

WORK FLOW: ANÁLISIS INTRODUCCIÓN

DISEÑO Y CONSTRUCCIÓN DE SOFTWARE



Introducción

■ Objetivos análisis

- ◆ **Analizar los requisitos en mayor profundidad**
 - Comprensión más precisa de los requisitos
 - Razonar sobre los aspectos internos del sistema
- ◆ **Refinarlos**
 - Usar un lenguaje más formal para apuntar detalles
 - Lenguaje de los desarrolladores
 - Descripción más fácil de mantener
- ◆ **Estructurarlos**
 - Facilitan su comprensión, su preparación, modificación y su mantenimiento

Introducción

- Consigue ...
 - ◆ Crear un modelo que captura el comportamiento deseado del sistema
 - Modelo de análisis
- Modelo de análisis
 - ◆ Qué necesita hacer el sistema
 - ◆ No cómo lo hará
 - Workflow de diseño

Introducción

Modelo de casos de uso	Modelo de análisis
Descripción con el lenguaje del cliente	Descripción con el lenguaje del desarrollador
Vista externa de sistema	Vista interna del sistema
Estructurado por los casos de uso; proporciona la estructura a la vista externa	Estructurado por clase y paquete estereotipados; proporciona la estructura a la vista interna
Contrato entre el cliente y los desarrolladores sobre qué debería y qué no debería hacer el sistema	Utilizado por los desarrolladores para comprender cómo debería darse forma al sistema (diseñarse e implementarse)

Introducción

Modelo de casos de uso	Modelo de análisis
Puede contener redundancias, inconsistencias, etc., entre requisitos	No debería contener redundancias, inconsistencias, etc., entre requisitos
Captura la funcionalidad del sistema, incluida la funcionalidad significativa del sistema	Esboza cómo llevar a cabo la funcionalidad dentro del sistema; sirve como primera aproximación al diseño
Define casos de uso que se analizarán con más profundidad en el modelo de análisis	Define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso

Análisis ≠ Diseño e Implementación

■ ¿Podemos

- ◆ Analizar los requisitos al mismo tiempo que diseñamos e implementamos el sistema?



Analizar



Implementar



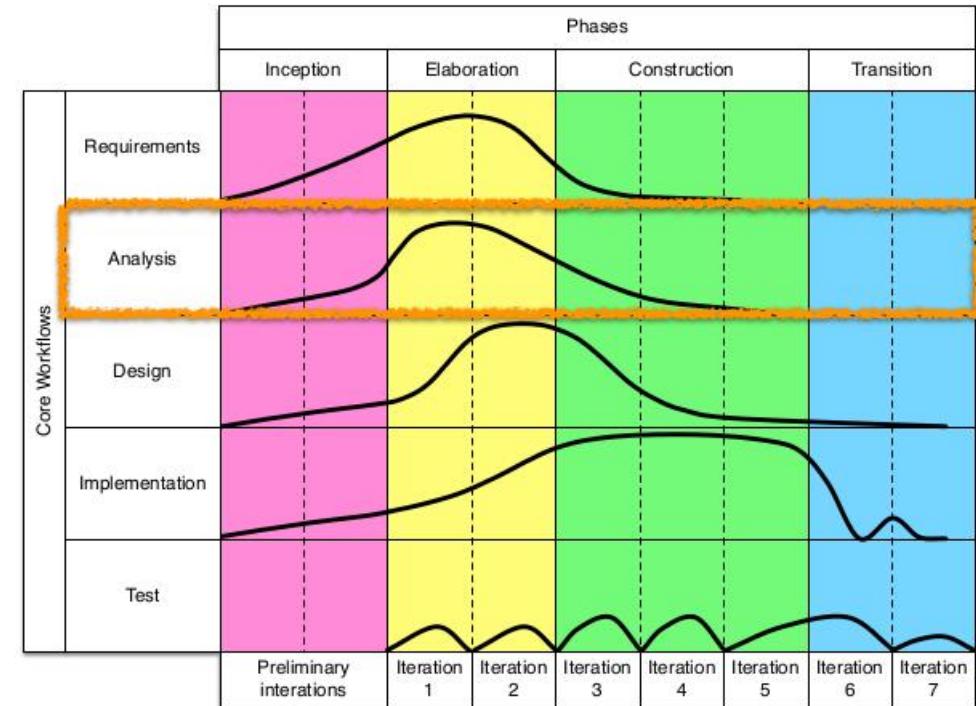
Diseñar

Análisis ≠ Diseño e Implementación

- Necesitamos...
 - ◆ Comprensión precisa y detallada de los requisitos
 - Nivel de detalle que no preocupa al cliente
 - ◆ Estructura de los requisitos
 - Entrada para dar forma al sistema
 - Conseguimos ...
 - ◆ Separación de intereses
 - Prepara y simplifica el diseño y la implementación
 - ◆ Afrontar el proyecto con más tranquilidad

