TEMA 2. UML

ANÁLISIS Y DISEÑO DE SISTEMAS DE INFORMACIÓN

Universidad Francisco de Vitoria
UFV Madrid

ÍNDICE

Tema 2: Introducción al análisis y diseño de sistemas de información

1. Introducción a UML.
2. Historia de UML.
3. ¿Para qué se utiliza?
4. Diagramas fundamentales UML.

1. INTRODUCCIÓN A UML

- El **lenguaje de modelado unificado (UML)** es una consolidación de las mejores prácticas que se han establecido a lo largo de los años en el análisis y diseño de proyectos software.
- UML **nos permite representar aspectos muy variados de un sistema de software**.
- Por ejemplo, requisitos, estructuras de datos, flujos de datos, y flujos de información dentro de un único marco que utiliza conceptos.
- Vamos a explicar por qué el modelado es una parte indispensable del software.

1. INTRODUCCIÓN A UML

- Imaginad que un cliente desea desarrollar un sistema de software.
- Uno de los primeros desafíos a los que nos enfrentamos es aclarar **lo que el cliente realmente quiere** y si **hemos entendido los requisitos** que cubrirán las necesidades del cliente en el sistema futuro.
  - Paso fundamental para que el proyecto tenga éxito.
- Pregunta: **¿cómo nos comunicamos con nuestro cliente?**

1. INTRODUCCIÓN A UML

- El **lenguaje natural no sería una opción** del todo apropiada, porque es impreciso y ambiguo.
  - Pueden surgir **malos entendidos** fácilmente y que se hablen de propósitos contradictorios conllevando graves consecuencias.
  - Ejemplos: informático o desarrollador VS gerente comercial.
- Lo ideal es poder crear un **buen modelo para el software** solicitado destacando los aspectos más importantes de forma clara.

1. INTRODUCCIÓN A UML

Lenguajes de modelado

- Se desarrollaron precisamente para tales escenarios y demostrar reglas claramente definidas mediante una descripción estructurada de un sistema.
- Estos lenguajes pueden ser:
  - **Textuales** (lenguaje de programación como Java)
  - **Visuales** (lenguaje que proporcione símbolos para transmisores, diodos, etc.)
- Pueden diseñar un dominio específico o para fines generales.
  - Lo más común es que se usen para fines específicos.

Modelado orientado a objetos

- Forma de modelado que obedece al paradigma de orientación a objetos.
- Conviene recordar:
  - Conceptos de la asignatura Programación Orientada a Objetos (POO).

Modelos

- Los modelos nos permiten describir sistemas de manera eficiente y concreta. Un **sistema** es un todo integrado formado por componentes que están relacionados entre sí.
- Han de ser percibidos como una **realidad única** orientado a un propósito específico.

2. HISTORIA DE UML

- En la década de 1980 y principios de la de 1990, surgieron diferentes tipo de modelado.
  - Fue el resultado de cubrir la necesidad de hacer frente a los **problemas de compatibilidad entre los modelos presentados** por los distintos socios de proyectos.
- Para poner fin a esto, en 1996 el **OMG (Object Management Group)** elabora la **estandarización** más importante para el desarrollo de software orientado a objetos. Será un estándar de **modelado uniforme**.
- El año anterior, 1995, **Grady Booch, Ivar Jacobson y James Rumbaugh** habían combinado sus ideas y enfoques en la OOPSLA (Programación Orientada a Objetos, Sistemas, Lenguajes y Aplicaciones). Pusieron en práctica tres objetivos.

2. HISTORIA DE UML

4 objetivos

- Uso de **conceptos orientados a objetos** para representar **sistemas completos** en vez de aplicarlo sólo a parte de ellos.
- Establecimiento de una **relación explícita** entre conceptos de **modelado** y **código de programa ejecutable**.
- Consideración de factores de **escalado** que son inherentes a sistemas críticos y/o complejos.
- Creación de una notación para diseñar modelos que pueda ser **procesado por máquinas** y que también puede ser leído por **humanos**.

