Curso de Modelado de Sistemas

CENDITEL, Mayo 2008

Licencia de Uso

Tabla de Contenido

Modelado de Sistemas

El modelado de sistemas es un proceso mediante el cual se representa o abstrae, a través de un modelo, las características o cualidades que más nos interesan de un objeto (cosa, fenómeno o sistema). Es importante mencionar que un mismo sistema puede ser modelado de diferentes formas dependiendo de la perspectiva del modelador.

El modelado de un sistema se puede dividir en varios tipos de modelado. A continuación mencionaremos los tipos de modelado más utilizados en la Ingeniería de Software:

- Modelado de procesos.
- Modelado funcional.
- Modelado de comportamiento.
- Modelado estructural.
- Modelado de implementación.

A continuación se indicaran en las siguientes secciones algunos de los lenguajes de modelado más utilizados así como algunas herramientas libres para modelado de sistemas. Seguidamente, se especificarán los tipos de modelado indicados con ejemplos para cada uno de ellos.

1. Lenguajes de Modelado

Los lenguajes de modelado permiten representar y comunicar el conocimiento que se tiene sobre un determinado objeto. Entre los lenguajes de modelado más recientes se encuentran: SysML (*System Modeling Language*), BPML (*Business Process Modeling Language*), WebML (*Web Modeling Language*) y el UML (*Unified Modeling Language*).

- UML

  Es un lenguaje gráfico que permite visualizar, especificar y documentar un sistema de software. Este lenguaje es un estándar administrado por la OMC (*Object Management Group*). Las versiones más recientes de este lenguaje son: UML 1.4, UML 1.5 y UML 2.0 (las herramientas libres existentes no tienen todos los diagramas que constituyen la versión 2.0 de UML). En el caso del curso se utilizará, en la medida de lo posible, la versión 2.0 del UML para el modelado de sistemas.

  El lenguaje UML nos permite modelar una aplicación desde diferentes perspectivas, estas son:

- Perspectiva funcional (Casos de uso de la aplicación).
- Perspectiva estructural (Diagramas de clases).
- Perspectiva de comportamiento (Diagramas de secuencia y de comunicación, etc.).
- Perspectiva de implementación (Diagramas de componentes y de despliegue).

En el UML 2.0 los tipos de diagramas se clasifican en dos grandes grupos:

- Diagramas para el modelado estructural:
  - Diagramas de clase.
  - Diagramas de objetos.
  - Diagramas de estructura compuesta.
  - Diagramas de componentes.
  - Diagramas de despliegue.
  - Diagramas de paquetes.
- Diagramas para el modelado de comportamiento:
  - Diagramas de casos de uso.
  - Diagramas de interacción (de secuencia, de comunicación, de temporización).
  - Diagramas de máquinas de estado.
  - Diagramas de actividad.

El lenguaje UML permite modelar no solo aplicaciones sino también los procesos que se automatizan en una aplicación. En este sentido, UML ofrece los diagramas de procesos y de actividad.

2. Algunas Herramientas Libres para el Modelado de Sistemas

- ArgoUML.
- Umbrello.
- Dia.
- CASEUML.

- BOUML.

3. Modelado de una Aplicación (Caso teórico-práctico)

A continuación se presenta una descripción de los modelos más utilizados para representar una aplicación de software.

3.1 Modelado de Procesos

El modelado de procesos se define como una disciplina que permite definir y especificar las prácticas de una organización. Es una representación que capta la estructura y la dinámica de la organización, es decir, sus procesos, sus flujos de información y el almacenamiento de sus datos. La finalidad del modelado de procesos es describir cada proceso de una organización, especificando los datos (entrada y salida), las actividades, los roles y las reglas que regulan cada proceso.

El modelado de proceso constituye un aspecto importante para entender y reestructurar (en caso de ser necesario) las actividades que lleva a cabo una organización en función de alcanzar sus objetivos. De igual manera, el modelo de procesos representa una de las actividades (análisis del dominio de la aplicación) más importantes del proceso de desarrollo, en vista de que permite al grupo desarrollador adquirir mayor comprensión sobre los procesos que debe automatizaren la aplicación a desarrollar.

3.1.1 Conceptos Básicos del Modelado de Procesos

- **Proceso**

  El proceso se define como un conjunto estructurado de actividades que se ejecutan para alcanzar un objetivo específico. Los procesos son activados por eventos, tienen asociadas actividades, actores, reglas, entradas y salidas. Un proceso puede formar parte de un proceso mayor que lo abarque o, bien puede incluir otros procesos (subprocesos) que deban ser incluidos en su función.

- **Actores**

  Los procesos son llevados a cabo por actores. El actor puede ser una persona o una máquina capaz de llevar a cabo el flujo de actividades definido para un proceso.

- **Entrada**

  Son prerrequisitos (información, solicitud, documento, etc.) que se requieren para ejecutar un proceso y que son de alguna manera transformados por el proceso.

- **Salida**

Constituyen los productos (información, solicitud, documento, ect.) obtenidos en la ejecución de un proceso.

- **Reglas**

  Las reglas constituyen una colección de políticas y restricciones de un proceso. Un ejemplo de reglas de procesos sería:

  Regla del proceso de venta en una tienda:"Un cliente al que facturamos más de 10.000 Bs se clasifica como un cliente de tipo A. A los clientes de tipo A les aplicamos un descuento del 10% en pedidos superiores a 3.000 Bs".

- **Eventos**

  Los procesos son activados por la ocurrencia de eventos. Un evento es un suceso de duración muy breve que tiene lugar dentro o fuera del sistema. El evento marca el punto de arranque de un proceso y en algunos casos el evento constituye el dato de entrada al proceso.

- **Recursos**

  Son prerequisitos que se requieren parra llevar a cabo un proceso pero que no son transformados por este.

- **Actividades**

  Una actividad constituyen un conjunto de acciones estructuradas e interrelacionadas que se llevan a cabo como parte de un proceso. Por lo general, cada actividad representa un subproceso de un proceso más grande, el cual ya no se puede seguir dividiendo en otros subproceso.

- **Acciones**

  Una acción representa la unidad fundamental de especificación de comportamiento, la cual no puede ser descompuesta en otras acciones.

3.1.2 Diagramas para el Modelado de Proceso

Existen varios tipos de diagramas y lenguajes utilizados para el modelado de proceso, entre estos e encuentran:

- Diagrama de cadena de valor.
- Diagrama de jerarquía de procesos.

