UCAB UNIVERSIDAD CATÓLICA ANDRÉS BELLO

DIAGRAMAS DE CLASES

Tabla de contenido

Objetivo 3
Introducción 4
Diagrama de clases de análisis 5
Diagrama de clases de diseño 9
Cierre 13
Referencias 14

Objetivo

Identificar el modelo basado en clases para especificar el dominio del problema y el diseño del *software*.

Introducción

Con la llegada del paradigma de orientación a objetos cambió el concepto de análisis dentro del proceso de desarrollo de *software*. Ahora "análisis" no es averiguar lo que desea el cliente/usuario; esto es "gestión de los requerimientos". Ahora "análisis" es identificar los "objetos", para luego llevarlos a clases y así poder construir nuestro sistema. Existen dos tipos de objetos/clases: los del mundo real y los que finalmente propondremos en nuestro diseño.

"Análisis" es identificar estos objetos del mundo real. "Diseño" es decidir cuáles de ellos serán los que transformaremos en las clases de nuestro sistema.

Los objetos del mundo real se conocen como los objetos del dominio del problema, o simplemente del dominio. Entonces, cuando hacemos análisis del dominio estamos analizando las capacidades comunes dentro de un dominio que puedan ser representadas en términos de objetos/clases.

Para su representación y mejor entendimiento, elaboraremos un diagrama de clases, llamado diagrama de clases de análisis.

Diagrama de clases de análisis

Partiendo de la identificación progresiva de los requerimientos, vamos identificando:

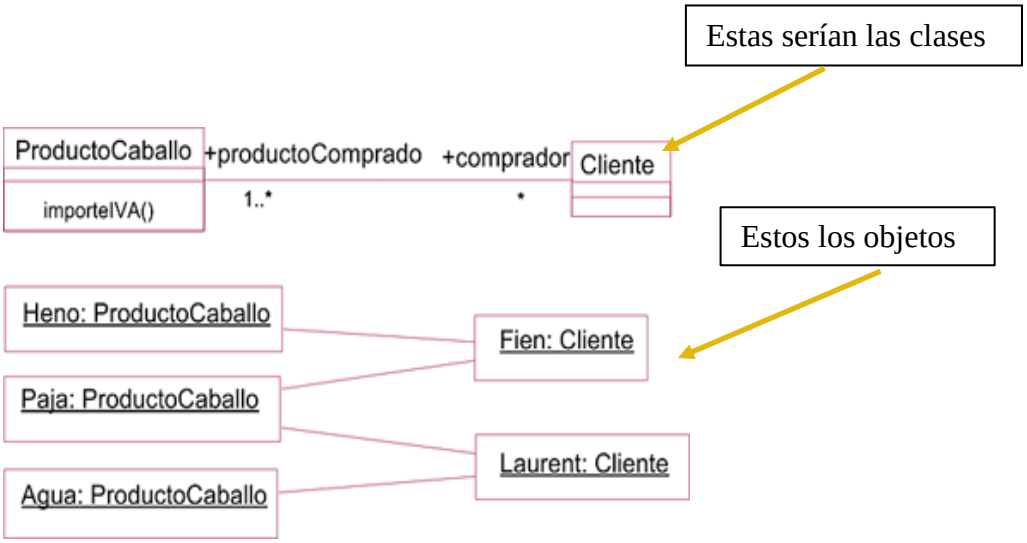
- Clases
- Servicios que se prestan entre ellas
- Asociaciones entre ellas.

El Diagrama de clases se elabora también en ambos enfoques: clásico y ágil. Es oportuno recordar algunas definiciones:

Objeto:

- Es una entidad con límites bien definidos que encapsula estado y comportamiento.
- Es una instancia de una clase.
- Es un paquete cohesivo de datos y funciones.
- Las funciones también son llamadas operaciones.
- Los objetos son únicos.
- Un objeto es una clase instanciada.

