IIC2113 - Diseño Detallado de Software

CLASE 2 - UML

Pontificia Universidad Católica de Chile
2020-2

Índice

01. Diagramas UML

Historia, qué son y los 14 diagramas

Historia

¿Cómo se definió UML?

1979

Strunk y White escriben el primer texto seminario llamado “The Elements of Style” para definir las guías de estilo para escribir correctamente en inglés.

OOP lidera la industria

1980 [I]

En la década de los 80 el paradigma orientado a objetos empezó a liderar la industria del software y se estableció como el paradigma dominante.

A principios de los 90 comenzaron a aparecer en el mercado distintos libros emblemáticos sobre el modelado gráfico para programación orientada a objetos (OOP).

1980 [F]

Todos tienen su propia notación

Primer esfuerzo de estandarización

1989

Se funda el "**Object Management Group (OMG)**" un consorcio sin fines de lucro que fomenta las buenas prácticas sobre OOP.

Iniciativas Privadas

1990 [I]

Grady Booch, Ivar Jacobson y Jim Rumbaugh contratados bajo la empresa Rational unieron esfuerzos para definir las propuestas de los estándares.

Rational define la primera versión. La OMG impulsa su estandarización. La OMG toma el rol de su mantención.

1997

UML 1.0

UML 2.0

2005

A lo largo de los años, entre UML 1.0 y 2.0 existieron varias versiones mejorando y aterrizando de mejor forma sus diagramas.

UML 2.5.1

2017

La última versión publicada a la fecha (Agosto 2020).
Se pueden encontrar la lista de cambios en la página de la OMG
https://www.omg.org/spec/UML

Historia UML

Eso hasta la fecha

¿Qué es UML?

- *"Unified Modeling Language (UML) es la familia de notaciones gráficas, respaldadas por un modelo, que ayudan a describir y diseñar sistemas de software, enfocado principalmente a sistemas orientados a objetos"* (Martin Fowler, 2003).
- El valor principal de UML es la comunicación y el entendimiento.
- Está pensado tanto para especialistas como para no-expertos.
- A pesar de que las reglas estándar (por la OMG) son rigurosas, los diagramas suelen ser modificados dependiendo de las necesidades y experiencia.
- UML es solo diagramación. No existe un *mapping* oficial directo con el código.

¿Cuándo usamos UML?

Pueden ser usados como ***forward engineering*** o como ***reverse engineering***.

- UML como un ***sketch***
  - Usado entre programadores para comunicar y expresar ideas.
- UML como un ***blueprint***
  - Es más formal. Pensado para definir los requerimientos o documentación formal del sistema. Generalmente por *expertos*.
- UML como ***lenguaje de programación***
  - Poco común en la práctica porque se requieren herramientas especializadas.
  - Tampoco existe un estándar para el paso de diagrama a código.

¿Cómo clasificamos los diagramas UML?

Según los estándares de UML:

- Diagramas de Comportamiento
  - Interacciones y comunicación.
- Diagramas de Estructura
  - Organización de la solución

El modelo 4+1 propone otra forma de clasificar los diagramas (lo veremos al final de esta clase).

Tipos oficiales de diagramas UML

Nombre	Propósito
1. De Actividad (Activity)	Comportamiento paralelo y de procedimientos.
2. De Clases (Class)	Clases, funcionalidades y relaciones.
3. De Comunicación (Communication)	Interacción entre los objetos. Énfasis en los *links*.
4. De Componentes (Component)	Estructura y conexión entre componentes.
5. Estructura Compuesta (Composite structure)	Descomposición de una clase en tiempo de ejecución.
6. De Despliegue (Deployment)	Despliegue de los *artefactos* hacia los *nodos*.
7. Global de Interacciones (Interaction overview)	Mezcla de secuencia y actividad.

Nombre	Propósito
8. De Objetos (Object)	Ejemplo de configuraciones e instancias.
9. De Paquetes (Package)	Estructura jerárquica en tiempo de compilación.
10. Secuencia (Sequence)	Interacción entre objetos. Énfasis en la secuencia.
11. Máquina de estados (State machine)	Como los eventos cambian un objeto a lo largo del ciclo.
12. De Tiempos (Timing)	Interacción entre objetos. Énfasis en el tiempo.
13. De Casos de Uso (Use case)	Cómo los usuarios interactúan con el sistema.
14. De Perfil (Profile)	Define estereotipos, clasificaciones y restricciones.

Diagrama de Clases sobre los diagramas UML

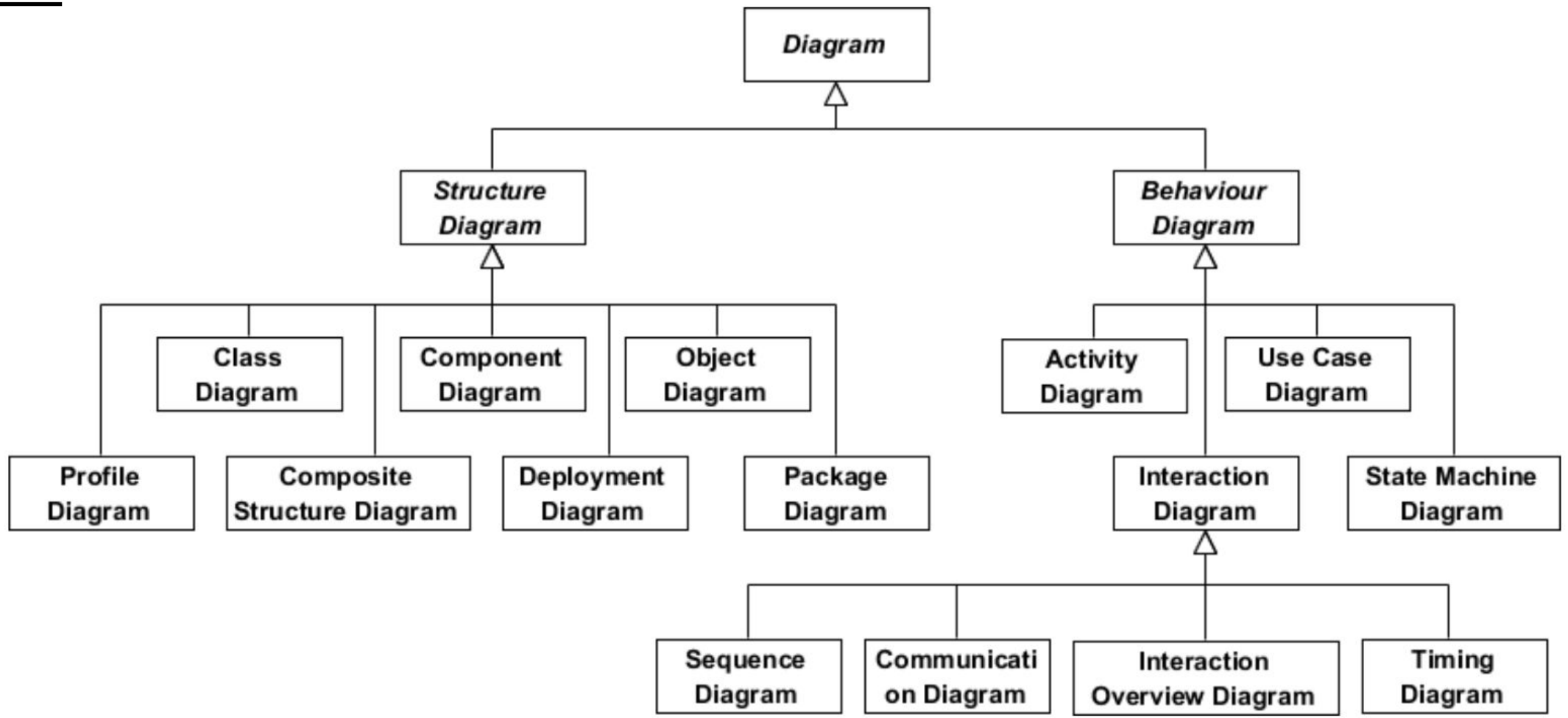
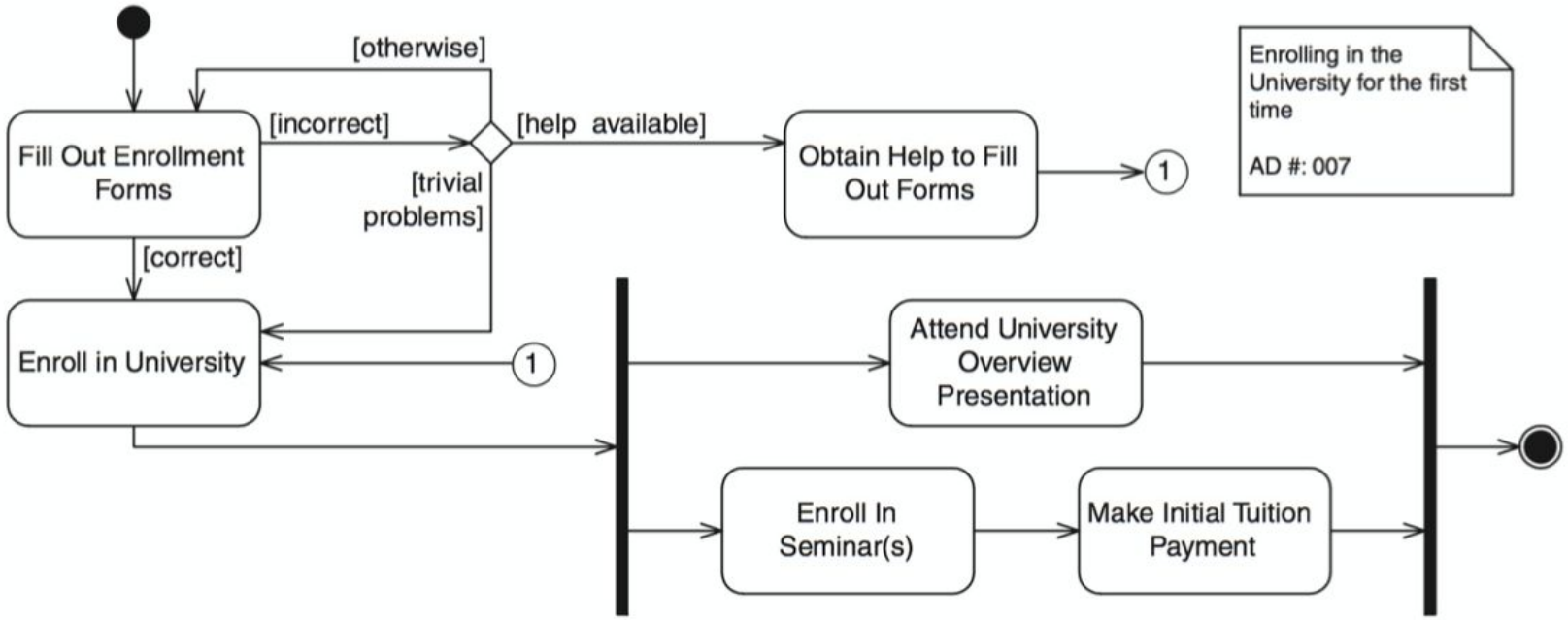