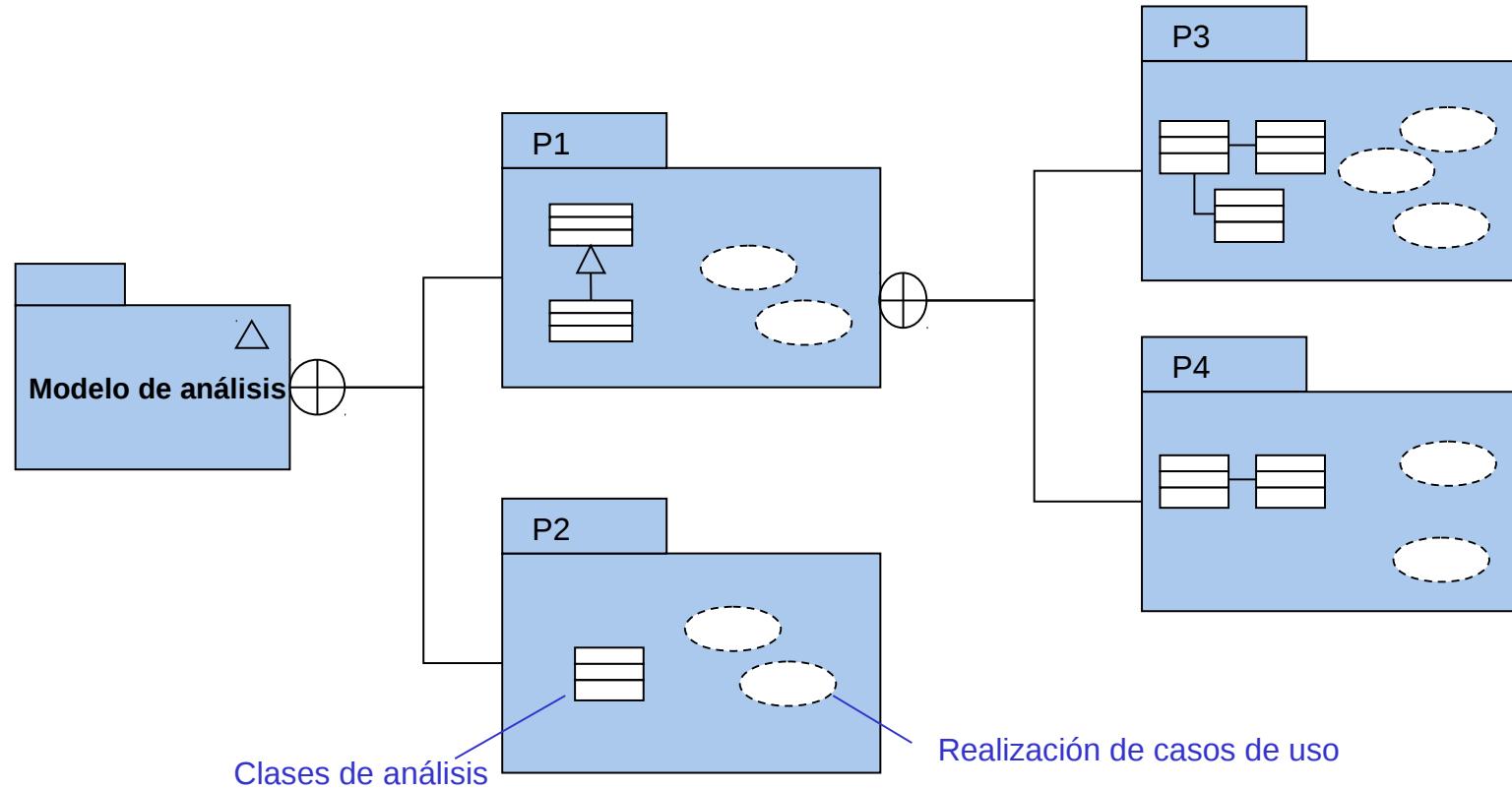
Papel de análisis en el ciclo de vida de un software

