

Análisis y diseño de aplicaciones I



UT3 Diseño y UML

1

Agenda



- Diagrama de casos de uso
- Diagrama de clases
 - Generalización
 - Asociaciones
 - Clase de asociación
 - Agregación
 - Composición
 - Dependencia
 - Atributos
 - Multiplicidad
 - Notas y comentarios
 - Enumeradores
 - Propiedades derivadas
 - Restricciones

2

Caso de uso (recap)



- Los casos de uso son una **técnica** para capturar **los requisitos funcionales** de un sistema.
- Describen las **interacciones** típicas entre los **usuarios de un sistema** y el propio sistema, proporcionando una **narrativa** de cómo se utiliza un sistema.
- En lugar de describir los casos de uso directamente, es más fácil comenzar describiendo escenarios. Un escenario es una secuencia de pasos que describe una interacción entre un usuario y un sistema.
- Los actores son los usuarios del sistema que desempeñan un papel específico en un caso de uso. Un caso de **uso es un conjunto de escenarios unidos por un objetivo común del usuario**. Aunque los casos de uso son una parte importante del UML, la definición de casos de uso en el UML es bastante escasa.

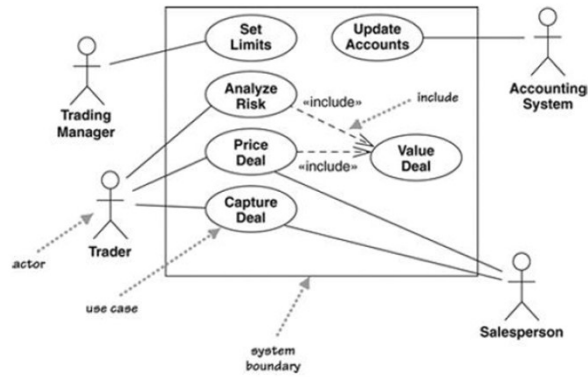
3

Diagrama de Caso de uso



- La mejor manera de pensar en un diagrama de casos de uso es que es una **tabla de contenidos gráfica para el conjunto de casos de uso**.
- El diagrama de casos de uso muestra los actores, los **casos de uso** y las **relaciones entre ellos**, como qué actores realizan qué casos de uso y qué casos de uso incluyen a otros. El UML incluye otras relaciones entre los casos de uso más allá de los simples "includes", como "extend", pero se recomienda ignorarlos para evitar confusiones.
- En cambio, se debe concentrar en la **descripción textual** de un caso de uso, donde se encuentra el verdadero valor de la técnica.

4



UML Distilled - Martin Fowler

5

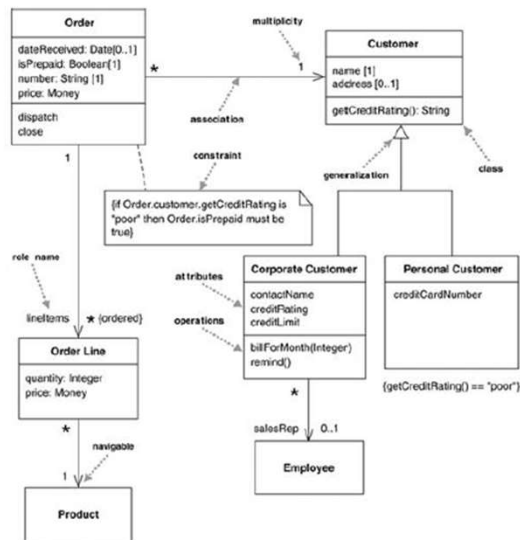
Tarea de Aplicación 4

Diagrama CU

6

3

Diagrama de clases



7

Diagrama de clases



- Describe los **tipos de objetos** en el sistema y los diversos tipos de **relaciones estáticas** que existen entre ellos.
- También muestran las **propiedades** y **operaciones** de una clase y las **restricciones** que se aplican a la forma en que los objetos están conectados.
- El UML utiliza el término "feature" como un término general que abarca las propiedades (**Estado**) y operaciones (**Comportamiento**) de una clase.

8

Generalización



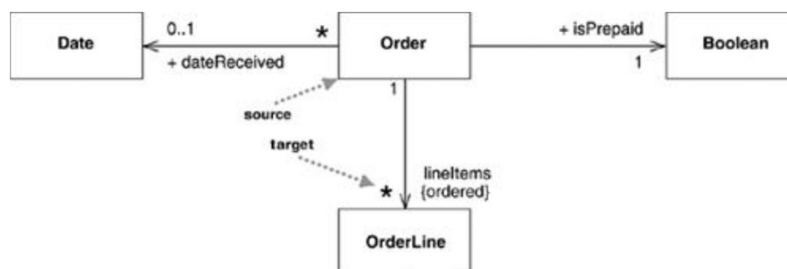
- Las similitudes pueden colocarse en una clase general (el supertipo), una relación de tipo generalización a sus subtipos.
- Aunque la herencia es un mecanismo poderoso, también trae mucho peso que no siempre se necesita...

9

Asociaciones



- Este tipo de relación es el más común y se utiliza para representar dependencia semántica. Se representa con una simple línea continua que une las clases que están incluidas en la asociación.

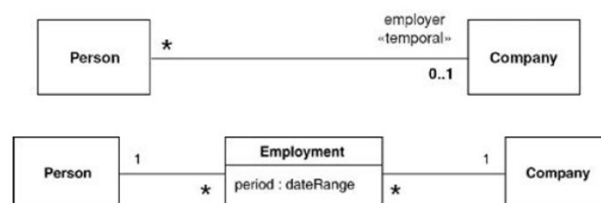


10

Clase de asociación



- Durante el proceso de diseño, puede surgir **comportamiento** u otros atributos que no son tienen un responsable claro en la asociación.
- En esos casos, surge la necesidad de tener una nueva clase con la información de la asociación.



11

Agregación



- Agregación se representa con rombo blanco. Refiere semánticamente a “**es parte de**”.
- La duda es; cuando es una agregación y cuando es una asociación. Martin nos responde:
 - As Jim Rumbaugh says, "Think of it as a modeling placebo" [Rumbaugh, UML Reference]

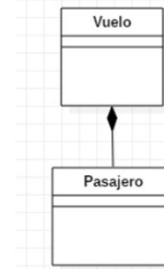


12

Composición



- Similar a la agregación, pero con una **relación jerárquica más fuerte** entre el **objeto** y las **partes** que lo componen.
- Los elementos que forman parte no tiene sentido de existencia si no es dentro del elemento que los compone.
- Tienen los mismos tiempos de vida, cuando el objeto padre muere todos los compuestos también.

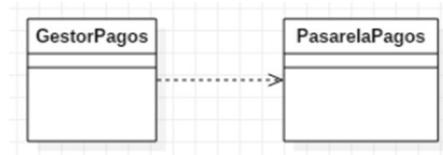


13

Dependencia



- Se utiliza para reflejar relaciones donde una clase requiere de otra para funcionar, pero esta no forma parte del Estado.



14

Atributos



- Visibilidad nombre: tipo multiplicidad = valor predeterminado {cadena de propiedades}
- La visibilidad indica si el atributo es público (+) o privado (-), o restringido (#)
- El nombre es el nombre del atributo.
- El tipo indica qué tipo de objeto puede almacenar el atributo,
- La multiplicidad indica cuántos objetos se pueden almacenar en el atributo.
- El valor predeterminado es el valor del atributo si no se especifica en la creación del objeto.
- Las propiedades permite indicar propiedades adicionales del atributo. como por ejemplo si es de solo lectura. La omisión de la cadena de propiedades indica que el atributo es modificable.

15

Multiplicidad



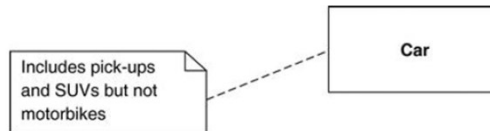
- La multiplicidad de una propiedad indica cuántos objetos pueden ocupar esa propiedad.
- Las multiplicidades más comunes son:
 - 1 (una orden debe tener exactamente un cliente)
 - 0..1 (un cliente corporativo puede o no tener un solo representante de ventas)
 - * (un cliente no necesariamente realiza un pedido y no hay límite superior en la cantidad de pedidos que un cliente puede realizar, cero o más pedidos).
- En general, las multiplicidades se definen con un límite inferior y un límite superior.
- Los elementos en una multiplicidad forman un conjunto por defecto, pero se puede agregar {ordered} para indicar que el orden es importante y {nonunique} para permitir duplicados.

16

Notas y comentarios



- Las notas pueden estar solas o pueden estar vinculadas con una línea discontinua a los elementos a los que están comentando.
- Pueden aparecer en **cualquier tipo de diagrama**.
- La línea discontinua puede ser incómoda ya que no se puede posicionar exactamente dónde termina esta línea. Por lo tanto, una convención común es poner un círculo abierto muy pequeño al final de la línea.
- A veces es útil tener un comentario en línea sobre un elemento del diagrama. Puedes hacer esto agregando un guión antes del texto: --.

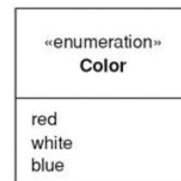


17

Enumeradores



- Set fijo de valores que no tienen comportamiento.
- En la práctica... Prefiero dejarlos “dinámicos...”

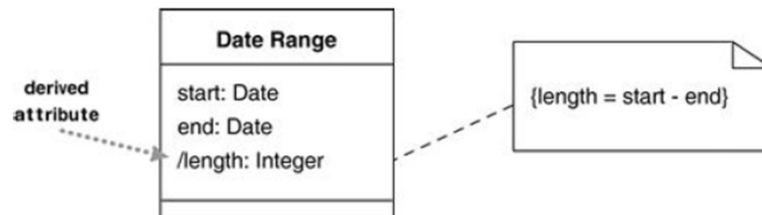


18

Propiedades derivadas



- Se calculan en base a otros atributos

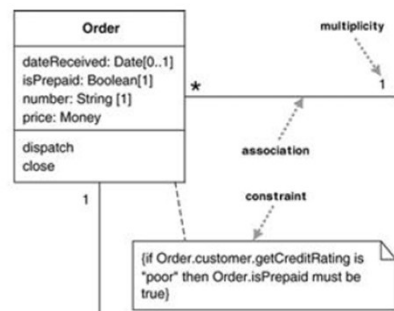


19

Restricciones



- Se permiten expresar restricciones, utilizando los corchetes {}.



20

Tarea de Aplicación 5

Clasificación inicial de sistemas.



21

Bibliografía



- Ingeniería del Software 7ma. Ed. - Ian Sommerville.pdf
- UML Distilled - Martin Fowler

22