GUIA EGEL CENEVAL

Ingeniería Software

CENEVAL
Raúl García Mateos

Contenido

Análisis de Sistemas de Información.

Diseño Orientado a objetos con UML.

Introducción al modelado orientado a objetos.

Modelo relacional. Modelo con fuerte base matemática. Supuso el nacimiento de las BD y los grandes sistemas de información.

Lenguajes Orientados a Objetos. Revolución en la industria del software.

UML (Unified Modeling Languaje). Pretende unificar tres metodologías más difundidas OMT, Bootch y OOSE, intentando que la industria de software termine su maduración como ingeniería. UML abarca todas las fases de un proyecto de desarrollo de software.

Modelado.

Objetivo. Conseguir Software de calidad, duradero y fácil de mantener. El modelo orientado a objetos tiende al refinamiento sucesivo para llegar a la fase de implementación con un diseño lo suficientemente explicitio.

- Idear una solida base arquitectónica, flexible al cambio.
- Desarrollar Software rápida y eficientemente.
- Minimizar el trabajo de recodificación.

Modelo.

Un modelo es una simplificación de la realidad.

- Nos apoyan a visualizar como es o queremos un sistema.
- Permiten especificar la estructura o comportamiento de un sistema.
- Nos proporciona plantillas para la contruccion de un sistema.
- Docuemntan las decisiones que hemos tomado.

Principiuos básicos de modelado.

1. Eleccion del modelo. Influye directamente sobre como se acomete el problema. El modelo orientado a objetos proporciona una arquitectura mas flexible y readaptable que los modelos orientados a funcionalidad o datos.
2. Nivel de precisión. Seleccionar el nivel de detalle necesario para cada fase del proyecto.
3. Modelos ligados a la realidad. Representar lo mas claramente posible los requisistos del sistema.
4. Modelo independientes. Construi modelos independientes que representen las partes mas diferenciadas del sistema y sus interrelaciones y que en conjunto representen la totalidad del sistema.

Orientacion a Objetos.

Programacion estructurada.

Los datos y el código se tratan por separado. Se construyen funciones y procedimientos para tratar esos datos pasándose de uno a otro hasta obtener un resultado.

Algoritmos + Estructura de Daros = Programas

Programacion Orientada a Objetos.

POO o OPP (Object Oriented Programming). Soporte fundamental en objetos. Un objeto es una extensión de un Tipo Abstracto de Datos (TAD). Un TAD es un tipo definido por el usuario que encapsula un conjunto de datos y las operaciones sobre estos datos.

Abstraccion.

Proceso mental en el que se evitan los detalles para centrarse en aspectos mas genéricos, para facilitar la comprensión.

Objeto.

- Abstraccion.
- Hrencia.
- Poliformismo.

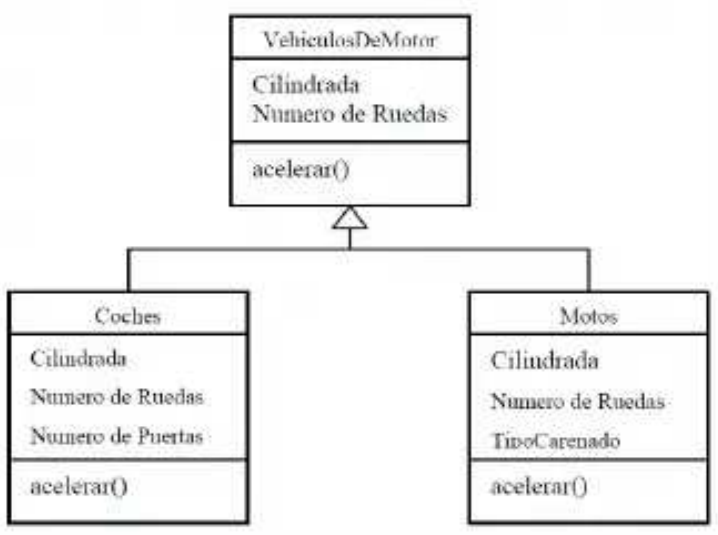
Ventajas de la orientación a objetos.

- Mantebilidad. Programas fáciles de leer e interpretar.
- Modificabilidad. Facilidad para modificar los programas.
- Reusabilidad. Los objetos se pueden utilizar numerosas veces y en distintos proyectos.
- Fiabilidad. Programas basados en el uso de objetos ya definidos y ampliamente testeados.

Conceptos básicos de la orientación a objetos.

Se basa en conceptos como:

- Clase. Es una descripción de un conjunto de objetos similares. Contiene atributos y operaciones sobre estos atributos.
- Objeto.Es una cosa, generalmente extraida del vocabulario del espacio del problema o solución. Todo objeto esta conformado por:
  - Nombre. Con el que se le puede identificar.
  - Estado. Generalmenta hay algunos datos asociados a el.
  - Comportamiento. Lo que se puede realizar al objeto y lo que puede hacer el objeto a otros objetos.
- Atributo. Es una característica concreta de una clase.
- Metodo. Es una operación concreta de una clase.
- Instancia. Es una mainfestacion concreta de una clase (Ocurrencia). Es un objeto con valores concretos.
- Herencia.Mecanismo por el cual se puede crear una nueva clase a partir de una existente. La clase hereda las características de otra clase y puede añadírsele funcionalidad o modificar la que tiene.



- Poliformismo. Posibilidad de que dos métodos implementen distintas acciones, aun teniendo el mismo nombre, dependiendo el objeto que lo ejecuta o de los parámetros que recibe.

Introduccion al lenguaje unificado de modelado (UML).

UML es un lenguaje unificado para escribir planos de software. Es una parte de un método de desarrollo de software. Es independiente del proceso aunque es dirigido por los Casos de Uso. Esta centrado en la arquitectura es interactivo e incremental.

