NOTA:
Esta presentación está basada en el tutorial de UML de OMG
www.celigent.com/omg/umlrtf/tutorials.htm

UML

Ingeniería del Software 2
Facultad de Informática

Juan Pavón Mestras
Dep. Sistemas Informáticos y Programación
Universidad Complutense Madrid

http://www.fdi.ucm.es/profesor/jpavon/is2

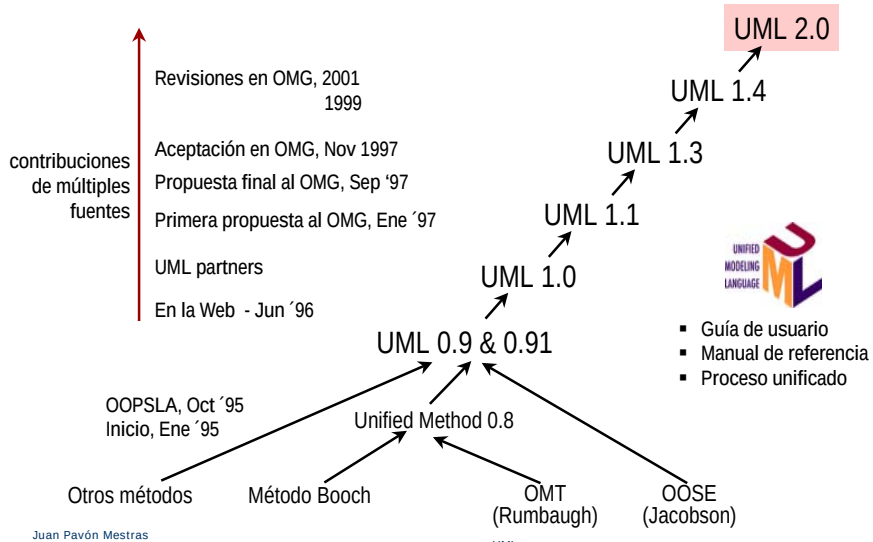
¿Qué es UML?

- *Unified Modelling Language*
  - Lenguaje *gráfico* para modelado de sistemas
    - especificar, visualizar, construir, documentar
  - Estándar abierto (OMG)
  - Soporta todo el ciclo de vida de desarrollo de software
    - Especificaciones de análisis, arquitectura, diseño, implementación e implantación
  - Soporta distintas áreas de aplicación
    - Sistemas distribuidos, tiempo real, aplicaciones monoproceso
    - Sistemas de información corporativos (MIS), Banca/Finanzas, Telecomunicaciones, Defensa/Espacio, Transporte, Distribución, Electromedicina, Ciencia, etc.
  - Soportado por herramientas
    - *Rational Rose, Together, Objecteering, Paradigm Plus, ...*

Objetivos de UML

- Definir un lenguaje de modelado visual fácil de aprender pero rico en significado
- Estándar, estable y configurable
  - Unificar las metodologías de análisis y diseño OO más conocidas (Booch, OMT, Objectory)
  - e incluir ideas de otros lenguajes de modelado
- Ser independiente de lenguajes de programación o procesos particulares
- Promover en el mercado el crecimiento de herramientas CASE OO con soporte a UML
- Soportar conceptos de desarrollo de alto nivel tales como colaboraciones, frameworks, patrones y componentes
- Tratar aspectos del desarrollo de software actual
  - escalabilidad, concurrencia, distribución, ejecutabilidad, etc.

Evolución de UML

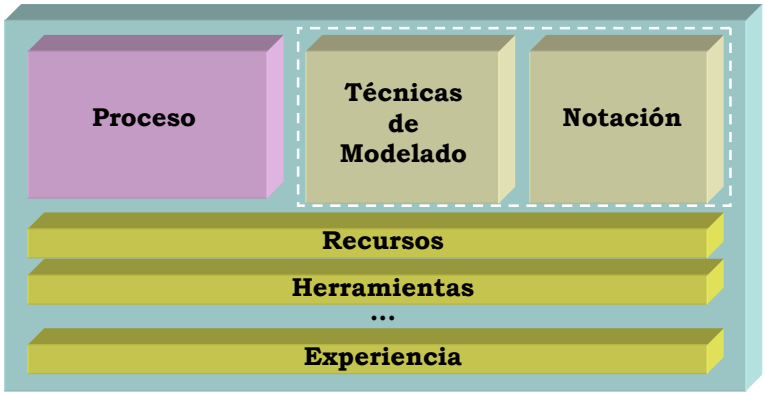


Contribuciones a la definición de UML (OMG)

Aonix	Microsoft
Colorado State University	ObjecTime
Computer Associates	Oracle
Concept Five	Ptech
Data Access	OAO Technology Solutions
EDS	Rational Software
Enea Data	Reich
Hewlett-Packard	SAP
IBM	Softeam
I-Logix	Sterling Software
InLine Software	Sun
Intellicorp	Taskon
Kabira Technologies	Telelogic
Klasse Objecten	Unisys
Lockheed Martin	…

Ámbito de UML

- Metodología de desarrollo de software OO



Modelar

- No se construye un edificio sin tener antes unos planos
- Permite estructurar la solución a un problema
  - abstrayendo para gestionar la complejidad
  - experimentando varias soluciones
  - reduciendo los costes
  - gestionando el riesgo de errores
- Ventajas de modelar:
  - Permite ver el sistema desde varias perspectivas haciendo más sencillo su entendimiento y desarrollo
  - Mejora la comunicación
    - con el cliente
    - del equipo de desarrollo

Notación gráfica

- *Graphics reveal data*

  Edward Tufte
  The Visual Display of Quantitative Information, 1983

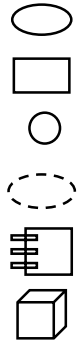
- 1 bitmap = 1 megaword

Elementos de UML

- Entidades
  - Estructurales
  - Comportamiento
  - Agrupamiento
- Relaciones
- Diagramas

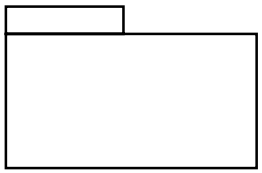
Entidades estructurales

- Casos de uso
- Clases
- Interfaces
- Colaboraciones
- Componentes
- Nodos



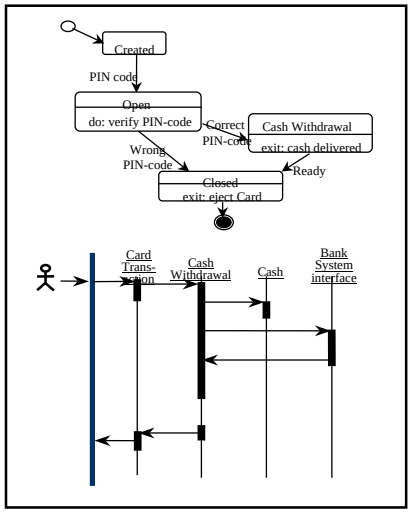
Entidades de agrupamiento

- Paquete
- Modelo
- Subsistema



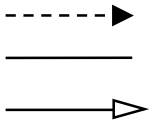
Entidades de comportamiento

- Máquinas de estado
- Interacciones



Relaciones

- Dependencia
- Asociación
- Generalización



Diagramas

- Casos de uso
- Clase
- Objetos
- Secuencia
- Colaboración
- Estados
- Actividades
- Componentes
- Implantación

Modelo de casos de uso

Modelo estructural

Modelo de comportamiento

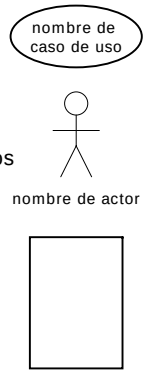
Modelo de implementación

Modelo de casos de uso

- Visión del sistema tal como se muestra a sus usuarios externos
  - Los desarrollan tanto los analistas *como los expertos del dominio*
- Particiona la funcionalidad del sistema en:
  - transacciones (*casos de uso*)
  - que tienen significado para los usuarios (*actores*)
- Se define mediante:
  - Diagramas de casos de uso
  - Descripción de los casos de uso
    - Mediante plantillas de texto
    - Acompañados de diagramas de interacción

