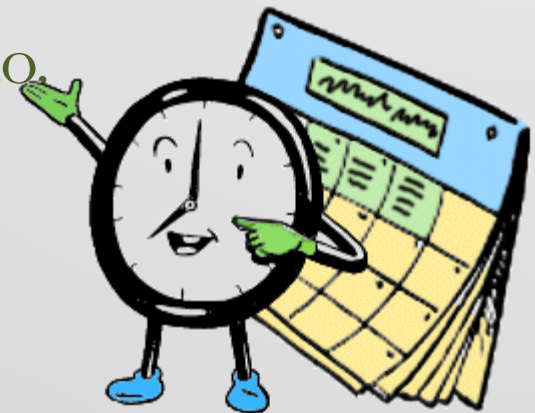




# Agenda

- Un breve Tour por UML
  - Un poco de Historia
  - Diagramas Estructurales.
  - Diagramas de Comportamiento.
- PATRONES



## QUÉ ES UML?

UML (Unified Modeling Language) es un conjunto de modelos estándar utilizados para el diseño de proyectos.

UML no describe la implementación de esos modelos.

# Un poco de historia...

- 1960-1969 Simula.
- 1970-1979 Smalltalk-72.
- 1980 Smalltalk-80.
- 1982 OOA (Booch)
- 1984 Objective-c, Object Lisp.
- 1985 C ++.
- 1986 Oodbms`s.
- 1987 Self – Eifel
- 1988 Primeras Oodbms comerciales.
- 1989 Responsibility driven design (Wirfs).
- 1990 OOA (Yourdon).
- 1991 OMT (Rumbaugh)

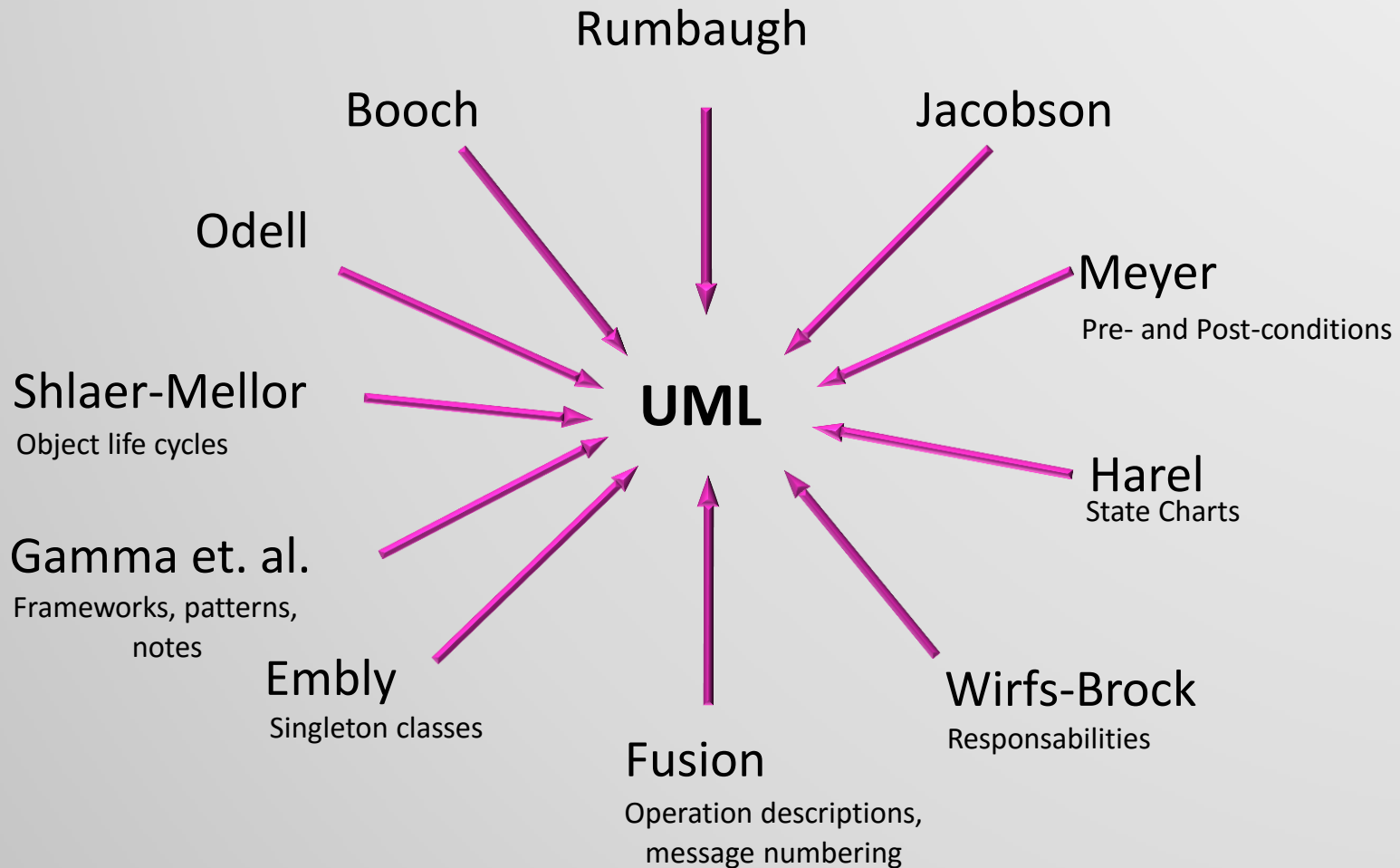
## Más de historia...

- 1991 OMT (Object Modeling Technique Rumbaugh)  
OOA (Booch)  
Primeros herramientas case.
- 1992 Objectory (Jacobson)  
OBA (Object Behaviour Analysis Goldberg, Rubin)
- 1993 Patterns.
- 1994 Design patterns (Gamma).
- 1994 Primera reunión de Booch y Rumbaugh. Rational Rose (primeros pasos de UML).
- 1995 Incorporación de Jacobson (OOSE).

## Más de historia...

- 1996 Object management group (OMG).  
Publicó una petición de propuestas para un enfoque estándar sobre el modelado orientado a objetos.
- 1997 Adoptó UML como un lenguaje de modelado unificado.
- 1998 Se publica el Proceso Unificado de Rational (RUP)
- 1999 UML 1.3
- 2003 UML 1.5
- 2004 UML 2.0

# UML aglutina enfoques OO



## Booch, Jacobson y Rumbaugh se fijaron cuatro objetivos:

- Representar sistemas completos (más allá de un solo programa) por conceptos de objetos;
- establecer una relación explícita entre los conceptos y los artefactos ejecutables;
- tener en cuenta los factores de escala inherentes a los sistemas complejos y críticos;
- crear un lenguaje de modelado utilizable tanto por los humanos como por las máquinas.



# Aspectos Novedosos

- Definición **semi-formal** del Meta-Modelo semántico asociado
- Incluye “estereotipos” como mecanismo de extensibilidad. Mediante estereotipos los elementos de modelado pueden diferenciarse de acuerdo a un uso particular
- Incluye un lenguaje para expresar restricciones mediante fórmulas bien formadas. OCL (Object Constraint Language) desarrollado por IBM

# Métodos Formales en Modelado

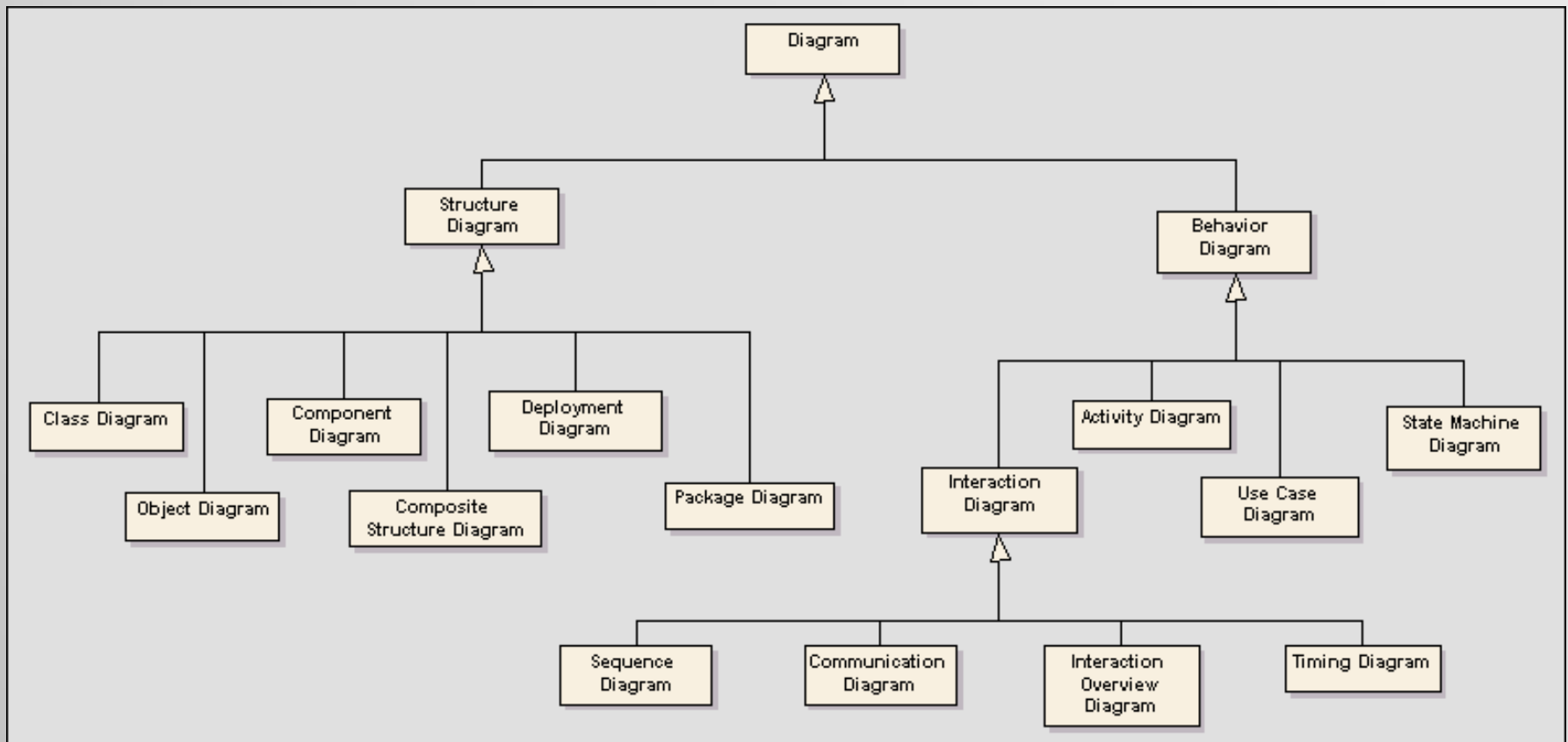
- Tipos de enfoques:
  - *no-formales*, usando lenguaje natural
  - *semi-formales*, notaciones (en parte graficas) con ciertas reglas y cuyas construcciones tienen una semántica más o menos precisa
  - *formales*, usando una notación gráfica o textual basada en un sistema formal (soporte matemático)
- Los métodos formales permiten determinar y expresar con mayor rigor las propiedades del software.

# ... Métodos Formales en Modelado

- Las principales mejoras al utilizar métodos formales son:
  - Mayor rigor en la especificación
  - Mejores condiciones para realizar la verificación y validación en forma más exhaustiva
  - Mejores condiciones para automatización de procesos de generación automática de prototipos y/o código final

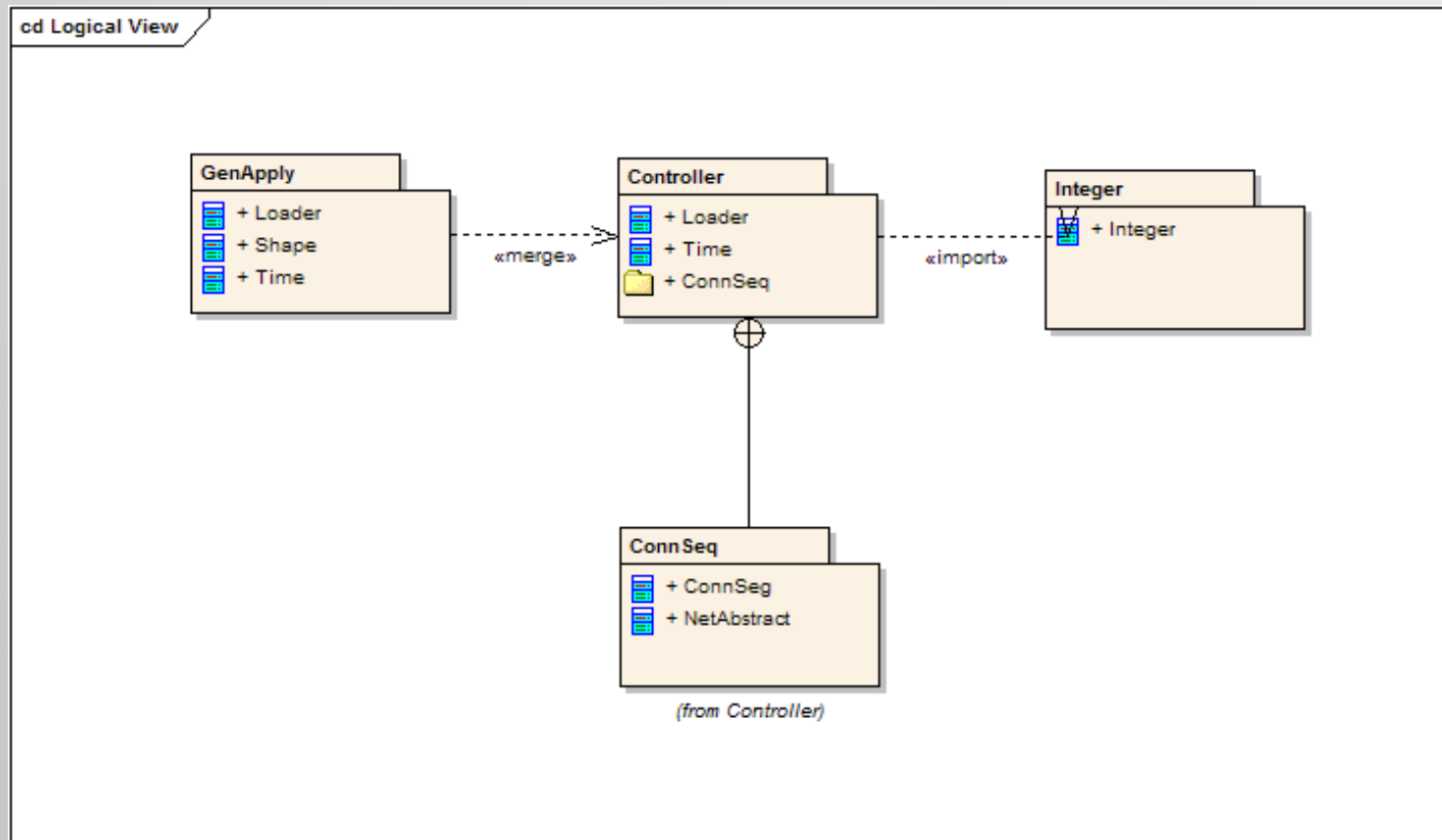
# Los diagramas son vistas del modelo

Un *modelo* es una descripción completa de un sistema desde una perspectiva en particular. Aquí se muestran los 13 diagramas definidos por la OMG en la especificación UML 2.0. Los diagramas estructurales muestran una vista estática del modelo, mientras los de comportamiento muestran la vista dinámica.

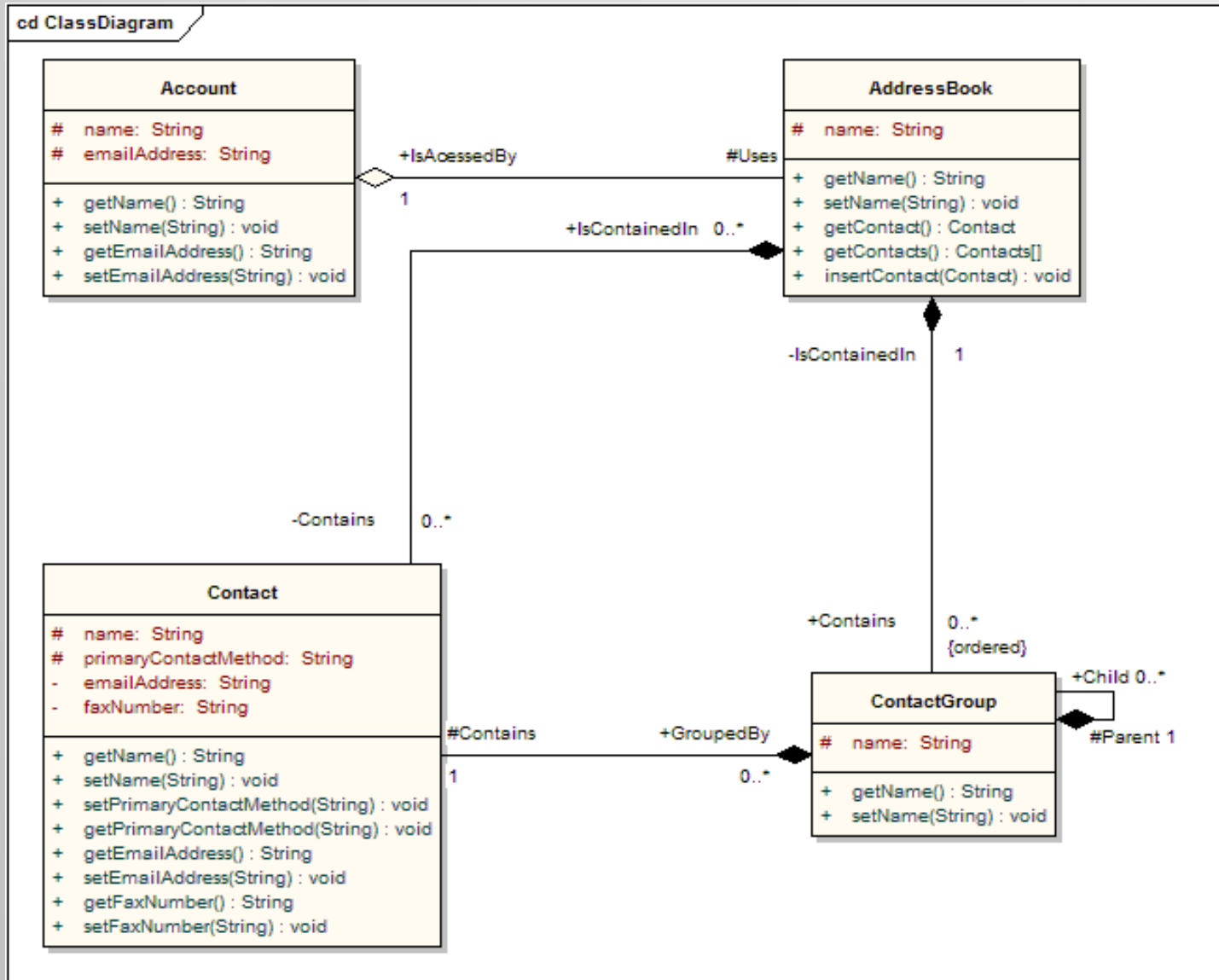


# Diagramas de Paquetes

Reflejan la organización de paquetes y sus elementos. Cuando se lo utilizan con clases, sirven para visualizar los namespaces, otro uso común es para organizar Casos de Uso.

