“UNIVERSIDAD NACIONAL DE INGENIERIA”

“Facultad de Ingeniería Industrial y de Sistemas”

Alumno: Vidal Flores, Jose Carlos

20130078D

Dirigido: HANCCO CARPIO, RONY JORDAN

Curso: LENGUAJES DE PROGRAMACIÓN ORIENTADOS A OBJETOS (ST-232 V)

“2015-I”

UML

El Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño.

La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de modelado y no así el proceso que se siguió para obtenerlo.

Los modelos son representaciones visuales de los sistemas a analizar, haciendo una analogía los modelos son los planos para la construcción de un edificio, por lo que un edificio puede tener uno o más planos, así mismo el mismo plano puede servir para construir otros edificios del mismo tipo, y que puede ser entendido por un especialista o un usuario dependiendo del tipo de plano; todas estas mismas características suelen tener los modelos en un lenguaje de modelamiento unificado.

Casos de Uso (Use Case)

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

Los casos de usos describen el sistema desde el punto de vista del usuario o cliente, es un diagrama entendible para los usuarios ya que se especifica las interacciones entre actores y las operaciones realizadas en el sistema, además de delimitar el sistema y la interacción que este tiene con su entorno.

Un diagrama de casos de uso consta de los siguientes elementos:

- ✓ Actor.
- ✓ Casos de Uso.
- ✓ Relaciones de Uso, Herencia y Comunicación.

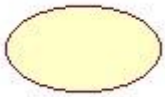
Elementos

Actor:

Una definición previa, es que un Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

Como ejemplo a la definición anterior, tenemos el caso de un sistema de ventas en que el rol de Vendedor con respecto al sistema puede ser realizado por un Vendedor o bien por el Jefe de Local.

Caso de Uso:



Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

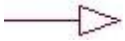
Relaciones:

a. **Asociación** 

   Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

b. **Dependencia o Instanciación** 

   Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

c. **Generalización** 

   Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de Uso (<<uses>>) o de Herencia (<<extends>>).

Este tipo de relación está orientado exclusivamente para casos de uso (y no para actores).

extends: Se recomienda utilizar cuando un caso de uso es similar a otro (características).

uses: Se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica.

De lo anterior cabe mencionar que tiene el mismo paradigma en diseño y modelamiento de clases, en donde está la duda clásica de usar o heredar.

Requerimientos

Es el proceso de indagar, por lo general en circunstancias difíciles, lo que se debe construir, y deben cumplir con la característica especial de ser útil para lo que el usuario necesite, con el tiempo un software va perdiendo eficacia y necesita de mantenimientos debido a que aparecen más requerimientos del sistema.

Requerimiento funcional

Son las capacidades que debe tener el sistema para cumplir con la función que necesita el usuario, estos requerimientos generalmente aparecen al inicio del proceso del diseño del sistema.

Requerimiento no funcional

Estos requerimientos aparecen conforme se va construyendo el sistema y se necesite de unas capacidades adicionales a las previstas para cumplir con el objetivo final del sistema.