Escuela : Tecnologías de la Información

ANÁLISIS Y DISEÑO DE SISTEMAS

SENATI

Semana 2:

RESULTADO DE APRENDIZAJE:

- Entender la importancia de utilizar UML para diseñar un sistema.
- Identificar los elementos del Diagrama de caso de uso
- Diseñando Modelo funcional de un proceso

CONTENIDO:

- Análisis Orientado a Objetos (AOO):
  - Conceptos de abstracción, modularidad e identificación de objetos para analizar y diseñar sistemas.
- Unified Modeling Language (UML)
  - Concepto
  - Representación gráfica de sistemas, procesos y estructuras.
  - Principios de diseño de los modelos UML.
  - Nomenclatura en UML
- Entidades y relaciones
  - ¿Qué son entidades?
  - ¿Qué son relaciones?
- Diagramas UML
  - Tipos de diagramas (diagrama de Casos de Uso; Diagrama de Clases; Diagrama de Secuencia; Diagrama de Actividades)
- Análisis de requisitos del sistema con UML

Análisis Orientado a Objetos (AOO) y UML

¿Qué es objeto?

Es una instancia de una clase especifica. Donde los objetos heredan los atributos y operaciones de una clase usada.

¿Qué es clase?

La clase representa a una entidad que tiene un estado interno y un comportamiento característico.

¿Qué es un atributo?

Un atributo son los datos a los que se refiere el esta interno y un comportamiento característico.

*Los objetos tienen un **estado interno** y un comportamiento, el estado de un determinado objeto **es un conjunto de parámetros con valores que caracterizan**.*

Análisis Orientado a Objetos (AOO) y UML

El propósito del análisis orientado a objetos es desarrollar una ***serie de modelos que describan el software de computadora al trabajar para satisfacer un conjunto de requisitos definidos por el cliente***.

Asimismo, el modelo de análisis **ilustra información, funcionamiento y comportamiento** dentro del contexto de los elementos del modelo de objetos.

Análisis Orientado a Objetos (AOO) y UML

El **Análisis Orientado a Objetos (AOO)** surge en los años 80 como respuesta a la complejidad creciente de los sistemas.

La **popularidad de las tecnologías de objetos** generó docenas de métodos durante los 80s y 90s. Cada método proponía:

- Un **proceso de análisis y diseño**.
- Un **conjunto de modelos**.
- Una **notación gráfica**.

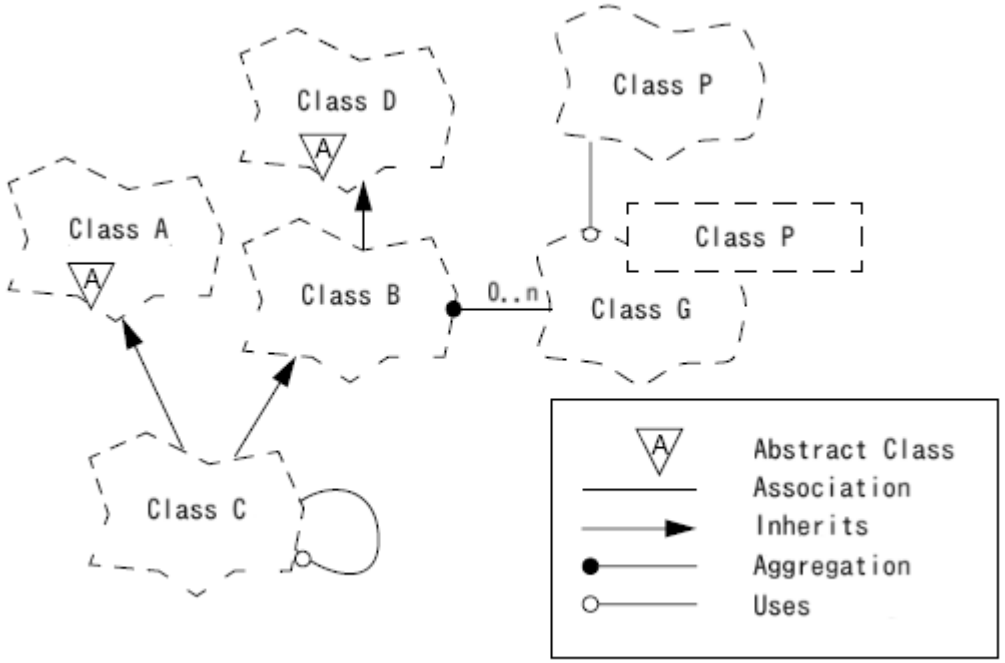
Con el tiempo, estas propuestas se unificaron en un estándar: **UML (1997, OMG)**.