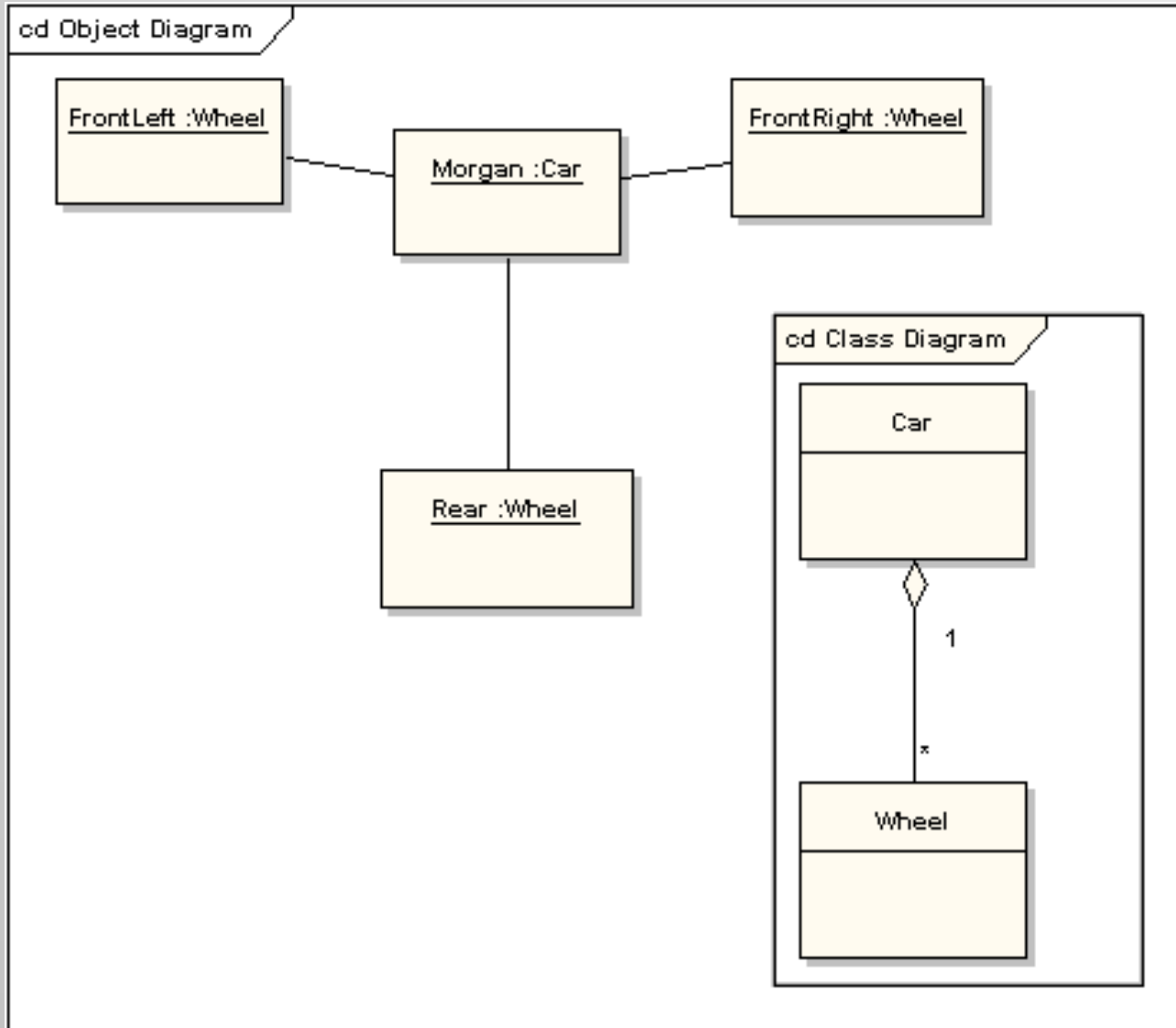


# Diagramas de Clases



Muestran la vista estática del modelo o parte de él, describiendo atributos y comportamiento, y las relaciones entre clases e interfaces.

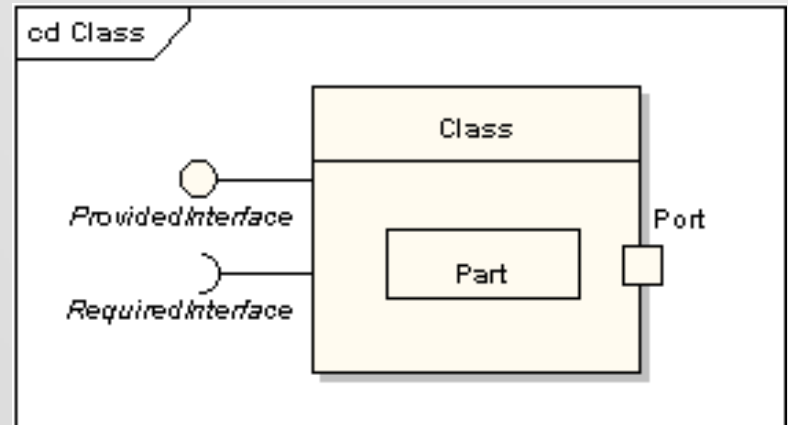
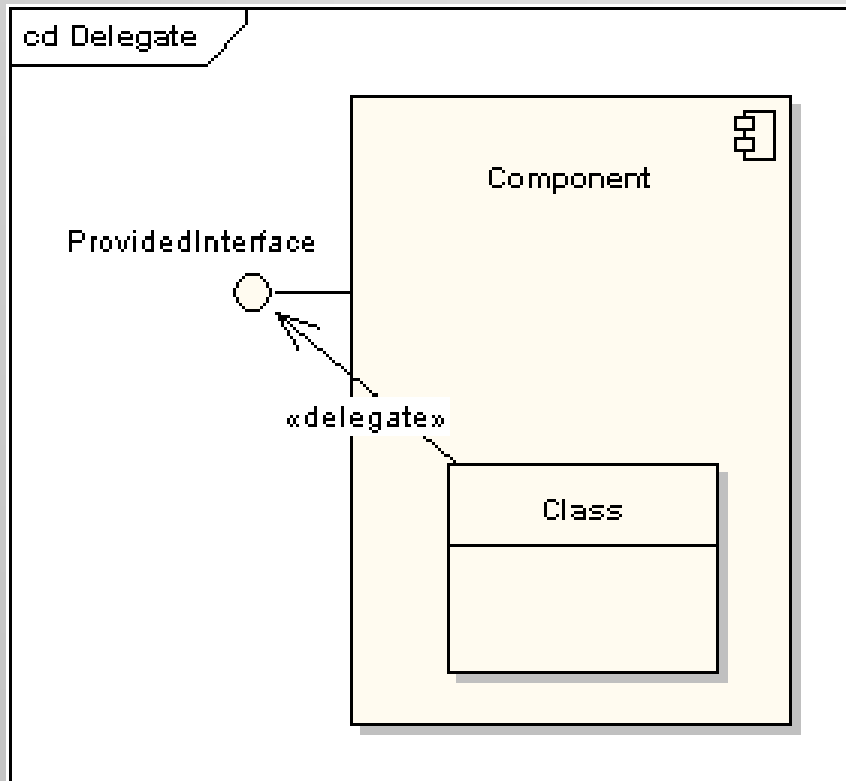
# Diagramas de Objetos



Es un caso especial del diagrama de clases. Se utiliza en un sub-conjunto de elementos del diagrama de clases para enfatizar las relaciones entre instancias de clases en un punto en el tiempo.

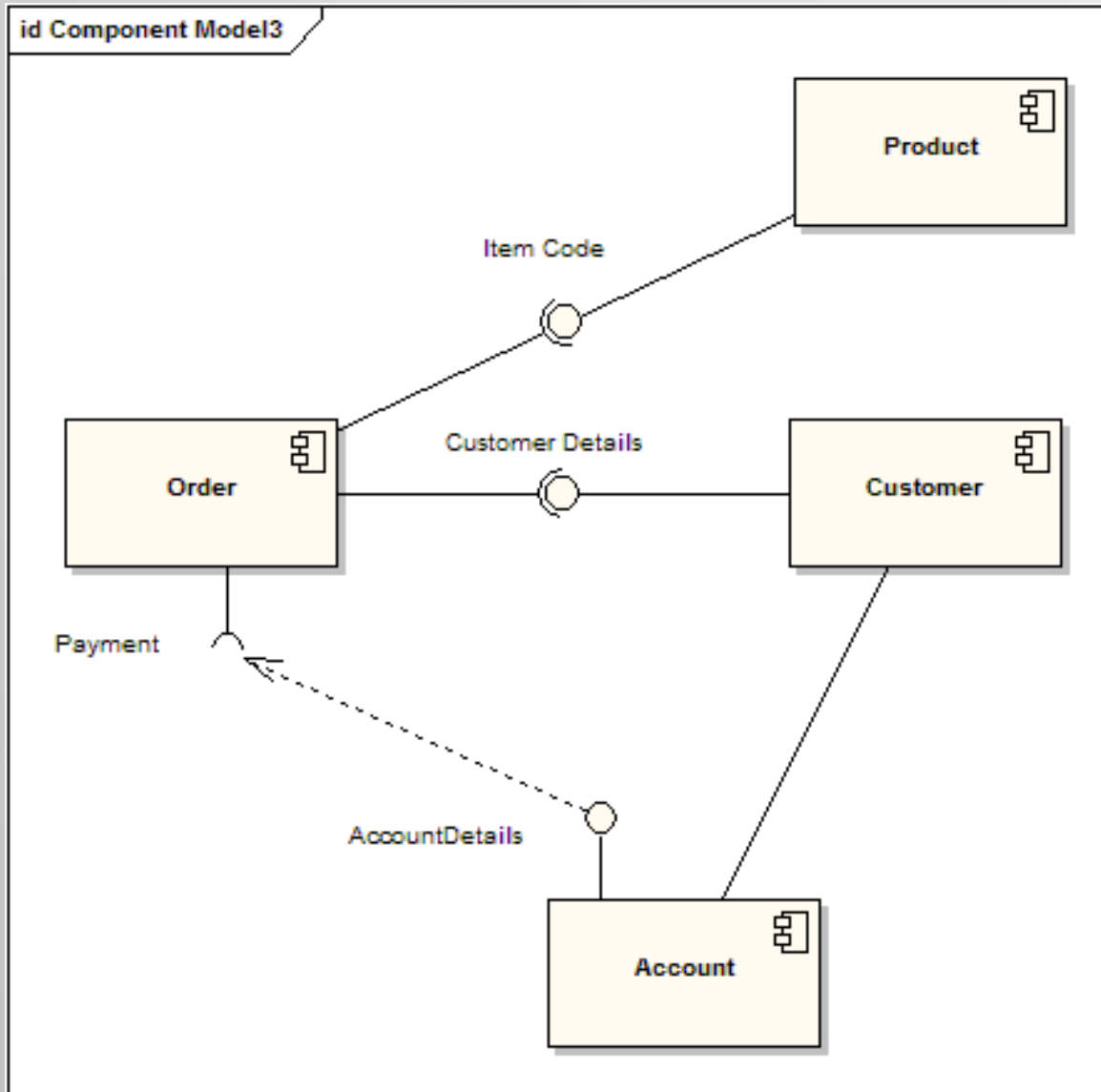
# Diagramas de Estructura Compuesta

Muestran la estructura interna de un clasificador, incluyendo puntos de interacción con otras partes del sistema.



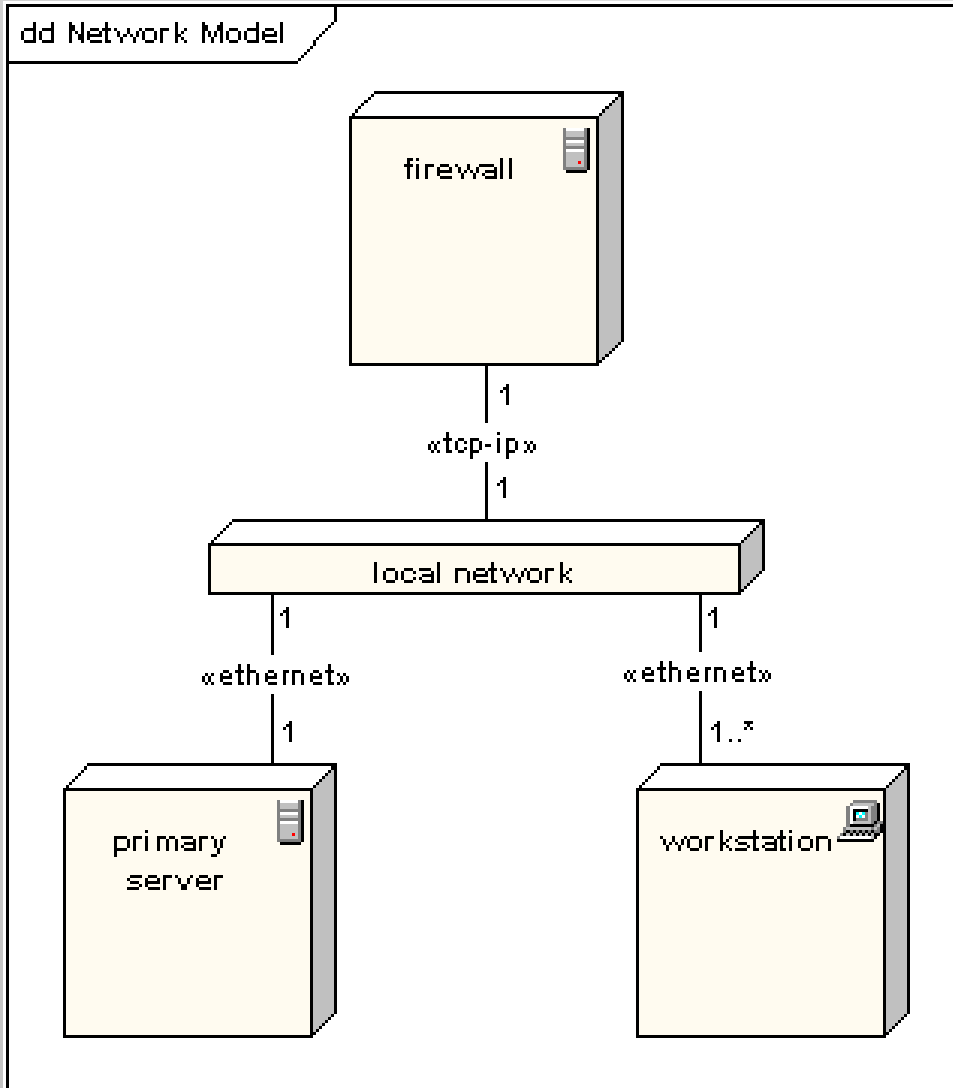


# Diagramas de Componentes



Muestran las piezas de software, controladores embebidos, etc., que construyen un sistema.

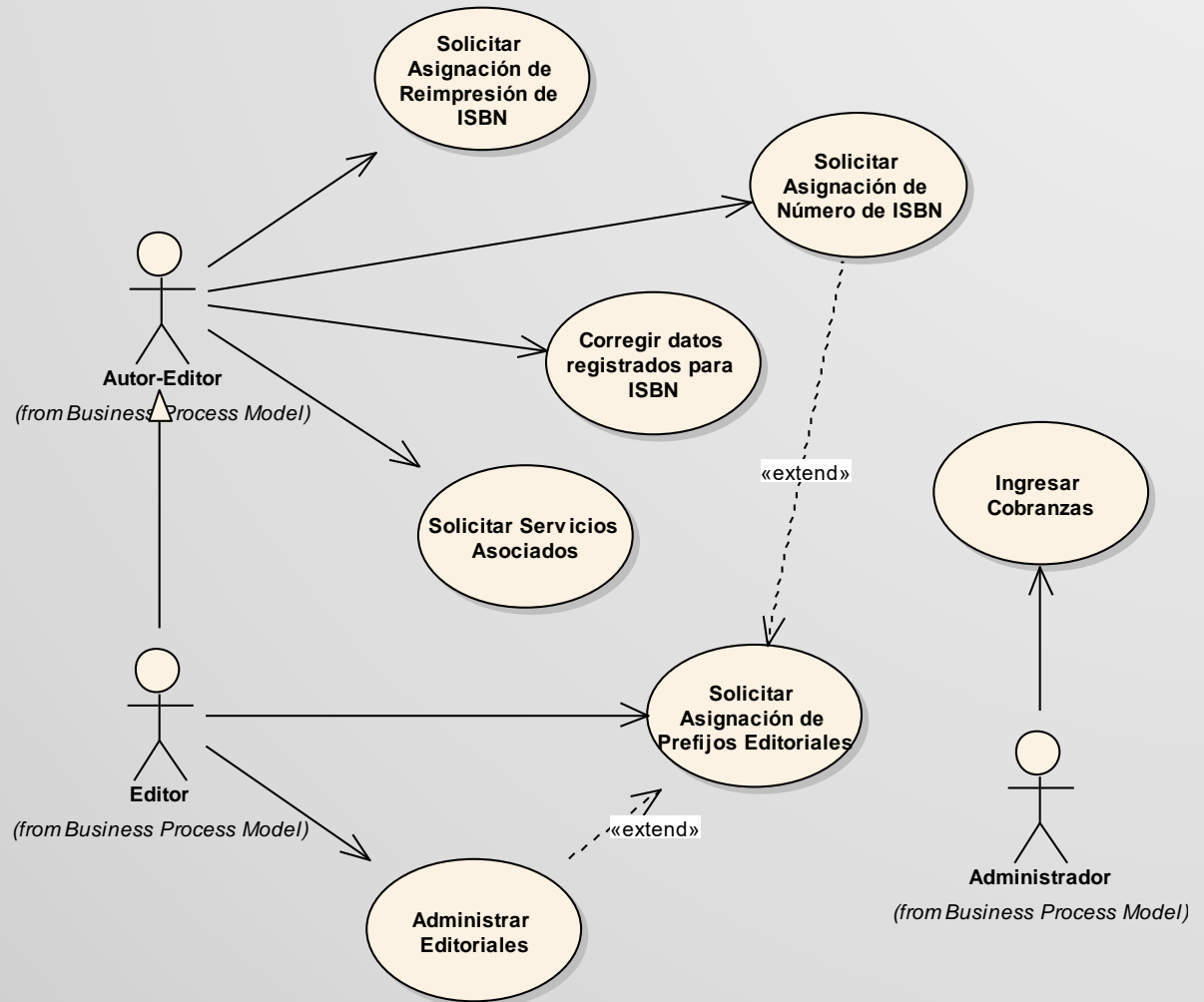
# Diagramas de Despliegue



Modelan la arquitectura en tiempo real del sistema. Muestra la configuración de elementos de hardware (nodos) y como los elementos de software y artefactos se mapean con esos nodos.