2. HISTORIA DE UML

- El resultado de sus esfuerzos fue el lenguaje de modelado unificado (UML) Modelado unificado Lenguaje (UML) en 1997 como respuesta a la necesidad de la OMG.
- Uno de los principales objetivos era llevar a cabo una **especificación consistente del núcleo del lenguaje de UML** documentado en un **metamodelo**. El metamodelo determina **qué elementos** del modelo proporciona el lenguaje UML y **cómo usarlos** correctamente.
- Para formular restricciones que tengan que cumplir elementos del modelo, el Object Constraint Language (**OCL**) introdujo la **lógica de predicados**.
- En versiones posteriores, se introdujo la definición que hacía realidad el **intercambio de metadatos XML** entre modelos.

3. ¿PARA QUÉ SE UTILIZA?

- UML **no está vinculado a una herramienta de desarrollo específica**, un lenguaje de programación, o una plataforma de destino concreta en la que se desarrolla el sistema.
- UML **tampoco ofrece un proceso de desarrollo de software**. Separa el lenguaje de programación del modelado.
- Se puede definir en un **proyecto específico** pero también es importante saber que los conceptos de lenguaje de UML favorecen el **proceso iterativo o incremental** de desarrollo.
- UML se puede utilizar de forma coherente **en todo el desarrollo de software**. No es necesario traducir un modelo a otro lenguaje de modelado.
  - Esto permite un **desarrollo de software iterativo e incremental**.
  - UML es adecuado para **varias áreas de aplicación con diferentes requisitos** en cuanto a complejidad, volumen de datos, tiempo real, etc.

3. ¿PARA QUÉ SE UTILIZA?

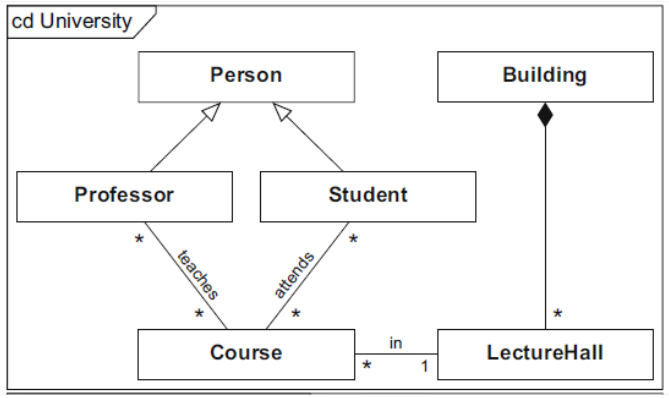
- Por tanto, UML es una notación para modelado gráfico que se utiliza para representar partes de un sistema de software (diseño, comportamiento, arquitectura, etc.).
  - Por ejemplo: hay que conocer muy bien dos tipos de flechas. Una que acaba en triángulo (herencia) y la que no lo hace (asociación). También conocer que una clase es un cuadrado y que si lleva un nombre subrayado y ":" delante de él es un objeto.
- De los 14 diagramas que se definen en UML versión 2.5.1, los más utilizados son: **diagramas de clases y diagramas de secuencia**.
- UML es fundamental hoy en día para **definir un diseño de software**, o para **interpretar cómo se va a comportar el sistema**. Se utiliza UML para comprenderlo y para que otros lo comprendan y sobre todo para **evitar que se malinterprete**.

4. DIAGRAMAS FUNDAMENTALES UML

- Definición de la última versión UML 2.5.1: https://www.omg.org/spec/UML/2.5.1/
- En UML, un **modelo** se representa gráficamente en forma de **diagramas**.
- El diagrama **proporciona una vista** de parte de la realidad descrita por el modelo.
- Existen diagramas que indican **qué usuarios** han de utilizar cierta funcionalidad y diagramas que muestran la **estructura del sistema** pero sin especificar el desarrollo concreto.
- También hay diagramas que expresan procesos que son soportados por el sistema y especificaciones de lo que no se debe hacer (procesos que no han de seguirse).

4. DIAGRAMAS FUNDAMENTALES UML

- **Taxonomía de los 14 diagramas de UML**. Se diferencian diagramas de **estructura** y diagramas de **comportamiento**.
- Los diagramas de **comportamiento** engloban los diagramas de **interacción** que se dividen en cuatro diagramas.
- Un diagrama generalmente se encuentra enmarcado por un rectángulo con un pentágono en la esquina superior izquierda donde se incluye el nombre del diagrama.