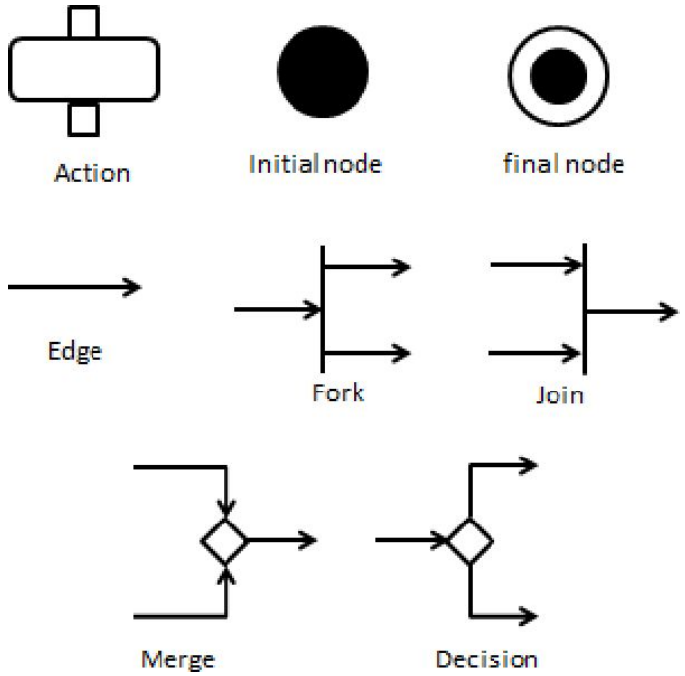
Diagrama por Visual Paradigm https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

1. Diagrama de Actividad

También conocido como diagrama de flujo (flowmap).

Modela procesos en base a reglas.

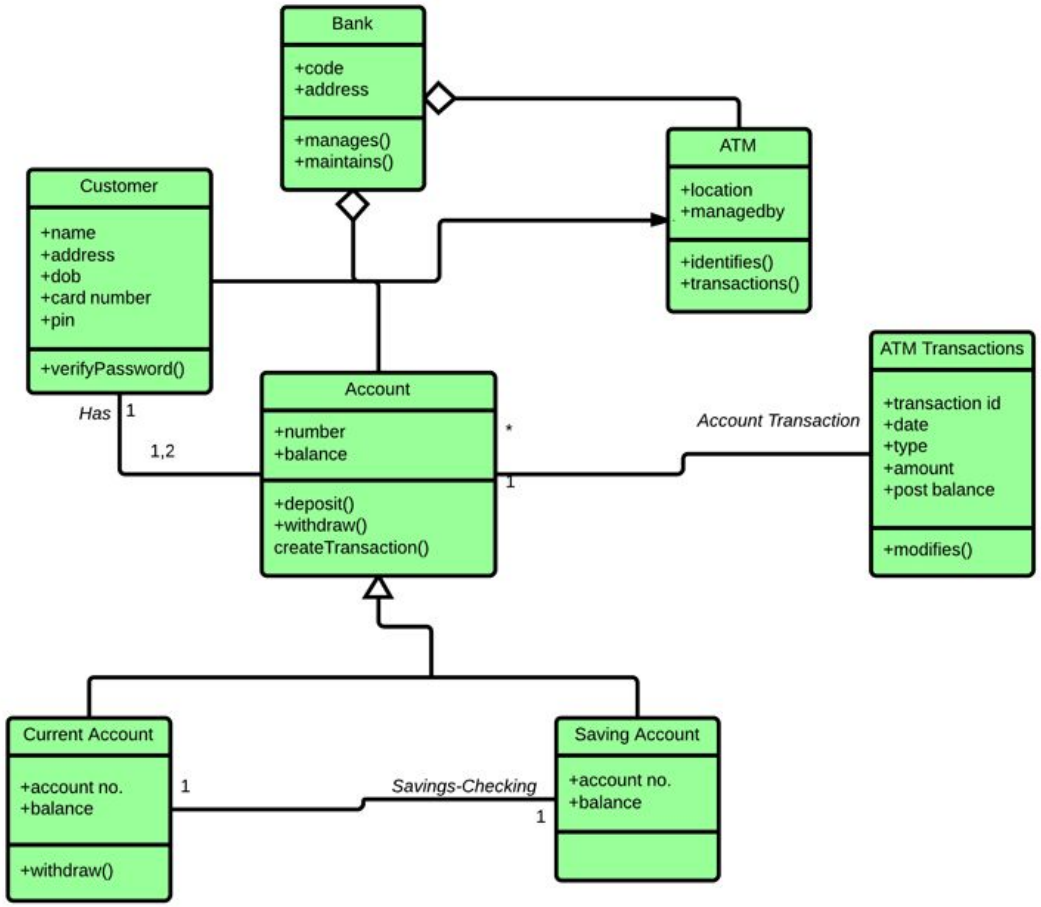




https://www.lucidchart.com/pages/uml-activity-diagram#section_3

2. Diagrama de Clases

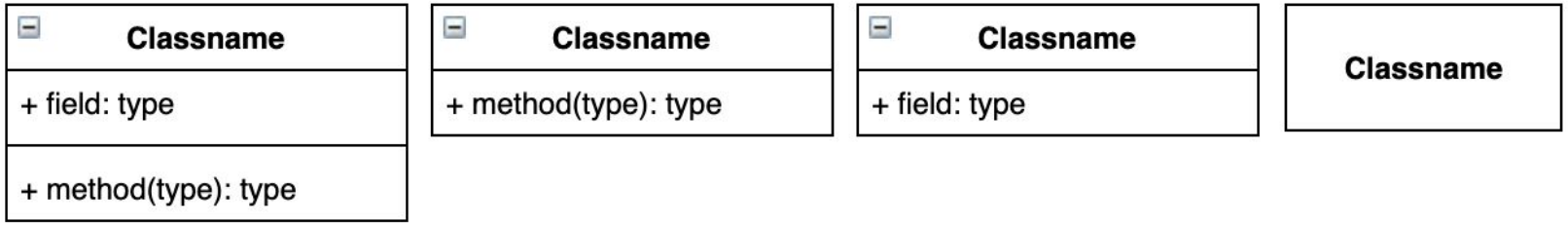
Los diagramas de clase buscan describir los distintos objetos presentes en un sistema, cómo interactúan y el conjunto de atributos y métodos de cada uno de ellos.



Los diagramas de clase buscan describir los distintos objetos presentes en un sistema, cómo interactúan y el conjunto de atributos y métodos de cada uno de ellos.

Diagrama de Clases - Notación

Rectángulos para representar las clases:



Líneas para representar las relaciones:

