel EBP se denomina caso de uso de nivel de objetivo de usuario, para remarcar que sirve (o debería servir) para satisfacer un objetivo de un usuario del sistema o actor principal
  - Procedimiento
    - Encontrar los objetivos de usuario
    - Definir un caso de uso para cada objetivo

Construcción del Modelo de casos de uso – Resumen (iii)

- Por lo general se define un caso de uso de nivel EBP por cada objetivo de usuario
- Se nombra cada caso de uso de forma similar al objetivo de usuario
- Los casos de uso se suelen nombrar comenzando por un verbo
- Una excepción típica a crear un caso de uso por objetivo, es agrupar objetivos separados en un caso de uso CRUD (*Create-Retrieve-Update-Delete* – Crear-Recuperar-Actualizar-Eliminar)
  - Por convención se denomina a este caso de uso **Gestionar<X>**

2. Análisis en el Proceso Unificado

Análisis en el Proceso Unificado



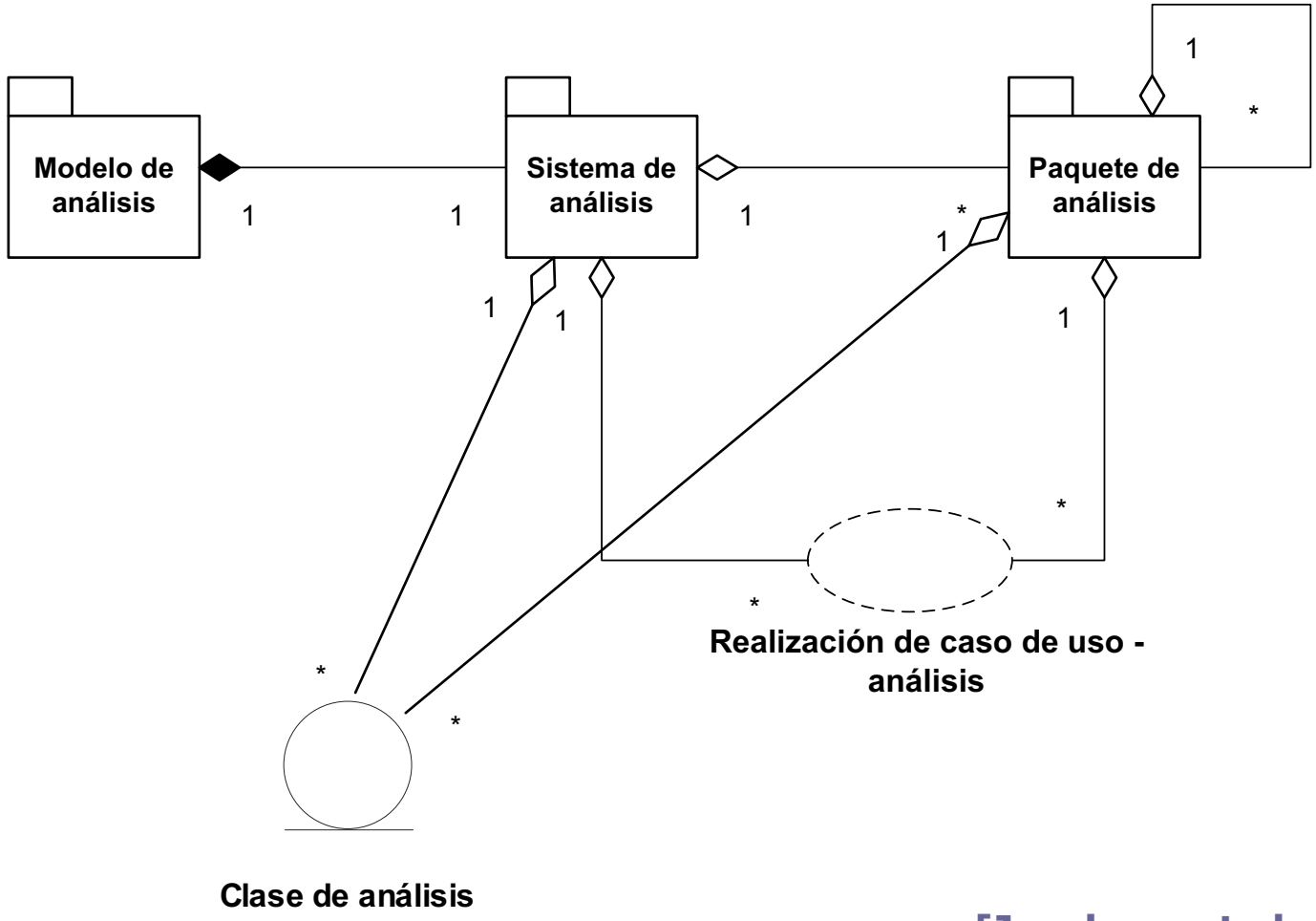
Artefactos propios del análisis en el Proceso Unificado

