Tema 6º: Diseño Orientado a Objetos

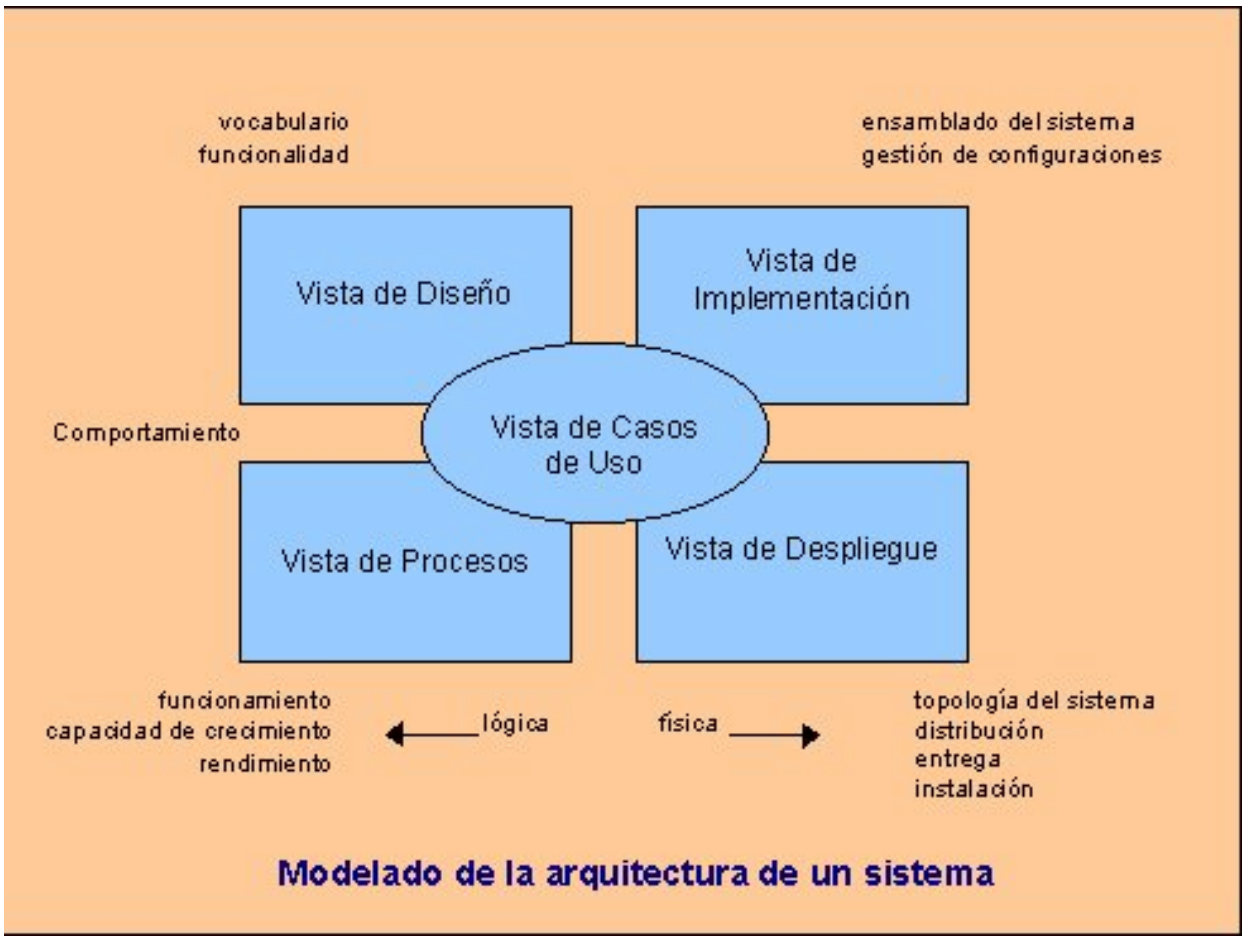
- Diseño preliminar y Diseño detallado
- Modelado de la Arquitectura del Sistema
- Abstracciones y mecanismos clave
- Elementos básicos del Diseño Orientado a Objetos
  - Diagramas de interacción
    - Diagramas de secuencia
    - Diagramas de colaboración
  - Diagramas de Clases y consulta de patrones de diseño.
  - Diagramas de objetos
  - Diagramas de actividades
  - Diagramas de estados
  - Diagramas de componentes
  - Diagramas de despliegue

Diseño Orientado a Objetos

- Diseño preliminar
  - Incluye los mismos diagramas del diseño detallado pero a nivel más sencillo (con menos detalle)
  - Es necesario para estimar costes y tiempos antes de realizar el diseño detallado
  - Es el primer paso dentro del proceso iterativo de diseño
- Diseño detallado
  - Es un refinamiento sucesivo de diagramas de diseño

Modelado de la Arquitectura del Sistema

[Booch 99, capítulos 1,2, 31]



Modelado de la Arquitectura del Sistema

Cinco vistas interrelacionadas

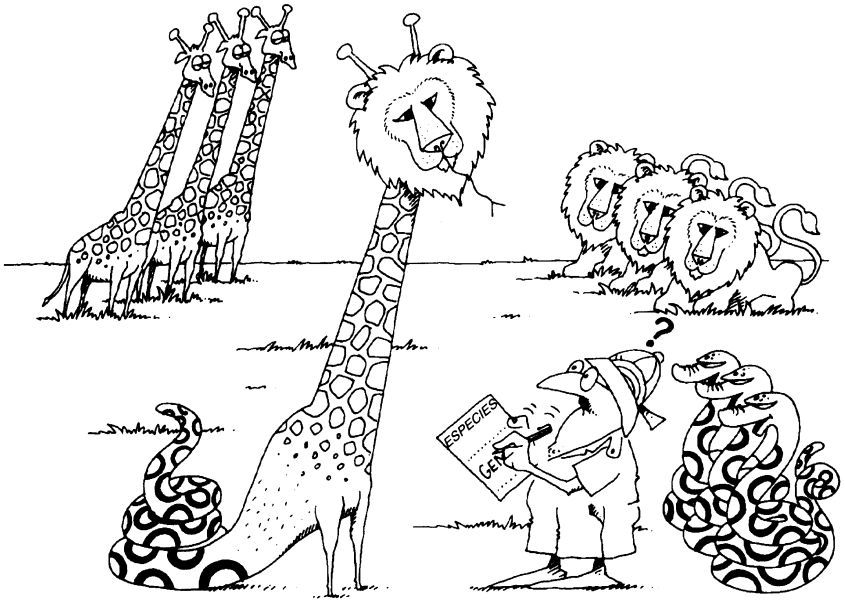
[Booch 99, capítulos 1,2, 31]

- **Vista de casos de uso**
  - Muestra los requisitos del sistema tal como es percibido por los usuarios finales, analistas, y encargados de pruebas. Utiliza los diagramas:
    - Caos de uso
    - Diagramas de Interacción
    - Diagramas de estados
    - Diagramas de actividades
- **Vista de diseño**
  - Captura el vocabulario del espacio del problema y del espacio de la solución. Utiliza los diagramas
    - Diagramas de clases
    - Diagramas de objetos
    - Diagramas de interacción
    - Diagramas de estados
    - Diagramas de actividades
- **Vista de procesos**
  - Modela la distribución de los procesos e hilos (*threads*)
  - Emplea los mismos diagramas de las vista de diseño, pero poniendo especial atención en las clases activas y los objetos que representan hilos y procesos
- **Vista de implementación**
  - Modela los componentes y archivos que se utilizan para ensamblar y hacer disponible el sistema físico. Emplea los diagramas:
    - Para modelar aspectos estáticos: Diagramas de Componentes
    - Para modelar aspectos dinámicos:
      - Diagramas de Interacción
      - Diagramas de Estados
      - Diagramas de Actividades
- **Vista de despliegue**
  - Modela los nodos de la topología hardware sobre la que se ejecuta el sistema
    - Para modelar aspectos estáticos: Diagramas de Despliegue
    - Para modelar aspectos dinámicos:
      - Diagramas de Interacción
      - Diagramas de Estados
      - Diagramas de Actividades

La importancia de una clasificación correcta (I)

[Booch 94]

- La identificación de clases y objetos es la parte más difícil del diseño orientado a objetos
- La identificación de clases y objetos implica descubrimiento e invención
- No hay recetas fáciles para identificar clases y objetos
- Clasificar es agrupar cosas que tienen una estructura común o exhiben un comportamiento común
- La clasificación ayuda a identificar jerarquías de generalización, especialización y agregación entre clases
- La clasificación también proporciona una guía para tomar decisiones sobre modularización

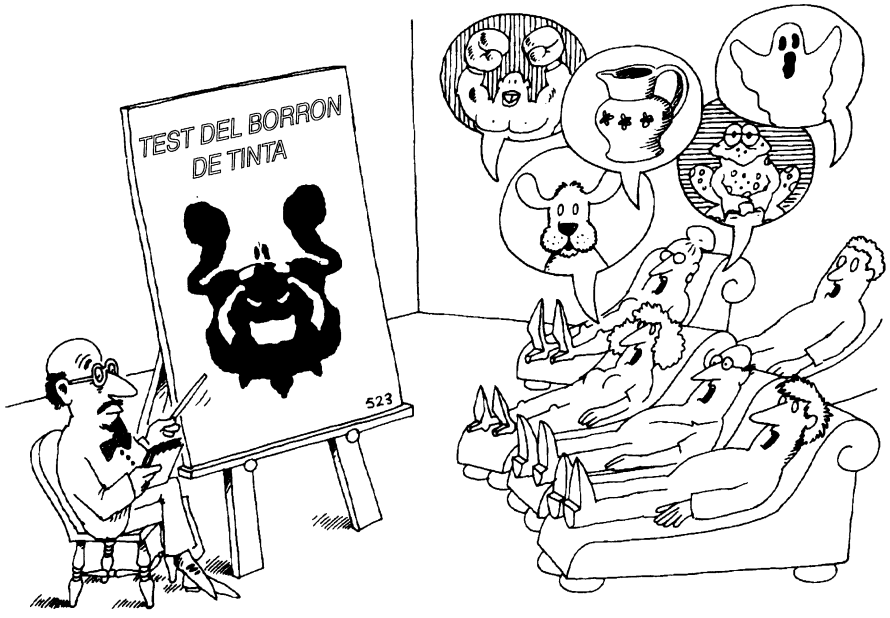


La clasificación es el medio por el cual ordenamos el conocimiento.

La importancia de una clasificación correcta (II)

[Booch 94]

- La dificultad de la clasificación
  - Un objeto debe tener una frontera o interfaz nítida
  - Sin embargo a veces las fronteras son difusas
  - El descubrimiento de un orden no es tarea fácil…; sin embargo, una vez que se ha descubierto el orden, no hay dificultad alguna en comprenderlo
- Ejemplos de clasificación
  - Clasificación de los organismos en géneros y especies
- La naturaleza incremental e iterativa de la clasificación
  - Cualquier clasificación es relativa a la perspectiva del observador que la realiza
  - No hay clasificaciones perfectas (unas son mejores que otras en función del dominio del problema)

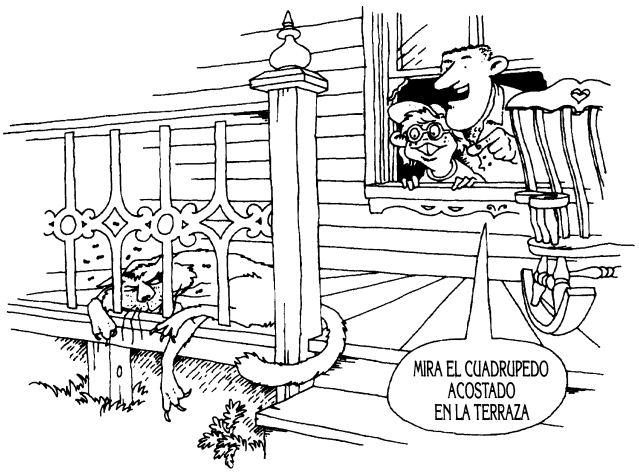


Diferentes observadores pueden clasificar el mismo objeto de distintas formas.

Abstracciones clave

son clases u objetos que forman parte del dominio del problema [Booch 94]

- Identificación de las abstracciones clave
  - Búsqueda de las abstracciones clave
    - Depende de las características de cada dominio
    - Descubrir las abstracciones de los expertos en el dominio
    - Se inventan nuevas abstracciones que son útiles para el diseño o la implementación
    - Se compara con otras abstracciones buscando similitudes
    - Se recomienda el uso de escenarios para guiar el proceso de identificación de clases y objetos
  - Refinamiento de las abstracciones clave
    - Cada abstracción debe ubicarse en el contexto de la jerarquía de clases y objetos
    - La colocación de clases y objetos en los niveles correctos de abstracción es una tarea difícil
    - Se recomienda elegir los identificadores con el siguiente criterio
      - **objetos:** *los identificadores deben ser* ***nombres propios***
      - **clases:** *los identificadores deben ser* ***nombres comunes***
      - **operaciones de modificación:** *los identificadores deben ser* ***verbos activos***
      - **operaciones de selección:** *los identificadores deben ser* ***verbos interrogativos o verbos en forma pasiva***

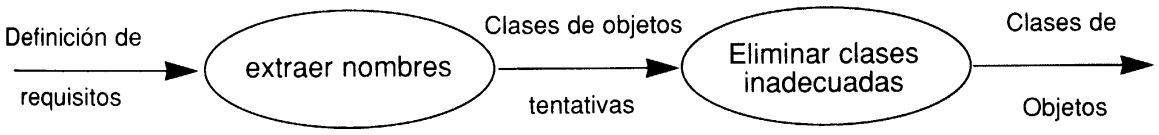


Las clases y objetos deberían estar al nivel de abstracción adecuado: ni demasiado alto ni demasiado bajo.