Análisis Orientado a Objetos (AOO) y UML

1. Método de Booch (1994)

- Propone un **microproceso** (tareas de análisis repetibles) y un macroproceso (etapas de desarrollo).
- Mantiene un enfoque evolutivo.
- Identifica clases y objetos, define relaciones y refina el modelo.
- **El método Booch incluye** seis tipos de diagramas: clase, objeto, transición de estados, interacción, módulo y proceso.

Ejemplo: Booch clasificaría primero clases como Cliente, Cuenta, Transacción, y luego refinaría sus atributos y operaciones.

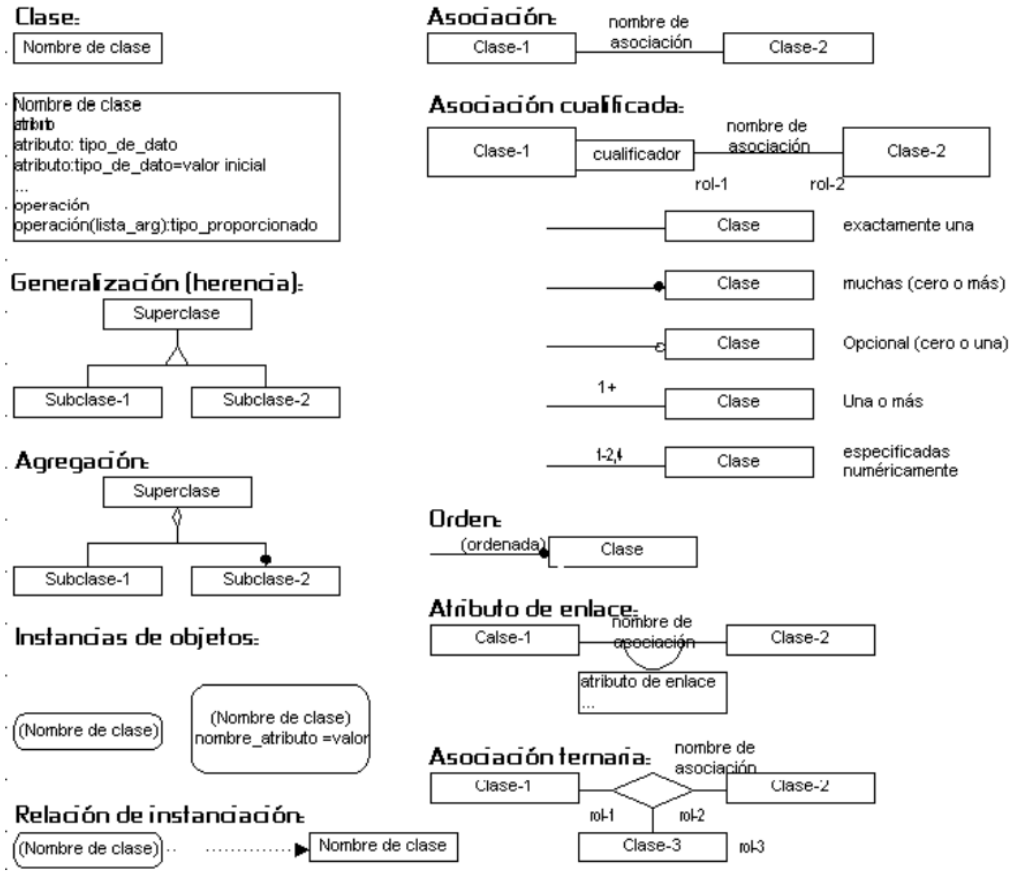


Análisis Orientado a Objetos (AOO) y UML

2. **Método de Rumbaugh (OMT, 1991)**

- Dinámico: comportamiento y estados.
- Funcional: flujo de información (similar a DFD).
- Es una de las metodologías de análisis y diseño orientadas a objetos, más maduras y eficientes que existen en la actualidad.

