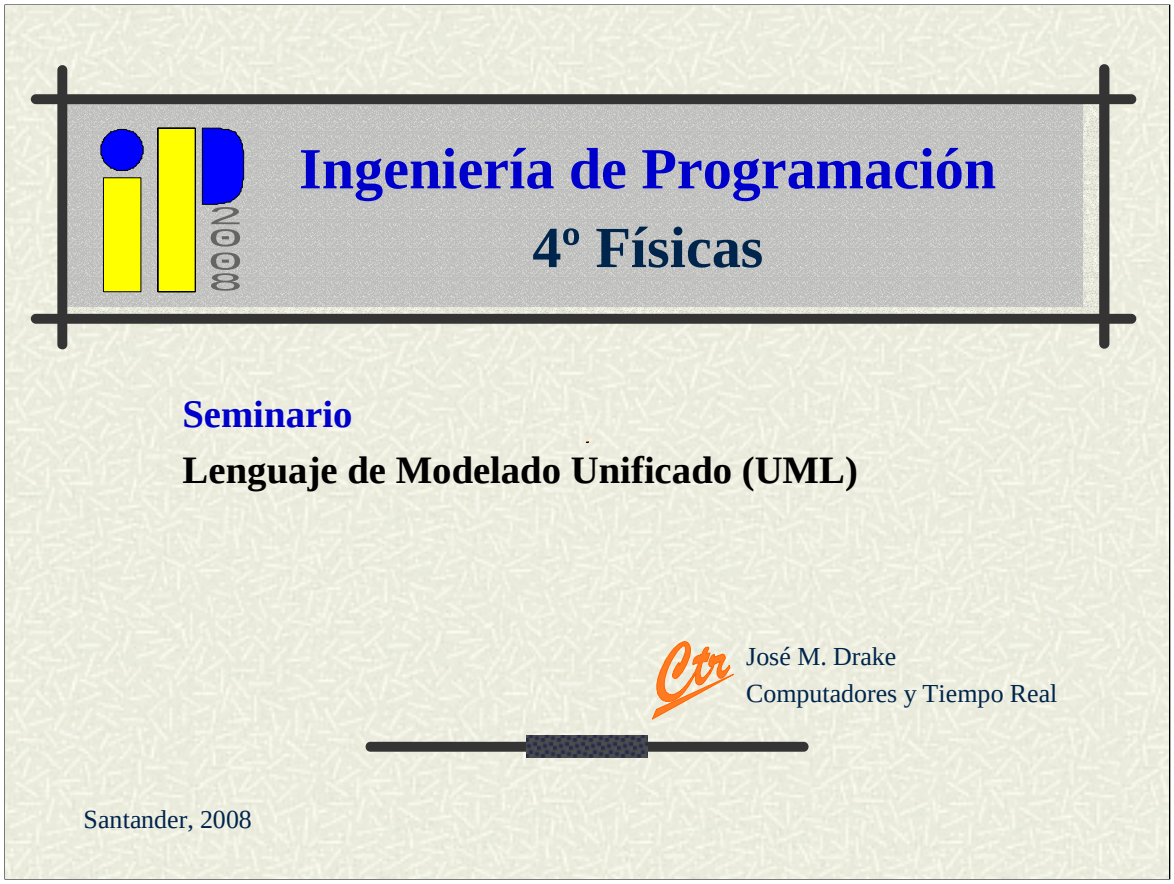
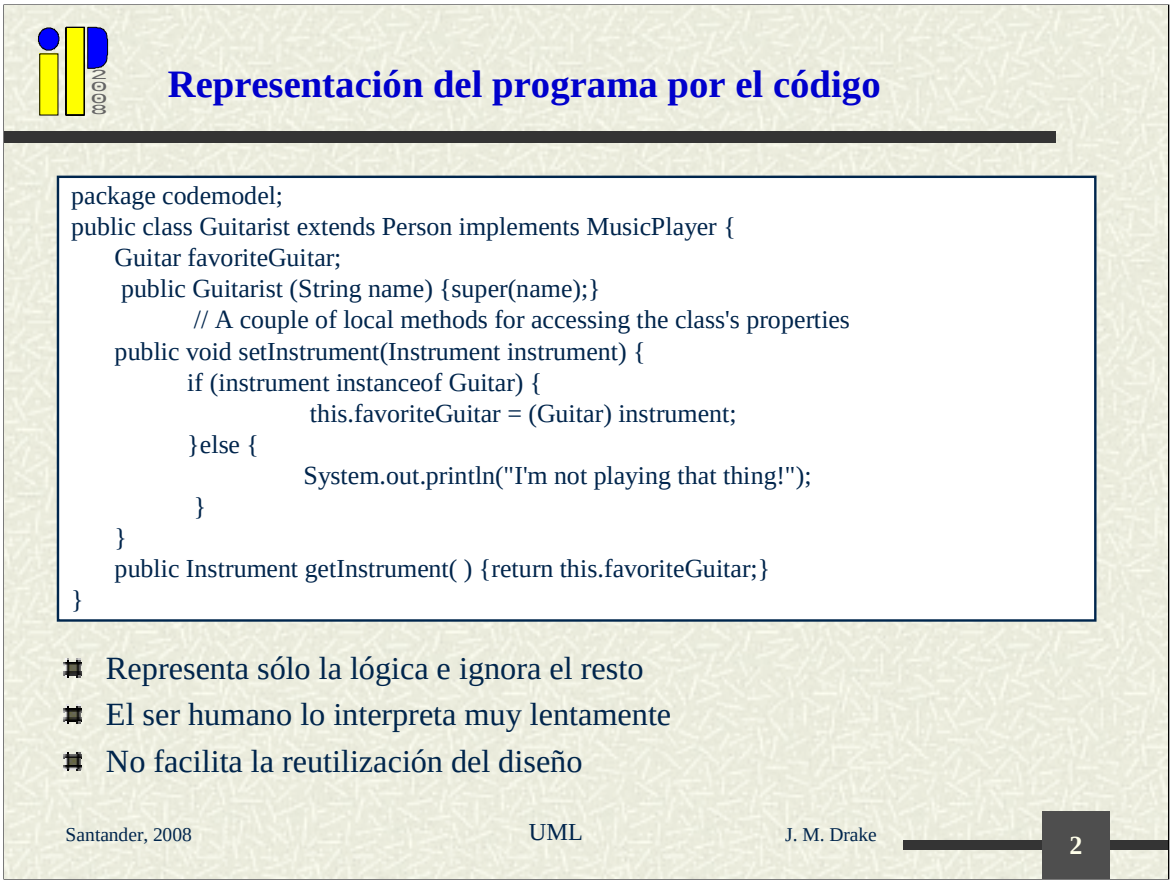

Ingeniería de Programación
4º Físicas
Seminario
Lenguaje de Modelado Unificado (UML)
José M. Drake
Computadores y Tiempo Real
Santander, 2008

Notas:



Notas:

El código es un ejemplo extremo de lenguaje para describir un programa. En él no se ha realizado ninguna abstracción. Cada línea de código contiene los detalles de como el programa debe trabajar.

Problemas que tiene el uso del código como medio de descripción son:

•El código ha sido escrito para ser interpretado por el compilador, y en consecuencia contiene muchos detalles que sólo tienen interés para él y que hacen compleja su interpretación por un diseñador humano.