Modelo de casos de uso

- Elementos de un diagrama de casos de uso
  - Caso de uso
    - Secuencia de acciones, incluyendo variantes, que puede realizar el sistema interaccionando con los actores del sistema
  - Actor
    - Un conjunto coherente de roles que juegan los usuarios cuando interaccionan con los casos de uso
  - Límite del sistema
    - Representa el límite entre el sistema físico y los actores que interaccionan con el sistema

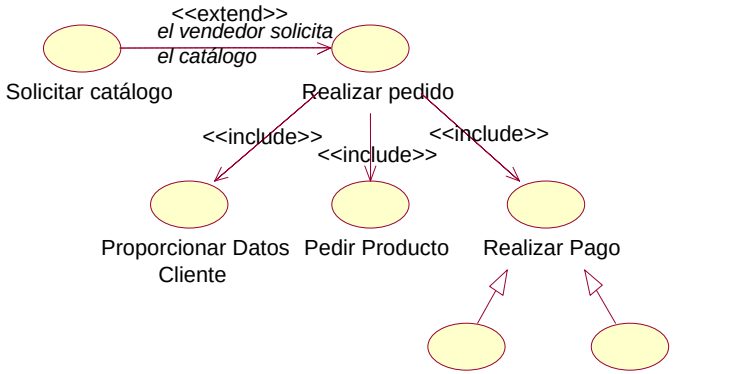


Modelo de casos de uso

- Relaciones en un diagrama de casos de uso
  - Asociación
    - La participación de un actor en un caso de uso
    - La instancia de un actor se comunican con instancias de un caso de uso
  - Generalización
    - Relación taxonómica entre un caso de uso más general y otro más específico
  - Dependencia
    - <<extend>> el comportamiento de un caso de uso extiende el de un caso de uso base
    - <<include>> el comportamiento de un caso de uso incluye el de un caso de uso base

Diagramas de casos de uso

- Asociaciones de dependencia y generalización



Adaptado de Fig. 3-54, *UML Notation Guide*

Diagrama de casos de uso

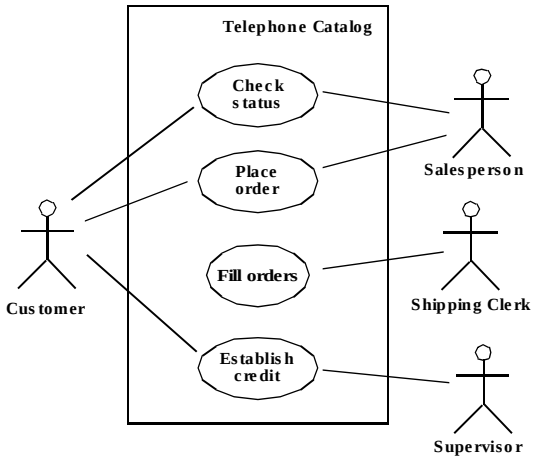


Fig. 3-53, *UML Notation Guide*

Descripción de casos de uso

- Se usa una plantilla:
  - Actores
  - Precondiciones
  - Postcondiciones
  - Secuencia normal
    - Acciones de los actores --- Respuesta del sistema
  - Alternativas
    - Distintos escenarios
    - Excepciones

Consejos para un buen modelo de casos de uso

- Asegurarse que cada caso de uso describe una parte significativa del funcionamiento del sistema
- Evitar un número excesivo de casos de uso
  - Un caso de uso no es un paso, operación o actividad individual en un proceso
  - Un caso de uso describe un proceso completo que incluye varios pasos
- Los casos de uso tienen que ser entendibles tanto por desarrolladores software como por expertos del dominio
  - Es una descripción de alto nivel del sistema
  - Evitar conceptos de diseño

Consejos para un buen modelo de casos de uso

- Cómo identificar casos de uso:
  - A partir de los actores
    - Identificar los actores relacionados con el sistema o la organización
    - Para cada actor, identificar los procesos que inician o en los que participan
  - A partir de los eventos
    - Identificar los eventos externos a los que puede responder el sistema
    - Relacionar los eventos con actores y casos de uso

Cuándo definir un modelo de casos de uso

- Para identificar los requisitos funcionales del sistema
  - y para identificar escenarios de prueba
- Normalmente en las primeras etapas de desarrollo
  - En metodologías dirigidas por casos de uso:
    - Empezar por los casos de uso y derivar de ellos los modelos estructural y de comportamiento
  - y si no:
    - Asegurarse que el modelo de casos de uso es consistente con los modelos estructural y de comportamiento

Ejercicio de modelo de casos de uso

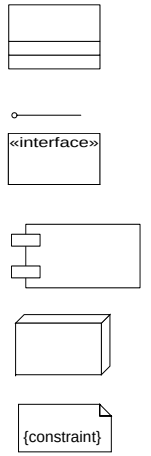
- Definir el modelo de casos de uso para un sitio web de reservas de hotel
  - Identificar actores
  - Diagrama de casos de uso
  - Descripción de casos de uso

Modelo estructural

- Visión del sistema que describe la estructura de los objetos, incluyendo su clasificación, relaciones, atributos y operaciones
  - Desarrollado por analistas, diseñadores y programadores
- Muestra la estructura estática del sistema
  - Las entidades que existen (clases, interfaces, componentes, nodos, etc.)
    - Captura el vocabulario del sistema
  - La estructura interna
  - La relación con otras entidades
- Se define mediante:
  - Diagramas estructurales estáticos
    - Diagrama de clases
    - Diagrama de objetos
  - Diagramas de implementación
    - Diagrama de componentes
    - Diagrama de implantación

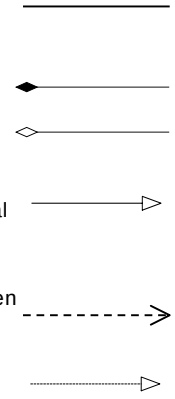
Modelo estructural

- Elementos de un diagrama estructural
  - Clase
    - Descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica
  - Interfaz
    - Un conjunto nombrado de operaciones que caracterizan el comportamiento de un elemento
  - Componente
    - Una parte modular, reemplazable y significativa del sistema que empaqueta implementación y expone interfaces
  - Nodo
    - Objeto físico de ejecución que representa un recurso computacional
  - Restricción
    - Condición semántica o restricción



Modelo estructural

- Relaciones en un diagrama estructural
  - Asociación
    - Relación entre dos o más clases que implica conexiones entre sus instancias
  - Agregación
    - Forma especial de asociación que especifica una relación todo-parte entre el agregado y la parte componente
  - Generalización
    - Relación taxonómica entre un elemento más general y otro más concreto
  - Dependencia
    - Relación entre dos elementos en la que un cambio en uno de ellos afecta al otro (elemento dependiente)
  - Realización
    - Relación entre una especificación y su implementación



Diagramas estructurales estáticos

- Muestra un grafo de elementos clasificados conectados por relaciones estáticas
- Dos tipos
  - Diagrama de clases: visión de clasificación
  - Diagrama de objetos: visión de instanciación