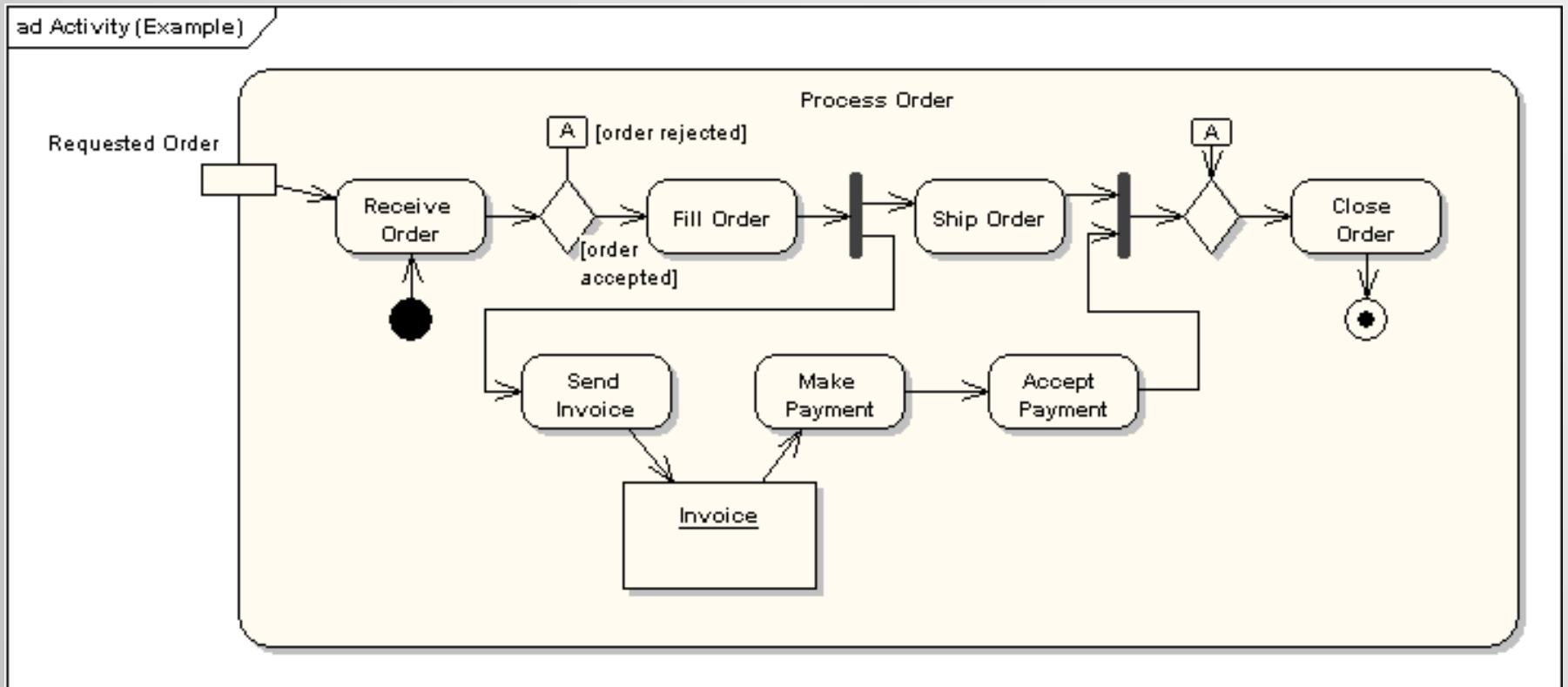
# Diagramas de Casos de Uso

Muestran lo que hace el sistema desde el punto de vista del usuario. Se utilizan para documentar las especificaciones funcionales de un sistema.



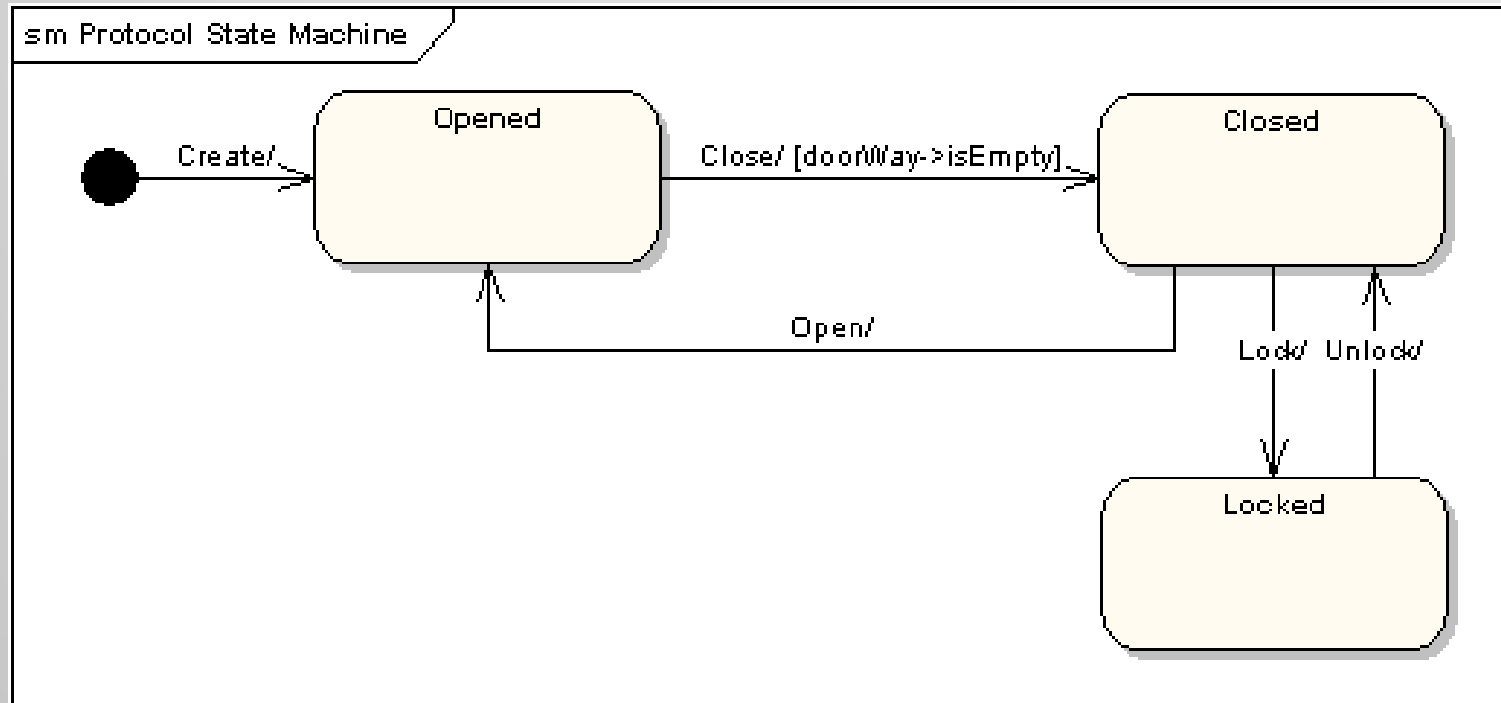
# Diagrama de Actividad

Permiten mostrar el flujo de Actividades para un proceso determinado.



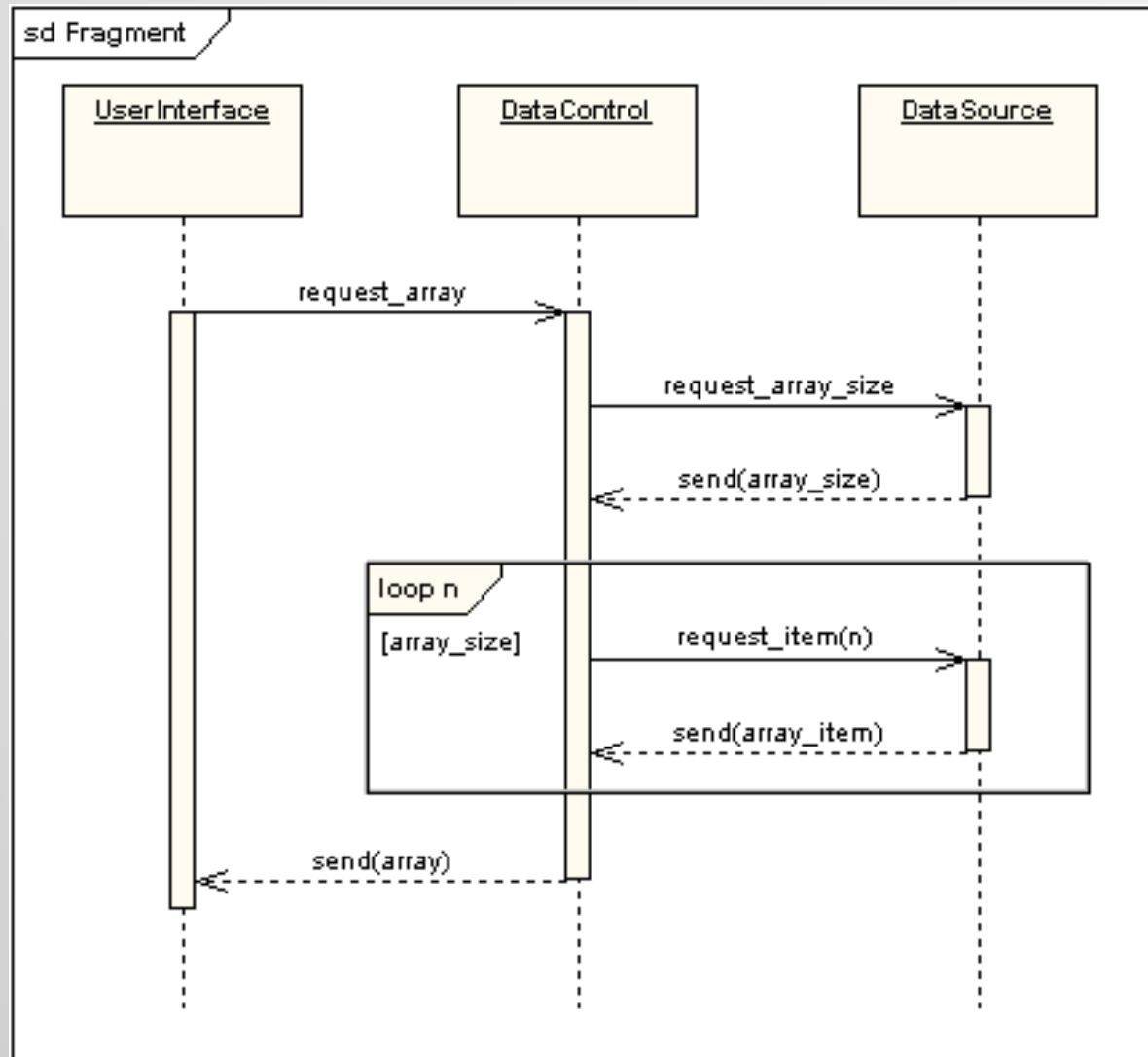
# Diagramas de Transición-Estados

Muestran los estados posibles de un objeto y las transiciones que pueden producirse entre esos estados.



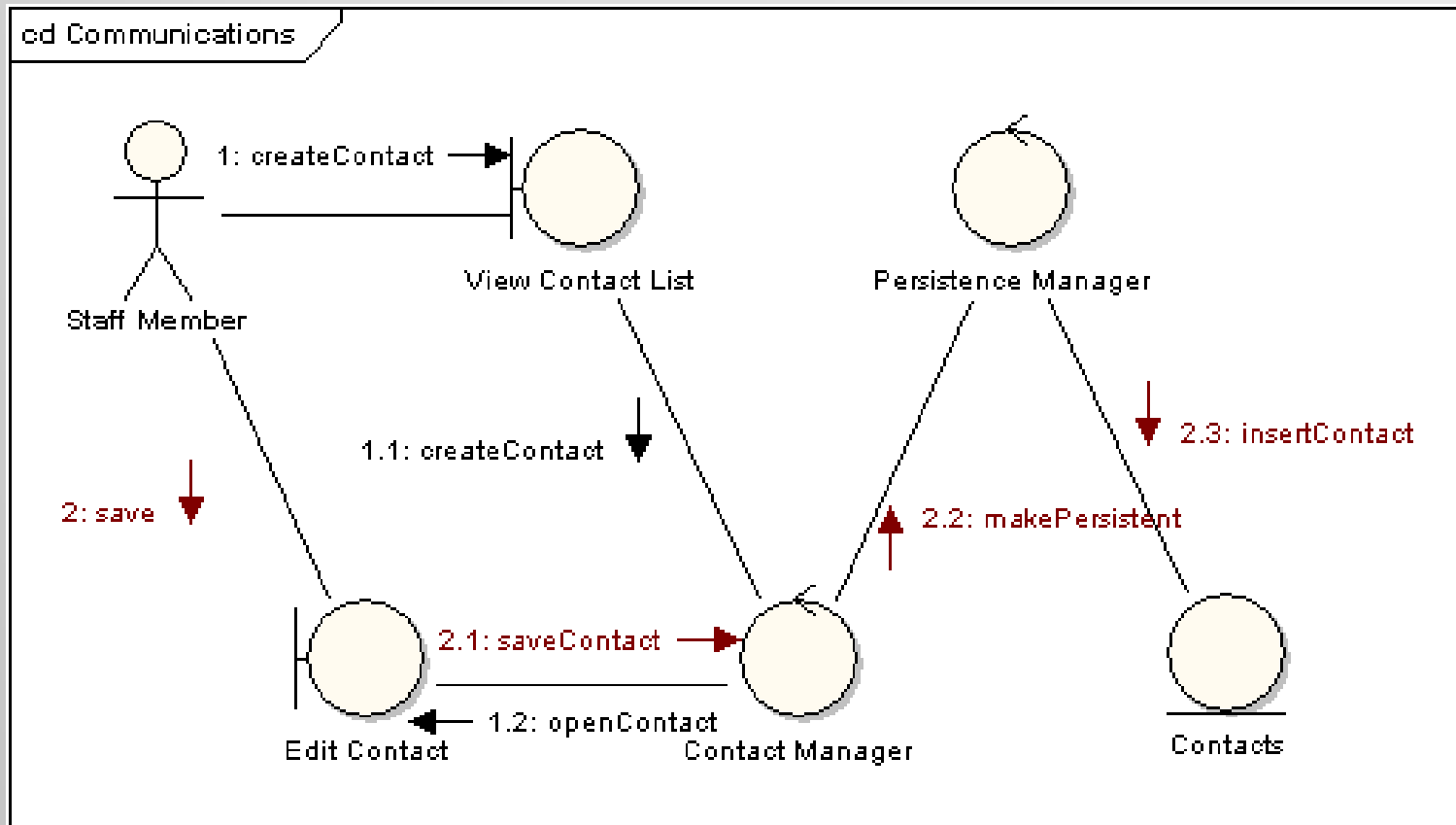
# Diagramas de Secuencia

Muestran la forma en que interactúan los objetos en el sistema. Ponen énfasis en el momento temporal de la interacciones.



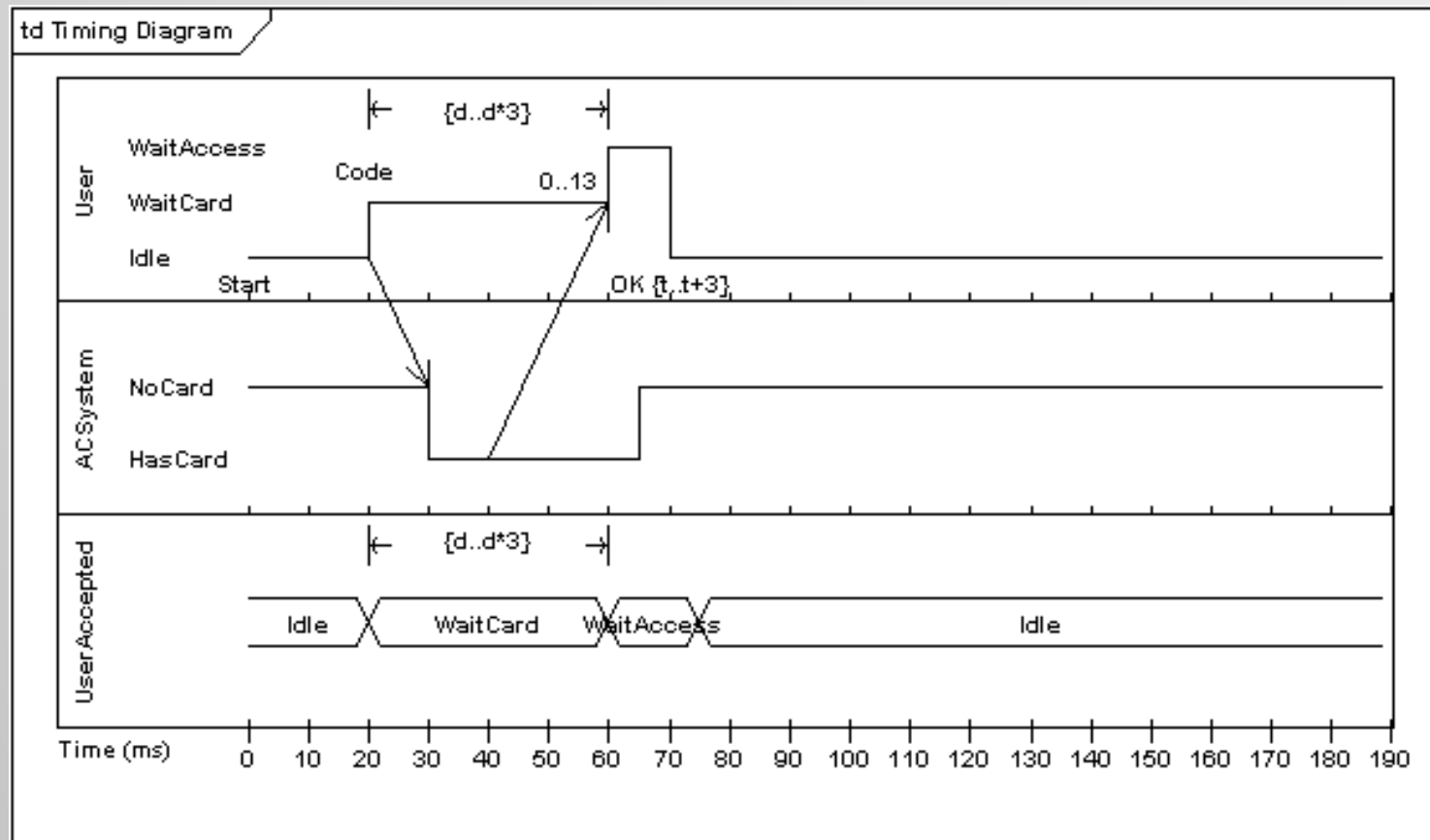
# Diagramas de Comunicación

Muestran básicamente la misma información que los de secuencia, pero hacen énfasis en los objetos que intervienen, más que en el tiempo.