- Modelo de análisis
- Clase de análisis
- Realización de casos de uso – análisis
- Paquete de análisis
- Descripción de la arquitectura (vista del modelo de análisis)

Modelo de análisis (i)

- Se representa mediante un sistema de análisis que denota el paquete de más alto nivel del modelo
- Se utilizan otros paquetes de análisis para organizar el modelo de análisis en partes más manejables
  - Representan abstracciones de subsistemas y posiblemente capas completas del diseño del sistema
- Las clases de análisis representan abstracciones de clases y posiblemente de subsistemas de diseño del sistema
- Los casos de uso se describen mediante clases de análisis y sus objetos
  - Esto se representa mediante colaboraciones denominadas **realizaciones de caso de uso - análisis**

Modelo de análisis (ii)



[Jacobson et al., 1999]

Clase de análisis (i)

- Representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema
- Características de las clases de análisis
  - Se centran en el tratamiento de los requisitos funcionales
  - Son más evidentes en el contexto del dominio del problema
  - Raramente definen u ofrecen una interfaz en términos de operaciones
  - Pueden definir atributos pero a un nivel bastante alto
  - Participan en relaciones de un claro carácter conceptual
  - Siempre encajan en uno de tres estereotipos básicos
    - **Entidad**
    - **Interfaz**
    - **Control**

Clase de análisis (ii)

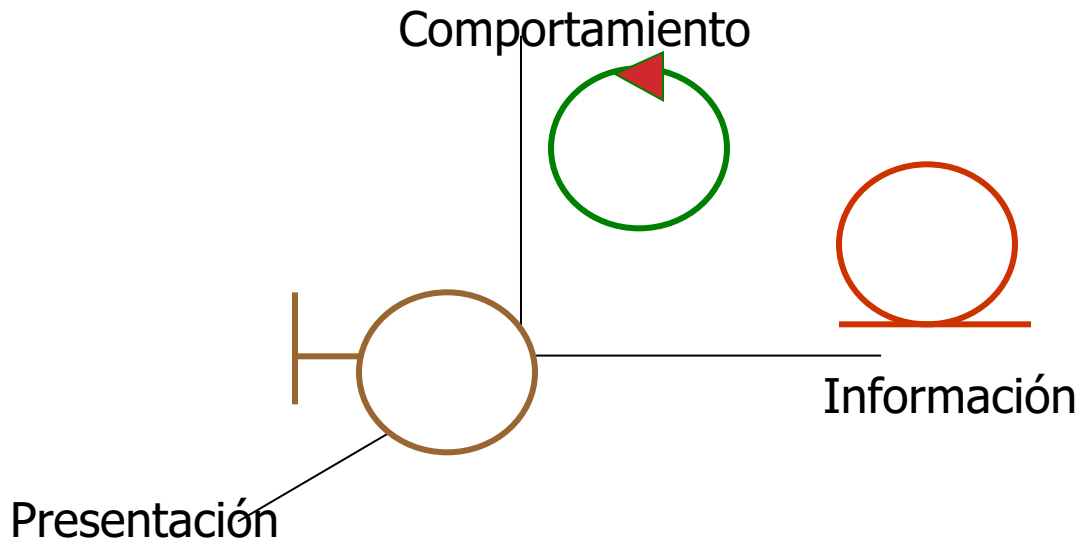
Especificación de objetos en el espacio de información definido por tres ejes

Se eligen tres tipos de objetos a fin de proporcionar una estructura más adaptable

Objetos Entidad: Información persistente sobre la que el sistema realiza un seguimiento

Objetos Interfaz: Representan las interacciones entre el actor y el sistema

Objetos Control: Representan las tareas realizadas por el usuario y soportadas por el sistema

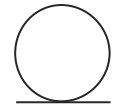


Clase de análisis (iii)

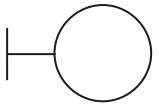
- ¿Por qué estos tres tipos de objetos (entidad, interfaz y control)?
  - Aparecen de forma natural en el texto del caso de uso
  - Se obtienen objetos especializados más pequeños
  - Dan lugar a modelos más resistentes a cambios
    - La interfaz cambia fácilmente
  - Ayudan a la construcción de diagramas de secuencia
  - Los objetos de control sirven de conexión entre los usuarios y los datos almacenados
    - Capturan las reglas de negocio (siempre sujetas a cambios)
    - Sirven de espacio de reserva para garantizar que no se olvida la funcionalidad