Identificación de clases y objetos

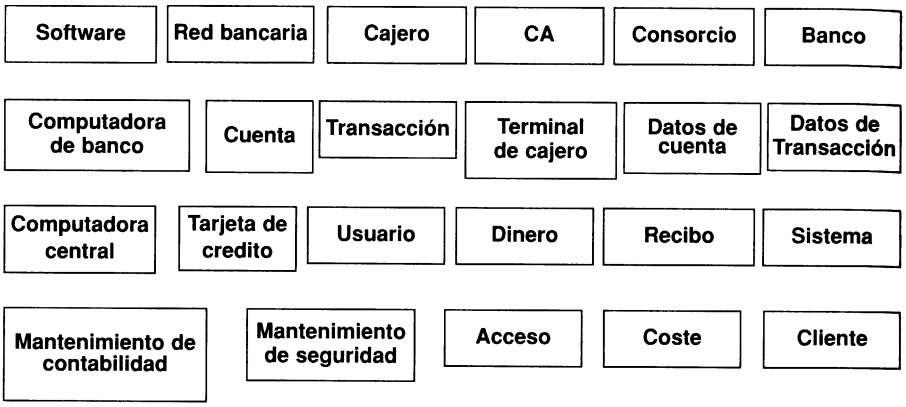
[Rumbaught 91]

- Enumerar los candidatos a clases (siguiendo uno de los métodos de análisis, por ejemplo subrayando sustantivos)
- No se preocupe demasiado por la herencia, ni por las clases de alto nivel
- Eliminar clases redundantes
- Eliminar clases irrelevantes
- Eliminar clases vagas
- Identificar atributos



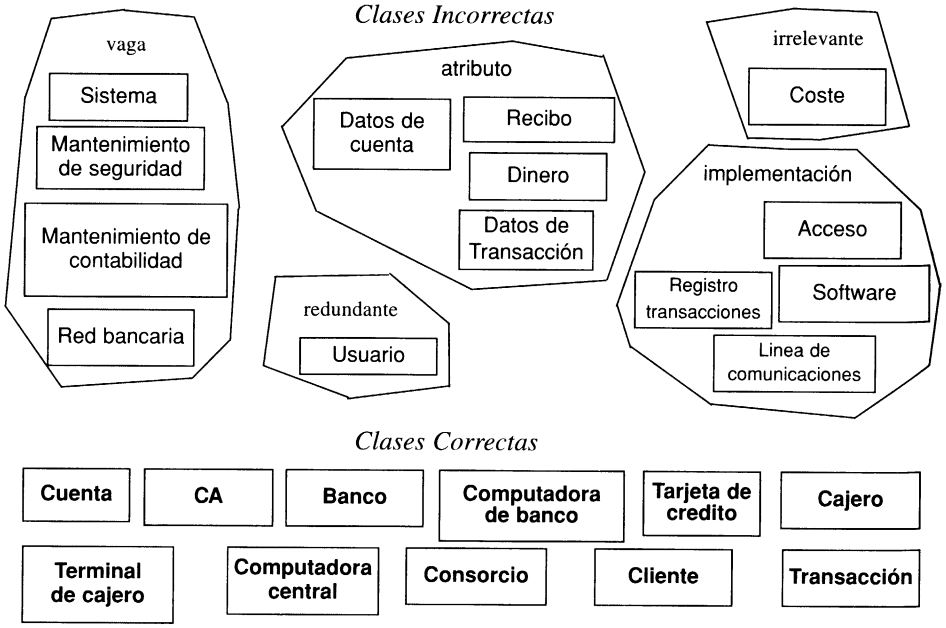
Ejemplo cajero automático (CA)

[Rumbaught 91]



Clases del CA extraídas de los nombres de la definición del problema

Clases de CA identificadas a partir del conocimiento del dominio del problema

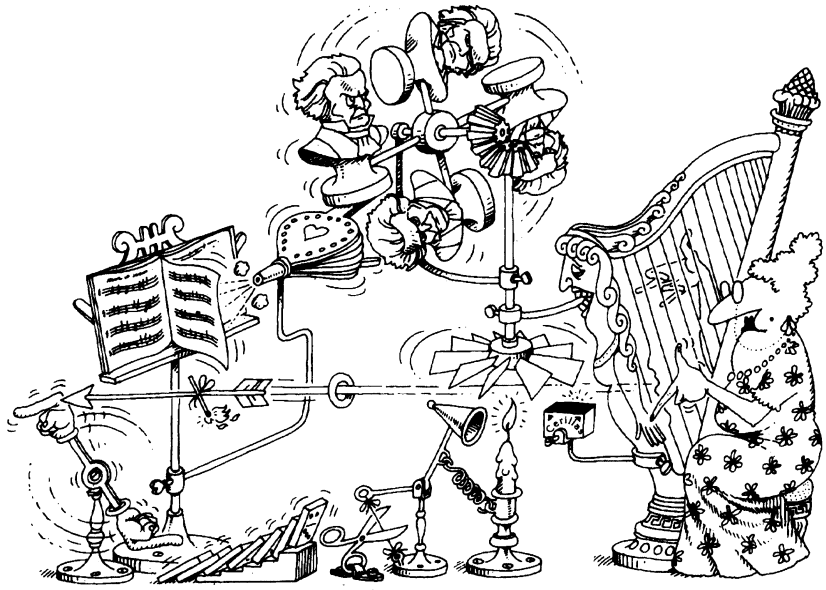


Eliminación de clases innecesarias en el problema del CA.

Mecanismos

Denotan las decisiones estratégicas de diseño respecto a la actividad de colaboración entre muchos tipos diferentes de objetos [Booch 94]

- Identificación de mecanismos
  - Búsqueda de mecanismos
    - Se utilizan escenarios
    - El interfaz de una clase individual es una decisión **táctica**, sin embargo los mecanismos son decisiones **estratégicas** en las que el desarrollador elige entre un conjunto de alternativas influyendo factores como coste, fiabilidad, facilidad de fabricación y seguridad



Los mecanismos son los medios por los cuales los objetos colaboran para proporcionar algún comportamiento de nivel superior.

Diagramas de Interacción

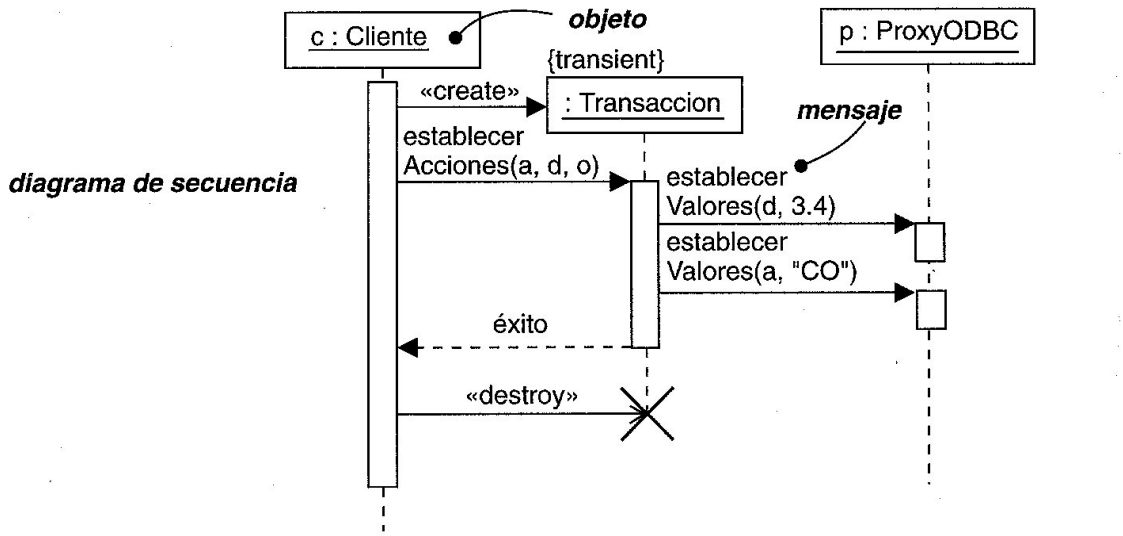
[Booch 99, capítulo 18]

- Son de dos tipos
  - Diagramas de Secuencia
  - Diagramas de Colaboración
- Modelan aspectos dinámicos del sistema
- Un diagrama de interacciones se usa para realizar una traza de la ejecución de un escenario
- A cada escenario le corresponde un diagrama de Interacción, que se suele poner con el mismo código pero con otra letra (E1.1 se corresponde con S 1.1 y C 1.1).
- Un diagrama de interacción muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos
- Un diagrama de secuencia es un diagrama de interacción que destaca la **ordenación temporal** de los mensajes
- Un diagrama de colaboración es un diagrama de interacción que destaca la **organización estructural** de los objetos que envían y reciben mensajes
- Un diagrama de interacción contiene
  - Objetos
  - Enlaces
  - Mensajes
- También puede contener
  - Notas
  - Restricciones

Diagramas de Secuencia (I)

[Booch 99, capítulo 18]

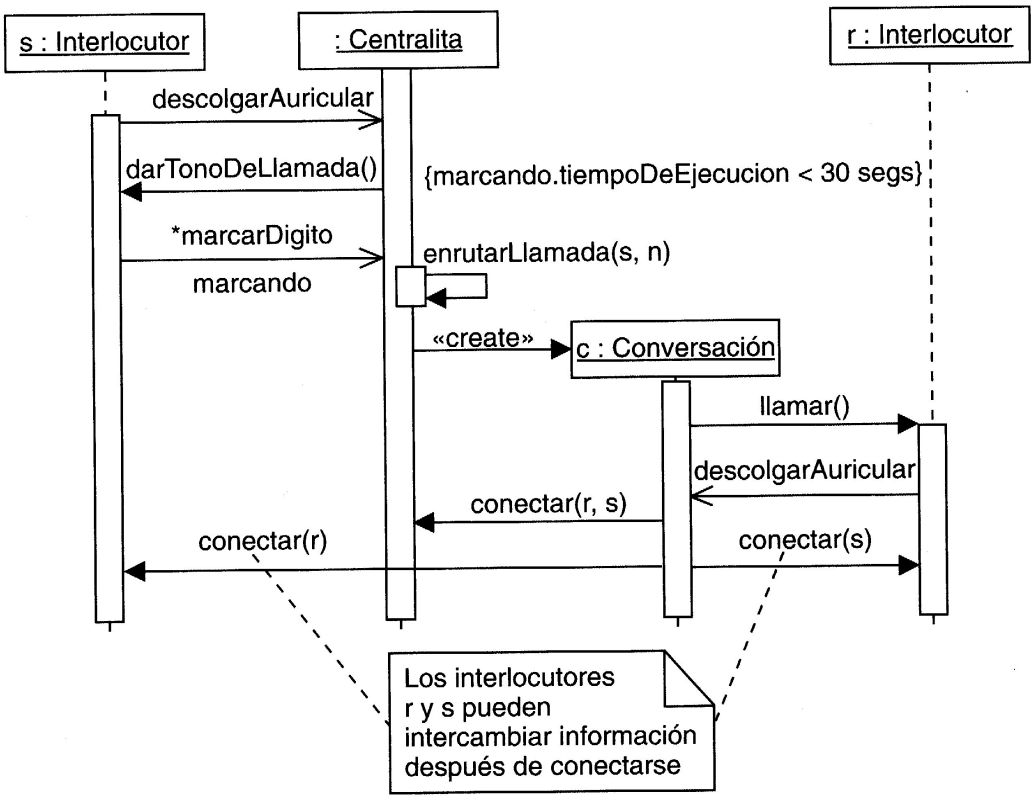
- Se hace un diagrama de secuencia por cada escenario
- Permiten en las fases iniciales de diseño
  - Razonar más en detalle como es el comportamiento de un escenario
  - Obtener nuevas clases y objetos en el escenario (enriquecimiento del diccionario de clases)
  - Detectar cuales son los métodos de las clases, al observar como se relacionan los objetos entre sí para llevar a cabo la tarea encomendada en el escenario
- Se utilizan en las fases de prueba para validar el código
- Si se desea más detalle se utilizan los diagramas de Colaboración



Diagramas de Secuencia (II)

[Booch 99, capítulo 18]

- Los diagramas de secuencia son mejores que los diagramas de colaboración para capturar la semántica de los escenarios en un momento temprano del ciclo de desarrollo.
- Un diagrama de secuencia destaca la ordenación temporal de los mensajes
- Se coloca a la izquierda el objeto que inicia la interacción, y el objeto subordinado a la derecha
- La línea de vida de un objeto es la línea vertical.
  - Es discontinua cuando el objeto no existe
  - Es gruesa y hueca formando un rectángulo cuando existe el objeto y se denomina foco de control
  - La vida del objeto comienza cuando recibe un mensaje estereotipado como <<create>>
  - La vida del objeto finaliza con la recepción de un mensaje estereotipado como <<destroy>> y muestra un aspa indicando el final



Refinamiento de Diagramas de Secuencia (III)