•El código describe únicamente en el software en sí, e ignora el resto del sistema. Aún pensando que el código es una definición completa y no ambigua de lo que el software hace, no nos indica el como se utiliza y quien debe utilizarlo. Con el código se pierde la visión completa del software.

•El código no es un medio adecuado para intercambiar ideas entre programadores y diseñadores humanos, los cuales deben de inventar otros recursos para comunicarse.

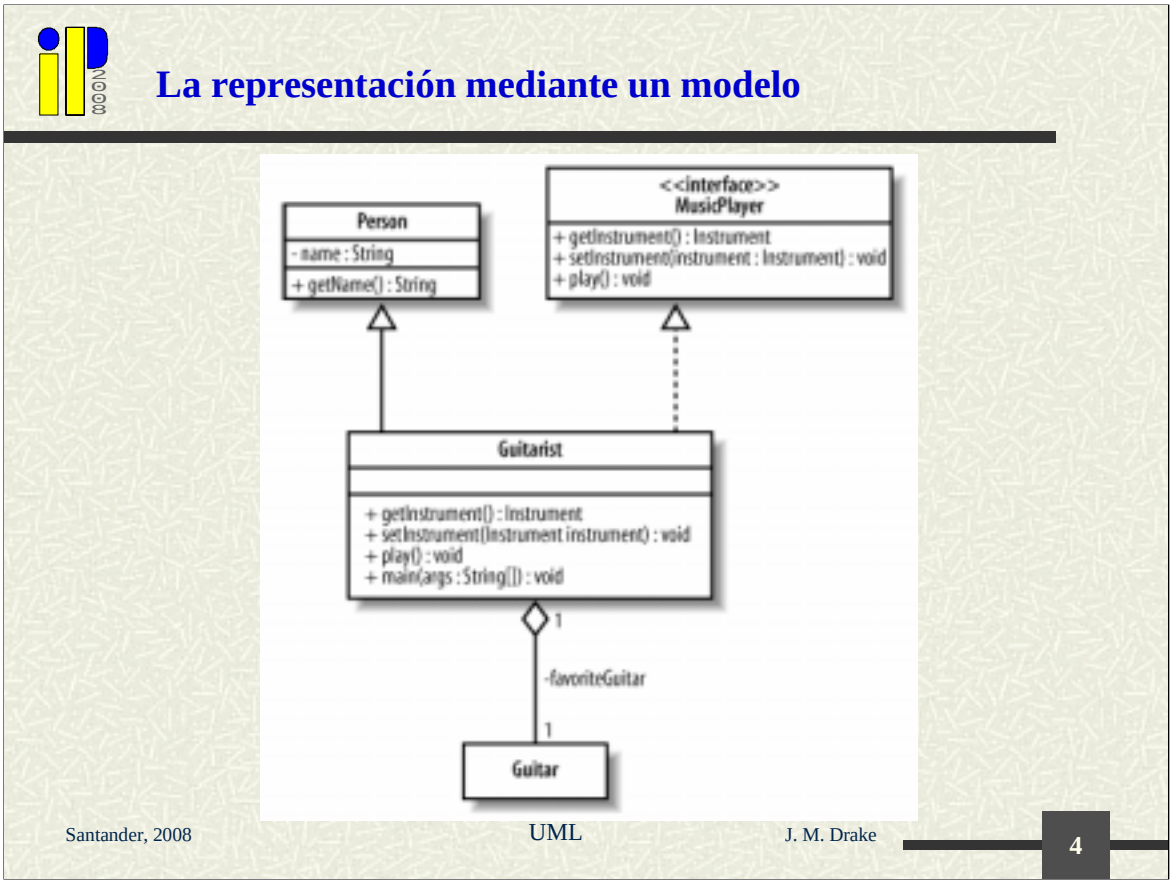
•El código debe interpretándose leyendo un texto, y esto es muy lento para las personas.

•Si lo que se está haciendo es diseñar el sistema, su representación como código dificulta su posibilidad de reutilización en otros sistemas, posiblemente desarrollados en otros lenguajes.

Notas:

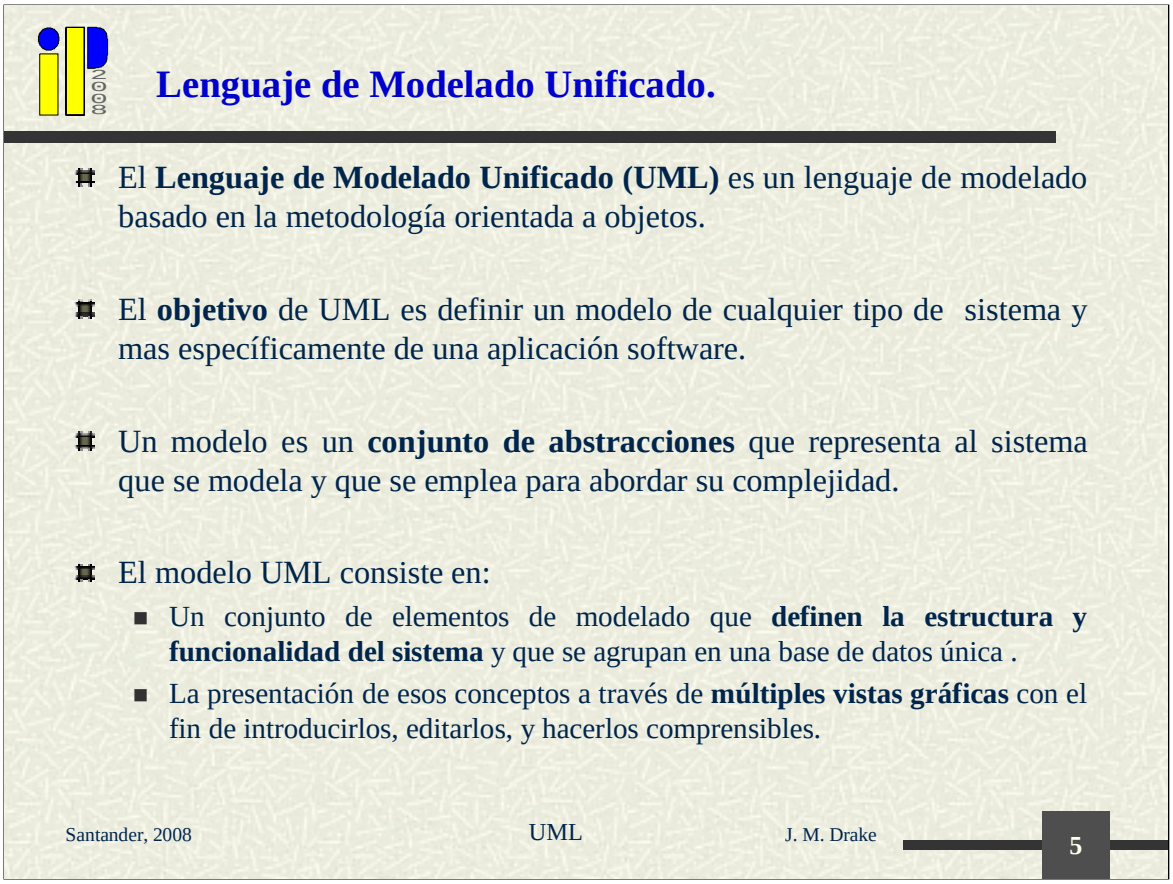
En el extremo opuesto al código se encuentra la descripción del software mediante lenguaje natural. Este es informal y no sigue una notación bien definida y concensuada. Los posteriores lectores de esta descripción pueden entender muy diferentes interpretaciones de lo que representan.