Clase de análisis (iii)

- Notación

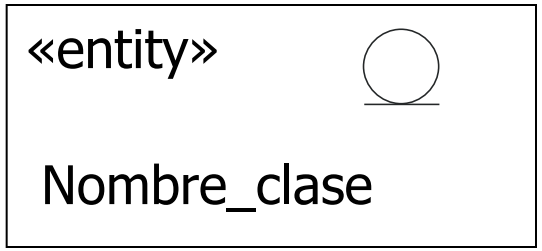


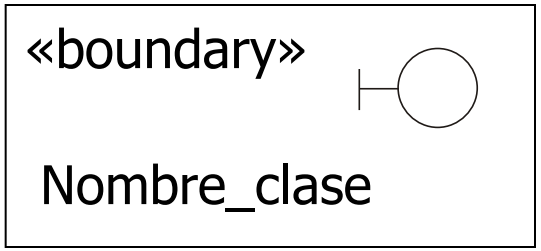
Clase de entidad

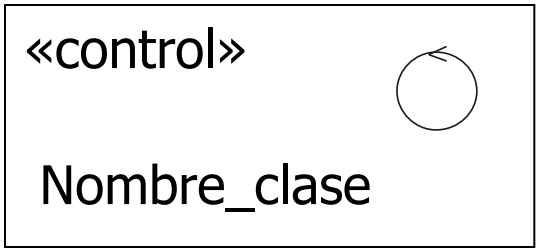


Clase de interfaz

Clase de control







Clase de entidad

- Para modelar la información que el sistema tiene que gestionar se utilizan los objetos entidad
- Esta información permanece en el sistema incluso cuando el caso de uso finaliza
- En el objeto entidad se incorpora, además de la información, el comportamiento asociado a esa información
- Estos objetos se identifican a partir de la descripción de los casos de uso
- Normalmente suelen corresponder a alguno de los conceptos que se manejan en el sistema. **Clases conceptuales**
- Las operaciones identificadas en el objeto tienen que ser suficientes para todas las posibles utilizaciones del objeto
- Hay que evitar en la medida de lo posible que estos objetos sólo sean portadores de información asignando todo el comportamiento dinámico a los objetos control

Clase de interfaz (i)

- Toda la funcionalidad que depende directamente del entorno se asigna a las clases de interfaz
- Modelan las interacciones entre el sistema y sus actores
- Interaccionan con los actores externos al sistema y con las clases del sistema
- Los objetos interfaz se encargan de trasladar las acciones de los actores a eventos en el sistema y éstos en "información" que se presenta al actor
- Representan una abstracción de elementos de la interfaz de usuario (ventanas, formularios, paneles...) o dispositivos (interfaz de impresora, sensores, terminales...)
- Cada actor necesita su/s propia/s interfaz/ces para llevar a cabo sus acciones
- Mantener la descripción a nivel conceptual, es decir, no describir cada botón, ítem de menú... de la interfaz de usuario
- Encapsula y aísla los cambios en la interfaz de usuario

Clase de interfaz (ii)

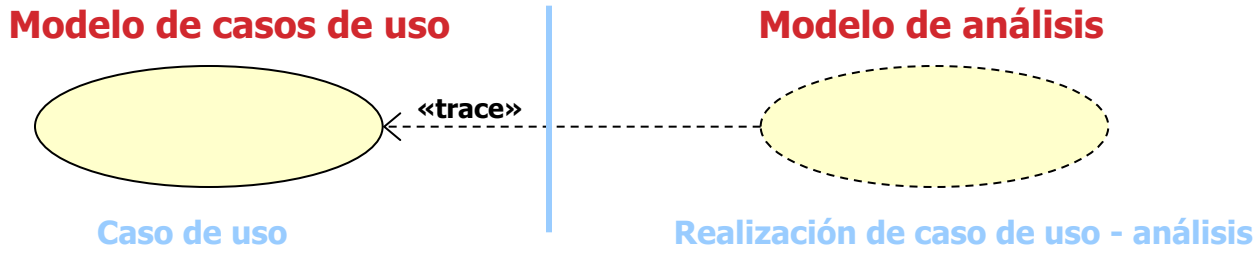
- Estrategias de identificación de clases de interfaz
  - A partir de las descripciones de los casos de uso
  - A partir de las descripciones de las interfaces de usuario
  - A partir de los actores
- Recomendaciones
  - Identificar formularios y ventanas necesarias para la introducción de datos en el sistema
  - Identificar avisos y mensajes a los que tiene que responder el usuario
  - No modelar los aspectos visuales de la interfaz
  - Utilizar siempre términos de usuario para describir las interfaces
- Identificar una **clase de interfaz central** para cada actor humano
  - Representa la "ventana" primaria con la que el actor interacciona
- Identificar una **clase de interfaz central** para cada actor sistema externo
  - Representa la interfaz de comunicación con el sistema externo
- Las clases de interfaz centrales pueden ser **agregaciones** de otras clases de interfaz