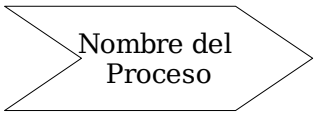
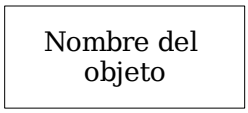
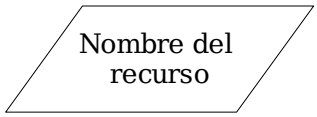
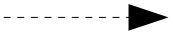
- Diagrama de relación entre procesos.
- Diagrama de procesos (caja negra).
- Diagramas de actividades.

En el caso específico del curso se estará utilizando el diagramas de jerarquía de procesos, el diagrama de proceso de caja negra, ambos del UML *Bussines,* y los diagramas de actividades del UML 2.0. A continuación pasaremos a describir cada uno de estos diagramas.

- **Diagramas del UML *Bussines***

  Este lenguaje es una extensión del UML, en el cual se plantean varios tipos de diagramas para el modelado de procesos. A continuación se presentan los símbolos más utilizados en UML *Business* para el modelado de procesos.

  Tabla Nº 1. Símbolos utilizados para representar diagramas de procesos según el UML*Bussines*

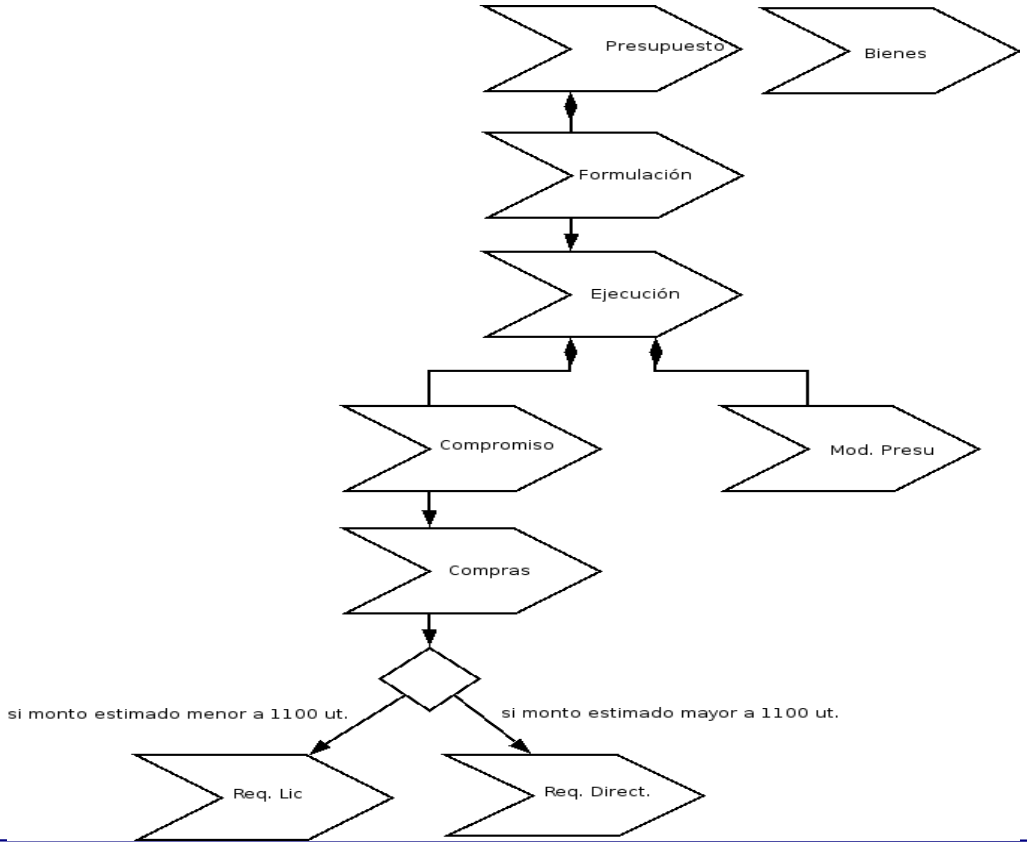
Nombre	***Símbolo***	***Utilidad***
Proceso		Se utilizada para representar un proceso.
Objeto		Se utiliza para representar los actores, las reglas, los objetivos, las entradas y salidas de un proceso.
Recurso		Se utilizada para representar cualquier tipo de información requerida en un proceso, pero que no es transformada por el proceso.
Flujo de control o secuencia		Se utiliza para indicar la secuencia de ejecución entre un conjunto de procesos.
Flujo de recursos		Se utiliza para indicar recursos que fluyen en un proceso o entre procesos.
Composición		Ase utiliza para representar la relación

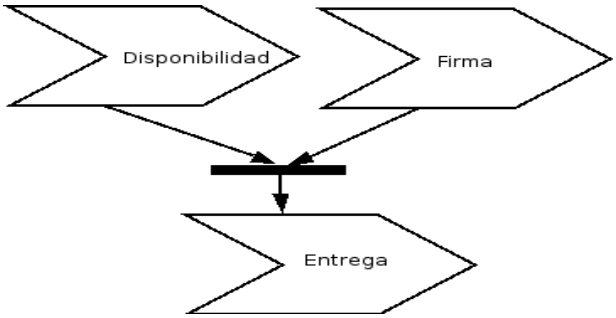
Nombre	*Símbolo*	*Utilidad*
		de composición entre procesos.
Decisión		Se utiliza para indicar bifurcaciones de decisiones entre procesos.
Barra de sincronización		Se utilizada para indicar la sincronización o concurrencia de procesos.

Diagrama de Jerarquía de Procesos

Este diagrama es utilizado para representar las relaciones de jerarquía y/o composición entre procesos.

Ejemplo: Diagramas de jerarquía de algunos procesos del SAID

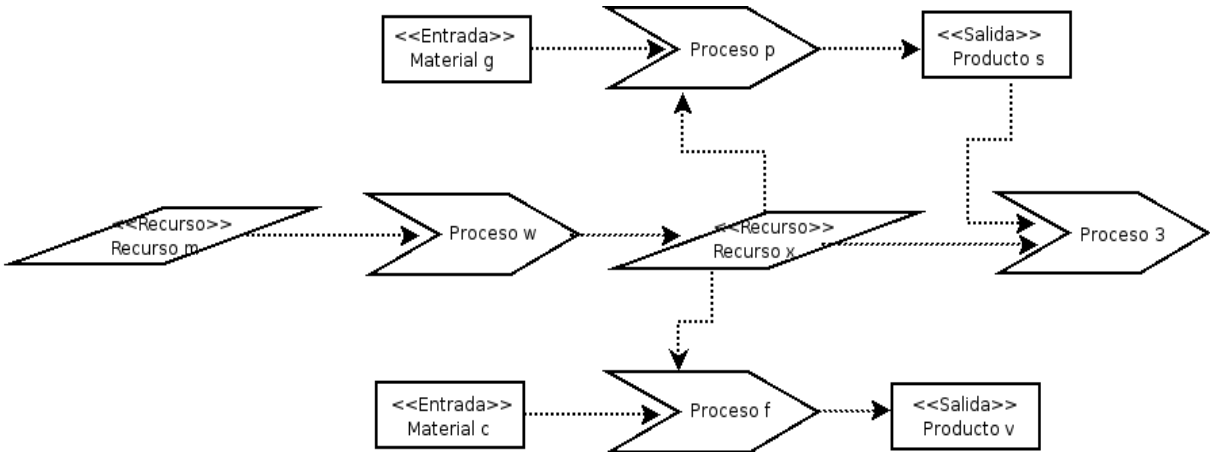




- **Diagrama de Relación entre Procesos**

  Este diagrama se utiliza para representar la relación entre procesos a través de los objetos de entrada y salida de cada proceso.