Así se representa un objeto en UML:

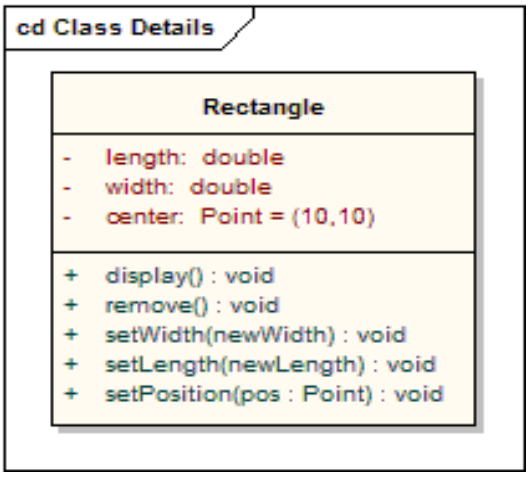


Fuente: ejemplo de Diagrama de objetos. Extraída de Debrauwer y Van der Heyde (2011)

Clase:

- Es el descriptor de un conjunto de objetos que comparten los mismos atributos, operaciones y relaciones.
- La visibilidad se aplica tanto a atributos como a operaciones:
    - Pública + (visible por todos los objetos)
    - Privada - (visible solo por el propio objeto)

Así se representa una clase en un UML:



Fuente: diagrama de clases UML 2. Extraída de Sparx Systems (s.f.)

Con las clases vamos a construir entonces dos diagramas:

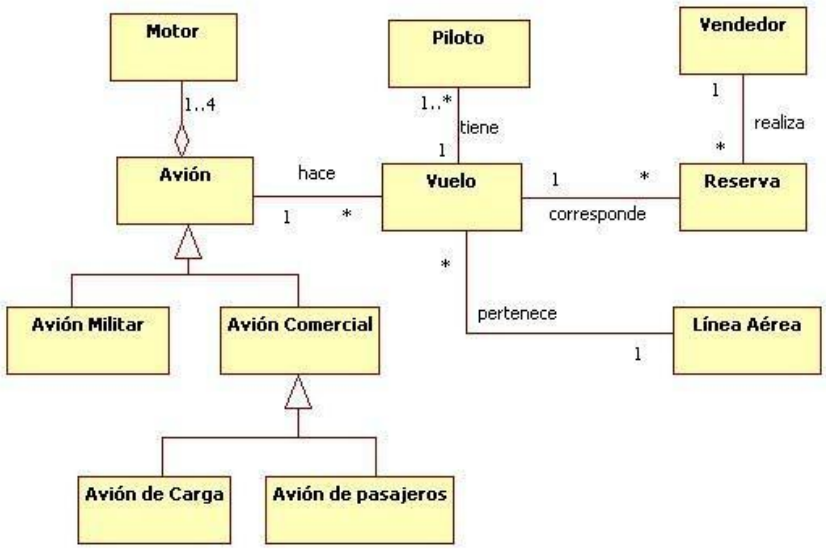
- Diagrama de clases de análisis: tiene las clases del dominio. También llamado modelo de dominio.
- Diagrama de clases de diseño: tiene las clases de la aplicación. También llamado modelo de diseño.

- **Modelo de análisis o dominio:** este modelo se usa para describir los conceptos del dominio del problema. Las clases pueden representar y estructurar objetos tales como información, productos, documentos y organizaciones; estos existen en el mundo real.

Los nombres de las clases y las asociaciones usadas en el modelo son conceptos definidos; esto quiere decir que todos los miembros del equipo entienden de qué se está hablando.