Clase de control

- Los objetos de control actúan como unión entre los objetos interfaz y entidad
  - Coordinación entre objetos entidad e interfaz
- No siempre aparecen
  - Normalmente toda la funcionalidad expresada en un caso de uso está asignada a los objetos entidad e interfaz
- Se identifican a partir de los casos de uso
  - Cada caso de uso tiene inicialmente solamente un objeto de control
- El tipo de funcionalidad asignada a estos objetos suele ser el comportamiento relacionado con transacciones o secuencias específicas de control
- El comportamiento de "conexión" entre objetos interfaz y entidad suele asignarse a este tipo de objetos
- Normalmente no se corresponden con entidades reales

Realización de casos de uso – análisis

- Es una colaboración dentro del modelo de análisis que describe cómo se lleva a cabo y se ejecuta un caso de uso en término de las clases de análisis y de sus objetos
- Ofrece una traza directa hacia un caso de uso concreto del modelo de casos de uso
- Una realización de un caso de uso posee
  - Una descripción del flujo de sucesos
  - Diagramas de clase de análisis
  - Diagramas de interacción



Paquete de análisis

- Es el medio para organizar los artefactos del modelo de análisis en piezas manejables
- Puede constar de clases de análisis, de realización de casos de uso, y de otros paquetes del análisis (recursivamente)
- Estos paquetes deben ser cohesivos y débilmente acoplados
- Pueden representar una separación de intereses de análisis
- Deben crearse basándose en los requisitos funcionales en el dominio del problema, y deben ser reconocibles por las personas con conocimiento del dominio
- Se suelen convertir en subsistemas en las (dos) capas superiores del modelo de diseño

