## Modelo Orientado a Objetos.

* El análisis OO (Orientado a objetos) esta enfocado en la investigación del problema y los requisitos.
* El diseño está enfocado en una solución conceptual, esta satisface lo pedido, pero no lo implementa.
* La implementación es la traducción de la solución a algún lenguaje de programación.

## Análisis OO vs Diseño OO.

* Durante el análisis se encuentran y describen el concepto de los objetos del dominio del problema.
* Durante el diseño se definen los objetos (software) y en como colaboran para satisfacer los requisitos.

## Análisis OO.

* La finalidad del análisis es crear una descripción del dominio desde una perspectiva de clasificación de objetos: identificación, atributos, interrelaciones significativas.
* El modelo de dominio NO es una descripción de los objetos de software, es una visualización de los conceptos del mundo real y sus vinculaciones.

## Abstracción.

* Es la propiedad que permite representar las características esenciales de un objeto, sin preocuparse por los demás.

## Encapsulamiento.

* Es la propiedad que permite asegurar que el contenido de la información de un objeto esta oculta al mundo exterior. (privatizar)
* Al separar el comportamiento del objeto de sui implementación, permite la modificación de este sin que se tengan que modificar las aplicaciones que lo utilizan.

## Clases conceptuales.

* Símbolo: Palabras o imágenes que representan la clase conceptual.
* Intensión: La definición de la clase conceptual.
* Extensión: Conjunto de ejemplos a los que se aplica la clase conceptual.

## Modelo de dominio.

* Análisis: Descomposición de un dominio de interés en clases conceptuales.
* Modelo de dominio: representación visual de las clases conceptuales del mundo real.
* Clase conceptuales.
* Asociaciones entre clases conceptuales.
* Atributos de las clases conceptuales.

## Responsabilidades.

* Una responsabilidad es un contrato u obligación de una clase.
* Una clase puede tener x numero de responsabilidades, pero debería tener al menos 1 y como mucho unas pocas.
* Se describen inicialmente con texto libre.

## Diseño OO (ORIENTADO A OBJETOS).

* La finalidad del diseño es definir los objetos software y sus colaboraciones.
* A diferencia del m0delo del dominio este no muestra conceptos del mundo real, sino clases software.

## Objetos.

* Un objeto es cualquier cosa real o abstracta, acerca de la cual almacenamos datos y las operaciones que controlan esos datos.

### Sus propiedades:

* El estado de un objeto abarca todas las propiedades (estáticas) del mismo y los valores actuales (dinámicas) de cada una de esas propiedades.
* El comportamiento nos muestra cómo actúa y reacciona un objeto, en términos de su cambio de estado y paso de mensajes.
* La identidad es lo que distingue de los demás objetos.

## Clases.

* Una clase especifica una estructura de datos y las operaciones que son permitidas para aplicar a cada uno de los objetos.
* Los objetos están vinculados por enlaces, que estos mandan mensajes a operaciones para activar los métodos.
* Mensaje: Es una solicitud para que se haga la operación indicada y se produzca el resultado.
* Operaciones: Es una función o transformación que se aplica al objeto de una clase.
* Métodos: Es como se implementa esa operación.

## Relaciones de asociación.

* Los enlaces entre objetos son instancias de la asociación entre sus clases.
* La asociación tiene un acoplamiento débil, en cambio la agregación y la composición tienen un acoplamiento más fuerte en las clases.

## Relaciones de jerarquía.

* La generalización es agrupar los elementos comunes de un grupo de clases y meterlas en una clase general que se le dice “Superclase”.
* La herencia es una técnica de los lenguajes de programación que consiste en construir una clase compartiendo atributos y/o operaciones de una u otras clases.
* Polimorfismo: Permite ejecutar distintas operaciones con la respuesta de un mismo mensaje.
* Permite referirse a objetos de otras clases mediante el mismo elemento del programa y realizar una misma operación de diferentes formas.
* La interacción que tienen las clases se escriben según los términos de las especificaciones que tienen las superclases.
* Existe un mismo método pero la forma de realizarlo en cada clase es diferente.

## Análisis Estructurado vs Análisis OO.

* En análisis estructurado se descompone el problema en funciones o procesos y estructuras de datos.
* En el análisis OO se busca descomponer el problema no en funciones, sino en partes más pequeñas llamadas objetos.

