

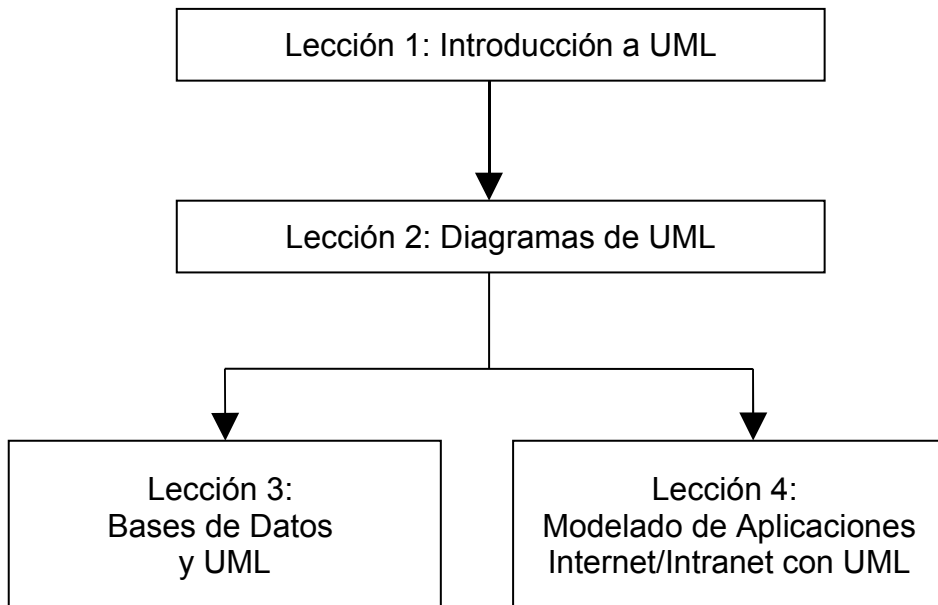
UNIDAD DIDÁCTICA: UML (UNIFIED MODELING LANGUAGE)



INDICE

<u>ESQUEMAS DE CONTENIDO.....</u>	<u>2</u>
<u>INTRODUCCIÓN DE LA UNIDAD</u>	<u>3</u>
<u>OBJETIVOS</u>	<u>5</u>
<u>LECCIONES.....</u>	<u>6</u>
<u>LECCIÓN 1. MODELADO DE SOFTWARE CON UML.....</u>	<u>6</u>
<u>LECCIÓN 2. DIAGRAMAS UML.....</u>	<u>6</u>
<u>LECCIÓN 3. INTERACCIÓN ENTRE LAS BASES DE DATOS RELACIONALES Y UML.....</u>	<u>6</u>
<u>LECCIÓN 4. MODELADO DE APLICACIONES INTERNET/INTRANET CON UML.....</u>	<u>6</u>
<u>CONCLUSIONES DE LA UNIDAD.....</u>	<u>7</u>
<u>GLOSARIO DE TÉRMINOS.....</u>	<u>8</u>
<u>BIBLIOGRAFÍA.....</u>	<u>14</u>

ESQUEMAS DE CONTENIDO



INTRODUCCIÓN DE LA UNIDAD

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh.

Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

En la lección 1 veremos las principales características de la orientación a objetos en general y de UML en particular, y aprenderemos qué es y qué no es UML. También estudiaremos las posibles “vistas” de un sistema software.

En la lección 2 veremos cómo se representan gráficamente en UML los conceptos principales de la orientación a objetos. Así estudiaremos todos los tipos de diagramas que ofrece UML, tanto los de estructura o estáticos como los dinámicos o de comportamiento.

A partir de esta lección podremos estudiar la lección 3 y luego la 4, o viceversa, aunque se recomienda seguir el orden expresado en la unidad didáctica.

En la lección 3 veremos cómo interaccionan las bases de datos relacionales con las aplicaciones orientadas a objetos, y presentaremos varias posibilidades para modelar este tipo de componentes con UML.

Por último, en la lección 4 veremos cómo podemos modelar las aplicaciones Internet/Intranet, utilizando para ello extensiones del lenguaje de modelado unificado.

A lo largo de la unidad didáctica se irán presentando ejemplos que clarifiquen los conceptos presentados. Así mismo, en los Ejercicios se profundizará en ejemplos de diagramas y modelado de bases de datos y aplicaciones Internet/Intranet. Los casos prácticos que emplearemos son similares a los que nos encontramos en proyectos reales, por lo que podremos aplicar los conocimientos adquiridos inmediatamente.

OBJETIVOS

- Entender el propósito de UML
- Aprender cómo UML cubre los elementos de orientación a objetos
- Conocer los diagramas UML
- Aprender cómo UML ayuda en el desarrollo software
- Entender la relación entre el modelado de bases de datos relacionales y UML
- Aprender a modelar aplicaciones Internet/Intranet con UML

LECCIONES

Lección 1. Modelado de software con UML

[Ver fichero *Leccion1.doc*]

Lección 2. Diagramas UML

[Ver fichero *Leccion2.doc*]

Lección 3. Interacción entre las bases de datos relacionales y UML

[Ver fichero *Leccion3.doc*]

Lección 4. Modelado de aplicaciones Internet/Intranet con UML

[Ver fichero *Leccion4.doc*]

CONCLUSIONES DE LA UNIDAD

UML 2.0 es la mayor revisión que se le ha hecho a UML desde la versión 1.0. El modelo conceptual ha sido reestructurado completamente y nuevos diagramas han sido incorporados. Los diagramas tradicionales también han sido mejorados.

La nueva versión permitirá a los fabricantes de herramientas CASE proporcionar a los analistas, arquitectos y desarrolladores herramientas cada vez más potentes, que les permitan aprovechar mejor los modelos y como consecuencia generar una mayor cantidad de código reduciendo significativamente el ciclo de desarrollo de sus aplicaciones.

A lo largo de las lecciones de esta Unidad Didáctica hemos estudiado las características de UML y la notación y función de sus principales diagramas. Además, hemos trabajado para aprender a poner en práctica los conocimientos adquiridos para modelar dos situaciones que se presentan habitualmente en los proyectos software: el modelado de bases de datos relacionales y el modelado de aplicaciones Internet/Intranet.

UML seguirá evolucionando, después de haberse convertido en el estándar de facto por la comunidad internacional, y habiendo recibido el apoyo de las principales compañías de software, por lo que su continuidad está garantizada.

GLOSARIO DE TÉRMINOS

- Agregación: La agregación representa una relación *parte_de* entre objetos.
- Asociación: La asociación expresa una conexión bidireccional entre objetos.
- Atributo: Es cada una de las características de un objeto: identificador, descripción...
- Caso de uso: Corresponde a cada cosa que puede hacer un usuario dentro del modelo de datos. La identificación de estos casos de uso se hace con base en los requerimientos de la aplicación a desarrollar.
- Clase: Definición de atributos y métodos para un conjunto de objetos.
- Clase base / padre / madre / ascendiente / superclase: Clase que define un conjunto de características comunes a todas las clases derivadas de ella.
- Depuración: Hace referencia a métodos para refinar el código del programa que se está desarrollando, identificando y eliminando todos los posibles errores que éste tenga.
- Diagrama: Es una representación gráfica de una colección de elementos de modelado.
- Diagrama de interacción: Indica la secuencia de acciones que deben seguirse para realizar una tarea en el modelo computacional. Este tipo de diagrama puede indicarse de dos maneras: diagrama de secuencia, en el cual se muestra la secuencia lineal de acciones en determinado momento;

diagrama de colaboración, muestra la secuencia de acciones de modo no lineal, resaltando las relaciones y/o dependencias entre diferentes clases del modelo.

- **Diseño:** Proceso de convertir los requisitos de un sistema en una manera de resolver el problema con el objetivo de posibilitar una implementación que cumpla el coste, prestaciones y calidad deseados.
- **Encapsulación:** Habilidad de poner todos los datos y funcionalidades de un tipo de datos en una estructura controlada y abstracta, de forma que esté aislada del resto del sistema. Las clases permiten al programador la libertad de encapsular atributos y métodos en una única estructura.
- **Escenario:** Secuencia específica de acciones que ilustra un comportamiento.
- **Escenario de interacción:** Cada uno de los momentos de interacción que tiene el usuario con la aplicación (registrarse, escoger reto, etc.).
- **Especificación de requisitos:** Descripción establecida de lo que se pide a un sistema.
- **Estado:** Valores de los atributos y enlaces de un objeto en un momento dado.
- **Evento:** Suceso que ocurre instantáneamente en un punto del tiempo.
- **Generalización:** La Generalización consiste en factorizar las propiedades comunes de un conjunto de clases en una clase más general.

- Herencia. El término herencia significa que una clase hereda de forma automática las características de su superclase (la clase que está inmediatamente por encima de ella en la jerarquía).
- Herencia múltiple: Se presenta cuando una subclase tiene más de una superclase.
- Implementación: Etapa en el ciclo de desarrollo de software en la que un diseño se plasma en una forma ejecutable en un sistema informático.
- Instancia: Realización de una clase. Un objeto es una instancia de una clase.
- Interacción: Comportamiento que comprende un conjunto de mensajes que se intercambian entre un conjunto de objetos.
- Interfaz: Colección de operaciones que se utiliza para especificar un servicio de una clase o componente.
- Invariante de clase: Es todo aquello que debe cumplir siempre cada clase. Por ejemplo, si se definiera la "clase Persona" se tendría el siguiente invariante: "una Persona siempre tiene nombre, apellido y documento de identidad. No existen dos personas con el mismo documento de identidad".
- Jerarquía de clases: Cuando unas clases heredan de otras se forma una estructura arbórea que se denomina jerarquía. Se puede pensar una jerarquía de clases como un árbol de familia, en el que las clases derivadas son las hijas de las clases base (madres).
- Lenguaje orientado a objetos: Lenguaje que tiene la capacidad de definir clases y objetos y que permite la herencia, la encapsulación y el polimorfismo.

- Mensaje: Mecanismo por el cual los objetos se comunican. Es el término usado por un método cuando se aplica a un objeto. Los métodos definen los mensajes que pueden enviarse a una clase de objetos.
- Método: Corresponde a cada una de las funciones que puede llevar a cabo un objeto, por ejemplo crearse, destruirse, dar su identificación, etc.
- Mensaje: Es la unidad de comunicación entre objetos.
- Modelo: Un modelo captura una vista de un sistema del mundo real. Es una abstracción de dicho sistema, considerando un cierto propósito. Así, el modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.
- Modelo estático: En la notación UML, el modelo estático corresponde al diagrama donde se muestran todas las clases definidas para la aplicación, indicando para cada clase sus atributos y métodos, así como las relaciones que tiene con las demás clases.
- Modelo dinámico: Corresponde al conjunto de casos de uso de la aplicación.
- Objeto: En el enfoque Orientado a Objetos un objeto es cualquier cosa que puede ser identificada plenamente en el mundo, es decir que tiene unas características y comportamiento particulares.
- Paquete: Un paquete ofrece un mecanismo general para la organización de los modelos/subsistemas agrupando elementos de modelado.
- Persistencia: La persistencia de los objetos designa la capacidad de un objeto para trascender en el espacio/tiempo.

- Polimorfismo: Es una característica presente en la programación orientada a objetos mediante la cuál dos métodos con el mismo nombre de dos clases pueden tener diferente comportamiento.
- Portabilidad: Capacidad que tiene una aplicación de ejecutarse en diferentes plataformas de hardware y software.
- Prueba piloto: Prueba de la aplicación realizada con un grupo representativo de la población objetivo.
- Relación: Conexión semántica entre elementos.
- Rol: Comportamiento de una entidad que participa en un contexto particular.
- RUP: Rational Unified Process. Proceso Unificado de Rational. Especifica una posible metodología para ayudar en el proceso de desarrollo software.
- Servicio: Conjunto de funciones relacionadas que trabajan conjuntamente para proporcionar una funcionalidad.
- Servidor: Componente de un sistema que proporciona un servicio a otro componente. El componente pidiendo el servicio se llama cliente.
- Sistema: Colección de componentes que interaccionan entre sí.
- Transición: Cambio del estado de un objeto causado por un evento.
- UML: Unified Modeling Language. Es una manera estándar de modelar los datos de determinada aplicación, con una notación para expresar los datos (atributos, métodos), las relaciones entre los mismos y el conjunto de requerimientos que pueden ser satisfechos en la aplicación.

BIBLIOGRAFÍA

- www.omg.org/uml/
- Meta-links www.cetus-links.org/oo_uml.html
- Martin Fowler, autor de “UML Distilled” (“UML Gota a Gota”)
<http://www.martinfowler.com/>
- Herramientas basadas en UML
www.objectsbydesign.com/tools/umltools_byPrice.html
- International Council in SE (INCOSE) www.incose.org/tools/
- Revista IEEE Software, Conferencias: OOPSLA, ECOOP
- Patrones <http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html>,
- Foro UML en yahoo: <http://groups.yahoo.com/group/uml-forum/>
- Perdita Stevens, Rob Pooley. “*Utilización de UML en Ingeniería del Software con Objetos y Componentes*”. Addison Wesley, 2002.
- Grady Booch, James Rumbaugh, Ivar Jacobson. “*The Unified Modeling Language User Guide*”. Addison-Wesley, 1999.
- UML 2 Toolkit. Eriksson, H. E., Penker, M., Lyons, B., Fado, D. OMG Press, Wiley. 2004.
- Craig Larman. “*Applying UML and Patterns*”. Prentice Hall. 2002.

- Andreas Hennig, Rainer Wasgint. "*Performance Modelling of Software Systems in UML-Tools for the Software Developer*". ESM'2002 Proceedings, pp.: 94-99.