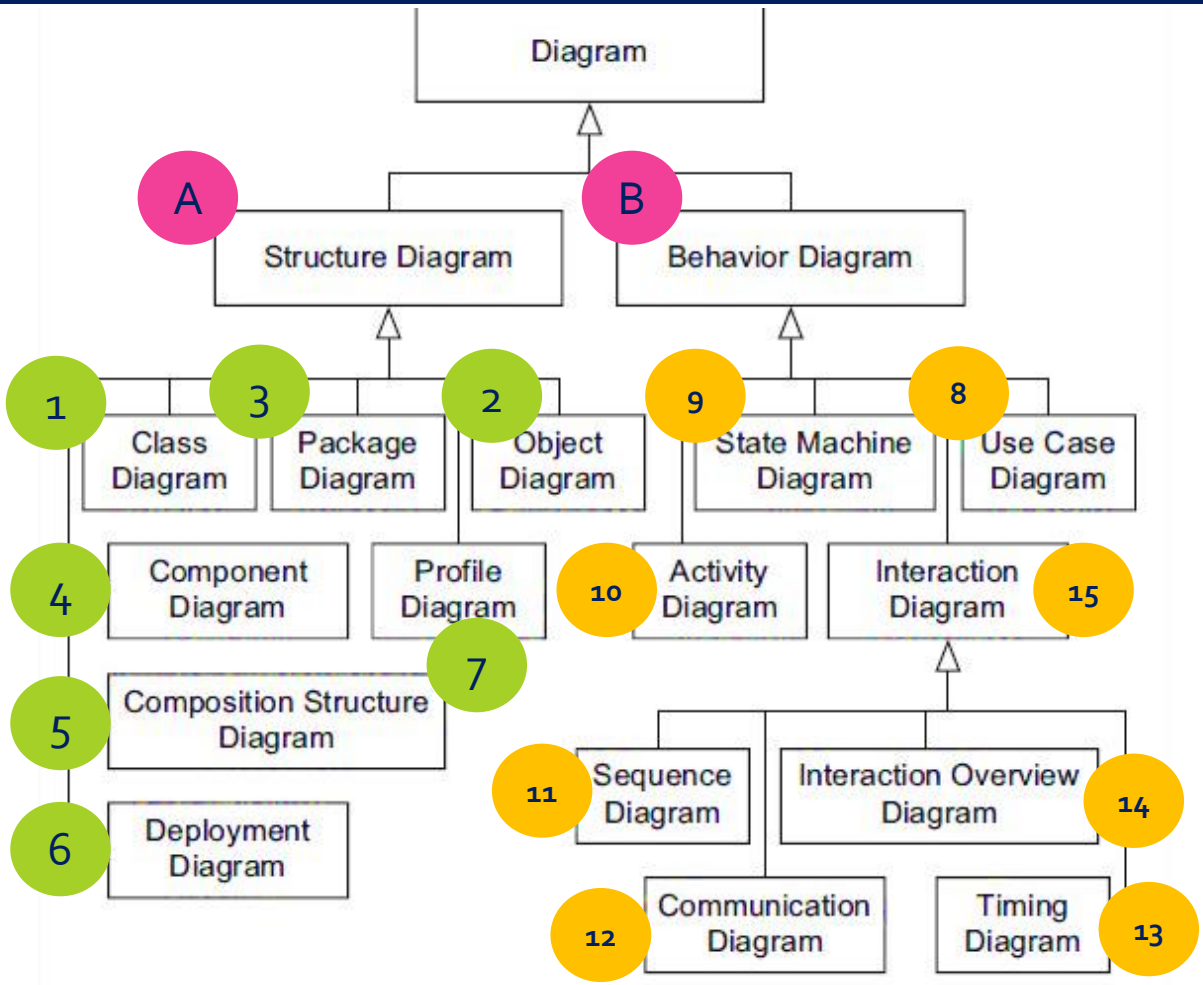
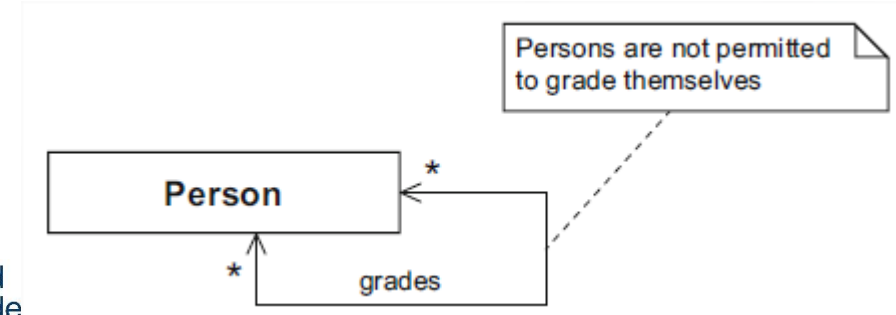