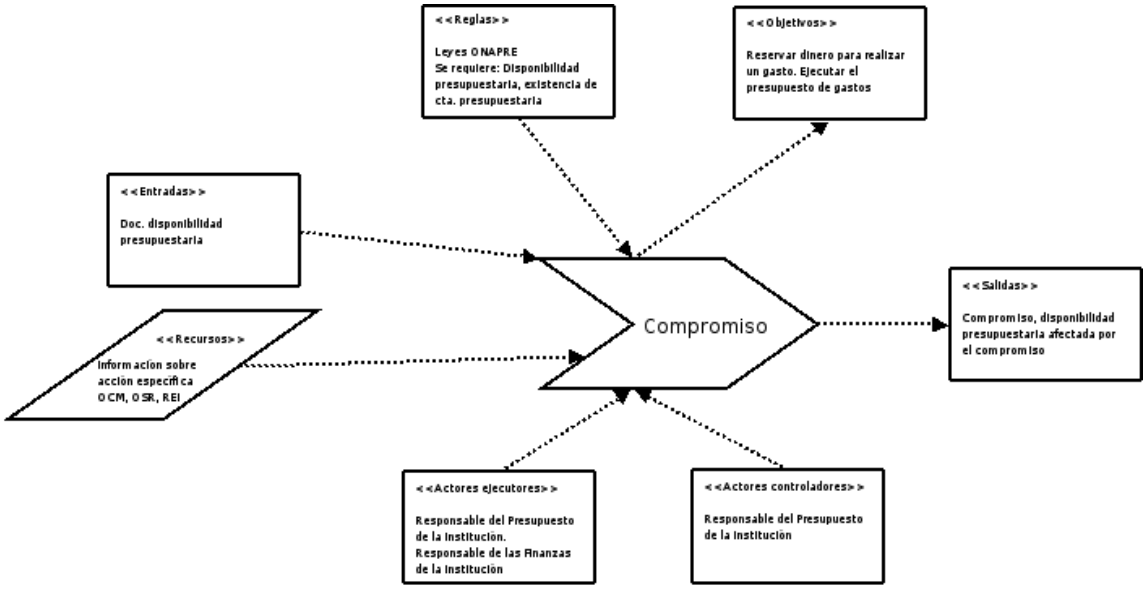
  Ejemplo del diagrama de relación entre procesos:



- **Diagrama de Procesos Caja Negra**

  Este diagrama es utilizado para representar los insumos de entrada a un proceso, los actores que intervienen y los productos que se obtienen en éste.

  Ejemplo del diagrama caja negra del proceso Compromiso del SAID:

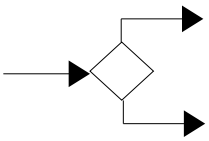
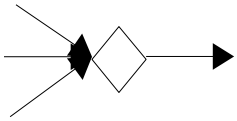
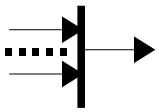
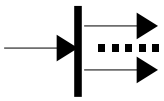


- **Diagramas de Actividades del UML 2.0**

Los diagramas de actividad se utilizan para representar la secuencia de acciones de una actividad y sus resultados. En la tabla que se muestra a continuación se indican los símbolos a utilizar para los diagramas de actividades, según el lenguaje UML 2.0.

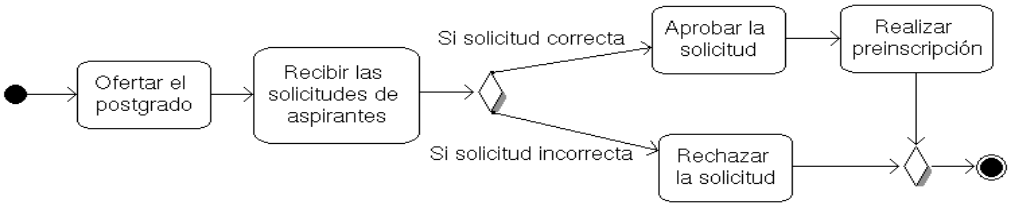
Tabla Nº 2. Símbolos utilizados para representar diagramas de actividades según el UML 2.0

Nombre	***Símbolo***	***Utilidad***
Acción	Nombre de la acción	Se utilizada para representar una acción.
Objeto	Nombre del Objeto	Se utiliza para representar objetos de entrada o salida a una acción.
Flujo de control o secuencia	——►	Se utiliza para indicar la secuencia de ejecución entre un conjunto de acciones.
Flujo de objetos	- - - - - - - - ►	Se utiliza para indicar recursos que fluyen entre acciones.

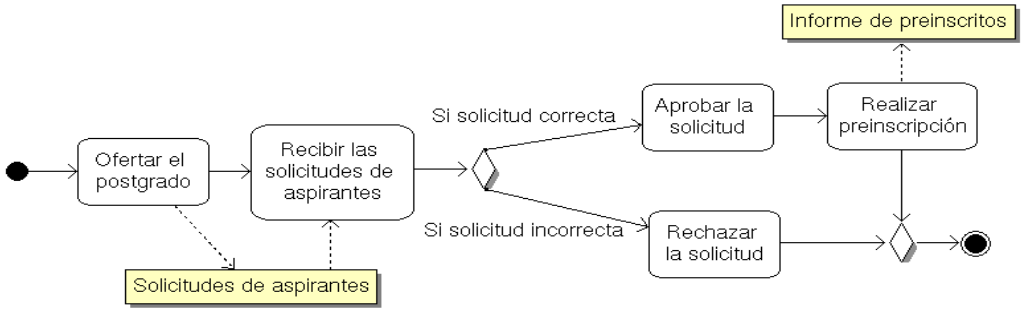
Nombre	***Símbolo***	***Utilidad***
Nodo de decisión		Se utiliza para indicar bifurcaciones de decisiones entre acciones.
Nodo de mezcla		Se utiliza para unir dos o más flujos de alternativos.
Nodo de concurrencia		Se utiliza para sincronizar múltiples flujos.
Nodo de sincronización		Se utiliza para dividir un flujo en dos o más flujos concurrentes (paralelos).
Almacén de datos	Nombre del Almacén de datos	Se utiliza para representar un almacén de información persistente (archivo o base de datos).
Inicio		Representación gráfica utilizada para indicar el inicio de un flujo de procesos o de acciones.
Fin		Representación gráfica utilizada para indicar el fin de un flujo de procesos o de acciones.

