Lenguaje UML

Booch Grady, James Rumbaugh, Jacobson Ivar. El lenguaje unificado de modelado. Guía del Usuario. Editorial Addison Wesley. Capítulos 1y 2 (págs. 3- 34).

Aprenda UML en 24 horas. Introducción al UML: HORA 1

Contenido

1 Perspectiva general del UML

1.1 Historia del UML

Años 70 y 80: metodologías de desarrollo estructurado.

- Usaban lenguajes de programación como Pascal, Cobol, Fortran.
- Desarrollo de grandes sistemas, especialmente contratados por el gobierno.
- Procesos de desarrollo organizado y con una amplia documentación del diseño e implementación del sistema.
- Las empresas fueron más reacias a adoptar estos métodos de desarrollo. Sus desarrollos eran internos según sus propias necesidades (sin la interacción entre cliente y contratista).
- Los sistemas comerciales se percibían como más simples y por tanto había menos necesidad de una revisión por parte de una organización externa.

1.1 Historia del UML

Años 80 y 90: metodologías de desarrollo orientada a objetos.

- Aparece el primer lenguaje orientado a objetos **Simula67** (1967), pero el movimiento de la orientación a objetos se dio con la difusión de **Smalltalk** a principio de los 80, seguido por otros lenguajes como Objetive C, C++, Eiffcl, y CLOS.

- Varios autores aportaron con publicaciones en el campo de la metodología orientada a objetos:
  - Shlaer/Mellor [Shlaer-88]
  - Coad/Yourdon [Coad-91]
  - Libros de Booch [Booch-91], Rumbaugh/Blaha/Premerlani/Eddy/Lorensen [Rumbaugh-91]
  - Wirfs-Brock/Wilkcrson/Wiener[Wirfs-Brock-90].
  - Goldberg/Robson [Goldberg-83], Cox [Cox-86], y Meyer [Meyer-88]

1.1 Historia del UML

Años 90 en adelante.

- Empezó un proceso de unificación. El primer intento exitoso de combinar y reemplazar los métodos existentes llegó cuando **Rumbaugh** se unió a **Booch** en **Rational Software Corporation en 1994**.

- Se dio como resultado una primera propuesta en 1995 llamada **Lenguaje Unificado de Modelado** (UML). En noviembre de 1997 se adoptó como ESTÁNDAR*.

*Estándar: Que sirve de patrón, modelo o punto de referencia

1.2. Objetivos del UML

- Es un Lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.

- UML no pretende ser un método de desarrollo completo sino trabajar con todos, o al menos con la mayoría de los procesos de desarrollo existentes.

- Pretende ser tan simple como fuera posible pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir.

2

La naturaleza y propósito de los modelos

2.1 ¿Qué es un modelo?

- Un modelo es una representación.
- Capta los aspectos importantes y simplifica u omite el resto.
- Un modelo de software se construye en un lenguaje de modelado, como UML. Tiene semántica y notación y puede adoptar varios formatos que incluyen texto y gráficos.
- Pretende ser más fácil de usar para ciertos propósitos que el sistema final.

2.2 ¿Para qué sirven los modelos?

- Para captar y enumerar los requisitos y el dominio de conocimiento para que todos los implicados puedan entenderlos y estar de acuerdo con ellos.

  Los diversos modelos de un sistema de software pueden capturar requisitos sobre su dominio de aplicación, las formas en que los usuarios lo utilizarán, su división en módulos, los patrones comunes usados en su construcción, y otras cosas.

- Para pensar el diseño de un sistema.

  Un modelo de un sistema software ayuda a los desarrolladores a explorar varias arquitecturas y soluciones de diseño fácilmente, antes de escribir el código.

2.2 ¿Para qué sirven los modelos?

- Para organizar, encontrar, filtrar, recuperar, examinar, y corregir la información en grandes sistemas.

  Un modelo de un sistema de software organiza la información en varias vistas. Cada vista es una proyección del modelo completo para un propósito determinado.

- Para explorar económicamente múltiples soluciones.

  Los modelos de un sistema de software grande permiten que se propongan y comparen varios diseños. Los modelos no se construyen al detalle, puede exponer muchas cuestiones que el diseño final debe tener en cuenta.