## Requisito.

* Es una condición o capacidad que el sistema debe cumplir.
* Entender de forma clara la necesidad del usuario.

## Requisito de software.

* Es una descripción completa de como se va a comportar el sistema que se va a desarrollar.
* Una capacidad de software que debe ser reunida o poseída por un sistema para cumplir con un contrato, especificación, estado u documentación formal.

## Requisitos de Casos de Uso.

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

## Casos de Uso.

* Los casos de uso son una técnica para documentar los requerimientos funcionales de un sistema desde el punto de vista del usuario.
* Se utiliza para saber qué es lo que debe hacer el sistema para los usuarios.
* Este muestra cómo se debería comportar el sistema cuando un usuario interactúe con este.
* Describe lo que hace el actor (usuario) y el sistema cuando interactúen con el.

## Actores.

* Alguien o algo que interactúa con el sistema.
* Este estimula al sistema con algún elemento o recibe información del sistema.
* Hay dos tipos de actores:

1. Actor primario: Este usa las funciones principales.
2. Actor secundario: Este hace tareas administrativas o de mantenimiento.

## Escenario.

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

## Pre y Post condiciones.

* Pre condiciones: Este establece lo que se debe cumplir antes de empezar un caso de uso. Estas no se prueban en el caso de uso, se da por hecho que son verdaderas.
* Post condiciones: Este establece que se tiene que cumplir cuando el caso de uso finaliza con éxito.

## Casos de uso – Asociación.

* Los actores solo se pueden conectar con casos de usos a través de asociaciones, estas indican el actor y el caso de uso se comunican entre si y que pueden enviar y recibir mensajes.

## Relaciones entre casos de uso.

* Dos casos de uso que estén conectados por una asociación pueden ser leídos o ejecutados como un solo caso de uso.

## Include.

* Un caso de uso incorpora explícitamente el comportamiento de otro caso de uso.
* Este se representa como una dependencia.
* Una dependencia es una relación de uso, esta declara que un caso de uso utiliza información y servicios de otro caso de uso.

### Ventajas:

1. Tienen descripciones cortas y entendibles.
2. La identificación de funcionalidades puede ayudar a descubrir nuevas implementaciones con componentes ya creados.

### Desventajas:

1. Es mas difícil de leer para los clientes.
2. Esta evita descripciones repetitivas para los flujos de eventos.

## Extend.

* Un caso de uso incorpora explícitamente otro caso de uso.
* La funcionalidad de un caso de uso tiene un conjunto de pasos que solo ocurren en algunas ocasiones.
* El extend consiste en interrumpir el caso de uso B y pasar a ejecutar el caso de uso A.
* Este se usa en puntos de una condición.
* Se ejecuta en ciertos casos.

## Generalización.

* Este se puede utilizar tanto para actores como casos de usos. Lo que hace es que adquiere todas las funcionalidades del actor o caso de uso que herede.

Probelmas frecuentes.

* Los limites del sistema no están definidos.
* Los nombres describen lo que hace el sistema, no lo que el actor tiene como objetivo.
* El modelo de este parece un diagrama de flujos.
* Los nombres de los actores no son muy importantes, porque diferentes nomres pueden definir el mismo rol.
* Tienen muchos casos de uso.
* Las relaciones que hay entre los actores y los casos de uso parecen “telarañas”.
* Estos tienen muchas relaciones.
* Un actor/caso de uso puede interactuar con muchos casos de uso/actores.

Casos de abuso. (que no hacer con casos de uso)

* Un caso de abuso es cuando un modelo tiene muchas inclusiones y extensiones que son innecesarias.
* NO HAY que representar la forma de moverse de las funciones con casos de uso.
* No hay que hacer una descomposición funcional, hay que representar de una forma grafica lo que el usuario podrá hacer en el sistema.

Otros problemas.

* Que las especificaciones de los casos de uso sean muy extensas.
* Que la especificación sea confusa.
* Que tenga poco contexto, los pasos se van a ver como un programa de PC.
* El caso de uso no describe de forma correcta el requisito formal.
* Las relaciones no describen de manera correcta como alguien puede utilizar el sistema.
* Que el cliente no entienda el caso de uso.
* Que los casos de uso no terminen.

Visibilidad de Atributos y Operaciones.

* Publica: Cualquier clase puede utilizar sus características.
* Privada: Cuando solo una clase puede utilizar su característica.
* Protegida: Cualquier descendiente de una clase puede usar esta característica.
* Paquete: Solo las que estén declaradas dentro del paquete pueden utilizar esta característica.