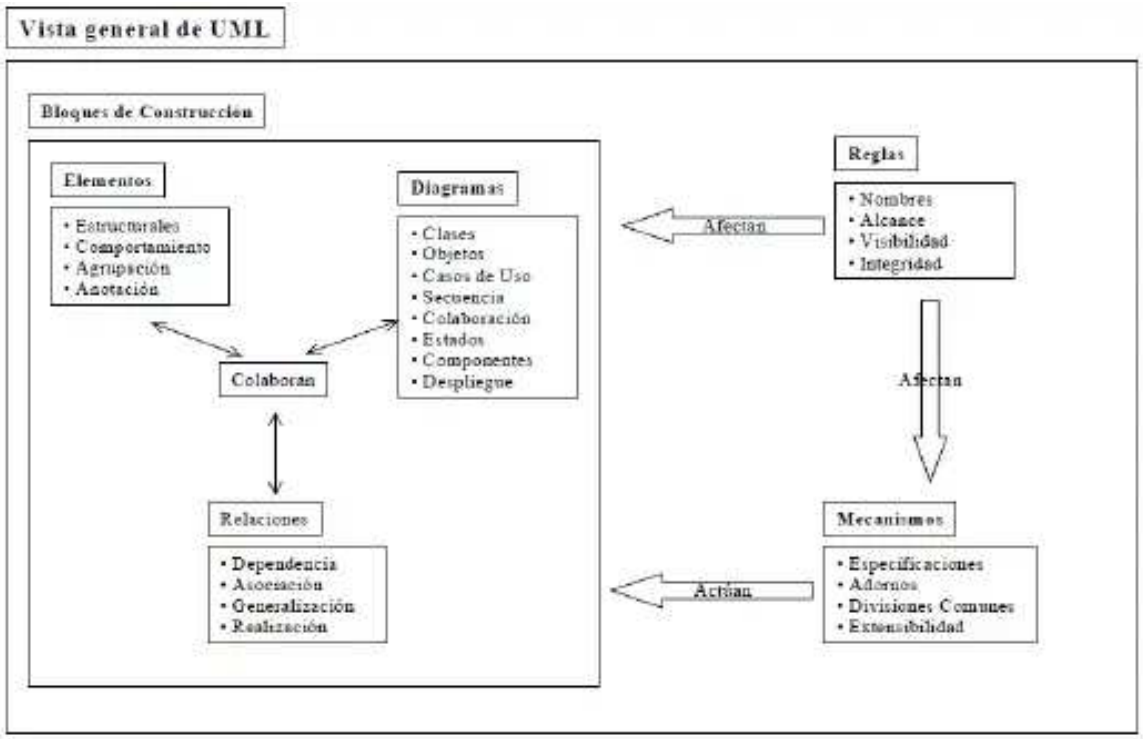
UML proporciona un vocabulario y las reglas para utilizarlo. Es un lenguaje de modelado que se utiliza para la representación conceptual y física de un Sistema.

UML nos ayuda a visualizar e interpretar grandes sistemas mediante graficos y texto. UML sirve para especificar modelos concretos y completos.

UML.

- Modela actividades de planificación de proyectos y de sus versiones.
- Expresa requisitos y pruebas sobre el sistema.
- Representa todos los detalles de la arquitectura.
- Documentación valida durante todo el ciclo del proyecto.

Vista general de UML.



UML se contituye de tres elementos básicos:

1. Bloques de construcción.
    a. Elementos. Abstracciones de primer nivel.
        i. Elementos Estructurales.
        ii. Elementos de Comportamiento.
        iii. Elementos de Agrupacion.
        iv. Elementos de Anotacion.
    b. Relaciones. Unen a los elementos entre si.
        i. Relacion de Dependencia.
        ii. Realacion de Asociacion.
        iii. Relacion de Generalizacion.
        iv. Relacion de Realizacion.
    c. Diagramas. Agrupaciones de elementos.
2. Reglas.
3. Mecanismos comunes.

UML proporciona un conjunto de reglas que dictan las pautas para realizar asociaciones entre objetos, son reglas semanticas que afectan los nombres, al alcance de dichos nombres, a la visibilidad de estos nombres, a la integridad de los elementos, en general a la vista dinámica del sistema.

UML propocina una serie de mecanismos comunes:

- Especificaciones. Proporcionan la explicación textual de sintaxis y semántica de los bloques de construcción.
- Adornos. Para conferir a los modelos de mas semántica.

- Divisiones Comunes. Permiten que los modelos se dividan al menos en un par de formas diferentes para facilitar la comprensión desde distintos puntos de vista.
    - Division Clase-Objeto.
    - Division Interfaz/Implementacion.
- Extensibilidad. Sirven para evitar posibles problemas que puedan surgir debido a la necesidad de poder representar ciertos matices (estereotipos).

Bloques de construcción de UML.

Elementos Estructurales.

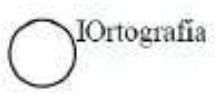
Son las partes estaticas del modelo y representan cosas que son conceptuales o materiales.

Clases.

Es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Una clase implementa una o mas interfaces. Graficamente se representa por un rectángulo que incluye su nombre, atributos y operaciones.

Interfaz.

Es una colección de operaciones que especifican un servicio de una determinada clase o componente. Describe un conjunto de especificaciones de operaciones, pero nunca su implementación.



Colaboracion.

Define una interaccion y es una sociedad de roles y otros elementos que colaboran para proporcionar un comportamiento. Una misma clase puede participar en diferentes colaboraciones. Representan la implementación de patrones que forman un sistema.

Casos de Uso.

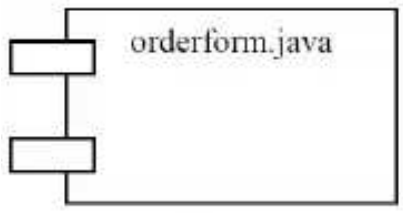
Un caso de uso es la descripción de un conjunto de acciones que un sistema ejecuta y produece un determinado resultado para un actor en particular. Se utiliza para Organizar los aspectos del comportamiento de un modelo. Un caso de uso es realizado por una colaboración.

Clase Activa.

Es una clase cuyos objetos tienen uno o más procesos o hilos de ejecución por lo tanto pueden dar lugar actividades de control.

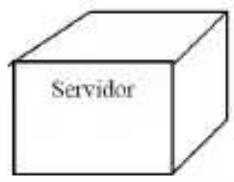
Componentes.

Es una parte física y remplazable del sistema. Se conforma de un conjunto de interfaces y proporciona la implementación de dicho conjunto. Representa típicamente el empaquetamiento físico de diferentes elementos lógicos como clases, interfaces y colaboraciones.



Nodos.

Es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional. Un conjunto de componentes puede residir en un nodo.

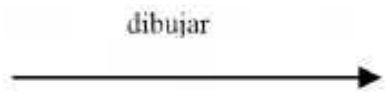


Elementos de Comportamiento.

Son las partes dinamicas de un modelo. Se podría decir que son los verbos del modelo, representan el comportamiento en el tiempo y espacio,

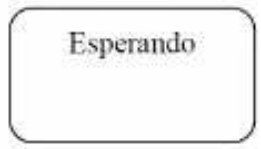
Interaccion.

Comprende un conjunto de mensajes intercambiados entre un conjunto de objetos, dentro de un contexto particular para conseguir un propósito específico. Una interaccion involucra muchos elementos como, mensajes, secuencias de acción, enlaces.