Ejemplo: Un préstamo de biblioteca → modelo de objeto (Préstamo), modelo dinámico (ciclo del préstamo) y modelo funcional (flujo de registrar préstamo)

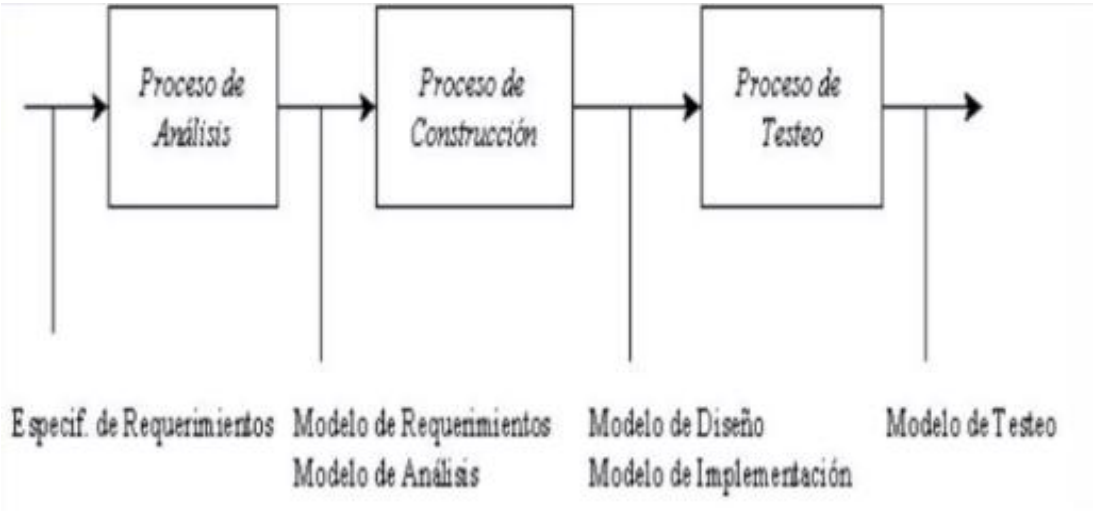


Análisis Orientado a Objetos (AOO) y UML

3. **Método de Jacobson (OOSE, 1992)**

- *Se centra en los casos de uso como eje del análisis.*
- Cada escenario describe cómo interactúa el usuario con el sistema.
- Abarca todo el ciclo de vida y prioriza la trazabilidad entre las diferentes fases, tanto hacia adelante como hacia atrás.

Ejemplo: Caso de uso “Retirar dinero” → se convierte en punto de partida para todo el análisis.



Análisis Orientado a Objetos (AOO) y UML

4. **Método de Coad y Yourdon (1991)**

- Uno de los más sencillos de aprender.
- Notación simple y reglas claras.
- Etapas:
  - Identificar objetos.
  - Definir jerarquías (generalización/especificación).
  - Definir relaciones todo-parte.
  - Identificar temas o subsistemas.
  - Definir atributos.
  - Definir servicios (métodos).

Ejemplo: Un sistema de ventas → objetos: Producto, Factura, Cliente. Luego se definen atributos (precio, fecha) y servicios (emitirFactura, registrarVenta).

Análisis Orientado a Objetos (AOO) y UML

5. **Método de Wirfs-Brock (1990)**

- Propone un proceso continuo sin separar análisis de diseño.
- Etapas:
  - Evaluar especificación del cliente.
  - Identificar clases candidatas.
  - Definir responsabilidades de cada clase.
  - Asignar colaboraciones entre clases.
  - Importancia en el concepto de “Responsabilidad de clase”.

Ejemplo: Clase Factura → responsabilidades: “Calcular total”, “Aplicar impuestos”, “Imprimir comprobante”.

Análisis Orientado a Objetos (AOO) y UML

Método	Año	Enfoque principal	Aporte clave
Booch	1994	Micro/macroproceso, evolución incremental	Refinamiento iterativo
Rumbaugh (OMT)	1991	Modelos: objeto, dinámico, funcional	Visión integral del sistema
Jacobson (OOSE)	1992	Casos de uso	Requisitos centrados en el usuario
Coad & Yourdon	1991	Objetos simples y jerarquías	Notación sencilla
Wirfs-Brock	1990	Responsabilidades de clases	Colaboración y diseño continuo

Análisis Orientado a Objetos (AOO) y UML

Principios de AOO

Abstracción

Consiste en identificar lo esencial de un objeto y dejar de lado los detalles irrelevantes.