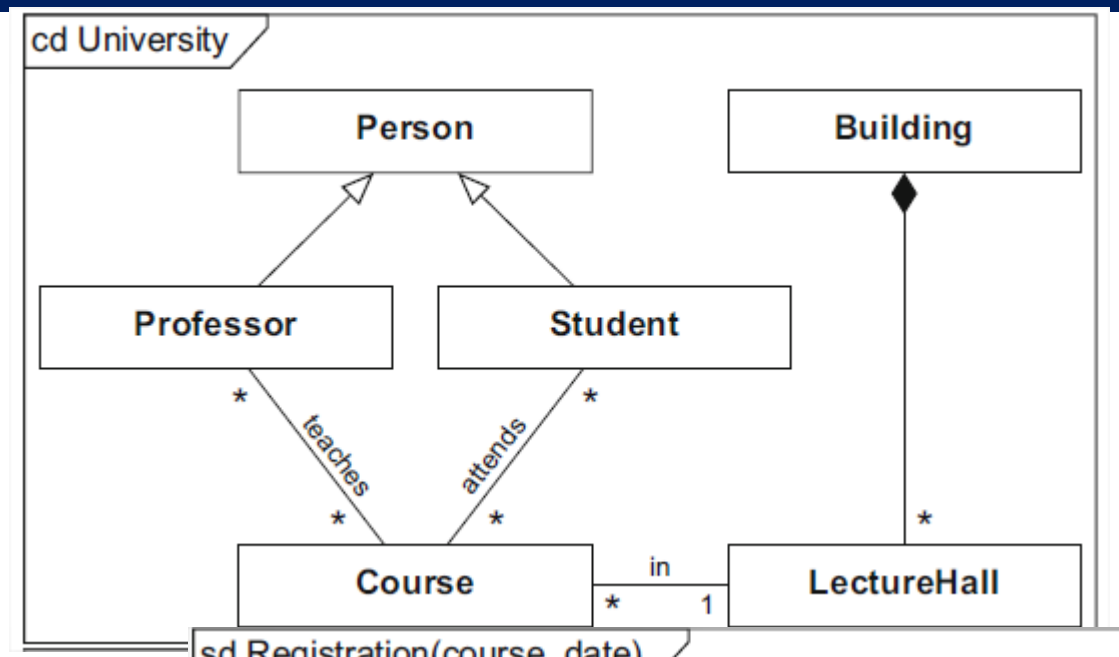
Diagrama procedente del libro UML @ Classroom
Relación de diagramas UML