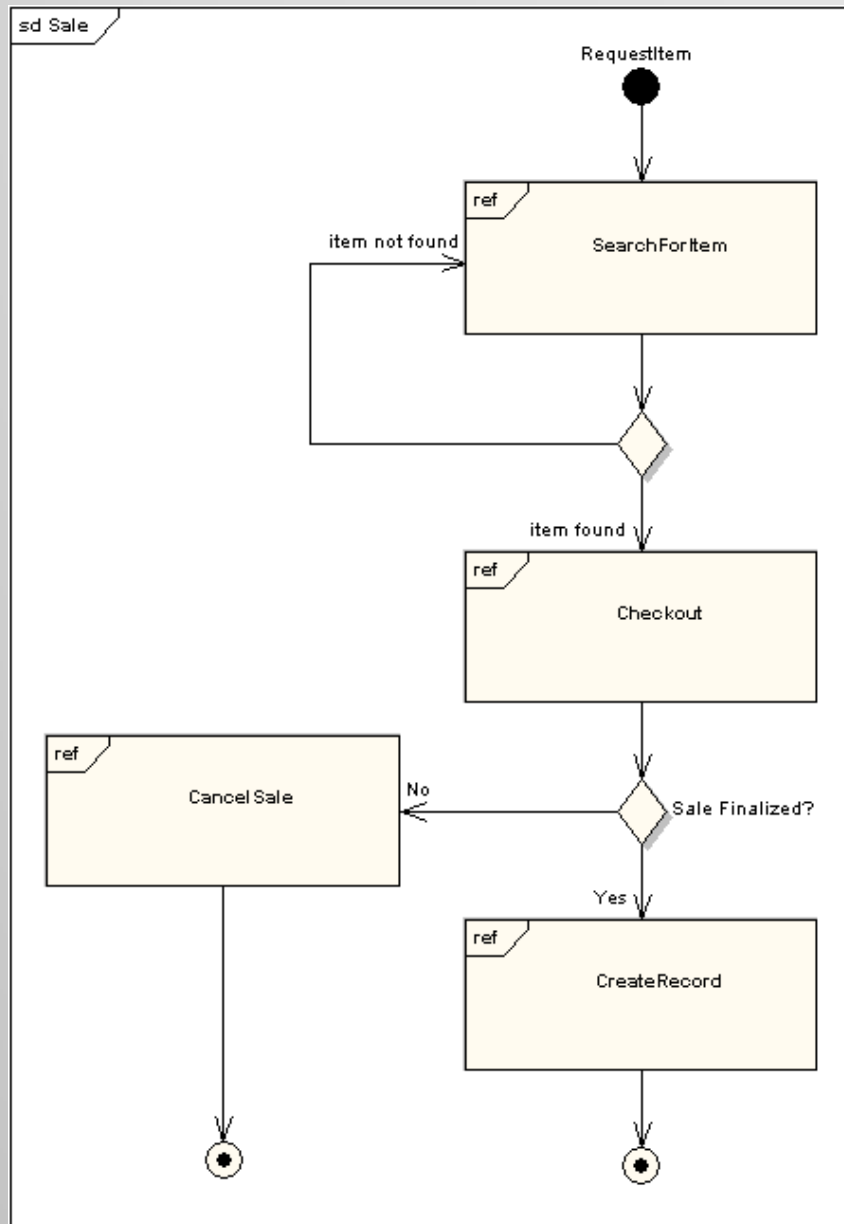
# Diagramas de Tiempo (Timing)

Es una fusión de los diagramas de estados y de secuencia brindando una vista de los estados de un objeto en el tiempo y los mensajes que modifican dicho estado.





# Diagramas de Interacción Global



Es una fusión de los diagramas de actividad y de secuencia donde cada nodo representa un diagrama de interacción, que puede ser cada uno de los ya mencionados.

# Para más Información














- Web sites
  - Enterprise Architect, EA  
<http://www.sparxsystems.com.au>
  - Object Management Group, <http://www.uml.org/>
  - *Rational Software Platform*,  
<http://www-136.ibm.com/developerworks/rational/>
  - Microsoft Solutions Framework site at  
<http://www.microsoft.com/MSF>

# Para más Información

- Libros
  - *UML y Patrones*, Craig Larman, 2da. Edición
  - *El lenguaje Unificado, El Proceso Unificado de Desarrollo de Software*, de Booch, Jacobson y Rumbaugh
  - *Dynamics of Software Development*, Jim McCarthy
  - *Rapid Development*, Steve McConnell
  - *Software Project Survival Guide*, Steve McConnell

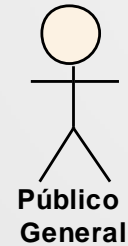
# Composición del Modelo de Casos de Uso

Elementos del Diagrama de Casos de Uso

 Actor	 Use
 Use Case	 Associate
 Collaboration	 Generalize
 Boundary	 Include
 Package	 Extend
	 Realize
	 Dependency
	 Trace

- Otros elementos: Prototipos de interfaz, Especificaciones suplementarias, Glosario, Descripción de la arquitectura.

# Los actores



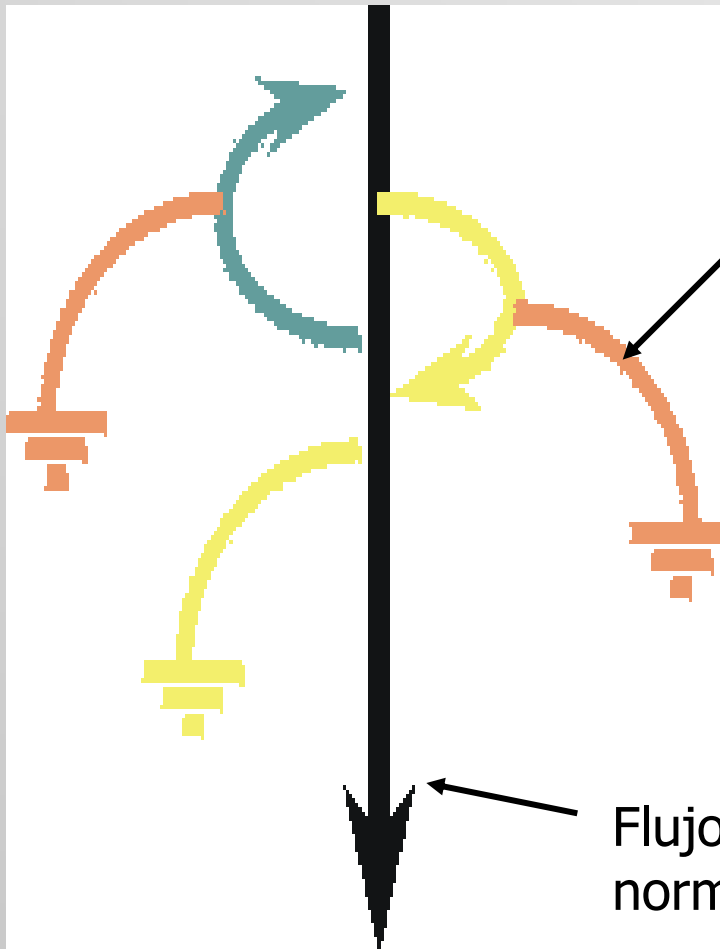
- ▶ *Actores principales*: personas que utilizan las funciones principales del sistema
- ▶ *Actores secundarios*: personas que efectúan tareas administrativas o de mantenimiento
- ▶ *Material externo*: dispositivos materiales imprescindibles que forman parte de la aplicación y que deben ser utilizados
- ▶ *Otros sistemas*: Aquellos con los que el sistema debe interactuar

# Los casos de uso

Solicitar  
Asignación de  
Número de ISBN

- ▶ Son *fragmentos* de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.
- ▶ Especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia.
- ▶ Es una especificación.
- ▶ Puede incluir diagramas de estados, de actividad, colaboraciones y diagramas de secuencia

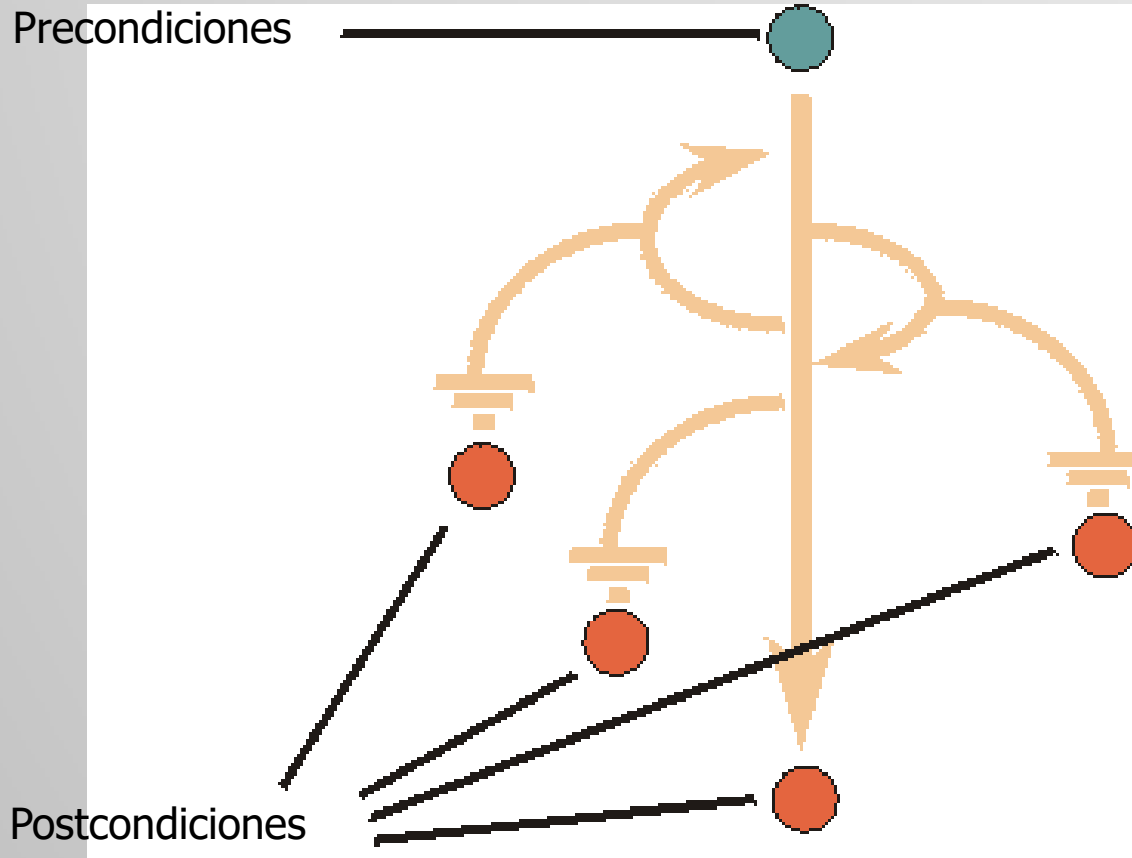
Un caso de uso está compuesto por uno o más  
*flujos de eventos.*



Flujo Alternativo: variantes  
del comportamiento normal

Flujo Principal: Funcionamiento  
normal o de éxito

# Un caso de uso tiene asociadas *precondiciones* y *postcondiciones*



*Precondiciones:* reflejan el estado en el que debe estar el sistema y su entorno para que pueda comenzar la ejecución del caso de uso.

*Postcondiciones:* reflejan el estado en el que queda el sistema y su entorno luego de la ejecución del caso de uso.

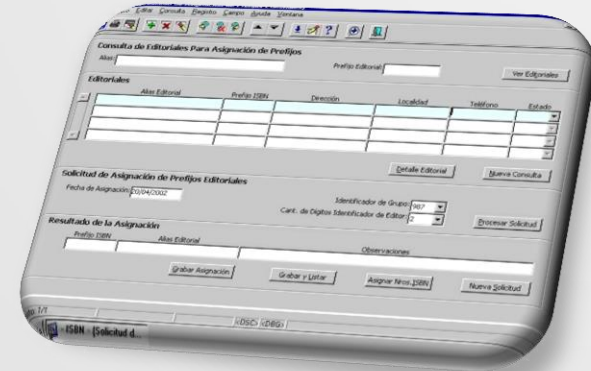


# Alternativas de documentación

- Texto o especificación (medio de preferencia, fácil de mantener y entender):
  - Nombre
  - Descripción
  - Flujos de Eventos (principal y alternativos)
    - Activación
    - Camino Básico
    - Caminos alternativos
  - Requerimientos especiales (no funcionales)
  - Precondiciones
  - Puntos de extensión
- Diagramas de Actividad (difícil de mantener y entender)
- Es importante que a medida que se van encontrando los casos de uso, se identifiquen y documenten los términos que forman parte del dominio del problema en un Glosario.

# Prototipos de Interfaz

- Se utilizan como ayuda para comprender y especificar las interacciones entre los actores humanos y el sistema.
- Sumamente útiles para estabilizar los requerimientos. A los usuarios les resulta mucho más sencillo tomar decisiones o hacer especificaciones sobre algo visible que explicarlas en palabras
- Documentación (no UML): dibujos en papel o en herramienta de dibujo, herramientas de interfaces gráficas, Excel, etc.



# Relaciones entre Actores y Casos de Uso

- ▶ La única relación posible entre los actores y los casos de uso es la *asociación de comunicación*.
- ▶ Las asociaciones de comunicación tienen un *sentido o navegabilidad*, que indica cuál de los extremos es el que inicia y comanda la interacción.

Ejemplo:



# Las relaciones entre actores

- *Relación de generalización*: Se utiliza cuando se tienen actores que representan un rol muy parecido pero con ligeras diferencias entre sí.

Se utiliza para indicar que un actor específico puede hacer lo mismo que otro más general, más otras cosas particulares de él.

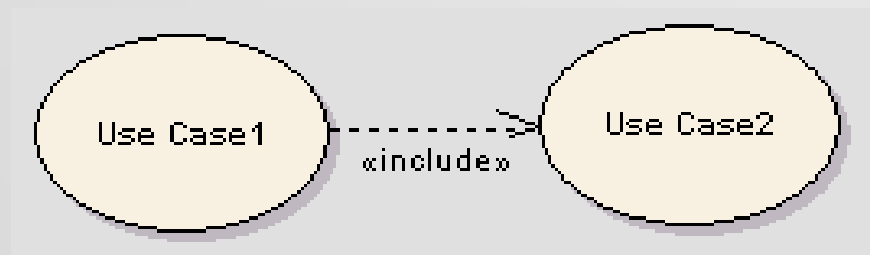
Ejemplo:



# Ejemplo de Inclusión

Se utiliza cuando se detecta que en al menos dos casos de uso diferentes conceptualmente existe alguna secuencia de pasos o flujo de eventos repetida.

Esta secuencia puede extraerse dando lugar a un nuevo caso de uso relacionado con los otros dos mediante una relación de inclusión.

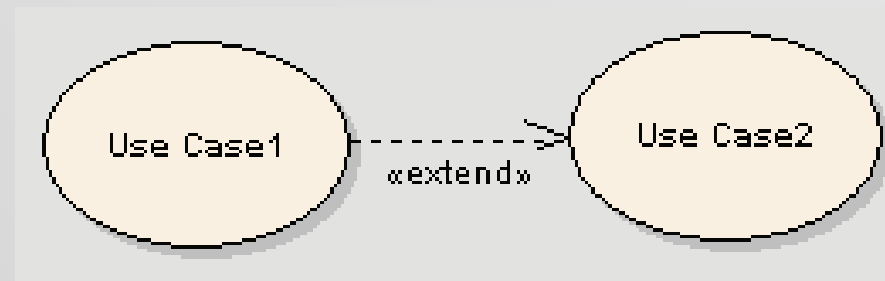


En el ejemplo se puede ver que el caso de uso 1 incluye la funcionalidad del caso de uso 2.

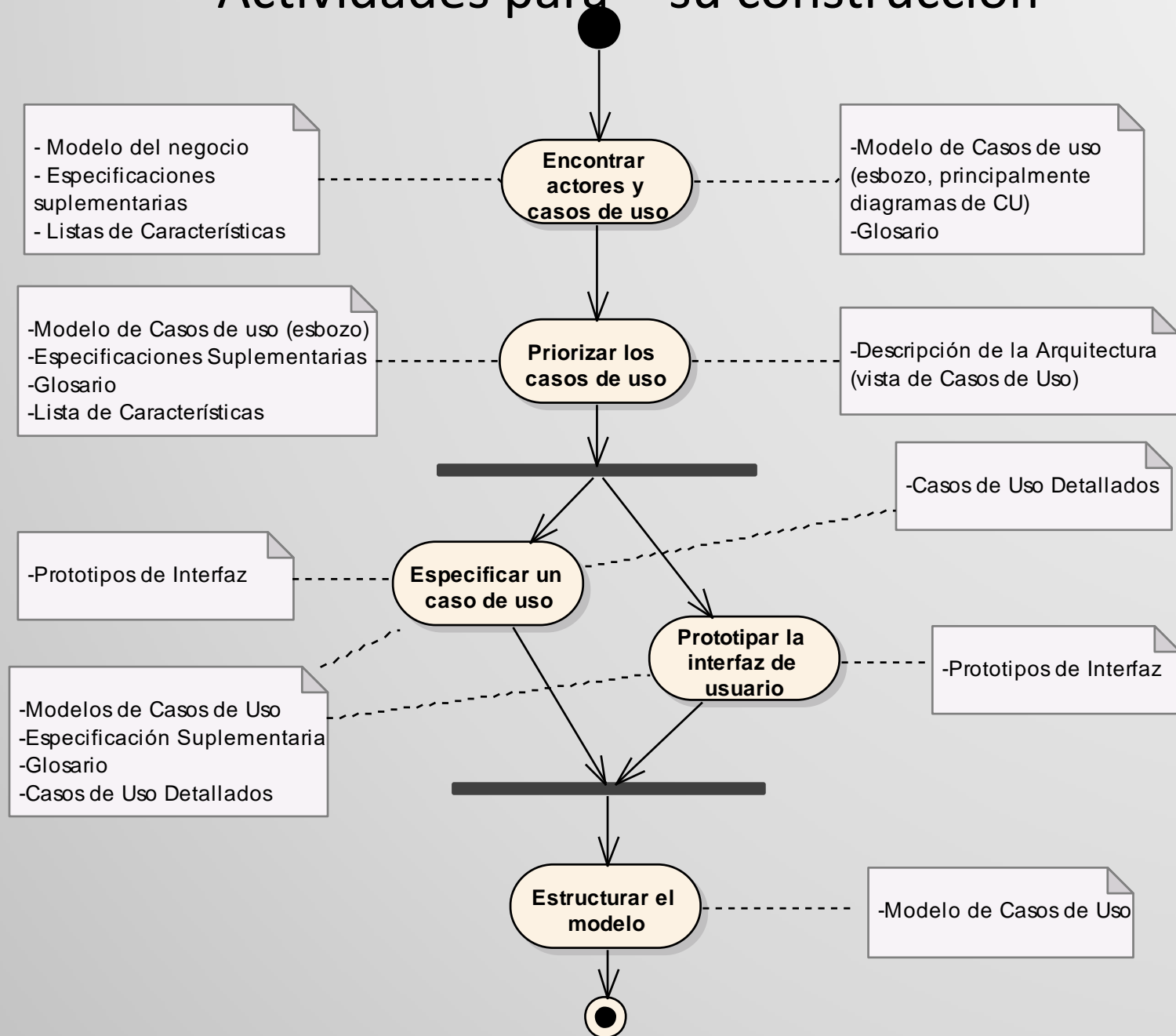
# Ejemplo de Extensión

Se utiliza cuando se desea indicar que mientras se está utilizando un caso de uso, se puede utilizar OPCIONALMENTE otro caso de uso, en algún punto de la ejecución del primero. Dentro de un caso de uso, algunos flujos alternativos pueden representar un comportamiento opcional seleccionado por el usuario. Cuando ese comportamiento puede ser a su vez utilizado por separado, resulta conveniente extraerlo a un nuevo caso de uso relacionado como extensión del original.

