Resumen metodología 1

UML

[**EL MODELO ORIENTADO A OBJETOS** 5](#_Toc116632633)

[El paradigma orientado a objeto se impuso por: 5](#_Toc116632634)

[análisis, diseño e implantación 5](#_Toc116632635)

[Conceptos: 5](#_Toc116632636)

[Análisis oo vs Diseño oo 5](#_Toc116632637)

[Abstracción y encapsulamiento 6](#_Toc116632638)

[Concepto de Herencia 6](#_Toc116632639)

[¿Qué significa herencia de atributos, métodos y asociaciones? 6](#_Toc116632640)

[¿Qué es una operación Polimórfica? 6](#_Toc116632641)

[¿Qué es generalización? 7](#_Toc116632642)

[Clases conceptuales 7](#_Toc116632643)

[El modelo de **dominio** 8](#_Toc116632644)

[como se visualiza el modelo de dominio 8](#_Toc116632645)

[Responsabilidades 9](#_Toc116632646)

[Tarjetas CRC 9](#_Toc116632647)

[Diseño OO 10](#_Toc116632648)

[Clases y Objetos 10](#_Toc116632649)

[¿Qué es una clase? 10](#_Toc116632650)

[¿Qué es un objeto? 11](#_Toc116632651)

[¿Qué diferencia existen entre ellos? 11](#_Toc116632652)

[Propiedad de los objetos 11](#_Toc116632653)

[Relaciones de Asociación 11](#_Toc116632654)

[¿Qué diferencia existe entre asociación, agregación y composición? 12](#_Toc116632655)

[**CASOS DE USO Y REQUISITOS** 13](#_Toc116632656)

[¿Qué es la ingeniería de requisitos? 13](#_Toc116632657)

[¿Qué es un requisito? 13](#_Toc116632658)

[¿Qué es un requisito de software? 13](#_Toc116632659)

[Caracteristicas deseables: ERS(Estructura del documento de requirimientos) 13](#_Toc116632660)

[¿Qué es la elicitación? 14](#_Toc116632661)

[¿Por qué es necesaria la elicitación? 14](#_Toc116632662)

[Técnicas de Elicitación 14](#_Toc116632663)

[Tipos de requisitos 14](#_Toc116632664)

[¿Quiénes son los actores en los requisitos? 15](#_Toc116632665)

[¿Cómo se relaciona los requisitos con los Casos de Uso? 15](#_Toc116632666)

[¿Qué es un caso de uso? 15](#_Toc116632667)

[Características 16](#_Toc116632668)

[Componentes de un caso de uso: 16](#_Toc116632669)

[Actores 16](#_Toc116632670)

[Tipo de actores: 16](#_Toc116632671)

[Escenario 16](#_Toc116632672)

[Pre y Post Condiciones 17](#_Toc116632673)

[Relaciones de Casos de Uso 17](#_Toc116632674)

[Asociación 17](#_Toc116632675)

[Relaciones entre Casos de Uso 18](#_Toc116632676)

[**<<Include>>** 18](#_Toc116632677)

[¿Cómo se útiliza el Include? 19](#_Toc116632678)

[<<Extend>> 19](#_Toc116632679)

[Diferencias entre include y extend 20](#_Toc116632680)

[Generalización 20](#_Toc116632681)

[Reuso de relaciones en casos de uso 20](#_Toc116632682)

[Problemas de casos de uso 20](#_Toc116632683)

[Caso de Abuso 21](#_Toc116632684)

# **EL MODELO ORIENTADO A OBJETOS**

## El paradigma orientado a objeto se impuso por:

* El aumento de complejidad de los sistemas.
* El aumento de la necesidad de reutilizar parte de códigos.
* Incorporar conceptos comunes de modelado.
* La necesidad de crear patrones de modelado y ayudar a los desarrolladores.

