***TEMA 4 PARTE 1***

**LA TRANSICIÓN DEL ANÁLISIS AL DISEÑO**

La transición del análisis al diseño es un activador que pretende modelar una manera de entender el sistema que se quiere desarrollar, esto quiere decir que es un Dominio del Problema. Ejemplo: [Aplicación]

Si con la actividad se pretende modelar una solución posible para un problema en un Dominio de la Solución. Ejemplo. [Aplicación] particular => [Aplicación]

**¿QUÉ ES DISEÑO DE SOFTWARE?**

El proceso de convertir la especificación de un sistema en un modelo computacional listo para ser codificado.

El diseño es la producción de una estructura software que realiza la especificación y después la implementación traduce esa estructura en un programa ejecutable.

En resumen el diseño es una forma de resolver los problemas mediante modelos, es decir, representan una solución completa para una necesidad software propuesta (problema).

**EL ANÁLISIS Y EL DISEÑO REPRESENTAN REALIDADES DIFERENTES**

La transición de un problema a su solución no es sencilla, ni inmediata, ni automática.

A pesar de que se usan de manera similar, estos dos modelos son cosas distintas. El análisis es una abstracción del problema en el mundo real, es decir en el que se construirá el sistema propuesto. El diseño es una abstracción interna de ese sistema propuesto, la solución planteada en el mundo real.

Tienen un propósito distinto el problema estudiar los requisitos sin tomar decisiones de implantación y la solución establecer la forma de construir el sistema antes de construirlo.

**¿QUÉ ES UN MODELO DE DISEÑO?**

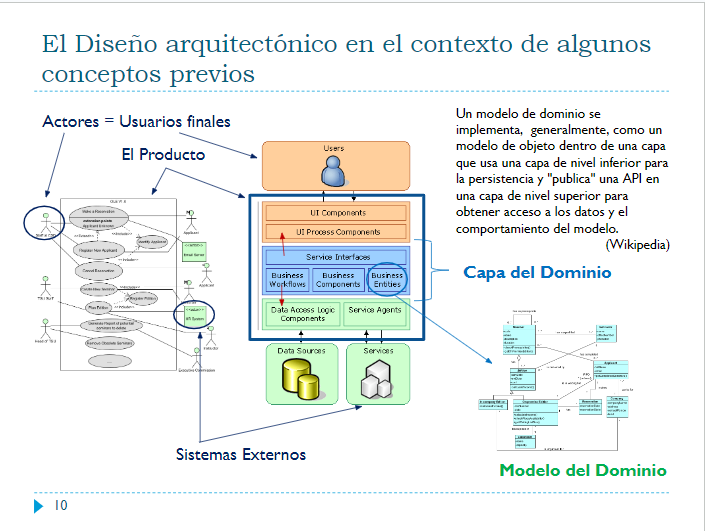
Es una definición de una solución software que satisface los requisitos.

Las soluciones tecnológicas son para implantar los requisitos del sistema, se necesita una construcción de un modelo del sistema antes de construirlo funcionalmente. Esto incluye aspectos de implementación; patrones de diseños, librerías de clases… También introduce nuevos artefactos ya que un objeto de análisis se puede implantar por un grupo de objetos de diseño. Tiene en cuenta la plataforma de implantación, así como consideraciones de eficiencia, rendimiento, optimización de recursos…

La dificultad es lo que determina dónde termina el análisis y dónde empieza el diseño

**¿CÓMO REALIZAR EL DISEÑO ARQUITECTÓNICO?**

Se refiere a la descomposición de un subsistema software en más subsistemas que interactúan entre sí. Es una descomposición en partes.



**DIAGRAMA DE PAQUETES UML - NOTACIÓN**

Se pueden modelar por medio de diagramas de paquetes UML para que sea una arquitectura lógica.

Los paquetes ofrecen un mecanismo para la organización de los modelos agrupando elementos de modelado. Cada paquete corresponde a un subconjunto del modelo y contiene clases, objetos, relaciones, componentes y diagramas asociados.

Un paquete puede contener otro paquete pero cada elemento solo está definido en un paquete.

Los paquetes son construcciones UML que le permiten organizar los elementos del modelo en grupos, haciendo que sus diagramas UML sean más simples y fáciles de entender. Los paquetes se representan como carpetas de archivos y se pueden usar en cualquiera de los diagramas UML, aunque son más comunes en diagramas de casos de uso y diagramas de clases porque estos modelos tienden a crecer.