También es textual, por lo que su interpretación por un humano es muy lenta y tediosa.



Notas:

En la figura se muestra una vista de la descripción UML de una sección de código. Si conoce el lenguaje UML y la semántica de sus símbolos, una simple ojeada del grafico da una visión general de lo que describe, y además proporciona las vías por las que acceder a una información mas detallada de cualquier elemento (incluido sus códigos). Esto se debe a que el UML es un lenguaje formal: un lenguaje sencillo con un conjunto de reglas bien definidas que evitan la ambigüedad. Cualquier persona con conocimientos de UML interpretaría este diagrama de la misma forma.



Notas:

El Lenguaje de Modelado Unificado (UML) es un lenguaje basado en la metodología orientada a objetos creado por el Object Management Group (OMG) con la intención de convertirse en un estándar para el modelado de aplicaciones software, tanto a efectos de servir de base de la ingeniería software, servir de método de intercambio de ideas entre los equipos que desarrollan software y como base a las herramientas que utilizan.

UML es un lenguaje de modelado universal con capacidad de describir cualquier tipo de sistema, sin embargo, su desarrollo semántico ha sido específicamente desarrollado para el modelado de aplicaciones software.

El objetivo de UML es permitir al diseñador formular un modelo del sistema y por modelo se entiende la definición de un conjunto de abstracciones semánticas que permiten describir simbólicamente los conceptos estructurales, de comportamiento y funcionales que se proponen para el sistema. ES un modo de abordarla complejidad de un sistema, centrándonos en ciertos aspectos del sistema por separado.

En UML los elementos que constituyen el modelo se definen de forma integrada y coherente dentro de una única base de datos. El modelo se presenta a través de múltiples vistas gráficas, que presentan parcialmente diferentes aspectos de la semántica del modelo. Las vistas son el recurso principal para introducir, editar y presentar el modelo. El conjunto de todas las vistas son parte o la descripción semántica completa del sistema.

Ventajas del UML

- Está basado en **metamodelo** con una semántica bien definida.
- Se basa en una **notación gráfica** concisa y fácil de aprender y utilizar.
- Es un **estándar** soportado y mantenido por la OMG (Object Management Group).
- Es independiente de cualquier fabricante comercial.
- Es **escalable** y permite representar y gestionar tanto sistemas masivos como pequeños sistemas.
- Es un lenguaje concebido y evolucionado como **herramienta de desarrollo** de aplicaciones software.

Notas:

- UML está basado en un modelo bien definido semánticamente que se denomina “UML Metamodel”. El modelo semántico es a la vez “amplio” (cubre la mayoría de los aspectos necesario para la especificación de sistemas software) y “profundo” (contiene la suficiente información para generar prototipos ejecutables destinados a su validación, o para generar esqueletos de código fuente).
- Es un lenguaje basado en una notación gráfica fácil de usar y comprender basado en un conjunto reducido de tipos de diagramas (diagramas de clases, diagramas de despliegue, diagramas de estados, diagramas de actividad, diagramas de secuencias y diagramas de casos de uso) estandarizados y que además se basan en un conjunto de principios comunes.
- Es un lenguaje gestionado en su creación y mantenimiento por el grupo internacional OMG (Object Management Group) que es independiente de cualquier fabricante con interese comercial, lo cual garantiza su permanencia y estabilidad, así como la existencia de múltiples fuentes y diferentes tipos de herramientas. La primera versión UML 1.1 fue propuesta a finales de 1997, y hace unos meses ha aparecido la versión UML 2.0. El curso se basará en la versiones 1.2 y 1.4. que son las soportadas por las herramientas.
- Es un lenguaje de tercera generación que ha heredado la experiencia de dos generaciones de herramientas previas y de cientos de especialistas que las han utilizado. Actualmente es un lenguaje refinado para que pueda ser eficientemente aplicado en el desarrollo de aplicaciones software, tanto si son proyectos pequeños llevados por una o dos personas o si son grandes proyectos desarrollados por cientos de personas.

Aspectos semánticos del UML

- **Descripción funcional del sistema**: Describe la especificación funcional del sistema con independencia de su implementación. Cómo se va a usar el sistema y cómo se espera que responda.
- **Elementos estructurales**: Describen los elementos de que se compone el modelo, así como los parámetros que los cuantifican y de las relaciones entre ellos.
- **Descripción de los comportamientos**: Describen los comportamientos de los elementos de por sí y las interacciones entre grupos de elementos.

Notas:

La semántica de un sistema se describe a través de tres aspectos complementarios:

- Los aspectos funcionales del sistema se refieren a la especificación funcional y no funcional (Q&S) del sistema que se desarrolla con independencia de la implementación que se proponga o realice. En UML la descripción funcional se realiza mediante diagramas de casos de uso. Estos diagramas modelan cómo se va a usar el sistema y cómo se espera que el sistema responda.
- Los elementos estructurales describen las “cosas” que constituyen el modelo, ya sean en fase de diseño como son las clases o tipos como en fase de ejecución como son los objetos o instancias. Así mismo, estos elementos se describen con diferentes niveles de abstracción, tanto a pequeño nivel próximo al código como son las clases, interfaces u objetos, como a gran nivel próximo a la organización arquitectural como son los subsistemas y los componentes. Así mismo, las relaciones entre las clases que describen sus interdependencias son también parte de la estructura del modelo.
- Los aspectos de comportamiento incluidos en el modelo definen como se comportan o interactúan entre si los diferentes elementos formulados en la estructura. Para los elementos estructurales individuales (clases, objetos, subsistemas, componentes, casos de uso) UML permite describir su comportamientos a través de diagramas de estados o diagramas de actividad que describen la secuencias de estado que siguen en función de las interacciones que reciben, mientras que para las agrupaciones de elementos estructurales, UML permite describir los posibles patrones de interacción a través de diagramas de colaboración o diagrama de secuencias.

Vistas del modelo UML.

- Un modelo UML de una aplicación puede tener muy **diferentes niveles de detalle**:
  - **Modelo arquitecturales**: con información cualitativa de alto nivel.
  - **Modelos detallados**: con información que permiten la validación o la generación de código.
- No se requiere una vista que contenga la **semántica completa** de la aplicación. La semántica reside en la base de datos.
- Las **múltiples vistas** permiten enfocar la atención en aspectos concretos con el adecuado nivel de detalle:
  - Vista de casos de uso
  - Vista de clases
  - Vista de despliegue
  - Vista de componentes
- Las vistas gráficas sirven de medio para introducción del modelo.
- La gestión de un modelo UML requiere una **herramienta específica** que mantenga la consistencia del modelo.