Permite centrarse **en qué hace** un objeto en lugar **de como lo hace**.

Ejemplo:

Un objeto "**Cuenta Bancaria**" se abstrae como número de cuenta, saldo, operaciones de depósito/retiro, sin importar cómo está implementado en el banco.

Encapsulamiento

Los atributos y comportamientos de un objeto están protegidos dentro de él.

Se controla el acceso mediante métodos, evitando la manipulación directa de datos.

Ejemplo:

No podemos modificar directamente el saldo de una cuenta, solo a través de métodos como depositar() o retirar().

Modularidad

El sistema se organiza en módulos independientes (objetos o clases) que colaboran entre sí.

Favorece la claridad, reutilización y mantenimiento del software.

Garantiza seguridad, integridad y modularidad.

Ejemplo:

Un sistema de biblioteca, "Libro", "Socio" y "Préstamo" son módulos separados pero relacionados.

Análisis Orientado a Objetos (AOO) y UML

Principios de AOO

Jerarquía

Los objetos pueden organizarse en niveles o clases para representar relaciones del mundo real.

Se apoya en herencia y composición.

Ejemplo:

Una clase “Empleado” puede tener subclases como “Profesor” y “Administrativo”.

Herencia

Una clase puede heredar atributos y métodos de otra.

Permite reutilizar código y crear estructuras mas generales o mas especificas.

Ejemplo:

Una clase “Vehículo” puede tener subclases “Auto” y “Motocicleta”.

- CuentaAhorro hereda Cuenta

Polimorfismo

Los objetos pueden responder de manera diferente a un mismo mensaje o método.

Permite la flexibilidad y extensibilidad del sistema.

Ejemplo:

El método calcularArea() funciona distinto en un “Círculo” y en un “Rectángulo”.

- CalcularIntereses() funciona distinto en ahorro.

Análisis Orientado a Objetos (AOO) y UML

Principios de AOO

Persistencia

Los objetos deben poder mantener su estado en el tiempo, incluso después de que el programe termine.

Se logra almacenando los objetos en base de datos o archivos

Ejemplo:

un objeto **"Usuario"** conserva sus datos de registro aunque el sistema se apague.

Reusabilidad

Los objetos y clases pueden reutilizarse en diferentes sistemas o aplicaciones.

Reduce costos y tiempos de desarrollo.

Ejemplo:

Una clase "Persona" puede usarse en sistemas de hospital, universidad o banco.

Análisis Orientado a Objetos (AOO) y UML

Ventajas de AOO

Análisis Orientado a Objetos (AOO) y UML

Comparación de AOO vs. Análisis funcional

Aspecto	Funcional	Orientado a Objetos
Unidad principal	Se basa en funciones o procesos, que describen las operaciones que el sistema debe realizar.	Se centra en objetos, que combinan datos (atributos) y comportamientos (métodos).
Relación con los datos	Los datos son externos y pasivos, manipulados por distintas funciones que los procesan.	Los datos están integrados dentro de los objetos, junto con las operaciones que los usan.
Cambios de requisitos	Los cambios son más difíciles, porque al modificar un proceso se ven afectados otros (alto acoplamiento).	Los cambios son más flexibles, ya que cada objeto maneja su propia lógica y es más independiente.
Reutilización	Baja: las funciones suelen ser muy específicas y no se adaptan fácilmente a otros sistemas.	Alta: las clases pueden usarse en distintos proyectos o ampliarse mediante herencia.

Análisis Orientado a Objetos (AOO) y UML

Comparación de AOO vs. Análisis funcional

Entonces:

El AOO refleja mejor la *realidad, integra datos y comportamientos* en objetos, facilita la reutilización y el mantenimiento.

El AF es más sencillo de entender en sistemas pequeños, pero *resulta rígido y difícil de mantener en proyectos grandes.*

UML

UML (Unified Modeling Language) - > Lenguaje de Modelado Unificado

Lenguaje de modelado estándar que se utiliza para visualizar un plan arquitectónico para elementos como actividades, procesos de negocio y esquemas de base de datos.