Descripción de la arquitectura

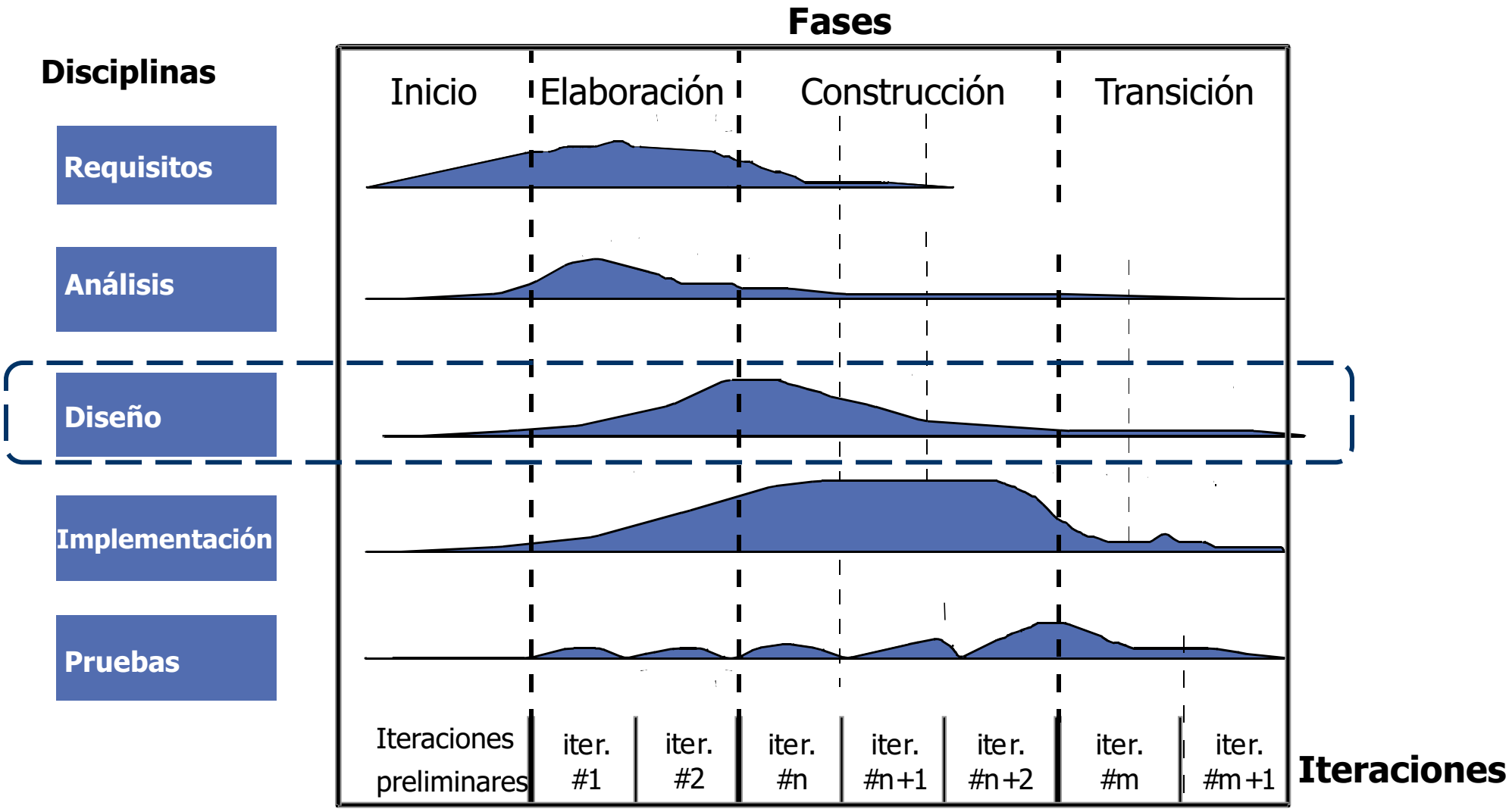
- Contiene una vista de la arquitectura del modelo de análisis, que muestra sus artefactos significativos para la arquitectura
  - Descomposición del modelo de análisis en paquetes de análisis y sus dependencias
  - Las clases fundamentales del análisis
  - Realizaciones de casos de uso que describen cierta funcionalidad importante y crítica