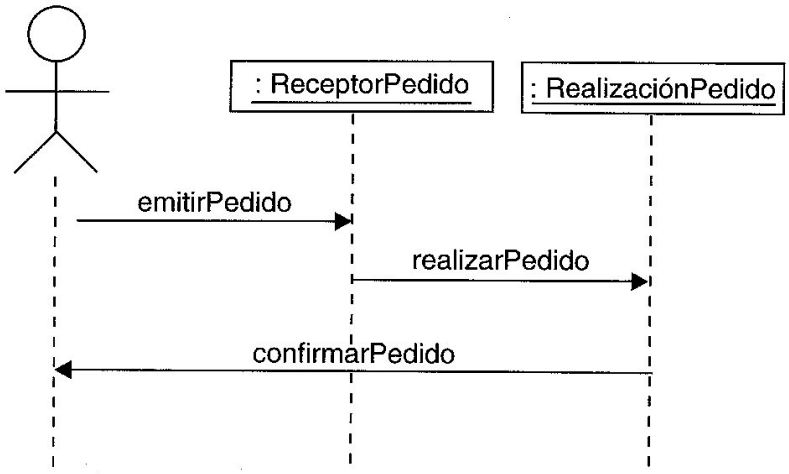


Diagrama de secuencia a un alto nivel de abstracción y poco detallado

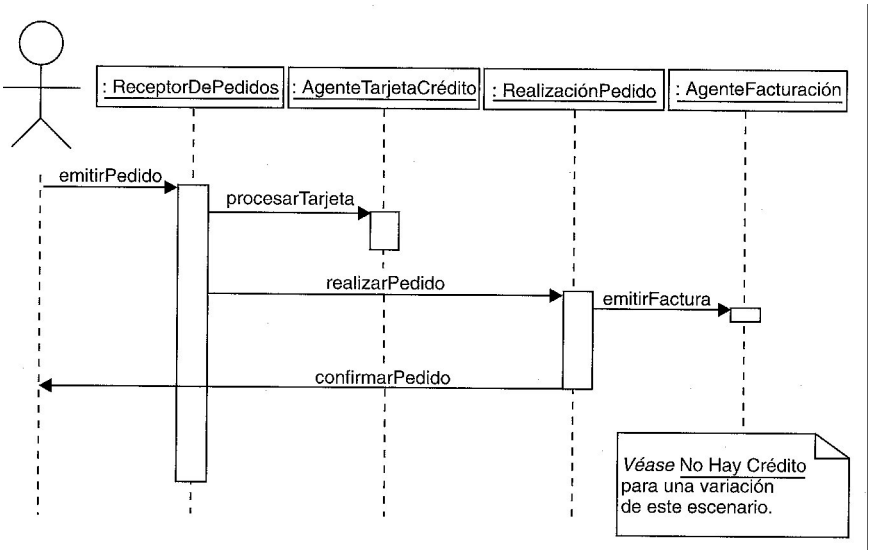
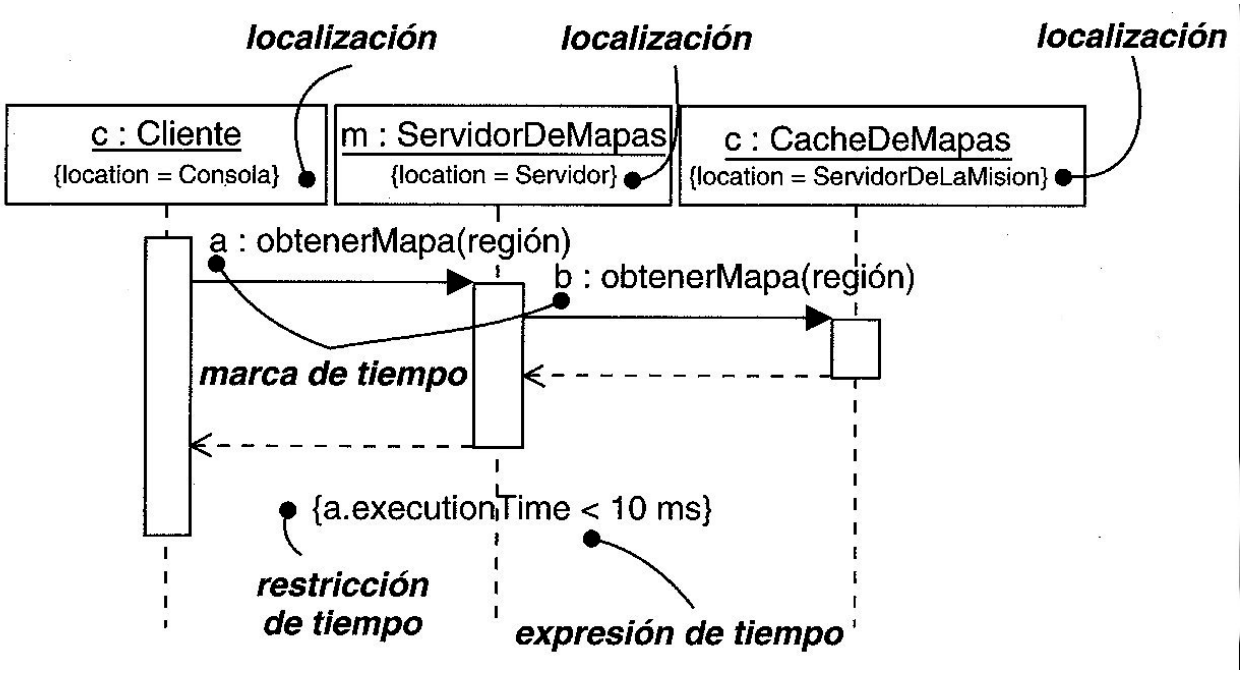


Diagrama de secuencia con menos nivel de abstracción y más detalle

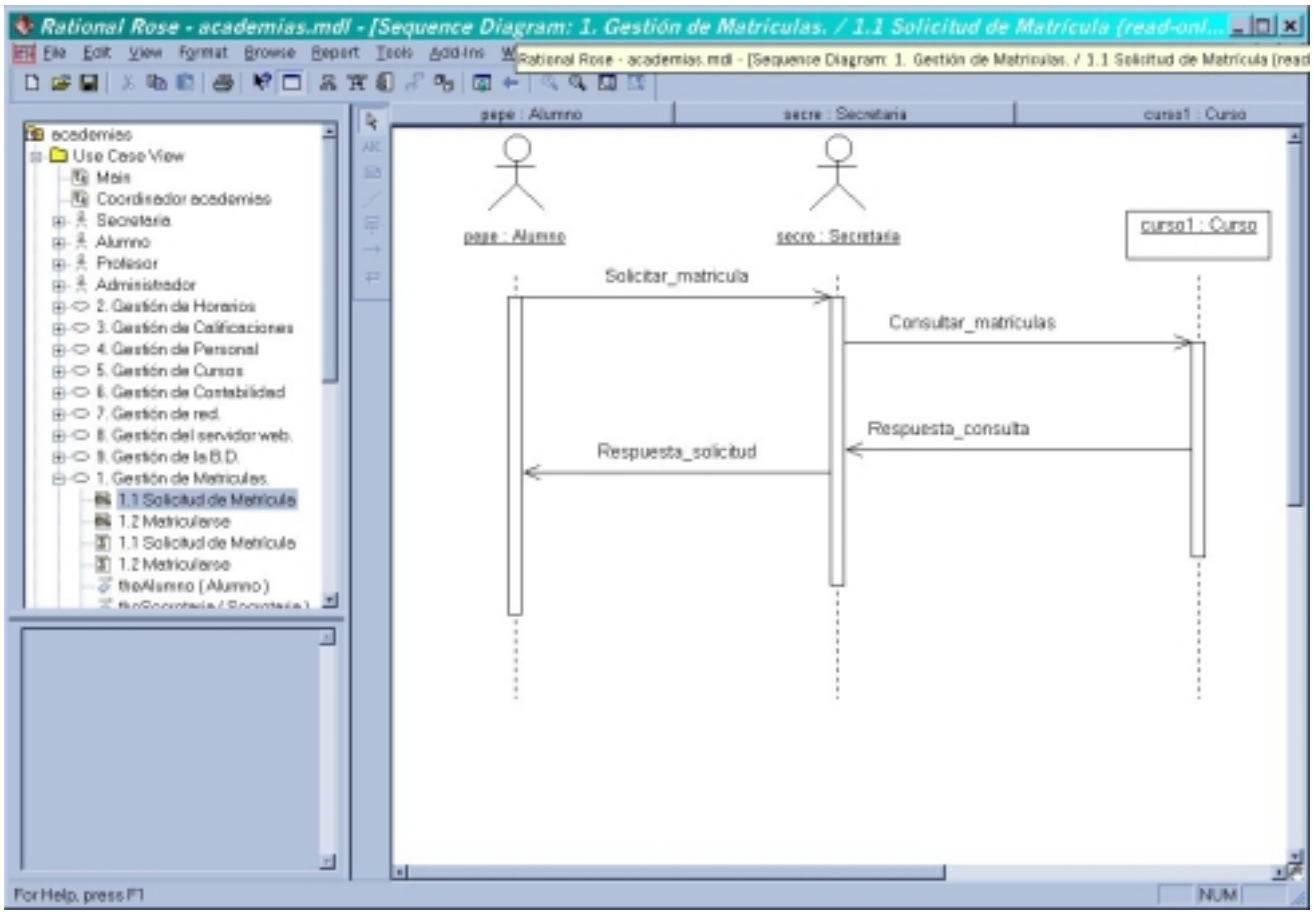
Diagramas de Secuencia (IV)

Restricciones de tiempo y localización

[Booch 99, capítulo 23]



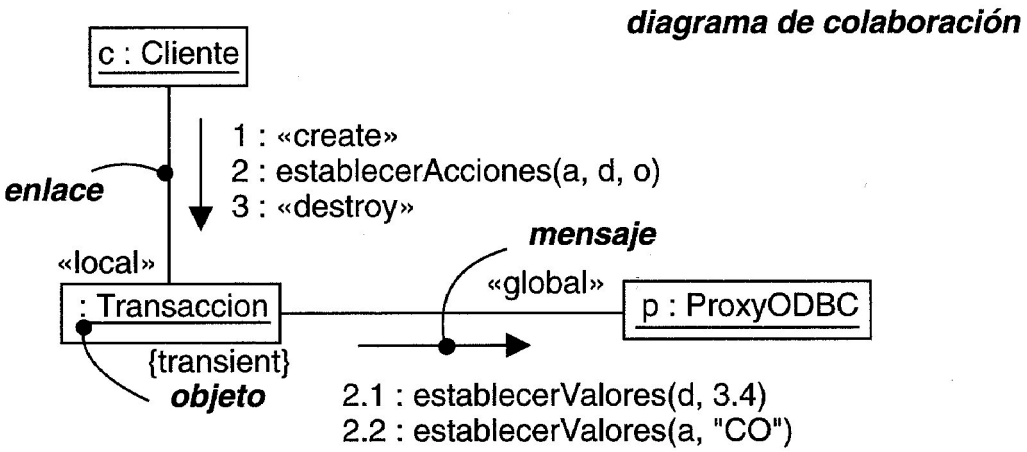
Diagramas de Secuencia (V)
en Rational Rose®



Diagramas de Colaboración (I)

[Booch 99, capítulo 18]

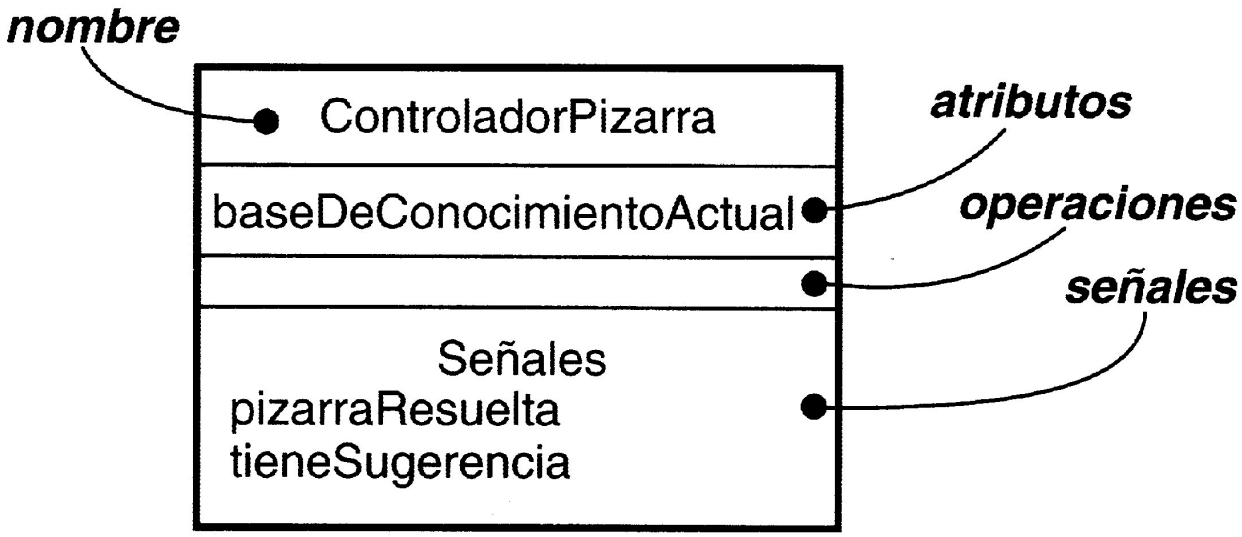
- Permiten profundizar en el nivel de detalle en los diagramas de Secuencia
- Expresan las colaboraciones de los objetos en tiempo de ejecución.
- Dan una visión del flujo de control en el contexto de la organización estructural de los objetos que colaboran.
- Los diagramas de colaboración tienen varias características que los distinguen de los diagramas de secuencia
  - El camino
    - Se puede asociar un estereotipo de camino a cada extremo del enlace
      - `<<local>>` El objeto es visible porque es local al emisor
      - `<<parameter>>` El objeto es visible porque es un parámetro
      - `<<global>>` El objeto es visible porque tiene alcance global
      - `<<self>>` El objeto es visible porque es el emisor del mensaje
  - El número de secuencia indica la ordenación temporal de los mensajes
    - Los mensajes se preceden de un número, que se incrementa secuencialmente (1, 2, 3,...)
    - Para representar el anidamiento se utiliza la numeración decimal de Dewey (dentro del mensaje 2; 2.1 es el primer mensaje dentro de 2; 2.2 el segundo; etc.)
  - Flujos que pueden modelar iteración. Por ejemplo
    - `*[i:=1..n]`
    - o sólo `*` si se quiere indicar iteración sin indicar detalles
  - Flujos que modelan bifurcaciones. Por ejemplo: `[x>0]`
  - Tanto en iteración como en bifurcación UML no impone formato de la expresión entre corchetes; se puede utilizar pseudocódigo o la sintaxis de un lenguaje de programación específico.