Los diagramas de actividades se dividen en:

- Diagrama de actividad para el flujo de control: se utiliza para representar la secuencia de ejecución entre las acciones que describen una actividad. Ejemplo:



- Diagrama de actividad para el flujo de objetos: se utilizan para representar además de la secuencia entre las acciones de una actividad los objetos de entrada y salida de una acción. Es importante mencionar que los objetos de salida de una acción no siempre representan el insumo de entrada de otra acción. Ejemplo:



3.2 Modelado Funcional

El modelado funcional se utiliza para representar la funcionalidad de un sistema. La funcionalidad es el conjunto de funciones que un sistema debe prestar al usuario, es decir, están referidas a las operaciones que los usuarios pueden realizar con el sistema. Es importante mencionar que estas funcionalidades dicen que hace el sistema, más no cómo lo hace.

El modelado funcional de una aplicación consta de un conjunto de diagramas de casos de uso y un conjunto de descripciones textuales de los casos de uso.

3.2.1 Conceptos Básicos del Modelado Funcional

- **Actor**

  Representa todas aquellas personas, entidades, dispositivos u otros sistemas que interactúa con el sistema.

- **Función**

  Una función representa un conjunto de operaciones que un sistema es capaz de realizar.

- **Caso de Uso**

  Es una especificación de una funcionalidad de un sistema.

3.2.2 Diagramas y Formatos Utilizados para el Modelado Funcional

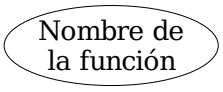
El diagrama más utilizado para modelar las funcionalidades de un sistema es el diagrama de casos de uso del lenguaje UML. En lo que respecta a la descripción textual del caso de uso se utilizan varios tipos de formatos. Estos formatos varían dependiendo del grado de especificación que se requiera para los casos de uso de un sistema.

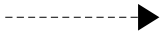
- **Diagramas de Casos de Uso**

  Estos diagramas permiten representar gráficamente las funcionalidades de un sistema y los actores relacionados a éstas.

  En la Tabla Nº 3 se indican los símbolos a utilizar para los diagramas de casos de uso, según el lenguaje UML 2.0.

  Tabla Nº 3. Símbolos utilizados para representar casos de uso según el UML 2.0

Nombre	***Símbolo***	***Utilidad***
Actor	 Rol	Se utilizado para representar el rol que juega una persona, una entidad o un dispositivo en el sistema.
Caso de uso		Usado para representar la función que el sistema pone a disposición de los actores. El nombre del caso de uso debe ser un

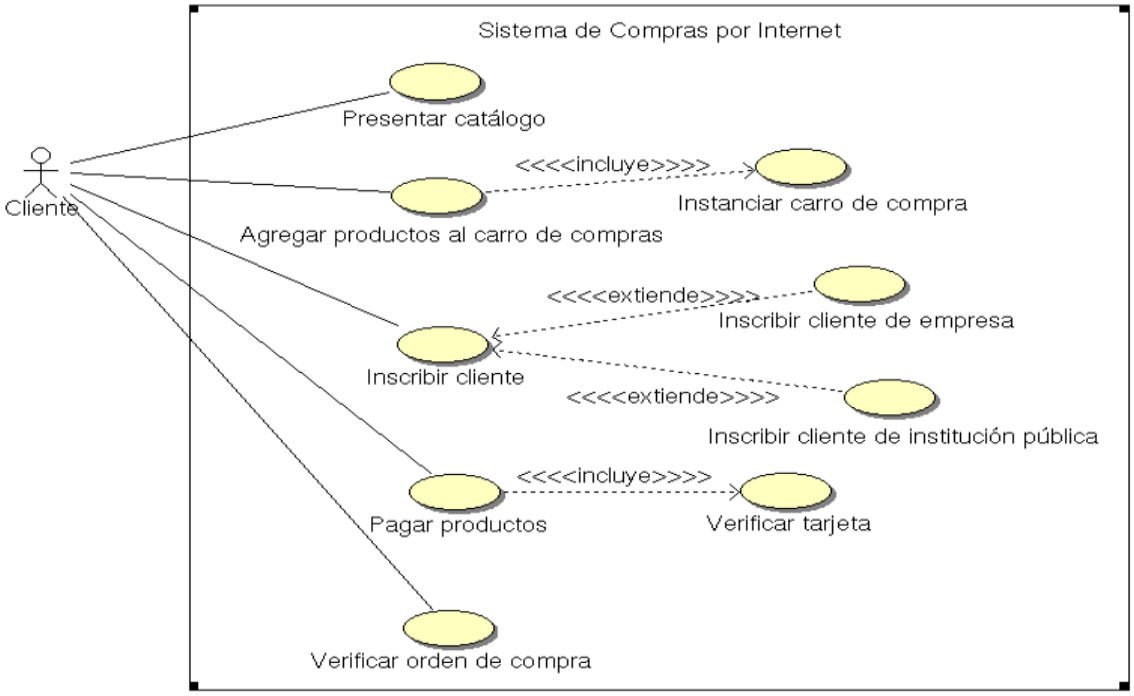
Nombre	***Símbolo***	***Utilidad***
		verbo.
Relación de comunicación		Usado para establecer un asociación bidireccional entre un actor y un caso de uso.
Relación de inclusión	<<incluye>> 	Se utiliza para establecer una relación entre dos casos de uso, en la cual un caso de uso específico reutiliza otro caso de uso encapsulado en distintos contextos a través de su invocación. (Ver Nota1)
Relación de extensión	<<extiende>> 	Usado para establecer una relación entre dos casos de uso, en la cual un caso de uso amplia su funcionalidad, mediante la extensión de sus secuencias de acciones, a través de otro caso de uso. (Ver Nota 2)
Relación de generalización		Se utilizada para establecer una relación entre dos casos de uso, en la cual un caso de uso (hijo) hereda comportamiento y atributos de otro caso de uso (padre). (Ver Nota 3)

