Contenido Temático
Tema 5: Modelado de Sistemas

Objetivos

Comprender el concepto de modelado de los sistemas e Introducir varios modelos de sistemas que pueden desarrollarse durante el proceso de ingeniería de requerimientos.

Modelado de los sistemas

Los requerimientos del usuario deberían redactarse en lenguaje natural debido a que tienen que ser comprendidos por personas que no son técnicos expertos. Sin embargo, pueden expresarse requerimientos del sistema más detallados de forma más técnica. Una técnica ampliamente usada es documentar la especificación del sistema como un conjunto de modelos del sistema. Estos modelos son representaciones gráficas que describen los procesos del negocio, el problema a resolver y el sistema que tiene que ser desarrollado.

Debido a las representaciones gráficas usadas, los modelos son a menudo más comprensibles que las descripciones detalladas en lenguaje natural de los requerimientos del sistema. Ellos constituyen también un puente importante entre el proceso de análisis y diseño. Pueden usarse modelos en el proceso de análisis para comprender el sistema existente que debe ser reemplazado o mejorado, o para especificar el nuevo sistema que sea requerido.

Pueden desarrollarse diferentes modelos para representar el sistema desde diferentes perspectivas.
El aspecto más importante de un modelo del sistema es que omite los detalles. Un modelo del sistema es una abstracción del sistema que se está estudiando en lugar de una representación alternativa de ese sistema. Idealmente, una representación de un sistema debería mantener toda la información sobre la entidad que se está representando. Una abstracción simplifica y resalta de forma deliberada las características más relevantes.

Diferentes tipos de modelos del sistema se basan en distintas aproximaciones de abstracción. Un modelo de flujo de datos (por ejemplo) se centra en el flujo de datos y las transformaciones funcionales sobre esos datos. Se omiten los detalles de las estructuras de datos. Por el contrario, un modelo de entidades de datos y sus relaciones documentan las estructuras de datos del sistema en lugar de su funcionalidad.

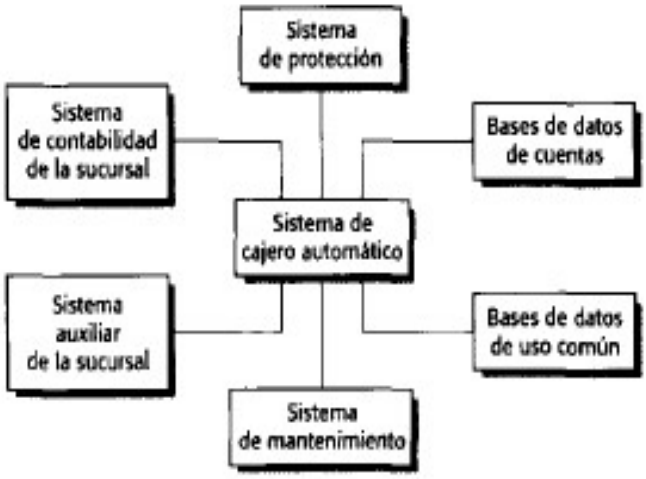
Modelos de contexto

En una de las primeras etapas de la obtención de requerimientos y del proceso de análisis se deben definir los límites del sistema. Esto comprende trabajar conjuntamente con los stakeholders del sistema para distinguir lo que es el sistema y lo que es el entorno del

sistema. Usted debería tomar estas decisiones al inicio del proceso para delimitar los costes del sistema y el tiempo necesario para el análisis. En algunos casos, el límite entre un sistema y su entorno está relativamente claro.

En otros casos, hay más flexibilidad, y usted decide lo que constituye el límite entre el sistema y su entorno durante el proceso de ingeniería de requerimientos.

La definición de un límite del sistema no es una decisión arbitraria. Aspectos sociales y organizacional pueden implicar que la situación de los límites de un sistema puedan ser determinados por factores no técnicos. Una vez que se han tomado algunas decisiones sobre los límites del sistema, parte de (a actividad del análisis es la definición de ese contexto y de las dependencias que el sistema tiene sobre su entorno. Normalmente, la producción de un modelo arquitectónico sencillo es el primer paso en esta actividad. La figura siguiente es un modelo arquitectónico que ilustra la estructura del sistema de información que incluye una red de cajeros automáticos.



En la figura, podemos ver que cada cajero automático) está conectado a una base de datos de cuentas, a un sistema de contabilidad de la sucursal, a un sistema de seguridad y a otro de mantenimiento de los cajeros. El sistema también está conectado a una base de datos de uso común que monitoriza cómo se usa la red de cajeros y también está conectado a un sistema auxiliar local de una sucursal. Este sistema auxiliar proporciona servicios tales como copias de seguridad e impresión. Éstos, por lo tanto, no necesitan incluirse en el propio sistema. Los modelos arquitectónicos describen el entorno de un sistema. Sin embargo, no muestran las relaciones entre otros sistemas del entorno y el sistema que se está especificando. Los sistemas extremos podrían producir o consumir datos del sistema. Podrían compartir datos con el sistema, o podrían ser conectados directamente, conectados a través de una red o no estar conectados. Podrían estar físicamente en el mismo lugar o localizados en edificios diferentes. Todas estas relaciones podrían afectar a los requerimientos del sistema que se está definiendo y deben tenerse en cuenta. Por lo

tanto, los modelos arquitectónicos sencillos normalmente se complementan con otros modelos, tales como modelos de procesos, que muestran las actividades de los procesos soportadas por el sistema. Los modelos de flujo de datos también pueden usarse para mostrar los datos que son transferidos entre el sistema y otros sistemas de su entorno. La figura siguiente ilustra un modelo de proceso de adquisición de equipos de una organización.

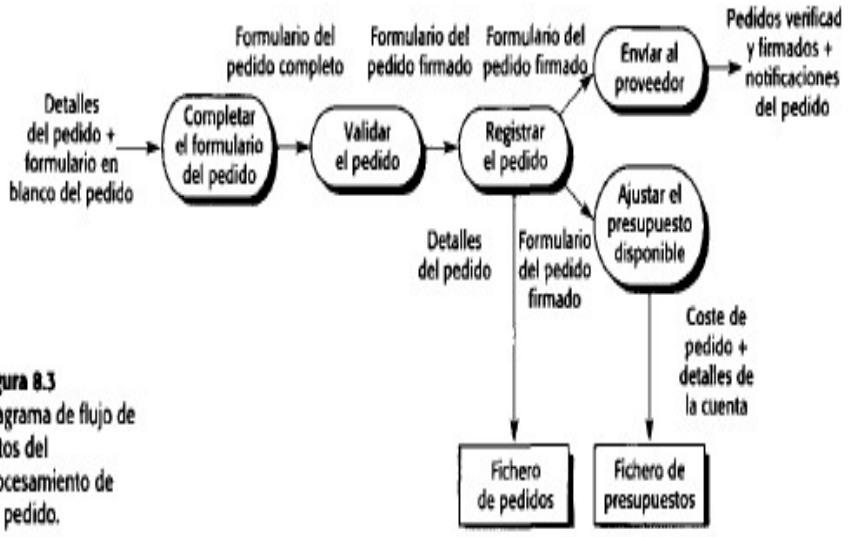


Figura 8.3
Diagrama de flujo de datos del procesamiento de un pedido.

Modelos de comportamiento

Los modelos de comportamiento se utilizan para describir el comportamiento del sistema en su totalidad. Aquí se analizan dos tipos de modelos de comportamiento: modelos de flujo de datos, que modelan el procesamiento de los datos en el sistema, y modelos de máquinas de estado, que modelan cómo el sistema reacciona a los eventos. Estos modelos pueden usarse de forma separada o conjuntamente, dependiendo del tipo de sistema que se esté desarrollando. La mayoría de los sistemas de negocio están fundamentalmente dirigidos por los datos. Están controlados por las entradas de datos al sistema con relativamente poco procesamiento de eventos externos. Un modelo de flujo de datos puede ser todo lo que se necesite para representar el comportamiento de estos sistemas. Por el contrario, los sistemas de tiempo real a menudo están dirigidos por eventos con un mínimo procesamiento de datos. Un modelo de máquina de estados (analizado en la Sección 8.2.2) es la forma más efectiva de representar su comportamiento. Otras clases de sistemas pueden estar dirigidas tanto por datos como por eventos. En estos casos usted puede desarrollar ambos tipos de modelos.

Modelos de flujo de datos

Los modelos de flujo de datos son una forma intuitiva de mostrar cómo los datos son procesados por un sistema. A nivel de análisis, deberían usarse para modelar la forma en la que los datos son procesados en el sistema existente. Estos modelos son una parte intrínseca de los métodos estructurados que proponía Yourdon.

Los modelos de flujo de datos se utilizan para mostrar cómo fluyen los datos a través de una secuencia de pasos de procesamiento. Estos pasos de procesamiento o transformaciones representan procesos software o funciones cuando los diagramas de flujo de datos se utilizan para documentar un diseño software. Sin embargo, en un modelo de análisis, el procesamiento se puede llevar a cabo por las personas o por las computadoras. Los modelos de flujo de datos son valiosos debido a que realizan un seguimiento y documentan cómo los datos asociados con un proceso particular fluyen a través del sistema, y esto ayuda a los analistas a comprender el proceso. Los diagramas de flujo de datos tienen la ventaja de que, a diferencia de otras notaciones de modelado, son sencillos e intuitivos. Normalmente es posible explicarlos a los usuarios potenciales del sistema, quienes pueden entonces participar en la validación del análisis.

Los modelos de flujo de datos muestran una perspectiva funcional en donde cada transformación representa un único proceso o función. Son particularmente útiles durante el análisis de requerimientos ya que pueden usarse para mostrar el procesamiento desde el principio hasta el final en un sistema. Es decir, muestra la secuencia completa de acciones que tienen lugar a partir de una entrada que se está procesando hasta la correspondiente salida que constituye la respuesta del sistema.

=>

Componentes del Modelo orientado al flujo.

El **Proceso**. Burbujas, función o transformación). Muestra una parte del Sistema que transforma entradas en salidas. La burbuja se debe describir con un nombre o frase. Es conveniente la utilización de Verbos y objetos.

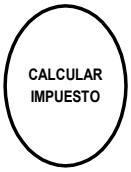


Fig. 04-01: Grafica de Proceso en los DFD

El **Flujo**. Describe el movimiento de bloques o paquetes de información de una parte del Sistema a otra.- Los flujos representan "datos en movimientos".

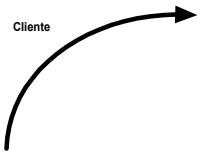


Fig. 04-02: Grafica de un Flujo en los DFD

El **Almacén**. Se utiliza para modelar una colección de paquetes de datos en reposo. Se representa con dos líneas paralelas. Los almacenes se conectan por flujos a los procesos.

Fig. 04-06: Almacen

El **Terminador**. Entidad. Es una persona, un Departamento, un Grupo, que están dentro o fuera de la Organización. En otros casos, la entidad puede ser otro Sistema con la cual existe una comunicación (retro-alimentación). (Las relaciones que pudieran existir entre las entidades, no se reflejan en un D.F.D. (no son partes del Sistema que se está estudiando).

Fig. 04-07: Terminador

Guías para la construcción de los DFD.

Se deberá escoger nombres con significados para los procesos, flujos, entidades y almacenes. Para ello es aconsejable NO utilice nombres de personas, más bien, usar la conjunción de verbo-objeto. Evitar utilizar abreviaturas y acrónimos y no utilizar terminología orientadas a Programación (Ej.: Rutinas, Subsistemas, Procesos, etc.)

Es conveniente numerar los procesos. El enumerar no implica una secuencia, pero si una referencia jerárquica y una forma de referencia. Y será necesario re dibujar los DFD las

veces que sean necesarios. Debe ser leído fácilmente, asimilado y placentero a la vista. Se deben evitar los DFD muy complejos. Debe ser estéticamente agradable.

Se busca construir DFD que sean consistentes para ello se deberá evitar la construcción de burbujas que sean "sumideros infinitos" (agujeros negros). Tienen entradas pero no salidas. Evitar burbujas de "generación espontánea". Tienen salidas, sin entradas. Y se aconseja tener cuidado con los flujos y procesos NO ETIQUETADOS, como así también, los almacenes de solo lectura o solo escritura.

DFD por niveles

Normalmente al construir los DFD nos preguntamos cuantos niveles debe existir o cual es la cantidad de niveles que se esperan en un sistema "típico".

La regla que se sigue es que los flujos de datos que salen y entran de una burbuja en un nivel dado, deben corresponder con los que entran y salen de toda la figura en el nivel inmediato inferior que la describe. Es decir, que sean consistentes.

Únicamente para almacenes locales, que utilicen solo las burbujas en una figura de menor nivel, no se mostraran en niveles superiores, dado que se incluyen de manera implícita en un proceso del nivel inmediato superior

Herramientas para definir el ambiente

A. *Declaración de propósitos*. Es una declaración textual breve y concisa del propósito del Sistema, dirigida al administrativo superior. También algún analista sugieren que la declaración de propósito debe incluir un resumen de beneficios tangibles y cuantificables a tal vez un análisis de costo-beneficios.
B. *Diagrama de contexto*. Distingue lo que es el sistema y lo que es el entorno del sistema.
C. *Lista de acontecimientos*. Es un listado sencillo de los acontecimientos del ambiente a los cuales debe responder el sistema. Al crear esta lista se debe distinguir entre un acontecimiento y un flujo relacionado con un acontecimiento. La lista de acontecimientos debe incluir no solo las interacciones normales entre el sistema y sus terminadores sino también situaciones de falla.

El sistema necesita cada flujo (normal, de control, temporal) de entrada del diagrama de contexto para reconocer que ha ocurrido un acontecimiento; debe necesitarlo para producir una respuesta a un acontecimiento, o ambas cosas. Cada flujo de salida debe ser respuesta a un acontecimiento. Cada acontecimiento no temporal de la lista de acontecimientos debe tener entradas a partir de las cuales el sistema puede detectarlo. Cada acontecimiento debe producir salidas inmediatas como respuesta o bien almacenar los datos que luego serán salidas, o debiera ocasionar un cambio en el estado del sistema.

Existen tres enfoques en el desarrollo del modelo de flujo: a) descendente (clásico); b) el ascendente; c) el medio.

El enfoque clásico

Se procede la única burbuja del diagrama de contexto a un DFD de nivel Superior (Nivel 0), en donde cada burbuja representa un Subsistema Principal, cada burbuja del nivel 0, se parte a continuación en figuras de nivel inferior u cada burbuja de nivel inferior se parten aún más, etc., hasta llegar a un nivel de burbujas atómicas (no requiere de mayor descomposición).

Problemas en el presente enfoque: a) parálisis del análisis, es decir, que no sabe por dónde empezar; b) el fenómeno de un a X- cantidad de analistas o c) partición física arbitraria.

El enfoque ascendente consiste en desarrollar el diagrama de más bajo nivel (burbujas atómicas) desde el inicio, y luego ir desandando hacia el arriba, hasta obtener el diagrama de contexto.

El enfoque medio (hibrido) requiere (después de desarrollar DFD inicial) una nivelación ascendente y también podría necesitarse alguna partición descendente. El enfoque es el siguiente: Se dibuja una burbuja o proceso por cada acontecimiento de la lista. Esta burbuja se nombra describiendo la respuesta que el sistema debe dar al acontecimiento asociado. Luego se dibujan las entradas y salidas apropiadas de tal forma que la burbuja pueda dar repuesta requerida y se dibujan los almacenes, como sea apropiado, para la comunicación entre burbujas. El borrador de DFD que resulta se compara con el diagrama de contexto y la lista de acontecimientos para asegurar que este completo y sea consistente. Este proceso se realiza por cada uno de los acontecimientos del sistema.

Modelos de máquina de estados

Un modelo de máquina de estados describe cómo responde un sistema a eventos internos o extremos. El modelo de máquina de estados muestra los estados del sistema y los eventos que provocan las transiciones de un estado a otro. No muestra el flujo de datos dentro del sistema. Este tipo de modelo se utiliza a menudo para modelar sistemas de tiempo real debido a que estos sistemas suelen estar dirigidos por estímulos procedentes del entorno del sistema.

Un modelo de máquina de estados de un sistema supone que, en cualquier momento, el sistema está en uno de varios estados posibles. Cuando se recibe un estímulo, éste puede disparar una transición a un estado diferente.

En la figura siguiente se muestra un diagrama que enseña un modelo de máquina de estados de un sencillo horno microondas equipado con botones para fijar la potencia y el temporizador y para iniciar el sistema. Los hornos microondas reales son actualmente

mucho más complejos que el sistema descrito aquí. Sin embargo, este modelo incluye las características esenciales del sistema. Para simplificar el modelo, se supone que la secuencia de acciones al usar el microondas es: 1. Seleccionar el nivel de potencia (ya sea media o máxima); 2. Introducir el tiempo de cocción; 3. Pulsar el botón de inicio, y la comida se cocina durante el tiempo establecido.

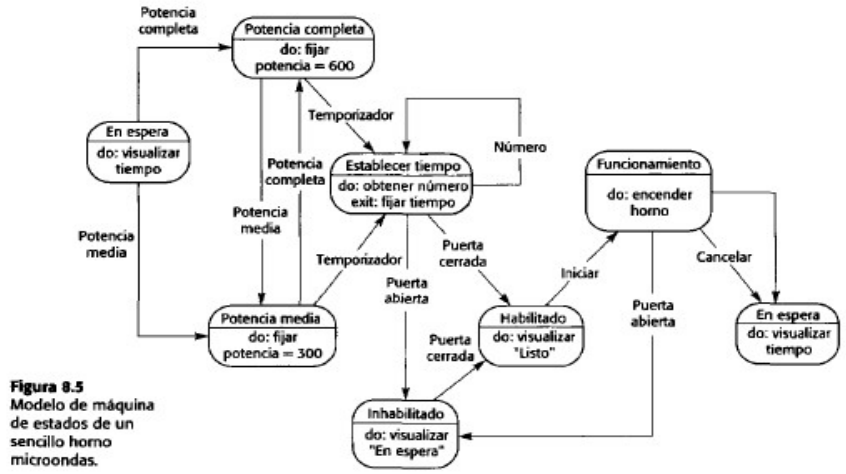
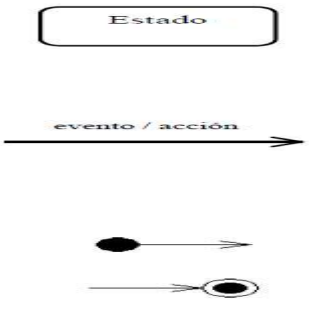


Figura 8.5
Modelo de máquina de estados de un sencillo horno microondas.

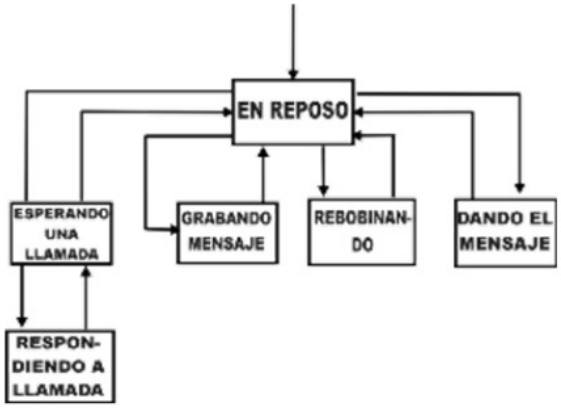
Por razones de seguridad, el homo no debería funcionar cuando la puerta esté abierta y. cuando se completa la cocción, suena un timbre. El homo dispone de una pantalla alfanumérica sencilla que se utiliza para visualizar varios mensajes de alerta y de precaución.

Componentes de un DTE

La notación que se presenta es de propósito general que se puede utilizar para cualquier tipo de modelado de máquina de estados. Los rectángulos redondeados en un modelo representan los estados del sistema. Las flechas etiquetadas representan estímulos que fuerzan transiciones de un estado a otro.



El problema con la aproximación de la máquina de estados es que el número de posibles estados crece rápidamente. Por lo tanto, para los modelos de sistemas grandes, es necesaria una cierta estructuración de estos modelos de estados. Una forma de hacer esto es mediante la noción de un superestado que encierra a varios estados separados. Este superestado se asemeja a un único estado de un modelo de alto nivel, pero se expande a continuación con más detalle en un diagrama distinto.



La figura anterior (un contestador de llamadas), muestra el comportamiento de un contestador de teléfono normal. Los principales componentes del diagrama son estados, y flechas que representan los cambios de estado.

Cada rectángulo representa un **estado** en el que se puede encontrar el sistema. Pudiendo ser este:

- Esperar a que el usuario dé su contraseña.
- Calentar una mezcla de sustancias químicas.
- Esperar la siguiente orden.
- Acelerar el motor.
- Mezclar los ingredientes.
- Esperar datos del instrumento.
- Llenar el tanque.
- Aguardar en reposo.

¿Cómo cambia un sistema de un estado a otro?.

Sí se tienen reglas ordenadas que gobiernan su comportamiento, entonces generalmente sólo algunos tipos de cambio de estado serán significativo y válidos.

Se muestran los cambios de estado válidos en el DTE conectando pares relevantes de estado con una flecha.

Así, la figura anterior muestra que el sistema puede ir del estado 1 al estado 2. También muestra que cuando el sistema se encuentra en el estado 2 puede ir al estado 3 o regresar al 1.

A pesar de que la figura proporciona información interesante acerca del comportamiento dependiente del tiempo de un sistema, no dice cuál es el estado inicial y final del sistema. La mayoría de los sistemas tienen un estado inicial reconocible y estado final reconocible:

Lo que identifica al estado 1 de la figura como inicial es la flecha "desnuda" que no está conectada a ningún otro estado, y lo que identifica al estado 5 como final es la ausencia de una flecha que salga de él.

El sentido común dice que un sistema sólo puede tener un estado inicial; sin embargo, puede tener múltiples estados finales. Los estados finales son mutuamente excluyentes, lo cual significa que sólo uno de ellos puede ocurrir durante alguna ejecución del sistema.

Condiciones y acciones.

Para completar nuestro DTE necesitamos añadir dos cosas más: las condiciones que causan un cambio de estado y las acciones que el sistema toma cuando cambia de estado. Como vemos en el siguiente diagrama, las condiciones y acciones se muestran junto a la flecha que conecta dos estados relacionados.

Una condición es un acontecimiento en el ambiente externo que el sistema es capaz de detectar; típicamente es una señal, una interrupción o la llegada de un paquete de datos. Esto usualmente hace que el sistema cambie de un estado de espera X a un estado de espera Y; o de llevar a cabo la actividad X a llevar acabo la actividad Y.

Como parte del cambio de estado, normalmente hará una o más acciones: Producirá una salida, desplegará una señal en la terminal del usuario, llevará a cabo un cálculo, etc.

Construcción del diagrama de transición de estados.

Así como en los DFD se utilizó la partición también es recomendable usarla en los DTE en donde los sistemas son muy complejos.

Para la construcción de DTE se puede seguir cualquiera de dos enfoques:

1. Se puede comenzar por identificar todos los posibles estados del sistema y representar cada uno como una caja separada en una hoja de papel. Luego, se pueden explorar todas las conexiones con significado (es decir, los cambios de estado) entre las cajas.

2. Como alternativa, se puede comenzar por el estado inicial, y luego metódicamente ir siguiendo un camino hasta el o los estados restantes; luego de los estados secundarios, proseguir al terciarios; el etc.

Cuando se termina de construir el DTE preliminar, deben seguirse las siguientes reglas para verificar la consistencia:

- ¿Se han definido todos los estados?
- ¿Se pueden alcanzar todos los estados?
- ¿Se han definido estados que no tengan caminos que lleven a ellos?
- ¿Se puede salir de todos los estados?
- ¿El sistema responde adecuadamente a todas las condiciones posibles?

El DTE representa una especificación de proceso para una burbuja de control en DFD. Como herramienta de modelado de alto nivel, el DTE puede servir incluso como especificación de proceso para todo el sistema. Si se representa todo el sistema como un diagrama de una burbuja, puede usarse el DTE para mostrar la secuencia de actividades en el sistema.

Modelos de datos

La mayoría de los sistemas software grandes utilizan bases de datos de información de gran tamaño. En algunos casos, esta base de datos es independiente del sistema software. En otros, se crea para el sistema que se está desarrollando. Una parte importante del modelado de sistemas es la definición de la forma lógica de los datos procesados por el sistema. Estos se denominan a menudo modelos semánticos de datos. La técnica de modelado de datos más ampliamente usada es el modelado Entidad-Relación (modelado DER), que muestra las entidades de datos y asociaciones.
Al igual que todos los modelos gráficos, a los modelos de datos les faltan detalles, y usted debería mantener descripciones más detalladas de las entidades, relaciones y atributos incluidas en el modelo. Usted puede reunir estas descripciones más detalladas en un repositorio o diccionario de datos. Los diccionarios de datos generalmente son útiles cuando desarrollamos modelos de sistemas y pueden utilizarse para gestionar toda la información de todos los tipos de modelos de sistemas. Un diccionario de datos es, de forma simple, una lista de nombres ordenada alfabéticamente incluida en los modelos del sistema. El diccionario debería incluir, además del nombre, una descripción asociada de dicha entidad con nombre y, si el nombre representa un objeto compuesto, una descripción de la composición. Se puede incluir otra información, como la fecha de creación, el creador y la representación de la entidad dependiendo del tipo de modelo que se esté desarrollando. Las ventajas de usar un diccionario de datos son las siguientes:

1. Es un mecanismo para la gestión de nombres. Muchas personas pueden tener que inventar nombres para las entidades y relaciones cuando están desarrollando un modelo de un sistema grande. Estos nombres deberían ser usados de forma consistente y no deberían entrar en conflicto. El software del diccionario de datos puede comprobar la unicidad de los nombres cuando sea necesario y avisar a los analistas de requerimientos de las duplicaciones de nombres.

2. Sirve como un almacén de información de la organización. A medida que el sistema se desarrolla, la información que enlaza el análisis, diseño, implementación y evolución se añade al diccionario de datos, para que toda la información sobre una entidad esté en un mismo lugar.

Todos los nombres del sistema, tanto si son nombres de entidades, relaciones, atributos o servicios, deberían introducirse en el diccionario. El software se utiliza normalmente para crear, mantener y consultar el diccionario. Este software debería integrarse con otras herramientas para que la creación del diccionario se automatice parcialmente. Por ejemplo, las herramientas CASE que soportan el modelado del sistema incluyen soporte para el diccionario de datos e introducen los nombres en el diccionario cuando se utilizan por primera vez en el modelo.

Diagramas de Entidad – Relación (DER)

Es un modelo de red que describe con un alto nivel de abstracción la distribución de datos de un Sistema.

Fig. 04-18: Una relación

El enlace que rige la unión de las entidades está representada por la relación del modelo. En un DER, cada entidad se representa mediante un rectángulo, cada relación mediante un rombo y cada dominio (conjunto donde toma valores el atributo) mediante un círculo.

Mediante líneas se conectan las entidades con las relaciones, igual que las entidades con los dominios, representando a los atributos.

Los Atributos Llaves se representan subrayando el correspondiente conjunto de valores. En ocasiones, una entidad no puede ser identificada únicamente por el valor de sus propios atributos. En estos casos, se utilizan conjuntamente las relaciones con los atributos para lograr la requerida identificación unívoca. Estas entidades reciben el nombre de entidades débiles y se representan en el DER con un doble rectángulo.

Por qué podríamos estar interesados en modelar datos de un Sistema?

A) Porque las estructuras de datos y las relaciones pueden ser tan complejas que sea necesario examinarlas independientemente del proceso. (Ej.: Normalmente a los usuarios ejecutivos les interesa más los datos que los detalles funcionales)

B) Permite la **conversación** con el grupo de Administradores de datos.

C) Permite la comunicación con el Grupo de Administración de Bases de Datos. (Ej: Índices, punteros, claves para llegar a los datos).

D) Enfatiza las relaciones con los almacenes de los D.F.D.

Las componentes de un E-R

Fig.: 04-17: Un tipo de Objeto

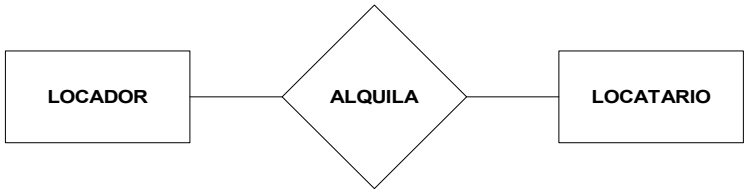


Fig. 04-18: Una relación

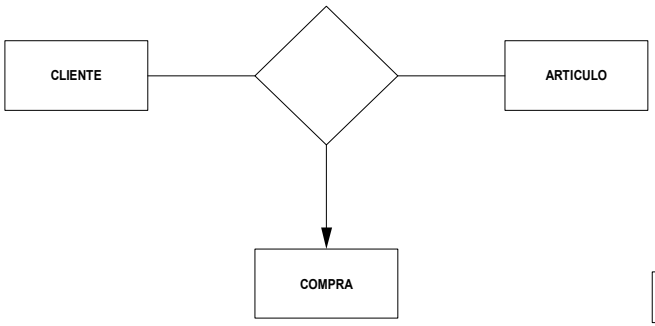


Fig. 04-21: Indicador asociativo de tipo de objeto

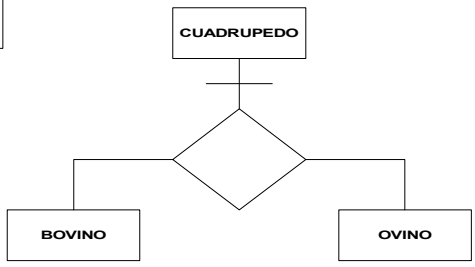
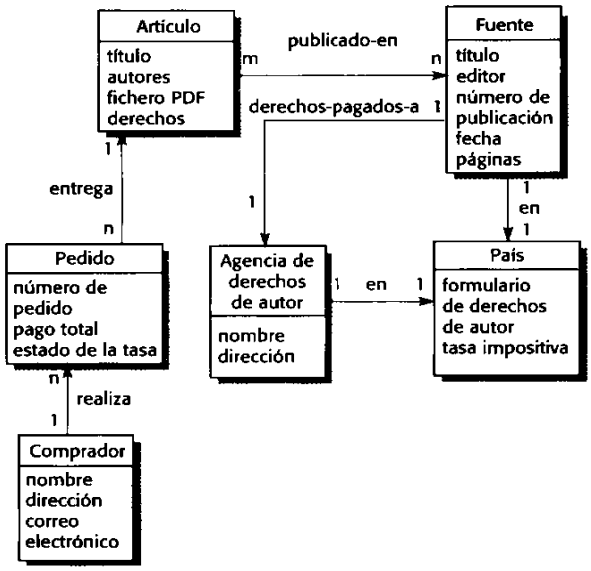


Fig. 04-22: Indicador de Subtipo/supertipo

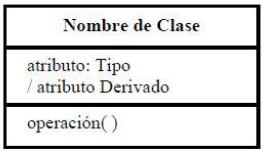
- Tipo de Objetos
- Relaciones
- Indicadores asociativos de tipo de objetos. Cuando una asociación tiene sus propios atributos y es necesario guardarlo, entonces es necesario dibujar la asociación como una entidad, y la asociación se dibuja sin ningún nombre.
- Indicadores de supertipo/subtipo

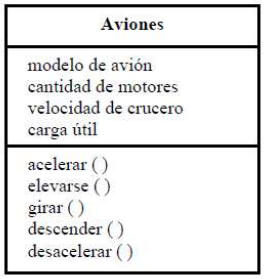
Diagrama de Clases

Los diagramas de clases describen la estructura estática de un sistema. Las cosas que existen y que nos rodean se agrupan naturalmente en categorías. Una clase es una categoría o grupo de cosas que tienen atributos (propiedades) y acciones similares. *(Ej.: "Aviones" que tiene atributos como el "modelo de avión", "la* cantidad de motores", etc. Entre las acciones de las cosas de esta clase se encuentran: "acelerar", "elevarse", "girar", "descender", "desacelerar").



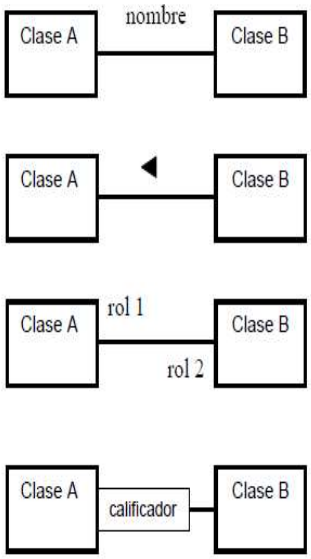
La figura anterior refiere a un sistema de pedidos en cuyo modelado (datos) se utilizó un diagrama de clases. Un rectángulo es el símbolo que representa a la clase, y se divide en tres áreas. Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que representan las asociaciones o maneras en que las clases se relacionan entre sí.





Las *clases se representan con rectángulos* divididos en tres áreas: la superior contiene el nombre de la clase, la central contiene los *atributos y la inferior las acciones.*

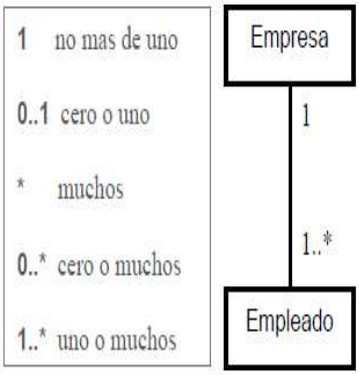
Asociaciones. ***S****on las que representan* a las relaciones estáticas entre las clases.



En la figura anterior se aprecia que el nombre de la *asociación va por sobre o* por debajo de la línea que la representa. Una flecha rellena indica la dirección de la relación. Los *roles se ubican cerca del* final de una *asociación. Los roles* representan la manera en que dos *clases* se ven entre ellas. No es común el colocar ambos nombres, el de la asociación y el de los roles a la vez. Cuando una asociación es *calificada, el* símbolo correspondiente se coloca al final de la asociación, contra la clase que hace de calificador.

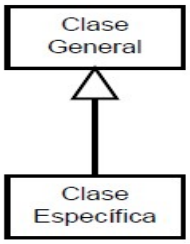
Multiplicidad

Las notaciones utilizadas para señalar la *multiplicidad se colocan cerca del final de* una *asociación. Estos símbolos indican el* número de instancias de una clase vinculadas a una de las instancias de la otra clase.



Por ejemplo, una empresa puede tener uno o más empleados, pero cada empleado trabaja para una sola empresa solamente.

Generalización (herencia). ***E****s otro nombre para herencia. Se refiere a una relación entre* dos clases en donde una Clase "Específica" es una versión especializada de la otra, o Clase "General".



El modelado orientado a objetos implica la identificación de clases de objetos que son importantes en el dominio que se está estudiando. Estos objetos se organizan a continuación en una taxonomía. Una taxonomía es un esquema de clasificación que muestra cómo una clase de objetos está relacionada con otras clases a través de atributos y servicios comunes. Para mostrar esta taxonomía, las clases se organizan en una jerarquía de herencia con las clases de objetos más generales al principio de la jerarquía. Los objetos más especializados heredan sus atributos y servicios. Estos objetos especializados pueden tener sus propios atributos y servicios.

(Ej.: Hombre es un tipo de Persona, por lo que la Clase "Persona" va a tener una relación de *generalización con la Clase* "Hombre".

En la generalización, la asociación se denota como: "Es un ..".

El diseño de jerarquías de clases no es fácil, ya que el analista necesita comprender con detalle el dominio en el que el sistema será implantado. Como ejemplo de los problemas

sutiles que surgen en la práctica, considere la jerarquía de elementos de la biblioteca. Podría parecer que el atributo Título podría situarse como el elemento más general, y a continuación ser heredado por elementos de niveles inferiores.

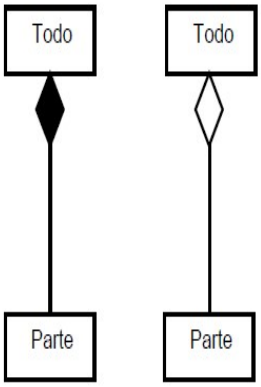
También pueden construirse modelos de herencia en los que una clase tiene varios padres. Sus atributos y servicios son una conjunción de los heredados de cada superclase. El problema principal con la herencia múltiple es el diseño de un grafo de herencia en donde los objetos no heredan atributos innecesarios. Entre otros problemas se incluye la dificultad de reorganizar el grafo de herencia cuando se requieren cambios y resolver conflictos de nombres cuando dos o más superclases tienen el mismo nombre pero diferentes significados. A nivel de modelado de sistemas, tales conflictos son relativamente fáciles de resolver alterando manualmente el modelo de objetos. Esto puede ocasionar más problemas en la programación orientada a objetos.

Agregación

Así como se adquieren atributos y servicios a través de una relación de herencia con otros objetos, algunos objetos son agrupaciones de otros objetos. Es decir, un objeto es un agregado de un conjunto de otros objetos. Las clases que representan a estos objetos pueden modelarse utilizando un modelo de agregación, tal y como se muestra en la Figura 8.13. En este ejemplo, se ha modelado un elemento de biblioteca, consistente en un paquete de estudio para un curso universitario. El paquete de estudio comprende apuntes de clase, ejercicios, soluciones ejemplo, copias de las transparencias usadas en las clases y cintas de vídeo.

La notación para la agregación consiste en representar la composición incluyendo una figura de diamante colocada sobre el elemento fuente del enlace

La *agregación es una relación en la que la* Clase “Todo” juega un *rol más importante* que la Clase "Parte", pero las dos clases no son dependientes una de otra. Se grafica con un rombo diamante vacío contra la Clase “Todo”.

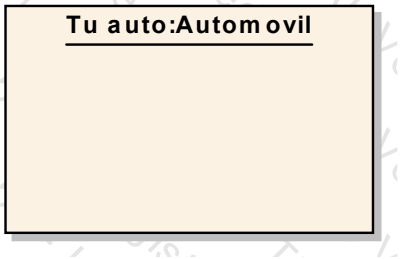


En la agregación se denota: “Es parte de …”

Composición. E*s un tipo especial de agregación que denota una fuerte* posesión de la Clase “Todo”, a la Clase “Parte”. Se grafica con un rombo diamante relleno contra la clase que representa el todo.

Diagrama de Objetos

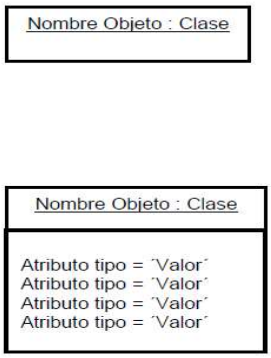
Los *Diagramas de Objetos están vinculados con los Diagramas de Clases. Un* objeto es una instancia de una clase, por lo que un *diagrama de objetos puede* ser visto como una instancia de un *diagrama de clases.*



Los diagramas de objetos describen la estructura estática de un sistema en un momento particular y son usados para probar la precisión de los diagramas de clases.

Nombre de los objetos

Cada *objeto es representado como un* rectángulo, que contiene el nombre del *objeto y* su *clase subrayadas y separadas por dos* puntos.



Atributos. Como con las *clases, los atributos se listan en* un área inferior. Sin embargo, los atributos de los *objetos deben tener un valor asignado.*

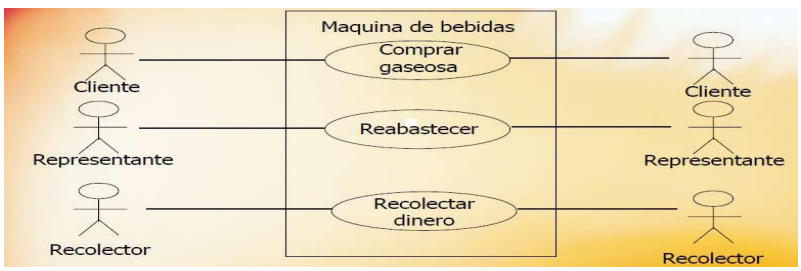
Modelado de comportamiento de objetos

Para modelar el comportamiento de los objetos, se tiene que mostrar cómo se utilizan las operaciones proporcionadas por los objetos. Se pueden modelar los comportamientos utilizando escenarios que son representados como casos de uso.

Una forma de modelar los comportamientos es utilizar diagramas de secuencia que muestran la secuencia de acciones implicadas en un caso de uso. Además de los diagramas de secuencia, también incluye diagramas de colaboración que muestran la secuencia de mensajes intercambiados por los objetos.

Diagrama de Casos de Uso (ya estudiados en el tema anterior)

Un *caso de uso es una descripción de las acciones de un sistema desde el* punto de vista del usuario. Es una herramienta valiosa dado que es una técnica de aciertos y errores para obtener los requerimientos del sistema, justamente desde el punto de vista del usuario. Los *diagramas de caso de uso modelan la funcionalidad del sistema* usando *actores y casos de uso. Los casos de uso son servicios o* funciones provistas por el sistema para sus usuarios.



Sistema. El rectángulo representa los límites del sistema que contiene los *casos de uso.* Los *actores se ubican fuera de los límites* del sistema.

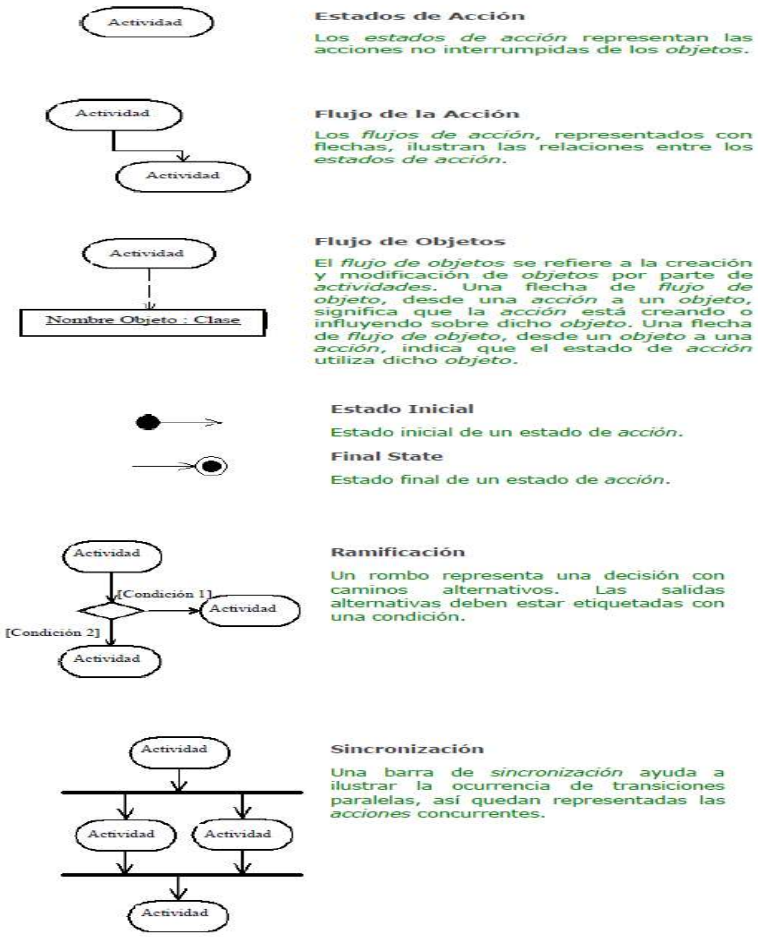
Casos de Uso. Se representan con óvalos. La etiqueta en el óvalo indica la función del sistema.

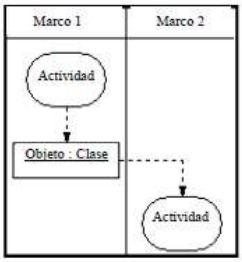
Actores. Los *actores son los usuarios de un* sistema.

Relaciones. Las relaciones entre un *actor y un caso de uso, se dibujan con una línea simple*. Para relaciones entre *casos de uso, se* utilizan flechas etiquetadas "incluir" o "extender." Una relación "incluir" indica que un *caso de uso es necesitado por* otro para poder cumplir una tarea. Una relación "extender" indica opciones alternativas para un cierto *caso de uso.*

Diagrama de Actividades

Representa el estado de la ejecución de un mecanismo bajo la forma de un desarrollo de etapas agrupadas secuencialmente en ramas paralelas de flujo de control. Está destinado a representar casos de uso (complejos). O las actividades de los distintos comportamientos de objetos (DE). El caso de uso está compuesto por actividades (las cuales tienen transiciones automáticas). Los flujos de eventos se representan por flechas punteadas. Las condiciones por un rombo.
La figura siguiente muestra los distintos componentes y sus significados





Marcos de Responsabilidad

Los *marcos de responsabilidad* agrupan a las *actividades* relacionadas en una misma columna.

La figura siguiente es un ejemplo de la utilización de un diagrama de actividades para un dispensador de bebidas (café).

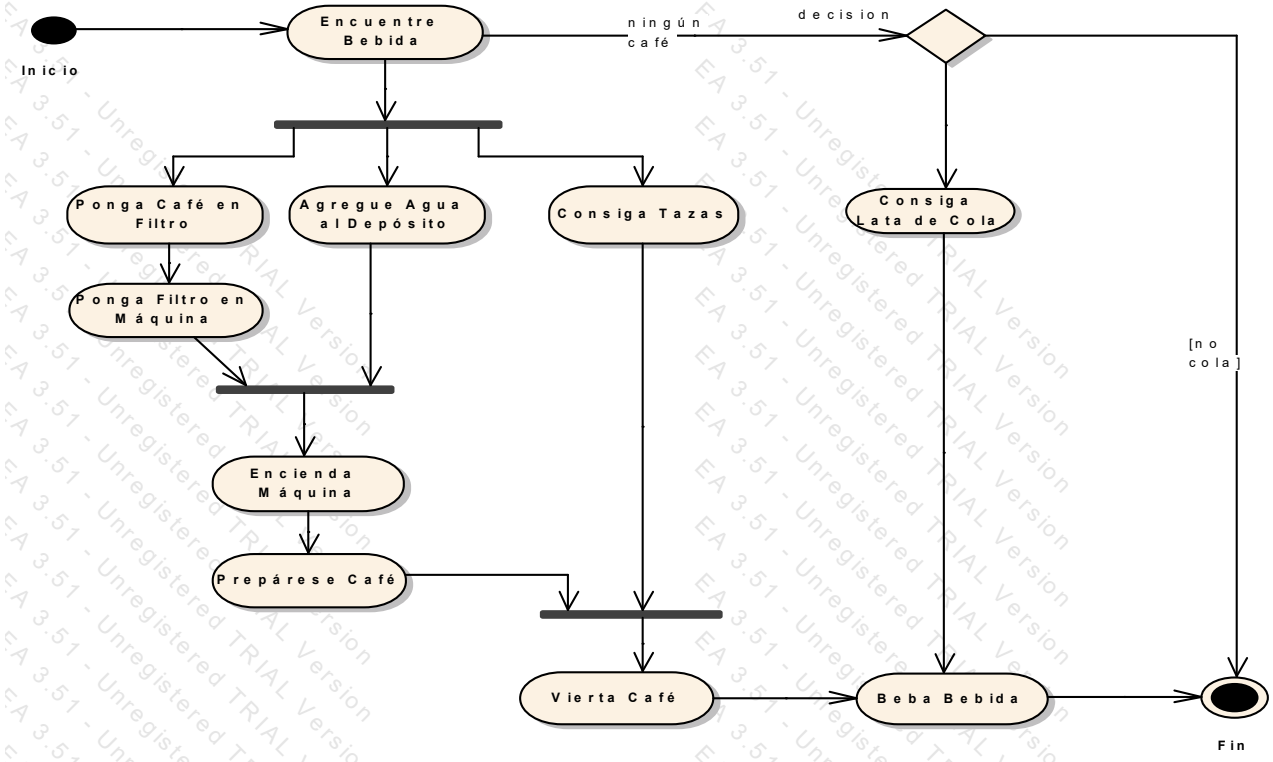
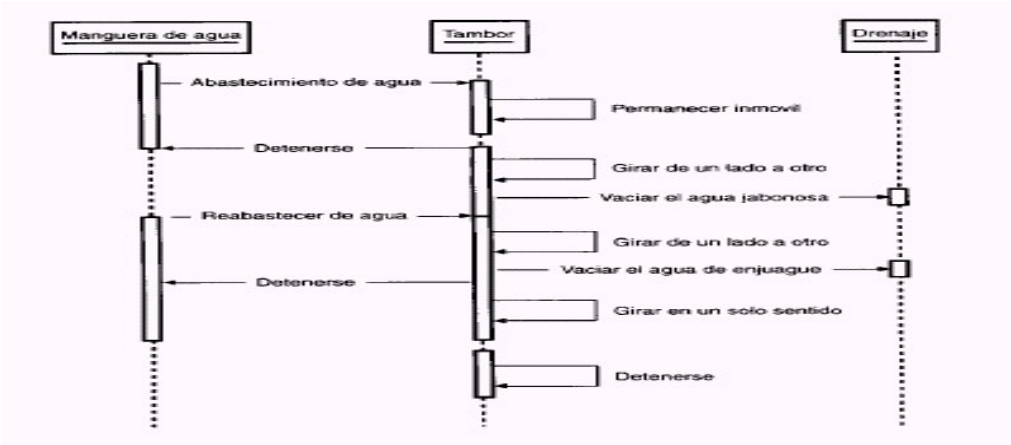


Diagrama de Secuencias

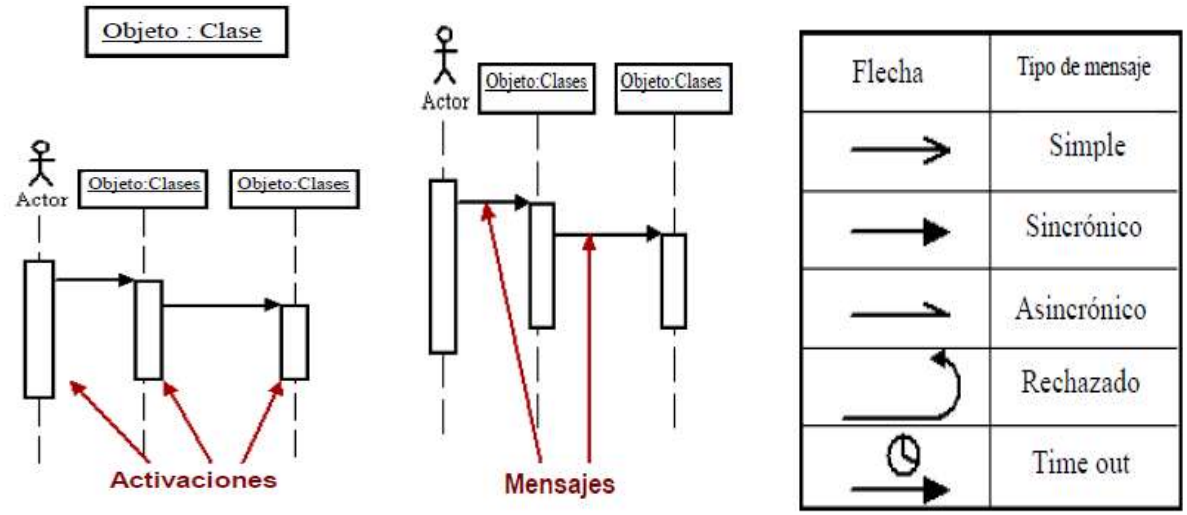
Los *diagramas de clases y los de objetos representan información* estática. No obstante, en un sistema funcional, los *objetos interactúan* entre sí, y tales interacciones suceden con el tiempo. El *diagrama de secuencias UML muestra la mecánica de la interacción con base en* tiempos.



Rol de la Clase. El *rol de la clase describe la manera en que* un *objeto se va a comportar en el contexto.* No se listan los atributos del *objeto.*

Activación. Los cuadros de *activación representan el* tiempo que un *objeto necesita para* completar una tarea.

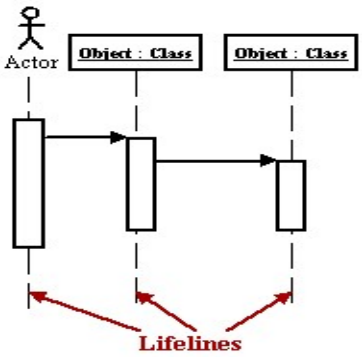
Mensajes. Los *mensajes son flechas que representan* comunicaciones entre *objetos. Las medias* flechas representan *mensajes asincrónicos.* Los *mensajes asincrónicos son enviados* desde un *objeto que no va a esperar una* respuesta del receptor para continuar con sus tareas.



Líneas de Vida. Las *líneas de vida son verticales y en línea* de puntos, ellas indican la presencia del *objeto durante el tiempo.*

Destrucción de Objetos. Los *objetos pueden ser eliminados* tempranamente usando una flecha etiquetada "<<destruir>>" que apunta a una X.

Loops. Una repetición o *loop en un diagrama de secuencias, es representado como un* rectángulo. La condición para abandonar el *loop se coloca en la parte inferior entre* corchetes [ ].



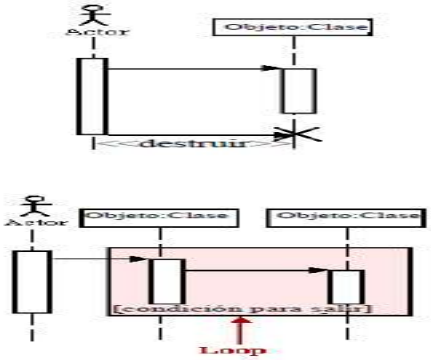


Diagrama de Colaboraciones

El diagrama de colaboraciones describe las interacciones entre objetos en terminos de mensajes secuenciados. Los diagramas de colaboracion representan una combinacion de informacion tomada de los diagramas de clases, de secuencias y de casos de uso, describiendo el comportamiento, tanto de la esctructura estatica, como de la estructura dinamica de un sistema.

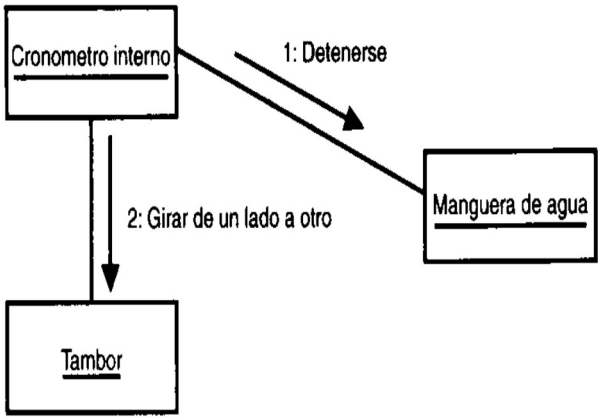


Diagrama de Colaboraciones

Un **diagrama de colaboración** es esencialmente un **diagrama** que muestra interacciones organizadas alrededor de los roles. A diferencia de los diagramas de secuencia, los diagramas de **colaboración**, también llamados diagramas de comunicación, muestran explícitamente las relaciones de los roles.

Un diagrama de colaboración muestra la misma información que un diagrama de secuencia pero de forma diferente. En los diagramas de colaboración no existe una secuencia temporal en el eje vertical; es decir, la colocación de los mensajes en el diagrama no indica cual es el orden en el que se suceden. Además, la colaboración de los objetos es más flexible y permite mostrar de forma más clara cuales son las colaboraciones entre ellos. En estos diagramas la comunicación entre objetos se denomina vínculo o enlace (link) y estará particularizada mediante los mensajes que intercambian.

Notación

Objeto

Un objeto se representa con un rectángulo dentro del que se incluye el nombre del objeto y, si se desea, el nombre de la clase, separando ambos por dos puntos.

Vinculo

En el diagrama, un vínculo se representa como una línea continua que une ambos objetos y que puede tener uno o varios mensajes asociados en ambas direcciones. Como un vínculo instancia una relación de asociación entre clases, también se puede indicar la navegabilidad del mismo mediante una flecha.

Mensaje

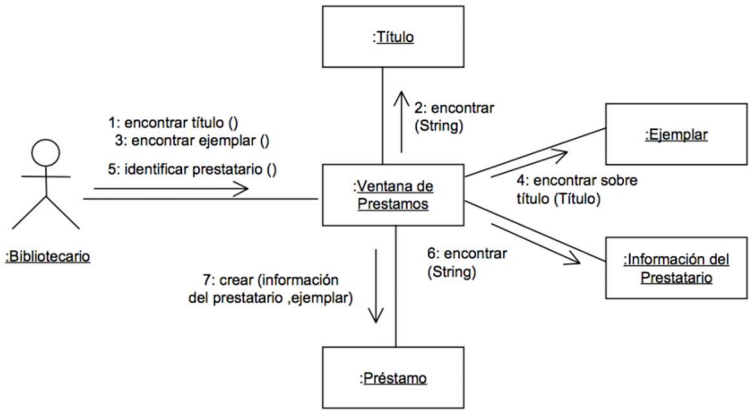
Un mensaje se representa con una pequeña flecha colocada junto a la línea del vínculo al que está asociado. La dirección de la flecha va del objeto emisor del mensaje al receptor del mismo. Junto a ella, se coloca el nombre del mensaje y sus argumentos.

A diferencia de los diagramas de secuencia, en los diagramas de colaboración siempre se muestra el número de secuencia del mensaje delante de su nombre, ya que no hay otra forma de conocer la secuencia de los mismos.

Además, los mensajes pueden tener asociadas condiciones e iteraciones que se representaran como en los diagramas de secuencia.

Ejemplo

En la siguiente figura se utiliza un diagrama de colaboración para el caso de uso: Prestar un ejemplar de una aplicación encargada de los préstamos y reservas de una biblioteca.



Otros Diagramas que se utilizan en el modelado de los sistemas.

De Componentes

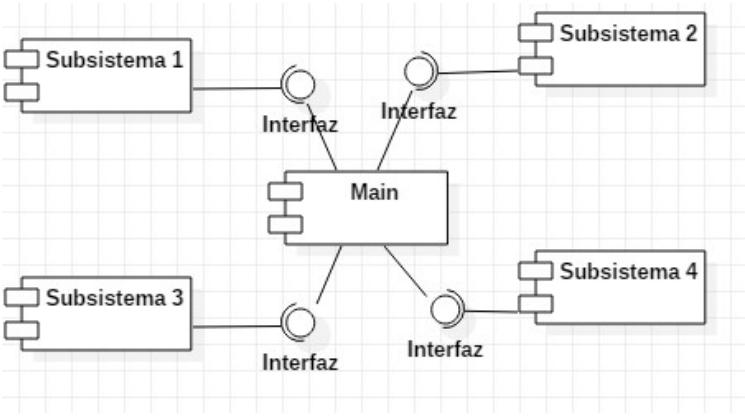
El **diagrama de componentes**. Está clasificado como diagrama de estructura y, como tal, representa de forma estática el sistema de información. Habitualmente se utiliza después de haber creado el diagrama de clases, pues necesita información de este diagrama como pueden ser las propias clases.

Este diagrama proporciona una vista de alto nivel de los componentes dentro de un sistema. Los componentes pueden ser un componente de *software*, como una base de datos o una interfaz de usuario; o un componente de *hardware* como un circuito, microchip o dispositivo; o una unidad de negocio como un proveedor, nómina o envío.

Algunos usos de este tipo de diagrama es el siguiente:

- Se utilizan en desarrollo basado en componentes para **describir sistemas con arquitectura orientada a servicios**.
- Mostrar la **estructura** del propio código.
- Se puede utilizar para centrarse en la **relación entre los componentes** mientras se ocultan los detalles de las especificaciones.
- Ayudar a **comunicar y explicar** las funciones del sistema que se está construyendo a los interesados o *stakeholders*.

Para su construcción se debe plantear en primer lugar identificar los componentes que utilizará el sistema de información, así como las distintas interfaces. Una forma típica y común para una primera aproximación en sistemas sencillos es utilizar un componente central al que los demás componentes se unen, y que se utiliza como componente gestor del sistema.



Elementos del diagrama de componentes

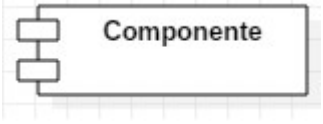
El diagrama de componentes está formado por tres elementos: **Componente, Interfaz y Relación de dependencia.**

Componente

Un componente es un bloque de unidades lógicas del sistema, una **abstracción ligeramente más alta que las clases**. Se representa como un rectángulo con un rectángulo más pequeño en la esquina superior derecha con pestañas o la palabra escrita encima del nombre del componente para ayudar a distinguirlo de una clase.

Un componente puede representar dos tipos de elementos**: componentes lógicos** (como por ejemplo componentes de negocio o proceso) o **componentes físicos** (como componentes .NET, EJB…). Por ejemplo, en una aplicación desarrollada en java habrá, con total seguridad, varios componentes ".java", que son componentes lógicos del sistema.

Es representado a través de un rectángulo que tiene, a su vez, dos rectángulos a la izquierda, tal y como se muestra en la siguiente figura:

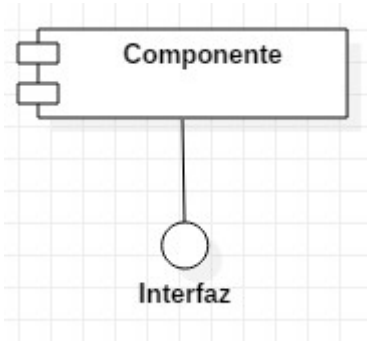


Lo ideal es que los componentes estén diseñados de forma que tengan una gran cohesión y un bajo acoplamiento, para favorecer su reutilización.

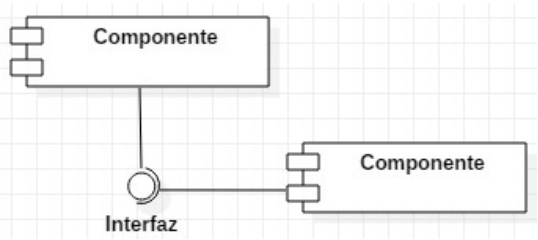
Interfaz

La interfaz está siempre asociada a un componente y se utiliza para representar la zona del módulo que es **utilizada para la comunicación** con otro de los componentes.

Se representa con una línea que tiene al final un círculo no relleno:



Otros módulos pueden conectarse a una interfaz. Esto se hace cuando un componente **requiere o utiliza** al otro componente mediante su interfaz, que son las operaciones externas que ofrece el componente. Se representa con una línea que termina en un semicírculo que rodea la interfaz del otro componente. En el diagrama se vería de la siguiente manera:

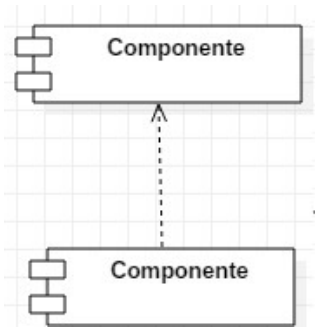


Relación de dependencia

Aunque puedes mostrar más detalles sobre la relación entre dos componentes utilizando la notación de interfaces (interfaz proporcionada y la interfaz requerida), también puedes usar una flecha de dependencia para mostrar la **relación entre dos componentes**. Es una relación más general.

La relación de dependencia representa que un componente requiere de otro para ejecutar su trabajo. Es diferente a la interfaz, pues esta identifica que un componente ofrece una serie de operaciones. En cualquier caso, en ocasiones para simplificar el diagrama no se usan las interfaces sino que solamente se utilizan relaciones de dependencia.

Una relación de dependencia se representa mediante una flecha discontinua que va desde el componente que requiere de otro componente hasta el requerido.



Las relaciones de dependencia pueden unir, además de componentes con otros componentes, componentes con interfaces.

Cómo dibujar un diagrama de componentes

Puedes utilizar un diagrama de componentes cuando quieras representar tu sistema como una colección de componentes e interfaces. Esto te ayudará a tener una idea de la futura implementación del sistema. Los siguientes son los pasos que pueden servir de guía al dibujar un diagrama de componentes.

- **Paso 1:** Determina el propósito del diagrama e identifica los artefactos como los archivos, documentos, etc. en tu sistema o aplicación que necesitas representar en su diagrama.
- **Paso 2:** A medida que descubres las relaciones entre los elementos que identificaste anteriormente, crea un diseño mental de tu diagrama de componentes.
- **Paso 3:** Al dibujar el diagrama, agrega primero los componentes, agrupándolos dentro de otros componentes como mejor te parezca.
- **Paso 4:** El siguiente paso es agregar otros elementos, como interfaces, clases, objetos, dependencias, etc. al diagrama de componentes y completarlo.
- **Paso 5:** Puede adjuntar notas en diferentes partes de su diagrama de componentes para aclarar ciertos detalles a otros usuarios.

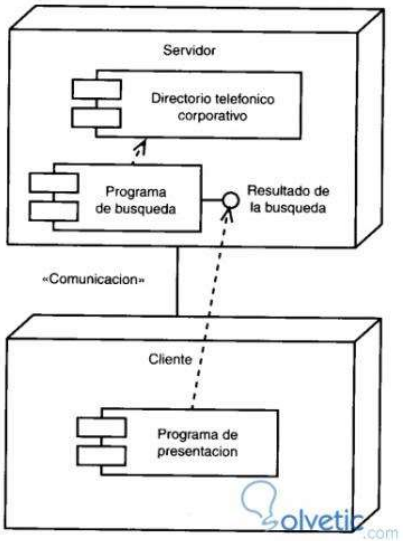
De Distribución

En el **diagrama de distribución** es donde representamos la *estructura de hardware* donde estará nuestro sistema o software, para ello cada componente lo podemos representar como **nodos**, el nodo es cualquier elemento que sea un recurso de hardware, es decir, es nuestra denominación genérica para nuestros equipos.

Dentro de la clasificación de los nodos tenemos que hay el nodo que puede ejecutar o procesar y el nodo que no ejecuta ni procesa, estos últimos pueden ser los dispositivos de salida como impresoras o monitores, es decir, los que están en contacto con el exterior.

Para **representar al nodo utilizaremos la figura del cubo**, dentro de nuestro cubo podemos escribir la información correspondiente al nodo, por ejemplo su nombre, veamos a continuación un nodo básico:

Ahora si necesitamos indicar información adicional de nuestro nodo, por ejemplo si pertenece a un paquete o tal vez los componentes que posee, podemos dividir el cubo en diferentes secciones donde iremos agregando la información representada en texto, observe la siguiente figura:



Incorporar Relaciones entre Nodos

Por supuesto que un nodo no es un ente aislado en un sistema, para ello en nuestro diagrama podremos representar como se asocia o se relaciona con otros nodos, a través de sus componentes o interfaces, con ello podemos dar una representación más exacta. Usualmente utilizamos la conexión como la forma de representación, sin embargo podemos utilizar la asociación y la agregación en nuestros nodos, para *representar la conexión utilizamos una línea discontinua*, donde vinculamos un componente de un nodo a otro componente de otro nodo, esta conexión no necesariamente es un cable, esta conexión también puede representar conexiones inalámbricas, como Wi-Fi, Bluetooth, infrarrojos, etc.

Como podemos notar en la figura anterior que es la representación de la distribución de un sistema cliente – servidor, que sucede en este diagrama, en el nodo cliente tenemos un componente de programa de presentación, dicho programa probablemente nos debe mostrar los recursos del servidor a los que podemos tener acceso, como por ejemplo el programa de búsqueda, al utilizar dicho programa, ocurre una conexión con el servidor quien procesa y hace un procesamiento de los datos y entonces retorna un resultado para dicha búsqueda.

Nuestro diagrama de nodos entonces nos da una representación bastante acertada de la estructura de los equipos que intervienen, ya que el cliente puede ser un terminal cualquier con dispositivos de entrada y salida de datos y el servidor es un equipo diferente que procesa los datos.

Herramientas para Modelado de Software

El modelado de software es el primer paso antes de desarrollar cualquier tipo de sistema. Normalmente, **el modelado se basa en la creación de Diagramas que explican el funcionamiento del software**. Para eso se usan, entre otros, los diagramas UML, los cuales permiten que los desarrolladores definan sistema, funcionamiento y funcionalidades.

Como una buena caja de herramientas, una buena herramienta de modelado ofrece todas las herramientas necesarias para conseguir hacer eficientemente varios trabajos, sin dejarte nunca sin la herramienta correcta.

Se espera que una buena herramienta para modelado (usar distintos diagramas) incluya lo siguiente:

- Soporte para toda la notación y semántica.
- Soporte para una cantidad considerable de técnicas de modelado y diagramas para complementar, diagramas de flujo, y diseño de pantallas de usuario. Posibilidad de reutilizar información obtenida por otras técnicas todavía usadas, como modelado tradicional de procesos.
- Facilitar la captura de información en un repositorio subyacente - permitiendo la reutilización entre diagramas.
- Posibilidad de personalizar las propiedades de definición de elementos subyacentes de modelos.
- Permitir a varios equipos de analistas (trabajo colaborativo) trabajar en los mismos datos a la vez.
- Posibilidad de capturar los requisitos, asociarlos con elementos de modelado que los satisfagan y localizar cómo han sido satisfechos los requisitos en cada uno de los pasos del desarrollo.
- Posibilitar la creación de informes y documentación personalizados en tus diseños, y la salida de estos informes en varios formatos, incluyendo HTML para la distribución en la Internet o Intranet local.

- Posibilidad para generar y 'reverse' código (por ejemplo C++, Java, etc.) para facilitar el análisis y diseño 'iterativo', para volver a usar código o librerías de clase existentes, y para documentar el código.

Las herramientas CASE, están tomando cada vez más relevancia en la planeación y ejecución de proyectos que involucren sistemas de información, pues suelen inducir a sus usuarios a la correcta utilización de metodologías que le ayudan a llegar con facilidad a los productos de software construidos. Todas las herramientas CASE prestan soporte a un lenguaje de modelado para acompañar la metodología y es lógico suponer, que un alto porcentaje de ellas soportan UML, teniendo en cuenta la amplia aceptación de este lenguaje y el valor conceptual y visual que proporciona, y su facilidad para extender el lenguaje para representar elementos particulares a determinados tipos de aplicaciones

Evaluación de herramientas de modelado con UML

La toma de una acertada decisión a la hora de escoger una herramienta de modelado, puede ser un factor importante para lograr la calidad de un proyecto. Por tal razón, se necesita tener un amplio conocimiento en aspectos como:

• Las herramientas que existen en el mercado, tanto comercial como libre.

• Las características de una buena herramienta de modelado.

• La manera como las herramientas satisfacen las necesidades de las personas que participan en un proyecto, para apoyar el proceso de Ingeniería de Software.

Para facilitar el entendimiento de los criterios a tener en cuenta al momento de evaluar una herramienta de modelado, es necesario analizar cuatro aspectos en los que estas herramientas deben satisfacer resultados.

1.- Enfoque Procedimental El enfoque procedimental se refiere a la forma como las herramientas hacen uso de las metodologías para guiar al usuario a través de un proceso de Ingeniería de Software.

2.- Apoyo Metodológico Una herramienta de modelado, debe apoyar el uso de una o varias de las metodologías para el desarrollo de sistemas de información; es importante que dicha herramienta garantice la unicidad y coherencia ya sea entre los diferentes modelos del sistema que representan vistas complementarias del sistema (estructurales, dinámicos, funcionales, etc.) o entre modelos que representan diferentes niveles de abstracción (de negocio, de análisis, de diseño, etc.). Esto permite guiar al usuario a través de etapas o fases permitiendo una trazabilidad entre diagramas de alto nivel que representan aspectos del dominio del problema y diagramas detallados que representan detalles de implementación. Para conseguir tal propósito, es importante que el navegador de la herramienta, muestre las etapas de la metodología o los modelos que propone la misma para cada etapa, fase o arquitectura, y de ser necesario, que le ayude al usuario a evolucionar o refinar un modelo al momento de pasar de una etapa a otra o de un aspecto o contexto arquitectónico a otro.

3.- Soporte completo UML Esta característica se refiere a la capacidad de las herramientas de construir todos los diagramas que propone UML, o por lo menos los más relevantes.

4.- Facilidad de extensión del lenguaje. Se presentan condiciones particulares de los ámbitos de desarrollo que no logran ser satisfechas.

Algunas de las herramientas de modelado de software son los Diagramas UML Online de Lucidchart, la Herramienta UML de Altova y MagicDraw, System Architect 2001, Enterprice Architect, Easy Case, ArgoUML, Rational, AR2CA, entre otras.

Bibliografía

Plantillas de Clase. Mgter Vallejos, Oscar A.
Ian Sommerville. Ingeniería del Software. 7ma.Edición. Pearson Educación. 2005.
Roger S. Pressman Ingeniería del Software. 7ma. Edición. McGraw Hill. 2010.
Craig Larman. UML y Patrones. 2da. Edición. Prentice Hall. 2003.
Aprendiendo UML en 24 Hs. 1ra Edicion. Joseph Schmuller. Editorial: Prentice Hall.