Notas:

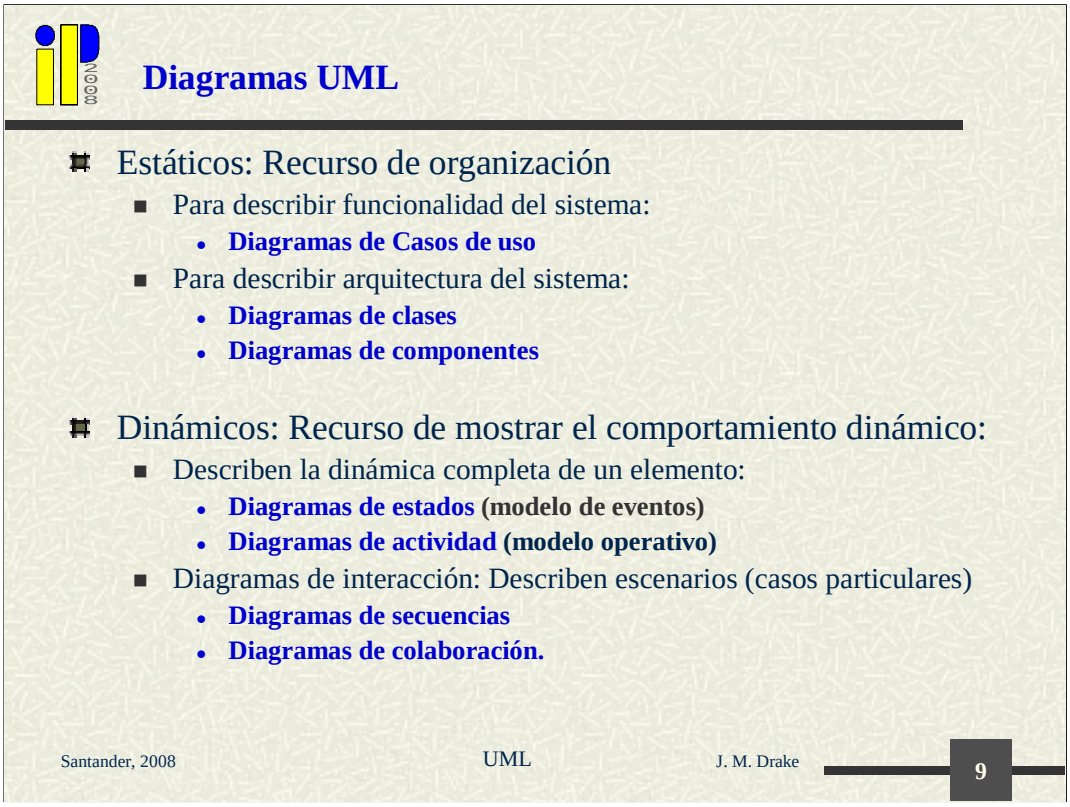
Con UML el diseñador crea un modelo de aplicación que describe de forma completa, consistente y precisa el sistema que se desarrolla. La información contenida en el modelo puede permitir análisis y simulaciones del sistema a través del modelo, o así mismo ser la base para la generación automática de esqueletos o módulos completos de código fuente destinado a la compilación.

Cada vista del modelo describe algún aspecto particular de la semántica del sistema que se muestra a efecto de presentar un cierto nivel de abstracción.

La semántica reside en la información contenida en la base de datos en que se almacenan los elementos de modelado, y esta no reside en una vista singular en particular. En un modelo UML bien presentado, el conjunto de las vistas describe la semántica de la aplicación.

La multiplicidad de las vistas es de gran utilidad ya que permiten enfocar la atención del usuario a aspectos concretos que requieren una atención especial, con una cantidad de detalles adecuados al estudio que se realiza.

Para gestionar un modelo UML se requiere una herramienta específica, ya que no solo debe permitir dibujar las vistas, sino mantener la coherencia entre los componente que se incluyen en las diferentes vistas.



Notas:

UML ofrece muchos recursos para expresar tanto los elementos estructurales que forman un sistema como su funcionalidad. Estos recursos pueden utilizarse bien para especificar la funcionalidad, o bien para describir detalladamente esa funcionalidad. Los elementos de modelado en UML se muestran a través de diagramas.

El tipo de diagramas que nos ofrece UML para definir la estructura de un sistema son:

•Diagramas de clases: Sirve para describir el conjunto de clases, objetos, interfaces, etc que forman el sistema (los elementos estructurales)

•Diagramas de componentes: Sirve para agrupar elementos de modelado en elementos de mayor nivel de abstracción.

Estos diagramas son estáticos, muestra como se organiza el sistema, no como funciona.

Para la descripción de la funcionalidad de estos elementos, es decir de su comportamiento dinámico existen dos tipos de diagramas:

•Aquellos que describen la dinámica completa de un elemento:

•**Diagrama de estado**: sirve para describir un elemento estructural a través de una máquina de estados finitos.

•**Diagrama de actividad**: es un diagrama de estado con una semántica específica formulado como forma de expresar algoritmos complejos a través de diagramas de flujo.

•Elementos para la descripción de mecanismos de operación de interacción entre grupos de objetos del sistema:

•**Diagrama de secuencias**: Permite representar a través de diagramas de transferencia de mensajes entre objetos organizados en el tiempo cualquier escenario particular de colaboración entre ellos.

•**Diagrama de colaboración**: Permite representar a través de diagramas gráficos los escenarios de colaboración que se establecen entre grupos de objetos.

Elementos de modelado estructural de bajo nivel

- Describen los elementos básicos que constituyen el sistema:
  - **Objeto**: Es un concepto de ejecución y es una estructura de datos junto con la descripción de las operaciones que actúan sobre ellos.
    - Los datos del objeto, que definen su estado, son los "**atributos**".
    - Las operaciones que actúan sobre los datos son los "**métodos**".
  - **Clase**: Es un concepto de diseño y describe las características comunes de un tipo de objetos, de los que se pueden crear múltiples instancias.
  - **Interfaz**: Representa un conjunto de operaciones que en conjunto representa un servicio. Es un elemento de diseño principalmente orientado hacia la reusabilidad. Una interfaz no se puede instanciar, es necesario definir una clase que la implemente.

Notas:

UML es un lenguaje de modelado que se basa en la orientación a objetos, por ello los objetos, las clases y la interfaces son los elementos de modelado estructural básico. Están destinados a representar e incorporar la información que corresponde con los conceptos de bajo nivel que constituyen la aplicación.

Un objeto modela una estructura de datos que se instancia durante la fase de ejecución de la aplicación. El objeto modela tanto la información que lleva asociada y que se describe mediante atributos y las operaciones que pueden operar sobre ella y que se describe mediante métodos. Estos métodos son invocados por otros objetos clientes o por otros métodos del mismo objeto.

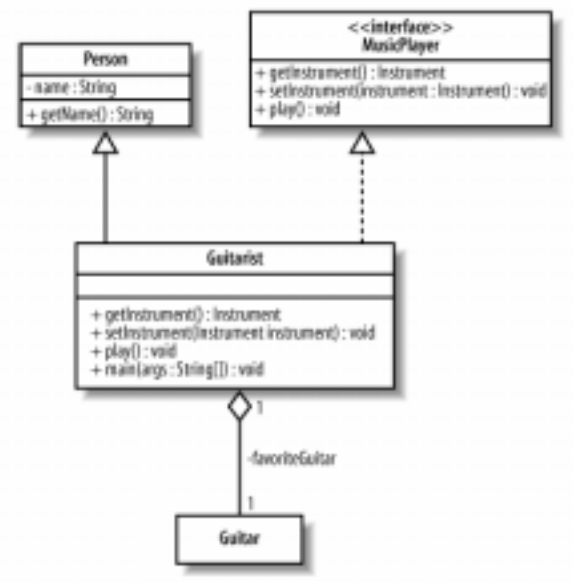
Una clase describe durante la fase de diseño a un conjunto de objetos que instancian las mismas estructuras de datos y admiten los mismos tipos de operaciones. Durante la fase de ejecución pueden instanciarse muchos objetos que correspondan a la misma clase, mientras que un objeto es siempre la instancia de una única clase.