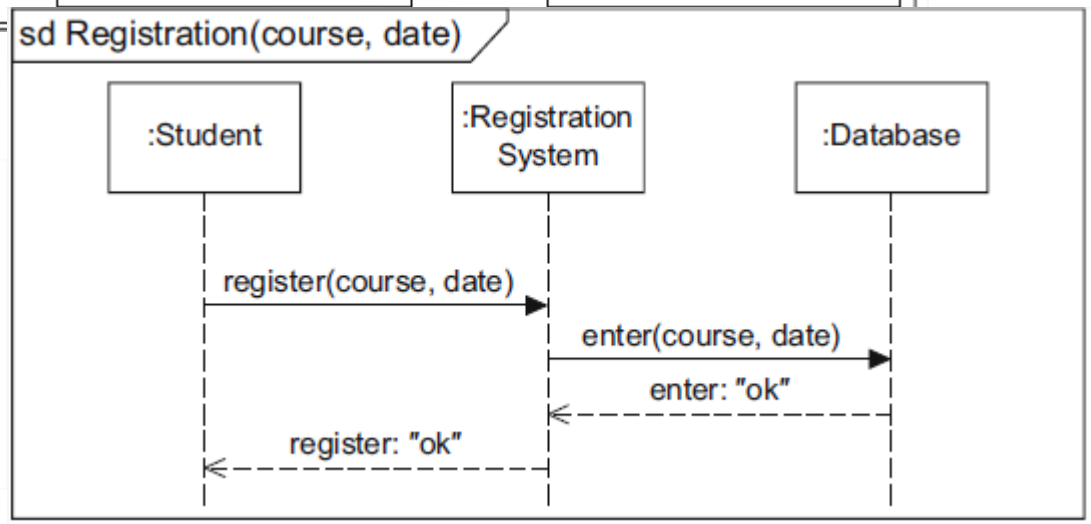
Universidad Francisco de Vitoria UFV Madrid

4. DIAGRAMAS FUNDAMENTALES UML

- Un concepto que puede aparecer en todos los diagramas es la **nota**
- Una nota puede contener **cualquier forma de expresión** que especifique el **diagrama** y sus **elementos**. Por ejemplo, en lenguaje natural o en lenguaje OCL (Object Constraint Language) – expresa restricciones sobre objetos o condiciones que han de mantenerse a lo largo de todo el proceso. Ej.: reglas de negocio.
- También se pueden incluir **comentarios** mediante notas al resto de los elementos del modelo.







Universidad Francisco de UFV Madrid

4. DIAGRAMAS FUNDAMENTALES UML

Diagramas de estructura

A

- UML ofrece siete tipos de diagramas para modelar la estructura de un sistema desde distintas perspectivas.
- El comportamiento dinámico de los elementos, los cambios en el tiempo, no se tienen en cuenta en estos diagramas.

1

- **Diagramas de clases**
- Al igual que los conceptos del diagrama de objetos, los conceptos del *diagrama de clases se* originan a partir del modelo conceptual de datos. Los conceptos especifican las estructuras de datos y las estructuras de objetos del sistema.

2

- **Diagramas de objetos.**
- Basado en las definiciones del diagrama de clases, muestra una foto del estado del sistema en una ejecución concreta de la acción.