En el ejemplo se puede ver que el caso de uso 1 extiende al caso de uso 2.

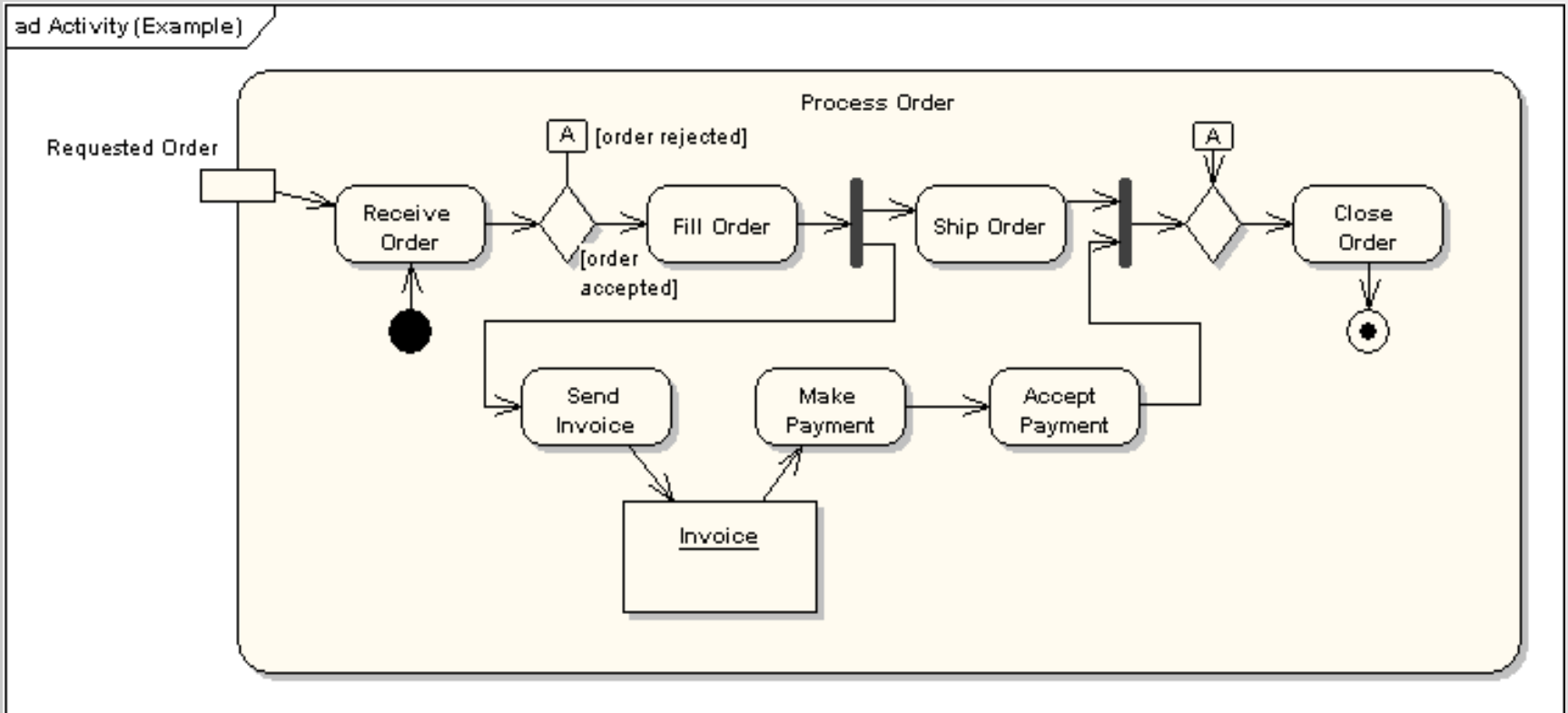


# Actividades para su construcción



# Diagramas de Actividad

Permiten mostrar el flujo de Actividades para un proceso determinado.





# Diagramas de Actividad

- ▶ Un diagrama de actividades es un caso especial de un diagrama de estados, en el cual casi todos los estados son estados de acción (identifican que acción se ejecuta al estar en él) y donde casi todas las transiciones son enviadas al terminar la acción ejecutada en el estado anterior.
- ▶ Puede dar detalle a un caso de uso, un objeto o un mensaje en un objeto. Sirven para representar transiciones internas, sin hacer mucho énfasis en transiciones o eventos externos.
- ▶ Generalmente modelan los pasos de un algoritmo.
- ▶ Son particularmente útiles para describir workflow y para descripción de comportamiento que tiene paralelismo
- ▶ Tiene un **inicio** y un **fin** como los diagramas de estado

# Diagramas de Actividad

## ► Estado de acción

- Representa un estado con acción interna, con por lo menos una transición que identifica la culminación de la acción (por medio de un evento implícito).
- No deben tener transiciones internas ni transiciones basadas en eventos (Si este es el caso, represéntelo en un diagrama de estados).
- Permite modelar un paso dentro del algoritmo. Se representan por un rectángulo con bordes redondeados.

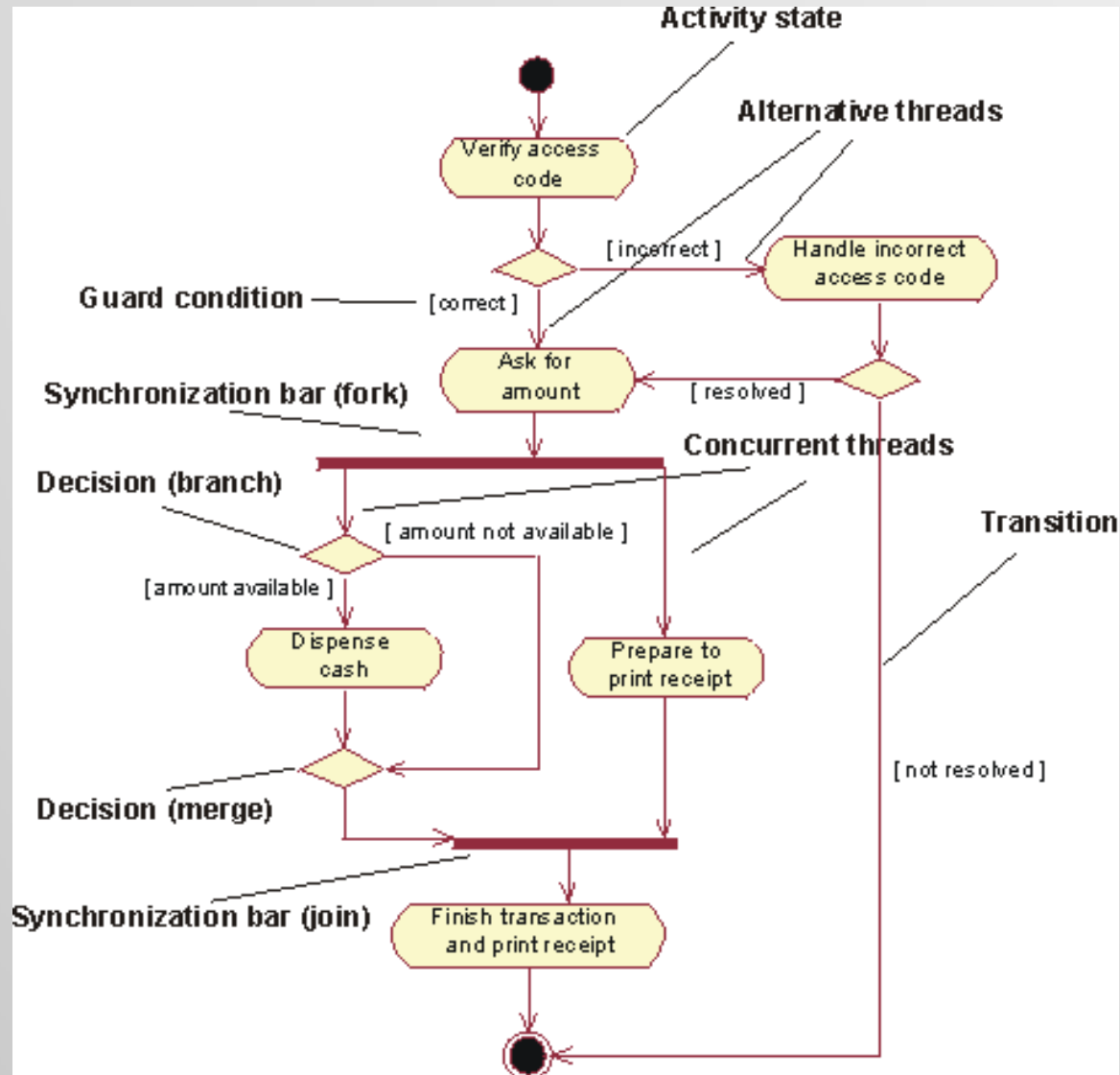
## ► Transiciones

- Las flechas entre estados representan transiciones con evento implícito. Pueden tener una condición en el caso de decisiones.

## ► Decisiones

- Se representa mediante una transición múltiple que sale de un estado, donde cada camino tiene una etiqueta distinta.
- Se representa mediante un diamante al cual llega la transición del estado inicial y del cual salen las múltiples transiciones de los estados finales.

# Diagrama de Actividad

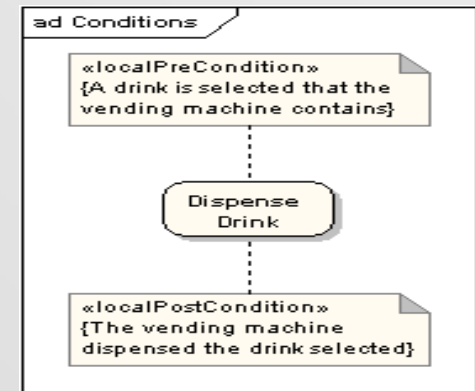
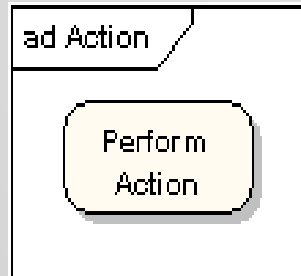


# Diagramas de Actividad

## ■ Acción (Action )

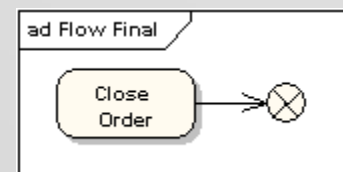
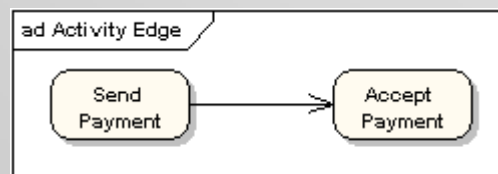
- Acción : Paso simple dentro de una actividad (rectángulo de bordes redondeados).
- Restricciones a una Acción (Action Constraints).

Ejemplo 'pre' y 'post' condiciones.



## ■ Flujo (Flow )

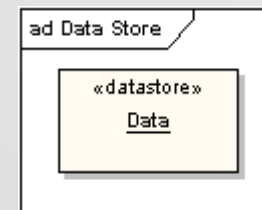
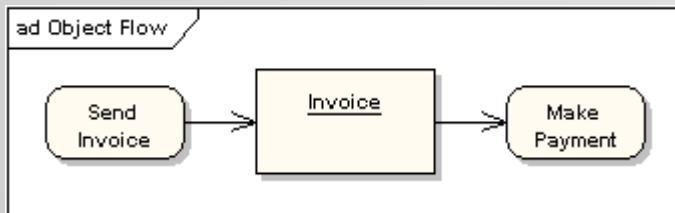
- Flujo de Control (Control Flow) : Muestra el flujo entre una actividad y otra (línea con punta de flecha).
- Flujo de Nodo Final (Flow Final Node) : Indica la finalización de un flujo de control.



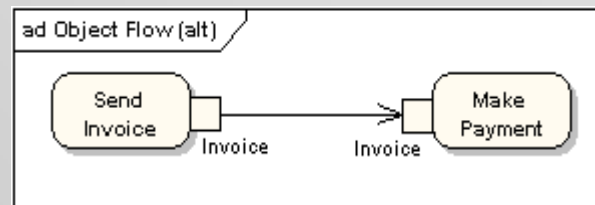
# Diagramas de Actividad

## ■ Objetos – Flujos de Objetos (Object and Object Flows)

- Flujo de Objetos : Es un camino a lo largo del cual pueden pasar objetos o datos. El objeto se representa con un rectángulo. El flujo se muestra como un conector con punta de flecha cuya dirección indica hacia dónde el objeto está pasando.
- Almacenamiento de datos (Data Store) : Objeto con la palabra clave 'datastore'.



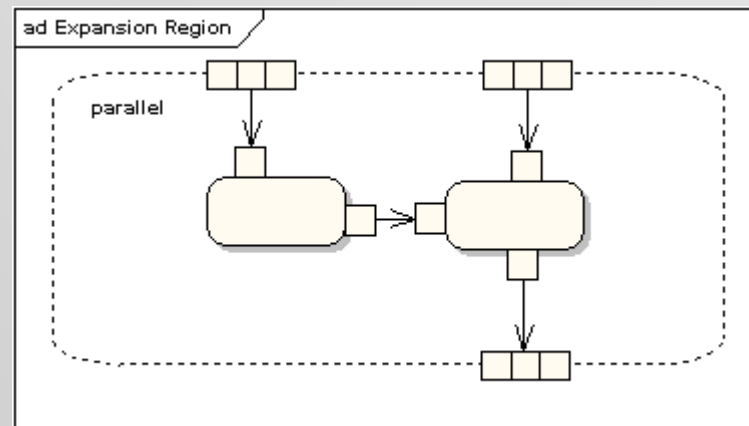
- Un flujo de objetos debe tener al menos un objeto en alguno de sus extremos, (p.e) notación que pretende ser usada para pins de 'input' y 'output' :



# Diagramas de Actividad

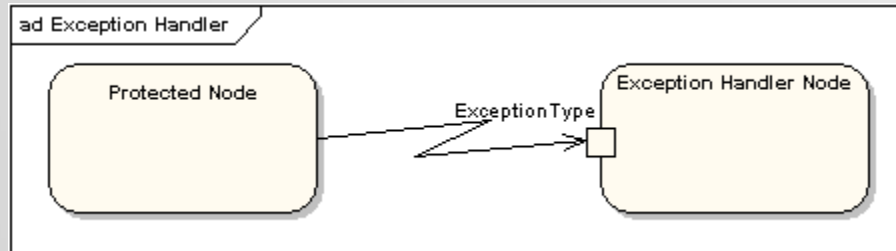
## ■ Región de Expansión (Expansion Region)

- Región de actividad estructurada que se ejecuta muchas veces. Los nodos de 'input' y 'output' se dibujan como un grupo de tres 'boxes', que representan una selección múltiple de ítems. La palabra clave 'parallel' o 'stream' se muestra en la esquina superior izquierda.



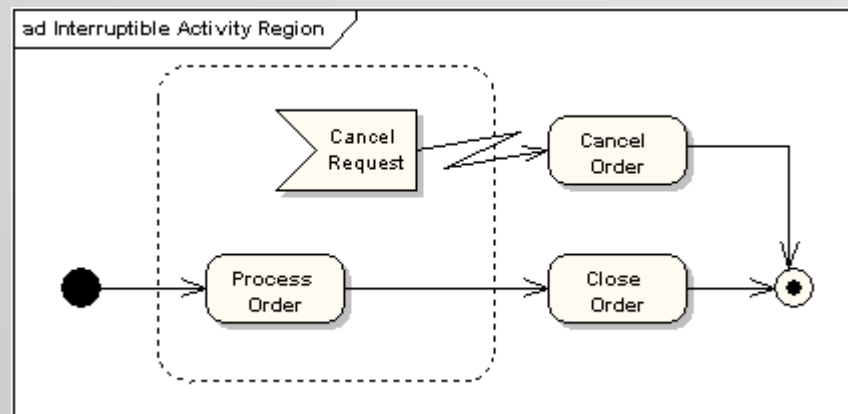
# Diagramas de Actividad

## ■ Manejadores de Excepciones (Exception Handlers)



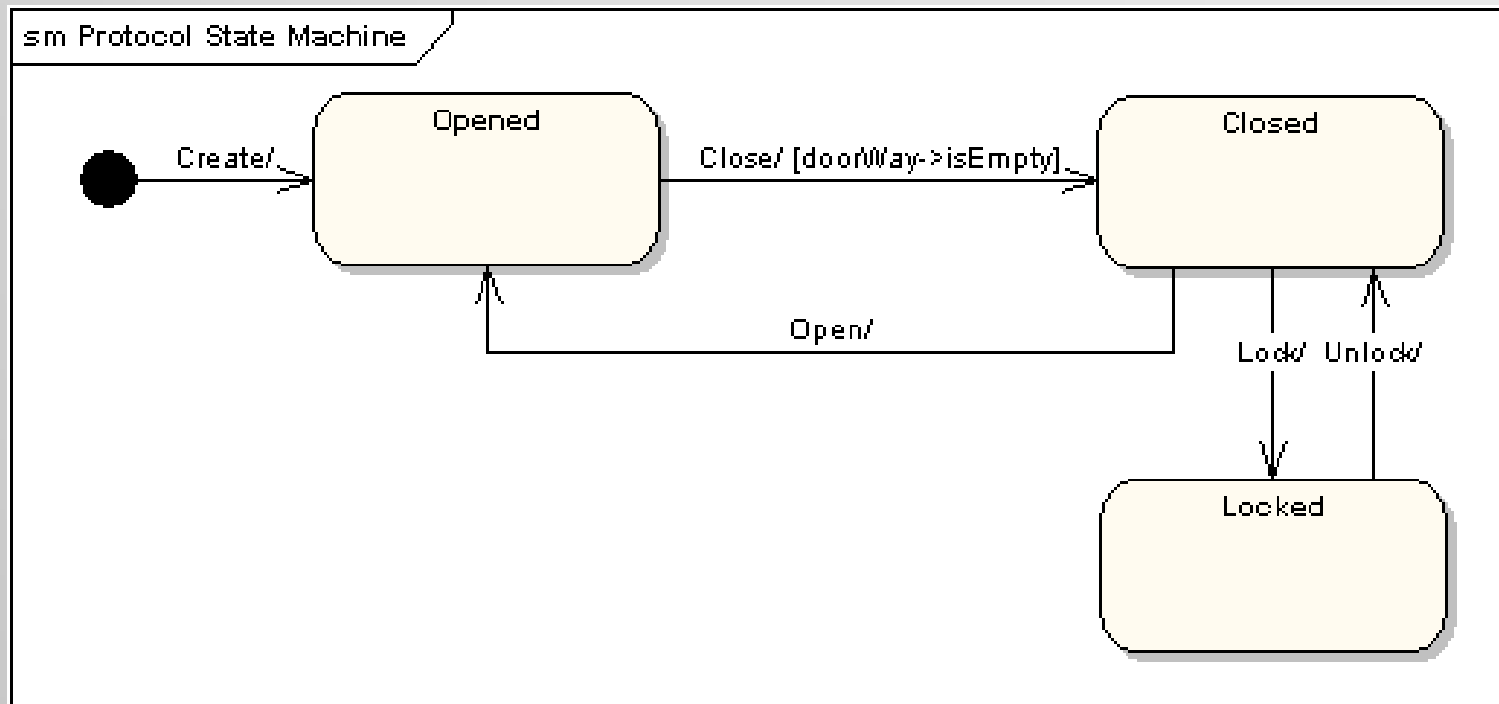
## ■ Región de Actividad interrumpible (Interruptible Activity Region)

- Esta región rodea un grupo de actividades que pueden ser interrumpidas. En el ejemplo, la acción 'Process Order', se ejecutará hasta completarse, a menos que una interrupción de 'request cancel' sea recibida, en cuyo caso el control pasará a la actividad 'Cancel Order'.