2.3 Qué hay en un modelo

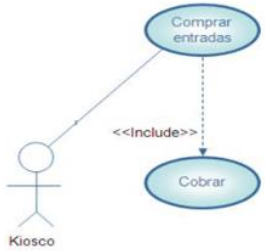
Los modelos tienen: Información semántica (**semántica**) y presentación visual (**notación**).

Semántica

- Capta el significado de una aplicación como una red de construcciones lógicas, por ejemplo clases, asociaciones, estados, casos de uso, y mensajes.
- Los elementos semánticos llevan el significado del modelo, es decir, transportan la semántica.
- Un modelo semántico tiene **reglas** para asegurar su corrección, y dinámicas de ejecución.

Notación

- Muestra la información semántica de modo que se pueda ser considerada, hojeada y corregida por los seres humanos.
- Los elementos de la presentación llevan la presentación visual del modelo (símbolos)



3 Un paseo por UML

3.1. Áreas conceptuales del UML

1. Estructura estática
2. Comportamiento dinámico.
3. Construcciones de implementación
4. Organización del modelo.
5. Mecanismos de extensión.

3.1. Áreas conceptuales del UML

1. Estructura estática

Define el universo del discurso, esto es, los conceptos clave de la aplicación, sus propiedades internas, y las relaciones entre cada una. Este conjunto de construcciones se conoce como **vista estática**.

Proporciona la base sobre la cual se construye el comportamiento dinámico.

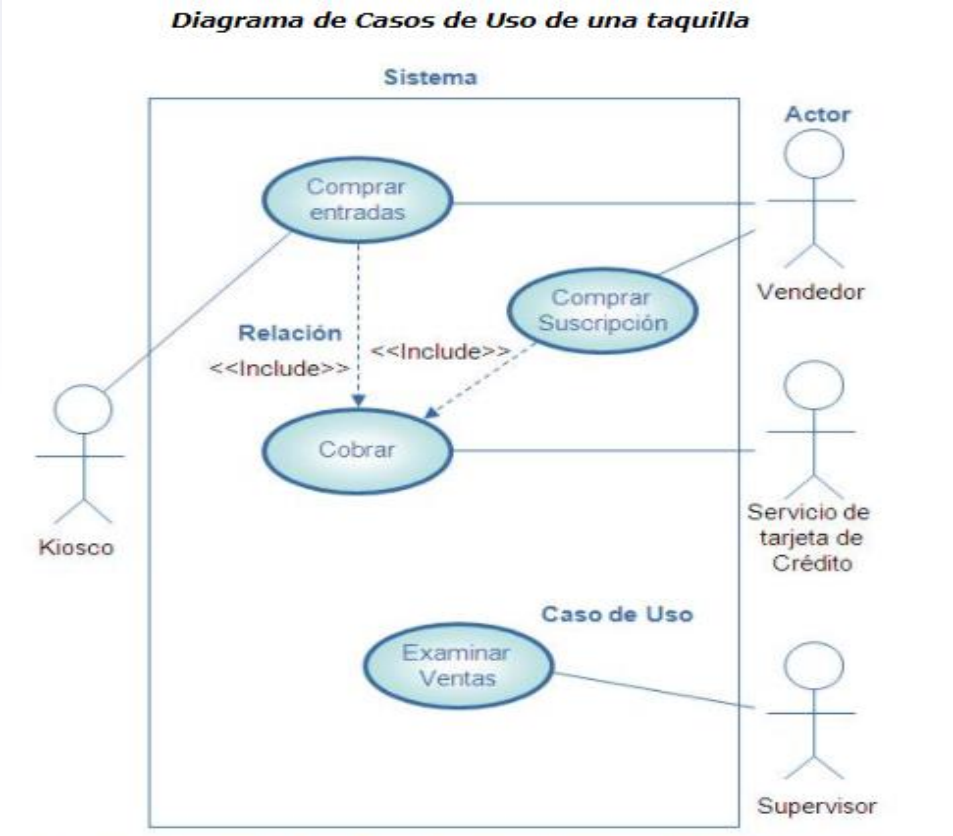
Se expresa con:

- Casos de uso
- Diagrama de clases

1. Estructura estática

Casos de uso

Modela la funcionalidad del sistema según lo perciben los usuarios externos, llamados actores. Un caso de uso es una unidad coherente de funcionalidad, expresada como transacción entre los actores y el sistema. El propósito de la vista de casos de uso es enumerar a los actores y los casos de uso, y demostrar qué actores participan en cada caso de uso.