## análisis, diseño e implantación

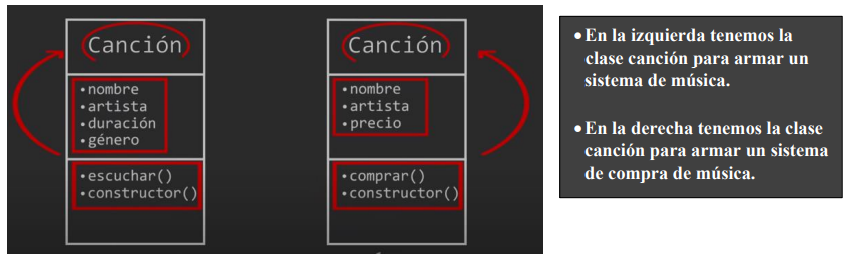
### Conceptos:

* **El análisis** pone en interés la investigación de un problema y los requisitos, donde se crea una descripción del **dominio** donde se **identificarán** conceptos**, atributos e interrelaciones** (Relación entre ambos) significativas. Nos concentramos en buscar y definir las clases, atributos y métodos.
* **El diseño** se busca **definir los objetos software** y en como **colaboran** para satisfacer los requisitos. Nos permite diseñar la lógica y el comportamiento que tendrá el sistema, también identifica como los objetos trabajan entre ellos.
* **La implantación** es la traducción de la **solución** a un lenguaje de **programación determinado.**

### Diagram Description automatically generatedAnálisis oo vs Diseño oo

* En el **análisis orientado** a objeto debemos prestar atención a encontrar y describir los objetos (**CONCEPTOS**) del problema que tenemos.
* En el **diseño orientado** a objeto debemos prestar atención a la definición de los objetos y en como colaboran para satisfacer los requisitos.

## Abstracción y encapsulamiento

* La **abstracción** es analizar las características principales (esenciales) de un objeto sin preocuparse de las no esenciales.
* El **encapsulamiento** nos permite que las propiedades de nuestro objetos se encuentren ocultas para las **distintas clases** que se encuentren en cualquier parte.

## Concepto de Herencia

La herencia es una propiedad que nos permite construir una clase a partir de una o varias clases donde se compartirán atributos y operaciones.

## ¿Qué significa herencia de atributos, métodos y asociaciones?

Esto significa que la clase hija tiene los mismos atributos, métodos y asociaciones que la clase padre.

## ¿Qué es una operación Polimórfica?

Una operación polimórfica (*Muchas formas*, poder actuar de formas diferentes.) ocurre cuando **los objetos** heredaron los atributos o métodos de la **clase** que lo creo, el objeto redefinirá los valores de los atributos o métodos, a esto se lo llama operación polimórfica.

***Se utiliza para modificar los valores de los atributos recibidos de una clase, nos permitirá realizar la misma operación, pero de diferente forma según el objeto que se esté utilizando.***

Diagram

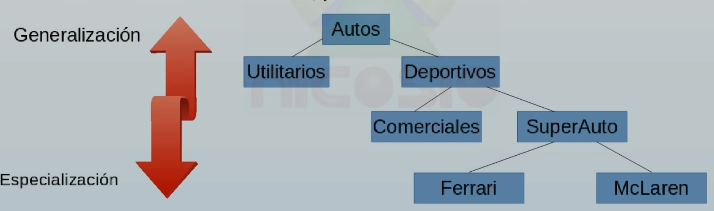
Description automatically generated

## ¿Qué es generalización?

La generalización es una relación entre un **objeto general** y otro más específico.

El beneficio que tiene la generalización es que nos permite **organizar estructuralmente las abstracciones.**

* La **clase hija** se basa en la **clase padre**
* Las dos clases son similares, pero **tienen algunas diferencias**.



## Clases conceptuales

Las clases de forma conceptual se las pueden considerar de la siguiente manera:

* **Símbolo**: Palabras como (protected, private) o imágenes que representan la **clase conceptual.**

Graphical user interface, application

Description automatically generated with medium confidence

* **Intensión:** Es la definición de la **clase conceptual.**

Table

Description automatically generated with low confidence

* **Extensión:** Son los conjunto de ejemplos a los que se aplica la clase conceptual.

Text

Description automatically generated with medium confidence

## El modelo de **dominio**

Diagram

Description automatically generatedEs una visualización de los conceptos del mundo real y como se vinculan, estos se representan mediante **diagramas de clases**.

El modelo de **dominio** nos sirve para:

* Identificar y representar conceptos del dominio de problema.
* Establecer y entender las relaciones entre los conceptos.
* Identificar atributos de cada concepto.
* Representar entidades en el modelado de procesos.

### como se visualiza el modelo de dominio

Diagram

Description automatically generated

**Atributos** de las clases conceptuales.

**Asociaciones** entre clases conceptuales.

Clase conceptual

## Responsabilidades

Una **responsabilidad** es una obligación de una clase, las responsabilidades se verán a medida que se va armando el **modelo,** podremos ver conjuntos de atributos y operaciones que logren cumplir esas responsabilidades.