Diagramas de Colaboración (II)
Modelado de procesos e hilos

[Booch 99, capítulo 22]

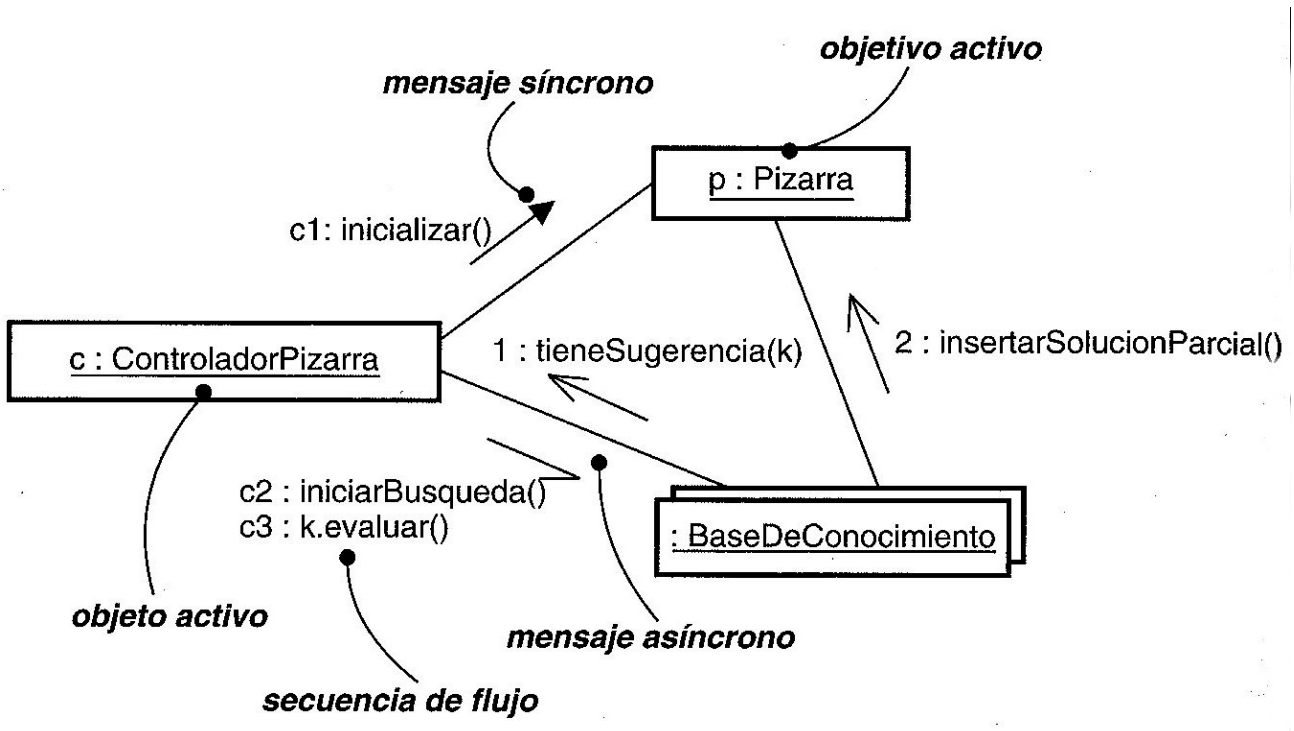
- Un **objeto activo** es un objeto que tiene un proceso o hilo y puede iniciar una actividad de control
- Una **clase activa** es una clase cuyas instancias son objetos activos
  - Una clase activa se representa con un rectángulo con líneas gruesas
- Un **proceso** es un flujo pesado que se puede ejecutar concurrentemente con otro procesos
- Un **hilo** es un proceso ligero que se puede ejecutar concurrentemente con otro hilos dentro del mismo proceso
- Los procesos e hilos se representan como clases activas estereotipadas.
  - UML tiene dos estereotipos estándar para aplicar a las clases activas
    - process
    - thread



Diagramas de Colaboración (III)
Modelado de procesos e hilos
Comunicación

[Booch 99, capítulo 22]

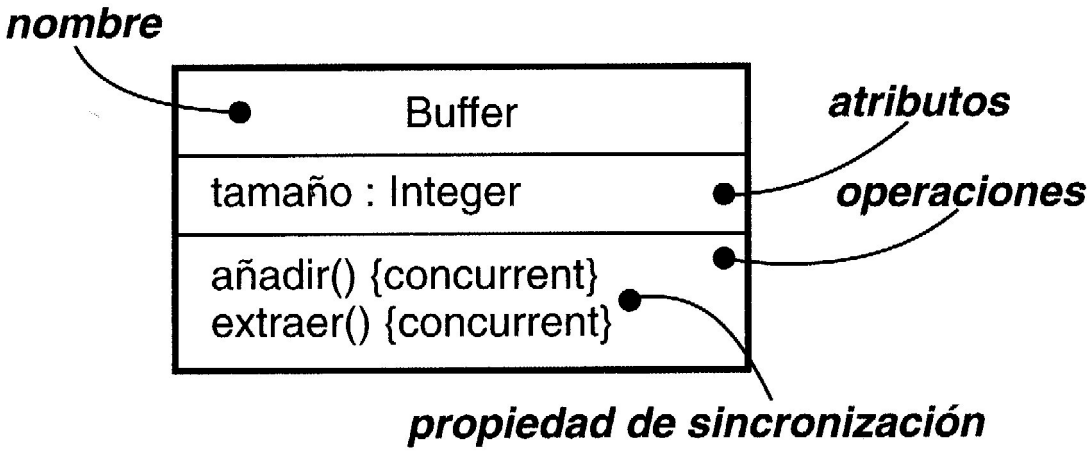
- Mensaje síncrono (flecha completa)
- Mensaje asíncrono (media flecha)



Diagramas de Colaboración (IV)
Modelado de procesos e hilos
Sincronización

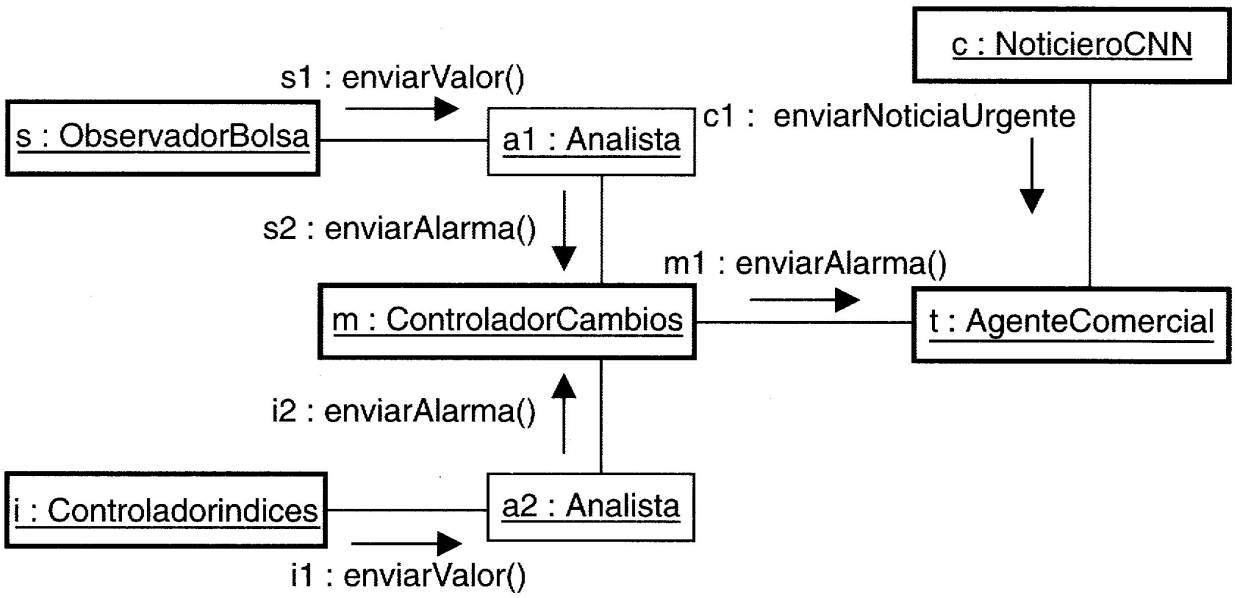
[Booch 99, capítulo 22]

- En UML se pueden modelar los tres enfoques de sincronización
  - Secuencial
  - Con guardas
  - Concurrentemente
- Se utiliza la notación de restricciones de UML asociar propiedades a cada operación



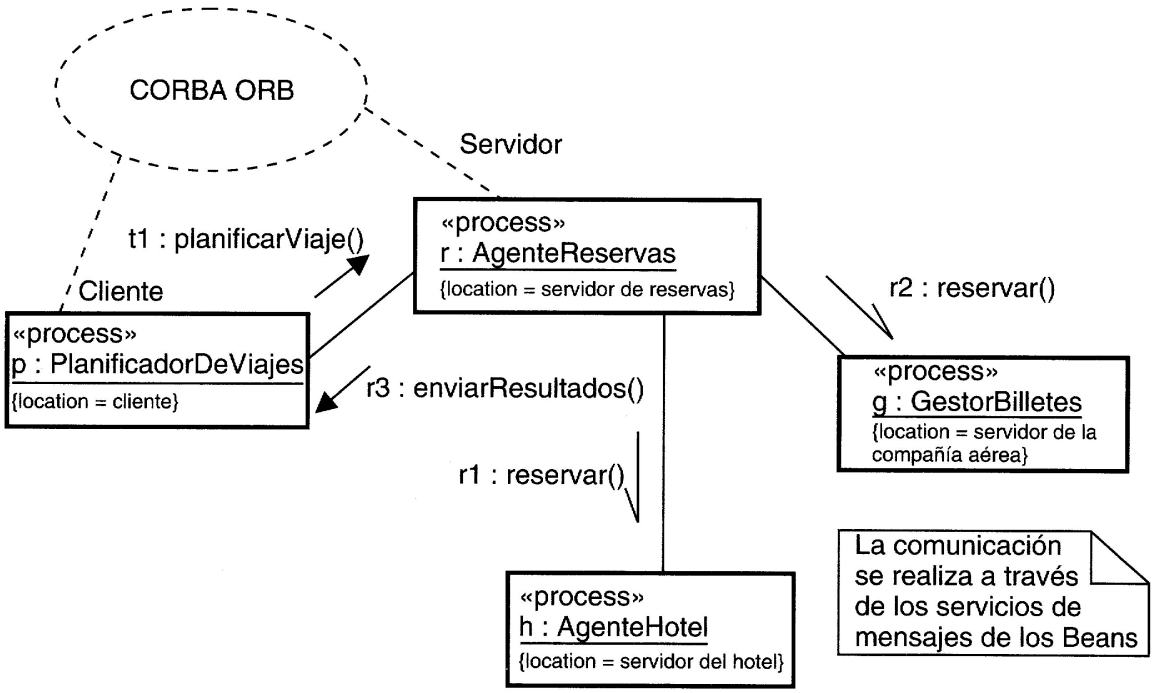
Diagramas de Colaboración (V)
Modelado de procesos e hilos
Modelado de flujos de control múltiples

[Booch 99, capítulo 22]



Diagramas de Colaboración (VI)
Modelado de procesos e hilos
Modelado de comunicación entre procesos

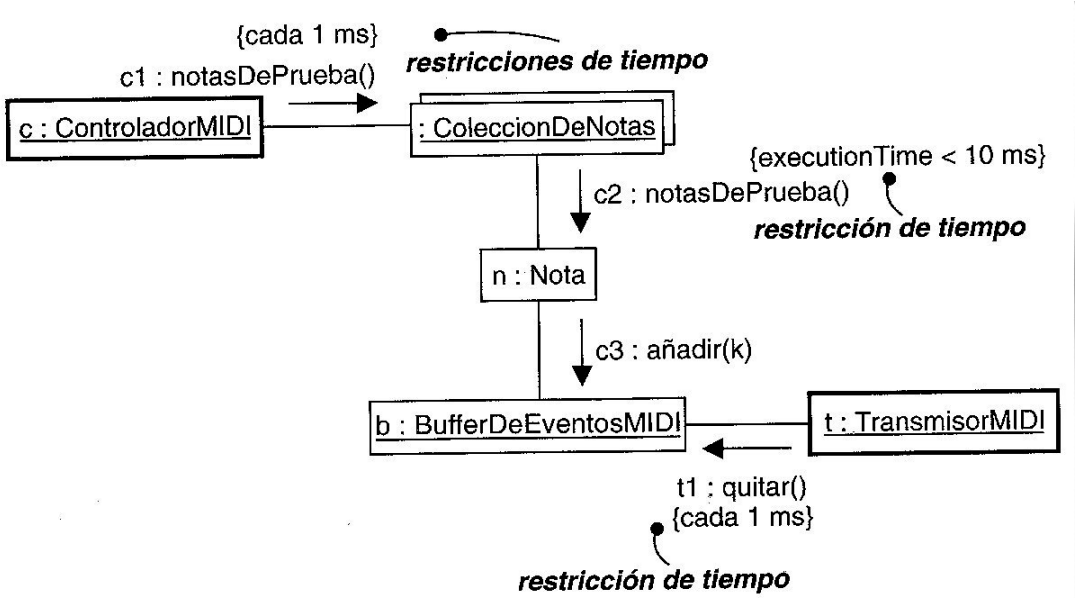
[Booch 99, capítulo 22]



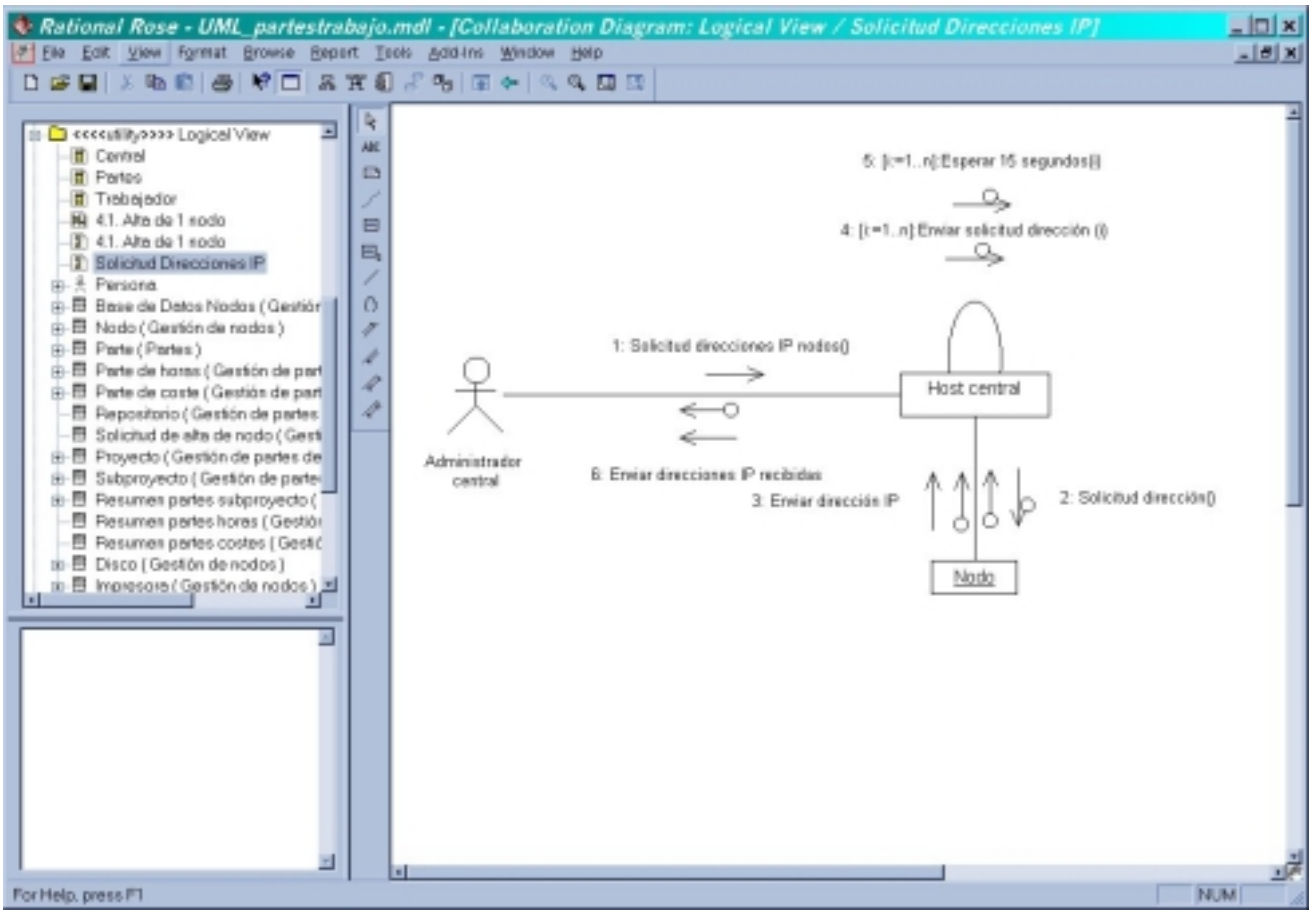
Diagramas de Colaboración (VII)

Restricciones de tiempo

[Booch 99, capítulo 23]



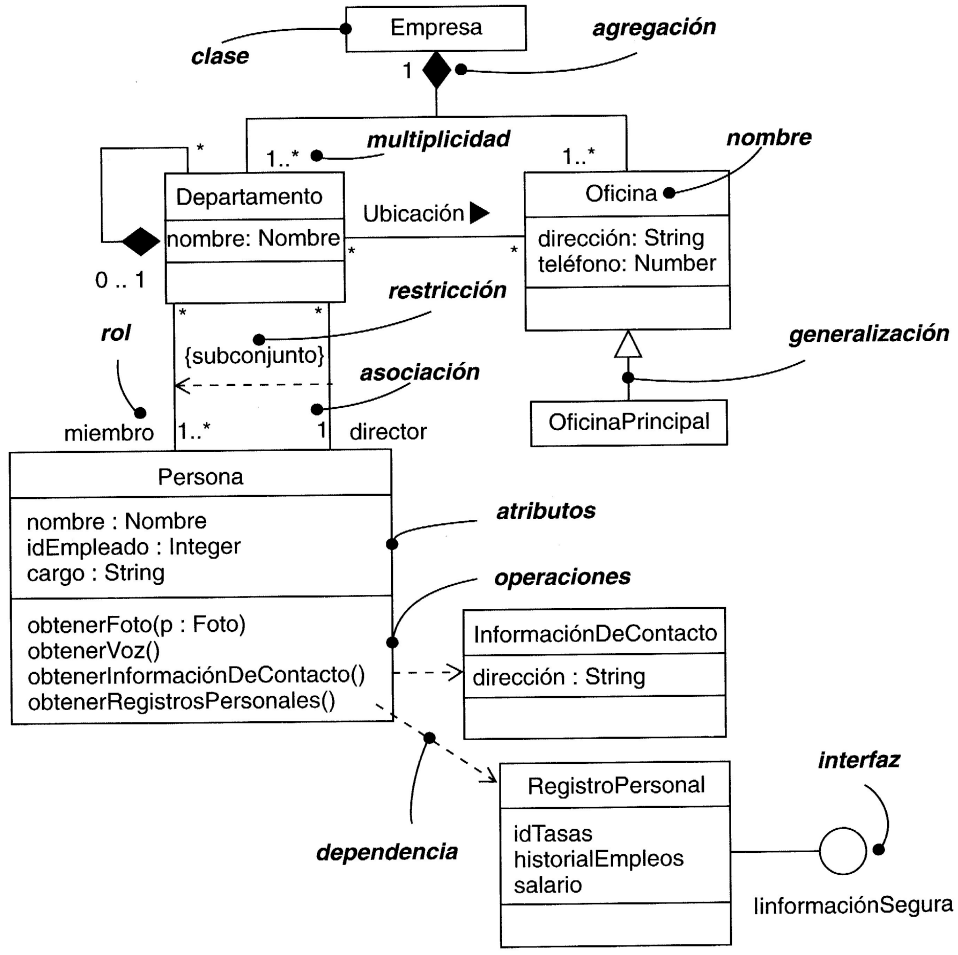
Diagramas de Colaboración (VIII)
en Rational Rose®



Diagramas de Clases (I)

[Booch 99, capítulos 8,9]

- Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema
- Los diagramas de clases contienen los siguientes elementos
  - Clases
  - Interfaces
  - Colaboraciones
  - Relaciones de dependencia, generalización y asociación



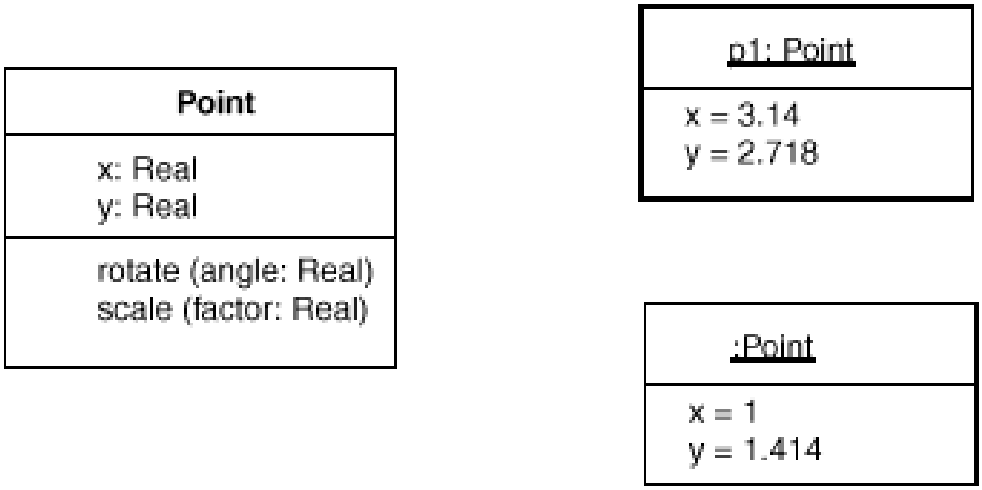
Diagramas de Clases (II)

- Estos diagramas son los más importantes del diseño orientado a objetos, son la *piedra angular* de nuestro diseño.
- Contienen toda la información de todas las clases y sus relaciones con otras clases
- Aunque son los más importantes no se llega a ellos directamente dado que tienen un gran nivel de abstracción dado que contemplan el modelo globalmente sin particularizarse en ningún escenario concreto
- Cuando se construyen los diagramas anteriores (Casos de uso, Secuencia, Colaboración) las herramientas van obteniendo nombres de clase y generando los atributos y operaciones de cada clase siguiendo las indicaciones dadas por las especificaciones de requisitos en los casos de uso y escenarios
- En el momento de hacer el primer diagrama de clases ya se tiene una lista de clases con algunos de sus atributos y operaciones. Sin embargo es necesario reflexionar y abstraer sobre la organización de esas clases estudiando las relaciones de herencia, agregación, etc.
- El diagrama de clases se refinará en las sucesivas iteraciones del modelo