Una clase de un paquete puede aparecer en otro paquete por la importación a través de una relación de dependencia entre paquetes

**¿CÓMO REALIZAR DISEÑO DE ALTO NIVEL CON OBJETOS?**

Con una estrategía de transformación, cambia el modelo por un modelo de diseño de clases de alto nivel.

El modelo de dominio inspira la capa de dominio, entonces se incluyen las clases de arquitectura, se consideran los requisitos que afectan al diseño lo que significa añadir objetos de diseño detallado.

**PROBLEMAS DE DISEÑO ORIENTADO A OBJETOS**

Encontrar los objetos apropiados, muchos de los objetos de un diseño cine del modelo de análisis. Pero los diseños orientados a objetos, a menudo terminan con clases que no tienen equivalentes en el mundo real.   
El cambio es la modelización rigurosa de la realidad conduce a un sistema que refleja las realidades de hoy, pero no necesariamente las de mañana. Las abstracciones que surgen durante el diseño son la clave para hacer un diseño flexible.

Otros problemas son determinar la granularidad de los objetos, el tamaño. Especificar la implantación de objetos y diseñar para el cambio.

***TEMA 4 PARTE 2***

**EL PROPÓSITO**

Para transformar un modelo de dominio en un Capa de Dominio se necesita añadir clases de diseño con sus atributos y operaciones utilizando una notación UML avanzada.

**ESTEREOTIPOS**

Los estereotipos añaden términos específicos a la notación general.

Existen dos categorías: los estereotipos estándar y de usuario. Ejemplos de estos son: CU: include y extend, tipos de clases, información extra de dependencias

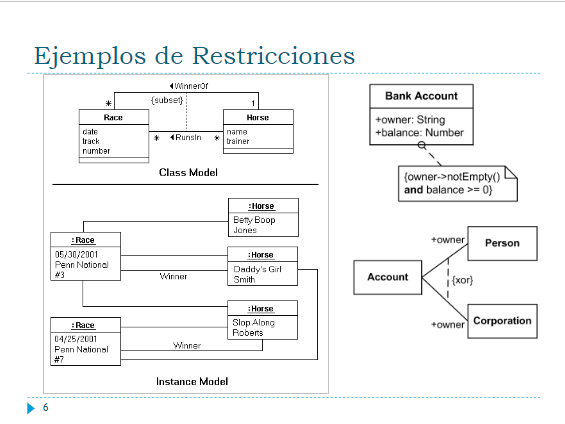
La información específica se puede añadir a los modelos UML a través de estereotipos y limitaciones. Un estereotipo es una palabra entre dos paréntesis angulares o cuñas. Significa que hay un uso especial o la intención del uso, de un elemento UML dado, estos elementos pueden ser cualquier cosa: una clase, método, asociación…

UML define un conjunto de estereotipos estándar, pero los usuarios también pueden definir sus propios estereotipos.

DAO: Data Access Object.

**RESTRICCIONES**

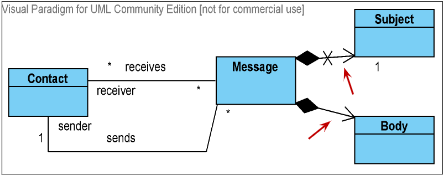
Generalmente se usan para especificar Reglas de negocio que se codifican en los métodos.

La regla de negocio es la referencia para el método dispacth(). La restricción dice que para un cliente con crédito insuficiente para que su orden sea enviada, tiene que estar pre-pagada. Las restricciones van más allá que las notaciones UML y las llamadas para un lenguaje en sí mismo: la OCL (Object Constraint Language) 

**NAVEGABILIDAD DE LAS ASOCIACIONES**

Si no tiene símbolos adicionales es bidireccional. A la hora de interpretar la navegabilidad de asociación el objeto en un extremo tiene referencia al objeto del otro extremo.

En las asociaciones unidireccionales los objetos en el final de la asociación no tienen referencia a los objetos en el otro extremo.



La punta de flecha muestra la navegación.