Además:

UML, no depende de una metodología específica (ágil, en cascada, etc.) ni de un lenguaje de programación (Java, Python, C#, etc.).

De lo cual, **se puede usar UML en cualquier proyecto**, sin importar la forma en que se gestione o el lenguaje en el que se implemente.

UML

Puntos de vista del sistema

Cuando se construye un sistema de software, **cada actor involucrado** lo entiende de manera distinta:

a) El usuario final necesita una visión sencilla (qué puede hacer el sistema).

b) El analista requiere una visión funcional y estructural (qué procesos y entidades existen).

c) El programador necesita detalles técnicos (cómo se implementan los objetos, métodos, relaciones).

*UML permite mostrar **todas estas visiones** de manera organizada.*

UML

Importancia de UML

Claridad	Flexibilidad	Consistencia
Proporciona una forma clara y precisa de representar el sistema. Permite entender como funciona y como se relaciona con sus componentes	Se puede usar en todas las fases del ciclo de vida del software (análisis, diseño, implementación y pruebas)	Establece una notación estándar y común para todo el equipo. Reduce mal entendido y errores de comunicación.

UML

Importancia de UML

Documentación	Reutilización
Los diagramas UML sirven como documentación. Facilita la comprensión de sistemas heredados y la colaboración en equipo.	Permite identificar y separar componentes reutilizables. Acelera el desarrollo y reduce costos en futuros proyectos.

UML

Diagramas que se pueden representar con UML

- Diagrama de casos de uso
- Diagrama de clases
- Diagrama de secuencia
- Diagrama de componentes
- Diagrama de despliegue

UML

Categorías de diagramas de UML

Tipo de Diagrama	Propósito	Ejemplos
Estructurales	Vista estática del sistema	Clases, Objetos, Componentes
De Comportamiento	Vista dinámica	Casos de uso, Actividades, Estados
De Interacción	Comunicación entre objetos	Secuencia, Comunicación

UML Entidades y Relaciones

ENTIDAD

Una **entidad representa un objeto o cosa** del mundo real que tiene características propias.

- Generalmente, se modela como una clase en UML.

- Cada entidad posee:
  - Atributos (propiedades o datos).
  - Operaciones o métodos (comportamientos).

Ejemplo:

En un sistema bancario, una entidad puede ser Cliente con atributos (nombre, DNI) y métodos (registrar(), actualizar()).

RELACION

Son las conexiones o asociaciones entre entidades.

Permiten representar cómo interactúan o dependen entre sí los objetos.

- En UML, las relaciones pueden ser de distintos tipos:
  - Asociación → relación simple (Un cliente realiza un pedido).
  - Agregación → una entidad agrupa a otras pero pueden existir de forma independiente (Una biblioteca contiene libros).
  - Composición → relación más fuerte, las entidades dependen de la principal (Un auto tiene un motor).
  - Herencia/Generalización → una entidad hereda características de otra (Empleado → Profesor, Administrativo).

AOO vs UML

AOO es la forma de pensar.

Se basa en los objetos como unidad fundamental: combinan atributos (datos) y métodos (comportamiento).

- Principios: abstracción, encapsulación, modularidad, herencia y polimorfismo.
- Busca entender, organizar y diseñar el sistema desde la perspectiva de los objetos.

Ejemplo:
En un sistema de biblioteca → objetos: *Libro*, *Usuario*, *Préstamo*.

UML es el idioma para expresarlo

Es una herramienta de modelado, no un paradigma de programación.