Clases

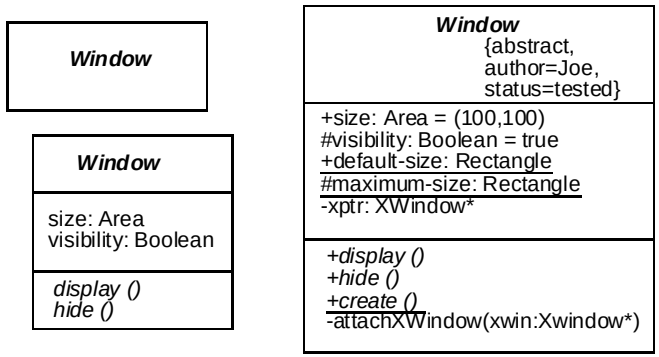


Fig. 3-20, *UML Notation Guide*

Clases: compartimentos con nombres

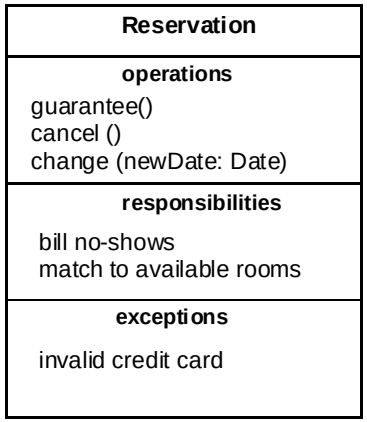


Fig. 3-23, *UML Notation Guide*

Clases: métodos

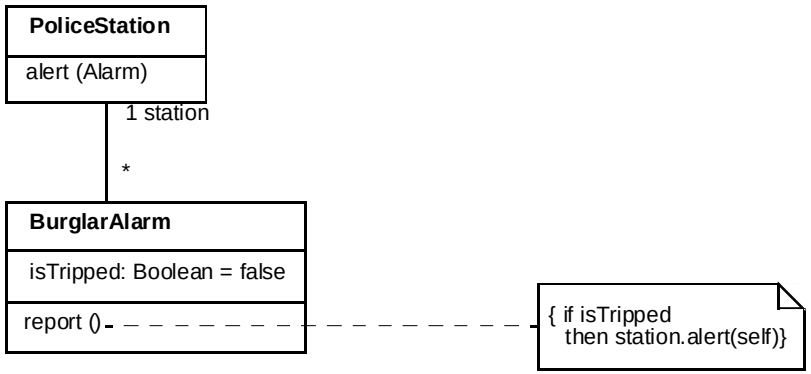


Fig. 3-24, *UML Notation Guide*

Tipos e implementación de clases

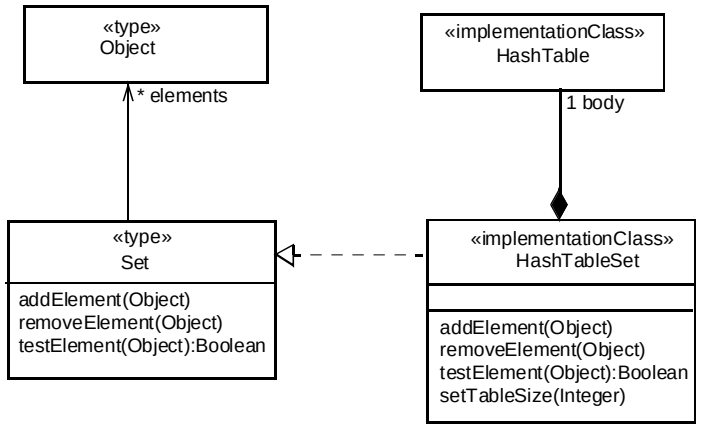


Fig. 3-27, *UML Notation Guide*

Interfaces

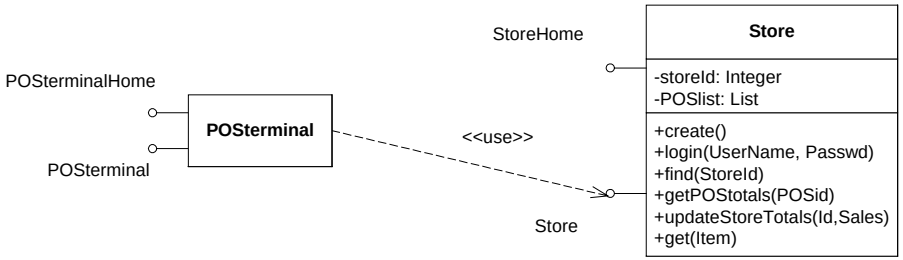


Fig. 3-29, *UML Notation Guide*

Interfaces: notación con estereotipo

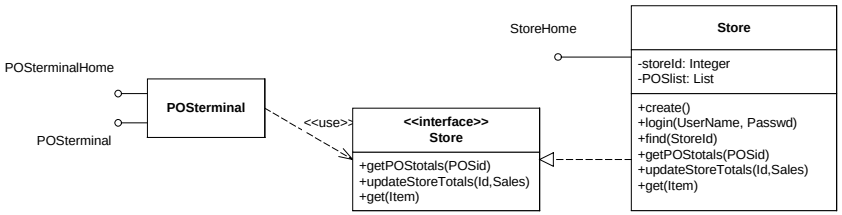


Fig. 3-29, *UML Notation Guide*

Asociaciones

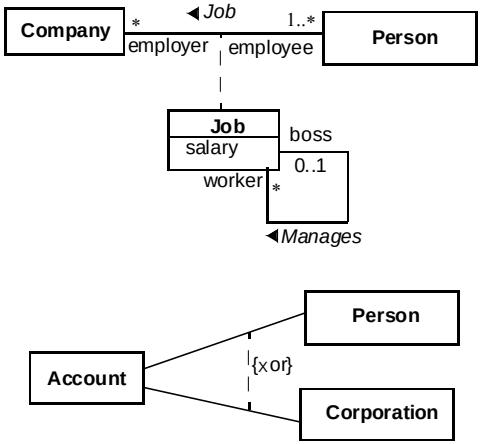


Fig. 3-40, *UML Notation Guide*

Extremos de asociaciones

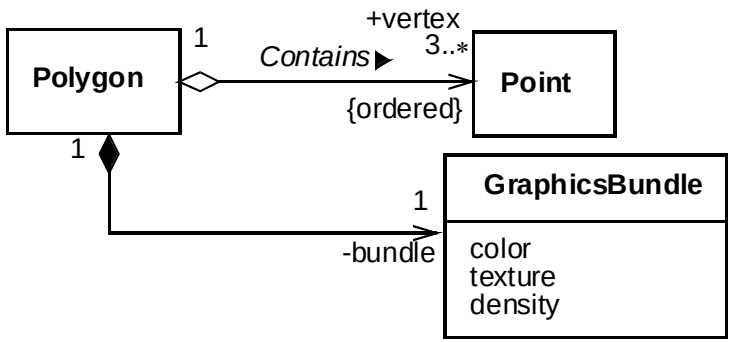


Fig. 3-41, *UML Notation Guide*

Asociaciones ternarias

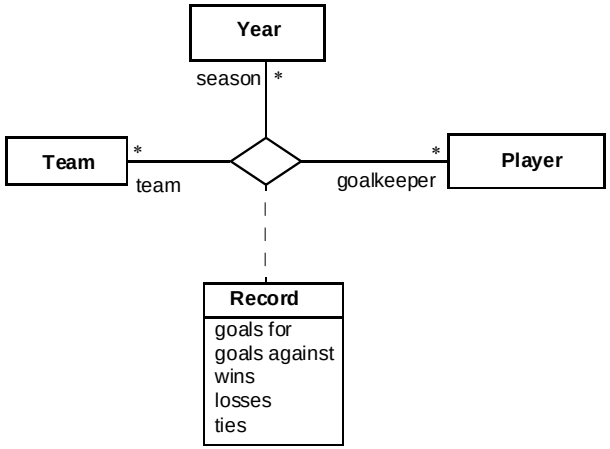


Fig. 3-44, *UML Notation Guide*

Composición