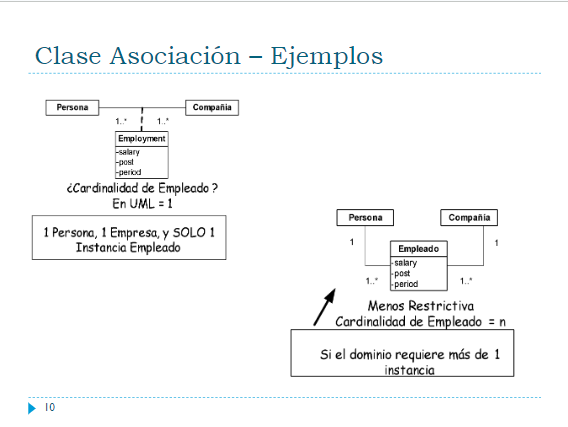
El marcador X significa NO-navegación, sin marcador significa sin especificar, de forma predeterminada no se permite la navegación.

**ASOCIACIONES DERIVADAS Y ROLES**

Redundancia para mejorar la comprensión o acceso que pueden obtenerse de más de una manera

Los roles de asociación son una función de la clase en un extremo de la asociación que juega un papel con respecto a la clase en el otro extremo.

**CLASE ASOCIACIÓN O CLASE ASOCIATIVA**

Modelando elementos con propiedades de ambas, una asociación y una clase. Contiene atributos que no pertenecen a ninguna de las dos clases relacionadas. Solo puede haber una instancia de una clase asociativa para par de instancias de las clases asociadas, en el caso de que se necesiten más instancias se debe interponer una nueva clase. 

Ln l

**ASOCIACIONES CUALIFICADAS**

Un calificador declara una partición del conjunto de instancias asociadas con respecto a una instancia al final cualificada. Una instancia de calificación incluye un valor para cada atributo calificador.

La asociación de objetos se ve determinada por la cardinalidad declarada:

* 0 ... 1, El valor de cualificación es único con respecto al objeto calificado, y designa un objeto asociado.
* 0 … \*, El conjunto de instancias asociadas se divide en subgrupos, cada uno seleccionado por una instancia de cualificación dada.

Multiplicidad:

* 1 o 0 … 1, La cualificación tiene consecuencias semánticas y de implementación.
* 0 … \*, No tiene consecuencias semánticas reales, pero sugiere una implementación que facilita el acceso a los grupos de instancias asociadas unidas por un valor de cualificación dado.

La cardinalidad de calificador se da asumiendo que el valor de la cualificación se suministra. La cardinalidad previa sin el calificador se supone que es 0 .. \*, normalmente es así. En una situación que prima la cardinalidad 1, un modelo mejor sería sin un calificador.

Una cardinalidad cualificada cuya cota es menos que cero indica que un valor clasificador dado puede estar ausente, mientras que límite inferior 1 indica que cualquier valor posible de calificador debe estar presente. Este último sólo es razonable para los cualificados con un número finito de valores (como valores enumerados o rangos de número entero) que representan tablas llenas indexadas por algún rango finito de valores.

**INTERFACES**

Representan una declaración de un conjunto coherente de funciones públicas y obligaciones, como un contrato.

También es una especificación la cual otros implementan y ofrecen servicios y otros consumen los servicios prestados.

**¿INTERFACES O CLASES ABSTRACTAS?**

La similitudes entres ellas es que ninguno de los dos tiene instancias y ambos permiten aplazar la implementación.

Las diferencias entre ellas es que las clases abstractas tienen tributos y pueden tener atributos y métodos, sin embargo las interfaces no pueden.

**HERENCIA VS COMPOSICIÓN Y DELEGACIÓN**

Herencia significa implementación. Delegación es una táctica donde una clase confía la responsabilidad en otra clase, una reutilización sin violar las restricciones de la herencia.

Si la herencia es múltiple es mejor no usarla y usar delegación.

**ALCANCE**

El alcance de instancia, una variable de instancia es un atributo cuyo calor se almacena en cada instancia de la clase. Una operación de instancia debe llamarse a través de una instancia.

El alcance de clase (Estático), una variable de clase es un atributo cuyo valor se almacena una sola vez y se comparte por todas las instancias. Una operación de class puede llamarse a través de la clase.

**RELACIONES**

Dependencia: Cuando los objetos de una clase trabajan esporádicamente con objetos de otra clase

Asociación: Cuando los objetos de una clase trabajan con objetos de otra clase por un tiempo prolongado

Agregación: Cuando una clase posee, pero comparte una referencia con objetos de otra clase

Composición/Delegación: Cuando una clase contiene objetos de otra clase

Herencia: Cuando una clase es un tipo de otra clase

—------------------------------------------------------------------------------------------------------------------------

Polimorfismo: varios objetos de diferentes clases, pero con una base común, se pueden usar de manera indistinta, sin tener que saber de qué clase exacta son para poder hacerlo