Las clases se pueden definir en función de otras clases definidas previamente, bien por especialización, extensión, agregación, asociación, etc. lo que reduce considerablemente la complejidad de la definición del gran número de instancias que requiere una aplicación.

Las interfaces permiten nombrar conjuntos de operaciones que representan un servicio completo y que pueden ser ofertados por diferentes clases que los ofertan. Su función es independizar el servicio de la implementación. Dos clases que realicen una misma interface, ofrecen el mismo servicio, lo que significa que si una clase requiere un servicio, lo puede obtener de cualquier clase que realice la misma interfaz.

Una interfaz no es directamente instanciables. Las interfaces se realizan a través de clases. Para ello la clase debe implementar un método concreto, por cada operación de la interfaz.

Diagrama de clases



Notas:

Un diagrama de clases muestra un conjunto de clases e interfaces y las asociaciones entre ellas. Sirven para describir la estructura estática de un sistema

Elementos de modelado estructural a alto nivel

- Describen los elementos de alto nivel que constituyen el sistema desde el punto de vista arquitectural:
  - **Paquete:** Elemento contenedor que permite organizar los elementos de modelado en bloques a efecto de su gestión durante la fase de diseño.
  - **Subsistema:** Elemento estructural de alto nivel que sirve para agrupar conjuntos de elementos de modelado en función de que van a contribuir a un aspecto de comportamiento del sistema.
  - **Componente:** Elemento estructural de alto nivel que agrupa conjuntos de elementos de modelado en función de que constituyen un bloque reemplazable. Para ello deben especificar las interfaces que ofrecen y que requieren.
  - **Nudo**: Modela un elemento o entorno físico del sistema en el que se puede ejecutar algún elemento software.

Notas:

Dada la complejidad de los sistemas que actualmente se desarrolla, es habitual que el sistema deba dividirse en grandes módulos que facilite su gestión. UML ha definido la semántica de ciertos elementos de modelado que facilitan la gestión de estas grandes unidades.

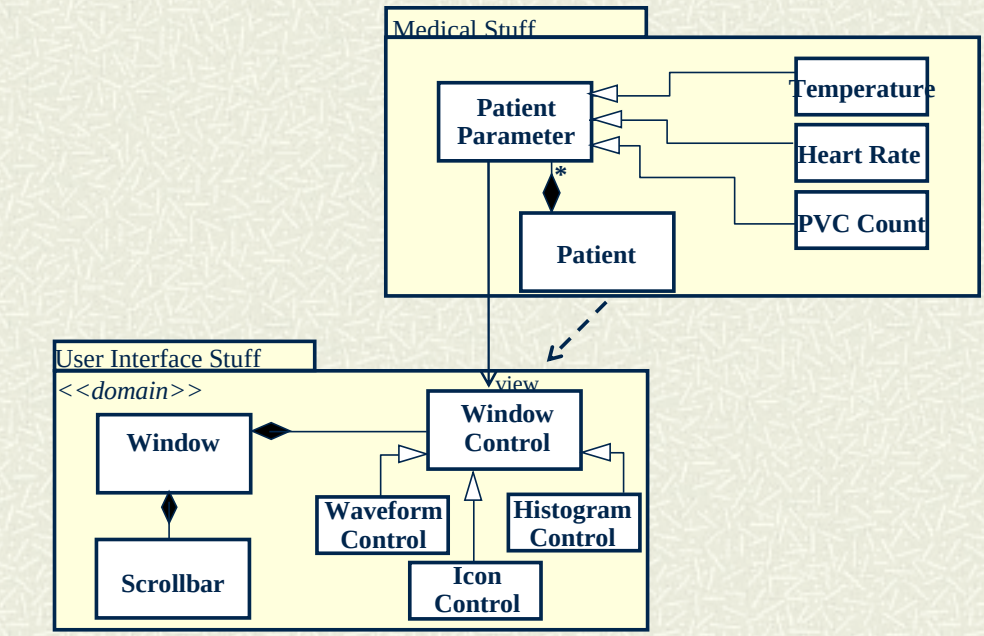
Los paquetes son elementos de modelado que tienen como función servir de contenedores de otros elementos de modelado a efectos de su organización. Los paquetes se utilizan para dividir los modelos en fragmentos a fin de que puedan ser gestionados mas eficientemente o procesados por personas o equipos diferentes. Los paquetes solo contienen elementos que existen en a fase de diseño y nunca corresponden a algo que sea instanciable. Sirven para organizar modelos o para proporcionar un dominio de denominación.

Los subsistemas son utilizados para organizar módulos de elementos que van a existir en la fase de ejecución del sistema y que se agrupan en función de que colaboran para proporcionar una determinada funcionalidad. Los sistemas son los elementos de mayor nivel de abstracción destinados a describir la arquitectura física de un sistema. Los subsistemas contienen información sobre las operaciones, servicios y especificaciones de QoS e implementación.

Los componentes son elementos de organización de módulos de un sistema, semejantes a los subsistemas, pero que por corresponder a servicios completos y bien definidos se caracterizan por constituir unidades reemplazables.

Los nudos son elementos de modelado de unidades hardware (procesadores, redes de comunicación, equipos, etc.) que constituyen la plataforma en la que se va a ejecutar el sistema que se desarrolla.

Representación de paquetes



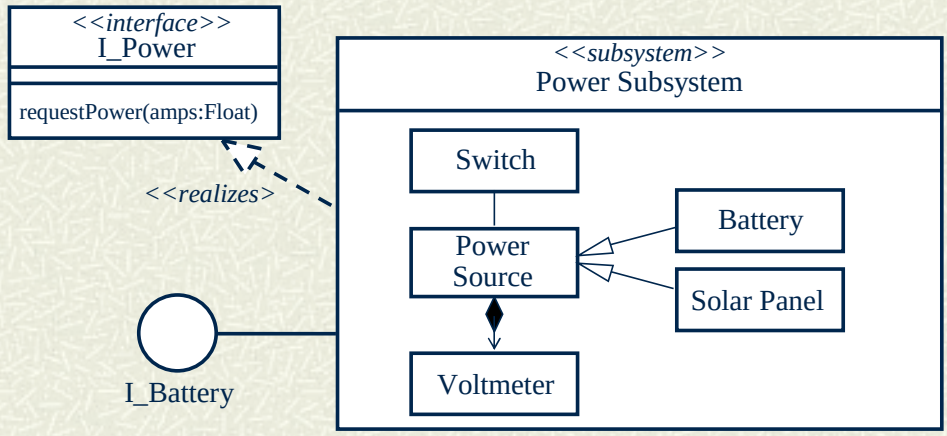
Notas:

En UML los paquetes se representan por un icono en forma de carpeta. En ella se incorporan los elementos de modelado que contienen.

Las carpetas como todo elemento UML pueden tener asignado un estereotipo. En el ejemplo la carpeta User Interface Stuff se le asignado el estereotipo <<domain>>, lo que significa que utilizamos el paquete como forma de agrupar elementos que corresponden a un mismo dominio del problema.

Las carpetas UML tienen la misma semántica que las carpetas de los sistemas de ficheros, y son elementos contenedores que pueden agrupar cualquier tipo de elemento de modelado o de diagrama de visualización. Aunque los paquetes pueden representarse en los diagramas de clases a efectos de organizar los elementos de modelado estructural, su contenido real debe ser observado a través de los navegadores del modelo, que se organizan en función de los paquetes que se definen.