Proceso Unificado: Actividades del análisis

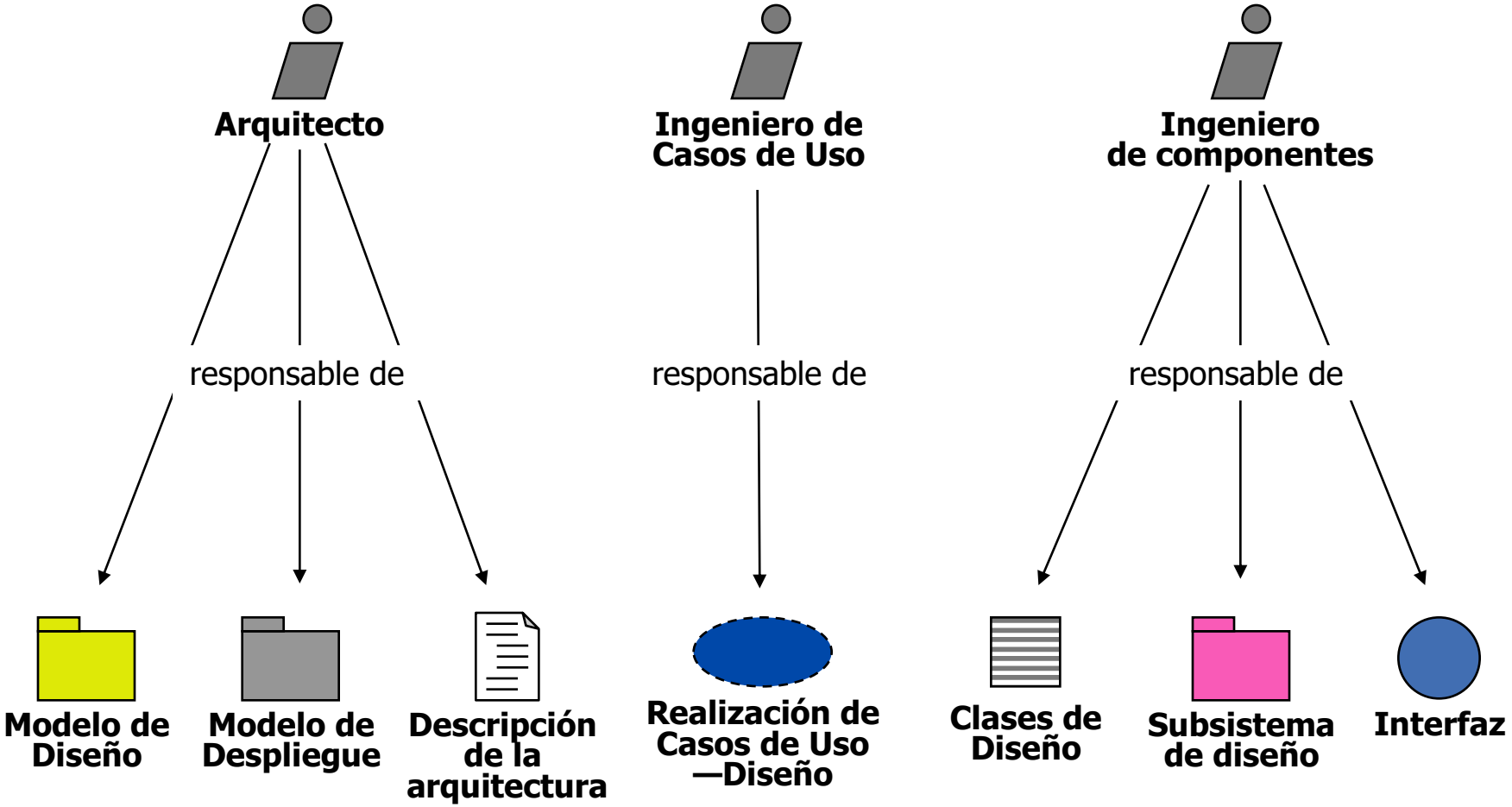
- Completar las descripciones de los casos de uso
  - Incluir los detalles internos de la actividad del sistema en respuesta a las acciones de los actores
- Para cada realización de caso de uso
  - Encontrar las **clases de análisis** a partir del comportamiento del caso de uso
  - Distribuir el comportamiento entre las clases de análisis
- Para cada clase de análisis resultante
  - Describir las responsabilidades
  - Describir atributos y asociaciones
  - Definir atributos
  - Establecer las asociaciones entre las clases de análisis
  - Describir Dependencias entre clases de análisis
- Evaluar el resultado del análisis de casos de uso

3. Diseño en el Proceso Unificado

Disciplina de diseño



Trabajadores y artefactos involucrados en el diseño



[Jacobson et al., 1999]

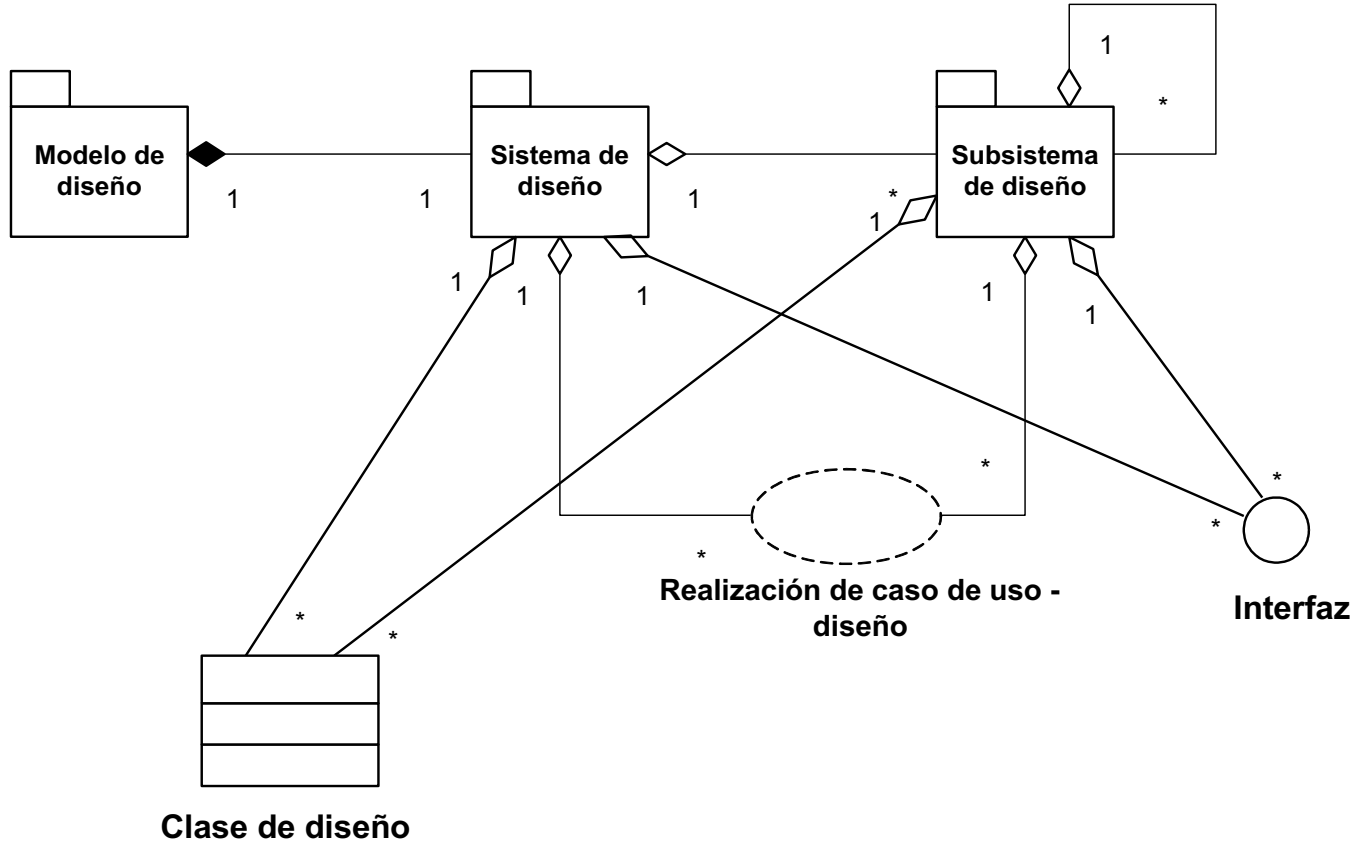
Artefactos propios del diseño en el Proceso Unificado

- Modelo de diseño
- Clase de diseño
- Realización de casos de uso – diseño
- Subsistema de diseño
- Interfaz
- Modelo de despliegue
- Descripción de la arquitectura

Modelo de diseño (i)

- Es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar
- El modelo de diseño se representa por un sistema de diseño que denota el subsistema de nivel más alto de modelado
  - La utilización de otro subsistema es una forma de organización del modelo de diseño en partes más manejables
- Los subsistemas de diseño y las clases de diseño representan **abstracciones** del subsistema y componentes de la implementación del sistema
- Los casos de uso son realizados por los objetos de las clases de diseño

Modelo de diseño (ii)



[Jacobson et al., 1999]

Clase de diseño

- Representa una abstracción de una o varias clases (o construcción similar) en la implementación del sistema
- Esta abstracción es sin costuras debido a [Jacobson et al., 1999]
  - El lenguaje utilizado para especificar una clase de diseño es el mismo que el lenguaje de programación utilizado
  - Se especifica la visibilidad de atributos y operaciones, utilizando con frecuencia los términos derivados de C++
  - Las relaciones entre clases de diseño suelen tener un significado directo cuando la clase es implementada
  - Los métodos de una clase de diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases
  - Una clase de diseño puede aparecer como un estereotipo que se corresponde con una construcción en el lenguaje de programación dado
  - Una clase de diseño se puede realizar, esto es, proporcionar interfaces si tiene sentido hacerlo en el lenguaje de programación
  - Una clase de diseño se puede activar, implicando que objetos de la clase mantengan su propio hilo de control y se ejecuten concurrentemente con otros objetos activos

Realización de caso de uso – diseño

- Es una colaboración en el modelo de diseño que describe cómo se realiza un caso de uso específico
- Una vista del modelo de diseño centrado en los siguientes artefactos significativos desde el punto de vista de la arquitectura
  - Diagramas de clases: Clases de diseño, sus características y relaciones
  - Diagramas de interacción: Diagramas de secuencia, diagramas de estado
  - Flujo de eventos – diseño
  - Requisitos de implementación

Subsistema de diseño

- Son una forma de organizar los artefactos del modelo del diseño en piezas más manejables
- Deben ser una colección cohesiva de clases de diseño, realizaciones de caso de uso, interfaces y otros subsistemas
- El acoplamiento entre subsistemas ha de ser mínimo
- Utilizados para separar los aspectos del diseño
- Las dos capas de aplicación de más alto nivel y sus subsistemas dentro del modelo de diseño suelen tener trazas directas hacia paquetes del análisis
- Los subsistemas pueden representar componentes de grano grueso en la implementación del sistema, es decir, componentes que proporcionan varias interfaces a partir de otros elementos de grano más fino, y que se convierten ellos mismos en ejecutables, ficheros binarios o entidades similares que pueden distribuirse en diferentes nodos
- Pueden representar productos software reutilizados que han sido encapsulados en ellos
- Pueden representar sistemas heredados o partes de ellos

Interfaz

- Especifica una colección de operaciones públicas, tipos y parámetros necesarios para acceder y usar las capacidades de una clase de diseño o un subsistema
- Una clase de diseño que realice una interfaz debe proporcionar los métodos que implementen las operaciones de la interfaz
- Un subsistema que realice una interfaz debe conectar también clases del diseño u otros subsistemas (recursivamente) que proporcionen la interfaz
- Las interfaces son una forma de separar la especificación de la funcionalidad
- La mayoría de las interfaces entre subsistemas se consideran relevantes para la arquitectura debido a que definen las interacciones permitidas entre los subsistemas

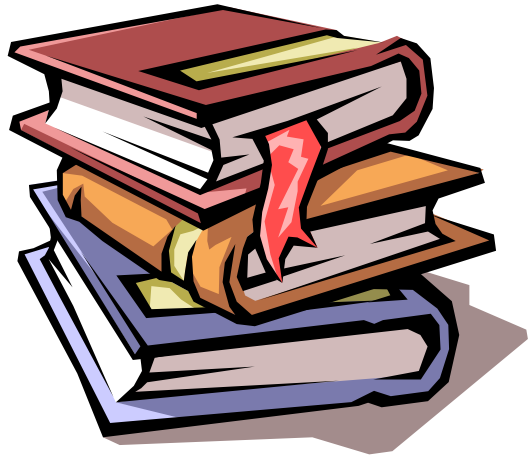
Modelo de despliegue

- Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo
  - Correspondencia entre la arquitectura software y la arquitectura del sistema
- Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar
- Los nodos poseen relaciones que representan medios de comunicación entre ellos
- El modelo de despliegue puede describir diferentes configuraciones de red
- La funcionalidad de un nodo se define por los componentes que se distribuyen en ese nodo

Descripción de la arquitectura

- Contiene una vista de la arquitectura del modelo de diseño que muestra sus artefactos relevantes para la arquitectura
  - Subsistemas, sus interfaces y las dependencias entre ellos
  - Clases de diseño fundamentales con una traza a las clases de análisis significativas y clases activas
  - Realizaciones de caso de uso – diseño que describan una funcionalidad importante y crítica y que deban desarrollarse pronto en el ciclo de vida
- Contiene una vista de la arquitectura del modelo de despliegue que muestra los artefactos relevantes para la arquitectura

4. Referencias



Referencias

[Booch et al., 1999] Booch, G., Rumbaugh, J., Jacobson, I. "*El Lenguaje Unificado de Modelado*". Addison Wesley, 1999

[Bruegge y Dutoit, 2000] Bruegge, B., Dutoit, A. "*Object-Oriented Software Engineering*". Prentice-Hall, 2000

[Cockburn, 2000] Cockburn, A. "*Writing Effective Use Cases*". Addison-Wesley, 2000

[Durán y Bernárdez, 2002] Durán, A., Bernárdez, B. "*Metodología para la Elicitación de Requisitos de Sistemas Software (versión 2.3)*". Informe Técnico LSI-2000-10, Universidad de Sevilla. http://www.lsi.us.es/%7Eamador/publicaciones/metodologia_elicitacion_2_3.pdf.zip. [Última vez visitado, 26-3-2014]. Abril 2002

[Jacobson et al., 1999] Jacobson, I., Booch, G., Rumbaugh, J. "*The Unified Software Development Process*". Object Technology Series. Addison-Wesley, 1999

[Larman, 2002] Larman, C. "*Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*". 2nd Ed. Prentice Hall, 2002

[Schneider y Winters, 2001] Schneider, G., Winters, J. P. "*Applying Use Cases: A Practical Guide*". 2nd Ed. Addison-Wesley, Object Technology Series, 2001

INGENIERÍA DE SOFTWARE I

Tema 6: Flujos de trabajo del Proceso Unificado

2º G.I.I.
Fecha de última modificación: 20-2-2018

Dr. Francisco José García Peñalvo / fgarcia@usal.es
Alicia García Holgado / aliciagh@usal.es

Departamento de Informática y Automática
Universidad de Salamanca

VNiVERSiDAD D SALAMANCA
CAMPUS OF INTERNATIONAL EXCELLENCE

GRIAL Research Group University of Salamanca