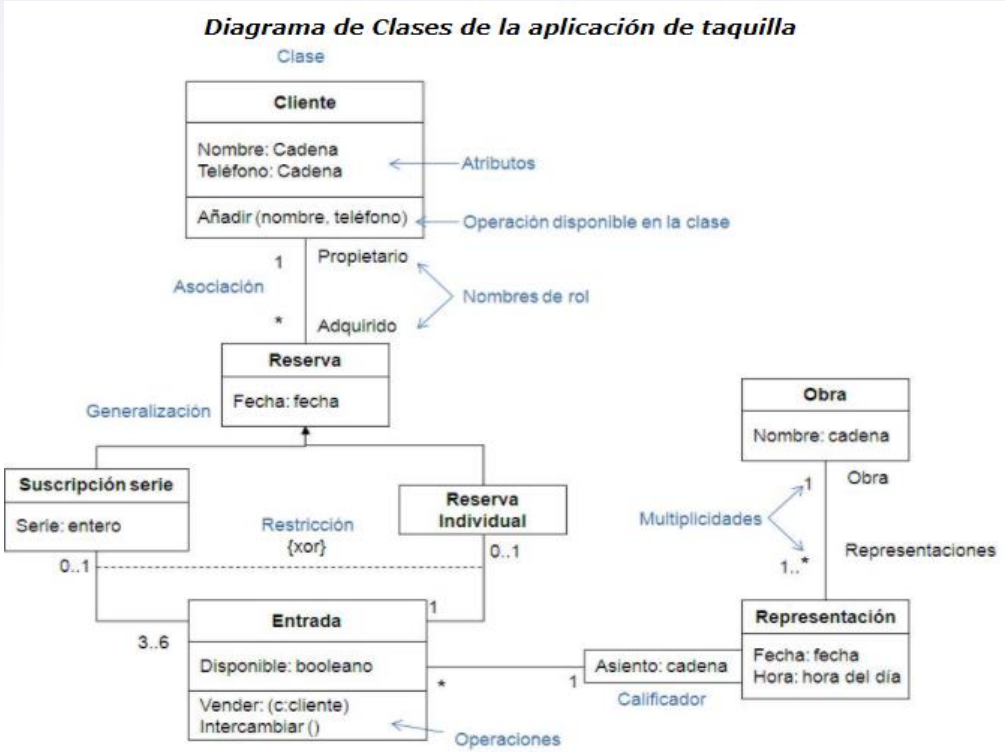


Fuente: Rumbaugh, James; Jacobson Ivar y Booch GRady, 2000

1. Estructura estática

Diagrama de clases

Modela los conceptos del dominio de la aplicación, así como los conceptos internos inventados como parte de la implementación de la aplicación. Esta visión es estática porque no describe el comportamiento del sistema dependiente del tiempo, que se describe en otras vistas.



Fuente: Rumbaugh, James; Jacobson Ivar y Booch GRady, 2000

3.1. Áreas conceptuales del UML

2. Comportamiento dinámico.

Describe el comportamiento de un sistema en el tiempo. El comportamiento se puede describir como serie de cambios a las fotos del sistema dibujadas a partir de la visión estática.

Hay dos formas de modelar el comportamiento:

1. La **interacción de los objetos** que se expresa mediante los diagramas de secuencia, diagramas de actividad y los diagramas de colaboración.
2. Los patrones de **comunicación entre los objetos** que se expresan con el diagrama de estados.

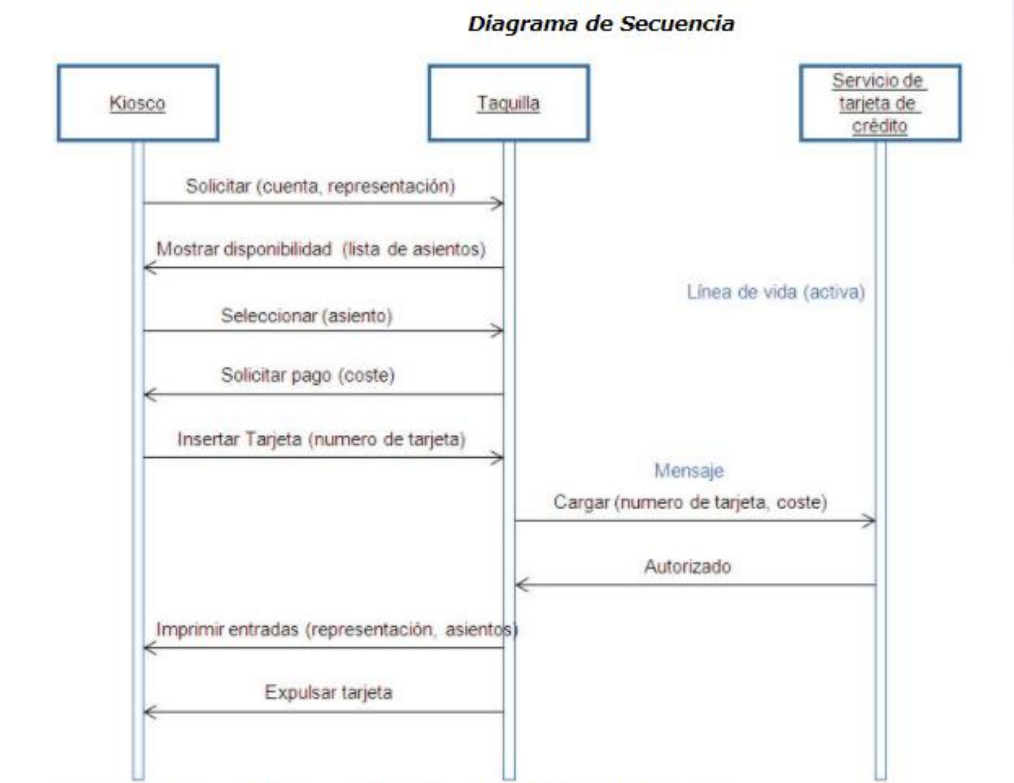
2. Comportamiento dinámico.

Vista de interacción: el diagrama de secuencia.

Muestra un conjunto de mensajes, dispuestos en una secuencia temporal.

Cada rol en la secuencia se muestra como una línea de vida, es decir, una línea vertical que representa el rol durante cierto plazo de tiempo, con la interacción completa.

Los mensajes se muestran como flechas entre las líneas de vida.



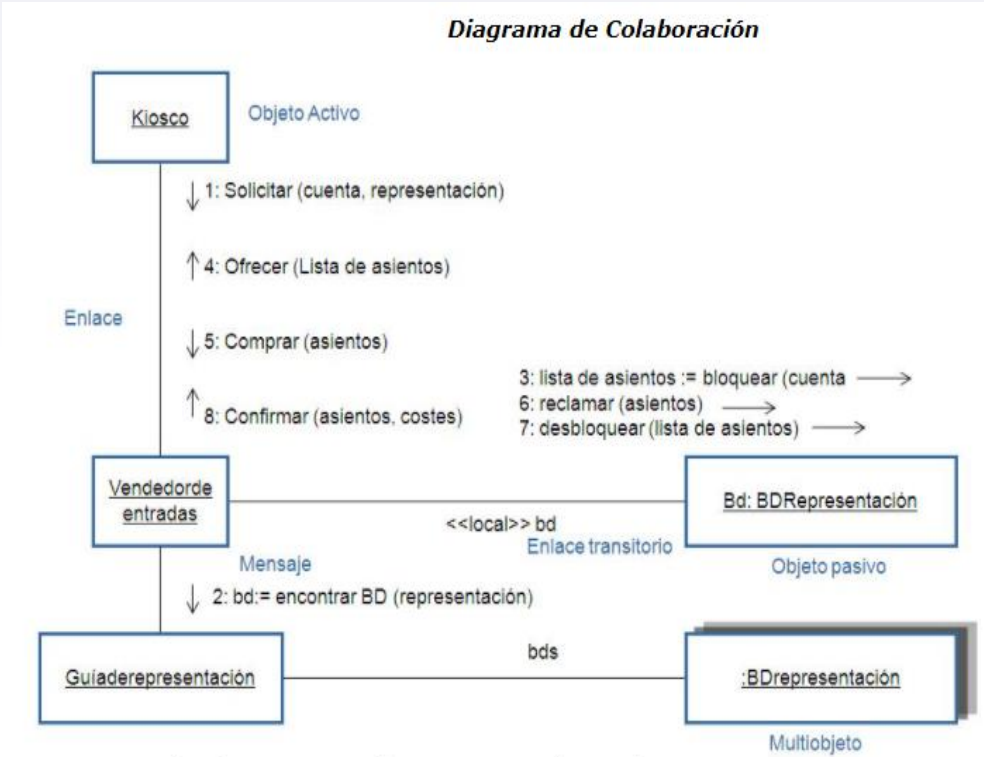
Fuente: Rumbaugh, James; Jacobson Ivar y Booch GRady, 2000