# Diagramas de Transición-Estados

Muestran los estados posibles de un objeto y las transiciones que pueden producirse entre esos estados.

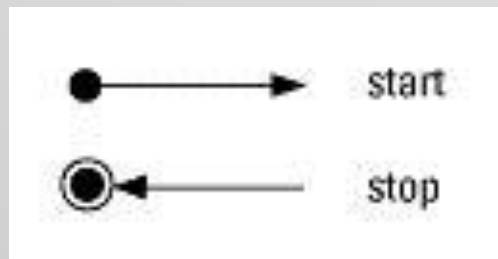




# Transición-Estados

## ► Estado

- Identifica un período de tiempo del objeto (no instantáneo) en el cual el objeto esta esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos.
- Se representa mediante un rectángulo con los bordes redondeados, que puede tener tres compartimientos: uno para el nombre, otro para el valor característico de los atributos del objeto en ese estado y otro para las acciones que se realizan al entrar, salir o estar en un estado (entry, exit o do, respectivamente).
- Se marcan también los estados iniciales y finales mediante los símbolos



# Transición-Estados

## ► Eventos

- Es una ocurrencia que puede causar la transición de un estado a otro de un objeto. Esta ocurrencia puede ser una de varias cosas:
  - Condición que toma el valor de verdadero o falso
  - Recepción de una señal de otro objeto en el modelo
  - Recepción de un mensaje
  - Paso de cierto período de tiempo, después de entrar al estado o de cierta hora y fecha particular
- El nombre de un evento tiene alcance dentro del paquete en el cual está definido, no es local a la clase que lo nombre.

# Transición-Estados

## ► Transición simple

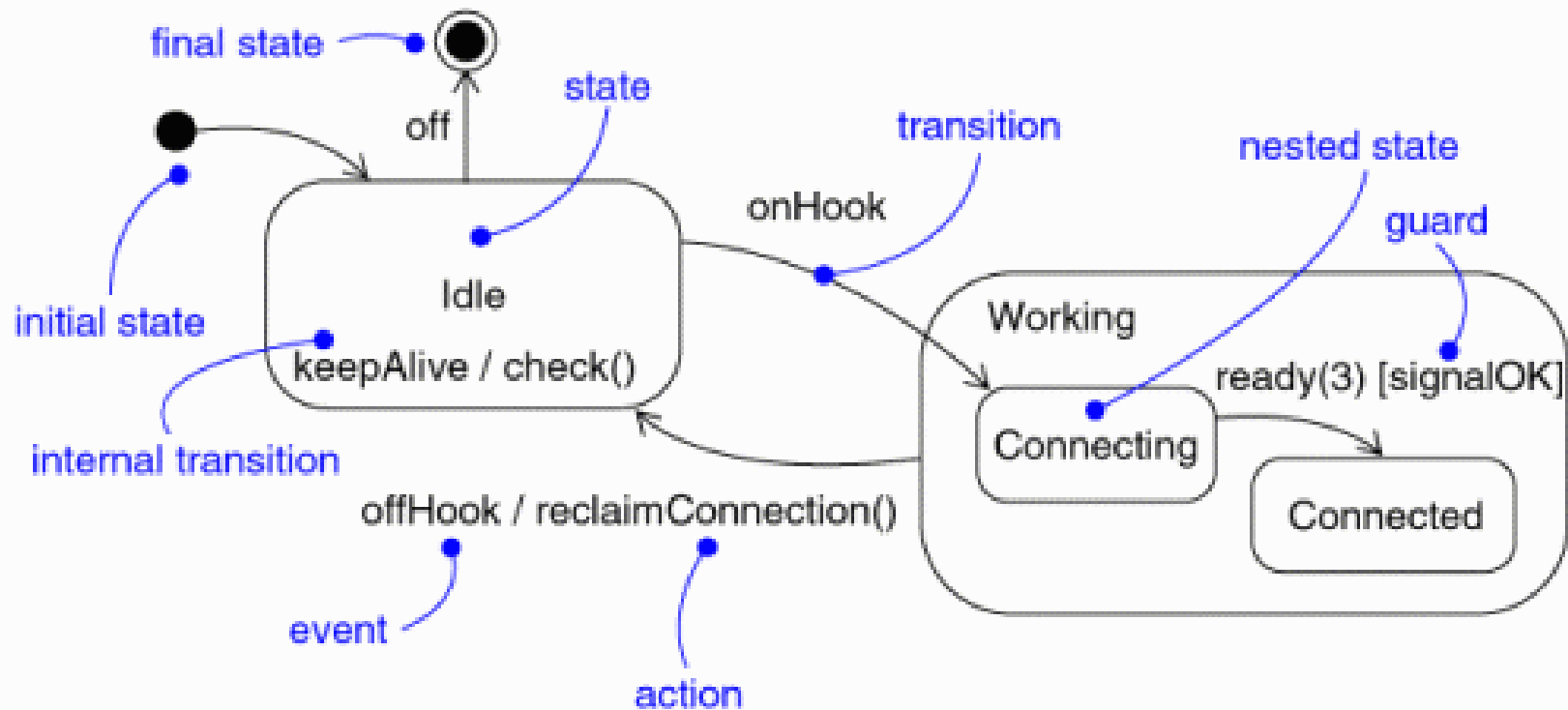
- Es una relación entre dos estados que indica que un objeto en el primer estado puede entrar al segundo estado y ejecutar ciertas operaciones, cuando un evento ocurre y si ciertas condiciones son satisfechas.
- Se representa como una línea sólida entre dos estados.

## ► Transición interna

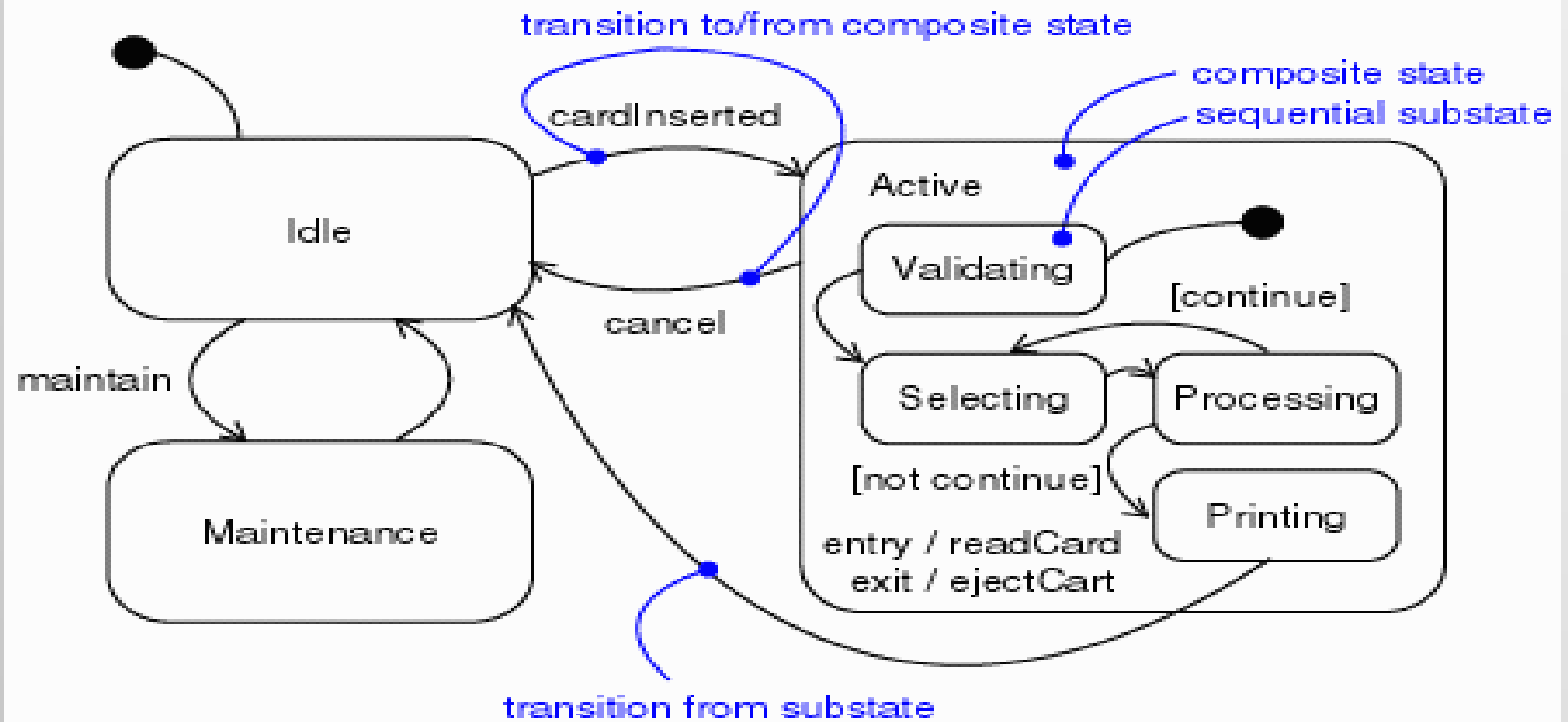
- Es una transición que permanece en el mismo estado, en vez de involucrar dos estados distintos. Representa un evento que no causa cambio de estado. Se denota como una cadena adicional en el compartimiento de acciones del estado.

# Transición-Estados

State Machine



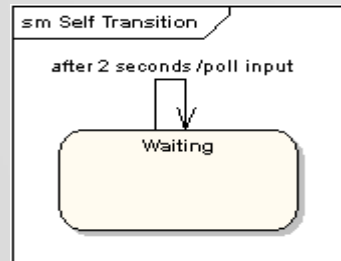
# Transición-Estados



# Transición-Estados

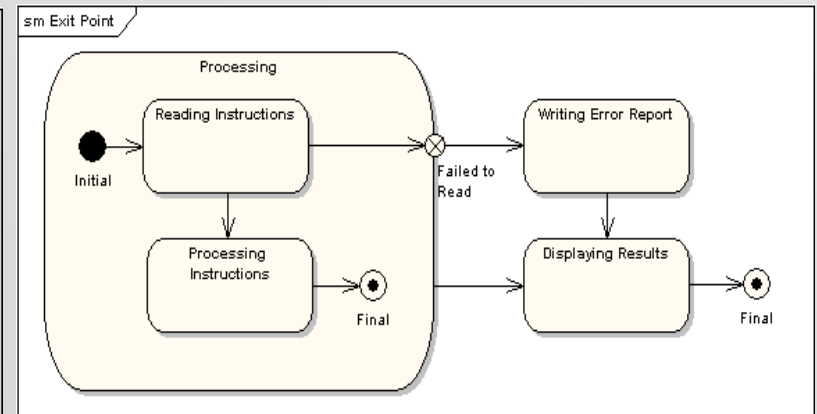
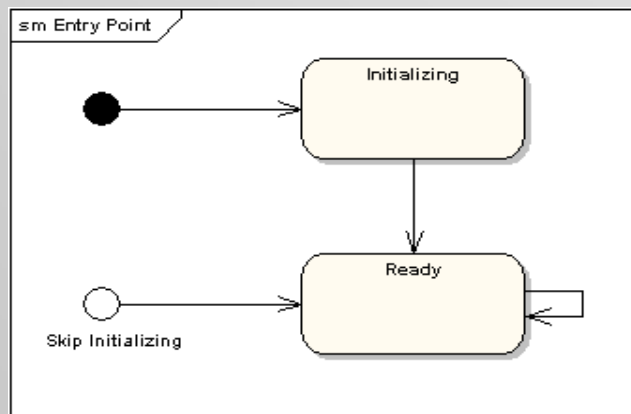
## ■ Transición sobre sí misma (Self-Transition)

- Un estado puede tener una transición que retorne a sí mismo. Es el más usado cuando se asocia un efecto con la transición.



## ■ Puntos de Entrada y Salida (Entry Point / Exit Point)

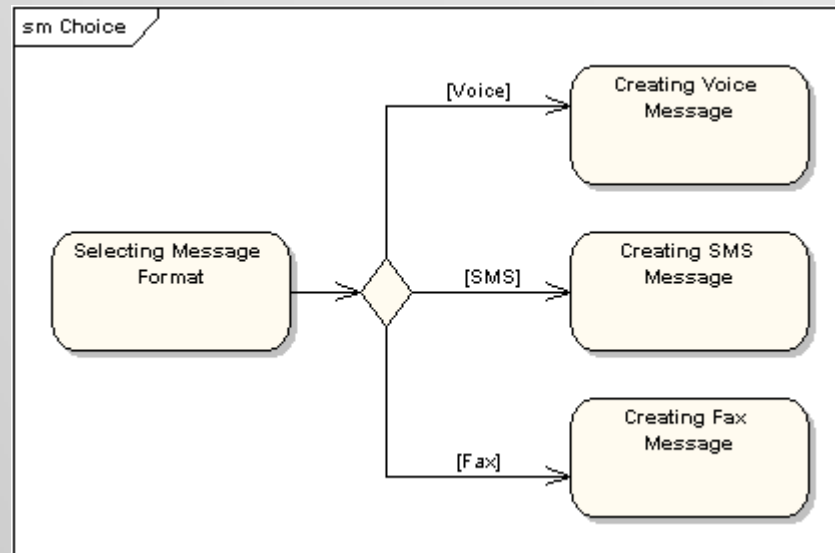
- Posibilidad de comenzar/finalizar en estados alternativos.



# Transición-Estados

## ■ Selección (Choice Pseudo-State)

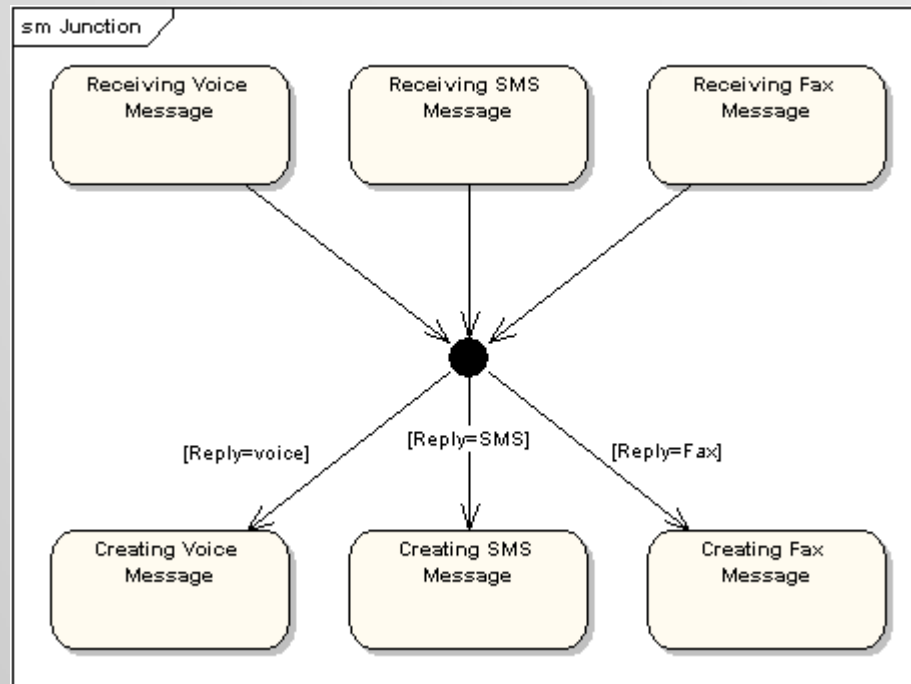
- Se representa como un diamante con una transición que llega y dos o más transiciones que salen. Cualquiera sea el estado al que se arrive después del mismo, es dependiente del formato de mensaje seleccionado durante la ejecución del estado previo.



# Transición-Estados

## ■ Intersección (Junction Pseudo-State)

- Para concatenar múltiples transiciones. Una intersección a donde llegan y de donde parten varias transiciones realiza un branch condicional estático, a diferencia de la selección, el cual es dinámico.

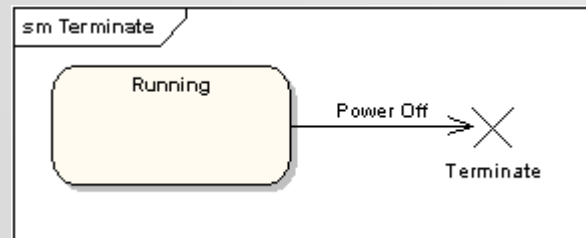




# Transición-Estados

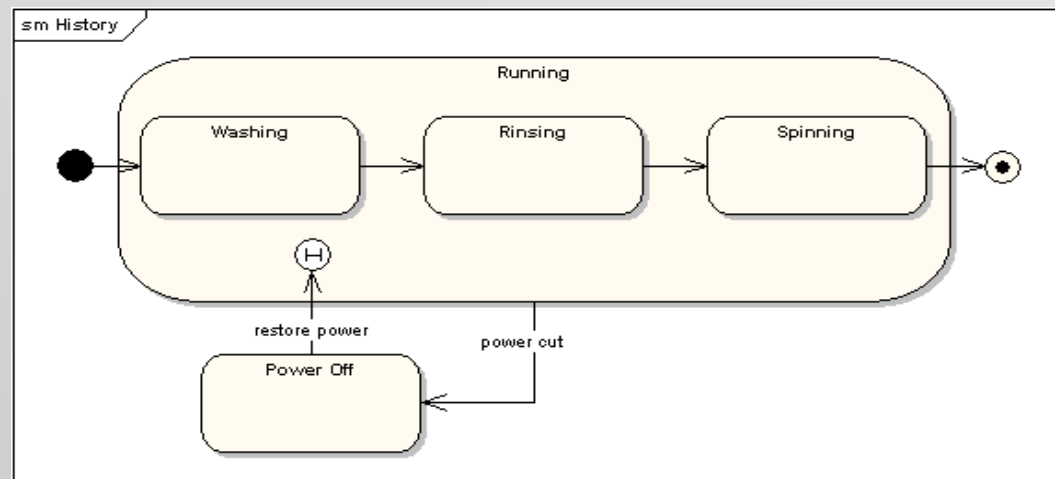
## ■ Finalizado (Terminate Pseudo-State)

- Indica que la línea de vida del estado ha finalizado.



