



## DESARROLLO DE SOFTWARE 3<sup>ER</sup> SEMESTRE

PROGRAMA DE LA ASIGNATURA:  
ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS

UNIDAD 3.  
METODOLOGÍAS DE DISEÑO PARA LA GENERACIÓN  
DE SISTEMAS ORIENTADOS A OBJETOS

CLAVE:  
INGENIERÍA:      TSU:  
15142313      16142313

CIUDAD DE MÉXICO, NOVIEMBRE DEL 2016

UNIVERSIDAD ABIERTA Y A DISTANCIA DE MÉXICO





## UNIDAD 3. METODOLOGÍAS DE DISEÑO PARA LA GENERACIÓN DE SISTEMAS ORIENTADOS A OBJETOS

### ÍNDICE

UNIDAD 3. METODOLOGÍAS DE DISEÑO PARA LA GENERACIÓN DE SISTEMAS ORIENTADOS A OBJETOS.....	3
PRESENTACIÓN DE LA UNIDAD .....	3
PROPÓSITOS DE LA UNIDAD .....	6
COMPETENCIA ESPECÍFICA.....	7
3.1. BOOCH.....	7
3.1.1. INTRODUCCIÓN .....	8
3.1.2. MODELOS .....	10
3.2. OOSE.....	16
3.2.1. INTRODUCCIÓN.....	17
3.2.2. MODELOS.....	19
3.3. OMT .....	24
3.3.1. INTRODUCCIÓN.....	25
3.3.2. MODELOS.....	27
3.4. UML .....	30
3.4.1. INTRODUCCIÓN.....	31
3.4.2. OCL ( LENGUAJE DE ESPECIFICACIONES DE OBJETOS) .....	32
CIERRE DE LA UNIDAD .....	34
PARA SABER MÁS .....	35
FUENTES DE CONSULTA .....	36



## UNIDAD 3. METODOLOGÍAS DE DISEÑO PARA LA GENERACIÓN DE SISTEMAS ORIENTADOS A OBJETOS

### UNIDAD 3. METODOLOGÍAS DE DISEÑO PARA LA GENERACIÓN DE SISTEMAS ORIENTADOS A OBJETOS

#### PRESENTACIÓN DE LA UNIDAD

Durante esta unidad se expondrán las diferentes metodologías, desde las que dieron origen a las actuales, para lograr y tener un diseño al momento de generar sistemas operativos.

Una de las primeras metodologías que surgieron fue la Booch, que permite realizar diagramas de clases, objetos, procesos, interacción; se abordará primeramente esa metodología. El siguiente modelo que se maneja en esta unidad es el OOSE (*Object Oriented Software Engineering* por sus siglas en inglés), el cual se utiliza más para las etapas de análisis, construcción y prueba; el siguiente modelo es el OMT que está enfocado hacia los objetos. De una manera muy general, la programación orientada a objetos es la más usada en la actualidad. Por último, conocerás la técnica UML, la cual es un concentrado de las técnicas anteriores.

Para el estudio de esta unidad es necesario que tengas presentes los siguientes conceptos:

- **Método**

Es un conjunto de normas y reglas con varios componentes como operaciones o pasos para lograr un objetivo; por ejemplo, los pasos que se requieren para hacer la suma de dos números.

- **Modelar**



## UNIDAD 3. METODOLOGÍAS DE DISEÑO PARA LA GENERACIÓN DE SISTEMAS ORIENTADOS A OBJETOS

Permite la captura del conocimiento a través de la obtención del modelo de un problema dado. Hay técnicas para modelado en las que se especifican los requerimientos, pero por lo general omiten ciertos aspectos de la implementación.

- **Metamodelo**

Se describen los modelos y conceptos de otros modelos, así como todas las relaciones existentes, y se describen los modelos construidos dentro de los métodos, así como la información que debe ser ingresada.

- **Vistas y notaciones**

Son utilizadas para la presentación de un modelo de información para que los usuarios de sistemas puedan interpretar, analizar y actualizar la información. Una vista sólo muestra una parte de la semántica del modelo, mientras que diferentes vistas pueden presentar la misma información en varias formas. Por otro lado, las notaciones permiten crear, analizar y manipular los modelos. Por lo general los modelos se esquematizan de diversas formas utilizando símbolos distintos, los cuales deben representar el mismo significado.

- **Proceso de desarrollo iterativo**

Este proceso ejemplifica el seguimiento de los pasos para la creación e implementación de modelos. Dicho proceso puede generarse en diferentes niveles de desarrollo, lo cual especifica qué modelos se deberán construir y cómo construirlos.

- **Proceso**

Es el conjunto de actividades que indican cómo producir un modelo; proporciona el esquema general de trabajo (*frameworks*) para el desarrollo. Dentro del proceso se especifican y describen los



## UNIDAD 3. METODOLOGÍAS DE DISEÑO PARA LA GENERACIÓN DE SISTEMAS ORIENTADOS A OBJETOS

mecanismos a ser desarrollados y así producirlos en dos formas: a alto nivel y a bajo nivel.

a) **A alto nivel:** Se describe el desarrollo dentro del ciclo de vida y las etapas de iteración dentro de él.

b) **A bajo nivel:** se describe un esquema de trabajo para la producción de modelos.

- **Patrón**

El patrón de diseño es la base de solución estándar utilizada para resolver un problema de diseño, ya que al considerar una solución, un patrón deberá incluir algunos criterios. En la actualidad se contemplan diferentes tipos de patrones con el fin de recolectar, dividir y exemplificar los patrones útiles para el diseño; de esa manera, se contempla que el contenido sea comprendido por otros diseñadores.

- **Reglas de diseño.**

Son útiles para especificar las características y/o atributos que se contengan para garantizar un diseño viable. Las reglas especifican todo lo que se puede desarrollar u omitir en el diseño.

Los métodos de AOO/DDO se dividen en dos tipos:

- Estáticos (enfocados a datos): lo más importante para un método estático es la estructura de los datos, pues tomando en cuenta esta consideración es que se construirá la arquitectura
- Dinámicos (enfocados a comportamientos o a responsabilidades): el método dinámico se representa por los diagramas de los objetos que muestran la forma en cómo las clases interactúan con otras.



## UNIDAD 3. METODOLOGÍAS DE DISEÑO PARA LA GENERACIÓN DE SISTEMAS ORIENTADOS A OBJETOS

A continuación se muestra un concentrado de los principales métodos de análisis y diseño orientado a objetos, identificados por sus nombres y siglas, así como sus autores.

Métodos	Significado	Autores
Booch	-----	Grady Booch
OOSE	<i>Object Oriented Software</i>	Jacobson
OMT	<i>Object Modeling Technique</i>	James Rumbaugh y Michael Blaha
UML	<i>Unified Modeling Language</i>	Grady Booch, autor del método Booch; James Rumbaugh, autor del método OMT e Ivar Jacobson

### PROPÓSITOS DE LA UNIDAD

Al término de esta unidad lograrás:

Identificar las diferentes metodologías para el diseño de sistemas orientados a objetos: Booch, OOSE (*Object-Oriented Software Engineering*/Ingeniería de Software Orientado a Objetos), OMT (*Object Modeling Technique*/Técnica de Modelado de Objetos) y UML (*Unified Modeling Language*/Lenguaje Unificado de Modelado) y distinguirás una de otra tomando en cuenta sus características.

- Definirás la metodología UML con apoyo de OCL (Lenguaje de Especificaciones de Objetos) y compararás la metodología UML con OCL.



## Competencia específica



Comparar las metodologías de diseño para la generación de sistemas orientados a objetos, mediante los diagramas con los métodos de modelado Booch, OOSE, OMT y UML.

### 3.1. BOOCH

El método de Booch, comúnmente conocido también como Diseño Orientado a Objetos de Grady Booch, brinda la facilidad de poder crear el desarrollo de análisis y diseño de un sistema orientado a objetos. El método de Booch es secuencial durante el desarrollo de sus fases, pues mientras la fase del análisis es completada, posteriormente la fase de diseño también debe serlo.

El método de Booch abarca el análisis y diseño de software, estructurándose desde un proceso de nivel micro donde un conjunto de tareas de análisis se van replicando en todas las etapas dentro de un macroproceso. Dentro del microporceso se identifican:

- Clases
- Objetos
- Transición de estados
- Módulos



- Procesos
- Interacciones

El macroproceso durante el diseño engloba cada una de estas actividades para agrupar en particiones y crear el prototipo de diseño.

### 3.1.1. INTRODUCCIÓN

A continuación se expone una breve descripción de las características del método Booch a manera de introducción al tema.

El método de Booch se basa en el análisis y el diseño, pero no en la implementación o las pruebas para el análisis y diseño, del cual es necesario analizar las dos dimensiones con las que se especifica la estructura y comportamiento de un sistema orientado a objetos:

- **Dimensión uno:** física/lógica.
- **Dimensión dos:** estática/dinámica.

En cada una de las dimensiones se especifica un grupo de diagramas, los cuales deberán mostrar una vista general de los modelos del sistema y del contenido sobre clases, relaciones y entidades.

- **Dimensión lógica:** son los pasos que especifican la existencia y significado de las fases principales y los mecanismos que forman la definición del problema hasta la planificación de acciones.



- **Dimensión física:** define la estructura de la cual está compuesta el hardware y el software para la implementación del sistema.
- **Dimensión estática:** está formada por los diagramas siguientes:
  1. Diagramas de clases: muestran las clases con sus relaciones.
  2. Diagramas de objetos: visualizan la existencia de objetos y sus relaciones en la etapa de diseño lógico de un sistema.
  3. Diagramas de módulos: muestran la asignación de clases y objetos, éstos representan una vista de la estructura de módulos de un sistema.
  4. Diagramas de procesos: representan los pasos secuenciales de todas las actividades durante el diseño físico de un sistema.
- **Dimensión dinámica:** la dimensión dinámica del comportamiento de un problema se expresa mediante las propiedades activas, el comportamiento individual y la colaboración entre objetos. Dentro de la dimensión dinámica se tienen los siguientes diagramas:
  1. Diagrama de transición de estados: también conocidos por sus siglas como *DTE*, enfatizan el comportamiento dependiente de cada instancia de una clase; es el conjunto de transiciones que pueden ocurrir en una entidad.
  2. Diagramas de interacción: muestran la interacción temporal en un conjunto de objetos y sus relaciones, incluidos los mensajes de un conjunto de objetos realizados entre éstos.

El método Booch fue un primer intento de clarificar lo que el cliente requiere, ya sea diseñado en un sistema o a través de diferentes diagramas los cuales son mostrados en el siguiente tema.



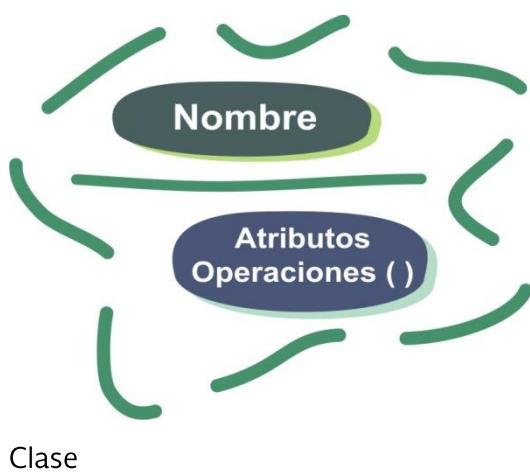
### 3.1.2. MODELOS

Los modelos que componen el método Booch para la dimensión estática como se vio en el tema anterior son los siguientes:

#### Diagramas de clases

Un diagrama de clases se utiliza para ejemplificar la existencia de clases y sus relaciones en la visión lógica que describe la estructura de un sistema. Los dos elementos esenciales de un diagrama de clases son las clases y sus relaciones básicas.

En la figura siguiente se muestra el diagrama de clase que es utilizado para representar una clase y sus atributos. En algunos diagramas de clases es necesario especificar algunos de los atributos contenidos, así como las operaciones que están asociadas con la clase:



Los atributos representan una parte de un objeto que es agregado; durante el diseño expresan una propiedad particular de una clase, donde se tiene lo siguiente:



A: representa el nombre del atributo.

C: representa la clase del atributo.

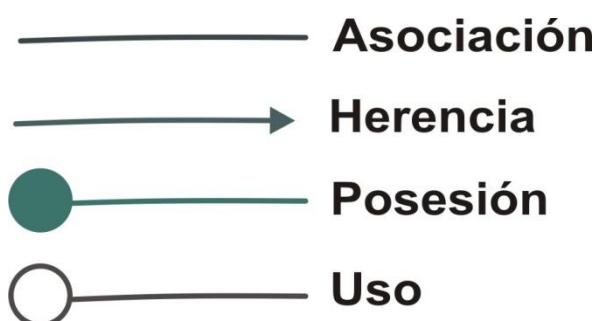
A:C: es la unión que es representada por el nombre y clase del atributo.

Las operaciones comúnmente nombradas *métodos* son aquellas actividades que denotan algún servicio proporcionado por la clase; se distinguen de los atributos porque usan paréntesis.

N(): representa el nombre de la operación.

R N(Argumento): clase de retorno de la operación, nombre y parámetros formales (si los hay).

Las relaciones existentes dentro de una clase representan una colaboración con otras y esta colaboración se efectúa de diferentes maneras. La simbología comúnmente utilizada para representar las conexiones entre las clases es la siguiente:



Conexiones entre clases

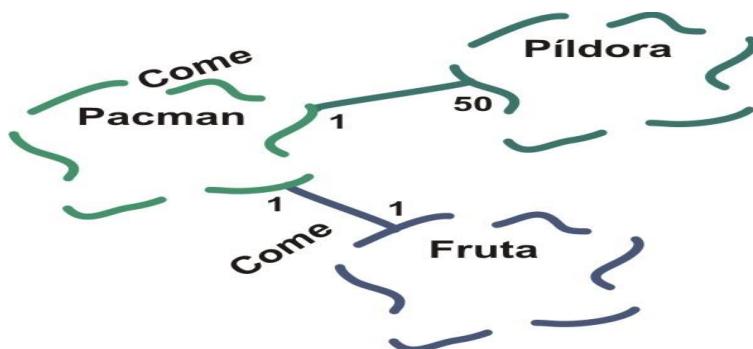
La conexión nombrada *asociación* representa la conexión semántica entre dos clases, las cuales se simbolizan con una expresión sustantiva exponiendo el origen de la relación.



La conexión de herencia representa la relación generalizada, la cual se muestra como una asociación con una cabeza de flecha. La flecha se apunta a la superclase, y el extremo opuesto de la asociación designa la subclase. La subclase hereda la estructura y comportamiento de la superclase. Las relaciones de herencia no pueden llevar indicaciones de cardinalidad.

La conexión de posesión representa la relación entre objetos la cual aparece como una asociación con un círculo relleno en el extremo que señala al agregado; la clase que está en el otro extremo denota la parte cuyas instancias están contenidas por el objeto agregado.

La conexión de uso representa una relación tipo cliente/servidor, en la cual se muestra una representación de asociación con una circunferencia en el extremo que denota al cliente. En esta relación, de alguna forma, el cliente depende del servidor para que éste le proporcione determinados servicios.



Utilización

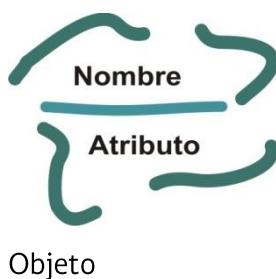
La figura muestra un diagrama de clases en donde las clases Pacman tienen una relación con las clases fruta y las clases píldoras; aquí se establece una relación de tipo utilización.



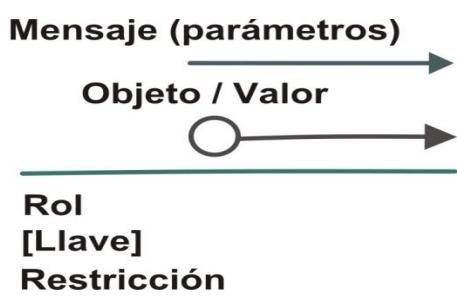
## Diagramas de objetos

Los diagramas de objetos son utilizados para representar la existencia de objetos y sus relaciones en el diseño lógico de un sistema. Los dos elementos que forman básicamente los diagramas de objetos son los *objetos* y sus *relaciones*.

Los *objetos* representan la unidad o instancia de una clase. En la siguiente figura, se representa el uso de un objeto dentro de un diagrama de este tipo, el cual es representado por atributos del objeto similar al diagrama de clases.



Las relaciones entre los objetos se representan por medio de los enlaces con otros objetos dentro de una instancia de una asociación, al igual que un objeto es una instancia de una clase.

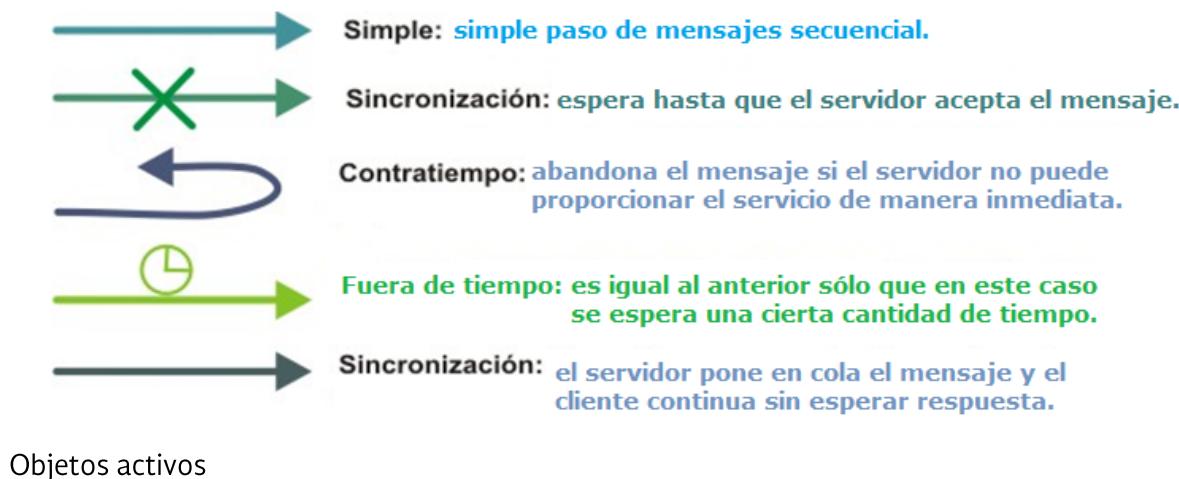


Relaciones entre objetos

El flujo de los datos puede seguir en la misma dirección que un mensaje o en dirección contraria. Mostrar explícitamente la dirección del flujo de datos



ayuda a explicar la semántica de un escenario particular. Los objetos activos se utilizan para incorporar su propio hilo de control.



### Diagramas de módulos

Los diagramas de módulos son utilizados para representar la asignación de clases y objetos a módulos en el diseño físico de un sistema; es representado por una vista dentro de la estructura de módulos de un sistema; sus elementos esenciales son los *módulos* y sus *dependencias*.

Un programa principal expresa un archivo que contiene la base del programa. La especificación y el cuerpo expresan los archivos que integran una declaración y una definición de las entidades.

Los subsistemas sirven para afinar el modelo físico de un sistema. Un subsistema es una parte contenida en otros módulos y otros subsistemas. Cada módulo integra los objetos y otros detalles del lenguaje.



Dependencias son las únicas relaciones que pueden existir entre los módulos, se trata de una dependencia de compilación; se representa por una línea direccionada que apunta al módulo, con respecto al cual existe la dependencia; las flechas denotan dependencias.

### Diagrama de procesos

Los diagramas de procesos se utilizan para esquematizar la asignación de procesos a procesadores durante el diseño físico de un sistema. Uno puede presentar una vista de la estructura de procesos de un sistema.

Los elementos del diagrama de procesos son:

- **Procesadores:** elemento de hardware capaz de interpretar las instrucciones; procesa o ejecuta los programas.
- **Dispositivos:** elemento de hardware de un sistema de cómputo, incapaz de ejecutar o procesar un programa.
- **Conexiones:** son líneas no dirigidas para indicar conexiones entre procesadores y/o dispositivos.

Diagrama de procesos

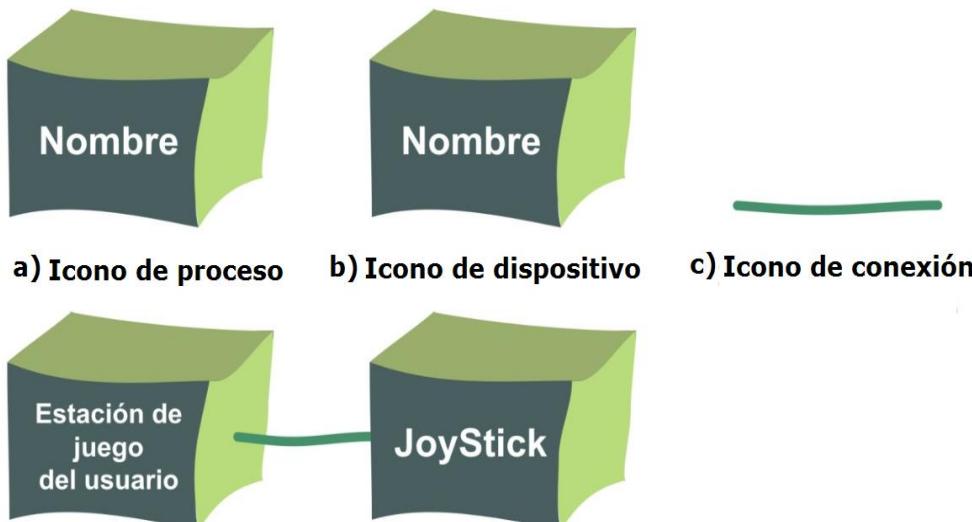


Diagrama de procesos



El proceso de diseño orientado a objetos no es posible plasmarse mediante reglas, aunque está bastante bien definido como para brindar un proceso predecible y repetible para una organización de software madura.

Establecer un buen proyecto mediante el análisis, hará posible construir un buen diseño, lo cual puede beneficiar las expectativas del usuario final.

### 3.2. OOSE

El método de OOSE (*Object-Oriented Software Engineering/Ingeniería de Software Orientado a Objetos*) fue desarrollado por Ivar Jacobson. Este modelo combina tres técnicas diferentes para el manejo de casos de uso, y sirve como un modelo central para otros modelos.

Este modelo es la base durante la etapa de análisis, construcción y prueba de los sistemas.

OOSE utiliza cinco técnicas, nombradas modelos, para representar un sistema:

- **Modelo de requerimientos:** es utilizado para establecer los límites del sistema, y se destacan por ser funcionales y útiles.
- **Modelo de análisis:** se utiliza para desarrollar la estructura del sistema, donde existen tres tipos de objetos: de interface, de entidad y de control.
- **Modelo de diseño:** es el subsecuente del modelo de análisis para el diseño y se adapta para establecerlo dentro de un ambiente de implementación.



- **Modelo de implementación:** se basa en el código fuente de los objetos especificados en el modelo de diseño.
- **Modelo de prueba:** se realiza para establecer las pruebas al modelo de implementación.

La principal idea de los modelos de OOSE es capturar el concepto inicial de todos los requerimientos funcionales y usar sus perspectivas. Es por eso que la relación entre ellos es importante. Para hacer posible el mantenimiento del sistema es también necesario que los modelos sean tangibles.

### 3.2.1. INTRODUCCIÓN

A continuación se expone una breve descripción de las características del método OOSE a manera de introducción al tema.

El método de OOSE proporciona un soporte para un diseño creativo de programas. La idea se plantea sobre el problema del diseño y construcción de software; contempla las siguientes fases:

- **Herramientas:** son todos los aspectos a utilizar para el desarrollo de las actividades, métodos y procesos.
- **Procesos:** permiten escalar los métodos, de tal manera que puedan ser aplicados a proyectos de forma interactiva y en partes.
- **Métodos:** se establecen de forma explícita considerando los procedimientos que deben seguirse para aplicar la arquitectura al proyecto.
- **Arquitectura:** establecer una buena estructura del sistema será de utilidad para comprenderlo fácilmente, si se desea cambiar y realizar pruebas y mantenimiento.



El diseño creativo define las actividades de un desarrollo; consiste en la creación de un conjunto de requerimientos y nociones vagas, para poder establecer un buen plan estructurado para su creación y un plan de acción para la implementación. Un diseño creativo toma como base la estructura paso a paso de los métodos y procesos, con la asistencia de herramientas, para convertir los requerimientos dentro de una arquitectura viable para la construcción de un proyecto incluyendo la creación de prototipos.

Durante el desarrollo de un sistema el proceso de cambio se debe considerar como un aspecto importante; todos los sistemas tienden a ser modificados durante su ciclo de vida.

En la actualidad, para el desarrollo de nuevos métodos de diseño, se debe conocer qué cambios son los principales dentro del sistema y la parte global del ciclo de vida, así como el costo del sistema. La primera versión de un sistema representa una pequeña parte de una composición durante el ciclo de vida de éste.

En el ciclo de vida del sistema, las actividades son las mismas tanto para desarrollar una nueva versión de un sistema, así como para uno totalmente nuevo. A continuación se muestran las actividades:

- **Construcción:** la primera actividad en la construcción consiste en la implementación de los detalles que conciernen a la arquitectura y construcción del plan, que es ir de una mayor abstracción a concretizar más el plan.
- **Diseño:** el diseño plasma el modelo de análisis en términos del ambiente de implementación y especifica la identidad de los bloques de construcción.



- **Pruebas:** las pruebas del sistema consisten en verificar el trabajo de cada uno de los paquetes de servicio definidos en el modelo de análisis.

### 3.2.2. MODELOS

En este tema se describen los modelos que comprenden el método OOSE. Desarrollar un sistema es una tarea complicada; por ello, para determinar el desarrollo se toman como base algunos aspectos importantes para trabajar con los cinco modelos, los cuales son:

1. **El modelo de requerimientos:** el principal objetivo es obtener los requerimientos funcionales que son la base para el análisis del sistema.
2. **El modelo de análisis:** el objetivo de este modelo es definir mediante las investigaciones una estructura de objetos robusta y flexible a los cambios.
3. **Modelo de diseño:** para el modelo de diseño su objetivo es obtener lo recabado durante el desarrollo de análisis para refinar la estructura de los objetos para el desarrollo del ambiente del sistema.
4. **El modelo de implementación:** el principal objetivo de este modelo es establecer las mejores prácticas para el proceso de implementación, el cual deberá cumplir para que el entorno del sistema no se vea afectado.
5. **El modelo de prueba:** su objetivo es verificar el buen funcionamiento del sistema, por lo que se basa en realizar diferentes tipos de pruebas, con el fin de localizar posibles fallas del sistema.

La necesidad básica de estos modelos es obtener el concepto inicial de todos los requerimientos funcionales y usar sus perspectivas. Por tal motivo la



relación entre ellos es importante para hacer posible el mantenimiento del sistema y para que los modelos sean tangibles.

### Modelo de requerimientos

En este modelo se tiene como interacción:

- Actores
- Casos de uso

Las características a utilizar para el modelo de requerimientos son:

- Un modelo de caso de uso
- Descripción de la interface
- Un modelo en el dominio del problema



**Actor**



**Caso de Uso**

Modelo de caso de uso

El modelo de caso de uso integra *actores* (usuarios del sistema) y *caso de uso* (requerimientos de utilidad para el sistema). Estos conceptos se utilizan para definir si existe contacto externo con el sistema (actores), y qué deberá poder hacer el sistema (caso de uso).

Los *actores* son la parte que representan quienes interactúan con el sistema, en todo caso puede definirse como el usuario final. Los actores representan todas las necesidades de cambio de información con el sistema.

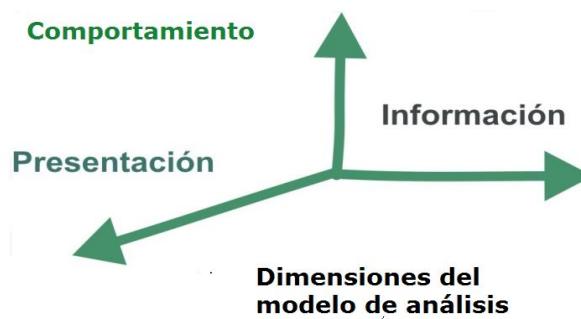


## Modelo de análisis

Como se mencionó anteriormente el modelo de análisis tiene como objetivo definir las limitaciones del sistema y especificar su comportamiento. Una vez que el modelo de requerimientos ha sido desarrollado, se puede iniciar el desarrollo del sistema.

Los requerimientos necesarios para este modelo se basa en la obtención de:

- **Información:** la información es una de las principales fuentes de ayuda para el desarrollo del sistema, pues se podrá describir cuál será el estado del sistema.
- **Comportamiento:** este requerimiento es de utilidad para saber determinar el comportamiento que va tomando el sistema durante su desarrollo, especificando cuándo y cómo el sistema cambia de estado.
- **Presentación:** es de utilidad pues representa la forma de cómo el sistema será presentado al mundo exterior.

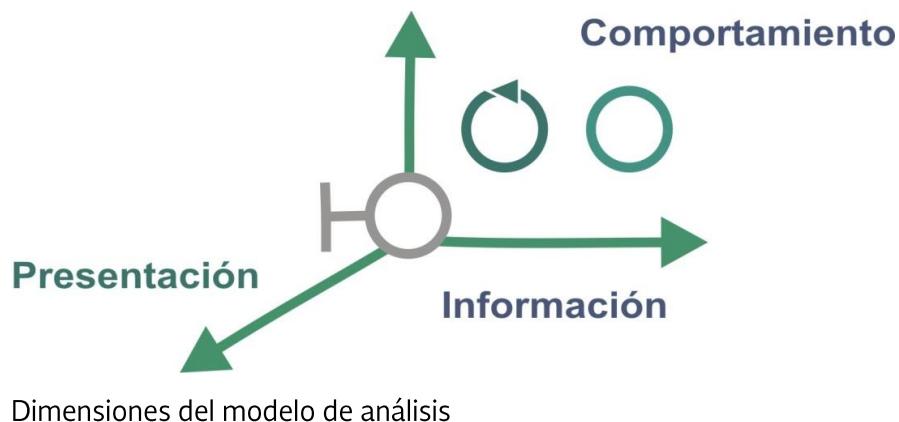


Dimensiones del modelo de análisis

En la siguiente figura se presenta como pueden existir varios tipos de objetos utilizados para la estructura del sistema en el modelo de análisis.

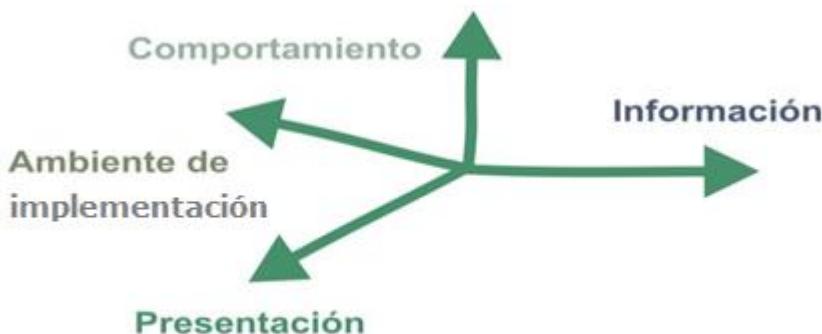


Al menos deberá ser capturada una de las tres dimensiones para cada objeto, tal como se muestra en la siguiente figura; sin embargo, cada uno de ellos tiene cierta inclinación hacia una de las dimensiones.



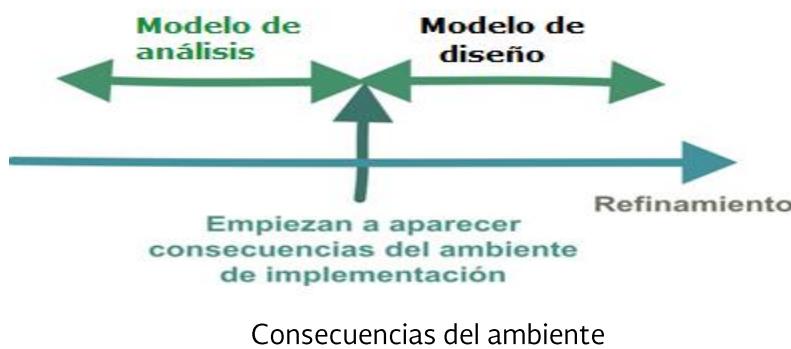
### El modelo de diseño

Durante el proceso de diseño del sistema se utiliza tanto el modelo de análisis como el modelo de requerimientos. Para ello se crea el modelo de diseño que es un refinamiento y formalización del modelo de análisis. Desarrollar el modelo de diseño sirve para adaptarlo al modelo de implementación.



Implementación del ambiente

La principal diferencia en el modelo de análisis y el modelo de diseño, es que el primero se utiliza para establecer un modelo conceptual o lógico del sistema. Y el modelo de diseño contiene la estructura del sistema (código fuente), por lo cual el modelo de diseño deberá ser una representación de la manera como el código fuente es estructurado, manejado y escrito.



### El modelo de implementación

El modelo de implementación consiste en denotar el código para que la información del lenguaje de programación que se utiliza sea la más adecuada para el buen funcionamiento del sistema. No es necesario requerir de un lenguaje de programación orientada a objeto, pero si es recomendable el uso de un lenguaje de programación orientada a objeto, desde la concepción inicial



hasta la construcción. Para este modelo la base para una buena implementación es determinada por el modelo de diseño.

### **El modelo de prueba**

El último modelo utilizado por OOSE es el modelo de prueba el cual explica simplemente el estado de resultados de la prueba. El modelo de requerimientos de nuevo representa una herramienta fundamental para el desarrollo de las pruebas de cada caso de uso, se verifica que los objetos se comuniquen correctamente en dicho caso de uso. De igual forma al simular el funcionamiento del sistema, se verifica la interface de usuario, que se especificó durante el modelo de requerimientos.

### **3.3. OMT**

La metodología de modelado de objetos (OMT, por sus siglas en inglés *Object Modeling Technique*) fue creada por Rumbaugh. En ella se desarrolla un modelo que representa lo que va hacer el sistema, con la finalidad de comprenderlo antes de desarrollarlo.

Esta técnica toma en cuenta tres líneas para su utilización, tales como:

- Modelo de objetos
- Modelo dinámico
- Modelo funcional



### 3.3.1. INTRODUCCIÓN

A continuación se redacta una breve descripción de las características del método OMT a manera de introducción al tema.

La creación de modelos a través de la técnica OMT se considera, por algunos expertos, como el modelo más importante, ya que en él se identifican las clases que van dentro del sistema, así como sus relaciones. También son considerados los atributos y operaciones lo cual representa la estructura del sistema; el modelado de objetos por lo general es representado por un diagrama de clases.

El *modelo dinámico* es representado por los aspectos temporales de comportamiento del sistema, mediante la secuencia de operaciones en el tiempo.

El *modelo funcional* es representado por los aspectos que se transforman de acuerdo a la función del sistema; este modelo comúnmente es representado mediante un diagrama de flujo.

Cada uno de los modelos mencionados especifican los aspectos del sistema; sin embargo, cada uno de éstos se especifica con los aspectos del sistema, pero contiene referencias a los demás modelos presentados, lo cual indica que los tres no son totalmente independientes.

Los pasos del proceso de desarrollo orientado a objetos son los siguientes:

- **Conceptualización:** este proceso define todos los requerimientos para la determinación que deberá tener el sistema, inicia identificando las



necesidades desde el punto de vista de los usuarios. La información por lo general es extraída de los casos de uso y del dominio del problema.

- **Análisis:** consiste en determinar el modelo para comprender el problema en el dominio de la aplicación.
- **Diseño del sistema:** este requerimiento determina la estructura del sistema en términos de subsistemas.
- **Diseño de objetos:** consiste en dimensionar, afinar y optimizar el modelo de análisis, agregando conceptos de programación.
- **Código:** este requerimiento lleva a la implementación las clases de objetos representadas en un lenguaje de programación.
- **Pruebas:** las pruebas a realizar son útiles para validar el comportamiento de las clases y objetos que se encuentran descritos en los escenarios.



La figura anterior es un claro ejemplo de cada paso del proceso, transforma algunas entradas para generar una salida diferente, comenzando en un alto nivel de abstracción hasta llevarlo a un nivel de detalle que finalmente representa la solución del problema.



### 3.3.2. MODELOS

En este tema se describen los modelos que comprenden el método OMT. Se tienen considerados ocho pasos para construir un modelo de objetos, entre ellos están los siguientes:

1. Identificación de objetos y/o clases.
2. Crear un diccionario de datos.
3. Identificación de las asociaciones y agregaciones entre los objetos.
4. Identificación de atributos y enlaces.
5. Organización y simplificación de las clases empleando herencia.
6. Verificación de las vías de acceso necesarias para llevar a cabo las probables consultas.
7. Realizar las interacciones necesarias para el refinamiento del modelo.
8. Agrupar las clases en módulos.

Modelo de objeto = Diagrama de modelo de objetos + Diccionario de datos.

En los diagramas de clase para la metodología de modelado de objetos, se especifican las clases que se descubrieron para el sistema analizado en términos del dominio del problema. Además se especifican los atributos y operaciones que distinguen a cada una de las clases y las relaciones con las que podemos conocer su responsabilidad en el sistema.



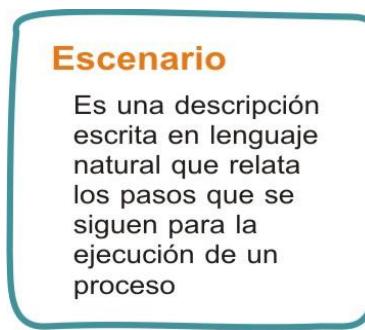
Nombre/Clase

Para el diagrama de clases, una clase es representada mediante un dibujo en forma de rectángulo donde se pueden contemplar tres separaciones; en la



primer parte se inserta el nombre de la clase; en la segunda y tercera, se pueden agregar los atributos o bien las operaciones, tal como se muestra en la figura anterior, donde no se insertaron atributos y operaciones, porque no son importantes en esa clase para la comprensión.

Un escenario es la representación escrita de los casos de uso y de la interacción de los actores con ellos para especificar el propósito del sistema; la siguiente imagen muestra un ejemplo de un escenario describiendo el caso de uso para desarrollo del sistema.



Escenario

Los diagramas de estados relacionan los sucesos y estados. Un diagrama de estados se representa mediante estados, transiciones, condiciones y acciones.

- Los estados son utilizados para representar las respuestas de los objetos de varios sucesos en determinado tiempo dentro del sistema. Cada respuesta puede modificar el estado del objeto.
- Las transiciones son representadas mediante símbolos en forma de flecha que salen del estado receptor hasta el nombre que se coloca en la flecha.
- Las condiciones adoptan una condición que se pueden pensar como una protección en las transiciones, si se cumple alguna condición la transición se dará y podrá pasar el objeto de un estado a otro; si dicha

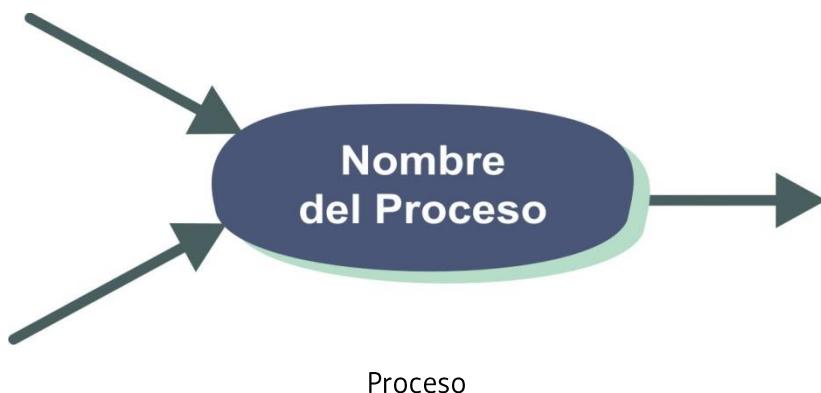


condición no se cumple inclusive podría pasar a otro estado, mediante otra transición o quedarse en el estado receptor, hasta que la condición se cumpla.

- **Acción:** es una operación que va asociada a un suceso y se representa mediante una barra “/” y el nombre de la acción, después del nombre de la transición.

Mediante el modelo funcional, se pueden visualizar resultados que se obtienen de un cálculo de valores; se especifican solamente entradas y salidas de los valores, mas no como son calculados éstos. El modelo funcional está formado básicamente de diagramas de flujo de datos. Los diagramas de flujo de datos son símbolos que muestran el flujo de los valores de datos a través de procesos, los cuales modifican dichos valores para transformarlos en otros. Los diagramas de flujo están compuestos de:

- **Procesos:** son exemplificados por medio de una elipse, los procesos tienen datos como entradas las cuales serán modificados.
- **Flujos de datos:** un flujo de datos interconecta la salida de los procesos a la entrada de otro, el cual es representado en el diagrama por medio de una flecha, la cual por lo general lleva el nombre o el tipo de dato.
- **Actores:** el actor está definido por un papel de usuario que puede jugar intercambiando la información con el sistema.
- **Almacenes:** constan de múltiples diagramas de flujo de datos.



### 3.4. UML

El lenguaje unificado de modelado es un conjunto de diagramas estandarizados utilizados para el modelado de sistemas orientados a objetos. Hoy en día es uno de los lenguajes de modelado de sistemas de software más utilizado, es un lenguaje de entorno gráfico que sirve para visualizar, construir y documentar un desarrollo de sistema. Algunos de los principales beneficios sobre la utilidad de este lenguaje son los siguientes:

- Minimiza el tiempo de desarrollo de sistemas.
- Fácil uso para modelar sistemas orientados a objetos.
- Se pueden establecer conceptos y artefactos ejecutables.
- Puede encaminar el desarrollo sobre un escalamiento en sistemas complejos de misión crítica.
- Se puede crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Utilidad para mejorar el soporte sobre la planeación y al control de proyectos.
- Óptimo para la reutilización y minimización de costos.



### 3.4.1. INTRODUCCIÓN

A continuación se redacta una breve descripción de las características del método UML a manera de introducción al tema.

UML no está considerado como un método, sino como un lenguaje para el modelado utilizado para desarrollar un sistema, existen varias metodologías, las cuales se vieron en temas anteriores. UML nos brinda un sistema de arquitecturas basadas con objetos, para el análisis y diseño.

La idea al desarrollar el lenguaje unificado de modelo se realiza bajo métodos existentes para el análisis y diseño de sistemas orientados a objetos. Dentro de los objetivos que se estipularon al desarrollar UML se tienen los siguientes:

- Otorgar a los desarrolladores un lenguaje con un entorno gráfico de fácil comprensión para poder intercambiar información de los modelos.
- Proporcionar mecanismos de extensos y especializados para ampliar los conceptos básicos.
- Puede ser independiente de un lenguaje en particular y del proceso de desarrollo.
- Puede otorgar bases formales para la comprensión del lenguaje de modelado.
- Integra una mejor práctica para el desarrollo.

Contemplando el hecho que el usuario desarrollador requiere de modelos para manejar el diseño de sistemas más complejos, así conforme se va teniendo la necesidad de crear aplicaciones más complejas, se tiene la necesidad de contar con mejores técnicas de modelado. Saber que se cuenta con una metodología universal para el desarrollo de sistemas de software es de gran beneficio en la



construcción de todo tipo de sistemas. Disponer de buenos modelos facilita la comunicación entre equipos de trabajo en un gran proyecto.

Dentro del esquema de UML, existen básicamente cuatro constructores gráficos utilizados: iconos, símbolos de dos dimensiones, uniones y cadenas.

- **Iconos:** un ícono es un símbolo gráfico de tamaño y forma definida, éstos pueden aparecer dentro del área de los símbolos, en la terminación de una unión, etc.
- **Símbolos de dos dimensiones:** los símbolos de dos dimensiones son de tamaño variable, y pueden contener listas de cadenas u otros símbolos.
- **Uniones:** son segmentos de línea con sus extremos terminados en algún símbolo de dos dimensiones.
- **Cadenas:** representan conceptos, pueden existir como un elemento dentro de un símbolo o dentro de un compartimiento de un símbolo, como elementos de una lista, como etiquetas de un símbolo o unión, o como un elemento estándar dentro de un diagrama.

### 3.4.2. OCL (LENGUAJE DE ESPECIFICACIONES DE OBJETOS)

El Lenguaje de Especificación de Objetos (OCL, por sus siglas en inglés: *Object Constraint Language*) es un lenguaje de descripción formal de expresiones en los modelos UML. Como forma parte del lenguaje UML y de algunos otros, se puede usar el lenguaje de especificaciones de objetos para determinar aquellas restricciones y expresiones que puedan representar algunas variantes y condiciones de los objetos al igual que las consultas a objetos para determinar sus condiciones de estado, la principal característica del lenguaje de



especificaciones de objetos es que está basado en características de un lenguaje común de modelado y lenguaje formal de expresión

El lenguaje OCL es de expresión sin efectos de borde, de esta forma, garantiza que una expresión OCL no tendrá efectos colaterales. Cabe destacar que generalmente no se puede modificar nada en el modelo. Lo anterior significa que nunca se podrá modificar el estado del sistema y a su vez una expresión OCL podrá ser utilizada para poder modificar un cambio específico en su estado inicial, por ejemplo, en una poscondición.

Una expresión OCL representa un cambio en el sistema; este cambio tiene que darse por un cambio de estado. La mayoría de los valores de todos los objetos, incluyendo los enlaces, no se verán afectados en alguna modificación, cuando una expresión OCL sea evaluada, sólo regresa el valor. Es importante no confundir o comparar al OCL, con un lenguaje de programación puesto que no lo es y por lo tanto, es imposible escribir lógica de programa o flujo de control en OCL.

También es imposible llamar procesos o bien activar operaciones que no sean consultas en OCL. Dado que el OCL es un lenguaje de modelado en primer lugar, es posible que haya cosas en él que no sean directamente ejecutables. Como el OCL es un lenguaje de modelado, y no un lenguaje de programación como se mencionó, cualquier consideración de implementación está fuera de su alcance, y no puede ser expresada en el lenguaje OCL.

A manera de resumen se entiende que el OCL es un lenguaje formal y como se verá más adelante forma parte del UML.



El OCL puede ser usado con distintos propósitos:

- Puede ser utilizado para definir las características sobre clases y tipos en un modelo de clases.
- Útil para definir características estáticas de tipo para prototipos.
- Sirve para plantear pre y poscondiciones sobre operaciones y métodos.
- Para establecer las restricciones sobre operaciones:

En el documento de *Semántica del UML*, OCL es utilizado en la sección “Reglas bien definidas” y en “Constantes estáticas sobre la meta-clase en la sintaxis abstracta”.

El OCL se forma de pre y poscondiciones, las cuales son restricciones o condiciones y éstas se aplican tanto en métodos como en operaciones.

### CIERRE DE LA UNIDAD

Has concluido la unidad 3 del curso. A lo largo de ésta se recordaron las metodologías de diseño para la generación de programas orientados a objetos, tales como Booch, OOSE, OMT, en cada uno de ellos se vio una breve introducción y sus modelos. Por último el origen de la metodología UML, la cual fue a través del OCL.

Es aconsejable que revises nuevamente la unidad en caso de que los temas que se acaban de mencionar no te sean familiares, o no los recuerdes, de no ser éste tu caso, ya estás preparado(a) para seguir con la unidad 4, en donde continuarás con el diseño orientado a objetos con UML, a través de la representación de objetos y clases con diagramas y documentación para el diseño del software con UML, en dichos diagramas se manejarán casos de uso,



escenarios del caso de uso, diagramas de actividades, secuencial, de clase y de gráfico de estado. Todo ello con el fin de obtener el prototipo final al terminar de la asignatura Análisis y diseño orientado a objetos.

### PARA SABER MÁS



Ideas Tomada de <http://goo.gl/2F4jYb>

En el siguiente documento encontrarás un ejemplo real de análisis aplicado al diseño de un sistema escolar:

- Desarrollo de un sistema de administración académica para la escuela “Cristóbal Colón” bajo plataforma web, disponible en <http://bibdigital.epn.edu.ec/bitstream/15000/3823/1/CD-3595.pdf>



## FUENTES DE CONSULTA



Libros Tomada de <http://goo.gl/wEZJP9>

- Booch, G. (2009). *Análisis y diseño orientado a objetos con aplicaciones*. México: Pearson Educación.
- Date, C. J. (2001). *Introducción a los sistemas de bases de datos* (S. Ruiz, trad.) (7<sup>a</sup> ed.). México: Pearson-Prentice Hall.
- Fowler, M. y Scott, K. (1999). *UML gota a gota*. México: Addison Wesley.
- Graham, I. (2002). *Métodos orientados a objetos*. México: Addison Wesley/Díaz de Santos.
- Jacobson, I. (1992). *Object-Oriented Software Engineering. A Use Case Driven Approach*. México: Addison Wesley.
- James, R., Blaha, M., Premerlani, W. y Eddym, F. (1990). *Object Oriented Modeling and Design*. México: Prentice Hall.
- Larman, C. (2004). *Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design*. México: Prentice Hall.
- Martin, J. y Odell, J. (1990). *Análisis y diseño orientado a objetos*. México: Prentice Hall.
- Quatrani, T. y James, R. (1997). *Visual modeling with rational rose and UML*. México: Addison Wesley.
- Wirfs, R. y Wiener, L. (1990). *Designing Object Oriented Software*. México: Prentice Hall.