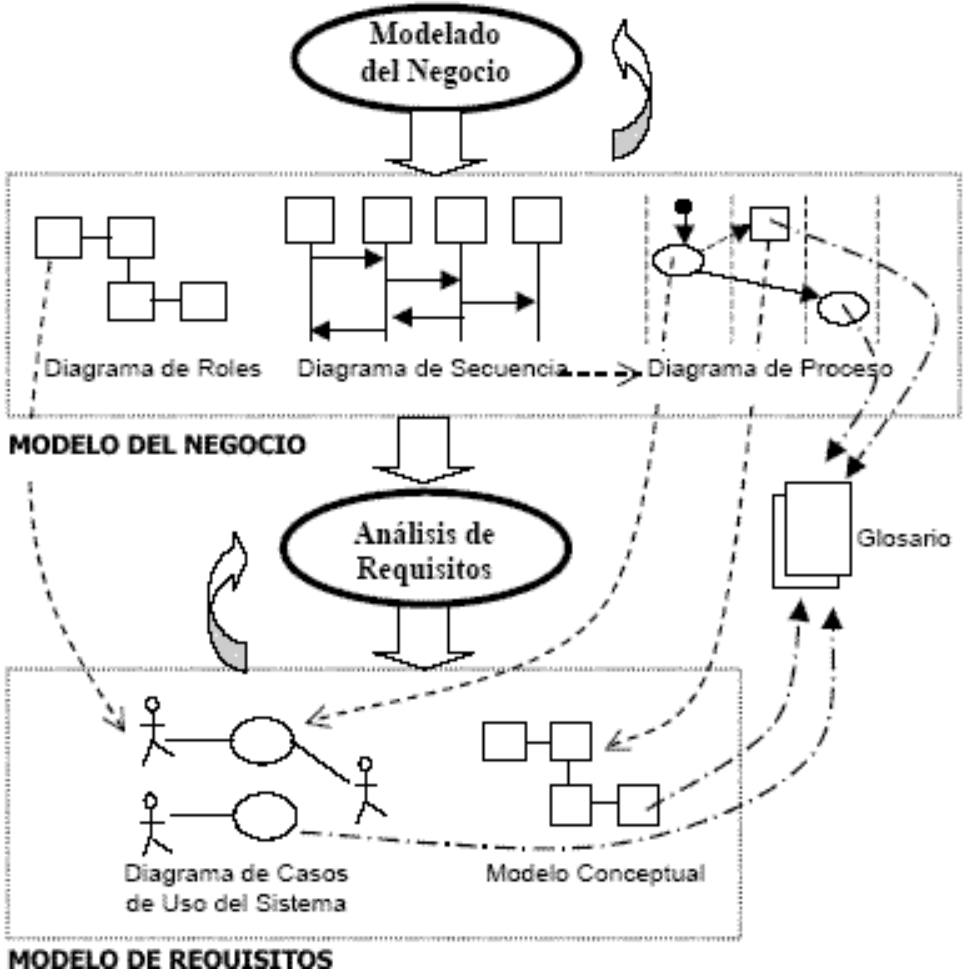
- Representa gráficamente los elementos descubiertos en el análisis (con AOO) y su diseño.
- Permite comunicar de manera visual cómo se estructura y se comporta el sistema

En la misma biblioteca → un diagrama de casos de uso mostrando cómo Usuario interactúa con Sistema de Préstamos.

Análisis de requisitos del sistema UML

El análisis de requisitos es la primera etapa del desarrollo de software donde se definen las necesidades del sistema desde la perspectiva de los usuarios y los interesados.

El UML (Unified Modeling Language) se utiliza como herramienta de modelado para representar estos requisitos de manera gráfica, clara y entendible.



Análisis de requisitos del sistema UML

Objetivos

- Identificar lo que el sistema debe hacer, sin entrar aún en los detalles técnicos de implementación.
- Comprender las necesidades de los usuarios y traducirlas en modelos claros.
- Definir límites del sistema (qué está dentro y qué queda fuera).
- Facilitar la comunicación entre usuarios, analistas y programadores

Análisis de requisitos del sistema UML

Actividades principales

1. **Identificación de actores**
   - Determinar quién interactúa con el sistema (usuarios, sistemas externos, dispositivos).
2. Definición de casos de uso
   - Especificar qué funcionalidades ofrece el sistema a los actores.
   - Representado con diagramas de casos de uso.
3. Modelado de entidades y relaciones
   - Identificar los objetos principales del dominio del problema y cómo se relacionan.
   - Representado con diagramas de clases.
4. Análisis dinámico
   - Describir cómo los actores y objetos interactúan en diferentes escenarios.
   - Usar diagramas de secuencia y diagramas de actividades.

ACTIVIDAD:

“Los sistemas de información no solo mueven datos… mueven ideas, decisiones y el futuro de las organizaciones”