Máquina de Estados.

Es un comportamiento que especifica las secuencias de estados por las que van pasando los objetos o las interacciones durante su vida en respuesta a eventos. Una máquina de estados involucra otros elementos como son estados, transiciones, eventos y actividades.



Elementos de Agrupacion.

Forman la parte organizativa de UML. Su elemento principal es el paquete que es un mecanismo de propósito general para organizar elementos en grupos.

Elementos de Anotacion.

Son las partes explicativas de los modelos de UML. Son comentarios que se pueden aplicar para describir, clasificar y hacer observaciones sobre cualquier elemento del modelo. E; tipo principal de anotación es la nota.

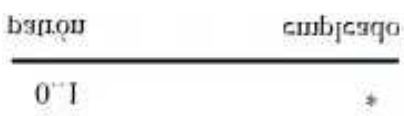
Relaciones.

Dependencia.

Relacion semántica entre dos elementos en la cual un cambio a un elemento (independiente) puede afectar la semántica de otro elemento (dependiente).

Asociacion.

Relacion estructural que describe un conjunto de enlaces (conexiones entre objetos).



Generalizacion.

Relacion de especialización / generalización en la cual el elemento especializado (hijo) pude sustituir a los objetos del elemento general (padre).

Realizacion.

Relacion semántica entre clasificadores, donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá. Se encuentra este tipo de relación entre interfaces y las clases y componentes que las realizan y entre casos de uso y las colaboraciones que los realizan.

Diagramas.

Se utilizan para representar diferentes perspectivas de un sistema. UML proporciona un amplio conjunto de diagramas que normalmente se usan en pequeños subconjuntos para poder representar las cinco vistas principales de la arquitectura de un sistema.

Diagrama de Clases.

Muestra un conjunto de clases, interfaces y colaboraciones, asi como sus relaciones. Cubren la vista de diseño estatico o la vista de procesos estatica (si incluyen clases activas).

Diagrama de Objetos.

Muestran un conjunto de objetos y sus relaciones. Fotos instantaneas de los diagramas de clase. Cubren vista de diseño estatica o procesos estatica.

Diagrama de Casos de Uso.

Muestran un conjunto de casos de uso y actores (tipo especial de clases) y sus relaciones. Cubren la vists estatica de los casos de uso y son especialmente importantes para el modelado y organización del comportamiento.

Diagramas de Secuencia y Colaboracion.

Son un tipo de diagramas de interaccion. Constan de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar unos objetos a otros. Cubren la vista dinámica del sistema.

- Diagramas de secuencia. Enfatizan el ordenamiento temporal de los mensajes.
- Diagramas de colaboración. Muestran la organización estructural de los objetos que envían y reciben mensajes.

Diagrama de Estados.

Muestran una máquina de estados compuesta por estados, transiciones, eventos y actividades. Cubren la vista dinámica de un sistema y son muy importantes a la hora de modelar el comportamiento de una interfaz, clase o colaboración.

Diagrama de Actividades.

Tipo especial de diagrama de estados. Muestra el flujo de las actividades dentro de un sistema. Cubren la parte dinámica de un sistema. Modelan el funcionamiento de un sistema resaltando el flujo de control entre los objetos.

Diagrama de Componentes.

Muestra la organización y las dependencias entre un conjunto de componentes. Cubren la vista de implementación estatica y se relacionan con los diagramas de clases que un componente sule tener una o mas clases, interfaces o colaboraciones.

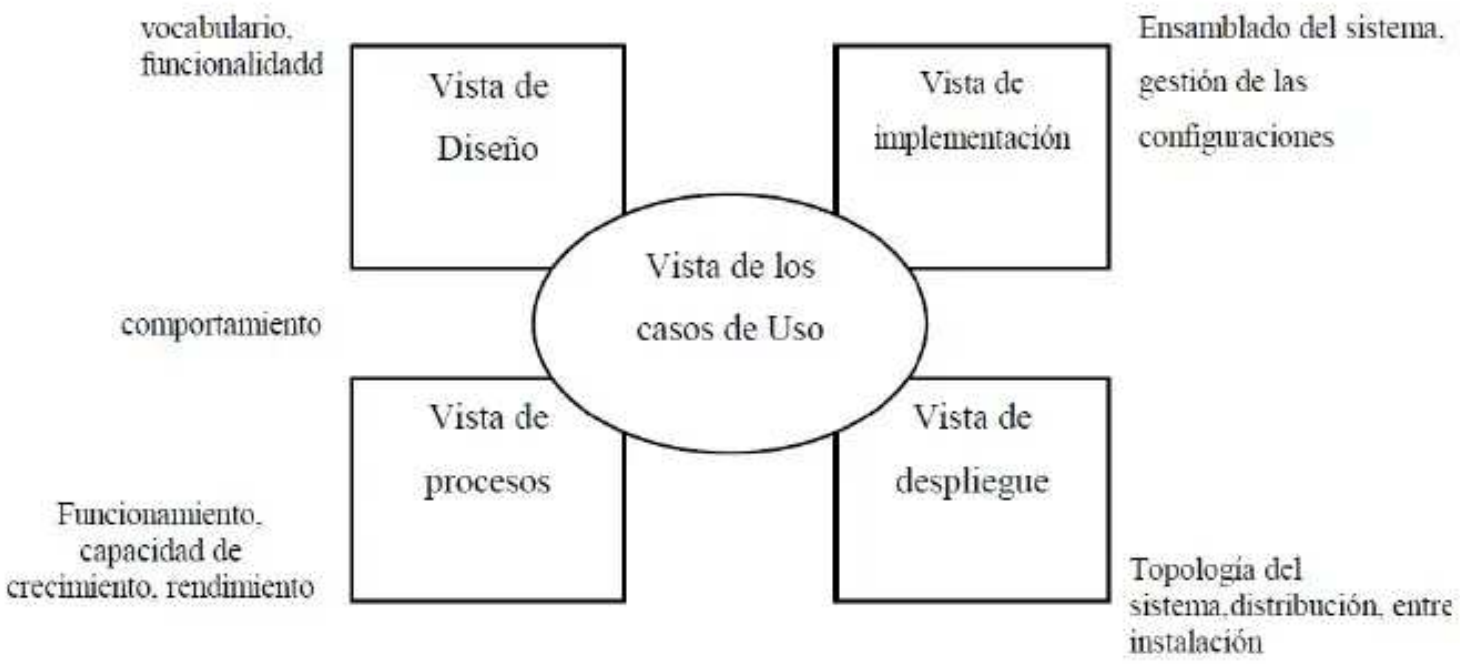
Diagrama de Despliegue.

Representan la configuración de los nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos. Muestran la vista de despliegue estatica de una arquitectura. Se relacionan con los componentes.

Arquitectura.

La arquitectura es el conjunto de decisiones significativas sobre:

- La organización del sistema.
- Selección de elementos estructurales y sus interfaces a través de los cuales se constituye el sistema.
- El Comportamiento. Como se especifica la colaboración entre componentes.
- Composición de los elementos estructurales y del comportamiento en subsistemas progresivamente más grandes.
- El estilo arquitectónico que guía esta organización: elementos estáticos y dinámicos y sus interfaces, sus colaboraciones y su composición.



La arquitectura no debe cedntrase únicamente en la estructura y comportamiento, debe de abarcar temas como:

- Uso.
- Funcionalidad.
- Rendimiento.
- Capacidad de adaptación.

- Reutilización.
- Capacidad de comprensión.
- Restricciones.
- Compromisos entre alternativas.
- Estética.

Vista de Casos de Uso.

Comprende la descripción del comportamiento del sistema tal y como es percibida por los usuarios finales, analistas y encargados de las pruebas

- Estáticos.
  - Diagramas de Casos de Uso
- Dinámicos.
  - Diagramas de interacción.
  - Diagramas de estados.
  - Diagramas de actividades.

Vista de Diseño.

Comprende las clases, interfaces y colaboraciones que forman el vocabulario del problema y la solución. Soporta principalmente los requisitos funcionales del sistema. Los elementos estáticos se representan por diagramas de clase y objetos, los dinámicos con diagramas de interacción, estados y actividades.

- Estáticos.
  - Diagramas de Clases.
  - Diagramas de Objetos.
- Dinámicos.
  - Diagramas de interacción.
  - Diagramas de estados.
  - Diagramas de actividades.

Vista de Procesos.

Comprende los hilos y procesos que forman los mecanismos de sincronización y concurrencia del sistema. Cubren el funcionamiento, capacidad de crecimiento y rendimiento del sistema.

- Estáticos.
  - Diagramas de Clases (Incluyendo Clases Activas)
  - Diagramas de Objetos.
- Dinámicos.
  - Diagramas de interacción.
  - Diagramas de estados.
  - Diagramas de actividades.

Vista de Implementación.

Comprende los componentes y archivos que un sistema utiliza para ensamblar y hacer disponible el sistema físico. Se ocupa principalmente de la gestión de configuraciones de las distintas versiones del sistema.

- Estáticos.
    - Diagramas de Componentes.
- Dinámicos.
    - Diagramas de interacción.
    - Diagramas de estados.
    - Diagramas de actividades.

Vista de Despliegue.

Contiene los nodos que forman la topología hadware sobre la que se ejecuta el sistema. Se preocupa principalmente de la distribución, entrega e instalación de las partes que constituyen el sistema.

- Estáticos.
    - Diagramas de Despliegue.
- Dinámicos.
    - Diagramas de interacción.
    - Diagramas de estados.
    - Diagramas de actividades.

Ciclo de Vida.

Son todas las etapas por la que pasa un proyecto, desde la concepción de la idea, pasando por el análisis, desarrollo, implantación y mantenimiento.

UML debe de considerar un proceso que fuese:

- Dirigido por Casos de Uso.
- Centrado en la Arquitectura.
- Iterativo e Incremental.

Modelo Estructural.

Es la parte de UML que se encarga de identificar todas las partes importantes de un sistema asi como sus interacciones. Para modelar las partes importantes de un sistema se utilizan las clases.

Representacion de las Clases en UML.

Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. En UML las clases se representan por rectángulos con distintos compartimientos para indicar los atributos y operaciones.

Una clase es identificada por un nombre que la distingue del resto, el nombre es una cadena de texto.

- Nombre simple. Solo el nombre de la clase.
- Nombre de camino. Nombre de la clase precedido por el nombre del paquete al que pertenece.

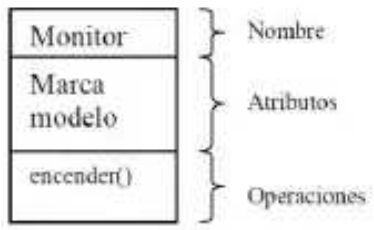


Figura 20. Representación de Clases

Atributo.

Es una propiedad de una clase identificada por un nombre. Describe un rango de valores que pueden tomar las instancias de la propiedad. Representan las propiedades de los objetos.

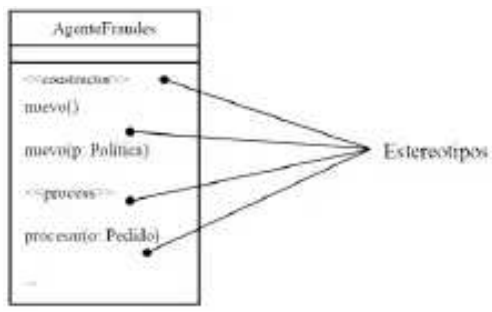
Operaciones.

Es una implementación de un servicio que puede ser requerido a cualquier objeto de la clase para que muestre su comportamiento. Representa algo que el objeto puede hacer.

Cuando se dibuja una clase no hay que mostrar todos los atributos, ni todas sus operaciones, solo se deben de mostrar los subconjuntos de estos que sean más relevantes utilizando estereotipos.

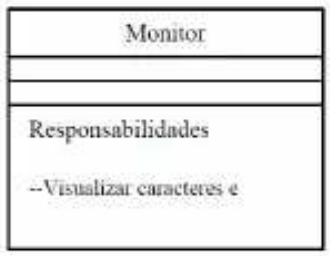
Estereotipos.

Son bloques básicos de construcción a los que se han añadido etiquetas, iconos o texto explicativo para proporcionar más semántica a estos bloques, consiguiendo diagramas más explicativos.



Responsabilidades.

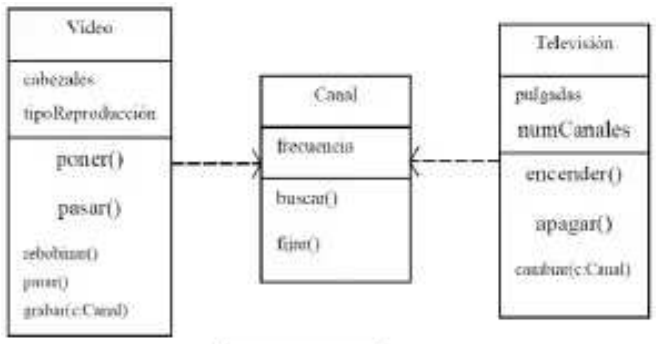
Es un contrato u obligación de una clase. Los atributos y las operaciones de una clase son características que permiten llevar a cabo sus responsabilidades. Para representar las responsabilidades se utiliza un cuarto comportamiento en el bloque de construcción de la clase, en el cual se especifican mediante frases cortas de texto libre las responsabilidades que debe de cumplir una clase.