Subsistemas

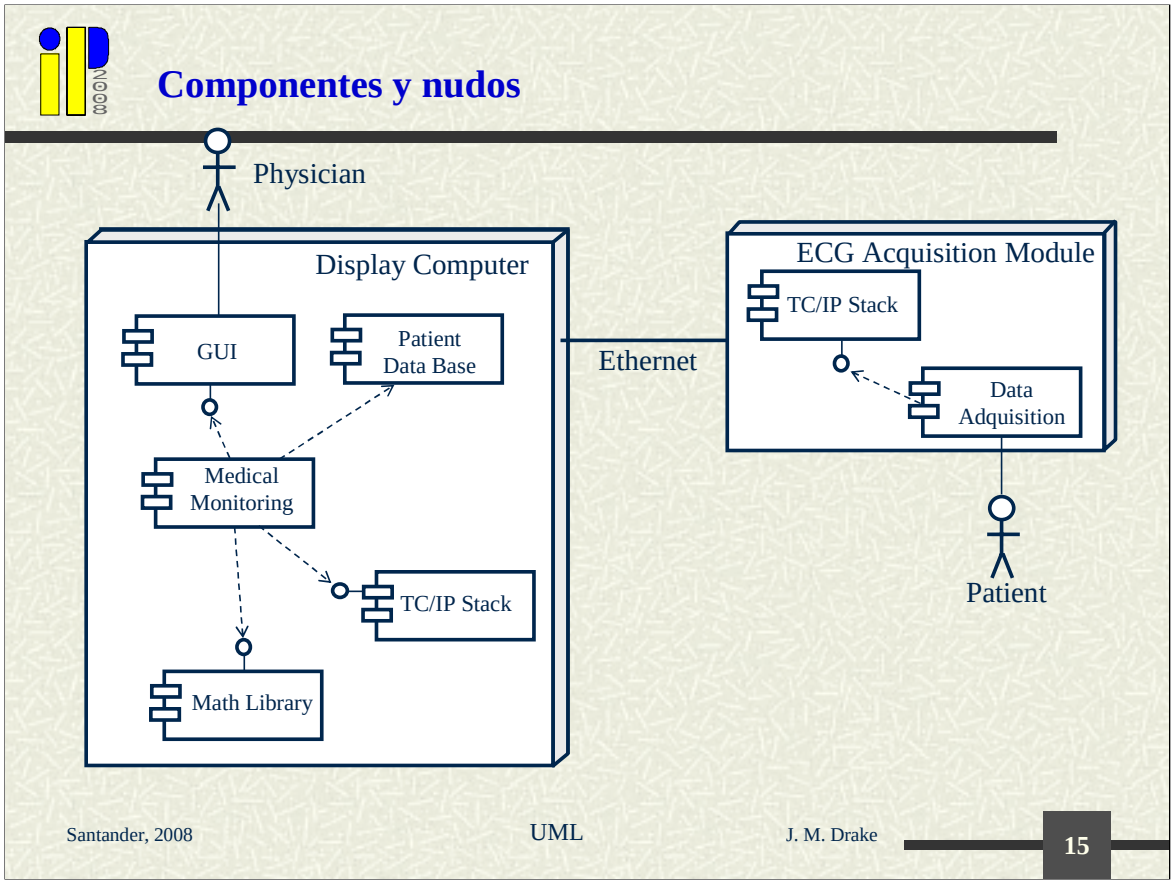


Notas:

Los subsistemas se representan mediante iconos de clases con el estereotipo <<subsystem>>.

Los elementos que incluye un subsistema pueden incorporarse mediante superposición (como en la figura) o a través del campo de atributos y haciendo referencia a clases definidas en el modelo, mediante asociaciones (habitualmente de tipo composición) con otras clases definidas en la vista.

La funcionalidad que ofrece el subsistema puede modelarse a través del campo de operaciones que ofrece la clase, o también expresando que el subsistema realiza interfaces que son descritas externamente al subsistema.



Notas:

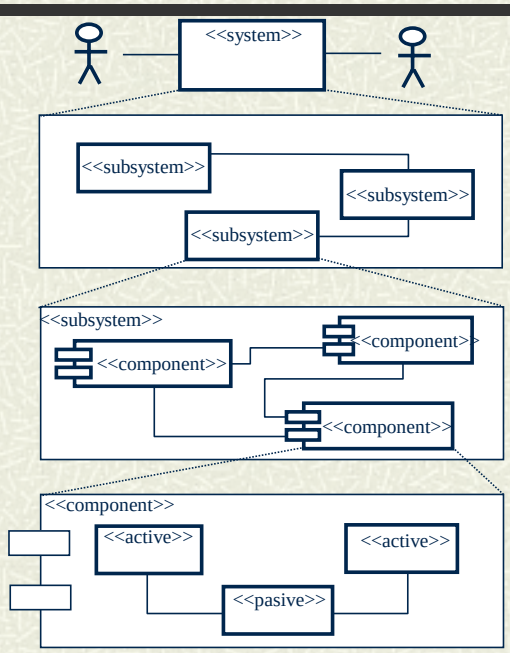
Los componentes se representan en UML a través de clases con el estereotipo <<component>> o a través de un icono específico (ver figura).

El nudo se representa mediante un icono de tres dimensiones.

En la figura se muestran también actores, que son elementos físicos que interaccionan con el sistema que se desarrolla pero que están fuera del ámbito de diseño.

A menudo los componentes, subsistemas y nudos se representan conjuntamente en diagramas de despliegue

Jerarquía de elementos estructurales



Notas:

Aunque los conceptos de sistema, subsistema, componentes tienen una semántica bien definida y suele estar claro su uso, en UML no está definida su jerarquía relativa, ni su granularidad.

En este curso seguiremos el criterio (no requerido en UML) de que los sistemas se componen de subsistemas (con los niveles de especificación que se necesiten), los subsistemas se componen de componentes (si el concepto de componente se considera necesario) y por último, unos y otros se describen en función de clases, interfaces y objetos.

Modelado del comportamientos

- Son recursos ofrecidos por UML para describir el comportamiento dinámico de los elementos estructurales (objetos, clases, sistemas, subsistemas y componentes) cuando el sistema se ejecuta.
  - **Diagramas de estados:** Se utiliza para describir la evolución interna o el efecto dinámico de una clase o de un método. Describe los estados del objeto durante su ciclo de vida junto con los eventos que hacen que el estado cambie.
  - **Diagrama de actividad**: Es un diagrama de flujo concebido para representar el efecto de un método.
  - **Diagrama de secuencias**: Describe la dinámica de interacción entre diferentes objetos, formulados como secuencias de eventos organizados en el tiempo.
  - **Diagrama de colaboración**: Describe la dinámica de interacción entre diferentes objetos, formulados en base a la organización estructural de los elementos y los mensajes que intercambian

Notas:

UML ofrece muchos recursos para expresar la funcionalidad de los elementos estructurales. Estos recursos pueden utilizarse bien para especificar la funcionalidad, o bien para describir detalladamente esa funcionalidad. En el primer caso se puede considerar como una herramienta de ayuda al análisis o diseño, en el segundo caso, puede llegar incluso a constituir un lenguaje de programación, del que se puede generar automáticamente el código.

Elementos para la descripción de elementos estructurales y su funcionalidad:

•Diagrama de estado: sirve para describir un elemento estructural a través de una máquina de estados finitos.

•Diagrama de actividad: es un diagrama de estado con una semántica específica formulado como forma de expresar algoritmos complejos a través de diagramas de flujo.