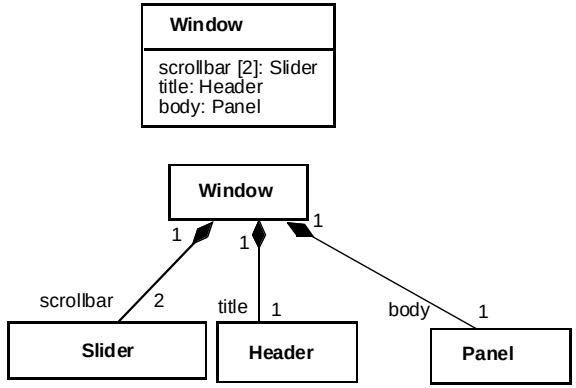


Fig. 3-45, *UML Notation Guide*

Composición

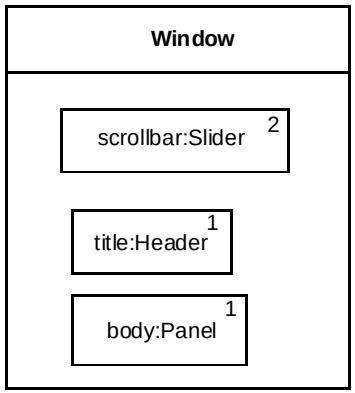


Fig. 3-45, *UML Notation Guide*

Generalización

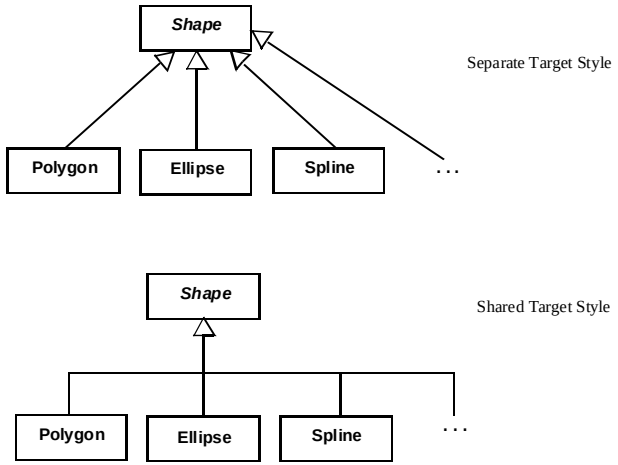


Fig. 3-47, *UML Notation Guide*

Generalización

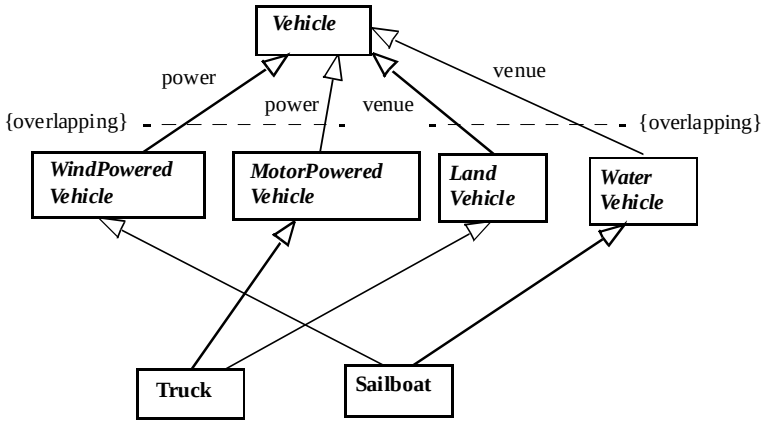


Fig. 3-48, *UML Notation Guide*

Dependencias

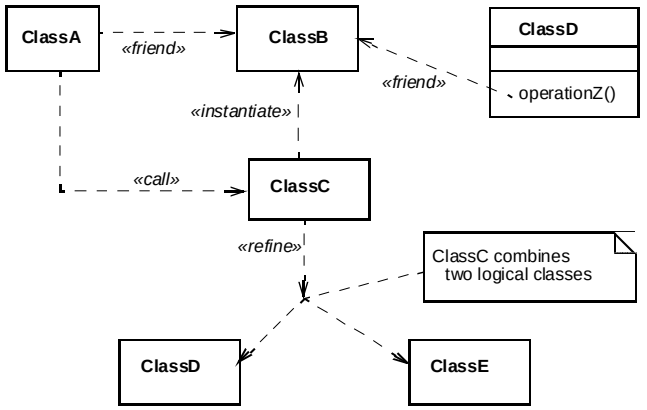


Fig. 3-50, *UML Notation Guide*

Dependencias

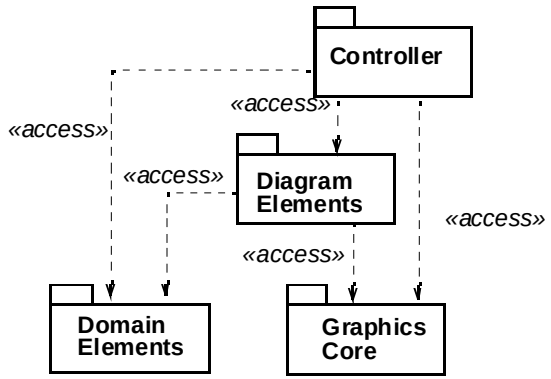


Fig. 3-51, *UML Notation Guide*

Atributos derivados y asociaciones

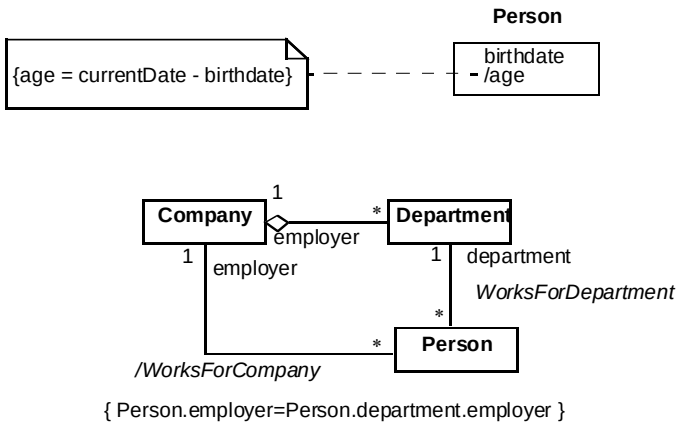


Fig. 3-52, *UML Notation Guide*

Objetos

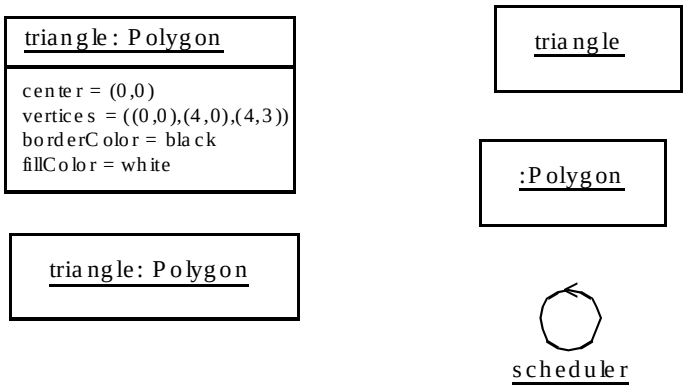


Fig. 3-38, *UML Notation Guide*

Objetos compuestos

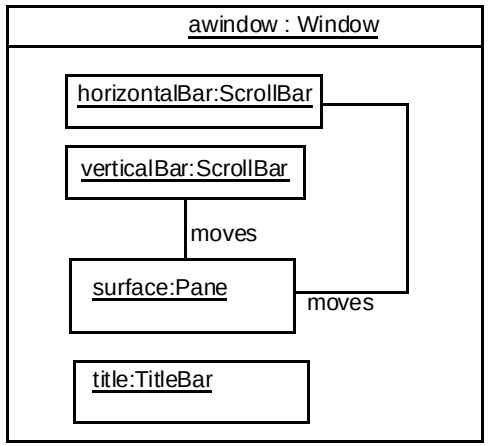


Fig. 3-39, *UML Notation Guide*

Restricciones y comentarios

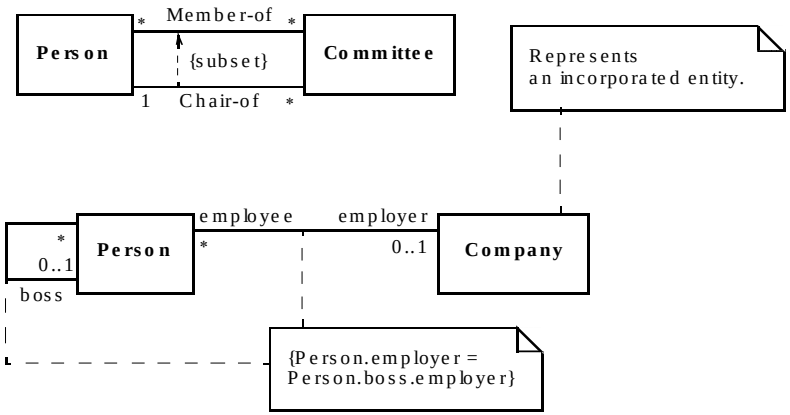
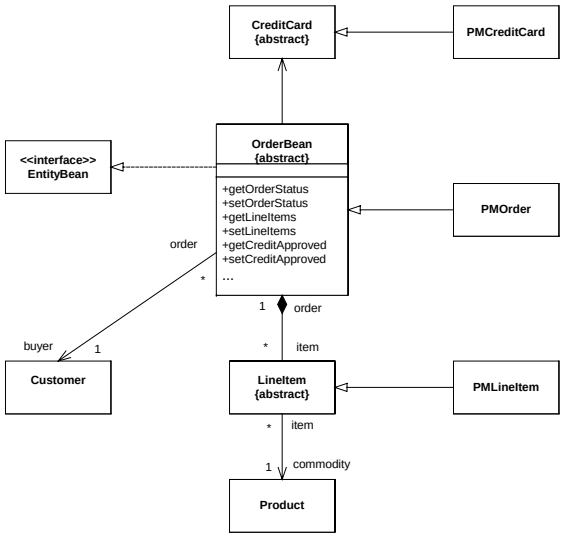


Fig. 3-17, *UML Notation Guide*

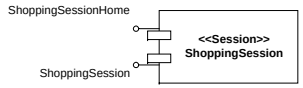
Ejemplo de diagrama de clases



Diagramas de implementación

- Muestran los aspectos de implementación del modelo
  - Estructura del código fuente y objeto
  - Estructura de la implementación en ejecución
- Dos tipos de diagramas
  - Diagrama de componentes
    - Organización y dependencias entre los componentes software
      - Clases de implementación
      - Artifactos binarios, ejecutables, ficheros de scripts
  - Diagrama de implantación
    - Configuración de los elementos de proceso en tiempo de ejecución y los componentes software, procesos y objetos que viven en ellos
    - Distribución de componentes

Componentes



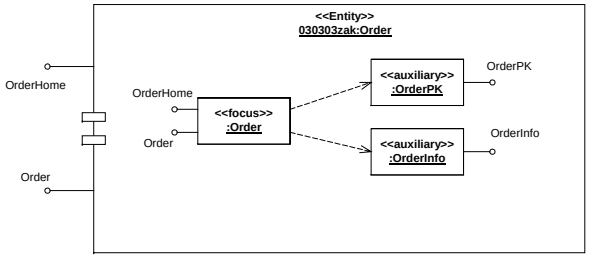


Fig. 3-99, *UML Notation Guide*

Diagrama de componentes

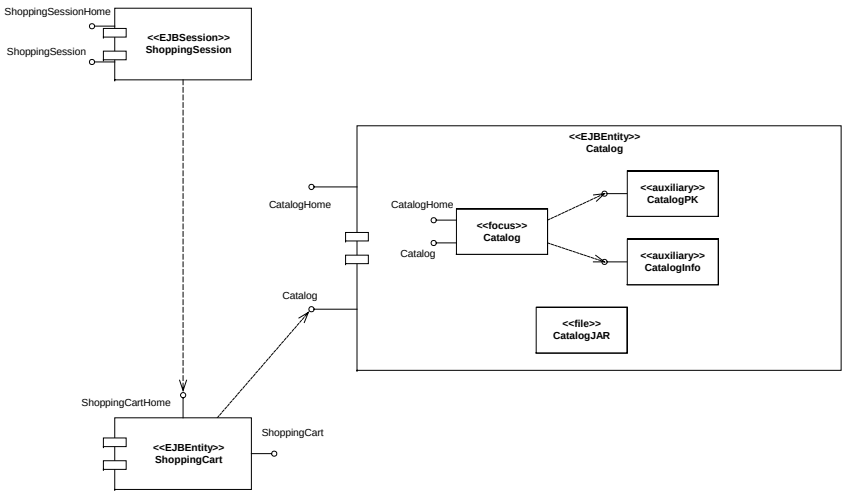


Fig. 3-95, *UML Notation Guide*

Diagrama de componentes con relaciones

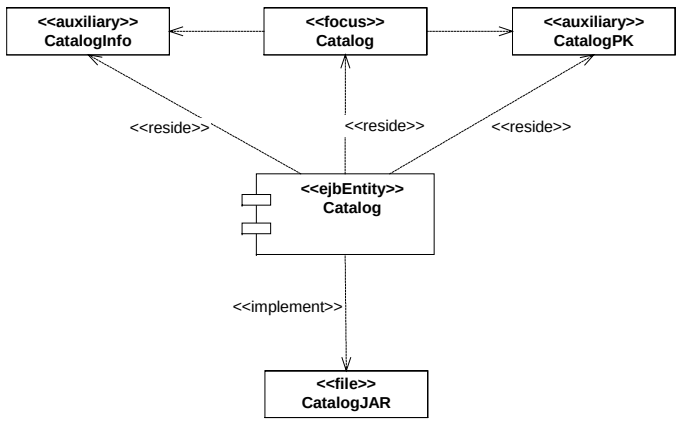


Fig. 3-96, *UML Notation Guide*

Diagrama de implantación

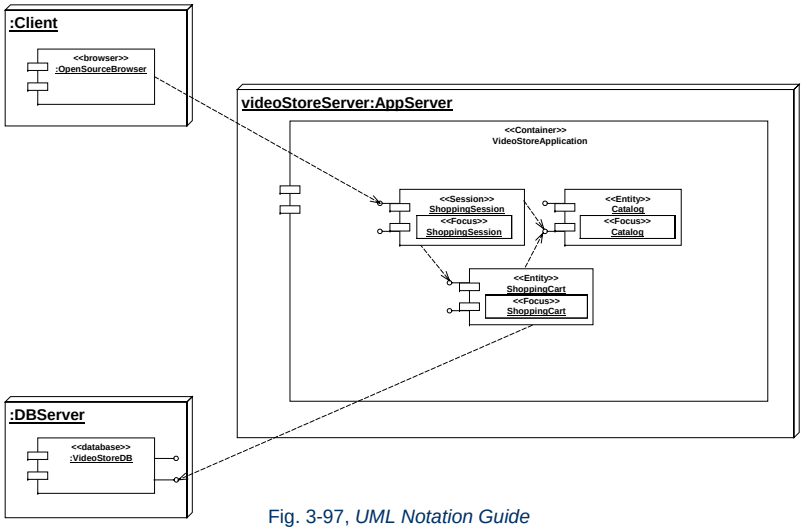


Fig. 3-97, *UML Notation Guide*

Diagrama de implantación

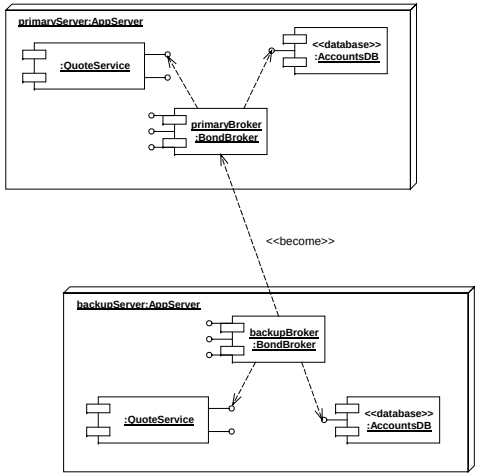


Fig. 3-98, *UML Notation Guide*

Consejos para un buen modelo estructural

- Definir un esqueleto que pueda extenderse y refinarse a medida que se aprende más sobre el dominio
- Utilizar construcciones básicas
  - La notación avanzada dejadla para los detalles, si necesaria
- Dejar los detalles de implementación para el final
- Cada diagrama estructural debería
  - mostrar un aspecto particular del modelo estructural
  - contener clases del mismo nivel de abstracción
- Si hay un gran número de clases, organizarlas en paquetes

Cuándo hacer un modelo estructural

- Utilizar planteamientos de análisis y diseño ascendente (bottom-up) y descendente (top-down)
  - Especificar la estructura de alto nivel usando clases que tenga significado arquitectural y elementos que gestionen la complejidad (paquetes, subsistemas,...)
  - Especificar la estructura de bajo nivel a medida que se descubren detalles, se reclasifica, y definen relaciones
- Si se conoce bien el dominio se puede empezar con un modelo estructural, si no
  - Si se utiliza un método dirigido por casos de uso asegurarse de la consistencia del modelo estructural con el de casos de uso
  - Si se utiliza un método dirigido por colaboraciones (basado en roles) asegurarse que el modelo estructural es consistente con el modelo de colaboración

Ejercicio de modelo estructural

- Realizar un diagrama de clases para el ejemplo anterior
- Definir un diagrama de implantación indicando qué clases residen en cada nodo

Modelo de comportamiento

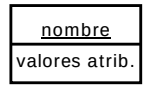
- Visión del sistema que describe las interacciones entre los elementos del sistema y la evolución del estado del sistema y sus componentes
- Se define mediante:
  - Diagramas de interacción
    - Diagrama de secuencias
    - Diagrama de colaboración
  - Diagramas de estado
  - Diagramas de actividad

Diagramas de interacción

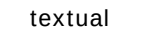
- Interacción
  - Comunicación entre instancias, incluyendo
    - Invocación de operaciones
    - Creación y destrucción de instancias

Diagramas de interacción

- Elementos de un diagrama de interacción
  - Instancia (objeto, valor de datos, de componente)
    - Una entidad con identidad única a la que se pueden aplicar un conjunto de operaciones (enviar señales), que tiene un estado y almacena los efectos de las operaciones (las señales)

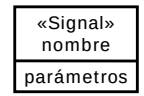


  - Acción
    - Especificación de una sentencia de ejecución.
    - Hay acciones definidas por defecto, como CreateAction, CallAction, DestroyAction, and UninterpretedAction



  - Estímulo
    - Comunicación entre dos instancias
  - Operación
    - Declaración de un servicio que se le puede solicitar a una instancia

  - Señal
    - Especificación de un estímulo asíncrono comunicado entre instancias

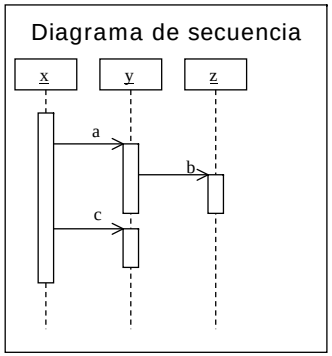


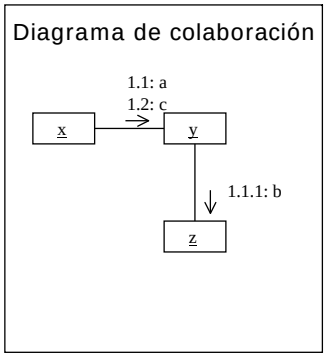
Diagramas de interacción

- Relaciones en un diagrama de interacción
  - Enlace
    - Conexión entre instancias
  - Atributo de enlace
    - Lugar nombrado de una instancia que guarda el valor de un atributo

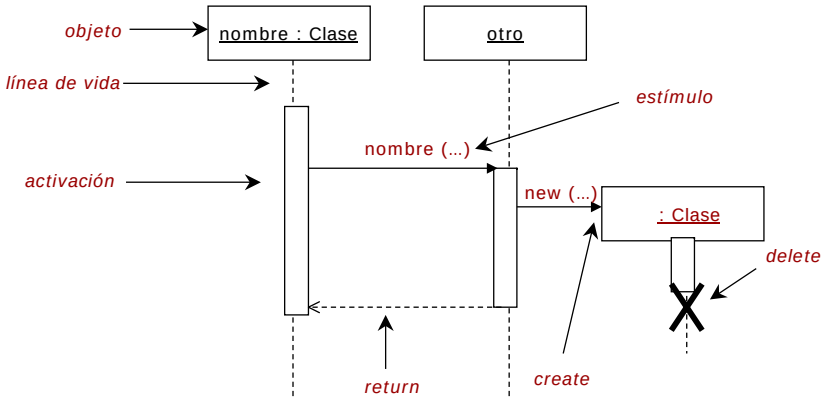
textual

Diagramas de interacción





Diagramas de secuencia



El estímulo se puede poner *[condición] variable := mensaje(parms)*

Diagramas de secuencia

- Recursión y condición

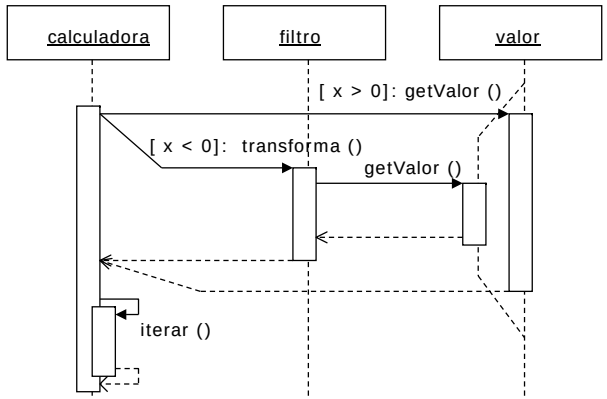
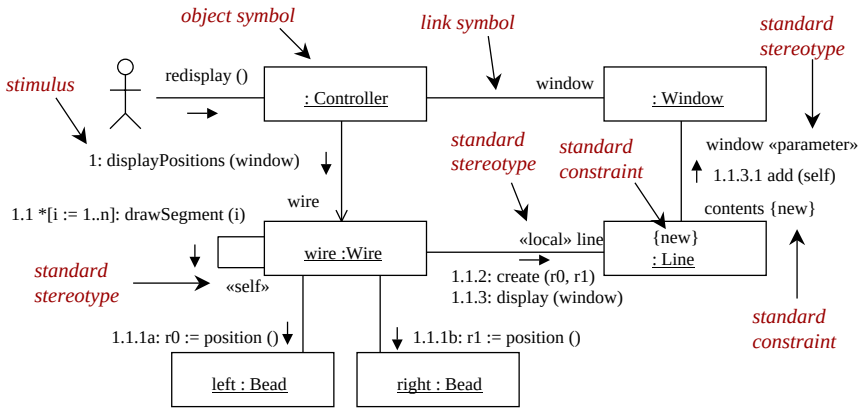


Diagrama de colaboración



Consejos para un buen diagrama de interacción

- Indicar el contexto de la interacción
- Expresar el flujo de izquierda a derecha y de arriba abajo
- Poner las instancias activas en la izquierda arriba y las pasivas a la derecha más abajo

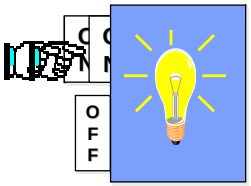
Cuándo usar diagramas de interacción

- Para especificar cómo interaccionan unas instancias con otras
- Para identificar las interfaces de las clases
- Para distribuir los requisitos y las responsabilidades
- *Diagramas de secuencia*
  - Ilustra escenarios típicos con el orden explícito de los estímulos
  - Para modelar tiempo real
- *Diagramas de colaboración*
  - Coordinación de la estructura y control de los objetos
  - Para concentrarse en los efectos sobre las instancias
  - Ayudan a agrupar objetos en módulos ya que las relaciones entre objetos son visibles

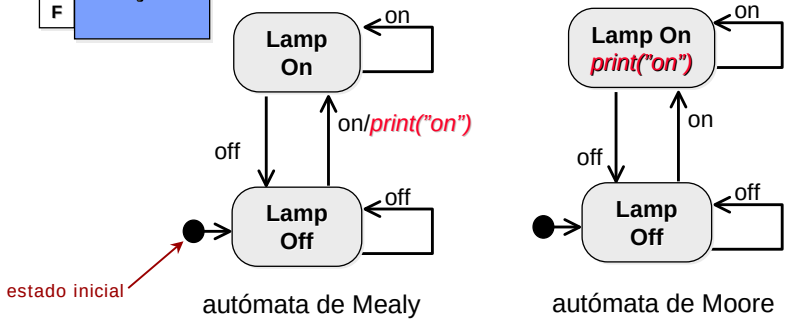
Diagramas de transición de estados

- Modela el ciclo de vida de un objeto por medio de un autómata vinculado a la clase
- Una máquina de estados extendida consta de:
  - un conjunto de señales de entrada (alfabeto de entrada)
  - un conjunto de señales de salida (alfabeto de salida)
  - un conjunto de estados
  - un conjunto de transiciones
    - señal de activación
    - acción
  - un conjunto de variables de estado
  - un estado inicial
  - un conjunto de estados finales (si el autómata termina)

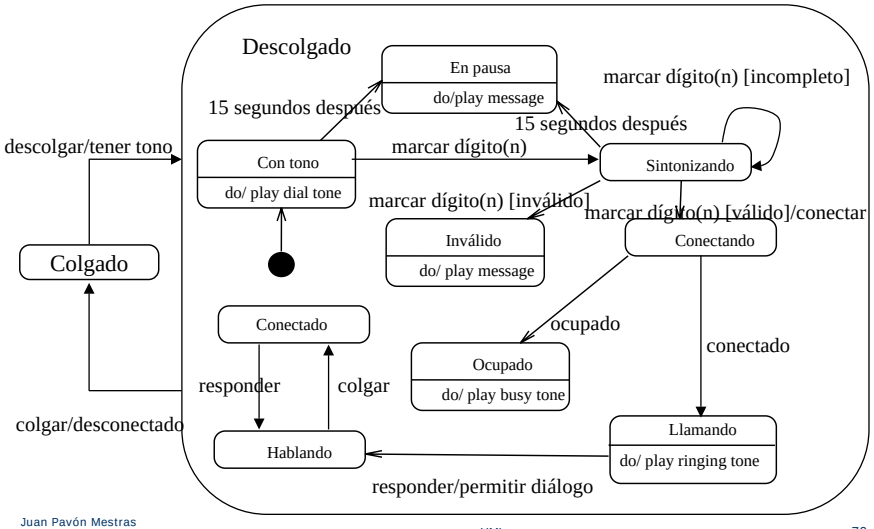
Diagrama de transición de estados



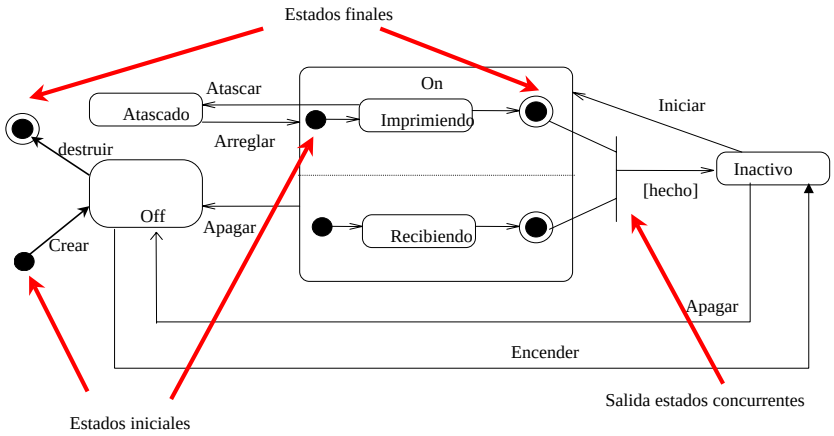
- El estado interno representa la experiencia pasada (la historia de eventos pasados)
- Asociado a cada transición puede haber una salida



Diagramas de transición de estados

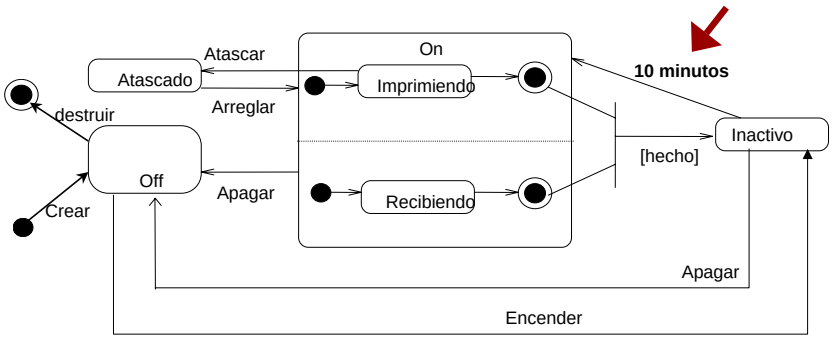


Diagramas de estado: Concurrencia



Diagramas de estado: Temporización

- A veces, se dispara una transición cuando pasa una determinada cantidad de tiempo
- Muchas excepciones son generadas por eventos de tiempo



Cuándo usar diagramas de estado

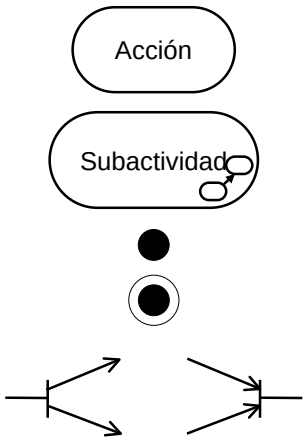
- Para capturar el comportamiento dinámico
  - Cuando tiene cierta complejidad
- Especialmente adecuado con sistemas reactivos
  - Orientados a eventos
    - Interfaces de usuario
    - Dispositivos
    - Protocolos de comunicación
  - Típico del modelo de servidor

Diagramas de actividad

- Muestra el flujo de control/datos entre objetos
  - Mostrar las *acciones* realizadas por una *operación*.
  - Mostrar el *trabajo interno* de un objeto.
  - Mostrar cómo se desarrollan un conjunto de acciones y cómo afectan a los objetos que las realizan.
  - Mostrar cómo se realiza una instancia de un caso de uso en términos de acciones y cambios de estado de los objetos.
  - Mostrar un modelo de negocio en términos de trabajadores (actores), flujos de trabajo, organización y objetos (factores físicos e intelectuales usados en los negocios).
- Se representan como las máquinas de estado pero...
  - Las transiciones suceden cuando se acaba un estado y no activadas por un evento externo
  - Soportan concurrencia dinámicamente

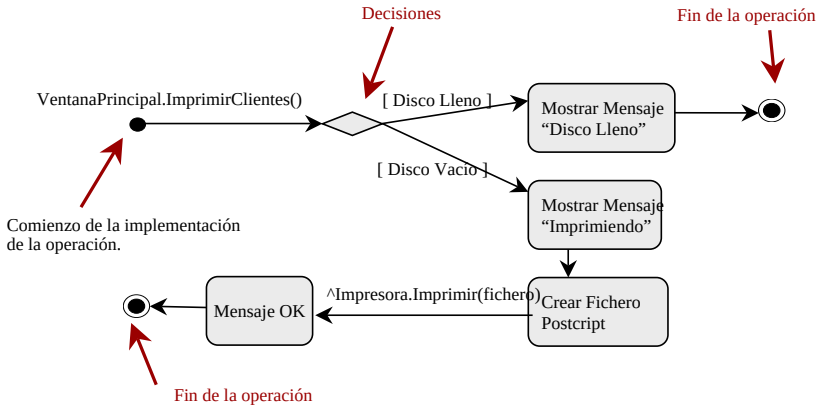
Diagramas de actividad

- Elementos de un diagrama de actividad
  - Acción (estado)
    - invocar una operación en un objeto
    - ejecutar una acción del usuario
  - Subactividad (estado)
    - Inicia otro grafo de actividad sin usar una operación
    - Usado para descomposición funcional
  - Estado inicial
  - Estado final
  - Fork y join