2. Comportamiento dinámico.

Diagrama de colaboración.

Modela los objetos y los enlaces significativos dentro de una interacción.

Un rol describe un objeto, y un rol en la asociación describe un enlace dentro de una colaboración. Los mensajes se muestran como flechas, ligadas a las líneas de la relación, que conectan a los roles. La secuencia de mensajes, se indica con los números secuenciales que preceden a las descripciones del mensaje.



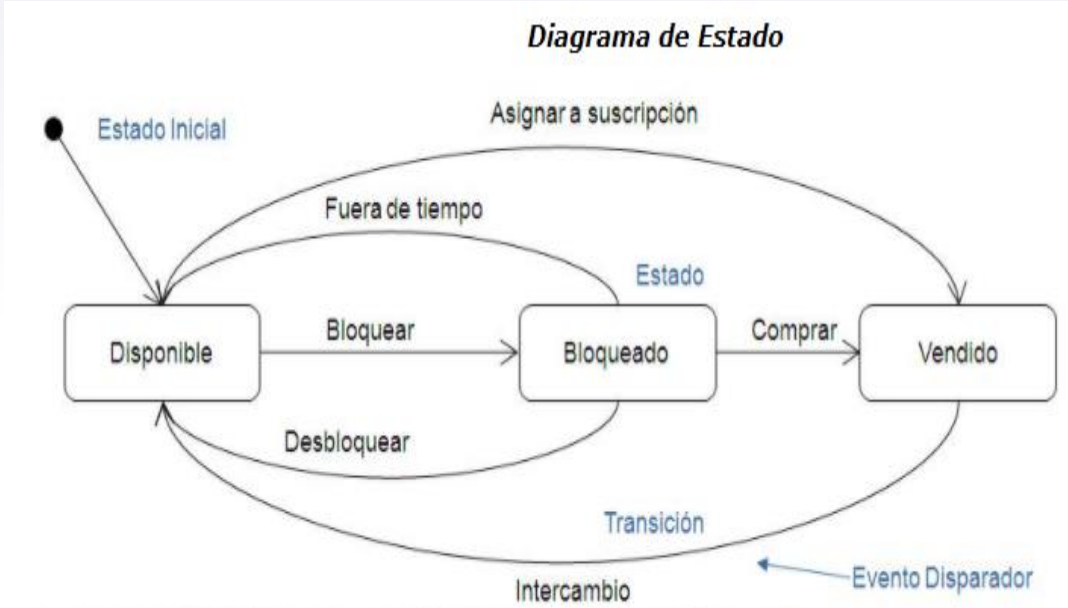
Fuente: Rumbaugh, James; Jacobson Ivar y Booch GRady, 2000

2. Comportamiento dinámico.

Diagrama de estados.

Modela las posibles historias de vida de un objeto de una clase.

Una máquina de estados contiene los estados conectados por transiciones. Cada estado modela un período de tiempo, durante la vida de un objeto, en el que satisface ciertas condiciones. Cuando ocurre un evento, se puede desencadenar una transición que lleve el objeto a un nuevo estado del mensaje.



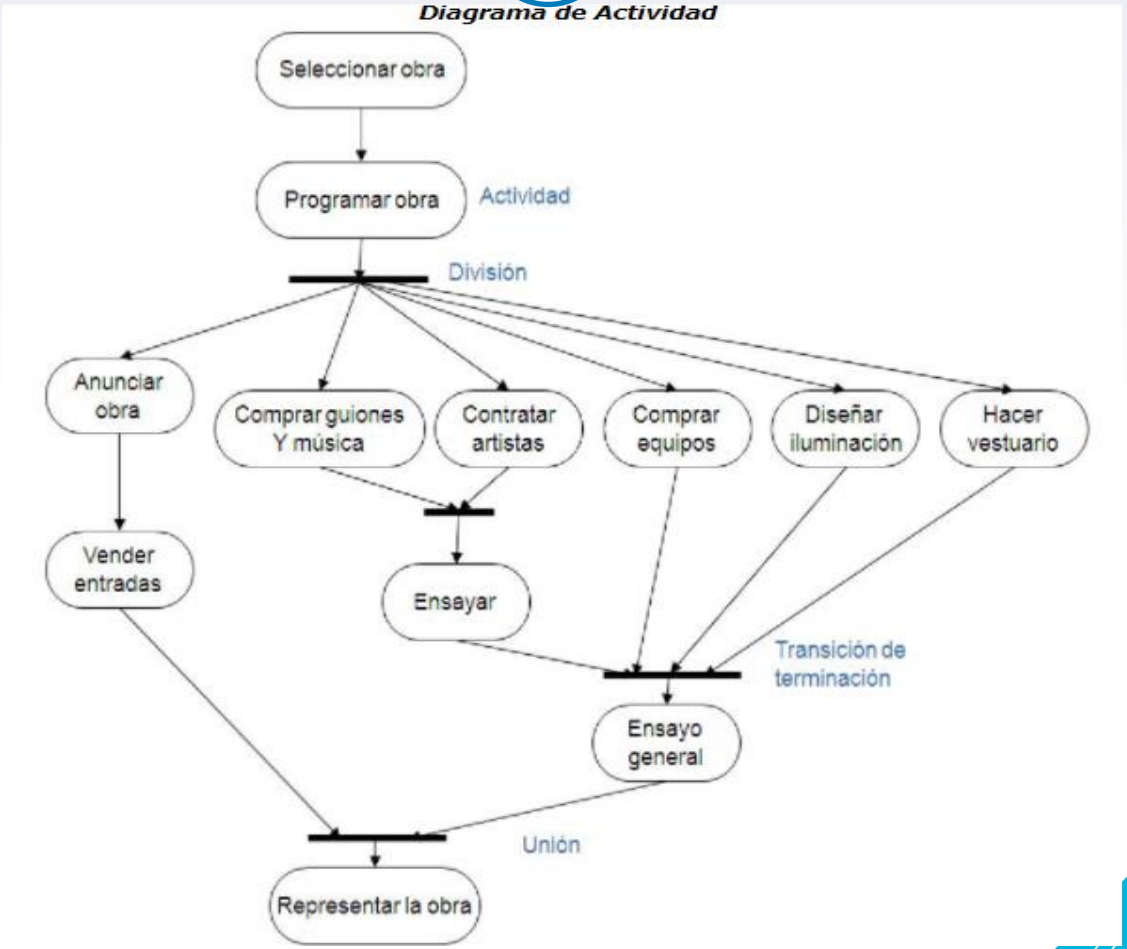
Fuente: Rumbaugh, James; Jacobson Ivar y Booch GRady, 2000

2. Comportamiento dinámico.

Diagrama de actividad

Es una variante de un diagrama de estados, que muestra las actividades de computación implicadas en la ejecución de un cálculo. Un estado de actividad representa una actividad: un paso en el flujo de trabajo o la ejecución de una operación.

Un diagrama de actividades describe grupos secuenciales y concurrentes de actividades.



3.1. Áreas conceptuales del UML

3. Construcciones de implementación

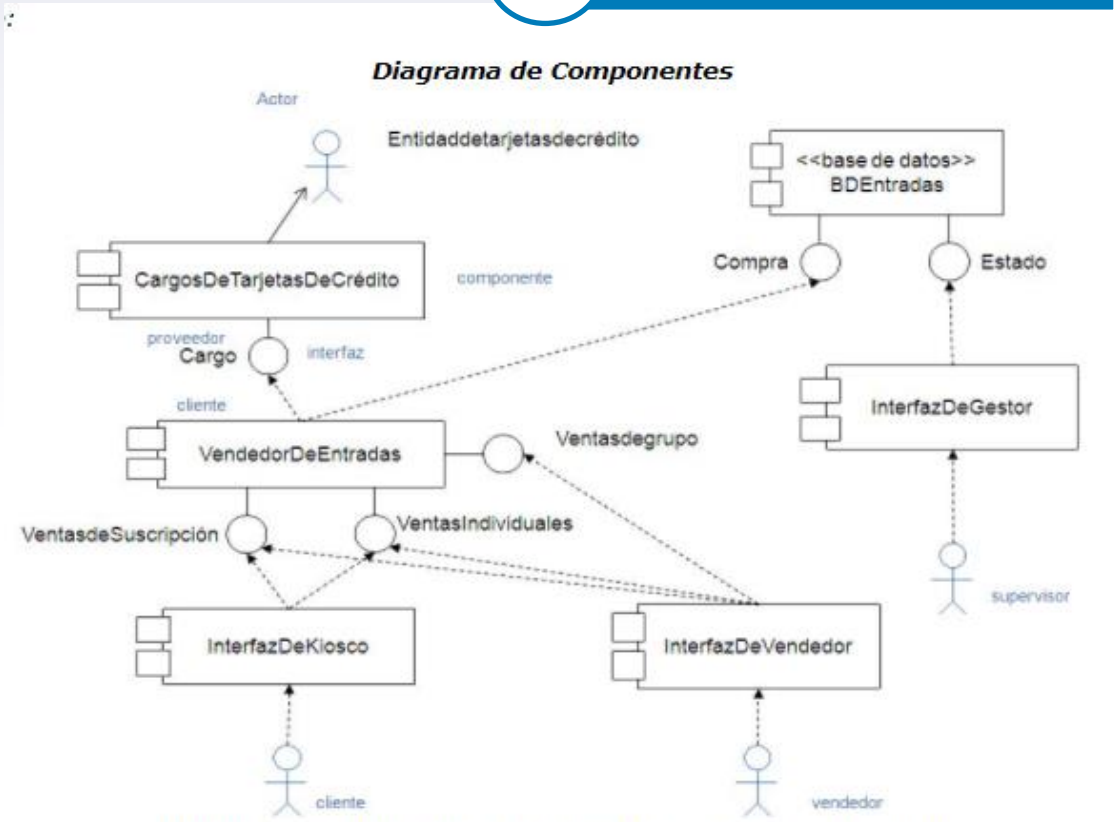
Muestra las relaciones entre los componentes físicos y lógicos del sistema final (hardware y software).

- La vista de implementación se expresa con el diagrama de componentes y el diagrama de despliegue.

3. Construcciones de implementación

Diagrama de componentes

Modela los componentes de un sistema (a partir de los cuales se construye la aplicación) así como las dependencias entre los componentes, para poder determinar el impacto de un cambio propuesto.



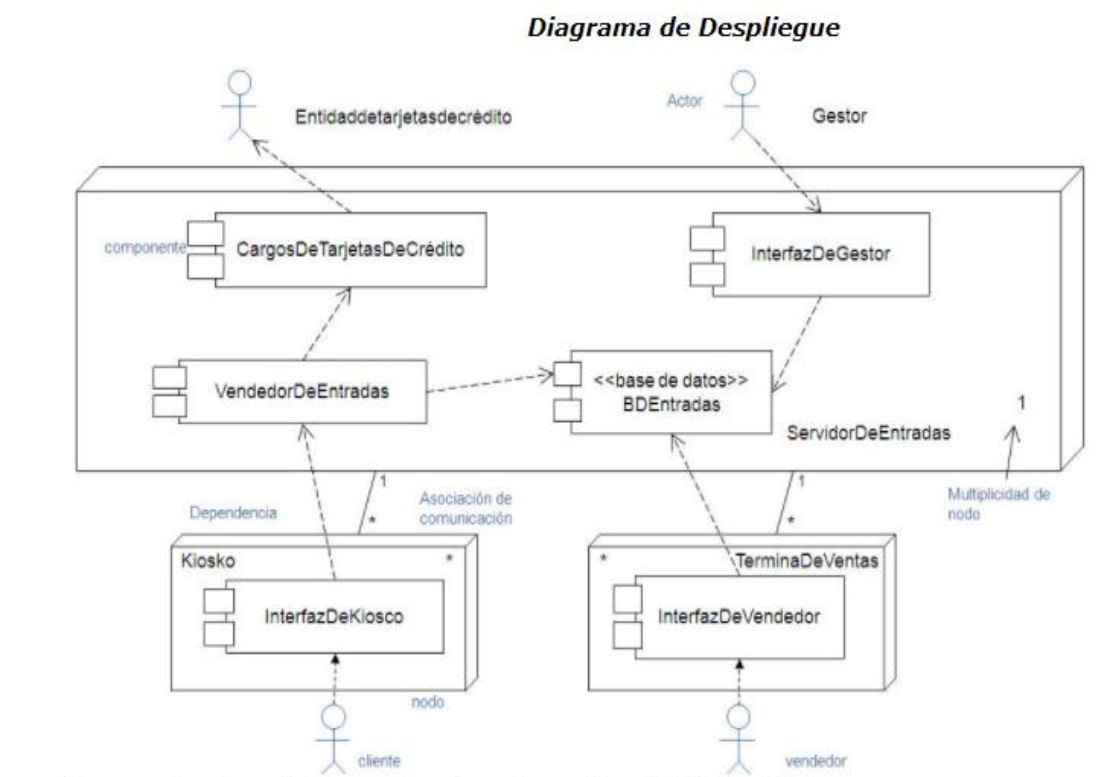
Fuente: Rumbaugh, James; Jacobson Ivar y Booch GRady, 2000

3. Construcciones de implementación

Diagrama de despliegue

La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos.

Un nodo es un recurso de ejecución, tal como una computadora, un dispositivo, o memoria. Esta vista permite determinar las consecuencias de la distribución y de la asignación de recursos.



Fuente: Rumbaugh, James; Jacobson Ivar y Booch GRady, 2000

3.1. Áreas conceptuales del UML

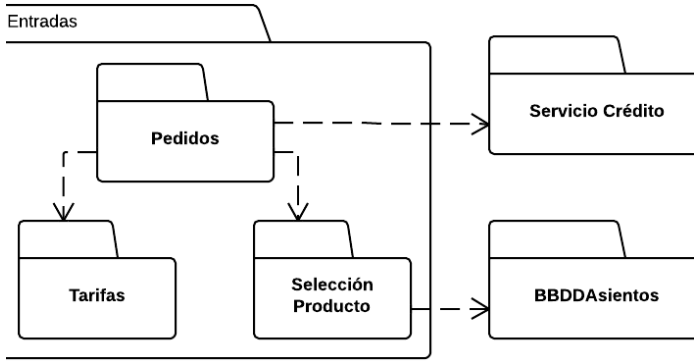
4. Organización del modelo.

Los ordenadores pueden manejar grandes modelos "planos", pero los humanos no. En un sistema grande, la información del modelo debe ser dividida en piezas coherentes, para que los equipos puedan trabajar en las diferentes partes de forma concurrente.

La organización se hace por medio de los paquetes. Estos son unidades organizativas, jerárquicas, y de propósito general de los modelos de UML.

4. Organización del modelo.

- Un modelo abarca un conjunto de paquetes que contienen los elementos del modelo, tales como clases, máquinas de estados, y casos de uso.
- Los paquetes pueden contener otros paquetes: por lo tanto, un modelo señala un paquete raíz, que contiene indirectamente todo el contenido del modelo.



3.1. Áreas conceptuales del UML

5. Mecanismos de extensión.

No importa cuan completo sea el conjunto de "facilidades" de un leguaje, la gente querrá hacer extensiones.

- Las extensiones incluye **estereotipos**. Estas son es una nueva clase de elemento de modelado con la misma estructura que un elemento existente pero con restricciones adicionales, una interpretación diferente de un icono, y un tratamiento diferente por los generadores de código y otras herramientas de bajo nivel.


Gracias