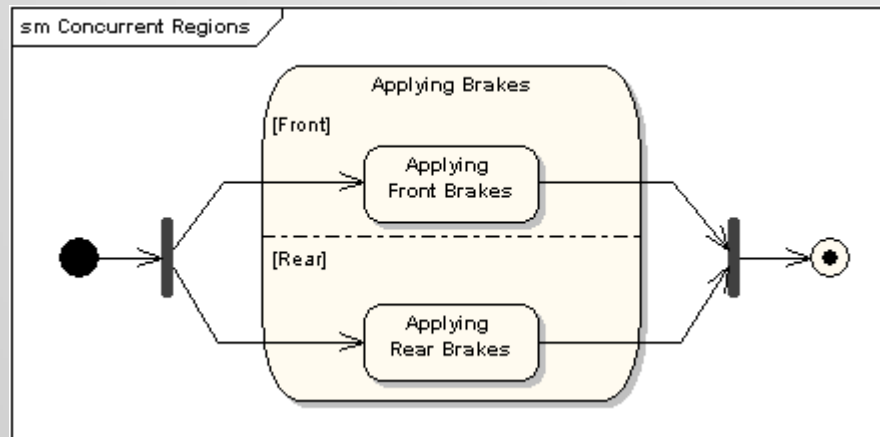
## ■ Estados Históricos (History States)

- Usado para indicar que un estado previo puede ser restaurado luego de una interrupción.



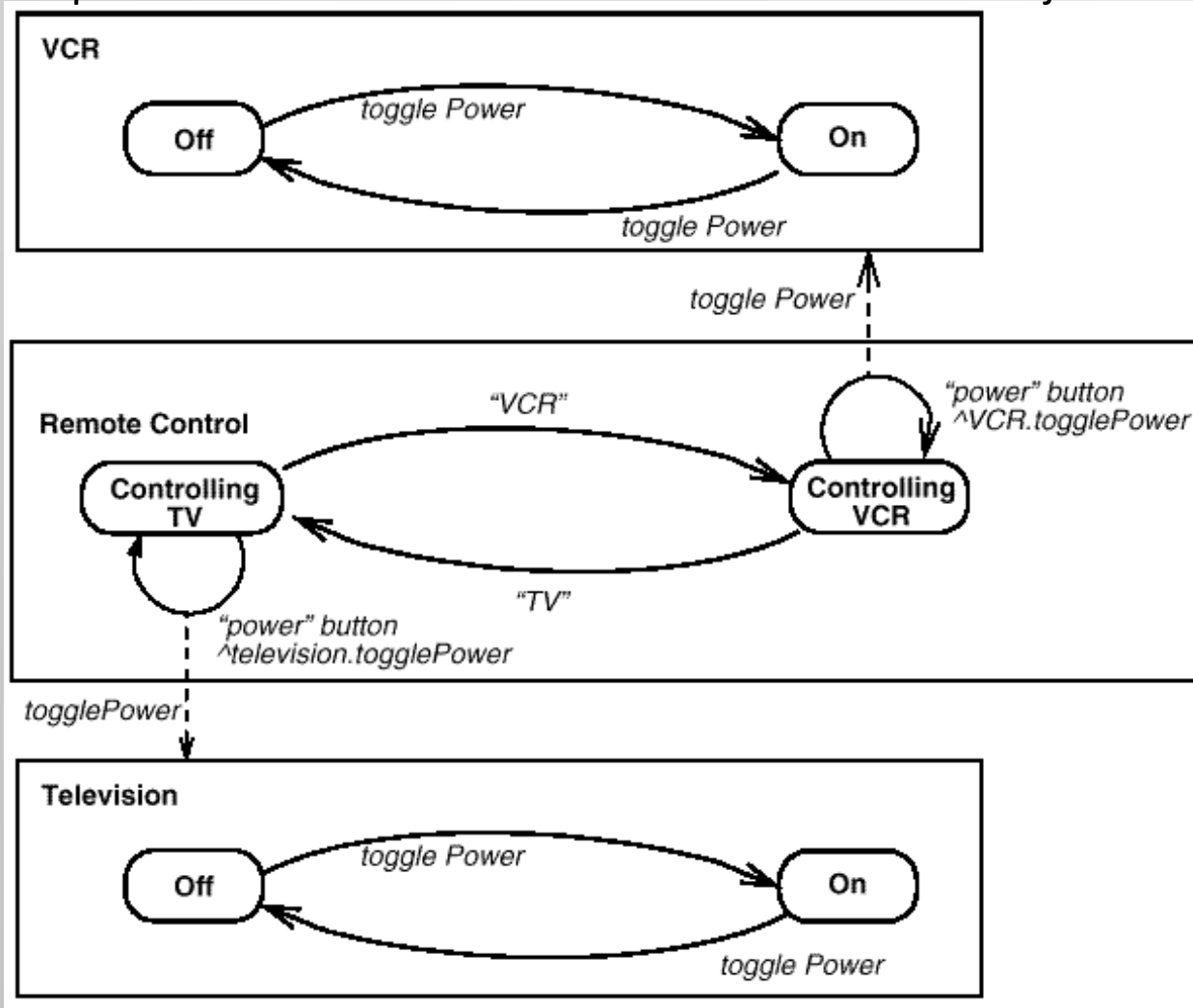
# Transición-Estados

- Regiones Concurrentes (Concurrent Regions)
- Un estado puede dividirse en regiones , los cuales contienen sub-estados que se ejecutan concurrentemente, de manera simultánea e independiente. Se usan fork y join para sincronizar los hilos de concurrencia.



# Transición-Estados : Envío de Mensajes

Además de mostrar la transición de estados por medio de eventos, puede representarse el momento en el cual se envían mensajes a otros objetos.



Esto se realiza mediante una línea punteada dirigida al diagrama de estados del objeto receptor del mensaje.

El diagrama representa un control remoto que puede enviar órdenes de encender o apagar al televisor o a la videograbadora.

## Para más Información

- Libros
  - *UML y Patrones*, Craig Larman, 2da. Edición
  - *El proceso Unificado de Desarrollo de Software*, Booch, Jacobson y Rumbaugh.
  - *Dynamics of Software Development*, Jim McCarthy
  - *Rapid Development*, Steve McConnell
  - *Software Project Survival Guide*, Steve McConnell
  - *Debugging the Development Process*, Steve Maguire
  - *Microsoft Secrets*, Michael A. Cusumano y Richard W. Selby

# Para más Información

- Web sites

- *Enterprise Architect, EA*

- <http://www.sparxsystems.com.au>

- *Object Management Group,*

- <http://www.omg.org/>

- *Rup*

- <http://www-306.ibm.com/software/rational/>

- *UML y modelado ágil*

- [“http://www.agilemodeling.com/essays/umlDiagrams.htm](http://www.agilemodeling.com/essays/umlDiagrams.htm)

# ***Modelo de Análisis***

# Características

- Es una especificación detallada de los requerimientos y actúa como una primera aproximación al diseño del sistema.
- Se utiliza para entender de forma mas precisa los casos de uso y refinarlos expresándolos como *colaboraciones* entre objetos conceptuales.
- Enfoca el análisis del sistema bajo tres perspectivas:
  - La *interfaz* entre el sistema y sus actores
  - La *información* utilizada en el sistema
  - La lógica de *control* del sistema
- Para cada caso de uso se efectúa una *realización* que muestra COMO se lleva a cabo la funcionalidad del caso de uso.

# Importancia del Modelo

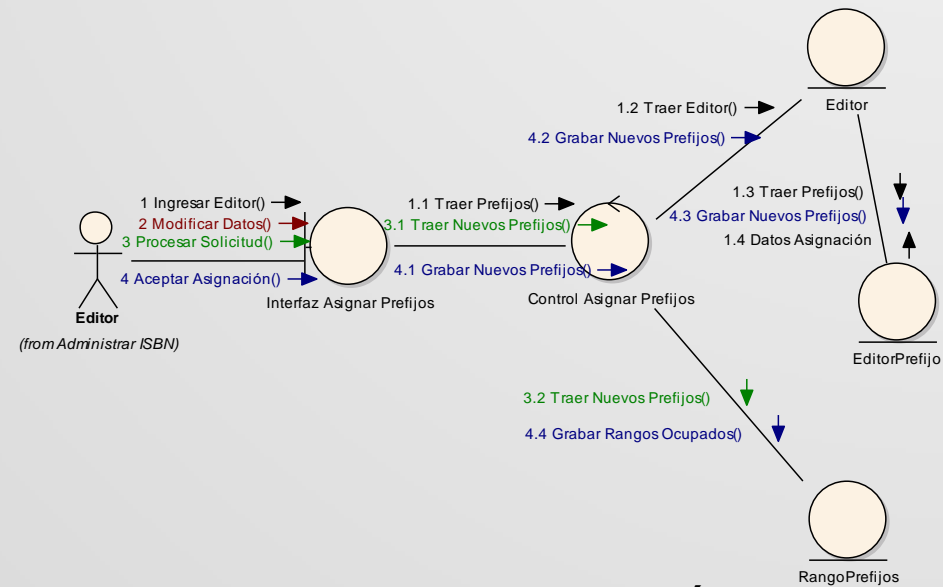
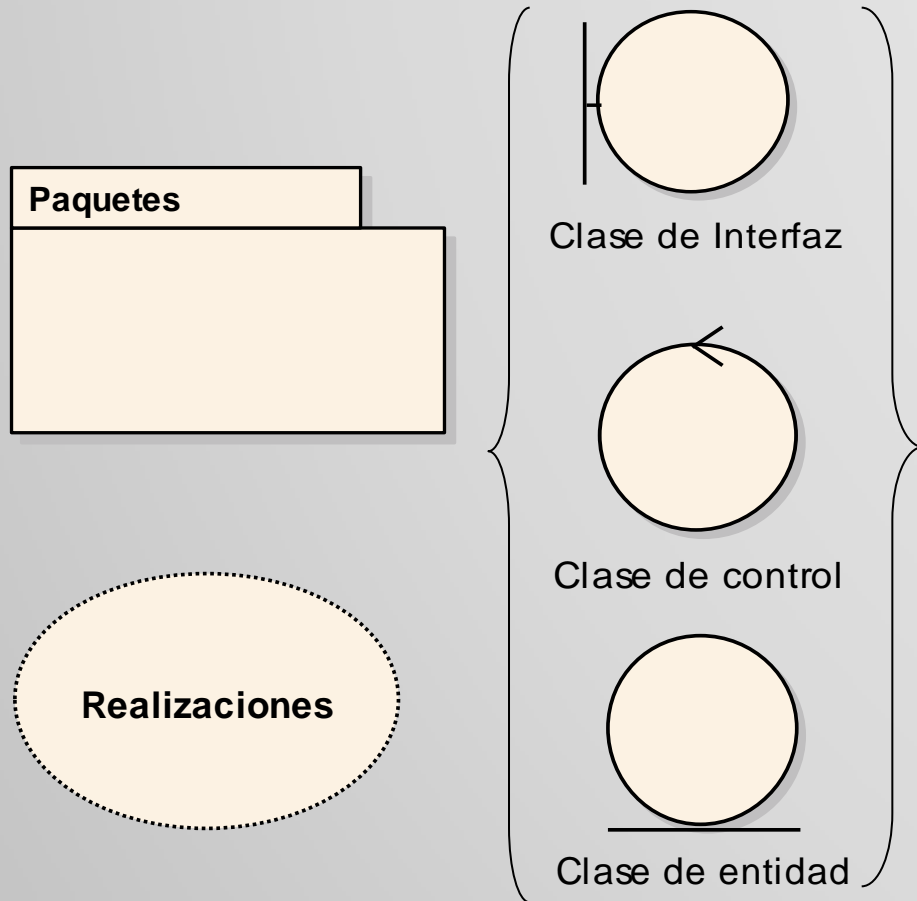
- Ofrece una especificación más precisa de los requisitos
- Se describe utilizando el lenguaje de los desarrolladores (mayor formalismo).
- Estructura los requisitos para facilitar su comprensión, preparación, modificación y mantenimiento.
- Es una primera aproximación al modelo de diseño, evitando la “parálisis” que suele ocurrir cuando se tratan de resolver muchos problemas simultáneamente (pasar directamente del análisis al diseño).



## Comparación con el Modelo Casos de Uso

<b><i>Modelo de casos de uso</i></b>	<b><i>Modelo de análisis</i></b>
Descrito en el lenguaje del cliente.	Descrito en el lenguaje de los desarrolladores.
Vista externa del sistema.	Vista interna del sistema.
Estructurado en casos de uso. Da estructura a la vista externa del sistema.	Estructurado con clases estereotipadas (clases de análisis). Da estructura a la vista interna del sistema.
Usado como un "contrato" entre el cliente y los desarrolladores sobre lo que el sistema debe y no debe hacer.	Usado por los desarrolladores para entender como el sistema debería ser diseñado e implementado.
Puede haber redundancias e inconsistencias entre requerimientos.	No debería haber redundancias e inconsistencias entre requerimientos.
Captura la funcionalidad del sistema.	Define como realizar la funcionalidad dentro del sistema, funciona como una primera aproximación al diseño.
Define casos de uso para que sean analizados en el modelo de análisis.	Define <i>realizaciones</i> de casos de uso, cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

# Composición del Modelo



Diagramas de comunicación

Otros elementos: Documento de descripción de la arquitectura, requerimientos especiales.

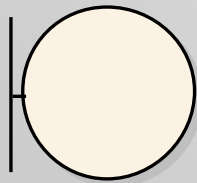
# Clases de Análisis

- Se enfocan solo en los requerimientos funcionales.
- Son “conceptuales” y de mayor granularidad, su comportamiento se define mediante responsabilidades en un nivel más alto y menos formal que las clases de diseño.
- Tienen atributos conceptuales (sin tipo ligado a lenguaje).
- Participan de relaciones simples, asociaciones o herencia, sin tener en cuenta la navegabilidad de las mismas.
- Siempre encajan en uno de tres estereotipos básicos:

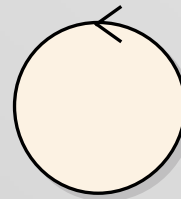
Interfaz

Control

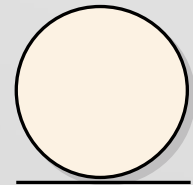
Entidad



Interfaz Asignar Prefijos

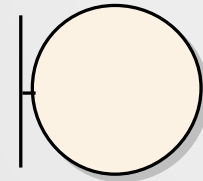


Control Asignar Prefijos



Editor

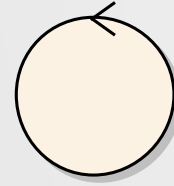
# Clase de Interfaz



Interfaz Asignar Prefijos

- Se utiliza para modelar la interfaz entre el sistema y sus actores.
- Representan abstracciones de ventanas, formularios, paneles, interfaces de comunicación, interfaces de impresoras, sensores, terminales y API.
- Cada clase de interfaz debería asociarse con al menos un actor, y viceversa.

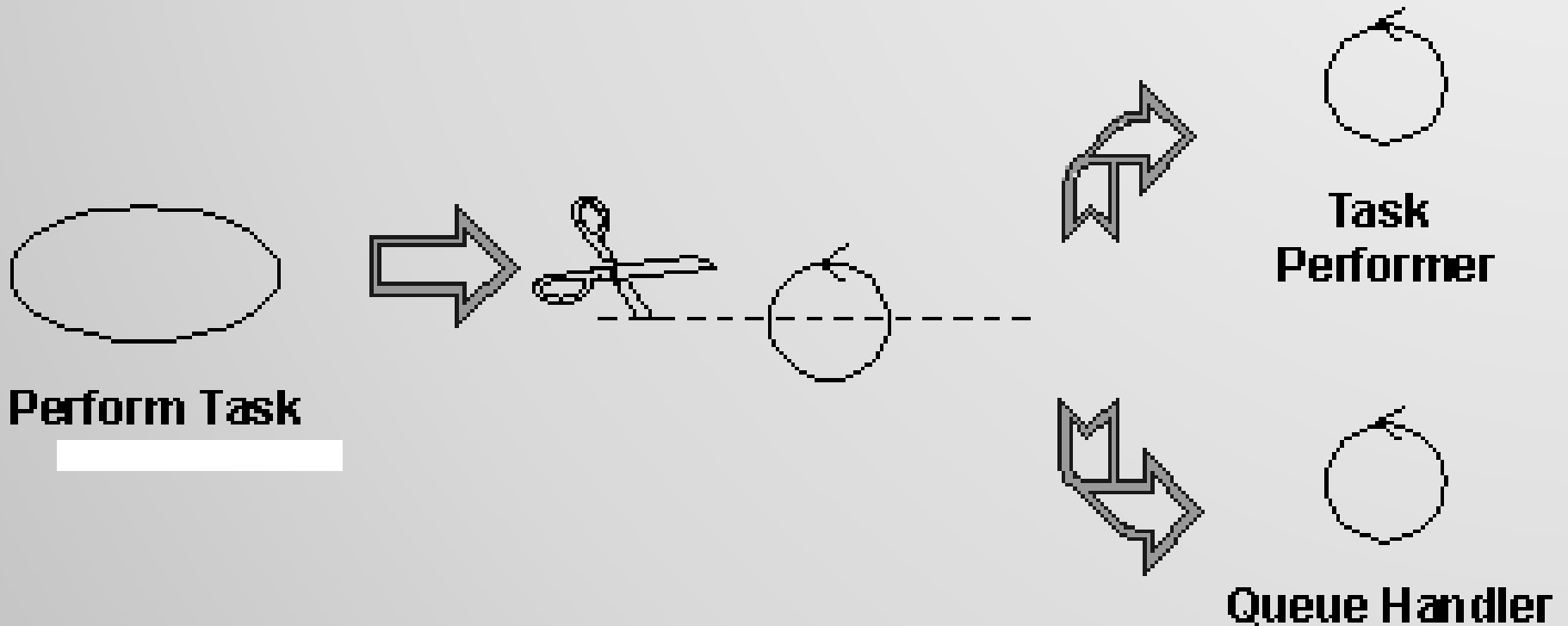
# Clase de Control



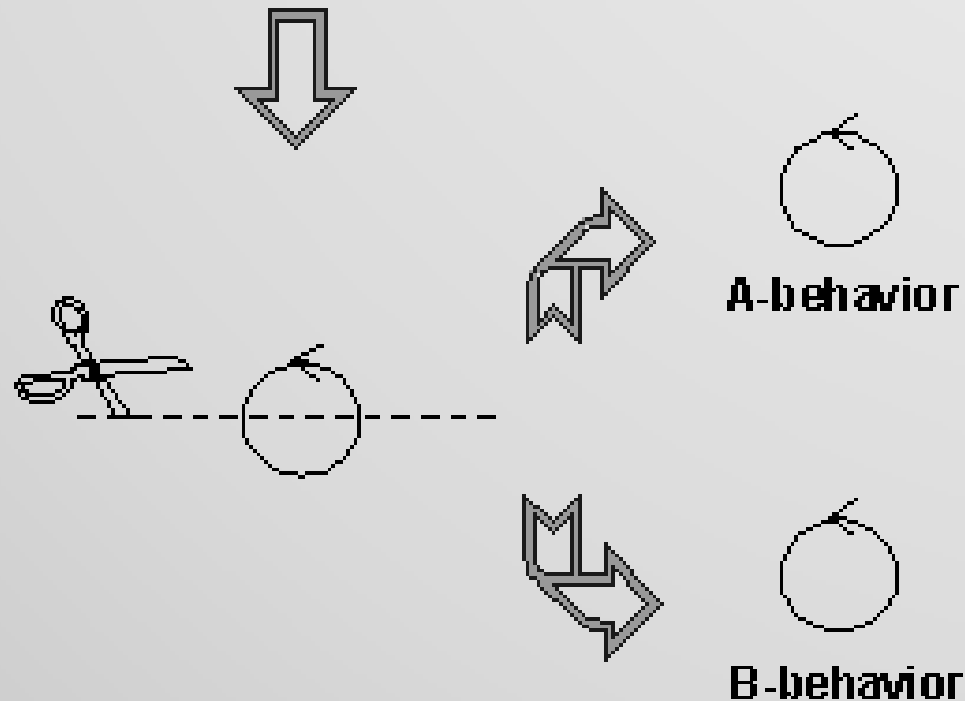
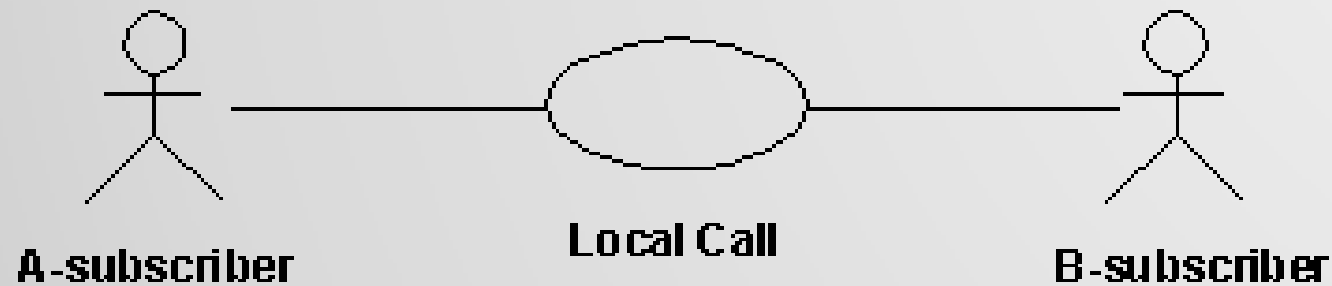
Control Asignar Prefijos

- representan coordinación, secuencia, transacciones y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto.
- Sirven para modelar los aspectos dinámicos del sistema.

# Conviene dividir las clases complejas

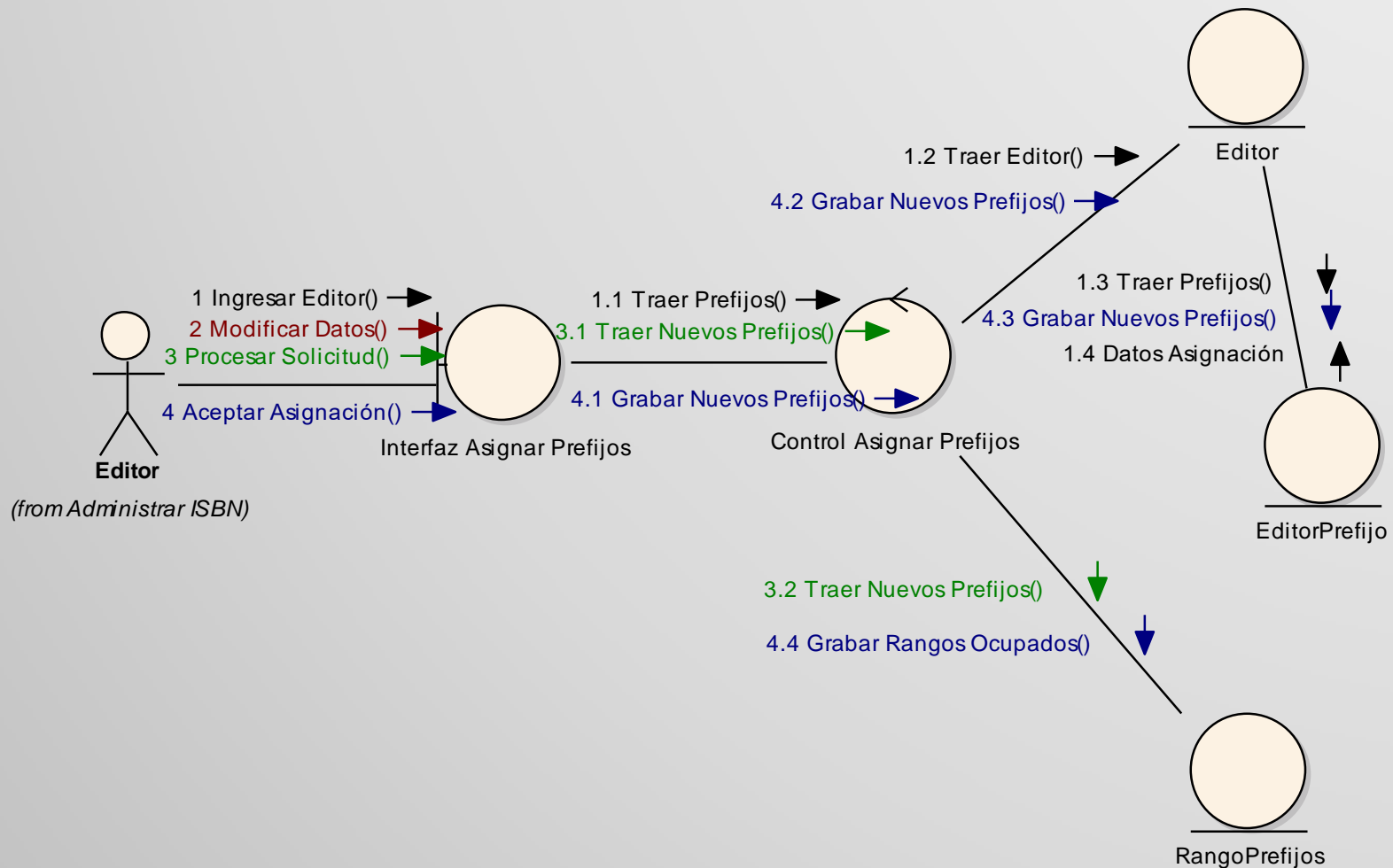


# Dividir las clases de control cuando dos actores comparten la misma clase



# Diagramas de comunicación

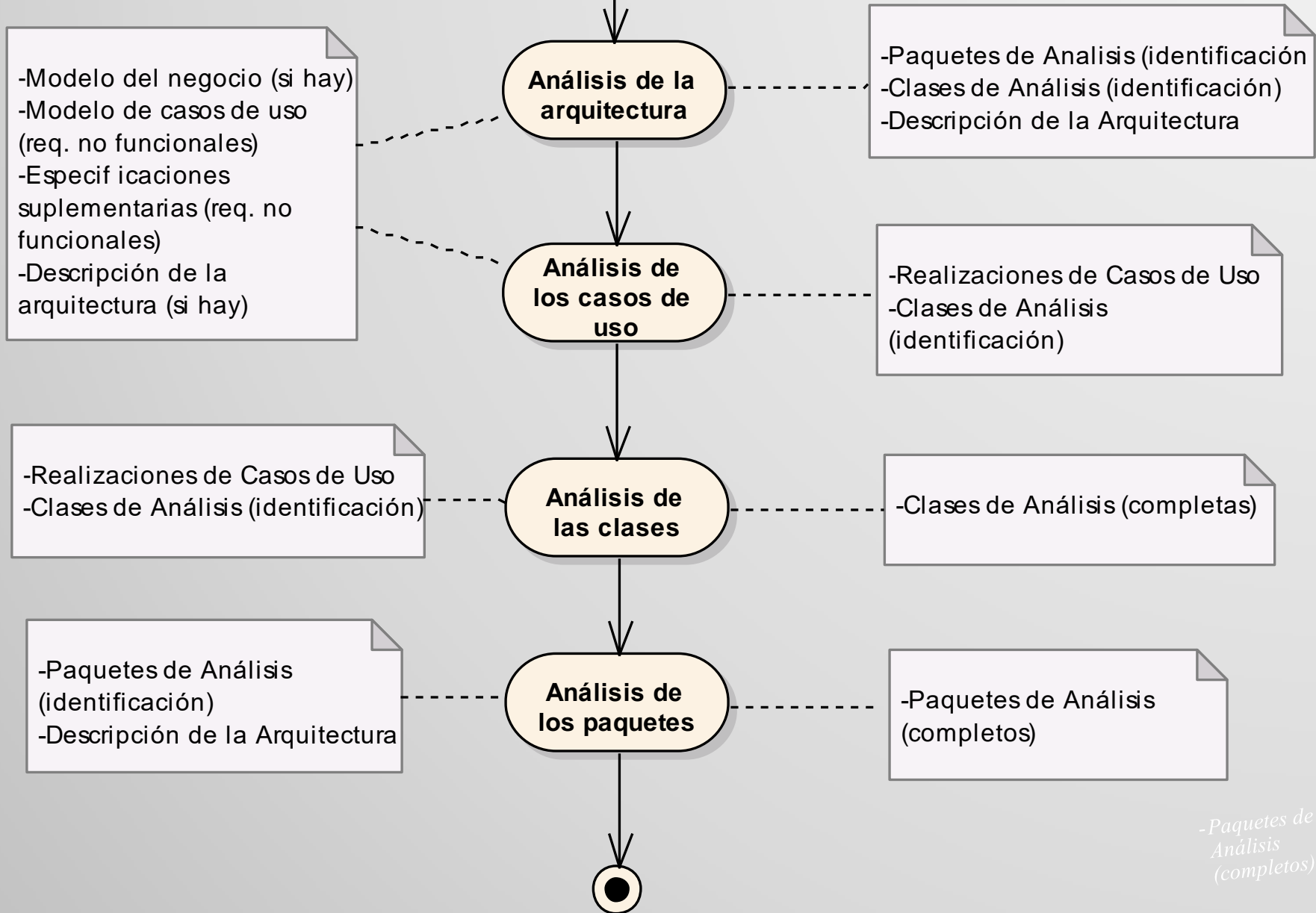
Consiste en describir la forma en que los objetos pertenecientes a las clases identificadas interactúan entre sí.





***Actividades para la  
Construcción del Modelo de  
Análisis***

# Actividades para su construcción



# Análisis de la Organización

- Identificación de paquetes de análisis.
  - Proporcionan un medio para organizar el modelo de análisis en piezas más pequeñas.
    - Los casos de uso requeridos para dar soporte a un determinado proceso de negocio.
    - Ídem para dar soporte a un determinado actor.
    - Los casos de uso que están relacionados.
- Identificación de paquetes de servicio.
  - Suele hacerse cuando el trabajo de análisis está avanzado.
- Identificación de clases de entidad obvias.
  - Prepara una lista con las clases de entidad más importantes que se identificaron durante la captura de los requisitos.
- Identificación de requisitos especiales comunes.
  - Anotarlo de forma que puedan ser tratados en las subsiguientes actividades de diseño e implementación.

# Análisis de un Caso de Uso

- Identificar las clases de análisis cuyos objetos 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.

# Clases de Intefaz

- Identificar una clase de interfaz central por cada actor humano, representando la ventana o pantalla con la que el actor interactúa con el sistema para la realización del caso de uso (considerar la reutilización de las interfaces).
- Identificar una clase de interfaz central por cada actor no humano (sistema externo, software, hardware, terminales, dispositivos, etc.). Esta clase representa la interfaz de comunicaciones con el actor.

# Clases de Control

Inicialmente, identificar una clase de control encargada de manejar el control y la coordinación necesarios para la realización del caso de uso (una por Caso de Uso).

# Clases de Control

Luego, refinar de acuerdo a los sig. criterios:

- Si el actor es quien determina la mayoría del control, se puede representar el control en la clase de interfaz.
- Si el control es demasiado complejo, se puede descomponer en varias clases de control más específicas.
- Si la misma clase de control es utilizada por dos actores diferentes (interfaz mediante), es recomendable descomponer en dos clases de control.

# Clases de Entidad

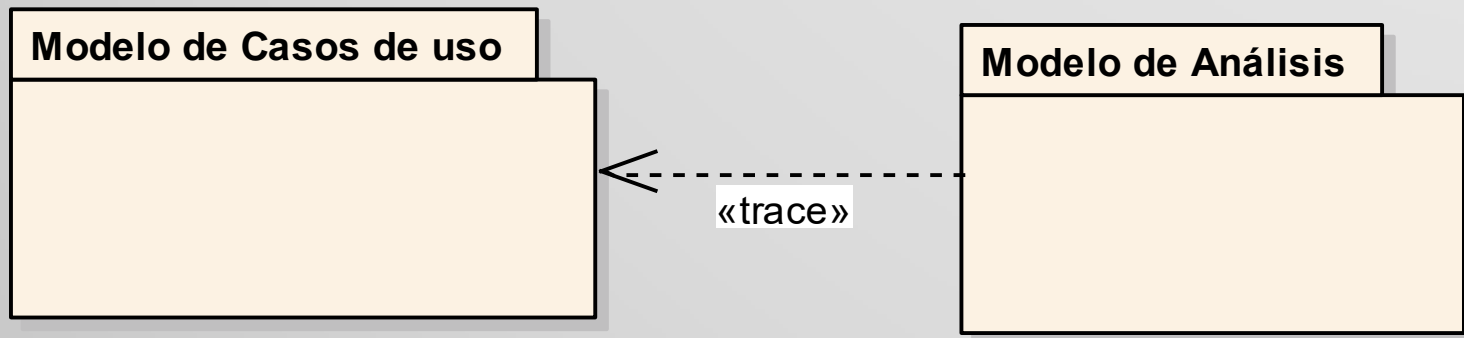
Estudiar en detalle la especificación del caso de uso y la documentación del negocio, considerando la información que debería ser almacenada o manipulada en la realización del caso de uso.

Evitar modelar toda la información manipulada como clases de entidad, muchas veces resulta más adecuado modelarla como atributos de una clase.

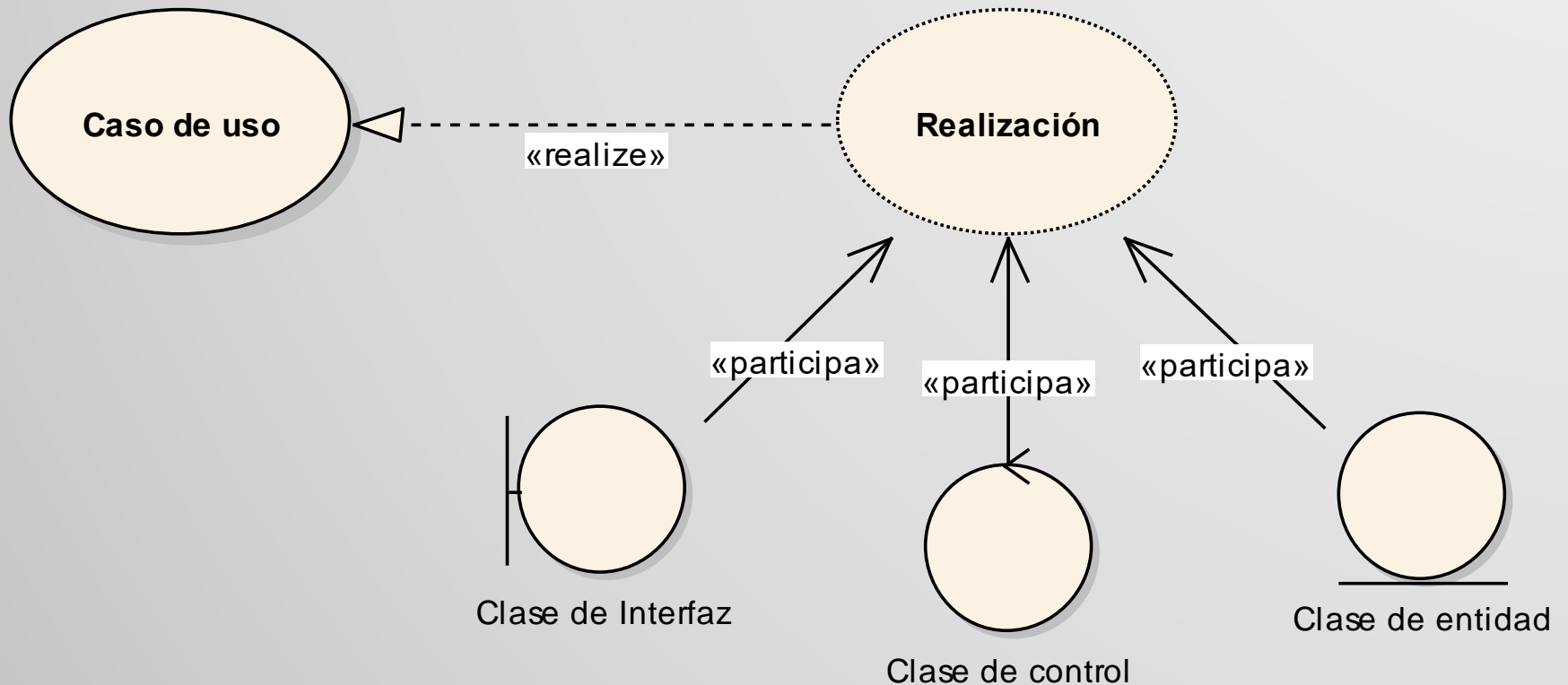


# Realización de Caso de Uso-Análisis

- Es una colaboración dentro del modelo de análisis que describe como se lleva a cabo y se ejecuta un caso de uso determinado en términos de las clases del análisis y de sus objetos del análisis en interacción.
- Proporciona una traza directa hacia un caso de uso concreto del modelo de casos de uso.



# Las realizaciones brindan TRAZABILIDAD:



# Realización de Caso de Uso-Análisis

- Diagrama de comunicación
  - Se muestran los objetos con las asociaciones correspondientes entre ellos. Los mensajes se agregan a las asociaciones y se representan con flechas cortas que indican la dirección del flujo del mensaje. La secuencia de mensajes se identifica con el esquema de numeración.