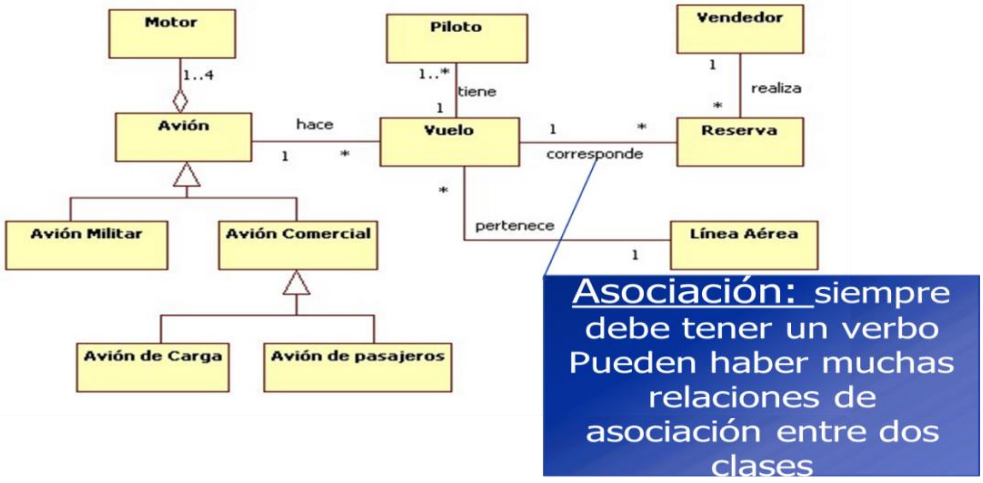
Este modelo de análisis no es un diagrama final que describe todos los posibles conceptos y sus relaciones. Es un primer intento por definir los conceptos claves que describen el negocio/problema, para lograr un entendimiento entre todos los miembros del equipo de desarrollo.

El nombre de la clase se escribe en singular. Algunos autores también lo llaman modelo conceptual. Veamos un ejemplo usando la notación UML:

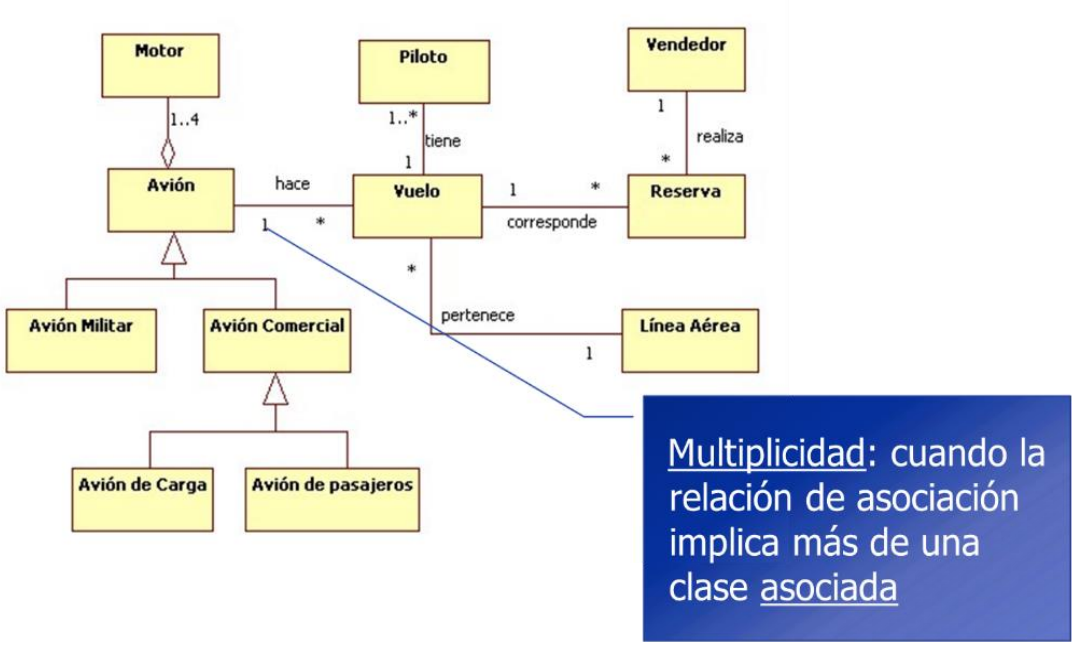


Fuente: elaboración propia

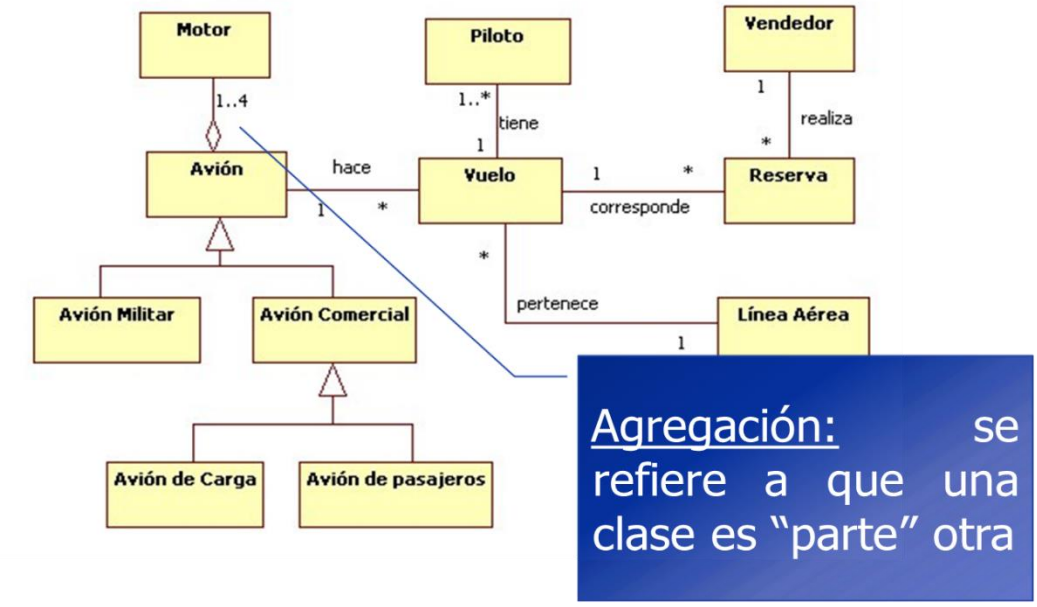
- No se identifican atributos ni operaciones.
- En él podemos proponer las siguientes relaciones:



Fuente: elaboración propia

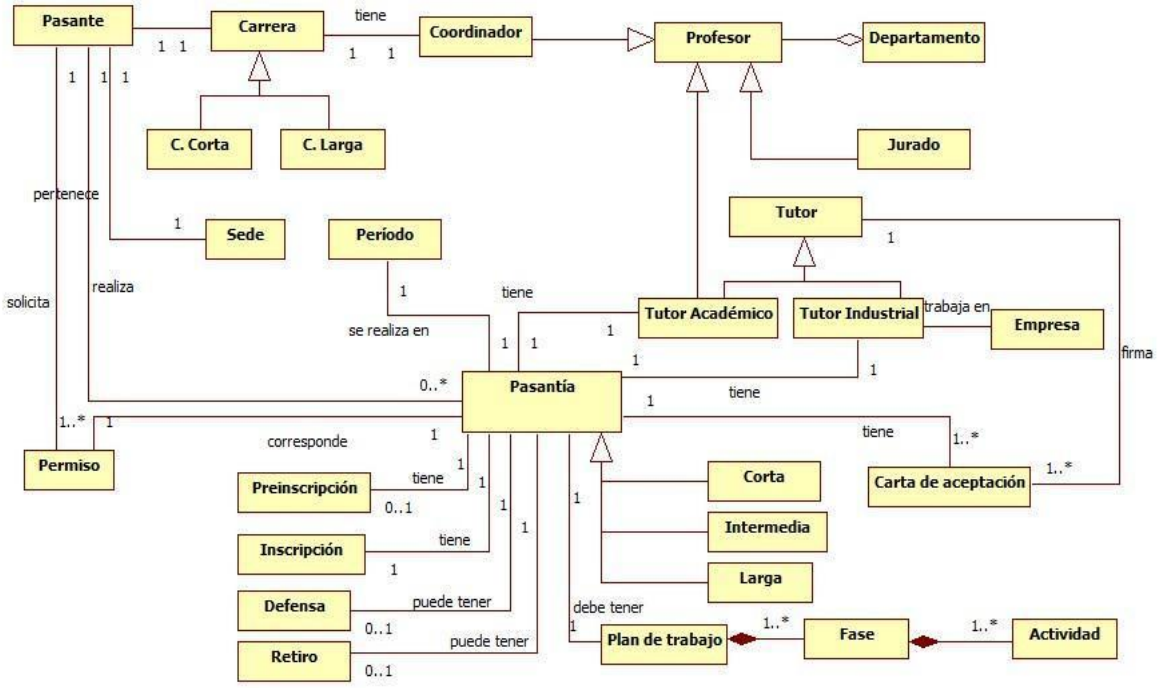


Fuente: elaboración propia



Fuente: elaboración propia

Aquí tenemos otro ejemplo:



Fuente: elaboración propia

No debes repetir los nombres; estos son únicos.

- Este modelo se va construyendo durante la actividad de análisis y debe ser registrado.
- Se utilizan herramientas de *software* para elaborar este diagrama.
- Pero, además, debemos organizarlos en un artefacto: el Diagrama de clases de diseño (DAS).

Diagrama de clases de diseño

En el paradigma de orientación a objetos, el diseño implica identificar las clases que implementarán los requerimientos de la aplicación. Esto se logra haciendo evolucionar el modelo de dominio (diagrama de clases de análisis) a un diagrama de clases de diseño.

Para el enfoque ágil este diagrama se hace uno por *Sprint.* Mientras que para el clásico es uno solo por aplicación.

Lo que se hace es refinar el diagrama de clases de análisis para obtener un diagrama de clases de diseño.

Una clase de diseño es aquella que está lo suficientemente detallada como para que se pueda generar el código fuente con ella. Sus especificaciones (detalles) deben ser tales que puedan ser implementadas en un lenguaje de programación particular. Se deben especificar los atributos y los métodos (operaciones) de cada clase.

En este proceso pueden aparecer nuevas clases; por lo tanto, el diagrama debe actualizarse continuamente.

Si el diagrama contiene muchas clases se puede empaquetar.

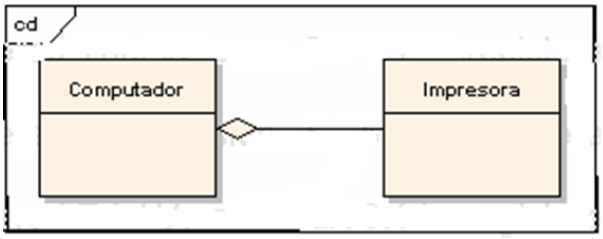
El diseño es una actividad altamente creativa: no hay diseño bueno ni diseño malo, solo hay un diseño que da "menos problemas". Esto se aclarará mejor más adelante. Por lo tanto, solo se puede contar con algunas sugerencias para la actividad de diseño.

Se considera un buen diseño si tiene:

- **Completitud**: es darles a los clientes lo que ellos están esperando; no falta ninguna funcionalidad de las solicitadas en el diseño
- **Suficiencia**: todas las clases se usan (no sobra ninguna)
- **Primitiva**: las operaciones de las clases deben ser simples
- **Alta cohesión**: las clases cohesivas son más fáciles de reutilizar
- **Bajo acoplamiento**: solo los enlaces necesarios entre ellas.

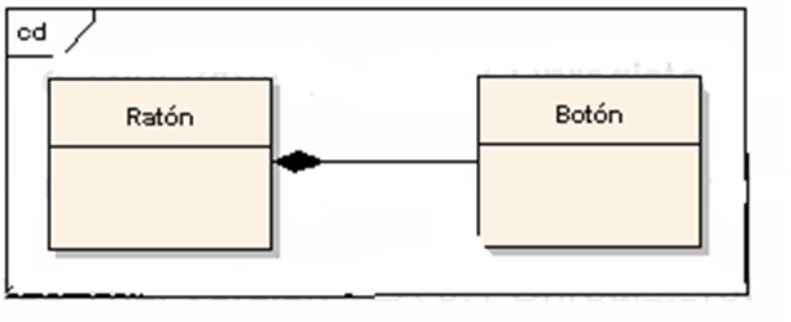
Cuando se diseña o elabora el diagrama de clases es conveniente refinar también las relaciones.

- UML define dos tipos de relaciones de asociación:
  - **Agregación:**



Fuente: elaboración propia

- **Composición:**

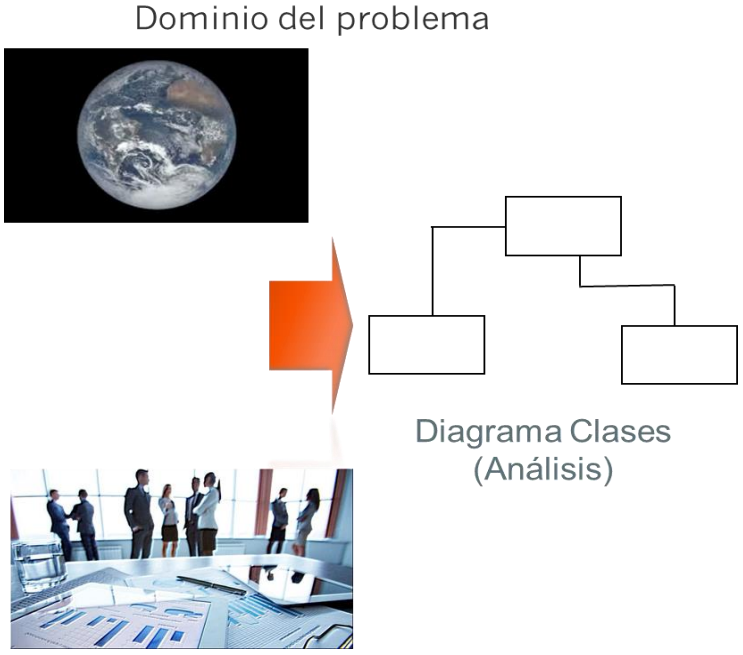


Fuente: elaboración propia

En paralelo, en este proceso creativo veremos que influye la herramienta que utilizaremos para implementar en código fuente este diseño. Básicamente se habla del ambiente de desarrollo (IDE por sus siglas en inglés: *Integrated Development Environment)*.

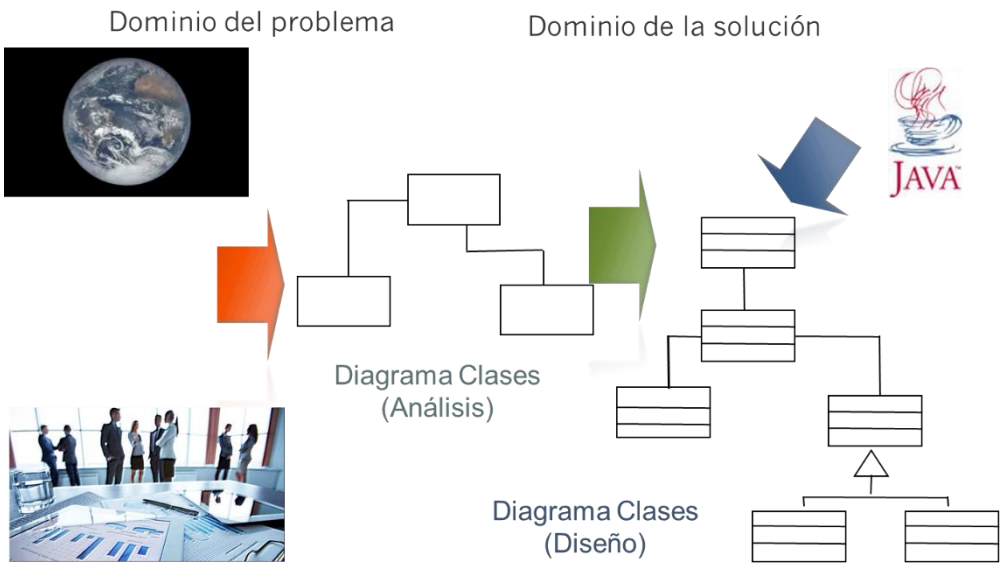
Por ello, es clave que el diseñador conozca muy bien su ambiente de desarrollo, porque puede proponer algunas clases que el IDE no le permita implementar, o viceversa. El IDE puede contar con clase, mecanismos etc. que le faciliten su diseño.

El proceso sería algo así:



Fuente: elaboración propia

A partir del dominio del problema construimos el Diagrama de clases de análisis.



Fuente: elaboración propia

Luego de refinar y considerar los aportes que puede hacer el IDE, obtenemos el modelo de clases de diseño.

Cierre

La llegada del paradigma de orientación a objetos hizo cambios muy fuertes en la forma de construir *software*. Una de ellas es el proceso de análisis. Ahora hay que encontrar los objetos: por eso se comienza por los reales para luego llevarlos a clases.

Esta es la razón por la cual el resultado del análisis lo vamos a representar en un modelo de análisis, utilizando un diagrama: el Diagrama de clases. Recuerda, un dibujo dice más que mil palabras. Pero este dibujo lo elaboramos siguiendo una sintaxis, la de UML, lo cual hace que su comprensión sea internacional.

A este diagrama también se le conoce como Modelo de dominio o conceptual. Se elabora tanto en el enfoque ágil como en el clásico, pues es necesario para que todo el equipo de desarrollo esté de acuerdo en los conceptos propios del dominio del problema.

Partiendo de este modelo de análisis, y de los beneficios que puede dar el IDE, se construye el modelo de diseño, el cual es consecuencia de un proceso de refinamiento del modelo de análisis y del aporte creativo del diseñador. Esta actividad también debería ser colaborativa.

Para el caso del enfoque clásico se obtiene al final un solo diagrama de clases. Para el enfoque ágil se obtiene un diagrama de clases para cada *Sprint*.

Referencias

Referencias de las imágenes

Debrauwer y Van der Heyde (2011). *Ejemplo de diagrama de objetos* [Imagen]. Recuperado de UML 2: Iniciación, ejemplos y ejercicios corregidos. Editions ENI. https://www.ediciones-eni.com/open/mediabook.aspx?idR=1bc6a15478c69638ef648dc1e7f0f362

Sparx Systems (s.f.). *Diagrama de clases UML 2* [Imagen]. Recuperado de *Sparx Systems*. Recuperado de http://www.sparxsystems.com.ar/resources/tutorial/uml2_classdiagram.php

Bibliografía sugerida

Eriksson, H., Penker, M., Lyons, B. y Fado, D. (2004). *UML2 Toolkit*. Wiley Publishing Inc. Recuperado de https://www.ecotec.edu.ec/documentacion/investigaciones/docentes_y_directivos/articulos/6008_TRECALDE_00278.pdf

Rumbaugh, J., Jacobson, I. y Booch, G. (2002). *El Lenguaje Unificado de Modelado: Manual de Referencia.* Addison Wesley.