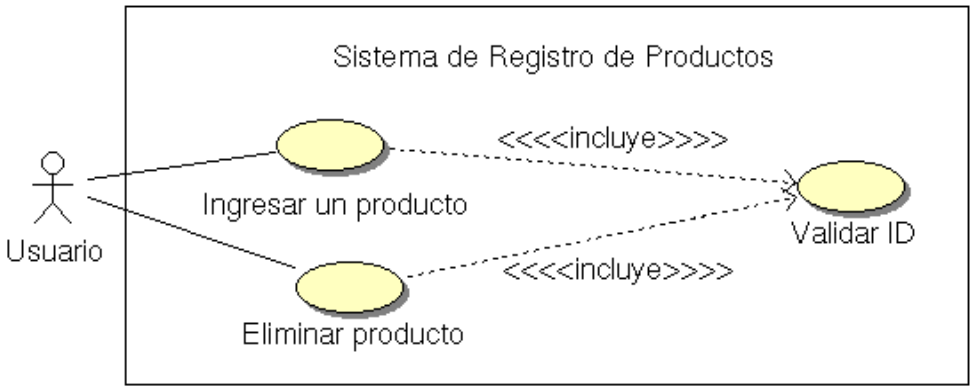
Nota 1: En la relación de inclusión el caso de uso base siempre utilizará el caso de uso incluido. La relación de inclusión es muy utilizada para evitar describir el mismo flujo de eventos en varios casos de uso, por ello se coloca el comportamiento común a esos casos de uso en un caso de uso aparte.

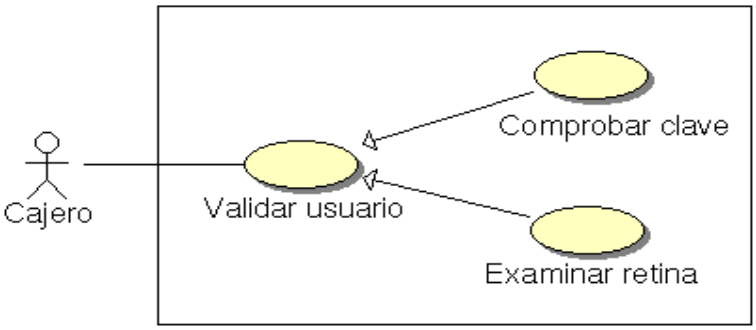
Nota 2: En la relación de extensión el comportamiento del caso de uso base es extendido por otro caso de uso. La relación de extensión se utiliza para especificar comportamientos diferentes de un mismo caso de uso ante ciertas circunstancias.

Nota 3: Esta relación de generalización se usa muy poco entre los casos de uso, dado que el significado que se le da a ésta es muy similar al significado dado a la relación de extensión.

Ejemplos de Casos de Uso:







- **Descripción Textual de Casos de Uso (Formato)**

  Esta descripción permite expresar en forma textual las actividades que se llevan a cabo en un caso de uso, incluyendo en ellas los detalles de interacción entre el sistema y el usuario. Esta descripción debe cubrir todo el proceso de ejecución de la función, desde que el usuario la invoca hasta que recibe los resultado de la misma.

  La descripción textual se utiliza para complementar los diagramas de casos de uso, dado que el conjunto de acciones que se llevan a cabo en éstos no pueden capturarse fácilmente a través de estos diagramas.

  A continuación se presenta en la Tabla Nº 4 el formato a seguir en el curso para la descripción textual de los casos de uso.

  Tabla Nº 4. Formato para la descripción textual de un caso de uso

Nombre del caso de uso: <Nombre del caso de uso>	
Actores participantes: <Se coloca el nombre de los actores que participan en el caso de uso>	
Condiciones de entrada: <Se indican las condiciones de entrada al caso de uso, entre ellas el evento que da inicio al caso de uso así como otras condiciones que se requieran para que éste se lleve a cabo>	
Condiciones de salida: <Se indican las condiciones de salida del caso de uso, es decir, los resultados que se obtienen una vez concluido el caso de uso>	
Flujo básico	<Se indica el flujo de actividades, en condiciones normales, del caso de uso. Cada una de estas actividades es enumerada>

Flujos alternativos	<Se indican los flujos alternativos del caso de uso, es decir, las actividades que se pueden generar en situaciones distintas a las condiciones normales establecidas para el caso de uso. Cada flujo alternativo representa el conjunto de acciones generadas dada una condición contraria a una actividad descrita en el flujo básico. Cada flujo alterno debe ser enumerado en base a la numeración que tenga la actividad del flujo básico de la cual se deriva éste>
Requisitos especiales	<En este campo se indican los requisitos especiales asociados al caso de uso en particular. Por ejemplo, se puede indicar el máximo y el mínimo de caracteres que debe contener una cadena, el tipo de dato al que debe pertenecer algún atributo indicado en el caso de uso, así como cualquier otra restricción que limite el caso de uso>

Ejemplos de Descripciones Textuales de Casos de Uso:

- *Descripción textual del Caso de Uso "Inscribir cliente"*:

Nombre del caso de uso: Inscribir Cliente	
Actores participantes: Cliente	
Condiciones de entrada: El cliente pulsa el icono *Inscribir cliente*	
Condiciones de salida: Cliente inscrito en el sistema	
Flujo básico	1. Cuando el cliente pulsa el icono *Inscribir Cliente* el sistema muestra por pantalla un formato en el cual el cliente debe indicar los siguientes datos: nombre, apellido, cédula de identidad, dirección y teléfono. 2. 2. El cliente indica los datos solicitados. 3. 3. El sistema registra los datos. 4. 4. El sistema presenta por pantalla un cuestionario en el cual pregunta al cliente si trabaja para una empresa o para una institución pública. 5. 5. Si el cliente indica que trabaja para una institución pública el sistema solicita los siguientes datos: nombre de la organización, ministerio al cuál está adscrita, dirección y teléfono. 6. 6. Si el cliente indica que trabaja para una empresa el sistema solicita los siguientes datos: nombre de la empresa, dirección y teléfono. 7. 7. El sistema registra los datos. 8. 8. El sistema muestra por pantalla el siguiente mensaje: Los datos fueron registrados correctamente.
Flujos	2.1 Si el cliente introduce algún(s) dato(s) de manera incorrecta el sistema

alternativos	presenta por pantalla el siguiente mensaje: "Por favor introduzca correctamente sus datos". 5.1 Si el cliente introduce algún(s) datos(s) de la institución de manera incorrecta el sistema presenta por pantalla el siguiente mensaje: "Por favor introduzca correctamente los datos de la institución". 6.1 Si el cliente introduce algún(s) datos(s) de la empresa de manera incorrecta el sistema presenta por pantalla el siguiente mensaje: "Por favor introduzca correctamente los datos de la empresa".
Requisitos especiales	– El campo del nombre del cliente debe estar comprendido entre 4 y 8 caracteres. – El campo del apellido del cliente debe estar comprendido entre 5 y 10 caracteres.