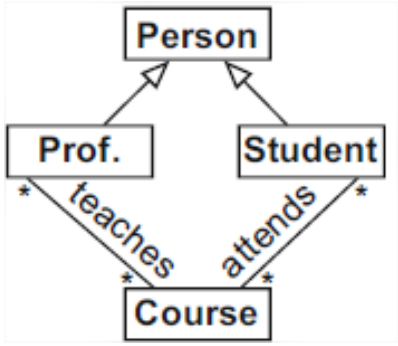


Diagrama de clases

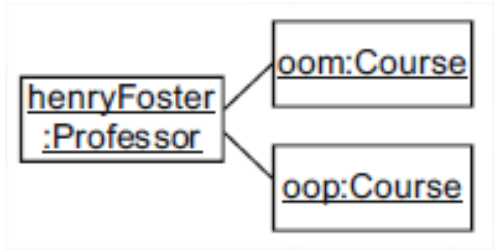


Diagrama de objetos

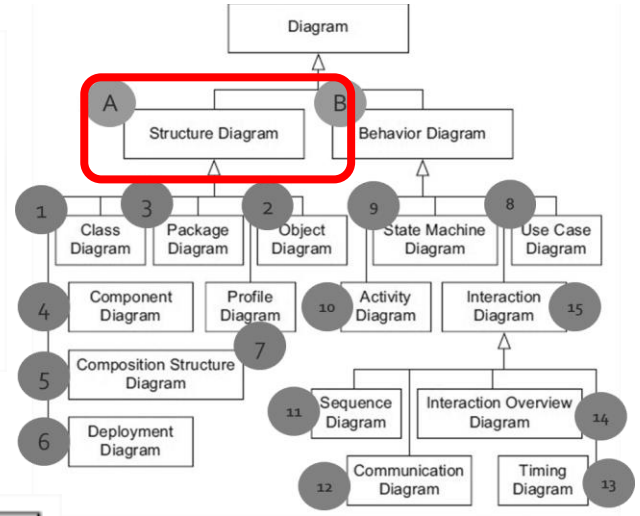
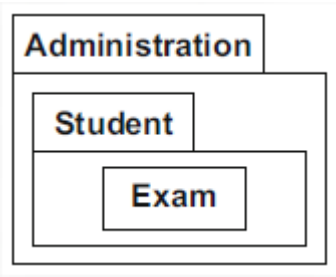


Diagrama procedente del libro UML @ Classroom

4. DIAGRAMAS FUNDAMENTALES UML

3

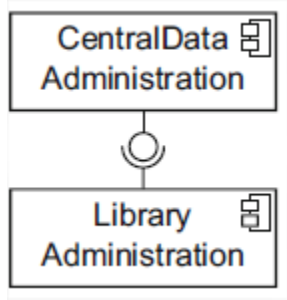
- **Diagramas del paquete**
- Agrupa los diagramas o elementos del modelo según propiedades comunes, como grandes funcionalidades.
- Ejemplo: un sistema de administración de la universidad puede contener paquetes de información de estudiantes, y sus exámenes.



Diagramas de paquete

4

- **Diagramas de componentes**
- Un componente es una unidad ejecutable que proporciona o utiliza servicios de otros componentes.
- UML no prescribe ninguna separación estricta entre conceptos orientados a objetos y orientados a componentes.
- Cuando se define un componente, se pueden diseñar dos vistas explícitamente: la vista externa (vista de caja negra), que representa la especificación del componente, y la vista interna (vista de caja blanca), que define su desarrollo.

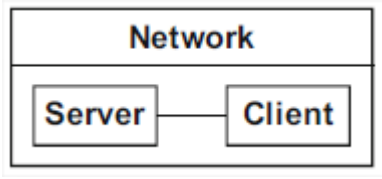


Diagramas de componentes

Diagrama procedente del libro UML @ Classroom

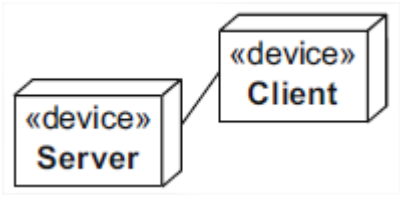
4. DIAGRAMAS FUNDAMENTALES UML

5
- **Diagramas de estructura compuesta**
- Agrupa los diagramas o elementos del modelo según propiedades comunes, como grandes funcionalidades.
- Ejemplo: un sistema de administración de la universidad puede contener paquetes de información de estudiantes, y sus exámenes.



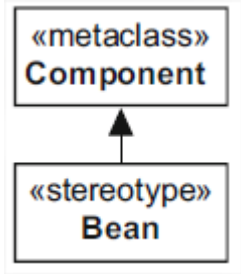
Diagramas de estructura compuesta

6
- **Diagramas de despliege**
- Un componente es una unidad ejecutable que proporciona o utiliza servicios de otros componentes.
- UML no prescribe ninguna separación estricta entre conceptos orientados a objetos y orientados a componentes.



Diagramas de despliegue

7
- **Diagramas perfiles UML**
- Usando *perfiles* , puede extender UML para introducir condiciones específicas de dominio.
- El núcleo de la definición del lenguaje de UML, el metamodelo, permanece sin cambios. Así se podrá reutilizar el modelado sin tener que hacer demasiados ajustes. Ejemplo: utilizar perfiles para introducir el concepto de Java Enterprise Beans.