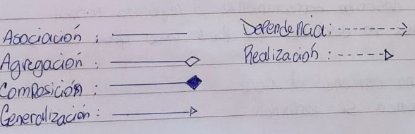
Instancia.

* Es una manifestación concreta de una bastraccion en la q2ue se pueden hacer un grupo de operaciones, esta tienen un estado que almacena el efecto que generan esas operaciones.

Interfaz.

* Es un grupo de operaciones que te especifican que servicios te brinda una clase.
* Explica el comportamiento visible por el exterior que tiene la clase.
* Define un conjunto de especificaciones y operaciones, pero no como se va a implementar.
* No puede tener instancias directas.
* La realización es la relación entre los clasificadores.

Relaciones.



Dependencias.

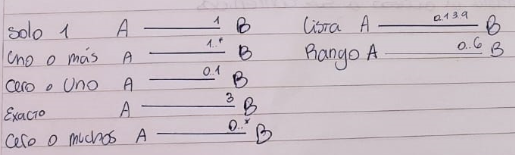
* Es una relación donde se declara que un elemento está utilizando información de otro elemento.
* Generalmente se utilizan en el contexto de las clases para indicar que una esta utilizando las operaciones de otras.

Asociación.

* Es una relación estructural en la que se indica que los objetos de una clase están conectados con los objetos de otra clase.

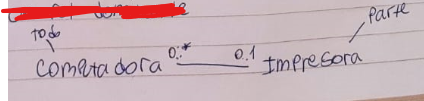
Multiplicidad.

* Define cuantas instancias de una clase A pueden asociarse con una instancia B.
* Esta representa un rango del tamaño posible del conjunto los objetos relacionados.



Agregación.

* Tipo TODO/PARTE.
* Es una relación no simétrica entre clases donde el extremo cumple el rol dominante.



Composicion.

* Es una forma de agregación con una fuerte relación de pertenencia y vidas concidentes de la parte del todo.
* El elemento desaparece al destruirse lo que lo contiene.

Navegacion.

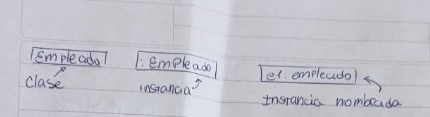
* Este indica que en una asociación es posible navegar de un objeto de un tipo a otro.

Paquetes.

* Es un mecanismo que ordena el modelo de una forma jerárquica.
* Estos elementos se organizan para poder entenderlos.
* Esto evita problemas con los nombres.
* Se facilita el reconocimiento y búsqueda de los elementos que cumple una función especifica.
* Controla el acceso a sus contenidos.

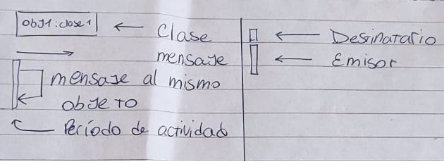
Diagrama de interacción.

* Muestran de que modo los elementos se comunican entre si a través de mensajes.
* Diagrama de comunicación: Esta hace énfasis a la organización estructural de los objetos.
* Diagrama de secuencia: Esta hace énfasis en la organización temporal.
* Si un objeto envia un mensaje a otro quiere decir que hay una relación de clases (en el diagrama de clases) entre ellos y que son visibles el uno para el otro.



Diagramas de secuencia:

* Este destaca por que ordena sus mensajes de forma temporal.



DSS (DIAGRAMA DE SECUENCIA DE SISTEMA).

* Estos se utilizan en la etapa de análisis para documentar a los diagramas de casos de uso.
* El actor efectúa eventos sobre el sistema, generando una operación como respuesta.
* Hay un DSS por cada escenario.
* Esta muestra los eventos que generan los actores externos.
* Los sistemas se tratan como cajas negras.

Diagrama de comunicación.

* De este destaca la organización estructural de sus objetos y de su envió de mensajes.
* Muestra las interacciones en torno a sus objetos y los enlaces que hay entre ellos.
* Este se usa en la fase de diseño para dar responsabilidad a las clases.

Enlaces.

* Es una conexión entre objetos, este indica que se puede navegar entre los objetos que estén “Enlazados” o que los objetos pueden estar visibles uno del otro.
* Es una instancia de asociación.
* Los mensajes se representan mediante una flecha con un nombre.