- *Descripción textual del Caso de Uso "Eliminar producto"*:

Nombre del caso de uso: Eliminar producto	
Actores participantes: Usuario	
Condiciones de entrada: El usuario pulsa el icono *Eliminar producto*	
Condiciones de salida: Producto eliminado de la lista de productos existentes	
Flujo básico	1. Cuando el cliente pulsa el icono *Eliminar producto* el sistema procede a validar el ID del producto a eliminar. 2. Si el producto está registrado en la base de datos el sistema lo elimina de ésta.
Flujos alternativos	1.1 Si el ID no coincide con el ID de ninguno de los productos que se tienen registrados el sistema reporta por pantalla el siguiente mensaje: "El producto que se quiere eliminar no existe en la base de datos".
Requisitos especiales	

3.3 Modelado de Comportamiento

El modelado de comportamiento se utiliza para representar lo que debe suceder en el sistema, para lo cual se hace énfasis en el flujo de control de datos entre los elementos que componen el sistema.

3.3.1 Conceptos Básicos del Modelado de Comportamiento

- **Objeto**

  Constituye una entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos). Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.

- **Clase**

  La clase representa las definiciones de las propiedades y comportamiento de un tipo de objeto concreto.

- **Método**

  Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". El método representa entonces lo que el objeto puede hacer. Los métodos pueden producir cambios en las propiedades de un objeto.

- **Mensaje**

  Es una comunicación dirigida a un objeto en la cual se le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que generó el mensaje.

3.3.2 Diagramas Utilizados para el Modelado de Comportamiento

Los diagramas más utilizados del lenguaje UML para modelar el comportamiento de un sistema son los diagramas de interacción y los diagramas de maquinas de estado. A continuación describiremos cada uno de ellos.

- **Diagramas de Interacción**

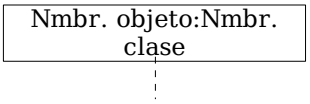
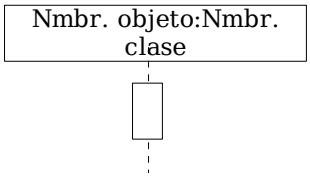
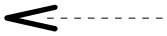
  Este tipo de diagrama se utiliza para modelar los elementos dinámicos de un sistema junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. En el contexto de las clases describen la forma en que grupos de objetos colaboran para proveer un comportamiento.

Existen dos tipos de diagramas de interacción: los diagramas de secuencia y los diagramas de comunicación. En el curso solo utilizaremos los diagramas de secuencia.

- **Diagramas de Secuencia**

  Estos diagramas permiten mostrar los objetos que se encuentran en un escenario (de un caso de uso) y la secuencia de mensajes intercambiados entre éstos para llevar a cabo la funcionalidad descrita por el escenario.

  En la Tabla Nº 5 se indican los símbolos que utilizaremos para los diagramas de secuencias, según el lenguaje UML 2.0.

Nombre	***Símbolo***	***Utilidad***
Línea de vida		Se utiliza para representar la existencia de un objeto a lo largo de un periodo de tiempo.
Ejecución		Utilizado para representar el periodo de tiempo durante el cual un objeto ejecuta una acción.
Destrucción	X	Se utiliza para indicar cuando un objeto es destruido.
Retorno		Usado para representar la respuesta a un mensaje.
Mensaje síncrono		Se utiliza para representar un mensaje síncrono. En este caso el emisor del mensaje espera que el destinatario termine de tratar su mensaje.
Mensaje asíncrono		Se utiliza para representar un mensaje asíncrono. En este caso el emisor del mensaje no espera que el destinario trate el mensaje y continua con sus quehaceres.

Nombre	*Símbolo*	*Utilidad*
Mensaje recursivo síncrono		Usado para representar la llamada que un objeto hace a un método de si mismo. En este caso el mensaje de llamada al método es de tipo síncrono.
Mensaje recursivo asíncrono		Se utiliza para representar la llamada que un objeto hace a un método de si mismo. En este caso el mensaje de llamada al método es de tipo asíncrono.

A las fechas utilizadas para representar los tipos de mensajes se le asocian etiquetas con el mensaje y los argumentos de éste. Es importante destacar que es posible añadir a los tipos de mensajes condiciones e iteraciones.

Las condiciones se utilizan para decidir, bajo ciertos parámetros, cuando se envía un mensaje. Esta se representa colocando entre corchetes la condición y seguidamente el mensaje. Ejemplo: [x= verdadero]mensajeB()

Las iteraciones se utilizan para indicar las veces en que un mensaje es enviado. La iteración se representa mediante un asterisco y una expresión entre corchetes indicando el número de veces en que se envía el mensaje. Ejemplo: *[i=1...10]mensajeC().

Para elaborar un diagrama de se secuencia de un caso de uso se requiere:

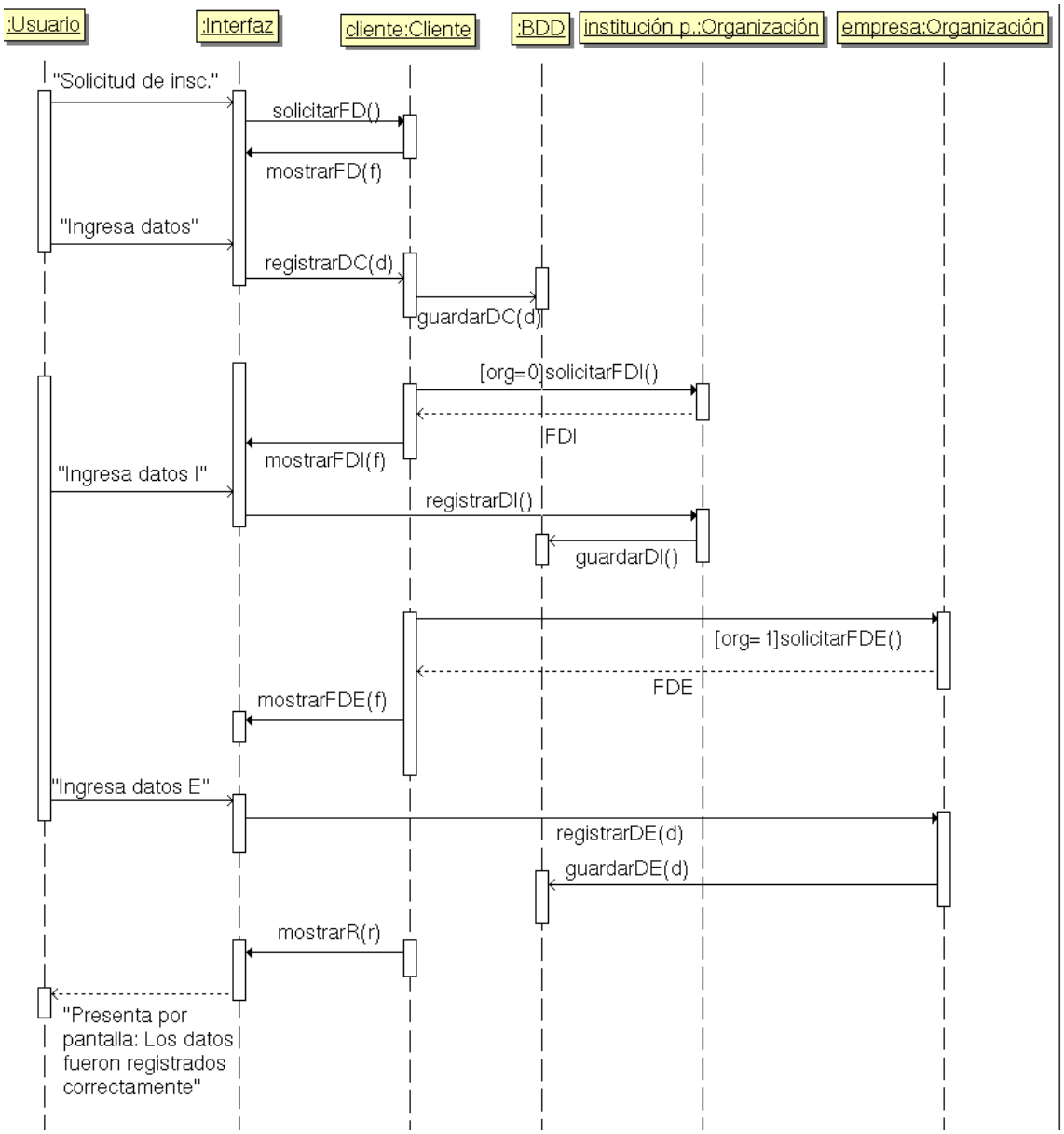
1. Tener a la mano la descripción textual del caso de uso.
2. Identificar los objetos asociados al caso de uso.
3. Identificar las interfaces requeridas para el caso de uso.
4. Identificar los métodos que se activan para ejecutar el caso de uso.
5. Decidir a que objetos o interfaces corresponden los métodos identificados.
6. Establecer la secuencia de activación de los métodos identificados.

Ejemplo de Diagrama de Secuencia:

A continuación se presenta el caso de uso "Inscribir Cliente" para el cual se modelará un diagrama de secuencia.

Nombre del caso de uso: Inscribir Cliente	
Actores participantes: Usuario	
Condiciones de entrada: El usuario pulsa el icono *Inscribir cliente*	
Condiciones de salida: Cliente inscrito en el sistema	
Flujo básico	1. Cuando el usuario pulsa el icono *Inscribir Cliente* el sistema muestra por pantalla un formato en el cual el usuario debe indicar los siguientes datos: nombre, apellido, cédula de identidad. 2. El usuario indica los datos solicitados. 3. El sistema registra los datos. 4. El sistema presenta por pantalla un cuestionario en el cual pregunta al usuario si trabaja para una empresa o para una institución pública. 5. Si el usuario indica que trabaja para una institución pública el sistema solicita los siguientes datos: nombre de la organización, ministerio al cuál está adscrita, dirección y teléfono. 6. Si el usuario indica que trabaja para una empresa el sistema solicita los siguientes datos: nombre de la empresa, dirección y teléfono. 7. El sistema registra los datos. 8. El sistema muestra por pantalla el siguiente mensaje: Los datos fueron registrados correctamente.
Flujos alternativos	
Requisitos especiales	

Diagrama de Secuencia para el Caso de Uso: Inscribir Cliente



3.4 Modelado Estructural

El modelado estructural se utiliza para representar las clases y objetos que componen la aplicación de software, además de las relaciones entre éstas.

3.3.1 Conceptos Básicos del Modelado Estructural

- **Objeto**

  Constituye una entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos). Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa).

  Atributos: constituyen las propiedades que un modelador le atribuye al objeto, por ejemplo, atributos de identificación, atributos físicos, etc. Cada atributo tiene asociado uno o más valores (por ejemplo, cédula= "14.556.953"), los cuales determinan el estado de un objeto para un instante determinado.

  Comportamiento: constituye el conjunto de operaciones (acciones) que un objeto puede realizar.

- **Clase**

  La clase es la unidad básica que encapsula toda la información de los objetos que tienen atributos y operaciones comunes. Los atributos y operaciones de un clase pueden ser de tres tipos:

  - *Público (+)*: indica que los atributos u operaciones de una clase pueden ser utilizados por otras clases.
  - *Protegido (#)*: indica que los atributos u operaciones de una clase sólo pueden ser utilizados por subclases.
  - *Privado (-)*: indica que los atributos u operaciones de un clase sólo pueden ser utilizados por los objetos que pertenecen a dicha clase.

  Las clases se dividen en tres tipos:

  - Clase interfaz: describe el comportamiento visible de una clase.
  - Clase abstracta o parametrizable: es una clase genérica que no puede ser instanciada pues posee métodos abstractos (aún no han sido definidos, es decir, sin implementación). Una clase abstracta se denota con el nombre de la clase y de los métodos con letra "itálica". La única forma de utilizar una clase abstracta es definiendo subclases, que implementan los

métodos abstractos definidos.

- Clase utilitaria: es una clase que contienen los atributos y comportamiento (funciones) de un objeto.

- **Instancia**

  Una instancia es cada objeto que pertenece a una clase.

- **Mensajes**

  Representan la especificación de un objeto junto con la invocación de uno sus métodos (operaciones). Ejemplo: nombreObjeto.nombreMétodo(listaParámetros).

- **Encapsulación**

  Es una propiedad que permite que los objetos sean definidos en su estructura y su comportamiento, obligando al uso de mensajes o de la invocación de sus métodos, si se quiere acceder al objeto.

- **Polimorfismo**

  Es una propiedad que permite que operaciones de diferentes clases tengan el mismo nombre sin importar la relación entre las mismas.

- **Herencia**

  Es una propiedad que tienen las clases (subclases) de heredar atributos y/o comportamiento de superclases.

- **Cardinalidad (Multiplicidad)**

  Es una propiedad que indica el grado de dependencia entre la relación de dos o más clases. La cardinalidad pueden ser de:

  - Uno o muchos: 1..* (1..n)
  - 0 o muchos: 0..* (0..n)
  - Muchos: *
  - Número fijo: m (m denota el número)

3.3.2 Diagramas Utilizados para el Modelado Estructural

Los diagramas más utilizados del lenguaje UML para modelar la estructura de un sistema son los diagramas de clase.