- Empieza ...
 - ◆ Al final de la fase de Comienzo
- Foco principal ...
 - ◆ Fase de Elaboración
- Continúa ...
 - ◆ Primeras iteraciones de la fase de construcción



Artefactos

■ Metamodelo



Artefactos (Clases de análisis)

■ Clases de análisis

- ◆ Modelan conceptos clave en el dominio de negocio
- ◆ Representan una abstracción de clases y/o subsistemas del diseño del sistema

■ Características

- ◆ Se centran en el tratamiento de requisitos funcionales
 - Los no funcionales se posponen al diseño/implementación
- ◆ Es más conceptual, de mayor granularidad

Artefactos (Clases de análisis)

■ Características

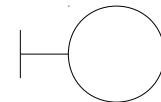
- ◆ El comportamiento se define mediante responsabilidades
 - Descripción textual menos formal y a un nivel más alto
- ◆ Define atributos, pero a alto nivel
 - Los tipos son conceptuales y reconocibles en el dominio del problema
 - Con frecuencia pasan a ser clases en el diseño/impl.
- ◆ Participa en relaciones, pero son más conceptuales
 - La navegabilidad no es importante
 - Pueden utilizarse generalizaciones

Artefactos (Clases de análisis)

■ Tres tipos básicos

◆ Clases de interfaz

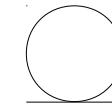
- Modelan la interacción entre el sistema y sus actores
- Implican recibir información y peticiones
 - Describir qué, no cómo se ejecuta físicamente
- Clarifican y reúnen los requisitos en los límites del sistema
- Representan abstracciones de ventanas, formularios, interfaces de impresoras
- Relacionarse, al menos, con un actor



Artefactos (Clases de análisis)

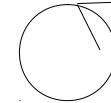
■ Tres tipos básicos

◆ Clases de entidad



- Modelan información que es persistente
- Modelan la información y el comportamiento asociado a una persona, objeto o suceso del mundo real
- Tienen una estructura de datos lógica y ayudan a comprender de qué información depende el sistema

◆ Clases de control



- Representan coordinación, secuencia, transacciones y control de otros objetos
- Encapsulan el control de un caso de uso concreto
- Modelan aspectos dinámicos del sistema

Artefactos (Realización de casos de uso)

- Definición
 - ◆ Colaboración dentro del modelo de análisis
 - Ilustra cómo las instancias de clases de análisis pueden interactuar para realizar el comportamiento del sistema especificado por un caso de uso
- Proporciona
 - ◆ Una traza directa hacia un caso de uso concreto
- Posee
 - ◆ Descripción textual del flujo de sucesos
 - ◆ Diagrama de clases
 - ◆ Diagramas de interacción

Artefactos (Paquete de análisis)

- Proporcionan un medio para organizar los artefactos del modelo de análisis
 - ◆ Clases de análisis
 - ◆ Realizaciones de casos de uso
 - ◆ Paquetes de análisis
- Características
 - ◆ Deben de ser cohesivos
 - Contenidos fuertemente relacionados
 - ◆ Débilmente acoplados
 - Dependencias entre paquetes mínimas

Trabajadores

■ Arquitecto

- ◆ **Responsable de**
 - Integridad del modelo de análisis
 - Correcto, consistente y legible como un todo
 - Arquitectura del modelo de análisis
- ◆ **No es responsable**
 - Del desarrollo y mantenimiento de los diferentes artefactos



Arquitecto

Trabajadores

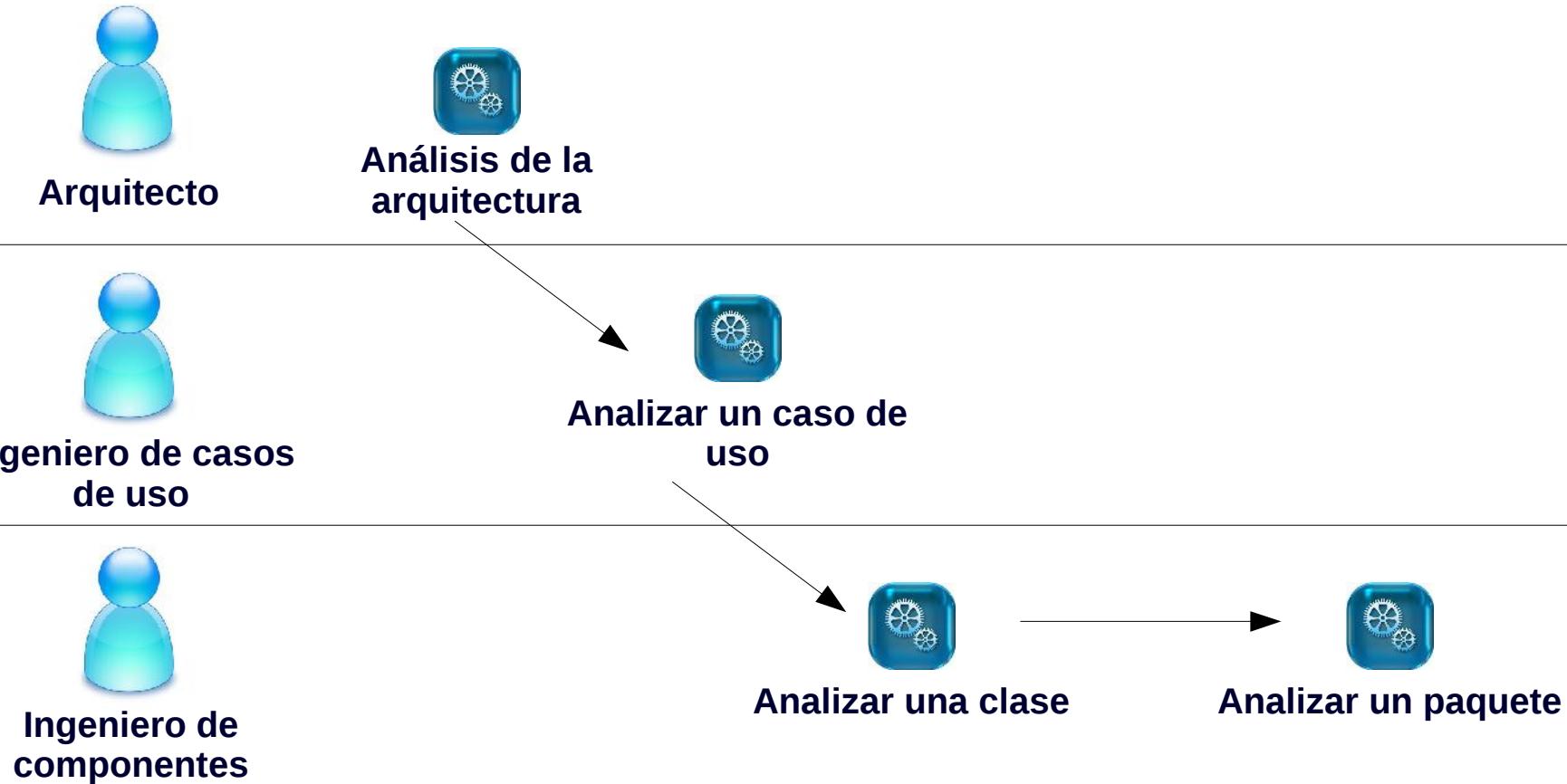
■ Ingeniero de casos de uso

- ◆ **Responsable de**
 - Integridad de una o más realizaciones de casos de uso
- ◆ **No es responsable**
 - De las clases de análisis ni de las relaciones

■ Ingeniero de componentes

- ◆ **Responsable de**
 - Mantener la integridad de uno varios paquetes
 - Garantizar que sus contenidos son correctos y sus dependencias con otros paquetes son mínimas y correctas
 - De las clases de análisis contenidas en él
 - Atributos, relaciones, requisitos especiales

Flujo de trabajo



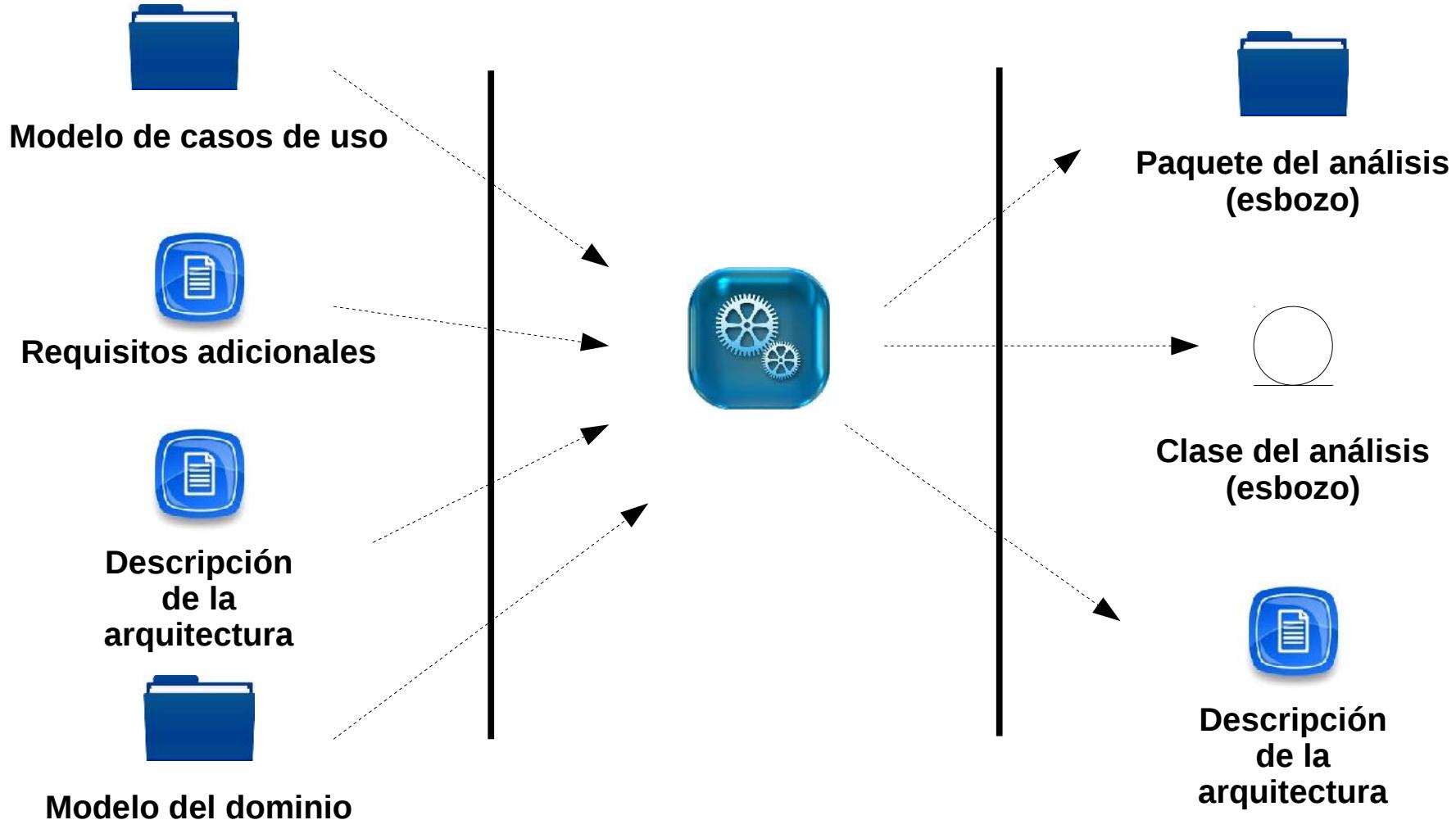
Actividades

■ Análisis de la arquitectura

- ◆ Esbozar el modelo de análisis y la arquitectura
- ◆ Identificar
 - Paquetes de análisis
 - Clases de análisis
 - Requisitos especiales comunes

Actividades

■ Análisis de la arquitectura



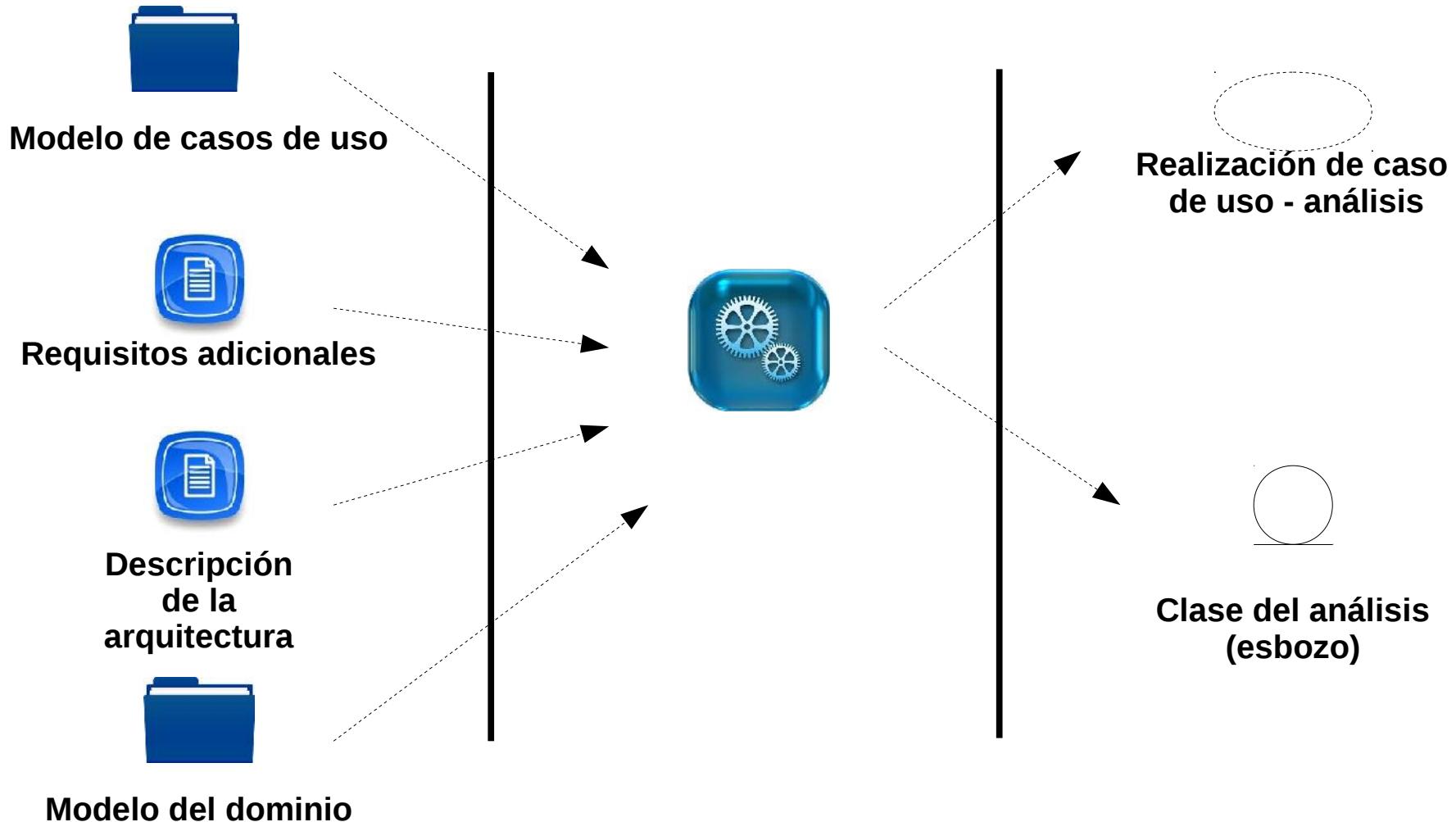
Actividades

■ Analizar un caso de uso

- ◆ Identificar las clases de análisis
 - Objetos que son necesarios para llevar a cabo el flujo de sucesos del caso de uso
- ◆ Distribuir el comportamiento del caso de uso entre los objetos del análisis que interactúan
- ◆ Capturar requisitos especiales sobre la realización del caso de uso

Actividades

■ Analizar un caso de uso



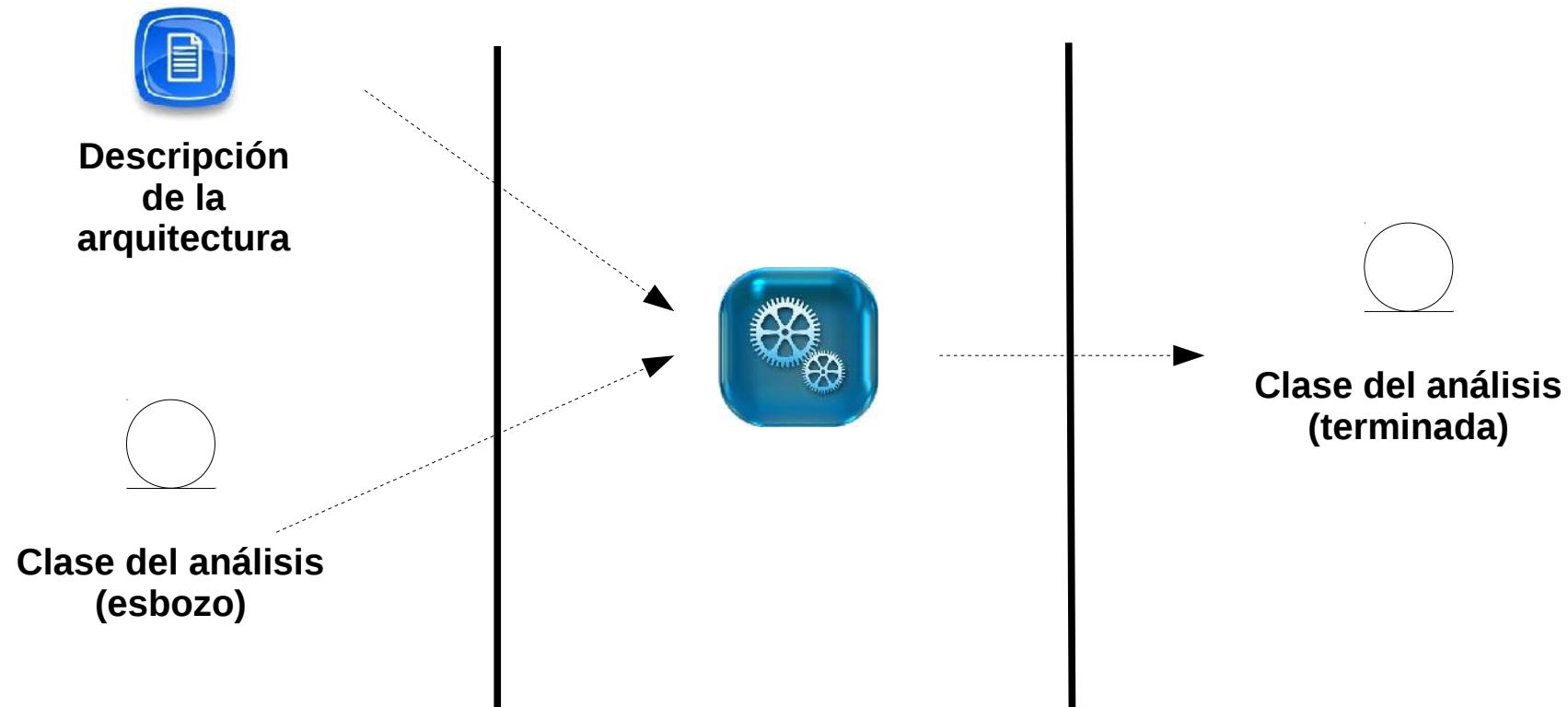
Actividades

■ Analizar una clase

- ◆ Identificar y mantener las responsabilidades de una clase del análisis
- ◆ Identificar y mantener los atributos y relaciones de la clase del análisis
 - Asociaciones
 - Agregaciones
 - Generalizaciones
- ◆ Capturar requisitos especiales sobre la realización de la clase del análisis

Actividades

■ Analizar una clase



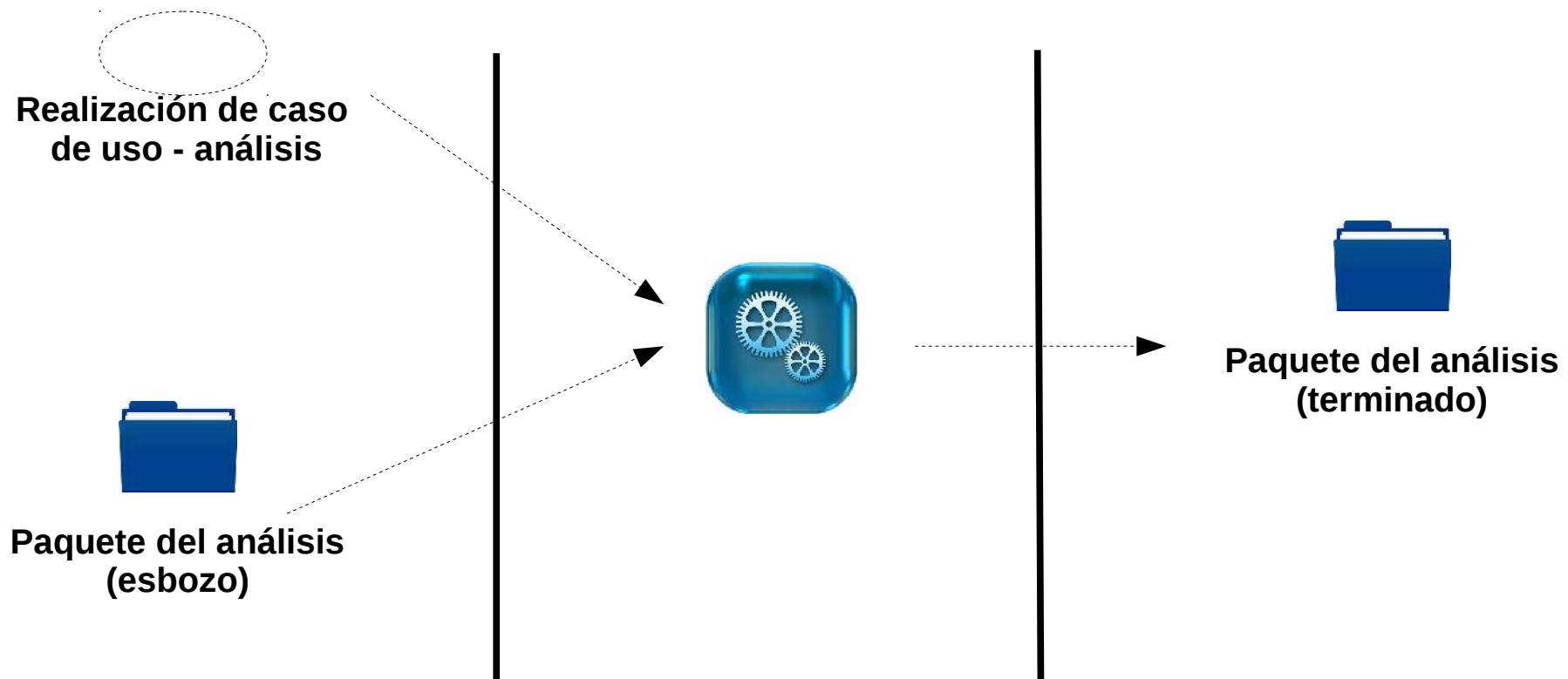
Actividades

■ Analizar un paquete

- ◆ Garantizar que el paquete es tan independiente de otros paquetes como sea posible
 - Limitar las dependencias con otros paquetes
 - Reubicar clases
- ◆ Garantizar que el paquete cumple su objetivo de realizar alguna de las clases del dominio o casos de uso
 - Contiene las clases correctas
 - Paquete cohesivo
- ◆ Describir las dependencias
 - Estimarse el efecto de los cambios futuros

Actividades

■ Analizar un paquete



Reglas generales

- El modelo de análisis → lenguaje del negocio
 - ◆ Abstracciones del modelo de análisis deben formar parte del **vocabulario del dominio del negocio**
- Modelos que cuenten una historia
 - ◆ Aclarar alguna parte importante del comportamiento deseado del sistema
- Capturar la idea general
 - ◆ No atascarse en detalles de cómo funcionará el sistema

Reglas generales

- Distinguir entre el dominio del problema y el dominio de la solución
 - ◆ Requisitos del problema
 - Clases: Cliente, Pedido, CarroCompra
 - ◆ Consideraciones detalladas de diseño
 - Bases de datos, comunicaciones
- Minimizar el acoplamiento
 - ◆ Las clases relacionadas deben conocer lo mínimo unas de otras
- Herencia si hay jerarquía natural
 - ◆ No por reutilizar código