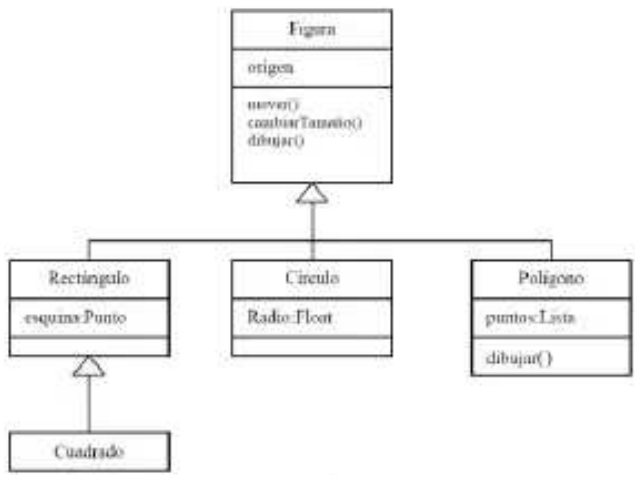
Relaciones.

Son la manera de representar las interacciones entre las clases. Existe tres tipos de relaciones:

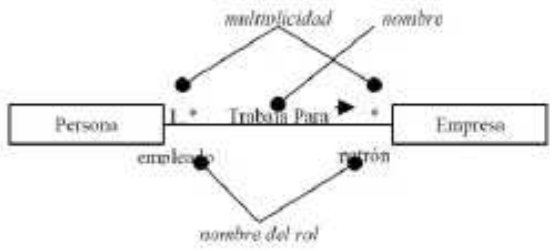
- Dependencia. Relación de uso entre dos elementos de manera que el cambio en la especificación del elemento independiente puede afectar al otro elemento implicado en la relación.



- Generalización. Relación entre un elemento general (Superclase o Padre) y un caso más específico de ese elemento (Subclase o Hijo).

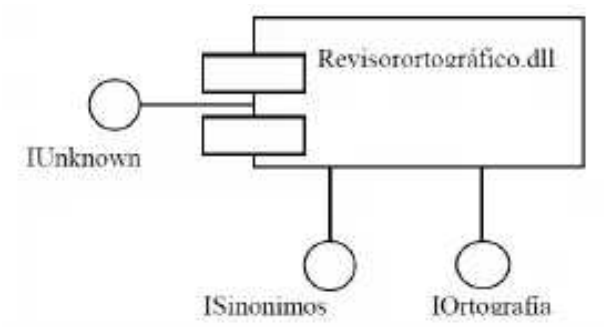


- Asociación. Especifica que los objetos de un elemento están conectados con los objetos de otro. Se utilizan para representar relaciones estructurales. Usa cuatro adornos para facilitar su comprensión:
  - Nombre. Describe la naturaleza de la relación.
  - Rol. Rol especifico que juega en la relación.
  - Multiplicidad. Cuantos objetos se pueden conectar a través de una instancia de la asociación.
  - Agregación. Asociación normal entre dos clases, representa una relación estructural entre iguales.
  - Composición. Variación de la agregación simple que añade una semántica importante.

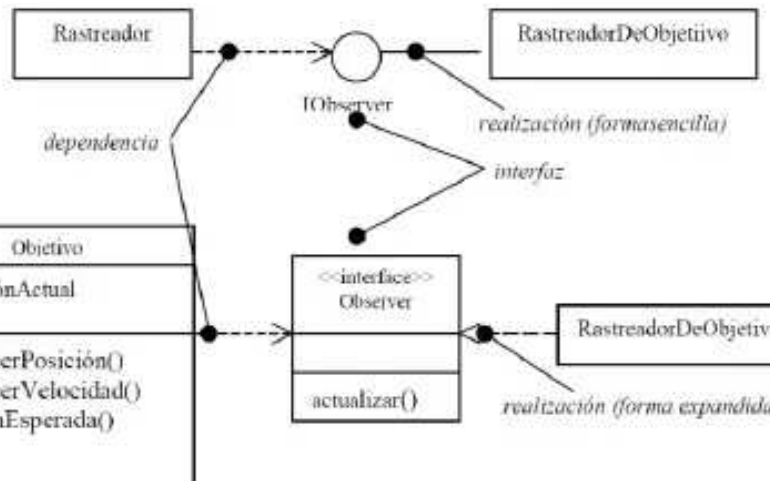


Interfaces.

Colección de operaciones que sirven para especificar que una clase o componente ofrece.

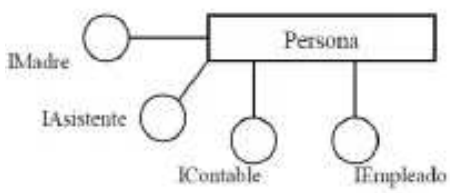


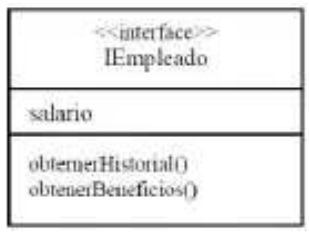
Una interfaz puede participar en relaciones de generalización, asociación y dependencia. Especifica un contrato para una clase o componente sin dictar su implementación.



Roles.

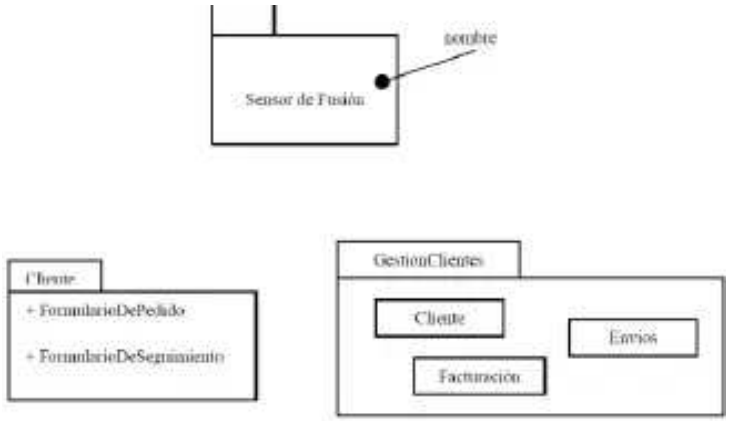
Denota el comportamiento de una entidad en un contexto en particular. Es la cara que una abstracción presenta al mundo.





Paquetes.

Mecanismo de propósito general para organizar elementos de modelado en grupos. También se pueden utilizar para las diferentes vistas de la arquitectura de un sistema.



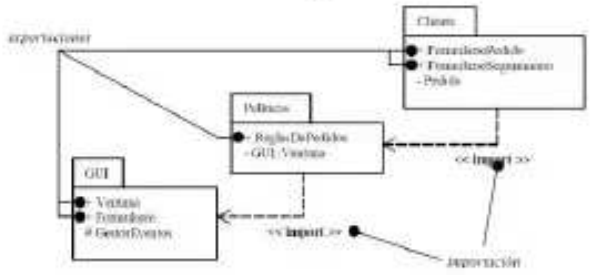
Términos y Conceptos.

Abstracciones que organizan un modelo. Ayudan a organizar los elementos en los modelos con el fin de poder comprenderlos más fácilmente.

Elementos Contenidos.

Un paquete puede contener otros elementos, incluyendo clases, interfaces, componentes, nodos, colaboraciones, casos de uso, diagramas, e incluso otros paquetes.

Para poder utilizar los elementos que se encuentran en otros paquetes se ha de importar el paquete que nos interesa.



Instancias.

Es una manifestación concreta de una abstracción a la que se le puede aplicar un conjunto de operaciones y que puede tener un estado que almacena los efectos de la operación. En UML una instancia se representa subrayando su nombre.

Operaciones.

Las operaciones que se pueden ejecutar sobre un objeto se declaran en la abstracción del objeto (en la clase).

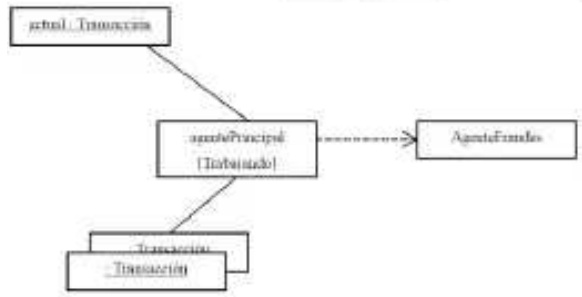
Estado.

Un objeto tiene estado, incluye todas las propiedades (normalmente estáticas) del objeto más los valores actuales (normalmente dinámicos) de estas propiedades.

Modelado de instancias concretas.

Una de las cosas para las que se emplean los objetos es para modelar instancias concretas del mundo real, para modelar instancias concretas:

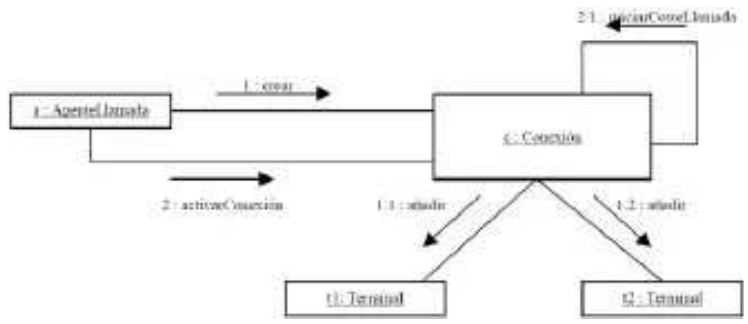
- Hay que identificar aquellas instancias necesarias y suficientes para visualizar, especificar, construir o documentar el problema.
- Hay que representar eso objetos en UML como instancias.
- Hay que mostrar el estereotipo, los valores, etiquetado y los atributos (con sus valores) de cada instancia necesarios y suficientes para modelar el problema.
- Hay que representar las instancias y sus relaciones en un diagrama de objetos u otro diagrama apropiado al tipo de instancia (nodo, componente…).



Modelado de instancias prototípicas.

La utilización más importante que se hace de las instancias es modelar las interacciones dinámicas entre los objetos. Para modelar instancias prototípicas:

- Hay que representar los objetos en UML como instancias.
- Hay que mostrar las propiedades de cada instancia necesarias y suficientes para modelar el problema.
- Hay que representar las instancias y sus relaciones en un diagrama de interacción o de actividades.



Diagramas de Clases y Objetos.

Diagramas de Clase.

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema. Son la base para dos diagramas relacionados, los diagramas de componentes y los de despliegue. Pueden contener notas y restricciones. También pueden contener paquetes o subsistemas. A veces se colocarán instancias en los diagramas de clases, especialmente cuando se quiere mostrar el tipo (posiblemente dinámico) de una instancia.

Los diagramas de componentes son muy parecidos a los diagramas de clase, simplemente que se muestran componentes y nodos respectivamente en vez de clases.

Usos comunes.

Los diagramas de clase se utilizan para modelar la vista de diseño estática de un sistema. Esta vista soporta principalmente los requisitos funcionales de un sistema, los servicios que el sistema debe de proporcionar a los usuarios finales.

Cuando se modela la vista de diseño estática de un sistema, normalmente se utilizan los diagramas de clases de una de estas formas:

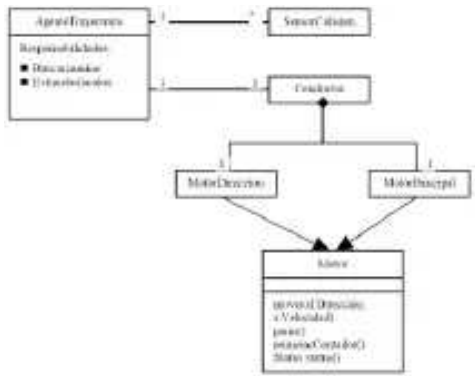
1. Para modelar el vocabulario de un sistema. Implica tomar decisiones sobre que abstracciones son parte del sistema en consideración y cuales caen fuera de sus límites.
2. Para modelar colaboraciones simples. Una colaboración es una sociedad de clases, interfaces y otros elementos que colaboran para proporcionar un comportamiento cooperativo mayor que la suma de todos sus elementos. Los diagramas de clases se emplean para visualizar y especificar este conjunto de clases.
3. Para modelar un esquema básico de una Base de Datos. Se puede pensar en un esquema como un plano para el diseño conceptual de una base de datos. Se pueden modelar esquemas para estas bases de datos mediante diagramas de clases.

Modelado de colaboraciones simples.

Ninguna clase se encuentra aislada. Cada una trabaja en colaboración con otras para llevar a cabo alguna semántica mayor que la asociada a cada clase individual. Esta colaboración se representa con el diagrama de clases.

Para modelar una colaboración:

- Hay que identificar los mecanismos (funciones o comportamientos) que se quieren modelar.
- Para cada mecanismo hay que identificar las clases, interfaces y otras colaboraciones que participan en esta colaboración y las relaciones entre estos elementos.
- Hay que usar escenarios para recorrer las interacciones entre estos elementos. Durante el recorrido, se descubrirán partes del modelo que faltaban y partes que eran semánticamente incorrectas.
- Hay 1que asegurarse de rellenar estos elementos con su contenido.



Modelado de un esquema lógico de Base de Datos.

Los diagramas de clase son un superconjunto de los diagramas entidad-relación, permitiendo el modelado del comportamiento. Para modelar un esquema de Base de Datos:

- Hay que identificar aquellas clases del modelo cuyo estado debe de trascender en el tiempo de vida de las aplicaciones.
- Hay que crear un diagrama de clases que contenga las clases que hemos identificado y marcarlas como persistentes (con un valor etiquetado estándar).
- Hay que expandir los detalles estructurales de estas clases, especificar los detalles de sus atributos y centrar la atención en las asociaciones que estructuran estas clases y en sus cardinalidades.
- Hay que considerar el comportamiento de las clases persistentes, expandiendo las operaciones que sean importantes para el acceso a los datos y la integridad de los mismos.
- Donde sea posible, hay que usar herramientas que ayuden a transformar un diseño lógico en un diseño físico.

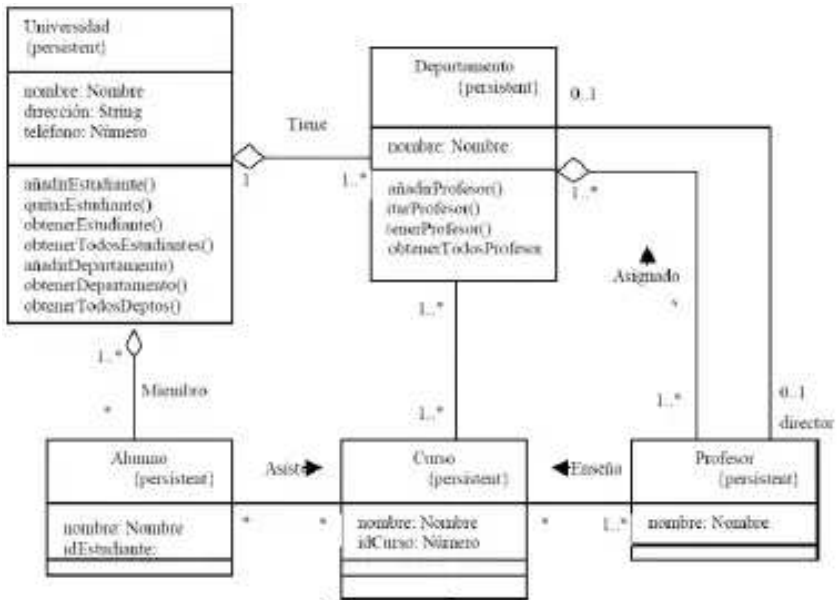
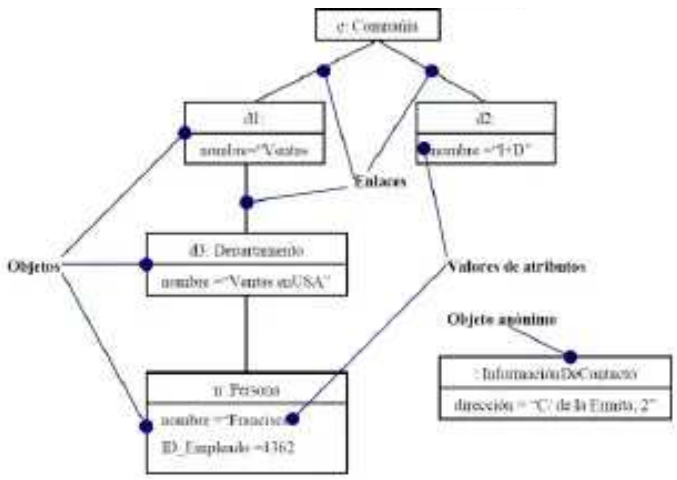


Diagrama de Objetos.

Contiene un conjunto de instancias de los elementos encontrados en el diagrama de clases. Expresa la parte estática de una interacción, consistiendo en los objetos que colaboran, pero sin ninguno de los mensajes enviados entre ellos.



Un diagrama de objetos es simplemente un diagrama que representa un conjunto de objetos y sus relaciones en un momento concreto. Contiene objetos y enlaces. Se puede tratar como una instancia de un diagrama de clases o la parte estática de un diagrama de interacción.

Usos Comunes.

Se emplean para modelar la vista de diseño estática o la vista de procesos estática del sistema, desde la perspectiva de instancias reales o prototípicas. Sustenta principalmente los requisitos funcionales de un sistema (los servicios que debe de proporcionar el sistema a los usuarios finales).

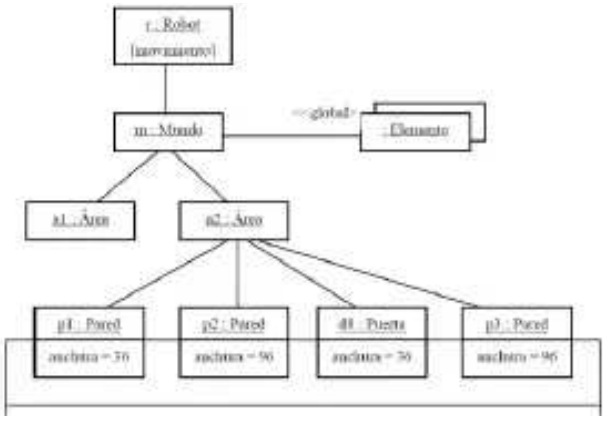
Los diagramas de objetos sirven para visualizar, especificar, construir y documentar la existencia de ciertas instancias en el sistema, junto a sus relaciones.

Modelado de estructuras de objetos.

Los diagramas de objetos se pueden utilizar para visualizar, especificar, construir y documentar la estructura de esos objetos. Son especialmente útiles para modelar estructura de datos complejas. Un diagrama de objetos muestra un conjunto de objetos relacionados entre si en un momento dado.

Para modelar diagramas de objetos:

- Hay que identificar el mecanismo (función o comportamiento) que se desea modelar.
- Para cada mecanismo hay que identificar las clases, interfaces y otros elementos que participan en esta colaboración y la relación entre estos elementos.
- Hay que considerar un escenario donde intervenga este mecanismo.
- Hay que mostrar el estado y los atributos de cada uno de esos objetos, si son necesarios para comprender el escenario.
- Analógicamente, hay que mostrar los enlaces de esos objetos, que representarían instancias de las asociaciones entre ellos.



Modelado del Comportamiento.

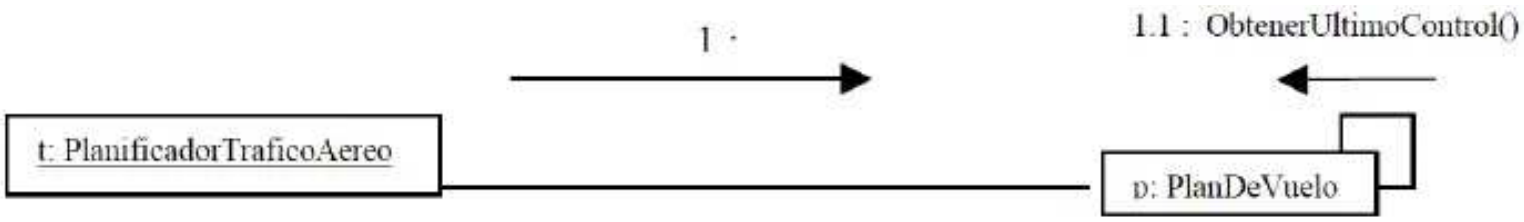
Interacciones.

En cualquier sistema los objetos interactúan entre si pasándose mensajes. Una interacción es comportamientos que incluye un conjunto de mensajes intercambiados por un conjunto de objetos dentro de un contexto para lograr un propósito.

Las interacciones modelan los aspectos dinámicos de un sistema. Las interacciones incluyen mensajes enviados entre objetos.

Las interacciones se utilizan para modelar el flujo de control dentro de una operación, una clase, un componente, un caso de uso o el propio sistema.

Una interaccion es un comportamiento que comprende un conjunto de mensajes intercambiados entre un conjunto de objetos dentro de un contexto, para lograr un propósito.



Un mensaje es la especificación de la comunicación entre objetos que transmite información, con la expectativa de que se desencadenara una actividad.

Contexto.

Una interacción puede aparecer siempre que unos objetos estén enlazados a otros. También aparecerán interacciones entre objetos en la implementación de una operación. La s interacciones aparecerán en el contexto de una clase para visualizar, especificar, constituir y documentar la semántica de una clase.

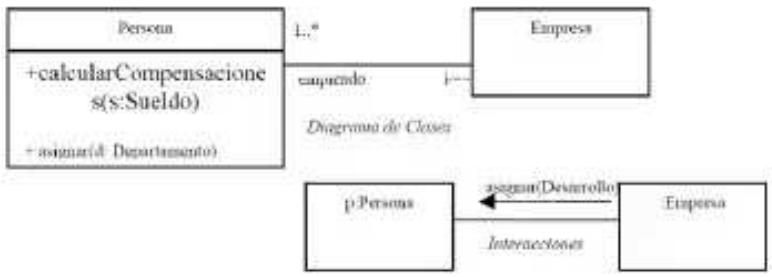
Objetos y Roles.

Los objetos que participan en una interaccion pueden ser elementos concretos o prototípicos.

Un diagrama de objetos se pude ver como una representación del aspecto estatico de una interaccion, que configura el escenario para la interaccion al especificar todos los objetos que colaboran entre si.

Enlaces.

Un enlace es una conexión semántica entre objetos, es una instancia de una asociación.



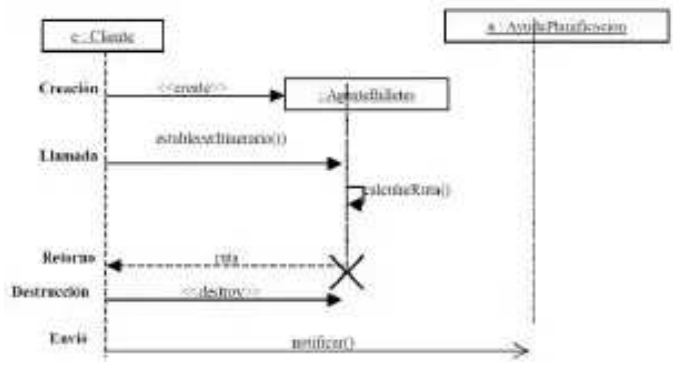
Mensajes.

Un mensaje es la especificación de una cominicacion entre objetos que transmite información, con la expectativa de que se desencadenara una actividad.

Cuando se pasa un mensaje, la acción resultante es una instrucción ejecutable que constituye una abstracción de un procedimiento computacional. Una acción puede producir un cambio en el estado.

En UML se modelan varios tipos de acciones:

- Llamada. Invoca una operación sobre un objeto.
- Retorno. Devuelve un valor al invocador.
- Envio. Envia una señal al objeto.
- Creacion. Crea un objeto.
- Destruccion. Destruye un objeto.



Modelado de un flujo de control.

Las interacciones se utilizan con el propósito de modelar el flujo de control que caracteriza el comportamiento de un sistema, incluyendo casos de uso, patrones, mecanismos y frameworks, o el comportamiento de una clase o una operación individual.

Cuando se modela una interacción, lo que se hace esencialmente es construir una representación gráfica de las acciones que tienen lugar entre un conjunto de objetos.

Para modelar un flujo de control:

- Establecer el contexto de la interacción, sistema, clase, operación individual.
- Establecer el escenario de la interacción, propiedades iniciales, atributos, estado y rol.
- Identificar los enlaces que conectan a los objetos que sean relevantes para los trayectos de comunicación.
- Especificar los mensajes que pasan de un objeto a otro, mediante una organización temporal.
- Adornar cada objeto con su estado y rol.

Ingresar imagen RGM.

Casos de Uso.

Especifica el comportamiento de un sistema o una parte del mismo. Es una descripción de un conjunto de secuencias. Se emplean para capturar el comportamiento deseado del sistema en desarrollo, sin tener que especificar como se implementa ese comportamiento. Proporcionan un medio para que los desarrolladores, usuarios y expertos de dominio lleguen a una comprensión común del sistema. Andan a

verificar y validar la arquitectura mientras evoluciona en el desarrollo.
Un factor clave al definir casos de uso es que no se especifica como se implementan.

Es una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta el sistema para poder producir un resultado observable.