- **Diagramas de clase**

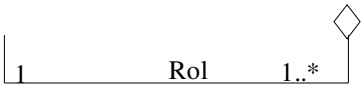
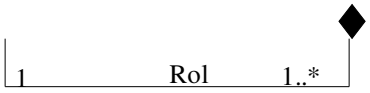
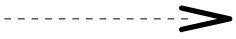
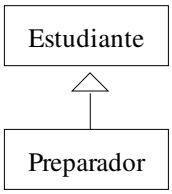
  Este tipo de diagrama permite representar las clases de objetos del sistema y la relación entre éstas. Las clases se pueden modelar de forma simple o de forma extendida. Cuando se modelan de forma simple sólo se coloca el nombre de la clase, cuando se modelan de forma extendida se coloca el nombre de la clase, los atributos y los operaciones.

  En el caso particular del curso se recomienda modelar las clases de un sistema de la siguiente forma:

  1. En primer lugar, representar la relación entre las clases de un sistema de forma simple, es decir, mostrando sólo el nombre de cada clase y la relación entre estas.
  2. En segundo lugar, representar cada clase de forma extendida, es decir, mostrando por cada clase su nombre, sus atributos y sus métodos, sin indicar relaciones entre clases.

  En la Tabla Nº 6 se indican los símbolos que utilizaremos para los diagramas de clase, según el lenguaje UML 2.0.

Nombre	***Símbolo***	***Utilidad***
Clase	nombreDeLaClase	Se utiliza para la representación sólo el nombre de la clase.
Clase	nombreDeLaClase - nombreAtributo: nombreDelTipo + nombreAtributo: nombreDelTipo # nombreAtributo: nombreDelTipo	Se utiliza para representar una clase con atributos.

Nombre	***Símbolo***	***Utilidad***
Clase	nombreDeLaClase - nombreAtributo: nombreDelTipo + nombreAtributo: nombreDelTipo # nombreAtributo: nombreDelTipo - nombreMétodo(Parámetros): Boolean + nombreMétodo(Parámetros): Boolean	Se utiliza para representar una clase con atributos y métodos.
Asociación		Se utilizan para representar una relación de asociación entre clases con roles y multiplicidad. El rol indica el papel que una clase desempeña para con otra clase.
Agregación		Se utilizan para representar una relación de agregación entre clases (Ver Nota 1).
Composición		Se utiliza para representar una relación de composición entre clases (Ver Nota 2).
Dependencia		Se utiliza para representar una relación entre una clase dependiente y otra independiente.
Generalización/ Especialización		Se utiliza para representar una relación de herencia entre varias clases (Ver Nota 3).
Instancia de una clase	nombreInstancia:nombreClase	Se utiliza para representar una instancia de una clase.

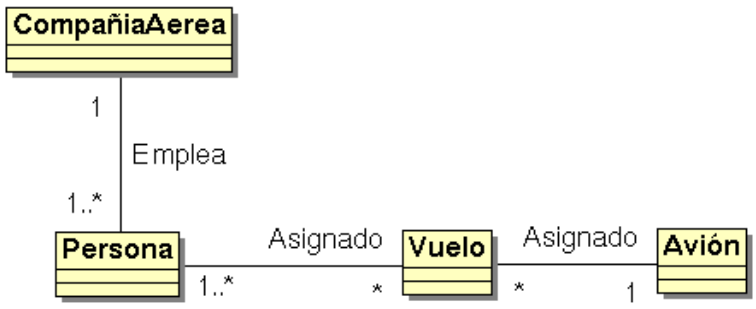
Nota 1: la agregación establece una relación "todo-partes". En este tipo de relación la existencia de los objetos de las partes no dependen de la existencia de los objetos a la cuales están agregados.

Nota 2: la composición establece una relación "es parte de" entre clase. Es un tipo particular de agregación en la cual la existencia de los objetos componentes depende de la existencia del objeto compuesto.

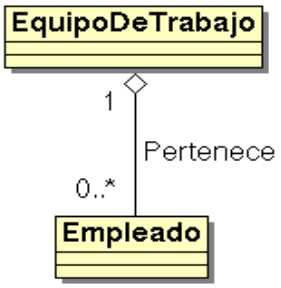
Nota 3: la generalización/especialización permite representar el hecho de que una o más clase específicas (subclases) hereden la estructura (atributos) y comportamiento de una clase genérica (superclase).

Ejemplos de diagramas de clases:

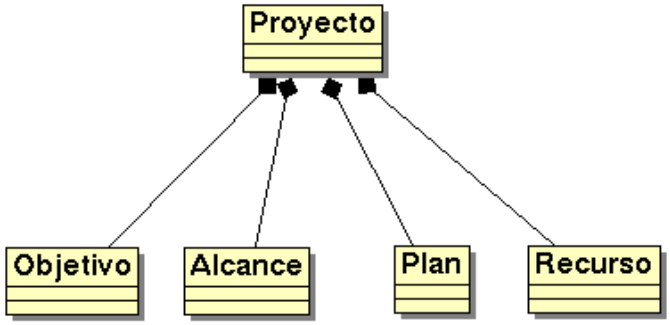
- Diagrama de clase simple para representar la relación de asociación:



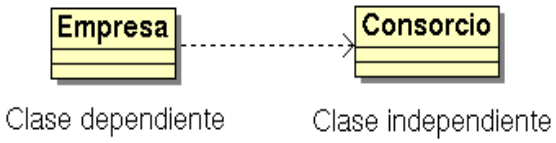
- Diagrama de clase simple para representar la relación de agregación:



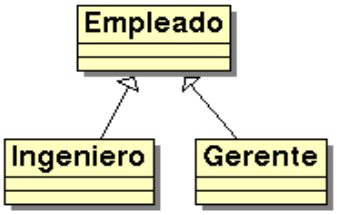
- Diagrama de clase simple para representar la relación de composición:



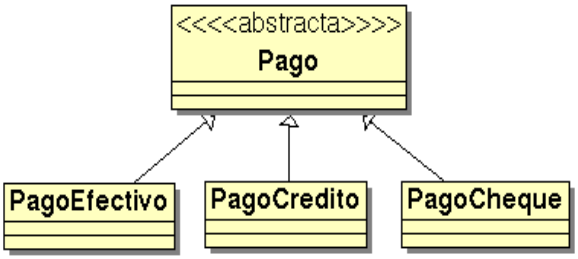
- Diagrama de clase simple para representar la relación de dependencia:



- Diagrama de clase simple para representar la relación de generalización (herencia):



- Diagrama de clase abstracta:



- Diagrama extendido de la clase “Pago”: