CONCEPTOS BÁSICOS DE ORIENTACIÓN A OBJETOS

Importancia del modelado

El modelado es una parte central de todas las actividades que conducen a la producción de buen software.

¿Qué es un modelo?

Un modelo es una representación simplificada de la realidad. Es un instrumento que pone de manifiesto determinados elementos que considera significativos del fenómeno analizado, por ello todo modelo es una mirada reduccionista de la realidad.

¿Por qué modelamos?

Construímos modelos para comprender mejor el sistema que estamos desarrollando, reduciendo su complejidad.

Las dos formas más comunes de enfocar el modelado en el desarrollo de software son la perspectiva orientada a objetos y la perspectiva algorítmica.

La visión tradicional del desarrollo de software toma una **perspectiva algorítmica**, basada en el procedimiento o **función**.

La visón actual del desarrollo de software toma una **perspectiva orientada a objetos** (OO), en este enfoque, el principal bloque de construcción de los sistemas es el **objeto**.

¿Por qué orientado a objetos (OO)?

Para la mayoría de las personas, la forma de pensar OO es más natural que las técnicas del análisis y el diseño estructurado. Después de todo el mundo está formado por objetos. Comenzamos a aprender sobre ellos desde la infancia (al agitar un sonajero nos damos cuenta de que hace ruido) y descubrimos que tienen cierto comportamiento. Si se piensa un poco, esto tiene sentido porque desde muy temprana edad **categorizamos los objetos y descubrimos su comportamiento.**

Antes que nada definimos a un **objeto como una instancia de una clase o categoría**:

Usted y yo, por ejemplo, somos instancias de una **clase** Persona.

Un objeto cuenta con:

- una estructura, es decir, propiedades que llamaremos **atributos**.
- acciones o actividades que el objeto es capaz de realizar, que las llamaremos **operaciones**.

Los atributos y operaciones, en conjunto, se conocen como **características** o **rasgos**.

Como **objetos** de la **clase** Persona, usted y yo contamos con los siguientes **atributos**: altura, peso, edad, color, etc.

También realizamos actividades (**operaciones**): comer, leer, hablar, escribir, etc.

Si tuviéramos que crear un sistema que manejara información acerca de las personas, como una nómina de un sistema de Recursos Humanos, sería muy probable que incorporáramos algunos de estos atributos y operaciones a nuestro software.

En nuestra OO una **clase** tiene otro propósito, además de la **categorización**, es en realidad una **plantilla** para fabricar objetos. Se podría hacer una comparación con el molde para hacer galletitas, este molde sería la clase y las galletitas (sus instancias) los objetos.

Otros conceptos:

Pero la OO se refiere a algo más que a clases y objetos (con sus atributos y operaciones), también considera otros aspectos como la **abstracción** , **herencia**, **polimorfismo**, **encapsulamiento**, etc.

Abstracción: es el acto o resultado de eliminar diferencias entre objetos de modo que podamos ver los aspectos comunes.

Todo objeto es único, sin embargo, mediante la abstracción, desde pequeños, eliminamos algunas diferencias para poder ver aspectos comunes entre los objetos. Así llegamos al concepto de árbol, botella, etc. Sin la abstracción, sólo sabríamos que cada cosa es diferente de las demás. Con la abstracción se omiten de manera selectiva varias características distintivas de dos o más objetos, lo que nos permite concentrarnos en las características que comparten.

Herencia: es el mecanismo por el que elementos más específicos incorporan la estructura y comportamiento de elementos más generales.

La herencia tiene que ver con los conceptos de generalización y especialización. La **generalización** consiste en factorizar los elementos comunes (atributos y operaciones) de un conjunto de clases en una clase más general llamada superclase.
La generalización es un método que exige una buena capacidad de abstracción. Los árboles de clases no crecen a partir de su raíz; se determinan partiendo de las hojas porque éstas pertenecen al mundo real mientras que los niveles superiores son abstracciones construídas para ordenar y comprender.

Un objeto es una instancia de una clase, por lo tanto, hereda sus características.

Polimorfismo: en ocasiones una operación tiene el mismo nombre en diferentes clases, pero el método (la forma en que la operación es realizada) es distinto. En la OO a esto se le llama polimorfismo.

Encapsulamiento: es el resultado (o acto) de ocultar los detalles de implantación de un objeto respecto de su usuario.

Por lo general, la mayoría de la gente que ve televisión no sabe o no se preocupa de la complejidad electrónica que hay detrás de la pantalla, ni de todas las operaciones que tiene que ocurrir para mostrar una imagen en la pantalla. El televisor sabe lo que tiene que hacer, sin mostrarnos el proceso necesario para ello.

¿Cuál es la importancia del encapsulamiento?

Su importancia es que permite reducir el potencial de errores que pudieran ocurrir. En un sistema que consta de objetos, éstos dependen unos de otros. Si uno de ellos falla, y los especialistas de software tienen que modificarlos de alguna manera, el ocultar sus operaciones de otros objetos significará que tal vez no será necesario modificar los demás objetos. Por ejemplo el monitor de de su computadora, en cierto modo, oculta sus operaciones a la CPU, es

decir, si algo falla en el monitor, lo reparará o reemplazará, pero es muy probable que no tenga que reparar o reemplazar la CPU al mismo tiempo que el monitor.

Un objeto a través del encapsulamiento oculta lo “que hace” a los otros objetos y al mundo exterior. Por ese motivo al encapsulamiento también se lo conoce como **ocultamiento de la información**.

Pero un objeto tiene que presentar un rostro al mundo exterior para poder iniciar sus operaciones, por ejemplo los botones, perillas y control remoto de la televisión. Este “rostro” se conoce como **interfaz**.

Interfaz: colección de operaciones que se utiliza para especificar el servicio de una clase.

Envío de mensajes: en un sistema los objetos trabajan en conjunto. Esto se logra mediante el envío de mensajes entre ellos.

Un objeto envía a otro un mensaje para realizar una operación y el objeto receptor ejecutará la operación.

Un televisor y su control remoto, puede ser un ejemplo muy intuitivo del mundo que nos rodea. Cuando desea ver un programa de TV, busca el control remoto y presiona el botón de encendido. ¿Qué ocurre? El control remoto le envía literalmente un mensaje al televisor para que se encienda. El televisor recibe el mensaje, lo identifica como una petición para encenderse y así lo hace. Cuando desea ver otro canal, presiona el botón correspondiente del control remoto, el cual envía otro mensaje al televisor para cambiar de canal. El control remoto puede enviar otros mensajes, como bajar el volumen, aumentar brillo, etc.

Sobre este ejemplo del televisor pensemos en las interfaces:

Muchas de las cosas que hace el control remoto también las puedo hacer desde el panel frontal del televisor.

Las interfaces que la televisión presenta (conjunto de botones y perillas) no es la misma que muestra al control remoto (un receptor de rayos infrarrojos).

Asociación: relación estructural que describe un conjunto de enlaces, donde un enlace es una conexión entre objetos.
Relación semántica entre dos o más clasificadores que implica la conexión entre sus instancias.
¿Qué queremos decir con esto? Los objetos se relacionan entre sí de alguna forma. Por ejemplo: cuando enciende su televisor, en términos de OO, usted se asocia con su televisor. La asociación de “encendido” es en una sola dirección. Pero hay otras asociaciones en dos direcciones, como el matrimonio, por ejemplo.

Multiplicidad: especificación del rango de cardinalidad permisible que puede asumir un conjunto.
Indica la cantidad de objetos de una clase que se relacionan con un objeto en particular de la clase asociada. Por ejemplo, un curso escolar, sólo lo imparte un instructor, por lo tanto el curso y el instructor están en una relación uno a uno. En cambio en un curso tipo seminario hay varios instructores, o sea que entre el curso y el instructor están en una relación uno a muchos.

Agregación: forma especial de asociación que especifica una relación del tipo “todo-parte” entre el agregado (todo) y la parte.

Su computadora es un ejemplo de agregación, es otro tipo de asociación entre objetos, es decir su equipo está constituído de diversos elementos.

Un tipo de agregación trae consigo una estrecha relación entre un objeto agregado y sus componentes. A esto se lo conoce como **composición**.

Composición: forma de agregación con fuerte pertinencia y un tiempo de vida coincidente entre el todo y las partes. Las partes, con una multiplicidad no fija, pueden ser creadas después del propio compuesto, pero una vez creadas viven y mueren con él. Tales partes pueden ser eliminadas antes que el compuesto.

Todos estos conceptos que hemos visto hasta ahora de la OO, nos ayudan a comprender el *área de conocimiento* de su cliente.

Dominio: área de conocimiento o actividad que se caracteriza por un conjunto de conceptos y una terminología (semántica) que entienden los profesionales y usuarios de esa área.

Es para esa área de conocimiento que existe el Lenguaje Unificado de Modelado (**UML**).

UML es una herramienta de modelado. Es un lenguaje estándar para escribir planos de software creado por Grady Booch, James Rumbaugh e Ivar Jacobson.

UML es un lenguaje para:

- visualizar
- especificar
- construir
- documentar

los artefactos de un sistema con gran cantidad de software.

Artefacto: pieza de información que es utilizada o producida por un proceso de desarrollo de software.