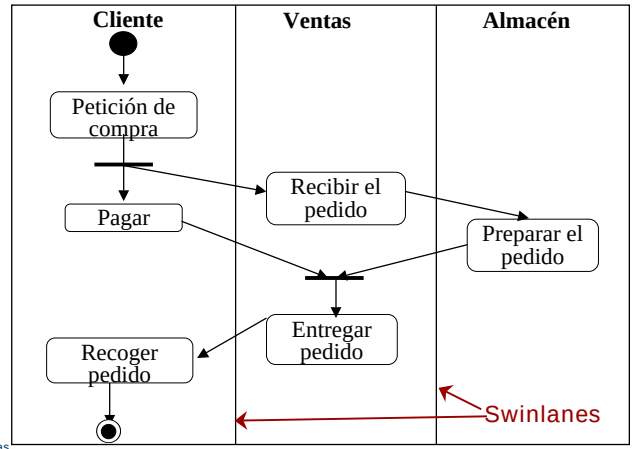
Diagramas de actividad

- Realización de una operación en un objeto



Diagramas de actividad

- Particiones
  - Definición de una transacción en múltiples objetos

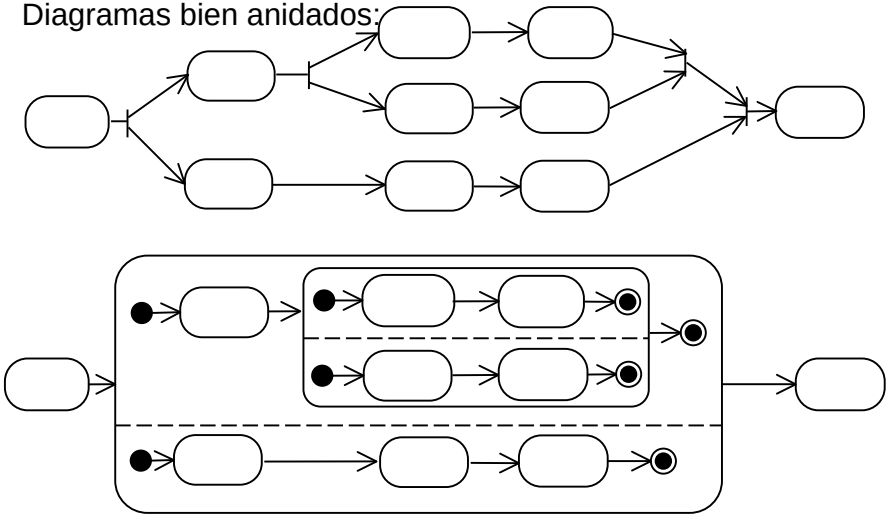


Consejos para un buen diagrama de actividad

- El flujo de control y el flujo de objetos no están separados. Ambos se modelan con transición de estados
- Se pueden mezclar máquinas de estados y flujos de control/objetos en el mismo diagrama
- Definir diagramas bien anidados
  - Ello asegura que habrá una máquina de estados ejecutable correspondiente
  - Evitar el *goto*

Consejos para un buen diagrama de actividad

Diagramas bien anidados:



Cuándo usar diagramas de actividad

- Aplicaciones que necesiten modelos de flujo de control o de flujo de datos/objetos
  - en vez de modelos dirigidos por eventos (para los que son más apropiadas las máquinas de estado)
  - Por ejemplo: modelos de procesos de negocio y workflows
- Cuando el comportamiento
  - no depende mucho de eventos externos
  - los pasos se ejecutan hasta acabar, y no son interrumpidos por eventos
  - requiere flujo de objetos/datos entre los pasos
- Se construyen normalmente durante el análisis para ver qué actividades ocurren en vez de qué objetos son responsables de ellas
  - Posteriormente se pueden usar particiones para asignar esas responsabilidades

Qué diagramas de comportamiento utilizar

- Utilizar diagramas de interacción (secuencia o colaboración) cuando haya interés en conocer el comportamiento de *varios objetos* dentro de *un caso de uso*.
- Utilizar diagramas de estado cuando haya interés en conocer el comportamiento de *un único objeto* dentro de *múltiples casos de uso*.
- Utilizar diagramas de actividad para observar el comportamiento a lo largo de *varios casos de uso*, o *muchos hilos de control*.

Ejercicio: Diagrama de actividad

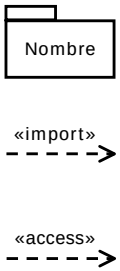
- Definir un diagrama de actividad para la operación de reserva de plaza de hotel

Paquetes

- Agrupación de elementos de modelo
  - Puede contener elementos de modelo de distintos tipos
    - Incluyendo otros paquetes: jerarquías de paquetes
- Define un espacio de nombres para sus contenidos
- Los paquetes se utilizan para:
  - Organizar un modelo grande
  - Para agrupar elementos relacionados
  - Para separar espacios de nombres
  - Para crear una visión general de un conjunto de modelos grande

Paquetes

- Conceptos
  - Paquete
    - Una agrupación de elementos de modelo
  - Import
    - Dependencia que indica que el contenido público del paquete de destino se añade al espacio de nombres del paquete origen
  - Access
    - Dependencia que indica que el contenido público del paquete de destino está disponible en el espacio de nombres del paquete origen



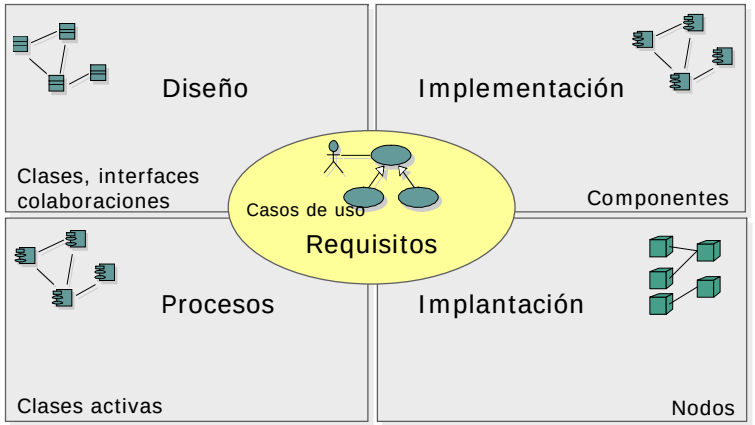
Visibilidad

- Todo elemento contenido tiene una visibilidad relativa al paquete que lo contiene
  - Un elemento *public* es visible a todos los elementos fuera del paquete
    - Se indica con '+'
  - Un elemento *protected* es visible sólo en los paquetes que heredan
    - Se indica con '#'
  - Un elemento *private* no es visible a los elementos fuera del paquete
    - Se indica con '-'
- La misma sintaxis de visibilidad se utiliza con los atributos y operaciones en las clases

Consejos para definir paquetes

- Juntar elementos de modelo que tengan una gran cohesión en un paquete
- Guardar elementos de modelo con baja cohesión en distintos paquetes
- Minimizar las relaciones, especialmente asociaciones, entre elementos de modelos de distintos paquetes
- Una implicación del espacio de nombres: un elemento importado en un paquete *no sabe* cómo se va a usar en el paquete importado

Arquitectura del sistema con UML



UML 2.0

- Definido en 2003, aunque se espera su aprobación definitiva a finales de 2004
  - Revisión de la experiencia en el modelado y en las herramientas de UML 1.x
  - Metamodelo definido con MOF
  - Soporte de Model Driven Architecture (MDA) y Service Oriented Architecture (SOA)
- Cambios principales:
  - Diagramas de actividad
  - Se pueden expresar excepciones
  - Diagramas de comunicación en vez de diagramas de colaboración
    - Similares pero contenidos en un marco
    - Soportan mensajes concurrentes
  - Casos de uso
    - Se pueden expresar condiciones en los puntos de extensión
- Pocas herramientas adaptadas a UML 2
  - Together Designer Community Edition

UML 2.0

- Diagramas de actividad
  - Los nodos, en vez de *actividades*, son *acciones*
    - Los diagramas de actividad muestran las acciones que constituyen una actividad
  - Una acción tiene precondiciones y postcondiciones
  - Una acción empezará cuando todos los flujos entrantes estén activos
  - También se puede definir la recepción de señales para iniciar una acción (por ejemplo, una señal de tiempo)
  - Las acciones se pueden descomponer (subactividades) y agrupar
  - Una actividad se puede particionar en grupos de acciones que corresponden a roles, lugares, organizaciones, etc.

Bibliografía

[UML 1.5] *OMG Unified Modeling Language Specification* v. 1.5, Marzo 2003.
OMG UML: www.omg.org/uml
También www.uml.org
Cetus links: www.cetus-links.org/oo_uml.html