Perfiles UML

Diagrama procedente del libro UML @ Classroom

4. DIAGRAMAS FUNDAMENTALES UML

B

Diagramas de comportamiento

- El comportamiento se refiere a las consecuencias directas de una acción en al menos un objeto. Afecta cómo cambian los estados de los objetos con el tiempo. Se puede especificar mediante las acciones de un objeto o con el resultado de las interacciones entre varios objetos.
- **Diagramas de casos de uso**

8

- Sirve para definir los requisitos que ha de cumplir el sistema. Define los usuarios y las funcionalidades que han de llevar a cabo en el sistema sin abordar el detalle del desarrollo. El caso de uso representa la acción que debe ejecutar el sistema.
- **Diagrama de máquina de estados**
- Dentro de su ciclo de vida, los objetos pasan por diferentes estados.

9

- Ejemplo: una persona tiene como estado "desconectado" antes de entrar en una página web, y éste cambia cuando el usuario inicia la sesión.
- Este diagrama describe el comportamiento de un objeto en forma de estados posibles y cómo desencadenan desde la ejecución de distintos eventos.

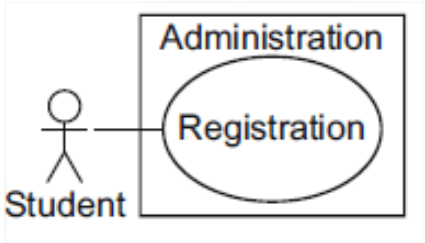
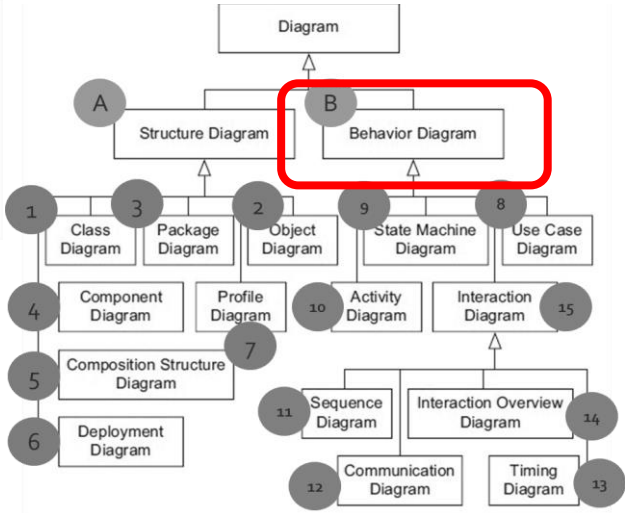


Diagrama de casos de uso



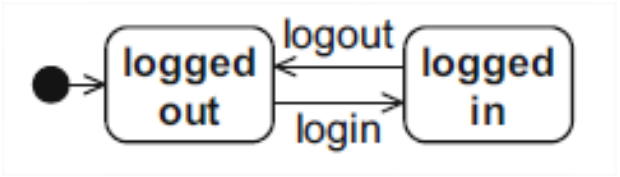


Diagrama de máquina de estados

Diagrama procedente del libro UML @ Classroom

4. DIAGRAMAS FUNDAMENTALES UML

10 **Diagramas de actividad**

- Puede modelar tanto procesos de negocio como procesos de software. Puede mostrar qué acciones son necesarias para un alumno a la hora de participar en una conferencia o en una tarea. Muestran el flujo de control y el fujo de datos para coordinar las acciones que componen una actividad. Es decir, el proceso en sí.

11 **Diagramas de secuencia**

- Describe las interacciones que son necesarias entre objetos para lograr realizar una tarea específica. La atención se centra en el orden cronológico de los mensajes intercambiados entre los socios de interacción. Permite controlar los mensajes, su tiempo y la modelar acciones complejas.

12 **Diagramas de comunicación**

- Describe la comunicación entre varios objetos, de forma similar al diagrama de secuencia. Aquí nos centramos en las relaciones entre los agentes de la interacción más que en el orden cronológico del intercambio de mensajes. El mensaje muestra claramente los actores que se comunican. No se pueden representar estructuras complejas de control.

13 **Diagrama de tiempos**

- *muestra* explícitamente los cambios de estado de la interacción entre los agentes que se debe a un evento en el tiempo o como resultado de un intercambio de mensajes.

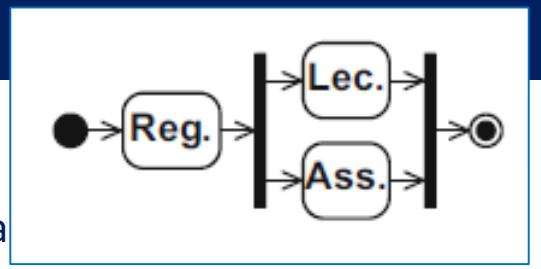


Diagrama de actividad

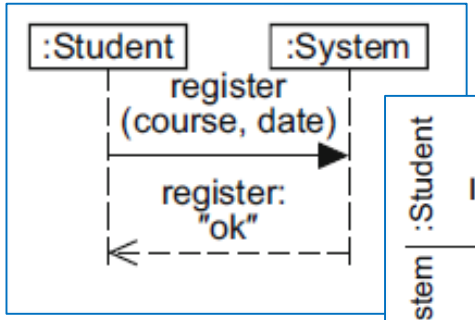


Diagrama de secuencia

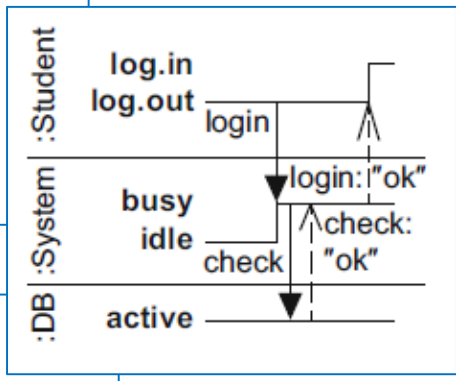


Diagrama de tiempos

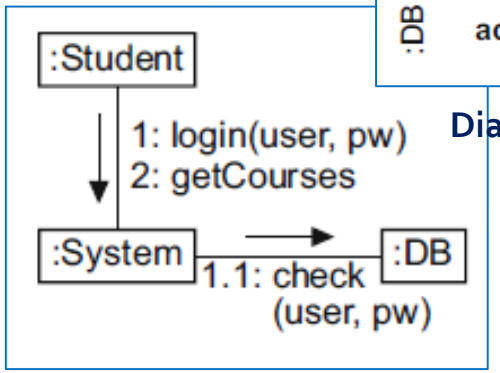


Diagrama de comunicación

4. DIAGRAMAS FUNDAMENTALES UML

14

- **Diagrama de interacción**
  - Diseña la conexión entre diferentes procesos de interacción estableciendo diagramas de interacción individuales. Vendría a ser: diagramas de secuencia, diagramas de comunicación, diagramas de tiempo y otros diagramas de descripción general de la interacción en una secuencia temporal.
  - También especifica las condiciones bajo las cuales los procesos de interacción pueden tener lugar.
  - Para modelar el flujo de control se utilizan los diagramas de actividad.
  - Ejemplo: un usuario de la universidad quiere iniciar sesión (es decir, generar una interacción con el sistema) antes de realizar cualquier otra actividad.

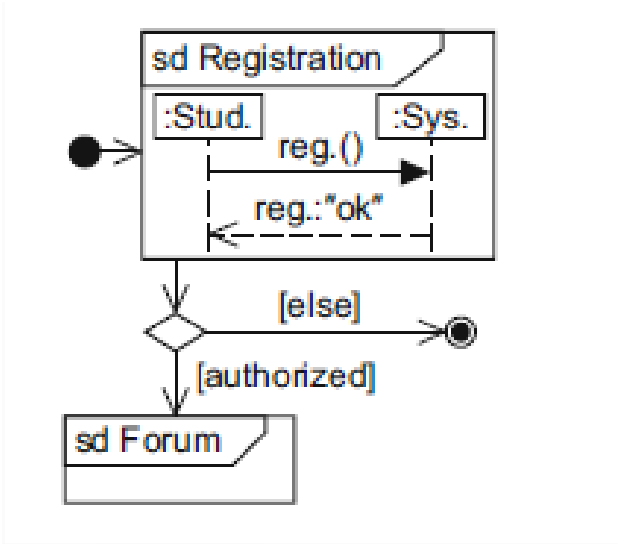


Diagrama de actividad