**Por ejemplo,** que nuestro sistema obligatoriamente debe tener un **sistema de devolución.**

## Tarjetas CRC

Estas son **COLABORACIÓN – RESPONSABILIDAD- CLASE**

**Sirve para descubrir e identificar clases.**

¿Para qué sirve?

* Permite **ver las clases** y como interactúan entre ellas de una forma fácil.
* La lluvia de ideas es una buena práctica para sugerir como rellenar las **tarjetas CRC.**
* Todas las ideas son buenas, no se censuran.
* Cada miembro tiene su turno y no es presionado, cualquier idea que brinde servirá para rellenar las **tarjetas CRC.**

¿Cómo se realizan?

* La técnica consiste en **crear una tarjeta** por cada clase u objeto, cada una de estas se **divide en tres zona.**
* **En la parte superior** se **coloca** el nombre de la clase. El nombre tiene que reflejar lo que hace esta clase.
* En el **lado izquierdo** se coloca las **responsabilidades** de dicha clase. Determinan que debe hacer, estas responsabilidades se encuentran en dos categorías: **hacer** y **conocer.**
* Table

  Description automatically generatedEn el **lado derecho** se colocan los **colaboradores** los cuales son los que ayudan a cumplir a estas con sus **responsabilidades**.

## Diseño OO

El diseño orientado a objeto es definir los **objetos software** y sus colaboraciones, aquí se utilizan **diagramas de interacción.**

* Una diferencia que tiene este del **modelo del dominio** es que este modelo no **muestra** conceptos del mundo real, sino que utiliza **clases software**, con atributos, operaciones y asociaciones.

## Clases y Objetos

### ¿Qué es una clase?

Una clase es una **plantilla** que nos permite crear múltiples objetos que tendrán las mismas características.

### ¿Qué es un objeto?

Un objeto es una instancia de una clase, y es alguna “**COSA**” concreta que ha sido creada usando una **clase especifica.**

### ¿Qué diferencia existen entre ellos?

La diferencia que existen entre ellos es que un **objeto** es una instancia de una clase, la cual **el** **objeto** **tomará todos sus atributos**, la **clase** es una plantilla para crear diferentes tipos de objetos según lo que se esté tratando de desarrollar.

### Propiedad de los objetos

* **El estado** de un objeto son los valores que poseen sus propiedades, (**se puede decir que** **son los datos asociados al objeto).**
* **El comportamiento** de un objeto es como actúa y reacciona un objeto, esto quiere decir que son las cosas que puede hacer nuestro objeto.
* **La identidad:** es la forma que se puede distinguir de otro objetos.

## Relaciones de Asociación

* **Los enlaces** entre objetos son instancias de la **asociación** entre sus clases.

Esto quiere decir que **dos clases** están **relacionadas** o **conectadas** en alguna forma.

### Diagram Description automatically generatedMultiplicidad

Nos indica la **cantidad de objetos** **que** **participan** **en una relación.**

Pueden ser de 1 a 1 o 1 a muchos(n).

## ¿Qué diferencia existe entre asociación, agregación y composición?

Todas estas son **formas de relacionar objetos**.

* Graphical user interface, application

  Description automatically generated**La asociación** es una relación que describe una conexión entre objetos, estas pueden relacionarse en ambas direcciones , pero lo mejor es que tenga un solo sentido:
* **La Agregación** es un tipo de asociación que sirve para indicar que **una clase** es parte de **otra clase.** La existencia del objeto **agregado** es **independiente** del objeto **contenedor.**

Diagram

Description automatically generated

* A picture containing text, clock, device, meter

  Description automatically generated**La composición** son los componentes de una clase, si desaparece el **contenedor**, desaparecen **las partes.**

# **CASOS DE USO Y REQUISITOS**

## ¿Qué es la ingeniería de requisitos?

La ingeniería de requisitos se busca conocer los requisitos que necesita algún cliente, este direcciona el proceso de **elicitación**, definición, modelado, análisis y especificaciones de su software.

La ingeniería de requisitos se centrará en dos aspectos principales, se busca conocer el propósito de lo que se va a desarrollar y saber el contexto en el que será usado.