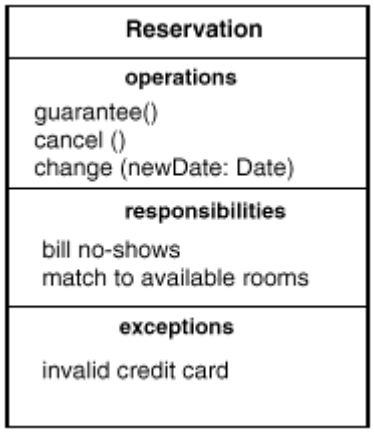
Clases y objetos en UML

Objetos

Clase

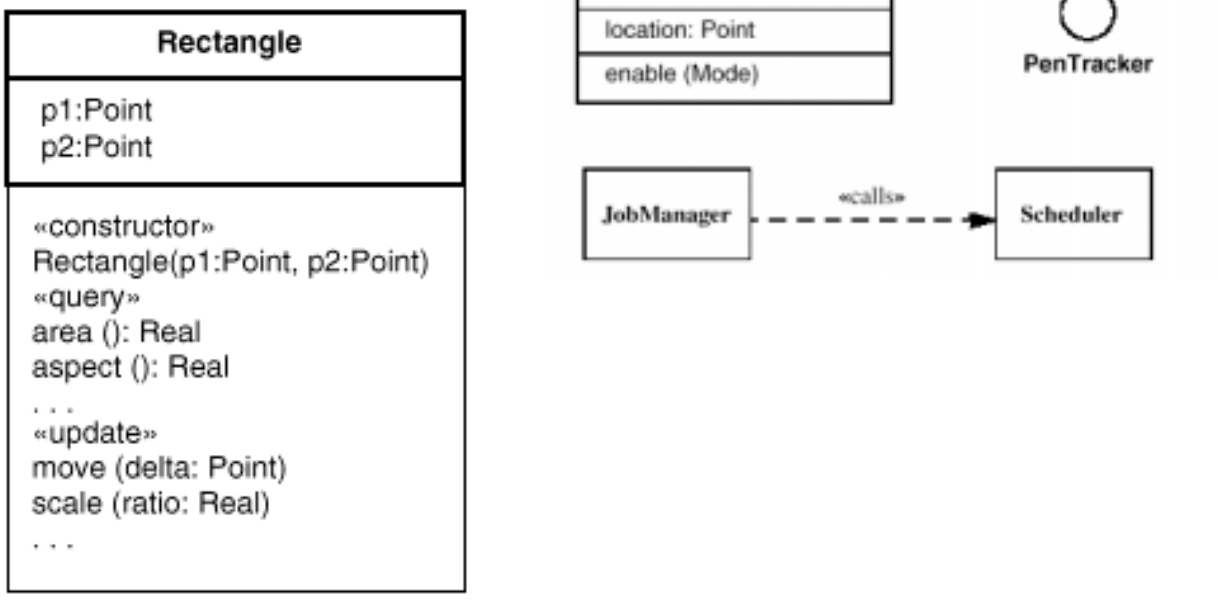


Los compartimentos de una clase pueden tener nombres



Estereotipos en UML

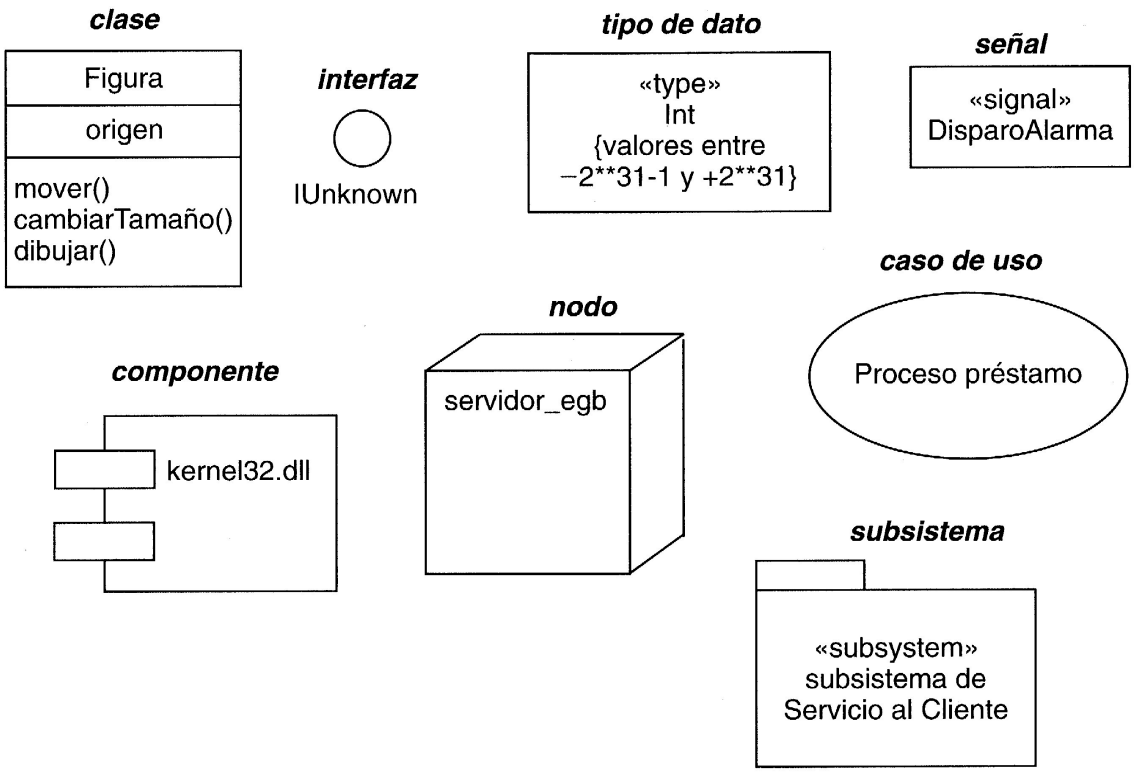
• Un estereotipo es una forma de clasificar las clases a alto nivel
• Los estereotipos se muestran con un texto entre doble ángulo << y >>, por ejemplo: <<*control*>>
• Los estereotipos también se pueden definir con un icono
• Muchas extensiones del núcleo de UML se pueden describir mediante una colección de estereotipos



Clasificadores (I)

[Booch 99, capítulo 9]

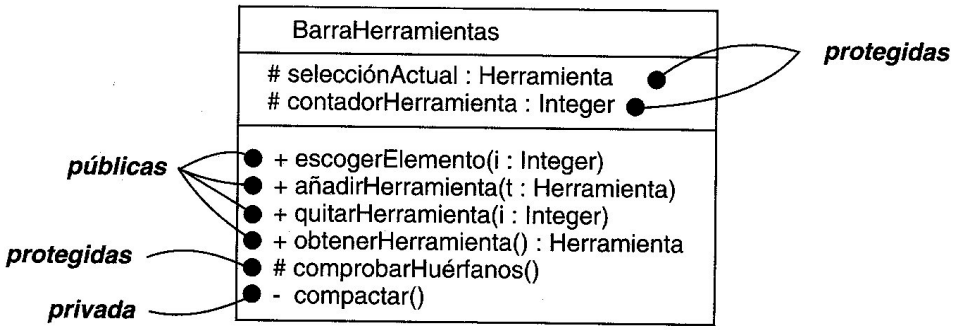
- Un clasificador es un mecanismo que describe características estructurales y de comportamiento
- Tipos de clasificadores:
  - Clase
  - Interfaz
  - Tipo de dato
  - Señal
  - Componente
  - Nodo
  - Caso de Uso
  - Subsistema



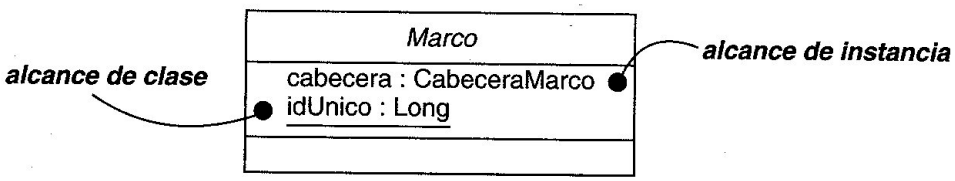
Clasificadores (II)

[Booch 99, capítulo 9]

- Visibilidad de atributos y operaciones de un clasificador
  - `public`
  - `protected`
  - `private`



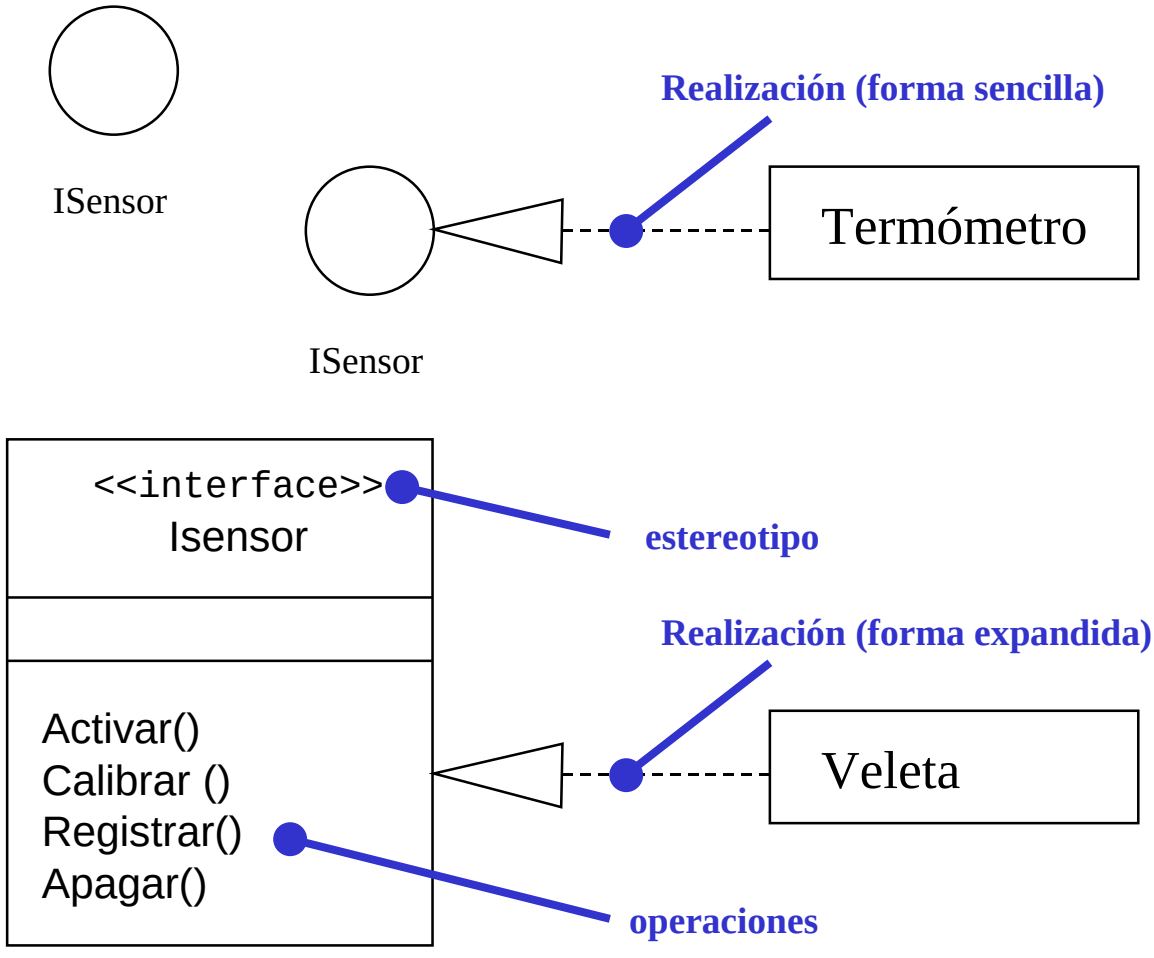
- Alcance de atributos y operaciones de un clasificador
  - `instancia`. Cada instancia del clasificador tiene su propio valor para la característica
  - `clasificador`. Sólo hay un valor de la característica para todas las instancias (static en C++ y Java)



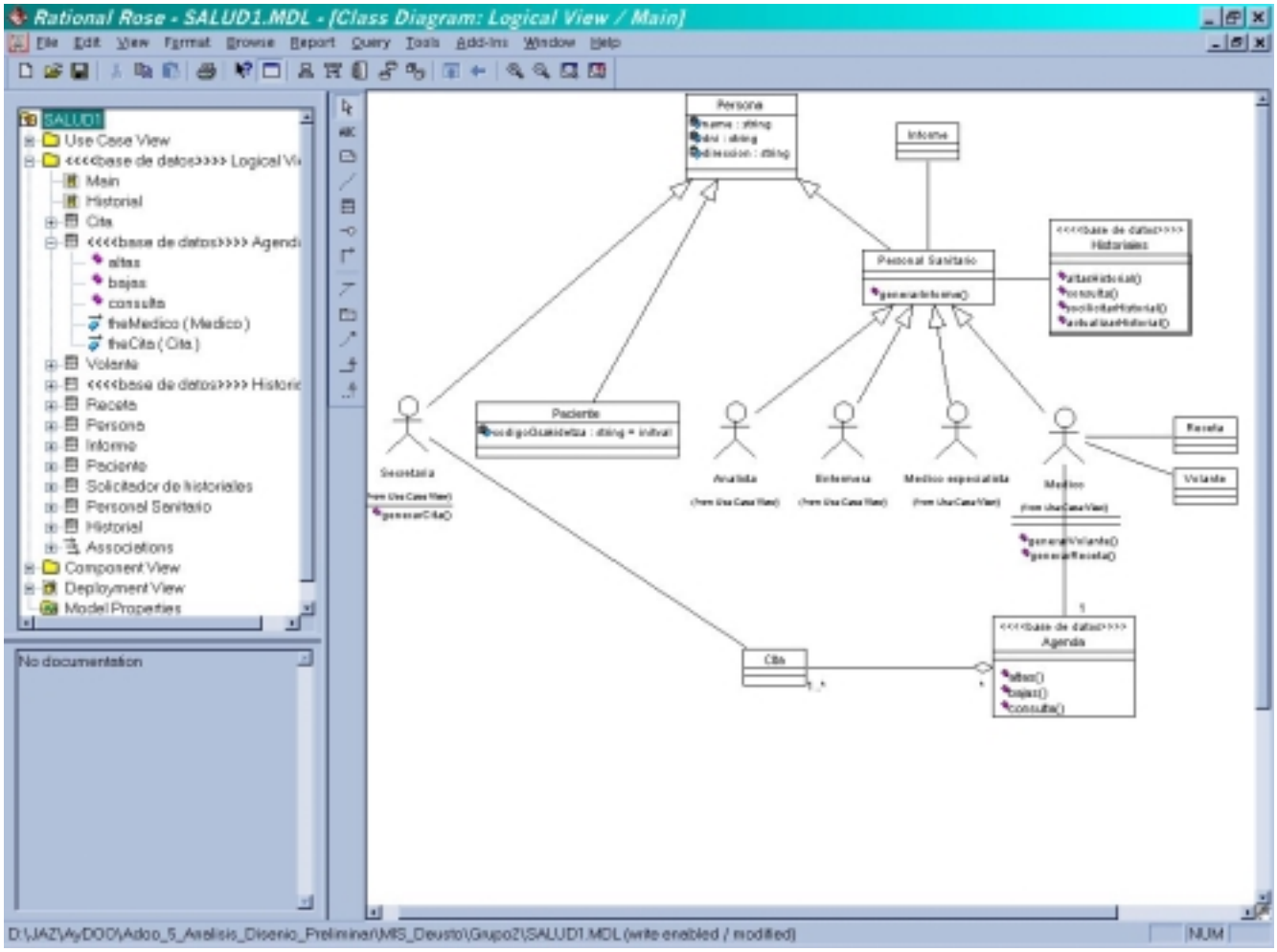
Interfaces

[Booch 99, capítulo 11]

- Una interfaz es una colección de operaciones que se usa para especificar un servicio de una clase o de un componente
- Una interfaz no tiene atributos
- Gráficamente una interfaz se representa por un círculo
- De forma expandida se puede ver como una clase con el estereotipo `<<interface>>`



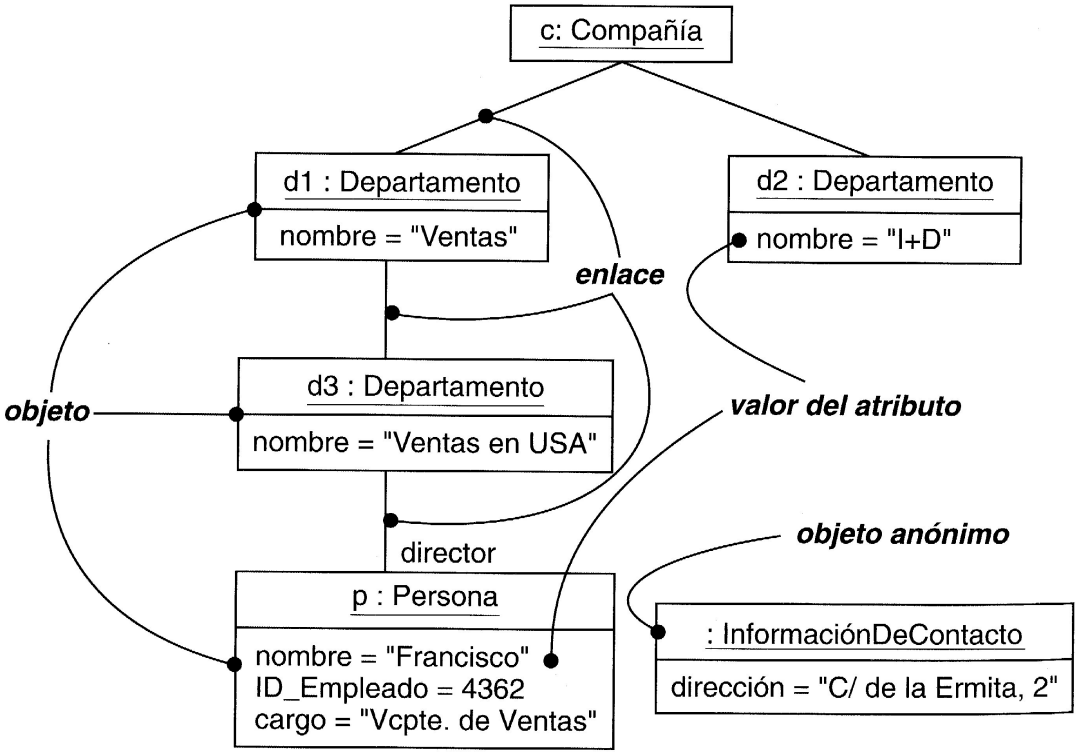
Diagramas de Clases en Rational Rose®



Diagramas de objetos

[Booch 99, capítulo 14]

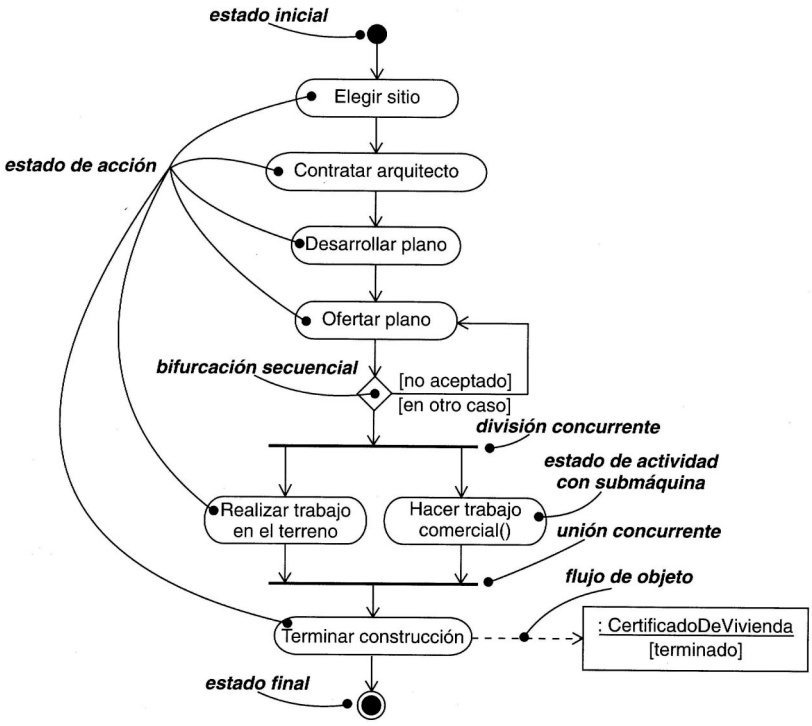
- Muestra un conjunto de objetos y sus relaciones
- Los diagramas de objetos representan instantáneas de instancias de los elementos encontrados en los diagramas de clases
- Estos diagramas cubren la vista de diseño estática o la vista de procesos estática de un sistema como lo hacen los diagramas de clases, pero desde la perspectiva de casos reales o prototípicos
- Muestran una especie de fotograma de un instante en tiempo de ejecución.
- Se realizan para mostrar momentos críticos del sistema o para aclarar detalles que pueden quedar confusos.
- Evidentemente no se hacen para todos los instantes de una ejecución



Diagramas de actividades

[Booch 99, capítulo 19]

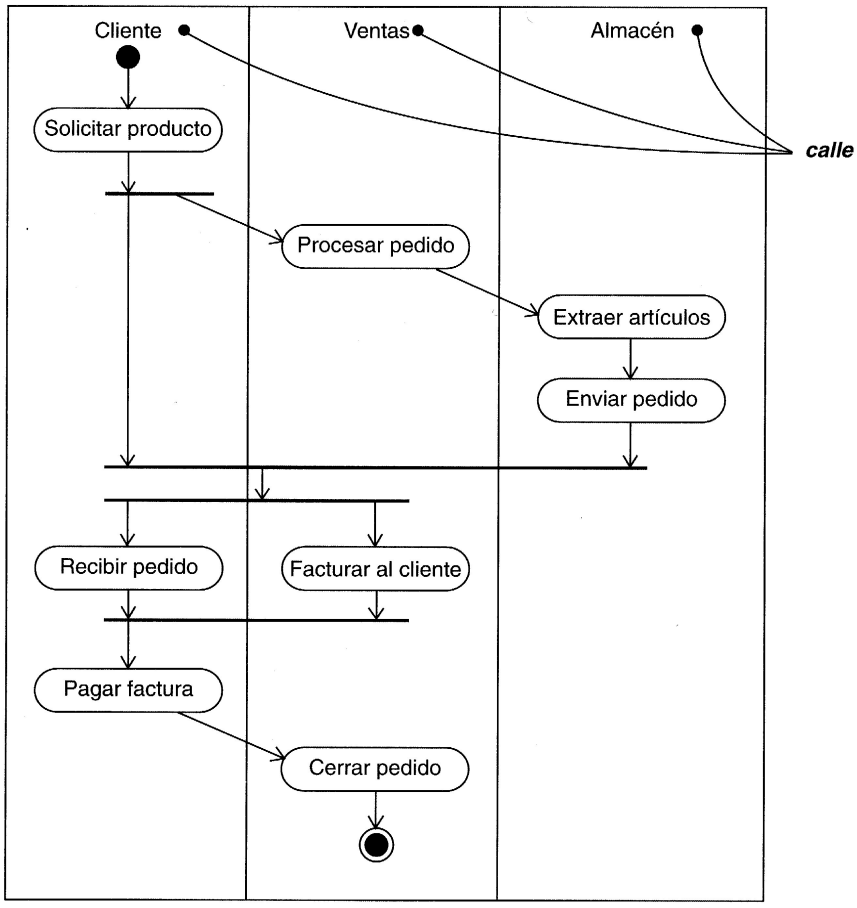
- Los diagramas de actividades son uno de los cinco diagramas que modelan aspectos dinámicos del sistema
- Un diagrama de actividades muestra el flujo de actividades
- Una actividad es una ejecución no atómica en curso dentro de una máquina de estados
- Las actividades producen finalmente una acción, que está compuesta de computaciones atómicas ejecutables que producen un cambio en el estado del sistema o la devolución de un valor.
- Un diagrama de actividad contiene:
  - Estados de actividad y estados de acción
  - Transiciones
  - Objetos
  - Restricciones



Diagramas de actividades
Calles (Swimlanes)

[Booch 99, capítulo 19]

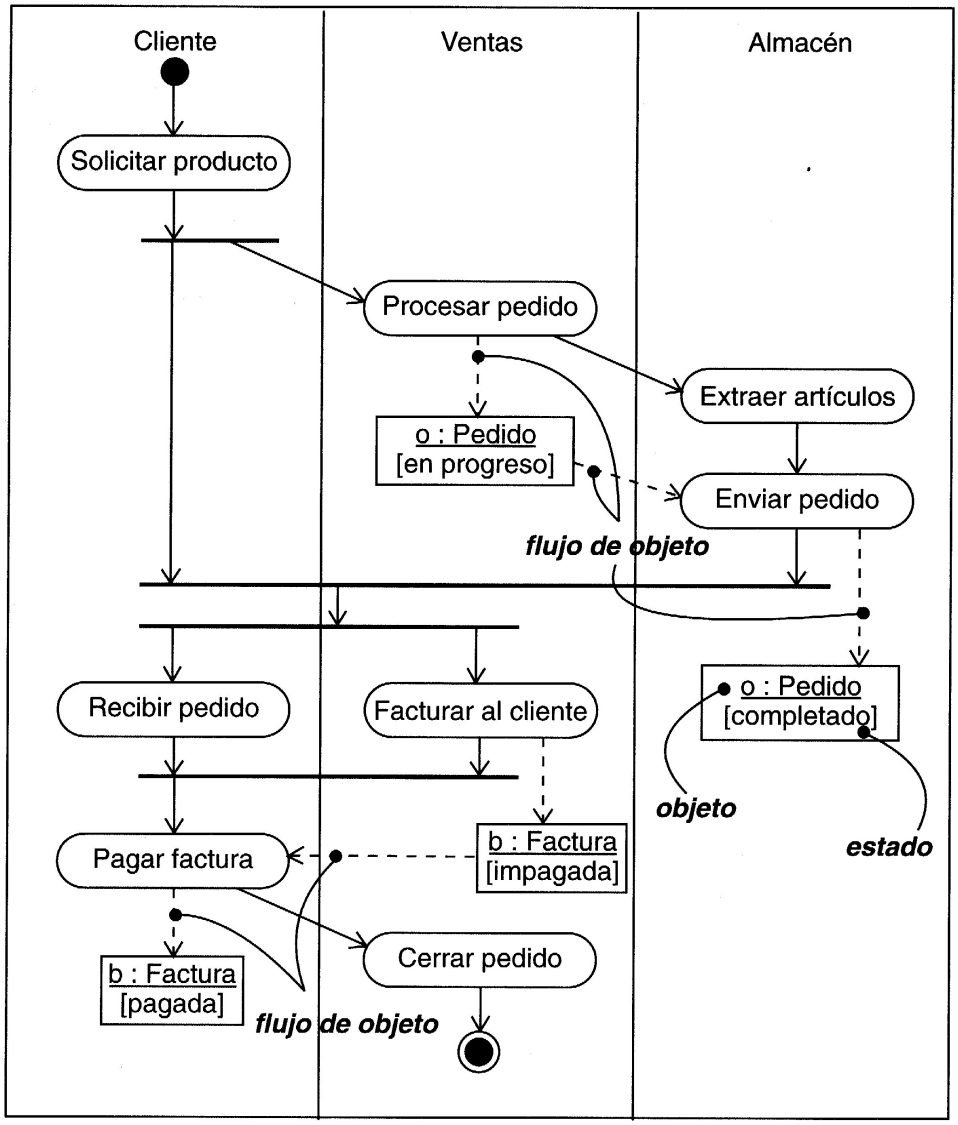
- Permiten modelar flujos de trabajo de procesos de organizaciones separándolos en grupos denominados **calles** (*swinlanes*)
- Cada grupo representa la parte de la organización responsable de esas actividades



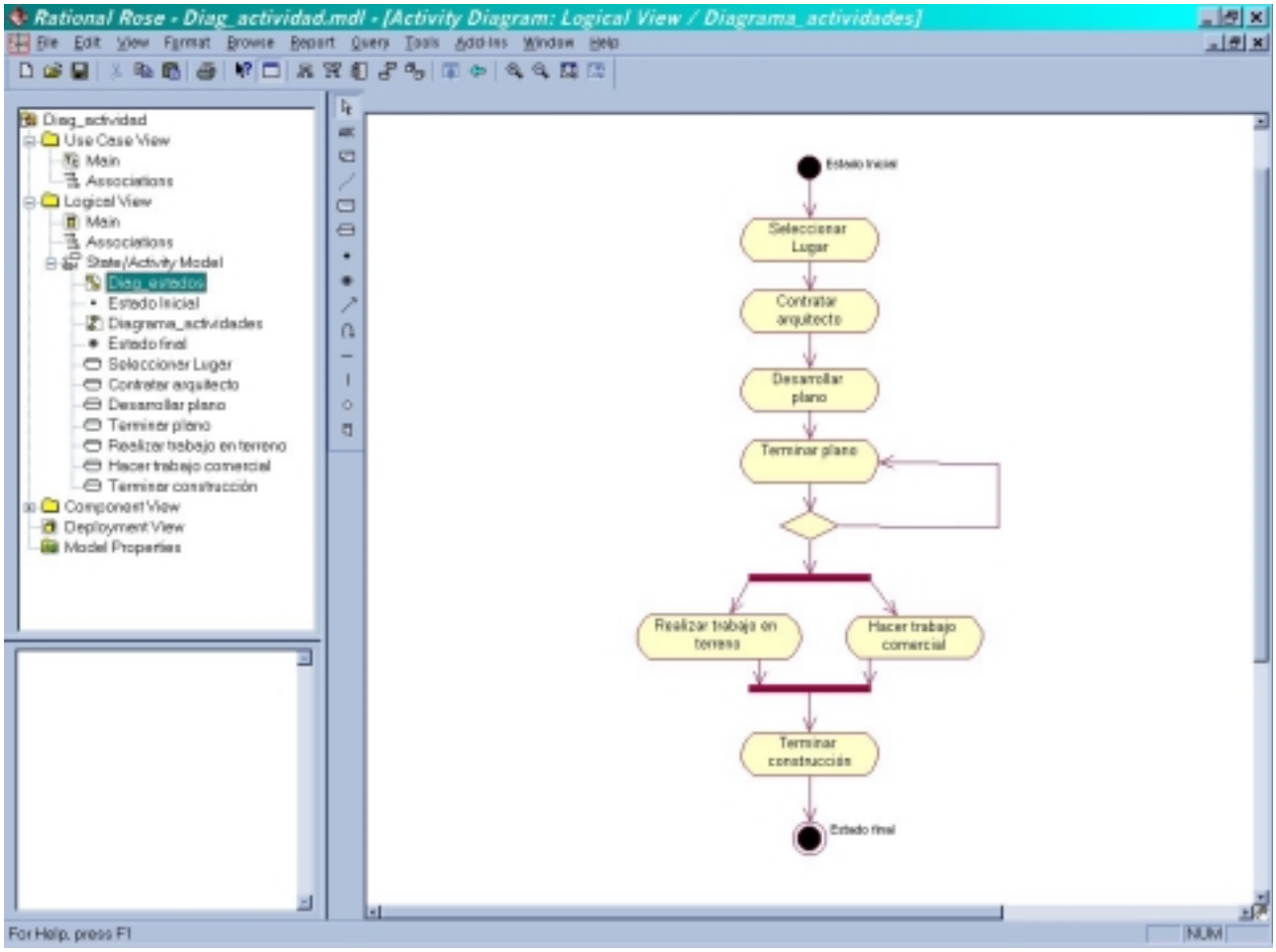
Diagramas de actividades
Flujo de objetos

[Booch 99, capítulo 19]

- Los diagramas de actividad pueden tener un flujo de control con objetos
- Pueden mostrarse también como cambian los valores de los atributos de los objetos



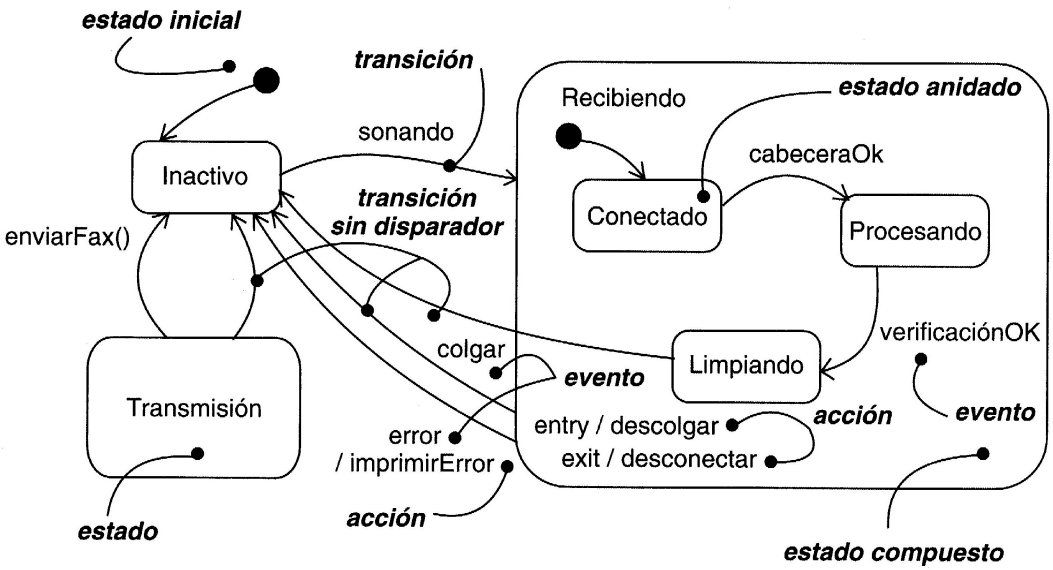
Diagramas de Actividades en Rational Rose®



Diagramas de estados

[Booch 99, capítulo 24]

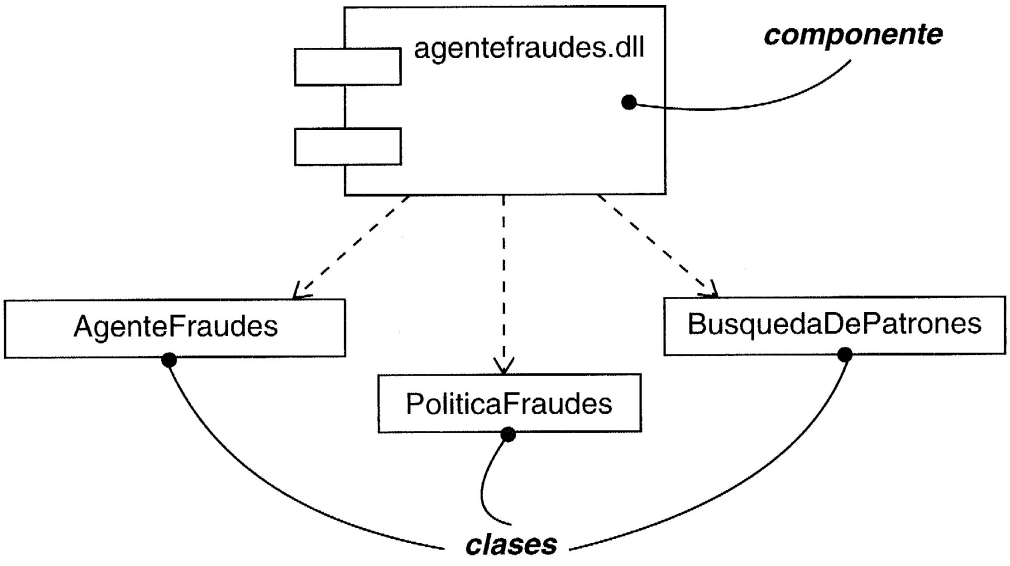
- Se utilizan para modelar aspectos dinámicos de un sistema
- Un diagrama de estados muestra una máquina de estados.



Diagramas de componentes

[Booch 99, capítulo 25]

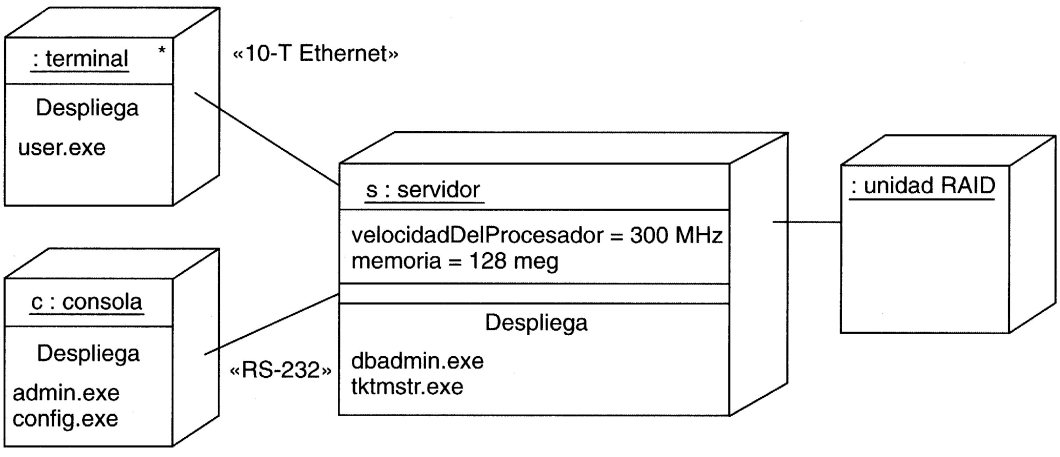
- Los componentes se utilizan para modelar los elementos físicos que pueden hallarse en un nodo, tales como ejecutables, bibliotecas, tablas, archivos y documentos
- Un componente es una parte física y reemplazable de un sistema que conforma con un conjunto de interfaces y proporciona la realización de esas interfaces



Diagramas de despliegue

[Booch 99, capítulo 26]

- Representan la topología del hardware
- Un nodo es un elemento físico que existe en tiempo de ejecución y que representa un recurso computacional



Referencias

- [Booch 94] G.Booch. *Object-oriented analysis and design with applications.* Benjamin Cummings (1994). Versión castellana: *Análisis y diseño orientado a objetos con aplicaciones.* 2ª Edición. Addison-Wesley/ Díaz de Santos (1996).
- [Booch 99] G. Booch, J. Rumbaugh, I. Jacobson. *The unified modeling language user guide.* Addison-Wesley (1999). Versión castellana *El lenguaje unificado de modelado.* Addison-Wesley (1999)
- [Coad 97] P. Coad, M. Mayfield. *Java design*. Prentice-Hall, 1997.
- [Cook 94] S. Cook and J. Daniels, *Designing Object Systems: Object-oriented Modelling with Syntropy*, Prentice-Hall Object-Oriented Series, 1994.
- [Eriksson 98] H-E Eriksson & M. Penker. *UML Toolkit. Wiley, 1998.*
- [Fowler 97] M. Fowler with K. Scott, *UML Distilled: Applying the Standard Object Modeling Language*, ISBN 0-201-32563-2, Addison-Wesely, 1997.
- [Gamma 95] Gamma E. et al. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
- [Granham 97] I. Graham, B. Henderson-Sellers, H. Younessi. *The OPEN Process Specification*. Addison-Wesley (1997).
- [Grand 98] Grand M. *Patterns in Java. Volume 1*. Wiley, 1998.
- [Grand 99] Grand M. *Patterns in Java. Volume 2*. Wiley, 1999.
- [Henderson-Sellers, 97] B. Henderson-Sellers. *OML: OPEN Modelling Language*. SIGBOOKS, 1997.
- [Jacobson 92] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard. *Object-Oriented software Engineering. A use case driven Approach.* Addison-Wesley (1992)
- [Jacobson 99] I. Jacobson, G. Booch, J. Rumbaugh. *The unified software development process*. Addison-Wesley (1999).
- [Larman 98] C. Larman. *Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design*. Prentice-Hall (1998). Versión castellana: *UML y patrones. Introducción al análisis y diseño orientado a objetos*. Prentice-Hall (1999).
- [Lee 97] R. C.Lee & W. M. Tepfenhart. *UML and C++, Prentice-Hall, 1997*
- [López 97] N. López, J. Migueis, E. Pichon. *Intégrer UML dans vos projects.* Editions Eyrolles, 1998. Versión castellana: *Integrar UML en los proyectos. Ediciones Gestión 2000 (1998).*
- [Muller 97] Muller P-A *Modélisation Object avec UML* Eyrolles 1997. Versión castellana: *Modelado de objetos con UML, Gestión-2000, 1997*
- [Odell 98] J.J. Odell. *Advanced Object-Oriented Analysis & Design Using UML*. SIGS, 1998.
- [OMG] *www.omg.org*
- [OPEN] *www.open.org.au*
- [Piattini 96] M.G. Piattini, J.A. Calvo-Manzano, J. Cervera, L. Fernández. *Análisis y diseño detallado de aplicaciones de gestión*. RA-MA (1996)
- [Rational] UML y herramienta Rational Rose en *www.rational.com*
- [Rumbaught 91] Rumbaught J., Blaha M., Premerlani W., Wddy F., Lorensen W. *Object-oriented modeling and design.* Prentice-Hall (1991). Versión castellana: *Modelado y diseño orientado a objetos. Metodología OMT*. Prentice-Hall (1996)
- [Rumbaught 99] Rumbaught J., I. Jacobson, G. Booch. *The Unified Modeling Language Reference Manual.* Addison-Wesley (1999)
- [Reenskaug 96] T. Reenskaug. *Working with Objects. The Ooram Software Engineering Method.*Prentice-Hall, 1996
- [Wilkinson 95] N. M. Wilkinson. *Using CRC Cards. An Informal Approach to Object-Oriented Development*, 1995, SIGS BOOKS, 1-884842-07-0