Elementos para la descripción de mecanismos de operación de interacción entre grupos de objetos del sistema:

•Diagrama de secuencias: Permite representar a través de diagramas de transferencia de mensajes entre objetos organizados en el tiempo cualquier escenario de intercolaboración entre ellos.

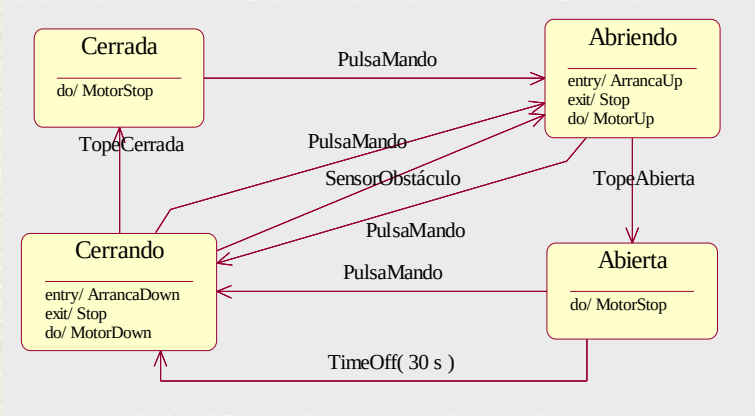
•Diagrama de colaboración: Permite representar a través de diagramas gráficos los escenarios de colaboración que se establecen entre grupos de objetos.

Ejemplo de diagrama de estado: Puerta de garaje

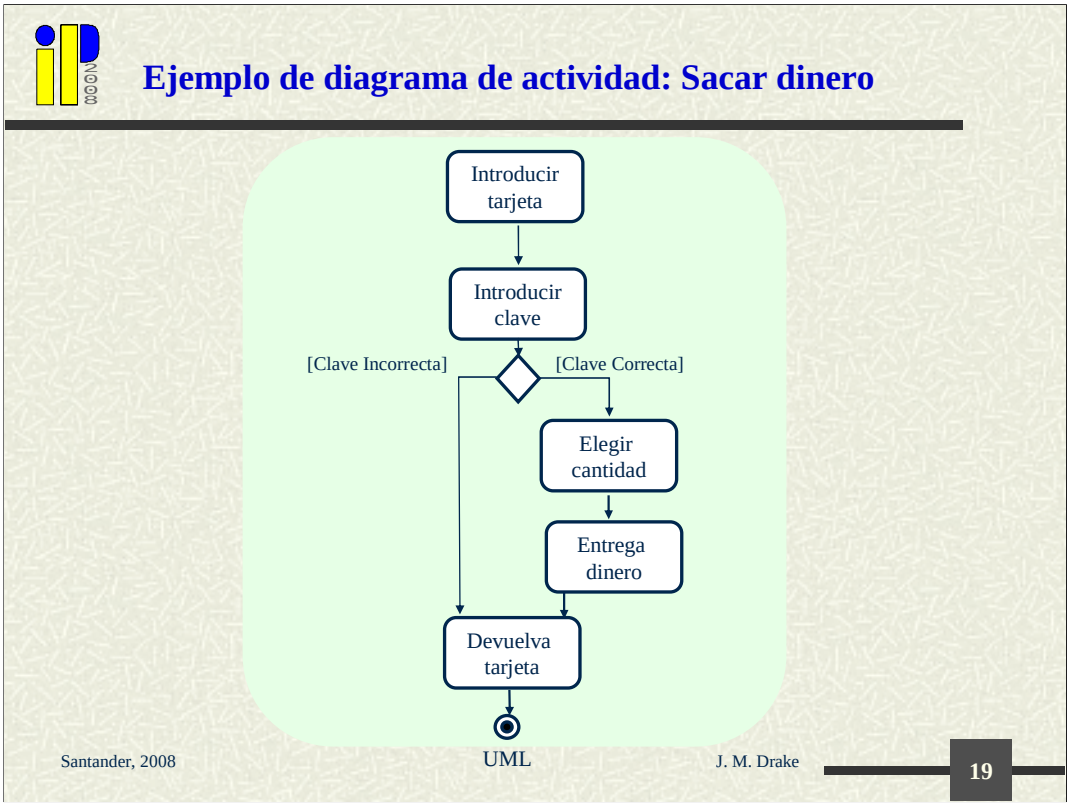
Eventos externos:
- PulsaMando
- TopeAbierto
- TopeCerrado
- SensorObstáculo

Acciones:
- ArrancaUP
- ArrancaDown
- Stop

Actividades:
- MotorUP
- MotorDown
- MotorStop



Notas:



Notas:

Actividad: Tarea que es realizada dentro de procedimiento o caso de uso que se describe.

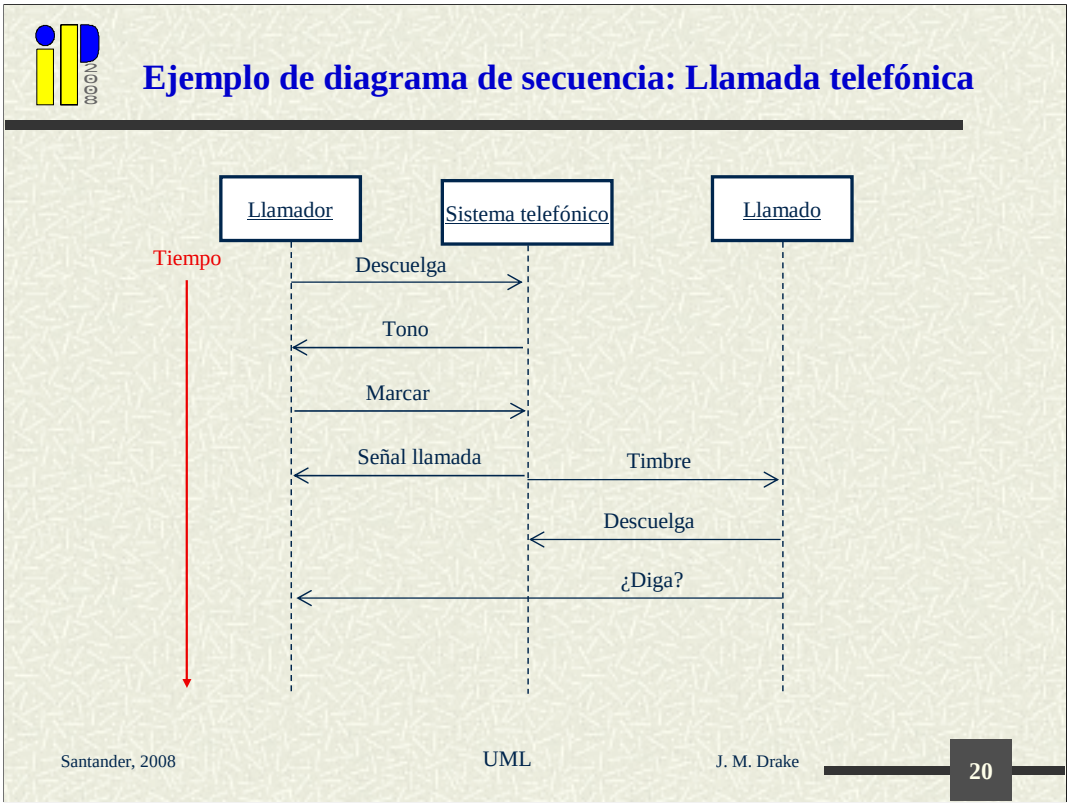
Estado: Etiqueta que hace referencia a un estado de ejecución del proceso descrito por el diagrama.