## ¿Qué es un requisito?

En sistemas, un requisito es una **capacidad** o **condición** que debe tener un sistema para lograr cumplir las **necesidades del usuario.**

## ¿Qué es un requisito de software?

Un requisito de software es la capacidad que debe tener el software para cumplir las necesidades del usuario o cumplir un objetivo para la toma de decisiones.

### Caracteristicas deseables: ERS(Estructura del documento de requirimientos)

* **Completa:** Los requerimientos y referencias deben estar definidos.
* **Consistente:** Debe ser coherentes sus requerimientos y sus documentos de especificación.
* **Inequívoca:** Su redacción debe **clara** de modo que no se mal interprete.
* **Correcta:** El software tiene que cumplir con los requisitos de la especificación.

## ¿Qué es la elicitación?

La elicitación es un forma de **obtener** los **requisitos** relevantes para un sistema

## ¿Por qué es necesaria la elicitación?

La elicitación es necesaria para lograr entender **el dominio del problema en particular**.

## Técnicas de Elicitación

Las técnicas para obtener requisitos son:

* Entrevistas.
* Brainstorming (Lluvia de ideas).

## Tipos de requisitos

Los tipos de requisitos son:

* **Requisitos de usuarios:** Son las necesidades que el usuario dicen de forma verbal.
* **Requisitos del Sistema:** Son los componentes que el sistema debe tener para lograr realizar determinadas tareas.

Para los ejemplos se usará un **sistema de venta de música online**.

* **Requisitos funcionales:** Son los servicios que el sistema **debe proporcionar**.
  + **Ejemplo:** 
    - Los usuarios comprarán créditos para adquirir canciones
    - Los usuarios buscarán canciones que deseen y las pagarán con dinero.
* **Requisitos no funcionales:** Son las **restricciones** que **afectan** alsistema.
  + **Ejemplo:**
    - El sistema no **puede** **tardar** más de cinco segundos en **mostrar** resultados, si pasa esto el sistema automáticamente **detiene** la búsqueda y **mostrará** los resultados encontrados.

## ¿Quiénes son los actores en los requisitos?

* **Usuario final:** Son las personas que utilizarán el sistema.
* **Usuario líder:** Son las personas que conocen el sistema y dan información al **equipo técnico.**
* **Equipo Técnico:** Actualizan y mejoran el sistema.
* **Programadores y analistas:** Los principales desarrolladores.

## ¿Cómo se relaciona los requisitos con los Casos de Uso?

Los casos de uso son una **técnica** para la **especificación de requisitos**.

Esto quiere decir que los casos de uso son requisitos funcionales.

## ¿Qué es un caso de uso?

Los casos de uso son una técnica para **documentar** los **requerimientos funcionales** de un sistema desde el punto de vista de los usuario.

* Responden a: ¿Qué tiene que hacer el sistema para los usuarios?

A picture containing diagram

Description automatically generated

* **Describe lo que hace el actor como el sistema cuando interactúa con él.**

### Características

* Representa como se comporta un sistema desde la perspectiva del usuario para lograr alcanzar los objetivos de negocio.
* Describe lo que el software **hace**, y no la manera en que lo hace.
* Se **componen** en una representación gráfica y una **especificación funcional.**
* Se escribe desde la **perspectiva del actor.**
* Describe la **interacción** del actor y el sistema.

### Componentes de un caso de uso:

Se compone de:

* **Diagrama**
  + Actores.
  + Funcionalidades (Casos de uso).
  + Relaciones
* **Especificación**
  + Escenarios.
  + Restricciones.

### Actores

Un actor interactúa con el sistema, con algún evento o recibe información del sistema.

* Un actor cumple algún rol (**Cliente, Banco, Empleado**)**.**

### Tipo de actores:

* **Actores primarios:** Son los que utilizan las funciones principales.
* **Actores secundarios:** realizan tareas administrativas o mantenimiento.

### Escenario

Un escenario es una **secuencia especifica** de acciones entre los actores y el sistema.

* Cada **secuencia específica** del caso de uso se denomina **Escenario.**

Ejemplo:

* Un caso de uso **Contratar Empleado** tiene variantes:
  + Contratar Externos (Escenario frecuente).
  + Contratar Extranjeros.

Estas **variantes** se expresan en una **secuencia diferente** y cada una de esas secuencias son diferentes **escenarios**.

### Pre y Post Condiciones

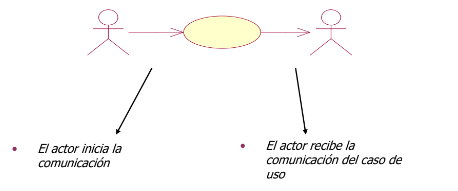
* **Precondiciones:** Que debe suceder para que el caso de uso se lleve a cabo.
* **Postcondiciones:** Que sucede después de realizar el caso de uso.

## Relaciones de Casos de Uso

### Asociación

Muestra la relación entre los **actores** y los **casos de uso**.

Los **actores** se conectan únicamente a los **casos de uso** a través de **asociaciones**, este indica que el actor y el caso de uso se comunican entre sí y pueden enviar y recibir mensaje.



### Relaciones entre Casos de Uso

Dos casos de uso que se encuentren ligados por una **asociación**, pueden ser leídos o ejecutados como un solo Caso de uso.

### **<<Include>>**

Una relación de **inclusión** entre casos de uso nos dice que **un caso de uso base incorpora explícitamente el comportamiento de otro caso de uso**.

La relación de inclusión es representada como una **dependencia.**

La **dependencia** es una relación que **declara** que un caso de uso utiliza la **información** y **servicios** de otro.

Text

Description automatically generated with medium confidence

Conclusión: Cuando relacionamos dos casos de uso con un “**include**” decimos que el **caso de uso base** incluye al segundo, el segundo es **esencial** del **primero,** sin el segundo, el primero no podría cumplir su objetivo.

#### ***Ventajas y Desventajas de esta asociación***

* **Ventajas:**
  + Las **descripciones** de los casos de uso son cortas y entendibles.
  + La **identificación** de **funcionalidades** común puede ayudar a descubrir el posible uso de **componentes** **existente** en la **implementación**.
* **Desventajas:**
  + Esta relación es más difícil de leer para los clientes.

##### Ejemplos de <<include>>

Diagram

Description automatically generated with low confidence

Diagram

Description automatically generated

### ¿Cómo se útiliza el Include?

Lo utilizamos para evitar repeticiones de **descripción de flujos de eventos,** también cuando un caso de uso **incorpora** explícitamenteel comportamiento de otro caso de uso.

### <<Extend>>

Al igual que el include, la relación de extensión entre casos de uso nos dice que un **caso de uso base** incorpora el comportamiento de otro caso de uso.

La diferencia, es que el extend hay situaciones que el caso de uso de **extensión** no sea necesario que ocurra, y cuando lo hace **ofrece** un valor extra (**Extiende**) al caso de uso.

Diagram

Description automatically generated with medium confidence

En este ejemplo se utiliza un **extend** porque se puede realizar una venta sin **acumular puntos VIP**, por lo tanto, es una extensión de **realizar venta** y se puede llegar a ejecutar en algún tipo de ventas.

### Diferencias entre include y extend

La diferencia que existe entre estos es que el include es necesario que ocurra el caso que se está **incluyendo**, pero el extend no es necesario que ocurra, es algo **opcional**, y si llegase a ocurrir simplemente ofrece un valor extra al caso de uso.

### Generalización

La generalización es aplicable para **actores** como para los **casos de uso**, esta nos dice que un caso de uso hereda el comportamiento y significado de otro.

### Reuso de relaciones en casos de uso

Podemos reutilizar un **caso de uso** cuando estamos relacionándolo con otro, esto nos ayudará a que sea más fácil de entender:

Diagram

Description automatically generated

## Problemas de casos de uso

* Los límites del sistema no están definidos.
* Los nombre **describen** lo que el **sistema** **hace** en lugar de **describir** lo que el **actor tiene como objetivo**.
  + Solución: Debemos **nombrar** los casos de uso desde la perspectiva del actor.
* Muchos casos de uso.
* Demasiadas relaciones entre **actores y caso de uso.**

### Caso de Abuso

Los casos de abuso surgen cuando **un modelo** **tiene** muchas **inclusiones** y **extensiones** las cuales algunas son innecesarias.

En resumen:

* Demasiadas Relaciones entre Actores y Caso de uso.
* **Un actor** interactúa con **cada** Caso de uso o **un caso de uso** interactúa con **cada** actor.

Diagram

Description automatically generated

#### ***SOLUCIÓN A LOS CASOS DE ABUSOS***

Para solucionar estos tipos de casos, podemos **volver** a escribir el caso de uso, pero esta vez centrándonos en las interacción usuario - sistema **más esenciales**.