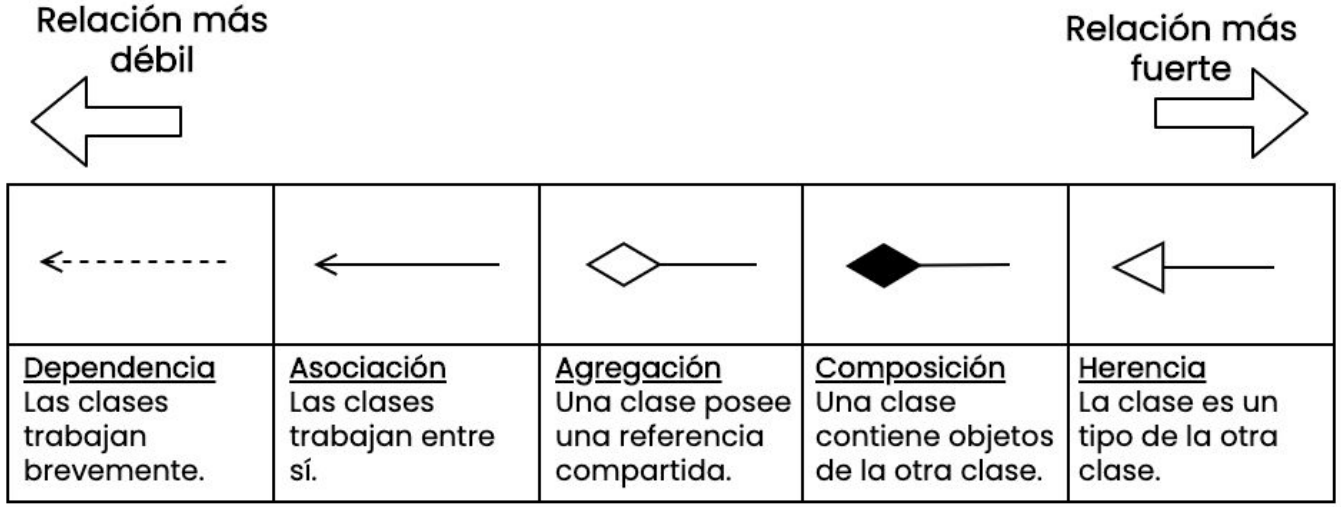
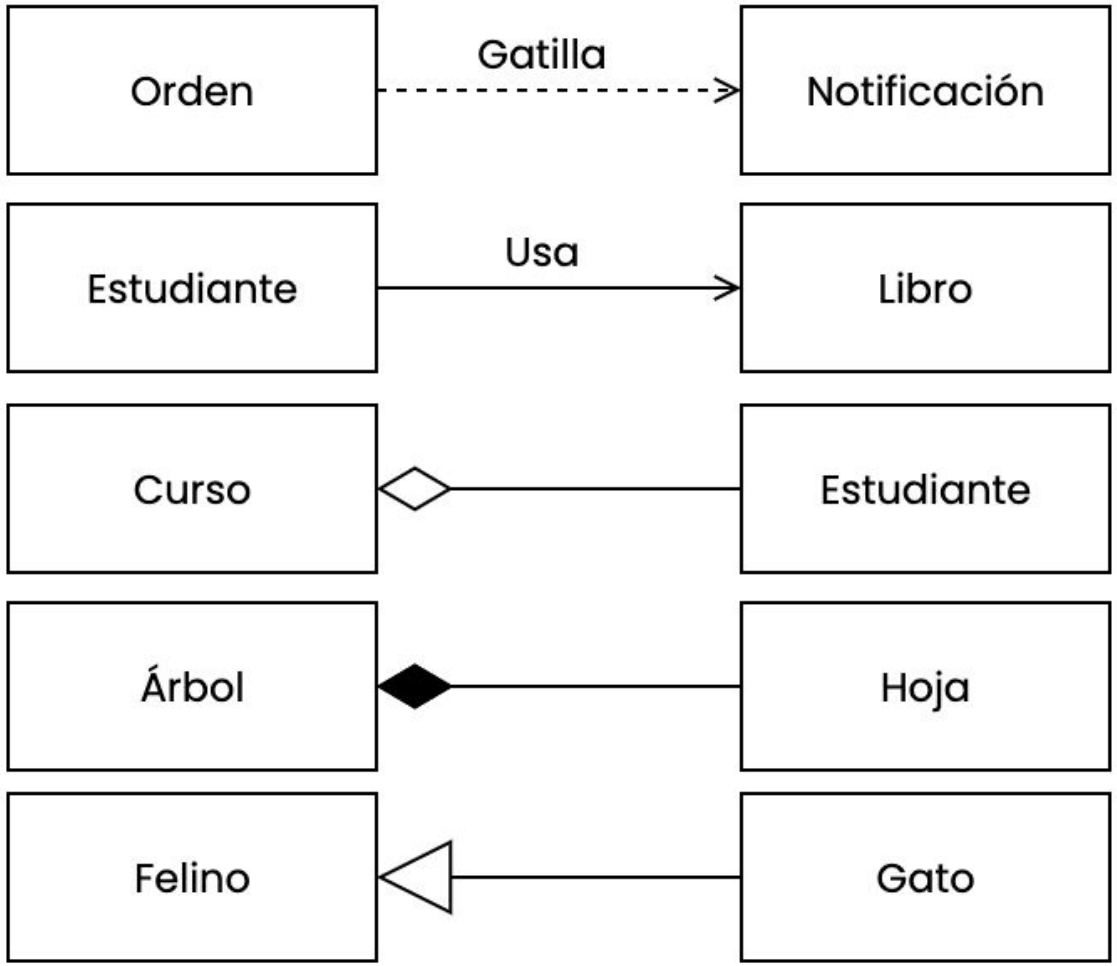


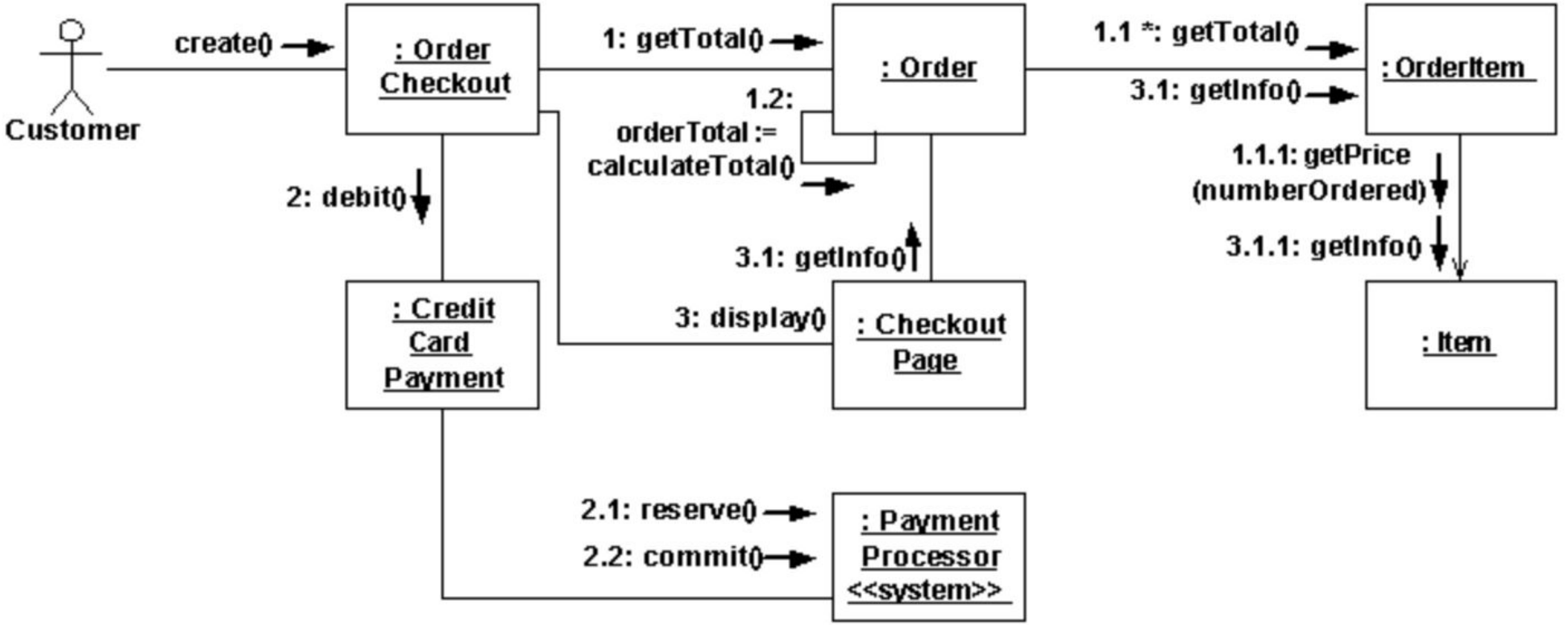
Diagrama de Clases - Ejemplos



Multiplicidad	Opción	Cardinalidad
0..0	0	La colección debe estar vacia
0..1		Una o cero instancias
1..1	1	Exactamente una instancia
0..*		Cero o más instancias
1..*		Al menos una instancia
5..5	5	Exactamente 5 instancias
m..n		Al menos m, pero no más de n

3. Diagrama de Comunicación

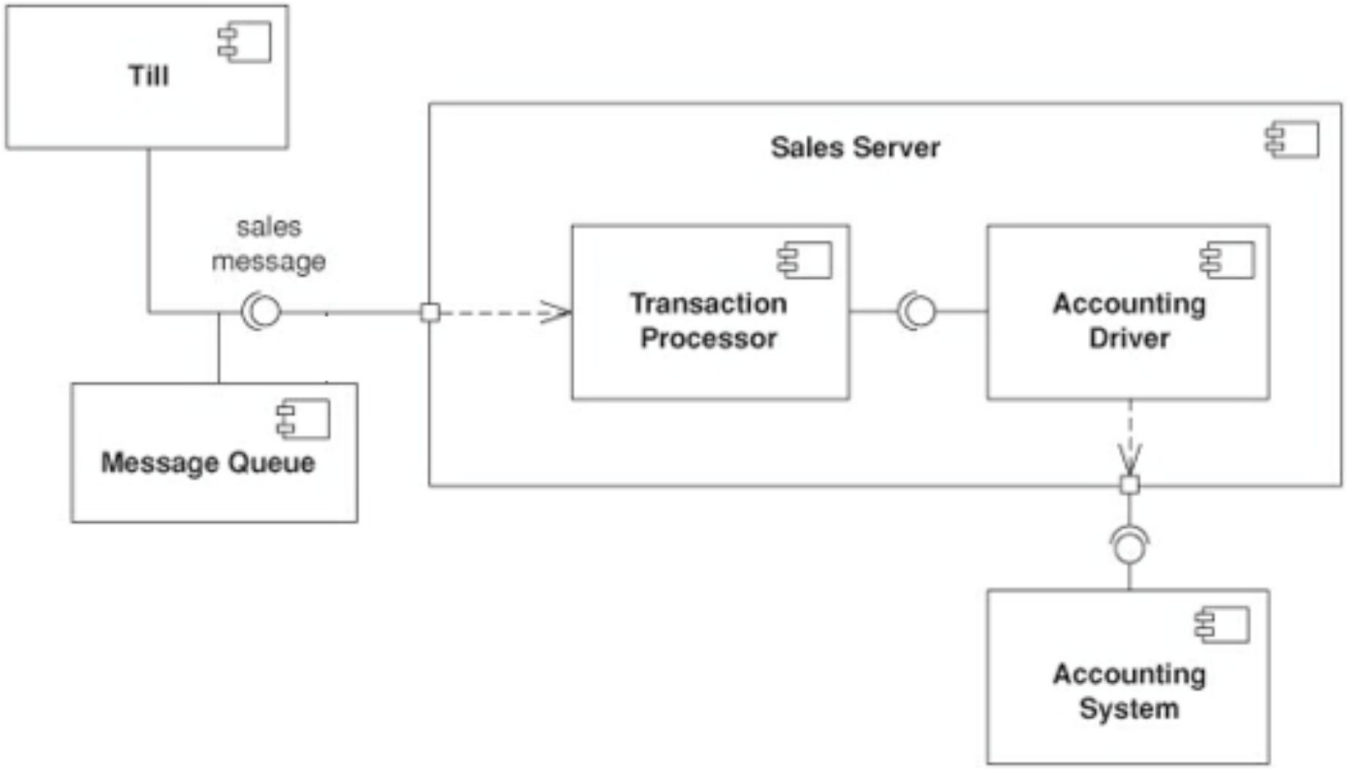
Modela la comunicación y dirección de los mensajes entre clases. Pone particular énfasis en los flujos.



http://agilemodeling.com/style/collaborationDiagram.htm

4. Diagrama de Componentes

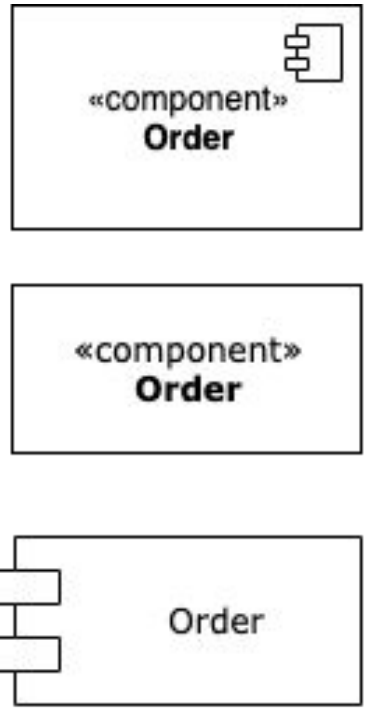
Modela la dependencia entre los componentes del software. Los componentes son aspectos del diseño de una solución de software más grandes y abstractos que una clase.



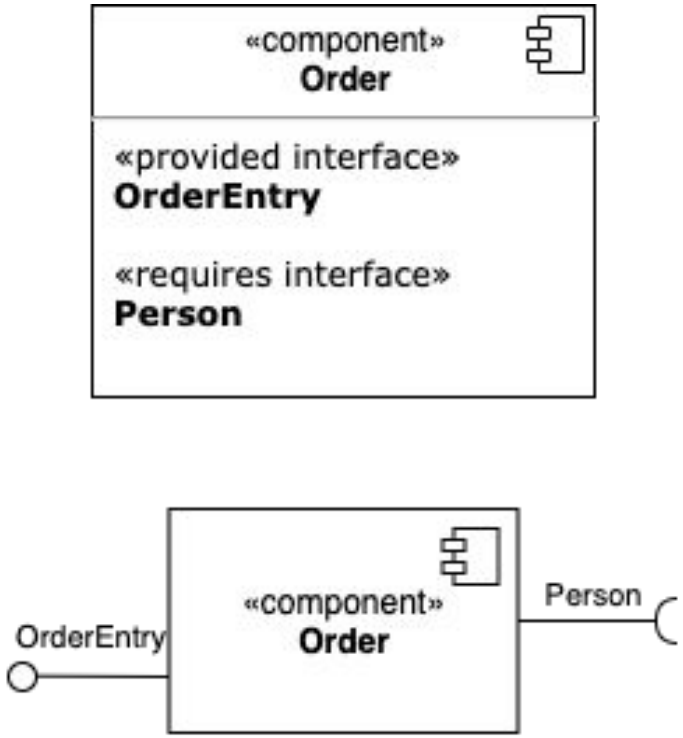
También se dice que modela la dependencia entre los componentes.

Diseño de Componentes - Notación

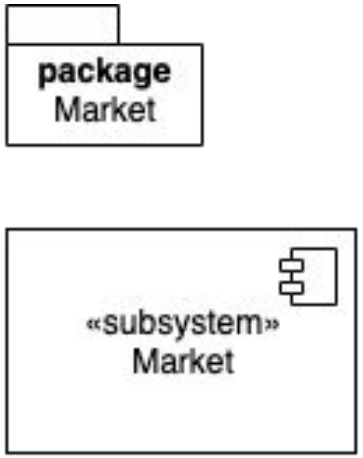
Ejemplos de componentes



Ejemplos de interfaces

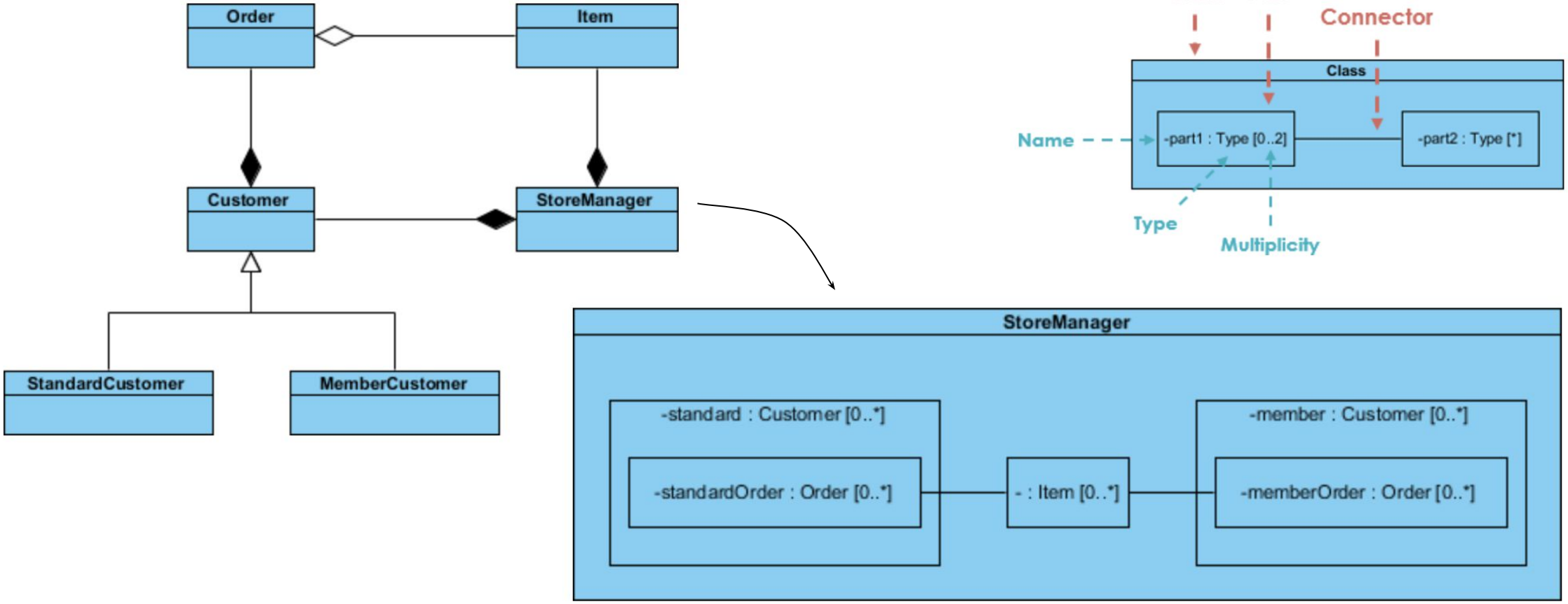


Otros elementos usados



5. Estructura compuesta

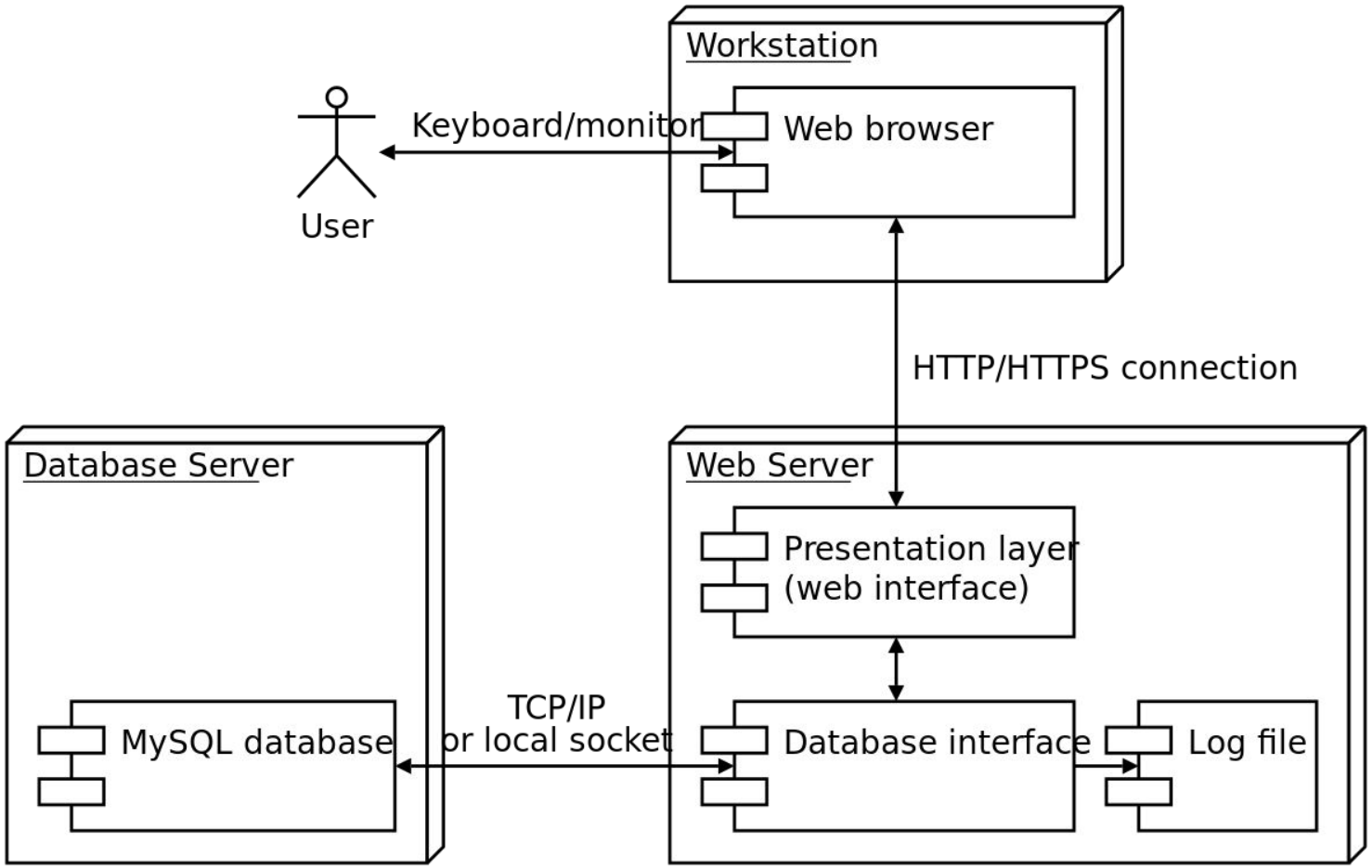
Modela la estructura interna o colaboración entre los elementos de una clase. Generalmente se usa para complementar otros diagramas.



https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-composite-structure-diagram/

6.Diagrama de despliegue

También conocido como diagrama de arquitectura.
Ilustra la ubicación física de cada componente de una solución de software compleja.

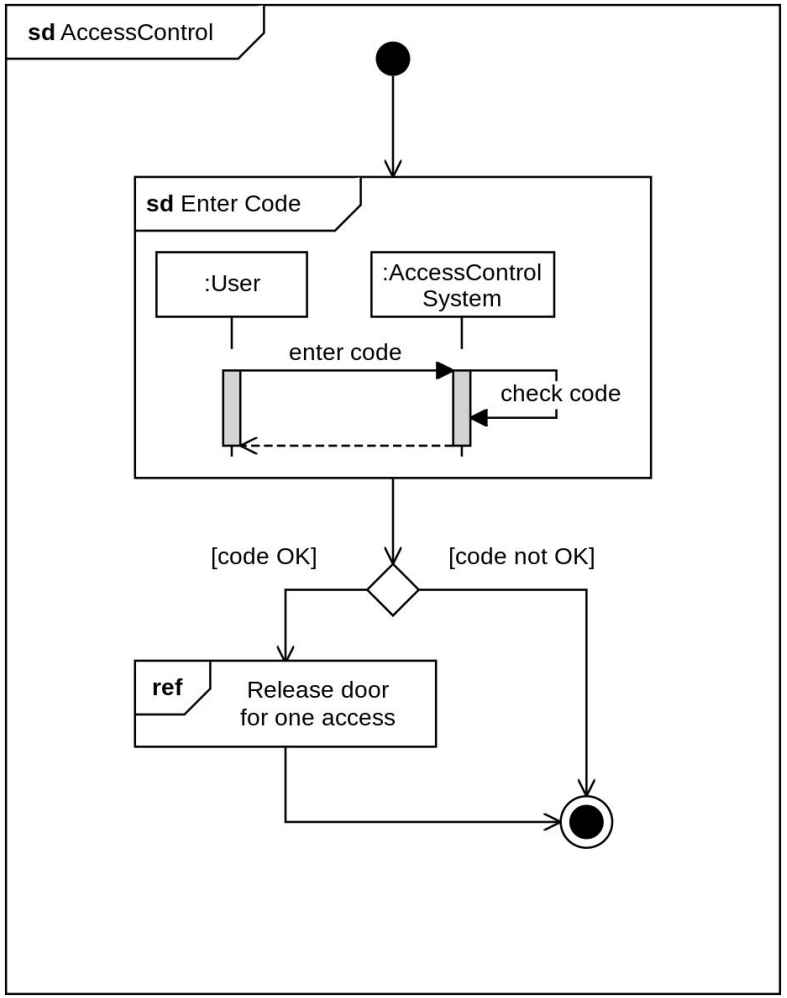


Se definen:
- **Nodos**: quien ejecuta el software, puede ser un hardware, una máquina, una máquina virtual, un contenedor, etc.-
- **Artefactos**: el build.
- **Canales de comunicación**: vía o protocolos por los cuales se comunican los distintos nodos y artefactos.

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-composite-structure-diagram/

7. Global de Interacciones

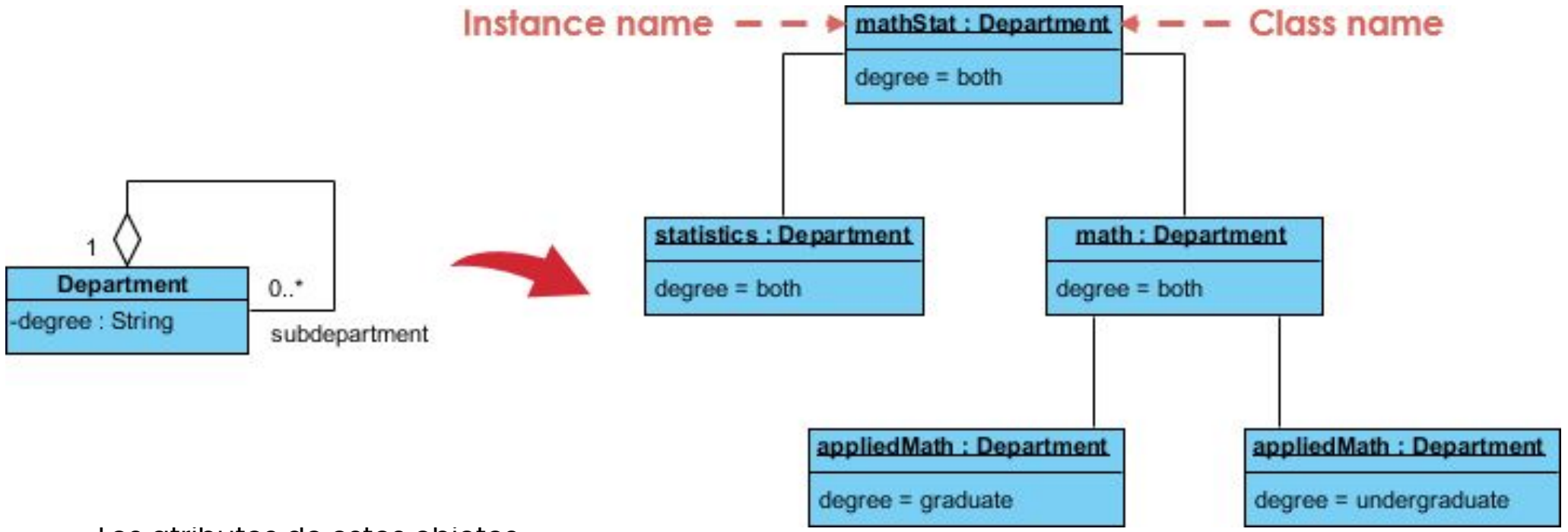
Similar al diagrama de actividad, con la diferencia que las actividades pueden ser representadas como diagramas de actividad (nesteado).



La notación es la misma que la del diagrama de actividades.

8. Diagrama de objetos

Diagrama que muestra un snapshot de una parte de la aplicación, ilustrando objetos instanciados presentes y sus referencias hacia otros objetos.

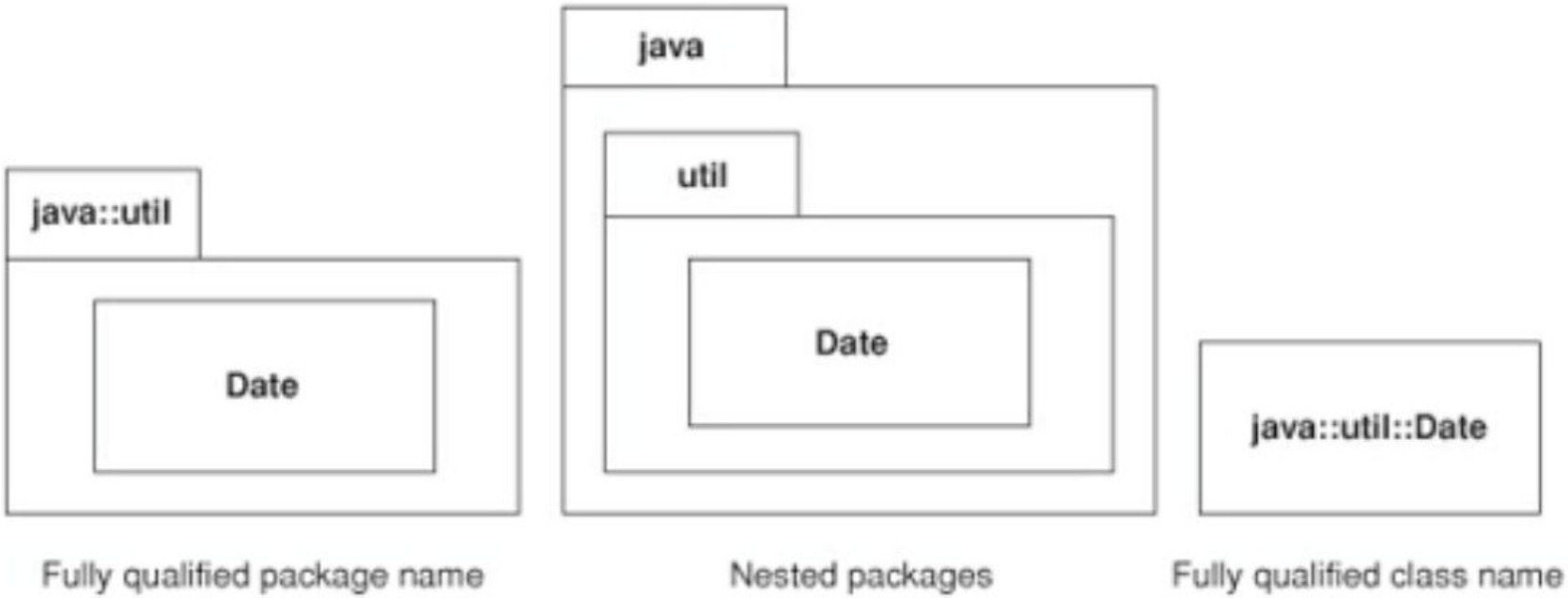


Los atributos de estos objetos pueden estar total o parcialmente definidos.

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-object-diagram/

9. Diagrama de paquetes

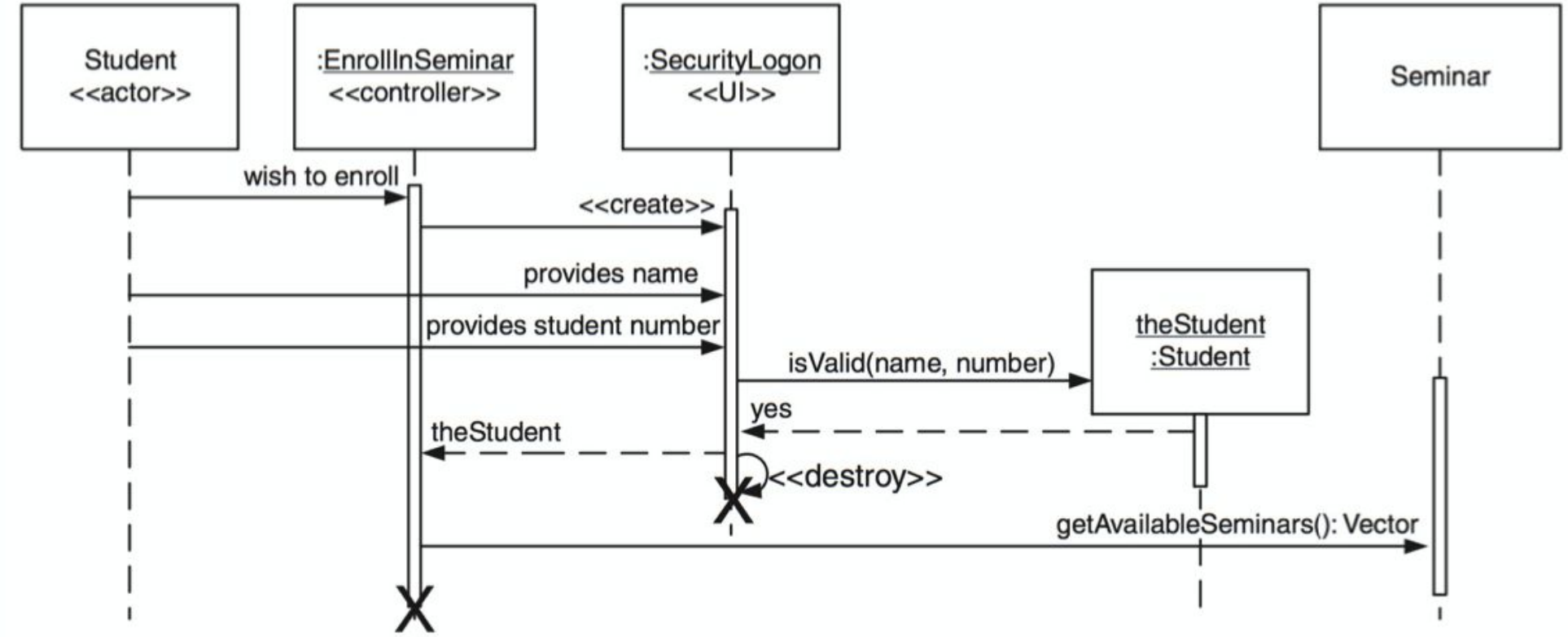
A medida que un sistema crece, puede volverse caótico gestionar el gran número de clases que define y se vuelve necesario agruparlas de manera lógica.



Los paquetes en UML se usan principalmente para agrupar conjuntos de clases, pero pueden usarse con otros componentes de UML también.

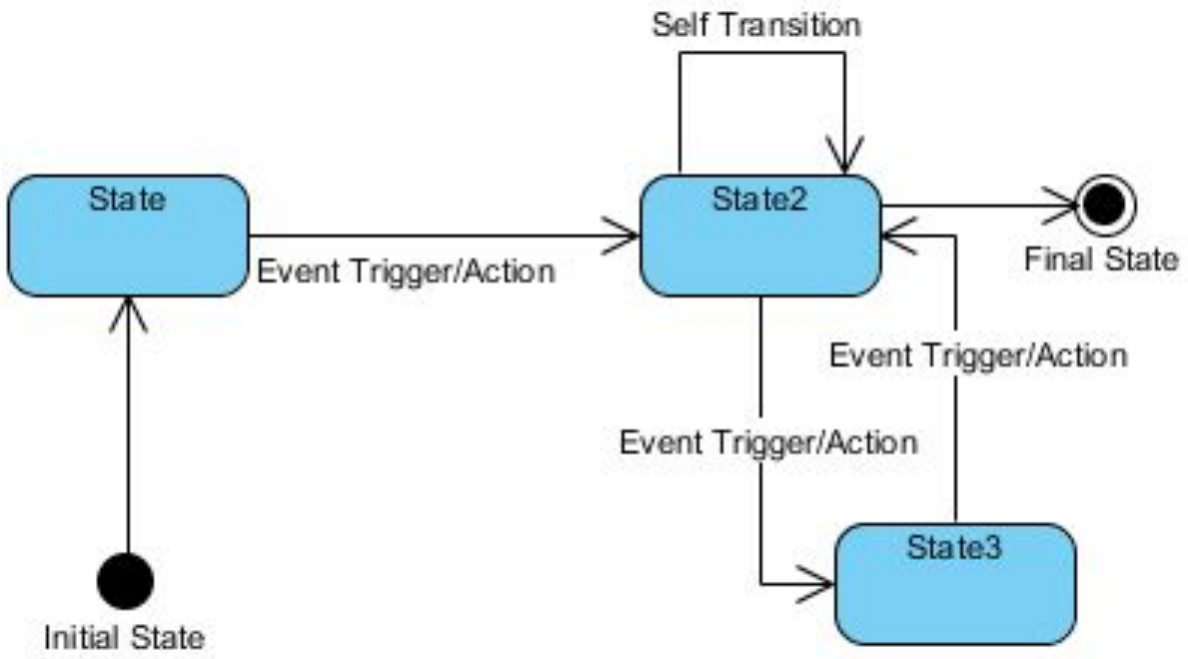
10. Diagrama de secuencia

Permite describir gráficamente transacciones (conjunto de operaciones) complejas entre múltiples agentes y cómo interactúan entre ellos.



11. Máquina de estados

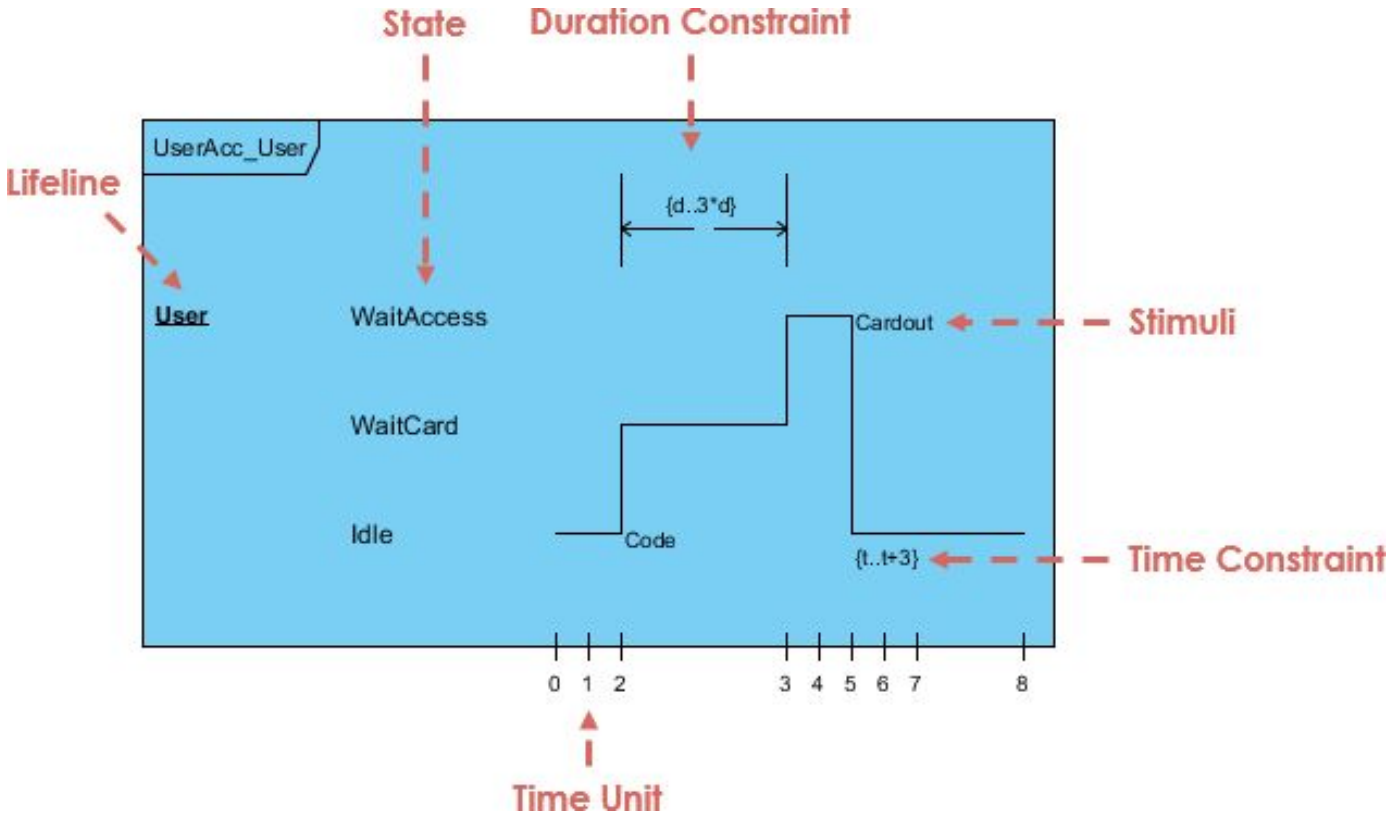
O Diagrama de Estados. Permiten definir el comportamiento de un sistema. Definen dos componentes esenciales: ***Estados*** *y* ***Transiciones***.



Describen como un sistema puede ir cambiando de estado a medida que se cumplan las condiciones delineadas en las transiciones.

12. Diagrama de tiempos

Permiten poner el énfasis en los cambios de estado de un sistema a medida que avanza el tiempo. Son muy útiles para representar interacciones complejas gatilladas solo por el paso del tiempo.

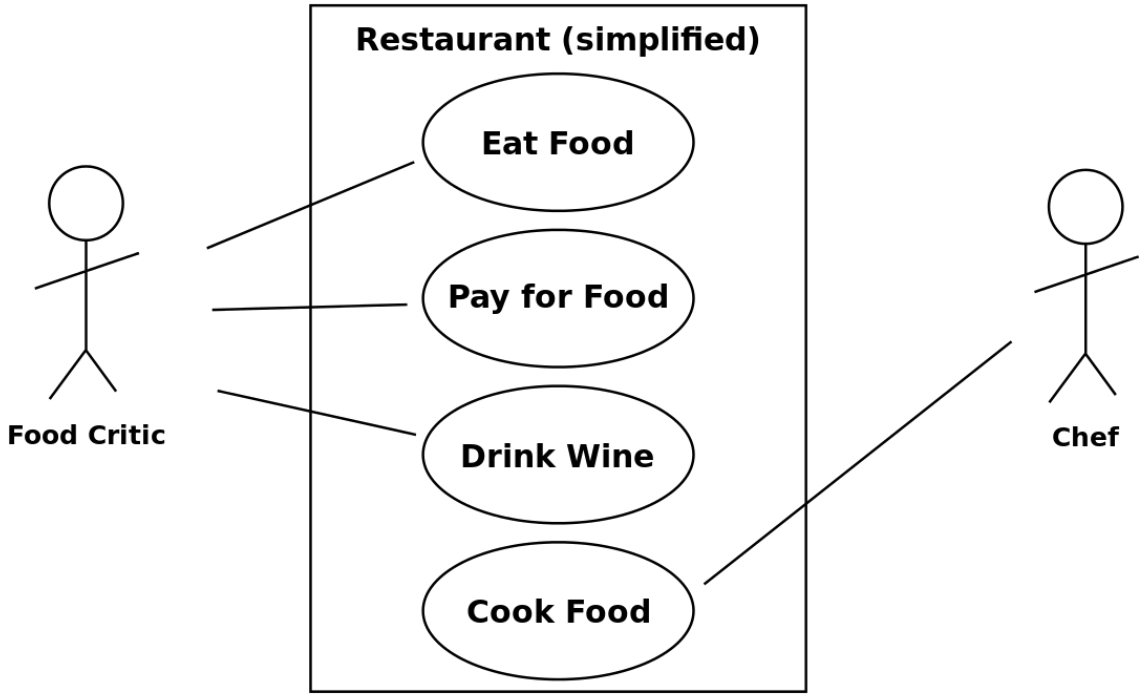


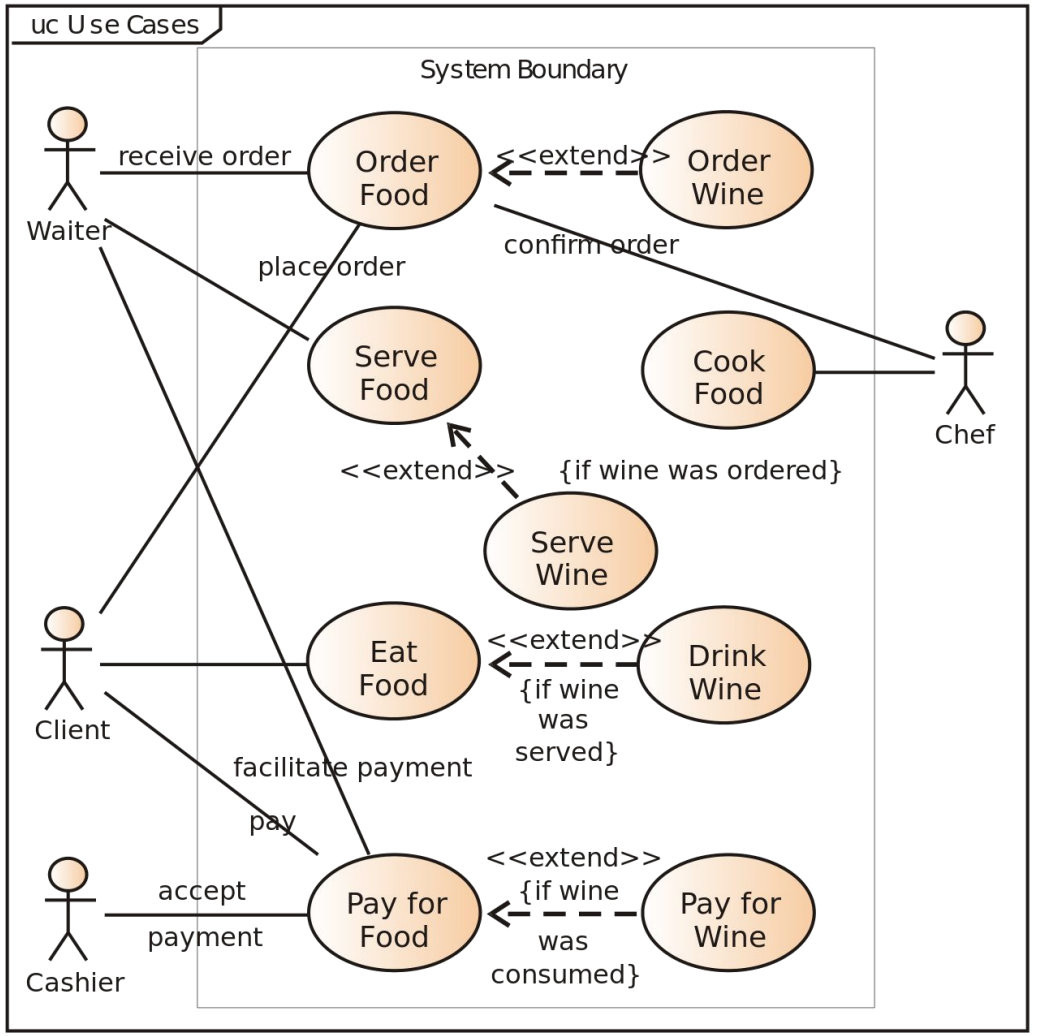
Aunque también soportan comunicación entre participantes mediante estímulos y paso de mensajes, pero no son aptos para ilustrar de manera legible comunicación en gran volumen

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-timing-diagram/

13. Diagrama de casos de uso

Estándar para graficar conjuntos de casos de uso de forma breve. Ilustra los nombres de los casos de uso, actores y relaciones entre ellos. Agrupan múltiples escenarios posibles de forma sintetizada.

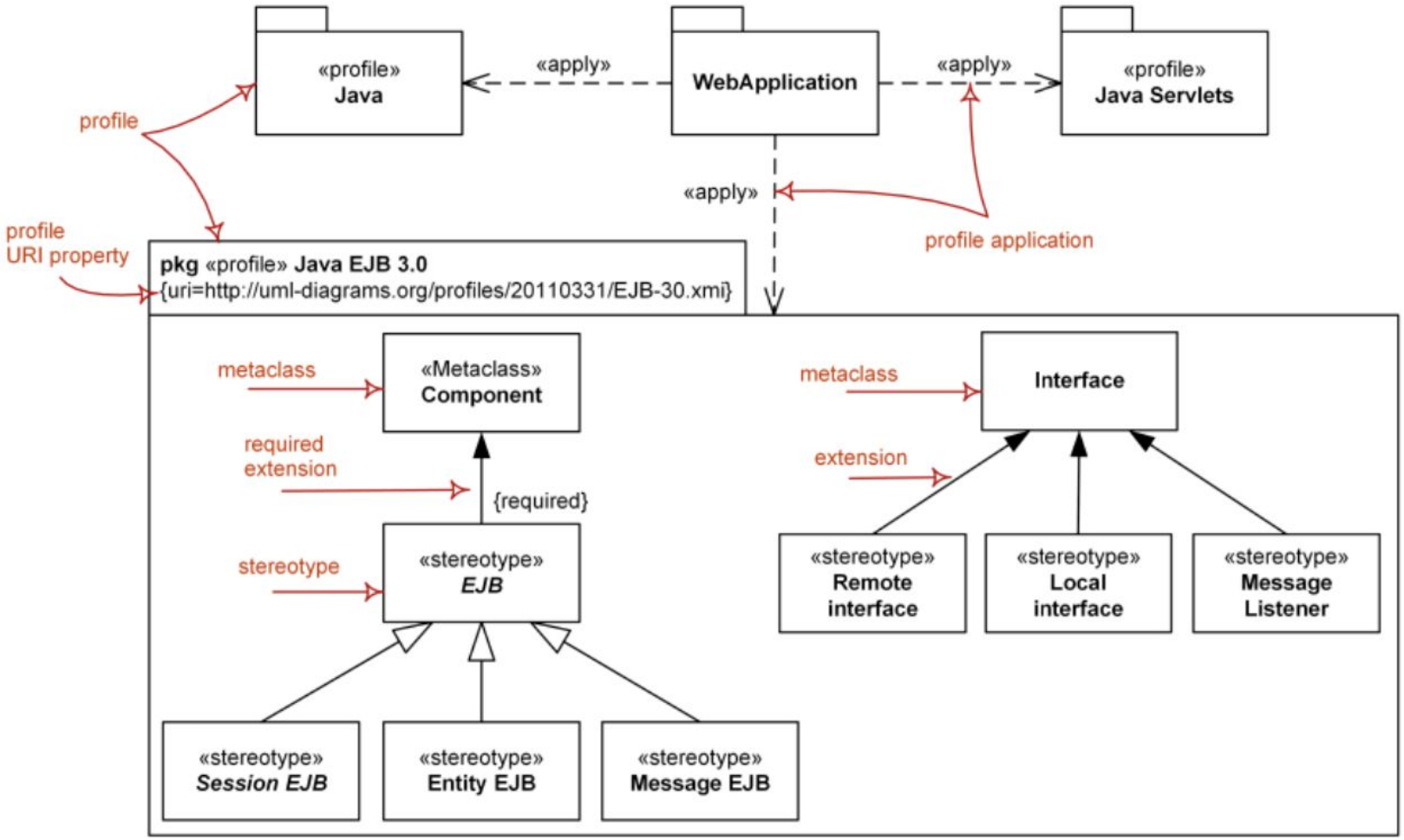




https://en.wikipedia.org/wiki/Use_case_diagram

14. Diagrama de perfil

Mecanismo para extender los modelos propuestos por UML a dominios o tecnologías específicas con elementos customizados.



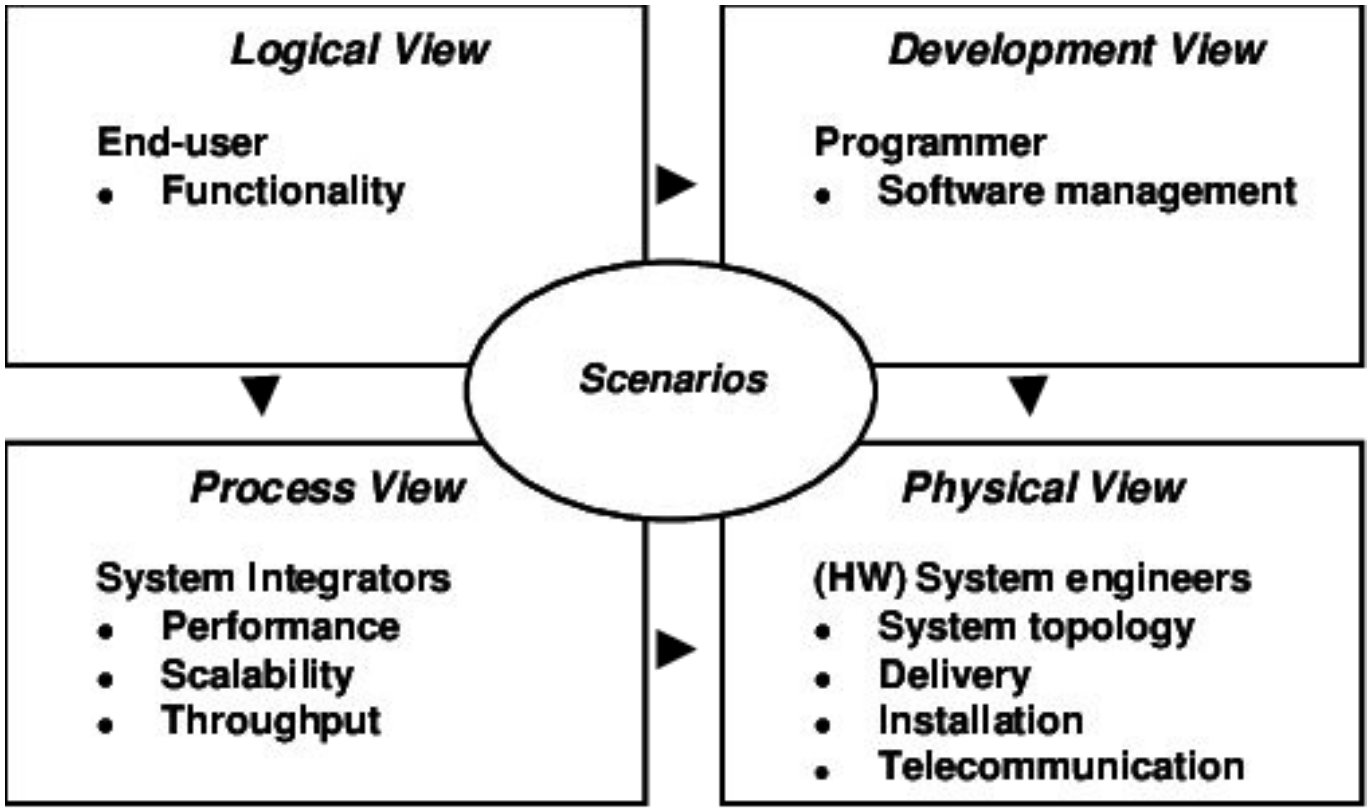
02. Modelo 4+1

¿Qué es el modelo 4+1?

Framework para describir la arquitectura de un software, basado en el uso de múltiples vistas concurrentes. Propuesto por Philippe Kruchten en 1995. Se basa en **vistas**. Una vista es una representación del sistema completo, enfocada desde una perspectiva con consideraciones específicas.

- Vista lógica
- Vista de implementación
- Vista Física
- Vista de escenarios

¿Qué es el modelo 4+1?



https://www.researchgate.net/publication/228459729_Architecture_for_Mechatronics_Software_Systems-A_Survey_of_Related_Concepts_Theories_and_Methods

Modelo 4+1 - Vista Lógica

Interesados: Usuarios finales.

Consideraciones: Requisitos Funcionales.

Objetivo: Intenta responder a la pregunta ¿Qué es lo que el sistema debería proveer a sus usuarios?.

Diagramas asociados: Diagrama de Clases, Diagrama de Comunicación, Diagrama de Secuencia, Diagrama de Estados.

Modelo 4+1 - Vista de procesos

Interesados: Analistas.

Consideraciones: Requisitos No Funcionales.

Objetivo: Intenta responder a la pregunta ¿Qué procesos y reglas tiene el sistema, y cómo interactúan los componentes?.

Diagramas asociados: Diagrama de Actividad.

Modelo 4+1 - Vista de implementación

Interesados: Desarrolladores y Líderes de proyectos.

Consideraciones: Organización del Software.

Objetivo: Intenta responder a la pregunta ¿Cómo se organiza el software del sistema?.

Diagramas asociados: Diagrama de Componentes, Diagrama de Paquetes (arquitectura).

Modelo 4+1 - Vista de física

Interesados: Arquitectos de Sistemas.

Consideraciones: Requisitos No Funcionales.

Objetivo: Intenta responder a la pregunta ¿Cómo es el ambiente de ejecución del sistema?.

Diagramas asociados: Diagrama de Despliegue..

Modelo 4+1 - Vista de escenarios

Interesados: Todos.

Consideraciones: Consistencia y validez.

Objetivo: Intenta responder a la pregunta ¿Qué es lo que el sistema debería hacer?.

Diagramas asociados: Diagrama de Casos de Uso.

¿Hay más diagramas aparte de UML?

- Por supuesto que sí.
  - El diagrama ERD (Entity Relationship Diagram) enfocado a bases de datos.
  - En UX existen diagramas enfocados a las vistas: wireframes. Además modificaciones de diagramas de flujo: task flow, wire flow, user flow, screen flow, sitemaps, entre otros.
  - En marketing existen diagramas enfocados a usuarios: empathy map, customer journey map, entre otros.
  - Existen diagramas exclusivos por metodología: para Agile, para LEAN, para Design Thinking, etc.-

03. Próxima Clase

Próxima clase

- Buenas prácticas de desarrollo
  - SOLID
  - Métricas de calidad
  - Code Smells
  - Refactoring
  - Testing

Clase 2 - UML

Bibliografía

- M. Fowler, UML Distilled.
- S. Ambler, The Elements of UML(TM) 2.0 Style.
- Kiruthika, R.S., Shobana, C. 4+1 View Model.
- P.B. Kruchten. The 4 + 1 view model of architecture.

IIC2113 - Diseño Detallado de Software

Fernanda Sepúlveda - mfsepulveda@uc.cl

Pontificia Universidad Católica de Chile
2020-2