Espera de evento: El flujo de control se suspende hasta que se genera un evento o mensaje.

Emisión de un evento: Se genera un evento o mensaje que puede ser un resultado o una forma de interacción con otras secciones de la tarea.

Bifurcación condicionada (Branching): El flujo de control pasa a una u otra rama del diagrama, en función de que satisfaga o no una cierta condición.

Convergencia (Merge): Varia líneas alternativas se encauzan hacia una secuencia de actividades.



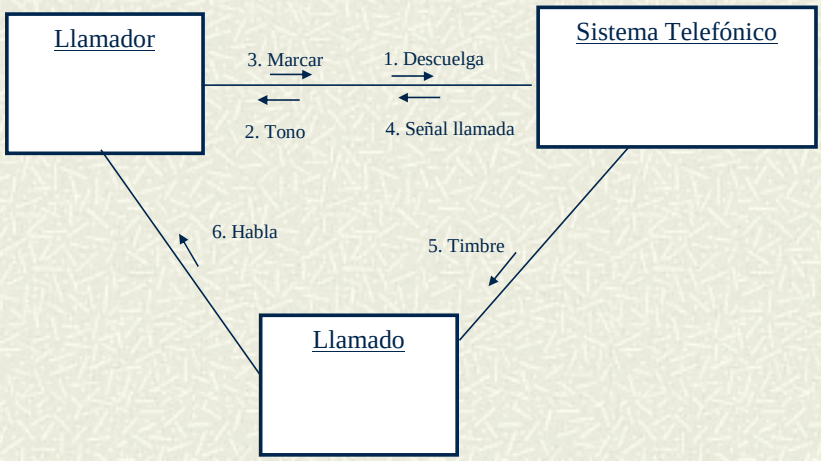
Notas:

En la figura se muestra un ejemplo de diagrama de secuencias que describe el proceso de comunicación de dos personas (Llamador y llamado) a través de una línea telefónica.

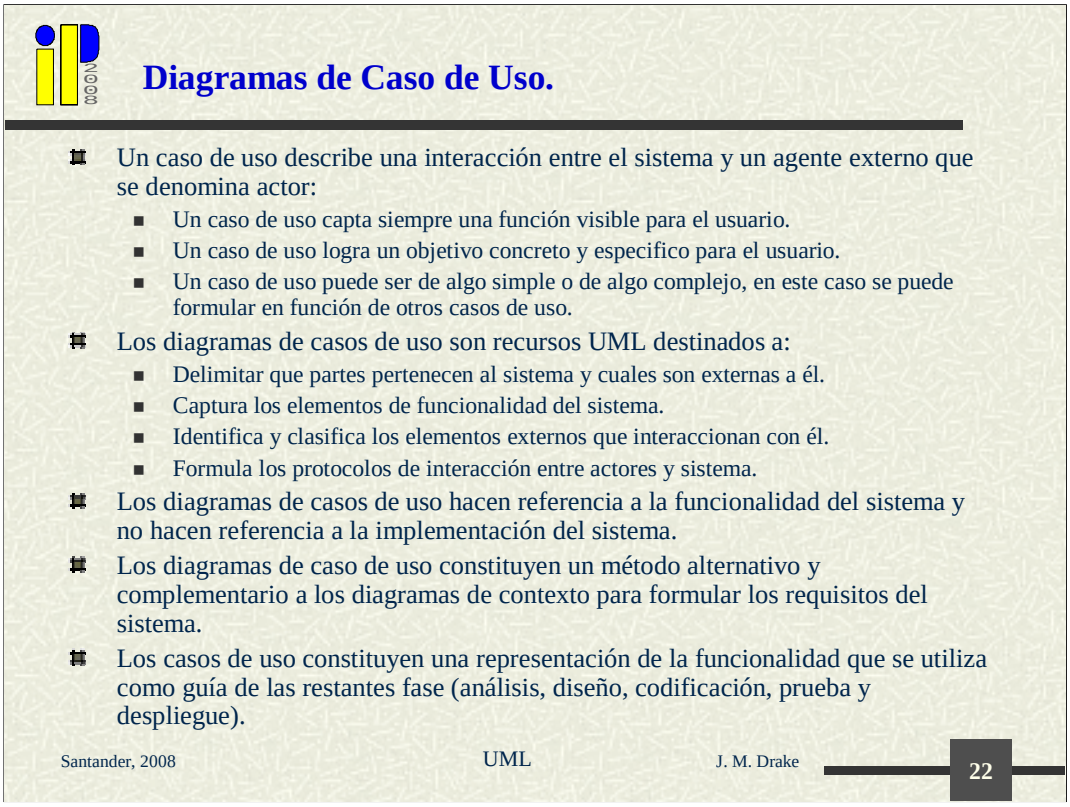
La interacción entre objetos se conciben como mensajes que se intercambian entre objetos.

El diagrama de secuencias no describe la totalidad de las posibilidades que se pueden plantear, sino que únicamente describe una secuencia posible. Por ejemplo, en este caso no se contempla la posibilidad de que el objeto Llamado esté comunicando, y en consecuencia falle la conexión. Esta situación diferente debería formularse con otro diagrama de secuencia.

Llamada telefónica con diagrama de colaboración



Notas:



Notas:

Los diagramas de casos de uso tienen son un método alternativo y complementario a los diagramas de contexto como medio de especificar los requisitos de una aplicación software.

Jacobson creó la idea de los casos de uso al observar que, a pesar del gran número de ejecuciones potenciales, muchas aplicaciones están concebidas en términos de un número relativamente pequeño de interacciones típicas.

Muestran los tipos básicos de interacción entre el sistema y los elementos del entorno que operan con él. Los diagramas de casos de uso proporcionan un recurso alternativo bien para verificar el diagrama de contexto o de ayuda para construirlo.

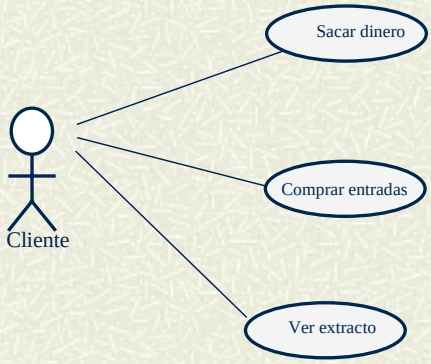
Los casos de uso capturan una vista general de la funcionalidad del sistema con un método muy adecuado para ser interpretado por personas no técnicas como son los usuarios y los expertos de dominio. Suelen interpretarse como una guía de los escenarios de uso del sistema sobre los que se especifican los requerimiento del sistema.

Los casos de uso son también un método de descomposición de la funcionalidad del sistema en elemento de funcionalidad más básica, ya que UML proporcionan una semántica para expresar los casos de usos mas generales en función de casos de usos mas simples.

Los casos de uso son contenedores de la información que describen los requerimientos del sistema. Estos se pueden formular con diferentes niveles de detalles, mas cualitativos para el uso de los expertos de dominio y mas detallado y formales para el uso de los analistas de la aplicación.

La mayoría de los estándares de documentación establecidos (incluida IEEE 830-1993)son anteriores a la definición del caso de uso y en consecuencia no los tienen en cuenta. Es habitual tener que extender su formulación para que tengan cabida.

Ejemplo de diagrama de casos de uso: Cajero



Notas: