INSTITUTO TECNOLOGICO DE MORELIA

DEPARTAMENTO DE SISTEMAS Y COMPUTACION

INGENIERÍA EN SISTEMAS COMPUTACIONALES

METODOLOGÍA OMT

Elaborado por:

Víctor Manuel Chávez Gaona #Control: 99120839

Juan Carlos Olivares Rojas #Control: 99120871

Morelia, Michoacán a 12 de Marzo de 2002

METODOLOGIA OMT (RUMBAUGH)

Introducción.

La metodología OMT (Object Modeling Technique) fue creada por James Rumbaugh y Michael Blaha en 1991, mientras James dirigía un equipo de investigación de los laboratorios General Electric.

OMT es una de las metodologías de análisis y diseño orientadas a objetos, más maduras y eficientes que existen en la actualidad. La gran virtud que aporta esta metodología es su carácter de abierta (no propietaria), que le permite ser de dominio público y , en consecuencia, sobrevivir con enorme vitalidad. Esto facilita su evolución para acoplarse a todas las necesidades actuales y futuras de la ingeniería de software.

Las fases que conforman a la metodología OMT son:

- ✓ **Análisis**. El analista construye un modelo del dominio del problema, mostrando sus propiedades más importantes. El modelo de análisis es una abstracción resumida y precisa de lo que debe de hacer el sistema deseado y no de la forma en que se hará. Los elementos del modelo deben ser conceptos del dominio de aplicación y no conceptos informáticos tales como estructuras de datos. Un buen modelo debe poder ser entendido y criticado por expertos en el dominio del problema que no tengan conocimientos informáticos.
- ✓ **Diseño del sistema**. El diseñador del sistema toma decisiones de alto nivel sobre la arquitectura del mismo. Durante esta fase el sistema se organiza en subsistemas basándose tanto en la estructura del análisis como en la arquitectura propuesta. Se selecciona una estrategia para afrontar el problema.
- ✓ **Diseño de objetos**. El diseñador de objetos construye un modelo de diseño basándose en el modelo de análisis, pero incorporando detalles de implementación. El diseño de objetos se centra en las estructuras de datos y algoritmos que son necesarios para implementar cada clase. OMT describe la forma en que el diseño puede ser implementado en distintos lenguajes (orientados y no orientados a objetos, bases de datos, etc.).
- ✓ **Implementación**. Las clases de objetos y relaciones desarrolladas durante el análisis de objetos se traducen finalmente a una implementación concreta. Durante la fase de implementación es importante tener en cuenta los principios de la ingeniería del software de forma que la correspondencia con el diseño sea directa y el sistema implementado sea flexible y extensible. No tiene sentido que utilicemos AOO y DOO de forma que potenciemos la reutilización de código y la correspondencia entre el dominio del problema y el sistema informático, si luego perdemos todas estas ventajas con una implementación de mala calidad.

La metodología OMT emplea tres clases de modelos para describir el sistema:

- ✓ **Modelo de objetos.** Describe la estructura estática de los objetos del sistema (identidad, relaciones con otros objetos, atributos y operaciones). El modelo de objetos proporciona el entorno esencial en el cual se pueden situar el modelo dinámico y el modelo funcional. El objetivo es capturar aquellos conceptos del mundo real que sean importantes para la aplicación. Se representa mediante diagramas de onjetos.
- ✓ **Modelo dinámico.** Describe los aspectos de un sistema que tratan de la temporización y secuencia de operaciones (sucesos que marcan los cambios, secuencias de sucesos, estados que definen el contexto para los sucesos) y la organización de sucesos y estados. Captura el control, aquel aspecto de un sistema que describe las secuencias de operaciones que se producen sin tener en cuenta lo que hagan las operaciones, aquello a lo que afecten o la forma en que están implementadas. Se representa gráficamente mediante diagramas de estado.
- ✓ **Modelo funcional.** Describe las transformaciones de valores de datos (funciones, correspondencias, restricciones y dependencias funcionales) que ocurren dentro del sistema. Captura lo que hace el sistema, independientemente de cuando se haga o de la forma en que se haga. Se representa mediante diagramas de flujo de datos

Modelo de Objetos

Esta es la parte principal de la Técnica para modelado ya que se fundamenta en la teoría de OO. La definición clara de las entidades que intervienen en el sistema es un paso inicial necesario para poder definir qué transformaciones ocurren en ellas y cuándo se producen estas transformaciones. Esta forma de pensar es inherente al paradigma de OO donde las clases y su jerarquía determinan el sistema.

Los diagramas de objetos permiten representar gráficamente los objetos, las clases y sus relaciones mediante dos tipos de diagramas: los diagramas de clases y los diagramas de casos concretos (instancias). Los diagramas de clases describen las clases que componen el sistema y que permitirán la creación de casos concretos, los diagramas de casos concretos describen la manera en que los objetos del sistema se relacionan y los casos concretos que existen en el sistema de cada clase. En los diagramas que componen este modelo se pueden representar los siguientes elementos del sistema: objetos y clases, atributos, operaciones, y relaciones o asociaciones.

Clases y Objetos

Los objetos y sus componentes se representan gráficamente en OMT de forma que es posible obtener una idea de los elementos que intervienen en el sistema estudiando el modelo. Los elementos y sus características con representación gráfica son los siguientes:

- ✓ Objetos. Un objeto es, sencillamente, algo que tiene sentido en el contexto de la aplicación. Se definirá un objeto como un concepto, abstracción o cosa con límites bien definidos y con significado a efectos del problema que se tenga entre manos.
- ✓ Clases. Describe un grupo de objetos con propiedades (atributos) similares, con relaciones comunes con otros y con una semántica común.
- ✓ Diagramas de objetos. Proporcionan un anotación gráfica formal para el modelado de objetos, clases y sus relaciones entre sí, son útiles, tanto para el modelado abstracto como, para diseñar programas reales. Hay dos tipos de diagramas de objetos
  - o Diagrama de clases. Esquema, patrón o plantilla para describir muchas instancias de datos posibles.
  - o Diagrama de instancias. Describe la forma en que un cierto conjunto de objetos se relacionan entre sí.
- ✓ Atributos. Los objetos pertenecientes a una clase presentan características que en OMT se denominan atributos. Sin embargo, no se deben de confundir los atributos, que son características que todos los objetos de una clase comparten, con otros objetos que pueden formar parte del objeto que estamos tratando.
- ✓ Operaciones y métodos. Del mismo modo que los objetos en OMT se pueden representar las operaciones que se realizan sobre ellos o que éstos realizan sobre otros objetos del sistema. Los objetos realizan acciones sobre otros objetos y definen acciones que se realizan sobre ellos mismos. Los objetos de una misma clase comparten estas operaciones, aunque también pueden añadir otras nuevas que no se definan en su clase a medida que se especializa el objeto en otras subclases. También pueden redefinir las operaciones en estas especializaciones ignorando las definiciones realizadas en las superclases. Las operaciones pueden llevar implícito el objeto sobre el que se realizan o que realiza la acción, de forma que es posible tener una misma operación que se efectúe de manera distinta según el objeto sobre el que se aplique. La implementación de las operaciones para cada uno de los objetos diferentes (o subclases) se denomina método. Los métodos implementan en cada una de las clases de forma específica para los objetos que representa.

Enlaces y Asociaciones

Las relaciones entre clases determinan el comportamiento del sistema y constituyen una parte muy importante del mismo ya que mediante las relaciones definimos la forma en que los objetos se comunican, lo que también se conoce como comportamiento.

Además las relaciones tienen una serie de características que son de interés para el modelado del sistema.

- ✓ Relaciones. En OMT se identifican a través de enlaces: conexiones físicas o conceptuales entre casos concretos de objetos. Una asociación en OMT

abstrae un conjunto de enlaces con una estructura y un significado comunes. Desde el punto de vista de la implementación, una asociación es un puntero que apunta desde un objeto a otro.

- ✓ Multiplicidad. Este término se encuentra relacionado con las asociaciones e indica el número de casos concretos de una clase que puede relacionarse con otro caso concreto. Las relaciones más frecuentes son las binarias, aunque pueden darse de cualquier orden.

Conceptos Avanzados de Enlaces y Asociaciones

- ✓ Atributos de los enlaces. Los enlaces así como los objetos pueden tener atributos, que son propiedades de los enlaces de una asociación.
- ✓ Modelado de una asociación en forma de clase. A veces resulta conveniente modelar las asociaciones como clases en lugar de cómo relaciones, cuando los enlaces pueden participar en asociaciones con otros objetos o están sometidos a operaciones.
- ✓ Clasificación. También podemos encontrar en una asociación de objetos que pertenecen a una clase con multiplicidad “muchos” que deben estar ordenados. Esta característica de los objetos es una restricción, ya que implica una condición que deben cumplir los elementos de la clase.
- ✓ Nombre de rol. Las asociaciones conectan clases u objetos que pertenecen a dichas clases, pero en ocasiones necesitamos restringir el conjunto de los objetos que se relacionan dentro de la clase. Mediante estas restricciones podemos especificar qué objetos se relacionan. Podemos conseguir este objetivo mediante los llamados nombres de rol. Un nombre de rol es un atributo derivado de una clase cuyo valor es un conjunto de objetos relacionados. Los nombres de rol se utilizan cuando las asociaciones se producen entre objetos de la misma clase ya que suelen producirse entre subconjuntos de esas clases. Una asociación binaria puede tener dos roles, uno por cada extremo de la asociación que son identificados por sus nombres.
- ✓ Cualificación. Las asociaciones muchos a muchos y uno a muchos pueden ser calificadas mediante un elemento calificador que reduce el conjunto de objetos relacionados, indicando un subconjunto de la clase que se califica en la asociación. Las asociaciones que se pueden calificar son las de multiplicidad uno a muchos y muchos a muchos.
- ✓ Agregación. Las relaciones de agregación son para OMT formas de asociación del tipo “es parte de”, como tales se definen entre una clase agregado y una clase componente y se indican con un rombo en la parte de la clase que actúa como continente. Las relaciones de agregación se establecen en los llamados objetos compuestos que contienen otros objetos y éstos pueden ser de dos tipos: aquellos que tienen existencia física más allá del objeto agregado, y los que no pueden existir sin el objeto agregado.

Generalización y Herencia

En el paradigma de la orientación a objetos uno de los elementos más importantes es la herencia. La cualidad que permite que los objetos hereden las características (atributos) y las operaciones (métodos) dentro de una estructura jerárquica conlleva una serie de consecuencias de máxima relevancia a la hora de diseñar un sistema informático. Los objetos heredan un comportamiento que puede ser modificado y unas estructuras de datos de forma que se permite y se facilita la reutilización de las clases y del código que implementa sus funcionalidades.

Ambos conceptos van unidos: herencia y estructura jerárquica, de forma que la herencia se produce por la existencia de una estructura entre los componentes del sistema y la estructura se consigue en la implementación del código a través de la herencia en los lenguajes OO.

La herencia está íntimamente relacionada con la forma concreta en que un lenguaje implementa la generalización, que es un término más abstracto.

- ✓ La generalización es la relación que existe entre una clase y las subclases que se derivan de la misma. A partir de una versión “en bruto” se generan versiones más especializadas de la misma que añaden características y operaciones. La superclase es la versión general y las subclases son especializaciones de la superclase. Las generalizaciones pueden tener discriminadores que indican qué aspecto de la superclase está siendo utilizado para obtener subclases más concretas.
- ✓ Anulación. Al implementar la herencia nos encontramos en numerosas ocasiones que las subclases redefinen operaciones que ya han sido definidas en las superclases. Las razones para esta nueva implementación de operaciones que existen en las superclases son variadas, a veces simplemente se le añaden nuevas acciones que van en consonancia con las nuevas características que añade la subclase; otras, se consigue optimizar las operaciones debido a que las subclases tienen características nuevas que lo permiten, y a veces se produce por una mala práctica de análisis donde no se prevén las operaciones de manera óptima.

Se han propuesto una serie de reglas a la hora de implementar la herencia para minimizar los errores y maximizar la reutilización de código:

a. Las operaciones de consulta, aquellas que no modifican valores de atributos, se heredan por todas las subclases.
b. Las operaciones de actualización, que modifican valores de atributos, se heredan por todas las subclases y se añaden las nuevas operaciones para aquellas que añadan atributos.
c. Las operaciones de actualización que se realizan sobre atributos que tengan algún tipo de restricción o asociación, se bloquean para nuevas subclases.
d. Las operaciones no pueden volver a definirse para hacer que se comporten de distinta manera de cara al exterior, es decir; todos los

métodos concretos de una operación deben tener el mismo protocolo.

e. Las operaciones se pueden refinar añadiendo comportamientos en las subclases.

Agrupación de entidades

Los elementos que hemos estudiado en el Modelo de Objetos se pueden agrupar para construir el modelo completo, así, las clases, las asociaciones y las generalizaciones forman lo que se denomina módulo y varios módulos forman el modelo de objetos. En un módulo no se deben repetir los nombres de las clases y de las asociaciones, aunque se puede hacer referencia a la misma clase dentro de distintos módulos. También se definen las denominadas hojas que se utilizan para descompones un Modelo de Objetos en unidades que podemos manejar. Una hoja es una parte de un módulo que podemos manejar con facilidad, sea en el formato que sea.

Modelo Avanzado de Datos

- ✓ Clases abstractas. En ocasiones puede ser de utilidad tener clases que definan propiedades y operaciones de forma general, dejando para sucesivos refinamientos la implementación concreta de las mismas. Una clase abstracta es precisamente una clase donde se introducen métodos y datos que se definirán en las subclases de la misma. Las clases abstractas siempre tienen que tener clases derivadas donde se especificarán las operaciones que no se hayan definido en la clase abstracta, nunca podrán tener objetos, se utilizan como clases bases o superclases de otras clases. La existencia de este tipo de clases facilita aún más la posibilidad de abstracción del modelo, dejando para posteriores refinamientos del problema la definición más exhaustiva de operaciones y datos. La implementación concreta de las clases abstractas depende del lenguaje OO. En OMT una clase abstracta se identifica cuando no tiene casos concretos (instancias) pero una subclase suya sí los tiene. Por ejemplo, una clase abstracta números podría definir una operación multiplicar que no se implementara hasta la definición de las subclases números enteros, números reales y números complejos, teniendo la información precisa en cada una de las subclases de la forma en que se multiplican los números.
- ✓ Herencia múltiple. La herencia múltiple es una característica que algunos sistemas OO poseen, mediante la cual es posible que una clase herede de varias superclases al mismo tiempo. Sin embargo, la herencia múltiple aumenta radicalmente la complejidad de los sistemas que la implementan ya que la búsqueda de las operaciones se dificulta cuando no se define en la clase derivada y hay que realizarla en las superclases.
- ✓ Clave candidata. No es más que un conjunto mínimo de atributos que define de forma única un objeto o enlace. Es decir, mediante una clave candidata tenemos definido el objeto o el enlace con una serie de atributos de forma que se distingue del resto de objetos o enlaces. Las claves

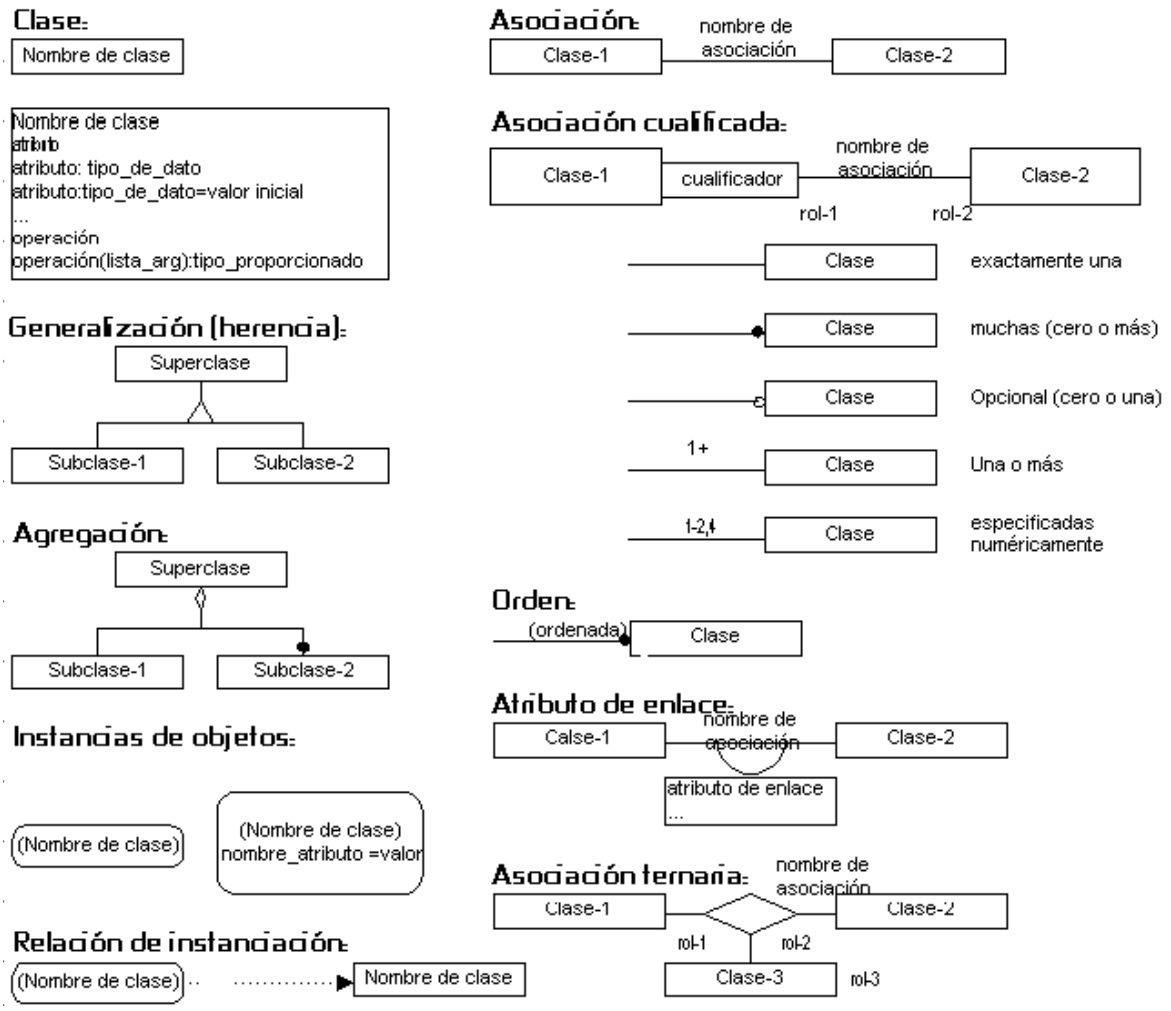
candidatas son restricciones y, por tanto, se representan como tales en OMT. Para encontrar una clave candidata en una asociación donde intervienen más de dos clases, debemos definir qué combinaciones de clases o enlaces en la asociación de los elementos definen la tercera clase o enlace de forma única.

- ✓ Restricciones. El modelo de objetos contiene diferentes entidades como son los objetos, las clases, los atributos, los enlaces y las asociaciones. Cada una de estas entidades tiene una serie de características inherentes a su naturaleza dentro del sistema que estamos modelando. Las características definen su significado dentro del sistema y también sus limitaciones. Estas limitaciones se denominan en el Análisis restricciones. Las restricciones pueden ser muy complejas y en este caso no pueden estudiarse en el modelo de objetos, sino que se especificarán en el modelo funcional. Las que intervienen en el modelo de objetos se representan mediante llaves junto a la entidad a que se refieran. Cuando la restricción implica más de una clase, se indica mediante una flecha discontinua que une los elementos que se vean implicados en la restricción. Las asociaciones también pueden tener restricciones.

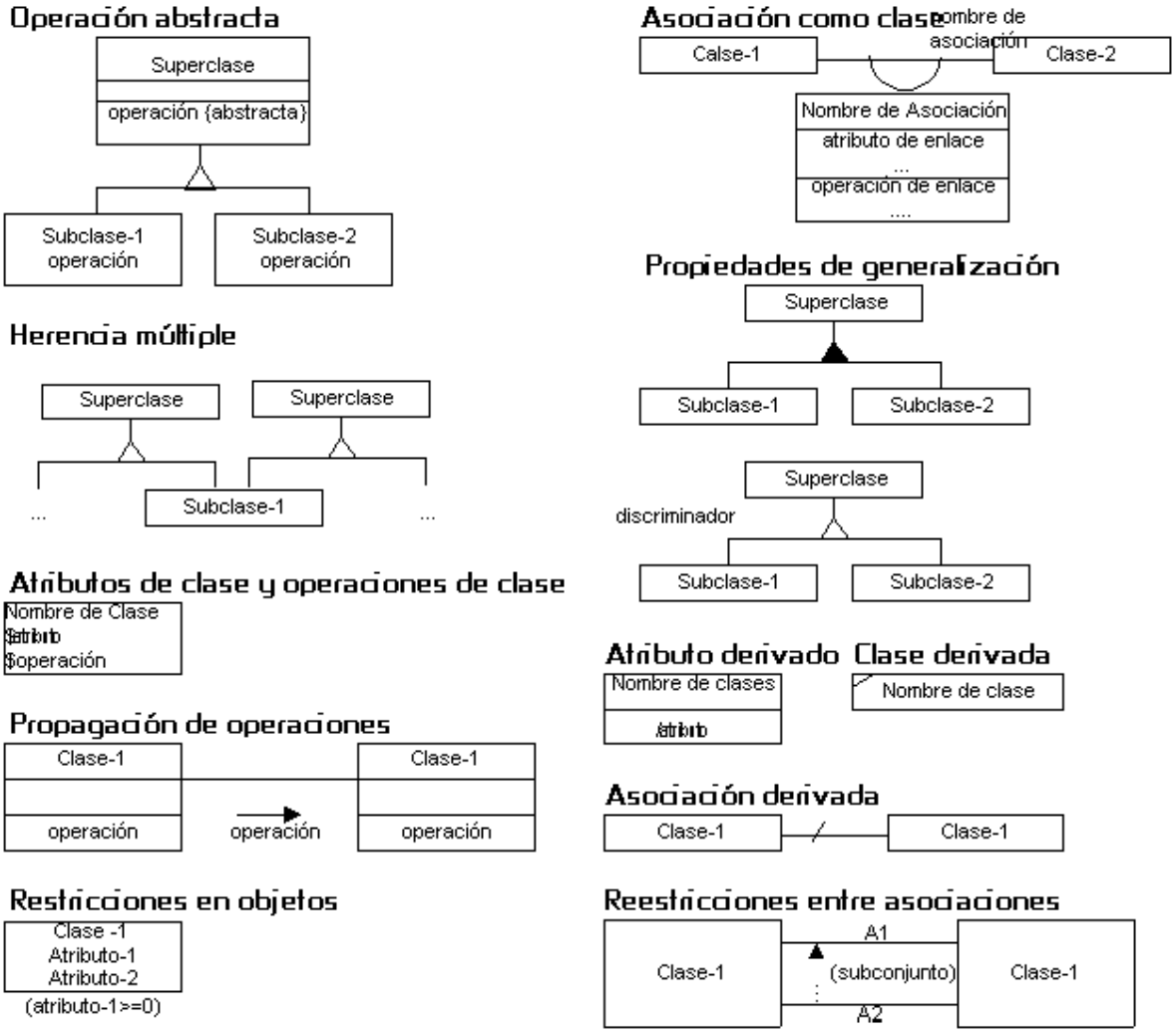
Construcción de un modelo de objetos

- Identificar las clases de objetos.
- Iniciar un diccionario de datos que contenga descripciones de clases, atributos y asociaciones.
- Agregar asociaciones entre clases.
- Agregar atributos a objetos y ligas.
- Organizar y simplificar las clases de objetos usando herencia.
- Probar las rutas de acceso usando escenarios e iterar los pasos anteriores según sea necesario.
- Agrupar las clases en módulos, basándose en “acoplamiento cercano” y función relacionada.

Notaciones del modelo de objetos



Notaciones del modelo avanzado de objetos



Modelo Dinámico

Los aspectos del sistema que están relacionados con el tiempo y con los cambios constituyen el modelo dinámico.

Los conceptos más importantes del modelado dinámico son los sucesos, que representan estímulos externos, y los estados, que representan los valores de los objetos. El diagrama de estados va a representar los sucesos y los estados que se dan en el sistema.

El modelo de objetos describe las posibles tramas de objetos, atributos y enlaces que pueden existir en un sistema. Los valores de los atributos y de los enlaces mantenidos por un objeto son lo que se denomina su estado. A lo largo del tiempo, los objetos se estimulan unos a otros, dando lugar a una serie de cambios en sus estados. Un estímulo individual proveniente de un objeto y que llega a otro es un suceso. La respuesta a un suceso depende del estado del objeto que lo recibe, y puede incluir un cambio de estado o el envío de otro suceso al remitente o a un

tercer objeto. La trama de sucesos, estados y transiciones de estados para una clase dada se puede abstraer y representar en forma de un diagrama de estados. El modelo dinámico consta de múltiples diagramas de estados, con un diagrama de estados para cada clase que posea un comportamiento dinámico importante, y muestra la trama de actividad para todo el sistema.

Sucesos y Estados

- ✓ Suceso. Un suceso, o evento, es algo que transcurre durante un período de tiempo. Un suceso puede preceder o seguir lógicamente a otro, o bien los dos sucesos pueden no estar relacionados. Se dice que dos sucesos que no tienen relación causal son concurrentes; no tienen efecto entre sí. Un suceso es una transmisión de información de dirección única entre un objeto y otro. No es como una llamada a subrutina, que proporciona un valor. En el modelo dinámico existe el concepto de clases de sucesos, que consiste en una estructura jerárquica, similar a la estructura de clases. Todo suceso aporta información de un objeto a otro. Los valores de datos aportados por un suceso son sus atributos.
- ✓ Escenarios y seguimiento de sucesos. Un escenario es una secuencia de sucesos que se produce durante una ejecución concreta de un sistema. El ámbito de un escenario es variable; puede incluir todos los sucesos del sistema, o que sean generados por ellos. Todo suceso transmite información de un objeto a otro. El primer paso para la construcción de escenario es identificar los objetos emisores y receptores de cada suceso. La secuencia de sucesos y los objetos que intercambian sucesos se pueden mostrar ambos en un escenario mejorado que se denomina de seguimiento traza de sucesos. El tiempo aumenta desde arriba hacia abajo, aunque en el seguimiento de sucesos no importa la temporización exacta, sino que importa la secuencia de los procesos.
- ✓ Estados. Un estado es una abstracción de los valores de los atributos y de los enlaces de un objeto. Los conjuntos de valores se agrupan dentro del estado de acuerdo con aquellas propiedades que afecten al comportamiento del objeto. Un estado especifica la respuesta del objeto a los sucesos entrantes. La respuesta a un suceso recibido por un objeto puede variar cuantitativamente, dependiendo de los valores exactos de sus atributos, pero cualitativamente la respuesta es la misma para todos los valores dentro del mismo estado, y puede ser distinta para valores de distintos estados. La respuesta de un objeto a un suceso puede incluir una acción o un cambio de estado por parte del objeto. Un estado corresponde al intervalo entre dos sucesos recibidos por un objeto. Los sucesos representan puntos temporales; los estados representan intervalos de tiempo. Los estados suelen estar asociados con que el valor de un objeto satisfaga alguna condición. Al definir estados, ignoramos aquellos atributos que no afectan al comportamiento del objeto, y agrupamos en un único estado todas las combinaciones de valores de atributos y de enlaces que tienen una misma respuesta a los sucesos. Por supuesto, todo atributo tiene algún efecto sobre el comportamiento, o no tendría sentido, pero es

frecuente que algunos atributos no afecten a la trama de control, y que se pueda pensar en ellos como valores de parámetros dentro de un estado. Tanto los sucesos como los estados dependen del nivel de abstracción utilizado. Los estados se pueden caracterizar de diferentes maneras, pero normalmente cada estado tiene un nombre y una descripción en la que se indica en la situación en la que se encuentra el sistema.

Diagrama de Estados

Un diagrama de estados relaciona sucesos y estados. Cuando se recibe un suceso, el estado siguiente depende del actual, así como del suceso; un cambio de estado causado por un suceso es lo que se llama una transición. Un diagrama de estados es un grafo cuyos nodos son estados, y cuyos arcos dirigidos son transiciones rotuladas con nombres de sucesos.

El diagrama de estados especifica la secuencia de estados que causa una cierta secuencia de sucesos. Si un objeto se encuentra en un cierto estado y se produce un suceso cuyo nombre corresponda al de una de sus transiciones, entonces el objeto pasa al estado que se encuentra en el extremo de destino de la transición. Se dice que la transición se dispara. Si hay más de una transición que sale de un estado, entonces el primer suceso que se produzca dará lugar a que se dispare la transición correspondiente.

Si se produce un suceso que no tiene ninguna transición que salga del estado actual, entonces el suceso se ignora. Una secuencia de sucesos se corresponde con un camino a través del grafo.

Un diagrama de estados describe el comportamiento de una sola clase de objetos. Dado que todas las instancias de un clase tienen el mismo comportamiento, todas ellas comparten el mismo diagrama de estados, por cuanto todas ellas comparten las mismas características de clase. Pero dado que todo objeto posee sus propios valores de atributos, cada objeto posee su propio estado, que es el resultado de la especial secuencia de sucesos que haya recibido. Todo objeto es independiente de los demás objetos, y procede a su paso.

Los diagramas de estados pueden representar ciclos vitales únicos o bien bucles continuos. Los diagramas de un solo uso representan objetos de duración finita y tienen estados iniciales y finales. Se entra en el estado inicial al crear el objeto; al entrar en el estado final estamos implicando la destrucción del objeto. Los diagramas de un solo uso son una “subrutina” del diagrama de estados, a la cual se puede hacer alusión desde distintos lugares en un diagrama de alto nivel.

El modelo dinámico es una colección de diagramas de estados que interactúan entre sí a través de sucesos compartidos.

Condiciones

Una condición es una función Booleana lógica que tiene a objetos como valores. Las condiciones se pueden utilizar como protecciones en las transiciones. Una transición con protección se dispara cuando se produce su suceso, pero sólo si la condición de protección es verdadera.

Operaciones

Los diagramas de estados tendrían muy poca utilidad si solamente describiesen tramas de sucesos. Una descripción de un objeto debe especificar lo que hace el objeto como respuesta a los sucesos.

Una actividad es una operación cuya realización requiere un cierto tiempo. Toda actividad está asociada a un estado. Entre las actividades se cuentan las operaciones continuas, tales como mostrar un imagen en una pantallas, así como las operaciones secuenciales que terminan por sí mismas después de un cierto intervalo de tiempo. Un estado puede controlar una actividad continua o una actividad secuencial que va avanzando hasta que termina o hasta que se produce un suceso que la hace finalizar prematuramente. La anotación “hacer: X” indica que la actividad secuencial X comienza al entrar en ese estado, y se detiene cuando ha finalizado. Si un suceso da lugar a una transición que sale de ese estado antes de que haya finalizado la actividad, entonces, la actividad finaliza de forma prematura.

Una acción es una operación instantánea que va asociada a un suceso. Una acción representa a una operación cuya duración es insignificante en comparación con la resolución del diagrama de estados. Realmente, no hay operaciones que sean instantáneas, pero se modelan de esta manera aquellas operaciones de las que no nos importa su estructura interna.

Las acciones también pueden representar operaciones internas de control, tales como dar valores a atributos o generar otros sucesos. Estas acciones no tienen contrapartidas en el mundo real, sino que son mecanismos para estructurar el control dentro de una implementación.

Diagramas de Estados Anidados

Los diagramas de estados se pueden estructurar, para hacer posibles unas descripciones concisas de sistemas complejos. Las formas de estructurar máquinas de estados son similares a las formas de estructurar los objetos: la generalización y la agregación. La generalización es el equivalente a expandir las actividades anidadas. Permite describir una actividad empleando un nivel alto, y expandirla después en un nivel más bajo añadiendo detalles, de forma similar a las llamadas a procedimientos anidados. Además, la generalización permite que los estados y los sucesos se dispongan en jerarquías de generalización con herencia de estructuras y comportamientos comunes, de forma similar a la herencia de atributos y de operaciones en las clases. La agregación permite que el estado se descomponga en componentes ortogonales, con una interacción

limitada entre ellos, de forma similar a una jerarquía de agregación de objetos. La agregación es el equivalente a la concurrencia de estados. Los estados concurrentes suelen corresponderse con agregaciones de objetos, posiblemente de todo un sistema, que tengan partes que interactúen.

- ✓ Anidamiento de diagramas de estados. Una actividad de un estado se puede expandir en forma de diagrama de estados de nivel inferior, en el cual cada uno representará un paso de la actividad. Las actividades anidadas son diagramas de estados de un solo uso, con transiciones de entrada y de salida, parecidas a subrutinas.
- ✓ Generalización de estados. Un diagrama de estados anidados es en realidad una forma de generalización de estados. Un objeto que se encuentre en un estado del diagrama de alto nivel tiene que estar precisamente en uno de los estados del diagrama anidado. Los estados del diagrama anidado son todos ellos refinamientos del estado del diagrama de alto nivel. Los estados pueden poseer subestados que hereden las transiciones de sus superestados, del mismo modo que las clases poseen subclases que heredan los atributos y operaciones de sus superclases. Toda transición o acción que sea aplicable a un estado es aplicable también a todos sus subestados, a no ser que sea invalidada por una transición equivalente del subestado. Las transiciones de un superestado son heredadas por todos sus subestados.

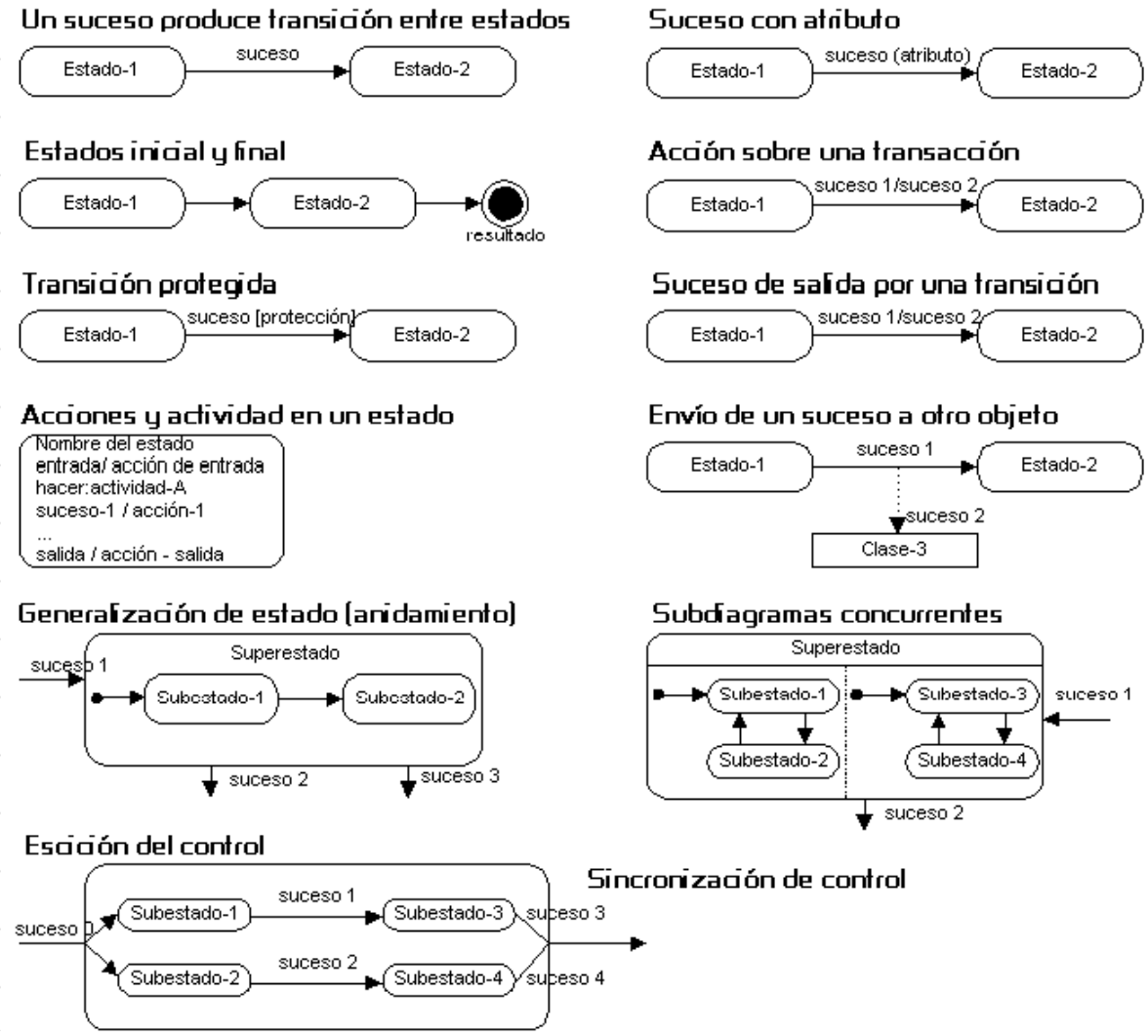
Concurrencia

- ✓ Concurrencia de Agregación. Un modelo dinámico describe un conjunto de objetos concurrentes, cada cual con su propio estado y con su propio diagrama de estados. Los objetos de todo sistema son inherentemente concurrentes, y pueden cambiar de estado independientemente. El estado de todo el sistema no se puede representar mediante un solo estado de un único objeto; es el producto de los estados de todos los objetos que lo componen. En muchos sistemas además, el número de objetos puede cambiar dinámicamente. Un diagrama de estados para un subsistema es una colección de los mismos, uno por cada componente. La agregación implica concurrencia. El estado agregado corresponde a los estados combinados de todos los diagramas que componen el subsistema. El estado agregado es un estado del primer diagrama y de todos los demás diagramas. En algunos casos los estados componentes interactúan. Las transiciones protegidas para un objeto pueden depender de que otro objeto se encuentre en un cierto estado. Esto permite la interacción entre diagramas de estados, manteniendo al mismo tiempo la modularidad.
- ✓ Concurrencia dentro de un objeto. La concurrencia dentro del estado de un único objeto surge cuando se puede descomponer el objeto en subconjuntos de atributos o de enlaces, cada uno de los cuales posee su propio subdiagrama.

Desarrollo de un modelo dinámico

- Preparar escenarios para las secuencias de interacción típicas.
- Identificar eventos entre objetos y preparar trazos de eventos para cada escenario.
- Preparar un diagrama de flujo de eventos para el sistema.
- Desarrollar un diagrama de estados para cada clase que tenga un comportamiento dinámico importante.
- Verificar que los eventos compartidos entre diagramas de estado sean consistentes y correcctos.

Notaciones



Modelo Funcional

El modelo funcional describe los cálculos existentes dentro del sistema siendo la

tercera parte del modelado. Dentro del modelado del sistema, el modelo funcional especifica lo que sucede, el modelo dinámico cuándo sucede, y el modelo de objetos especifica a qué le sucede.

El modelo funcional muestra la forma en que se derivan los valores producidos en un cálculo a partir de los valores introducidos, sin tener en cuenta el orden en el cual se calculan los valores. Consta de múltiples diagramas de flujo de datos, que muestran el flujo de valores desde las entradas externas, a través de las operaciones y almacenes internos de datos hasta las salidas externas. También incluyen restricciones entre valores dentro del modelo de objetos. Los diagramas de flujo de datos no muestran el control ni tampoco información acerca de la estructura de los objetos; todo esto pertenece a los modelos dinámico y de objetos.

Diagramas de flujo de datos

El modelo funcional consta de múltiples diagramas de flujo de datos, que especifican el significado de las operaciones y de las restricciones. Un diagrama de flujo de datos (DFD) muestra las relaciones funcionales entre los valores calculados por un sistema, incluyendo los valores introducidos, los obtenidos, y los almacenes internos de datos. Un diagrama de flujo de datos es un grafo que muestra el flujo de valores de datos desde sus fuentes en los objetos mediante procesos que los transforman, hasta sus destinos en otros objetos. Un diagrama de flujo de datos no muestra información de control como puede ser el momento en que se ejecutan los procesos o se toman decisiones entre vías de datos alternativas; esta información pertenece al modelo dinámico. Un diagrama de flujo de datos no muestra la organización de los valores en objetos; esta información pertenece al modelo de objetos.

Un diagrama de flujo de datos contiene procesos que transforman datos, flujos de datos que los trasladan, objetos actores que producen y consumen datos, y de almacenes de datos que los almacenan de forma pasiva.

- ✓ Procesos. Un proceso transforma valores de datos. Los procesos del más bajo nivel son funciones puras, sin efectos laterales. Un grafo completo de flujo de datos es un proceso de alto nivel. Los procesos pueden tener efectos laterales si contienen componentes no funcionales, tales como almacenes de datos u objetos externos. El modelo funcional no especifica de forma única el resultado de un proceso que tenga efectos laterales, solamente indica las posibles vías funcionales; no muestra la que realmente se recorrerá. Los resultados de estos procesos dependen del comportamiento del sistema, según se especifica en el modelo dinámico. El diagrama muestra solamente el patrón de entradas y salidas. El cálculo de valores de salida a partir de los de entrada también debe ser especificado. Un proceso de alto nivel se puede expandir en todo un diagrama de flujo de datos, de forma muy parecida a la manera en que se puede expandir una subrutina en otra subrutina de nivel inferior. Eventualmente, la recursividad

finaliza, y los procesos atómicos deben describirse directamente, en lenguaje natural, mediante ecuaciones matemáticas, o por algún otro medio. Los procesos se implementan como métodos de operaciones que se aplican a clases de objetos. El objeto destino suele ser uno de los flujos de entrada, sobre todo si esa misma clase de objeto es también un flujo de salida.

- ✓ Flujo de datos. Un flujo de datos conecta la salida de un objeto o proceso con la entrada de otro objeto o proceso. Representa un valor de datos intermedio dentro de un cálculo que no es modificado por el flujo de datos. El mismo valor se puede enviar a varios lugares. En algunas ocasiones un valor de datos agregado se descompone en sus componentes, cada uno de los cuales va a un proceso diferente. La combinación de varios componentes en un valor agregado es justamente lo contrario. Cada flujo de datos representa un valor en algún momento del cálculo. Los flujos de datos internos al diagrama representan valores intermedios dentro de un cálculo, y o tienen necesariamente ningún dignificado en el mundo real. Los flujos en la frontera de un diagrama de flujo de datos son sus entradas y salidas. Estos pueden ser inconexos (si el diagrama es un fragmento de un sistema completo), o bien pueden estar conectados con objetos.

- ✓ Actores. Un actor es un objeto activo que controla el grafo de flujo de datos produciendo o consumiendo valores. Los actores están asociados a las entradas y salidas del grafo de flujo de datos. En cierto sentido, los actores yacen en la frontera del grafo, pero hacen que concluya el flujo de datos como consumidores y productores de datos, así que en algunos casos se llaman sumideros. Las flechas entre el actor y el diagrama son las entradas y salidas del diagrama.
- ✓ Almacenes de datos. Un almacén de datos es un objeto pasivo dentro de un diagrama de flujo de datos que almacena datos para su posterior utilización, se limita a responder a solicitudes de almacenamiento y acceso a datos. Los almacenes de datos permiten acceder a los valores por un orden distinto a aquel en que fueran generados. Tanto los actores como los almacenes de datos son objetos. Se diferencian en que su comportamiento y utilización suelen ser distintos, aún cuando en un lenguaje orientado a objetos ambos pudieran ser implementados como objetos. Por otra parte, un almacén de datos se podría implementar como un fichero, y un actor en un dispositivo externo. Algunos flujos de datos son también objetos, aún cuando en muchos casos son valores puros, tales como enteros, que carecen de una identidad individual.
- ✓ Diagramas de flujo de datos anidados. Un diagrama de flujo de datos resulta muy útil para mostrar la funcionalidad de alto nivel de un sistema, y su descomposición en unidades funcionales más pequeñas. Un proceso se puede expandir en otro diagrama de flujo de datos. Todas las entradas y salidas del proceso lo serán también del nuevo diagrama, también puede tener almacenes de datos que no se muestren en el diagrama de nivel superior. Los diagramas se pueden anidar hasta una profundidad arbitraria, y todo el conjunto de diagramas anidados forman un árbol. Esto permite

que cada nivel sea coherente y comprensible, y con toda la funcionalidad global que puede ser arbitrariamente compleja. Un diagrama que hace referencias a si mismo repetidamente representa un cálculo recursivo. El anidamiento de diagramas concluye con funciones sencillas que deben ser especificadas como operaciones.

- ✓ Flujos de control. Un diagrama de flujo de control muestra todas las posibles vías de computación para los valores. No muestra cuales son las vías que se ejecutan ni en qué orden. Las decisiones y la secuenciación son problemas de control, que forman parte del modelo dinámico. Una decisión afecta a si una o más funciones llagan incluso a ejecutarse, en lugar de proporcionarnos un valor. Aún cuando las funciones no poseen valores de entrada procedentes de estas funciones de decisión, a veces resulta útil incluirlas en el modelo funcional, para mostrar su dependencia con respecto a los datos. Esto se hace incluyendo flujos de control en el diagrama de flujo de datos. Un flujo de control es un valor booleano que afecta a si un proceso es o no ejecutado. El flujo de control no es un valor de entrada al proceso en sí. Los flujos de control pueden ser útiles en ocasiones, pero duplican información del modelo dinámico.

Especificación de Operaciones

Los procesos de los diagramas de flujo deben ser implementados eventualmente como operaciones que se aplican a objetos. Todo proceso atómico del más bajo nivel es una operación. Los procesos de nivel superior también se pueden considerar operaciones, aún cuando una implementación pueda estar organizada de forma distinta del diagrama de datos que representa como consecuencia de la optimización. Toda operación se podrá especificar de diferentes maneras, entre las que están:

- funciones matemáticas,
- tablas de valores de entrada y salida,
- ecuaciones que especifican la salida dependiendo de la entrada, condiciones previas y posteriores,
- tablas de decisión,
- pseudocódigo,
- lenguaje natural.

La especificación de una operación se compone de dos partes. La primera de ellas es la que indica la interfaz de la operación: los argumentos que requiere y los valores que proporciona. La segunda, es la que explica la transformación de los valores de entrada para producir los valores de salida.

La especificación externa de una operación describe solamente cambios visibles fuera de ella. Durante la implementación de una operación, se pueden crear valores internos por conveniencia o por optimización. Algunos pueden incluso formar parte del estado interno de un objeto. El propósito de la especificación es indicar lo que debe hacer una operación lógicamente, y no como debe ser

implementada. Por tanto, el estado del objeto en sí debe dividirse en información visible externamente e información privada, interna. Los cambios del estado interno de un objeto que no sean externamente visibles no modificarán su valor.

Las operaciones de acceso son operaciones que leen o escriben atributos o enlaces de un objeto. No es necesario enumerarlos o especificarlos durante el análisis. Durante el diseño, es necesario observar cuáles de las operaciones de acceso van a ser públicas, y cuáles privadas para esa clase de objetos. La razón para restringir el acceso no es una razón de corrección lógica, sino más bien para encapsular las clases con el objetivo de protegerlas contra errores y para permitir modificaciones en la implementación en un futuro. Las operaciones de acceso se derivan directamente de los atributos y asociaciones de la clase dentro del modelo de objetos.

Las operaciones no triviales se pueden desglosar en tres categorías: consultas, acciones y actividades. Una consulta es una operación que carece de efectos laterales en el estado visible externamente de cualquier objeto; es una función pura. Una consulta sin parámetros recibe el nombre de atributo derivado porque tiene la forma de un atributo.

Una acción es una transformación que posee efectos laterales sobre el objeto destino, o sobre otros objetos del sistema que resulten alcanzables desde él. Las acciones no tienen una duración temporal: son instantáneas. Dado que el estado de un objeto queda definido por sus atributos y enlaces, todas las acciones deben de ser definibles en términos de actualizaciones de los atributos y enlaces básicos. Se puede definir una acción en términos del estado del sistema antes y después de la acción, no es necesario un componente de control.

Las acciones se pueden describir de diferentes maneras, incluyendo las ecuaciones matemáticas, árboles, tablas de decisión, enumeración de todas las posibles entradas, cálculo de predicados y lenguaje natural. Es importante que la especificación sea clara y no ambigua. Es necesaria una especificación formal. Hay varios elementos en la especificación: el nombre de la función, las entradas y salidas, las transformaciones de valores y las restricciones.

Una actividad es una operación hecha por o sobre un objeto que tiene una cierta duración temporal, por oposición a las consultas y acciones, que se consideran instantáneas. Una actividad tiene efectos colaterales como consecuencia de su duración temporal. Las actividades sólo tienen sentido para actores y objetos que generen operaciones propias, porque los pasivos son solamente depósitos de datos. Los detalles de una actividad son especificados por el modelo dinámico, así como por el modelo funciones, y no se pueden considerar tan sólo una transformación. En la mayoría de los casos, una actividad corresponde a un diagrama de estados del modelo dinámico.

Restricciones

Una restricción muestra la relación entre dos objetos al mismo tiempo o bien entre distintos valores del mismo objeto en instantes diferentes. Las restricciones se pueden expresar como una función total (un valor que es especificado completamente por otro) o como una función parcial (un valor que está restringido, pero no completamente especificado por otro valor). Las restricciones pueden aparecer en todas las clases del modelo.

Las restricciones de objetos especifican que algunos objetos dependen entera o parcialmente de otros objetos. Las restricciones dinámicas especifican relaciones entre los estados o sucesos de distintos objetos. Las restricciones funcionales especifican limitaciones aplicables a operaciones.

Una restricción entre valores de un objeto a lo largo del tiempo es lo que suele denominarse un invariante.

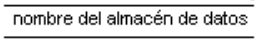
Construcción de un modelo funcional

- Identificar valores de entrada y salida.
- Usar diagramas de flujo de datos para mostrar dependencias funcionales.
- Describir las funciones.
- Identificar restricciones.
- Especificar criterios de optimización.

Notación

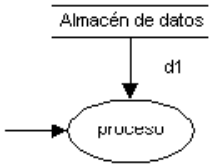
Proceso

Almacén de datos u Objeto archivo



Objetos actor (como fuente o sumidero de datos)

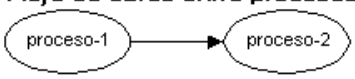
Acceso a un valor de un almacén de datos



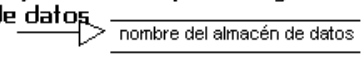
Acceso y actualización de un valor de un almacén de datos

Duplicación de un valor de datos

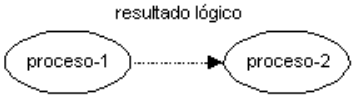
Flujo de datos entre procesos



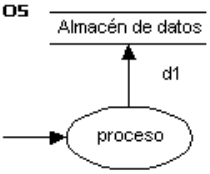
Flujo de datos que da lugar a un almacén de datos



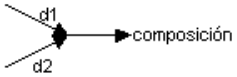
Flujo de control



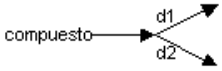
Actualización de un valor de un almacén de datos



Composición de un valor de datos



Descomposición de un valor de datos



Fase de Análisis.

El objetivo del análisis es desarrollar un modelo del funcionamiento del sistema. El modelo se expresa en términos de objetos y relaciones, el control dinámico de flujo y las transformaciones funcionales. El proceso de capturar los requerimientos y consultar con el solicitante debe ser continuo a través del análisis. A saber:

1. Contar con una descripción inicial del problema (enunciado del problema).
2. Construir un modelo de objetos. Modelo de objetos = diagramas del modelo de objetos + diccionario de datos.
3. Desarrollar un modelo dinámico. Modelo dinámico = diagramas de estado + diagrama global de flujo de eventos.
4. Construir un modelo funcional. Modelo funcional = diagramas de flujo de datos + restricciones.
5. Verificar, iterar y refinar los tres modelos:
   - Agregar al modelo de objetos operaciones clave que sean descubiertas durante la preparación del modelo funcional. No deben mostrarse todas las operaciones durante el análisis, sólo las más importantes.
   - Verificar que las clases, asociaciones, atributos y operaciones sean consistentes y completos al nivel seleccionado de abstracción. Comparar los tres modelos con el enunciado del problema y el

conocimiento relevante al dominio y probar los modelos usando varios escenarios.
- Desarrollar escenarios más detallados (incluyendo condiciones de error) como variaciones de los escenarios básicos, para verificar aún más los tres modelos.
- Iterar los pasos anteriores según sea necesario para completar el análisis.

Documento de análisis = enunciado del problema + modelo de objetos + modelo dinámico + modelo funcional.

Fase de Diseño de sistemas.

Durante el diseño de sistemas, se selecciona la estructura de alto nivel del sistema. Existen varias arquitecturas canónicas que pueden servir como un punto de inicio adecuado. El paradigma orientado a objetos no introduce vistas especiales en el diseño del sistema, pero se incluye para tener una cobertura completa del proceso de desarrollo de software. Los pasos son:

1. Organizar el sistema en subsistemas.
2. Identificar la concurrencia inherente al problema.
3. Asignar subsistemas a procesadores y tareas.
4. Escoger la estrategia básica para implantar los almacenamientos de datos en términos de estructuras de datos, archivos y bases de datos.
5. Identificar recursos globales y determinar los mecanismos para controlar su acceso.
6. Seleccionar un esquema para implantar el control del software:
   - Usar la ubicación dentro del programa para mantener el estado,
   - o implantar directamente una máquina de estado,
   - o usar tareas concurrentes.
7. Considerar las condiciones de frontera.
8. Establecer prioridades de decisión sobre características deseables del producto de software.

Documento de diseño de sistemas = estructura de la arquitectura básica del sistema + las decisiones de estrategias de alto nivel.

Fase de Diseño de objetos.

Durante el diseño de objetos se elabora el modelo de análisis y se proporciona una base detallada para la implantación. Se toman las decisiones necesarias para realizar un sistema sin entrar en los detalles particulares de un lenguaje o base de datos particular. El diseño de objetos inicia un corrimiento en el enfoque de la orientación del mundo real del modelo de análisis hacia la orientación en la computadora requerida para una implantación práctica. Los pasos son:

1. Obtener las operaciones para el modelo de objetos a partir de los otros modelos:
   - Encontrar una operación para cada proceso en el modelo funcional.
   - Definir una operación para cada evento en el modelo dinámico, dependiendo de la implantación del control.
2. Diseñar los algoritmos para implantar las operaciones:
   - Escoger los algoritmos que minimicen el costo de implementación de las operaciones.
   - Seleccionar las estructuras de datos apropiadas para los algoritmos.
   - Definir clases internas y operaciones nuevas según sea necesario.
   - Asignar las responsabilidades para las operaciones que no están asociadas claramente con una sola clase.
3. Optimizar las rutas de acceso a los datos:
   - Agregar asociaciones redundantes para minimizar los costos de acceso y maximizar la conveniencia.
   - Reacomodar los cálculos para una mayor eficiencia.
   - Guardar los valores derivados para evitar recalcular expresiones complicadas.
4. Implantar el control del software introduciendo el esquema seleccionado durante el diseño de sistemas.
5. Ajustar la estructura de clases para incrementar la herencia:
   - Reacomodar y ajustar las clases y las operaciones para incrementar la herencia.
   - Abstraer el comportamiento común de los grupos de clases.
   - Usar delegación para compartir comportamiento donde la herencia sea semánticamente inválida.
6. Diseñar la implantación de las asociaciones:
   - Analizar las travesías de las asociaciones.
   - Implantar cada asociación como un objeto distinto o agregando atributos objeto-valor a una o ambas clases en la asociación.
7. Determinar la representación de los atributos de los objetos.
8. Empaquetar las clases y las asociaciones en módulos.

Documento de diseño = modelo de objetos detallado + modelo dinámico detallado + modelo funcional detallado.

Ventajas

1. Proporciona una serie de pasos perfectamente definidos al desarrollador.
2. Tratamiento especial de la herencia.
3. Facilita el mantenimiento dada la gran cantidad de información que se genera en el análisis.
4. Es fuerte en el análisis

Desventajas

1. Hay pocos métodos para encontrar inconsistencias en los modelos.
2. Interacción de objetos no soportada explícitamente en ninguna herramienta gráfica.
3. Al ser un análisis iterativo es difícil de saber cuando comenzar con el diseño.
4. Es débil en el diseño

APLICACIONES

Esta Tecnología puede ser aplicada en varios aspectos de implementación incluyendo:

- Archivos.
- Base de datos relacionales.
- Base de datos orientadas a objetos.
- Estructura de datos.
- Multimedia.
- Interactivas.
- Web.
- Cliente/servidor.
- Distribuidas.

Y en general, en prácticamente cualquier actividad de ingeniería que requiera hacer un análisis de un problema para poder resolver un problema.

Herramientas CASE que soportan OMT

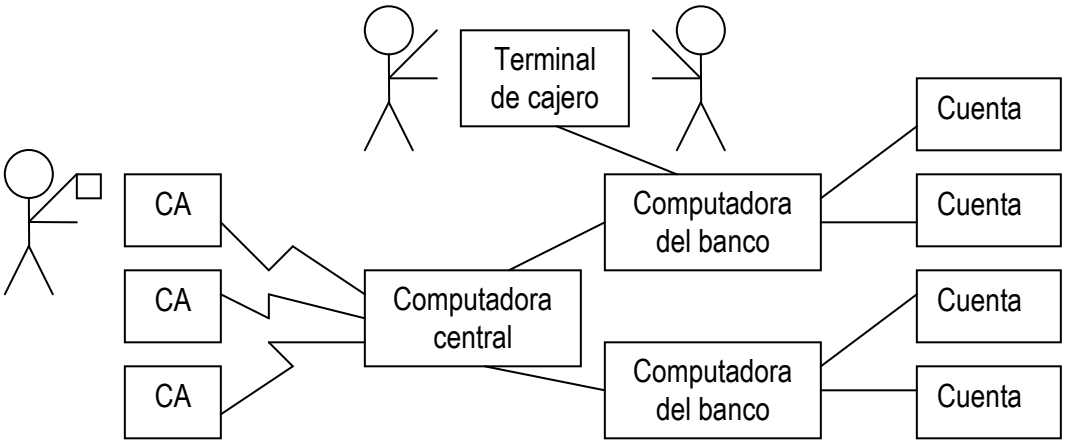
- Excelerator II Intersolv Inc.
- MetaEdit MetaCASE Consulting YO
- ObjectMarker, Mark V Software
- BOCS, Berard Software Eng.
- ObjectTeam, Candre Technologies, Inc.
- OMTool, Martin Marietta.
- Paradigm Plus, Protosoft.
- Software Through Pictures, Interactive Development Enviroment
- System Architect, Popkin Software.

EJEMPLOS

Sistema de cajero automático: ATM (Automated Teller Machine)

Diseñar el software para dar soporte a una red bancaria automatizada, que incluya tanto cajeros humanos como cajeros automáticos (CA), y que deberán ser compartidos por un consorcio de bancos. Cada banco proporciona sus propias computadoras para mantener sus cuentas y procesar transacciones relativas a

ellas. Las terminales de cajero son propiedades de cada banco, y se comunican directamente con las computadoras del banco. Los cajeros humanos insertan los datos de la cuenta y de la transacción. Los cajeros automáticos se comunican con una computadora central que aprueba las transacciones con los bancos adecuados. Los cajeros automáticos admiten tarjetas, interaccionan con el usuario, se comunican con el sistema central para llevar a cabo la transacción, entregan dinero e imprimen recibos. El sistema necesita mantener unos registros adecuados y también las oportunas medidas de seguridad y debe admitir accesos concurrentes a una misma cuenta de forma correcta. Los bancos proporcionarán su propio software para sus computadoras; el analista debe diseñar el software para los CA y para la red. El coste del sistema compartido será prorrateado entre los bancos de acuerdo con el número de clientes que tengan sus tarjetas de crédito.

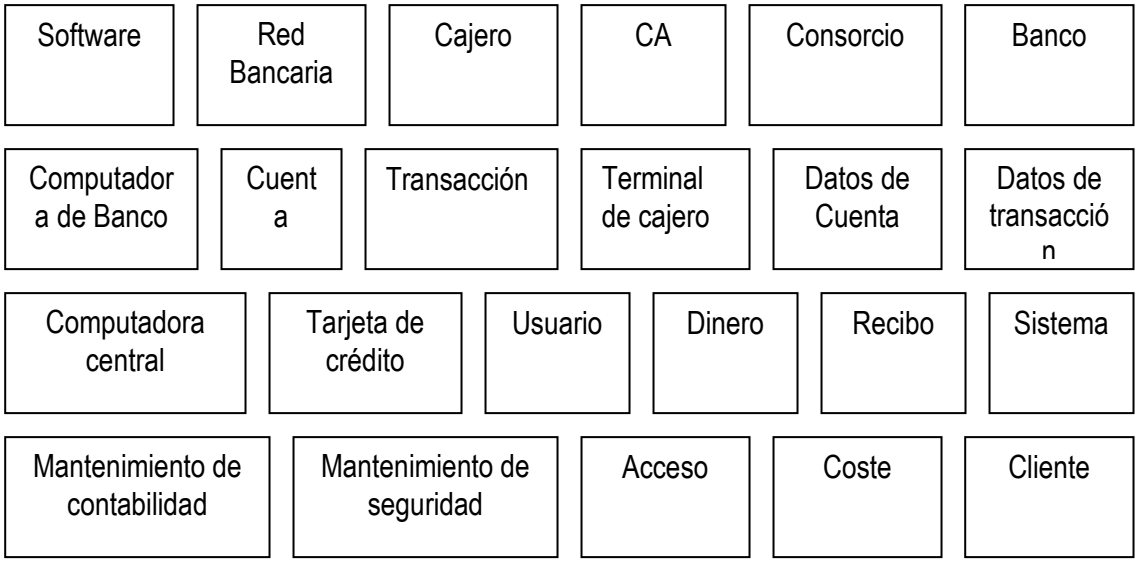


Una red de CA

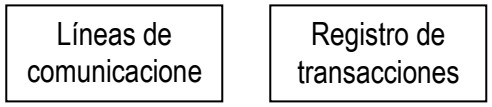
ANÁLISIS

MODELO DE OBJETOS:

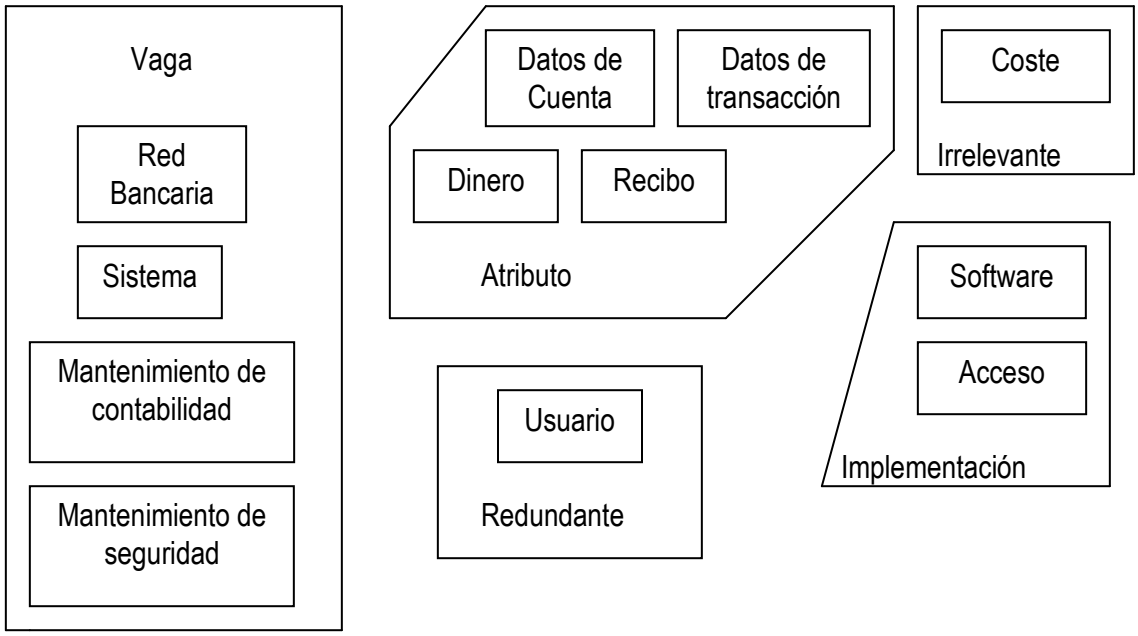
- Identificar los objetos y la clase.



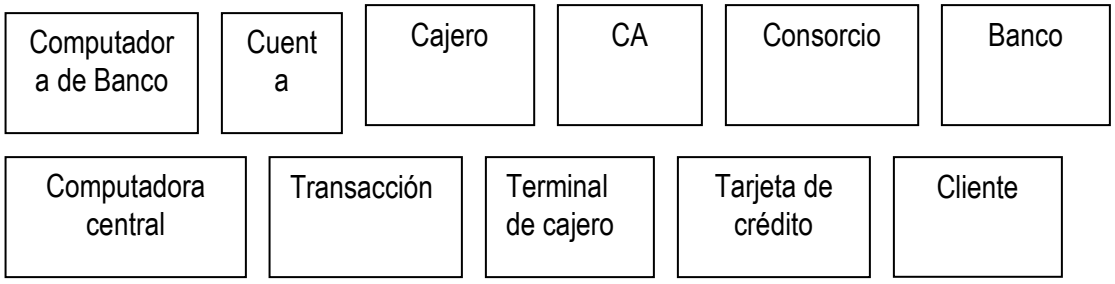
Clases extraídas de los nombres de definición del problema



Clases de CA identificadas a partir del conocimiento del dominio del problema



Clases Incorrectas



Clases Correctas

- Preparar un diccionario de datos

Cuenta→ Cuenta individual de un banco a la cual se le pueden aplicar transacciones. Las cuentas pueden ser de varios tipos; como mínimo serán de ahorro o a la vista. Un cliente puede tener más de una

CA→ Punto que permite a los clientes introducir sus propias transacciones empleando una tarjeta de crédito como identificación. El CA interacciona con el cliente para obtener información de la transacción, la envía a la computadora central para su verificación y procesamiento, y suministra dinero al usuario. Suponemos que el CA no necesita funcionar independientemente de la red.

Banco→ Una institución financiera que tiene cuentas para clientes y que proporciona tarjeta de crédito que autorizan para acceder a dichas cuentas a través de la red de CA.

Computadora de banco→ Computadora de un banco, que tiene una interfaz con la red de CA, y con los puestos de cajeros del propio blanco. Éste puede tener su propia red interna de computadoras para procesar las cuentas, aunque sólo nos concierne la que se comunica con la red.

Tarjeta de Crédito→ Tarjeta que le ha sido asignada a un cliente del banco, y que le autoriza para acceder a cuentas empleando un CA. Cada tarjeta contiene un número y código de banco, que estarán codificados, con toda probabilidad, de acuerdo con los estándares nacionales para tarjetas de crédito y de débito (bancarias). El código del banco le identifica de forma única dentro del consorcio. El número de la tarjeta determina las cuentas a las cuales puede acceder. Una tarjeta no accede necesariamente a todas las cuentas del cliente. Toda tarjeta bancaria es poseída por un único cliente, pero pueden existir múltiples copias de ella de modo que es preciso considerar la posibilidad de su uso simultáneo en varias máquinas distintas.

Cajero→ Empelado de un banco que está autorizado para efectuar transacciones en los terminales de cajero y para admitir y proporcionar dinero y cheques a los clientes. Las transacciones, el dinero y los cheques gestionados

por cada cajero deben ser insertados en las computadoras y controlados debidamente.

Terminal de cajero→ Puesto en el cual los cajeros introducen transacciones de sus clientes. Los cajeros proporcionan dinero y cheques; el terminal imprime recibos. El terminal de cajero se comunica con la computadora del banco para verificar y procesar las transacciones.

Computadora central→ Computadora explotada por el consorcio y encargada de despachar las transacciones entre los CA y las computadoras de los bancos. Verifica los códigos de los bancos, pero no procesa directamente las transacciones.

Consorcio→ Organización de los bancos que contrata y explota la red de CA. La red sólo admite transacciones para los bancos del consorcio.

Cliente→ Poseedor de una o más cuentas de un banco. Un cliente puede ser una o más personas o compañías; la correspondencia no es relevante para este problema. Una misma persona que tenga una cuenta en distintos bancos será considerada como varios clientes distintos.

Transacción→ Única solicitud completa de operaciones que afecta a cuentas de un solo cliente. Hemos especificado que los CA deben de proporcionar dinero, aunque no debería excluirse la posibilidad de imprimir cheques o de admitir dinero o cheques. Quizá sea necesario también ofrecer la flexibilidad para operar sobre cuentas de distintos clientes, aunque esto no se necesita todavía. Las distintas operaciones deben cuadrar correctamente.

Diccionario de datos para las clases de CA.

- Identificar asociaciones entre objetos

Locuciones Verbales

La red bancaria incluye cajeros y CA
El consorcio comparte los CA.
El banco proporciona la computadora del banco.
La computadora del banco proporciona las cuentas.
La computadora del banco procesa las transacciones de cada cuenta.
El banco posee el punto de caja.
El punto de caja se comunica con la computadora del banco.
El cajero introduce las transacciones para la cuenta.
Los CA se comunican con la computadora central para la transacción.
La computadora central verifica la transacción con el banco.
El CA admite tarjetas bancarias.
El CA interacciona con el usuario.
El CA proporciona dinero.

El CA imprime recibos.
El sistema gestiona accesos concurrentes.
Los bancos aportan su software.
Los costes se prorratean entre los bancos.

Locuciones verbales implícitas

El consorcio está formado por bancos.
Los bancos tienen cuentas.
El consorcio posee la computadora central.
El sistema se encarga del registro.
El sistema se encarga de la seguridad.
Los clientes tienen tarjeta de crédito.

Conocimiento del dominio del problema

Las tarjetas de crédito acceden a cuentas.
Los bancos emplean cajeros.

Asociaciones para la definición del problema de un CA.

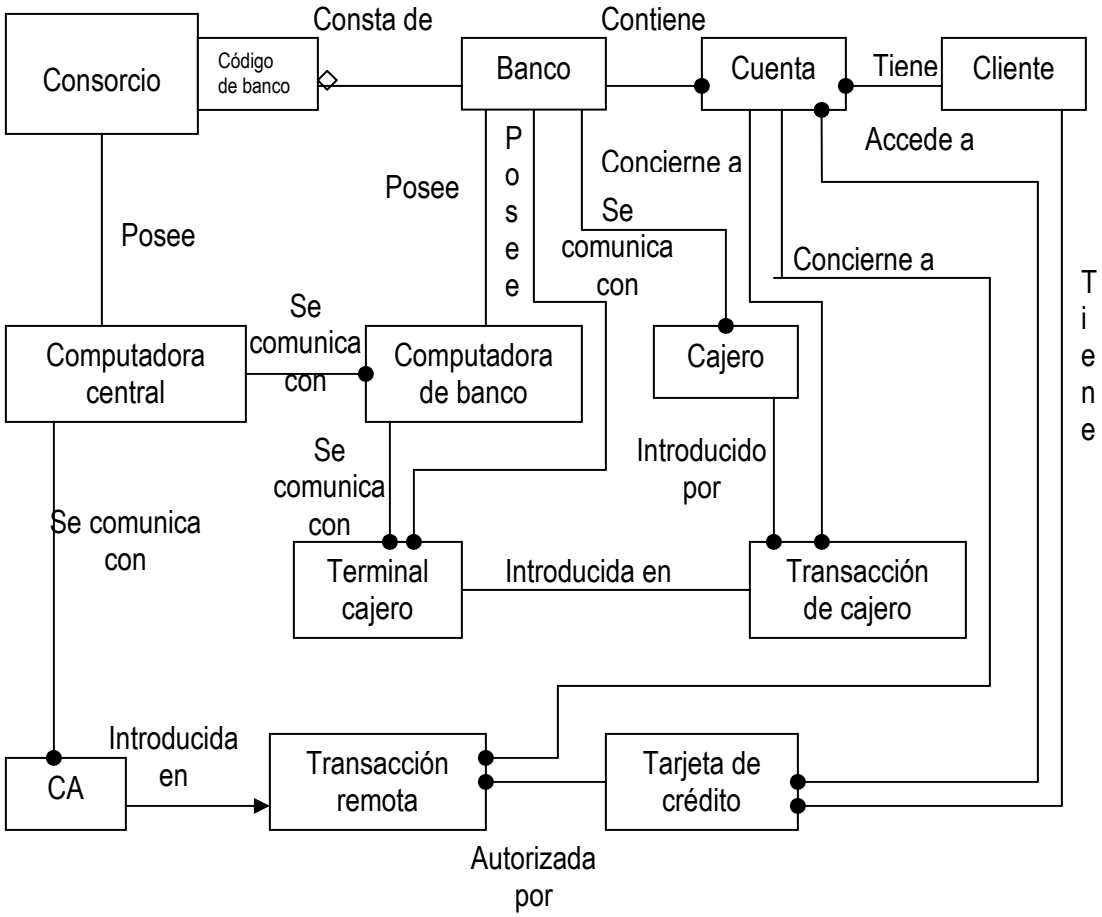
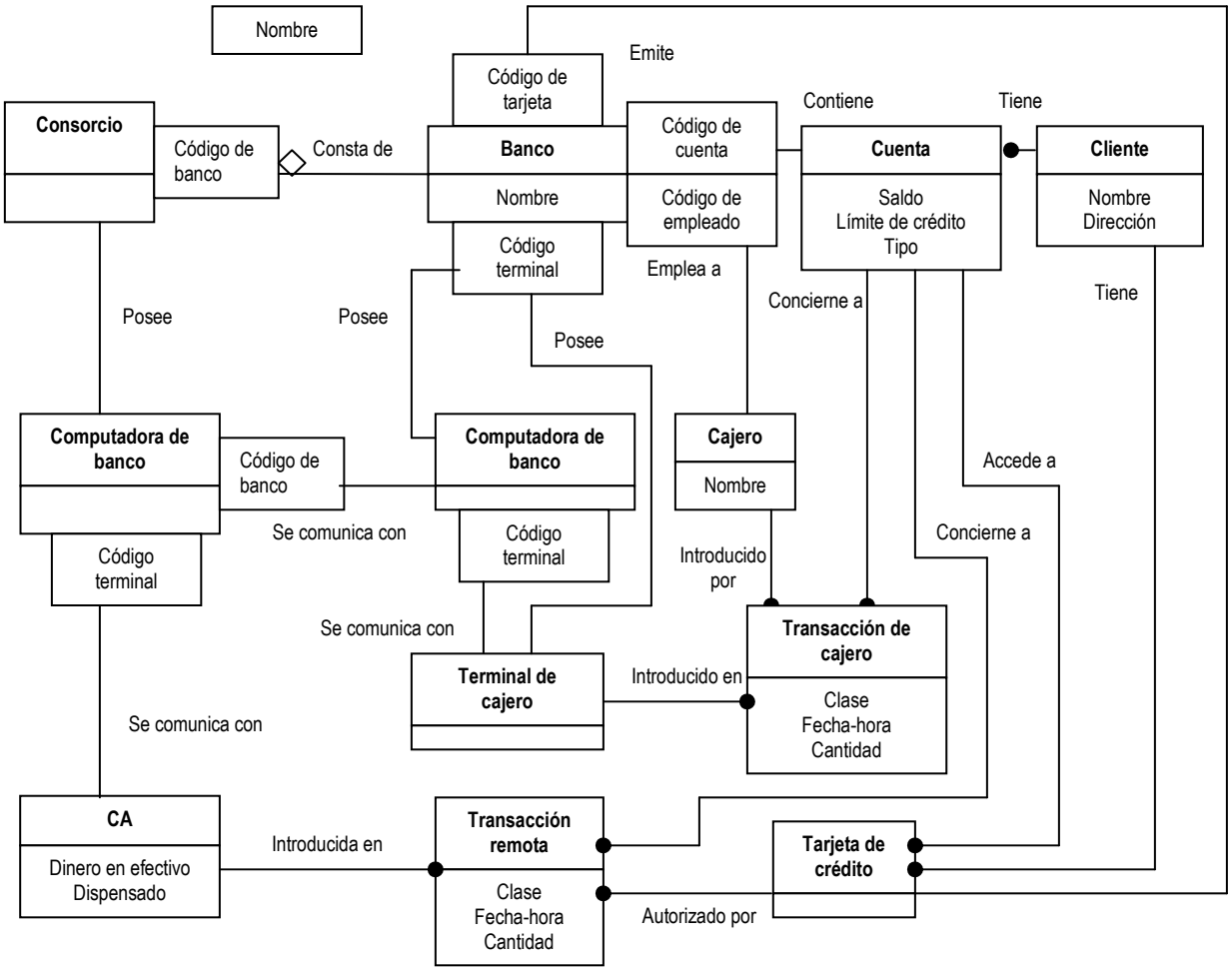
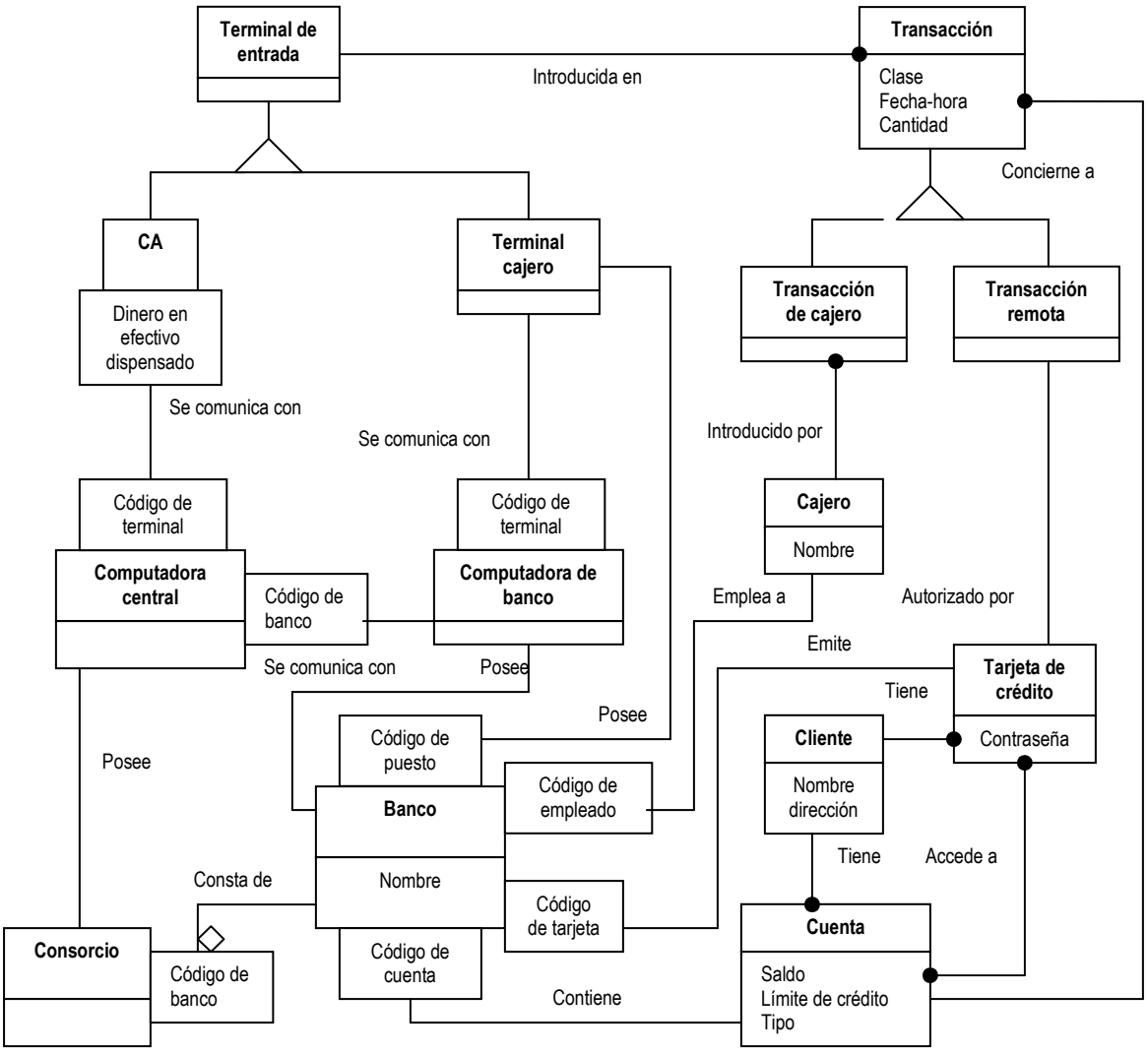


Diagrama inicial para un sistema ATM.



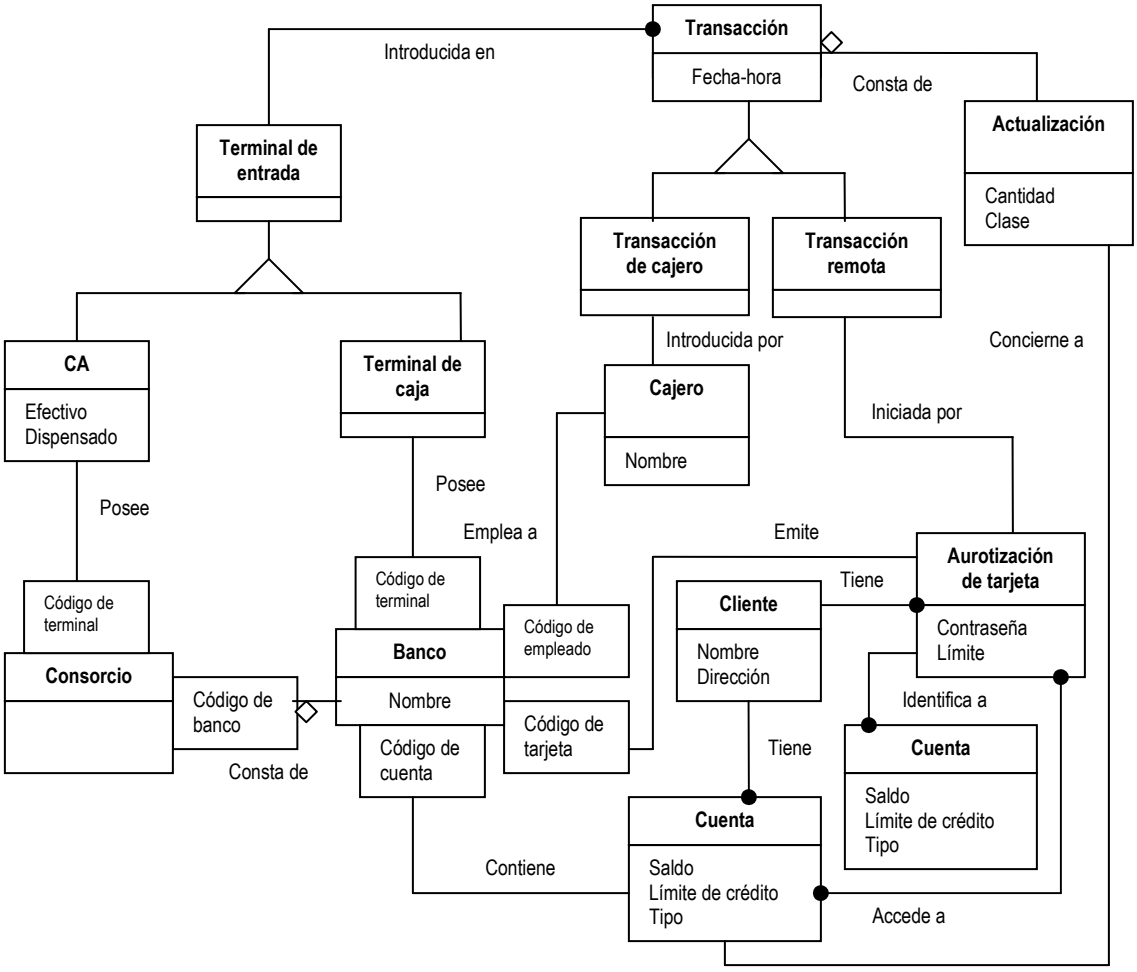
Modelo de objetos de un CA con sus atributos

- Organizar y simplificar la clase de objetos usando herencia.



Modelo de objetos de un CA con atributos y herencia

- Verificar que existen las vías de acceso adecuadas para las probables consultas.
- Iterar y refinar el modelo.



Modelo de objetos del CA después de otra revisión

- Agregar las clases en módulos

MODELADO DINÁMICO:

- Se preparan escenarios de secuencias típicas de interacción.

El CA pide al usuario que inserte una tarjeta; el usuario inserta una tarjeta de crédito.
El CA admite la tarjeta y lee su número de serie.
El CA solicita la contraseña; el usuario escribe “1234”.
El CA verifica el número de serie y la contraseña con el consorcio; esta la comprueba con el banco “39” y notifica la aceptación al CA.
El CA pide al usuario que seleccione la clase de transacción que desea (retirar fondos, hacer un ingreso o una transferencia); el usuario selecciona retirar fondos.
El CA verifica que la cantidad se encuentre dentro de los límites de crédito predefinidos, y pide al consorcio que procese la transacción; éste pasa la

solicitud al banco, que eventualmente confirma el éxito de la misma y proporciona el nuevo saldo disponible en cuenta.
El CA proporciona el dinero y pide al usuario que lo recoja; éste toma el dinero.
El CA pregunta si el usuario desea continuar; éste dice que no.
El CA imprime un recibo, expulsa la tarjeta y pide al usuario que la recoja; el usuario toma el recibo y la tarjeta.
El CA pide a un usuario que inserte una tarjeta.

Escenario normal de un CA.

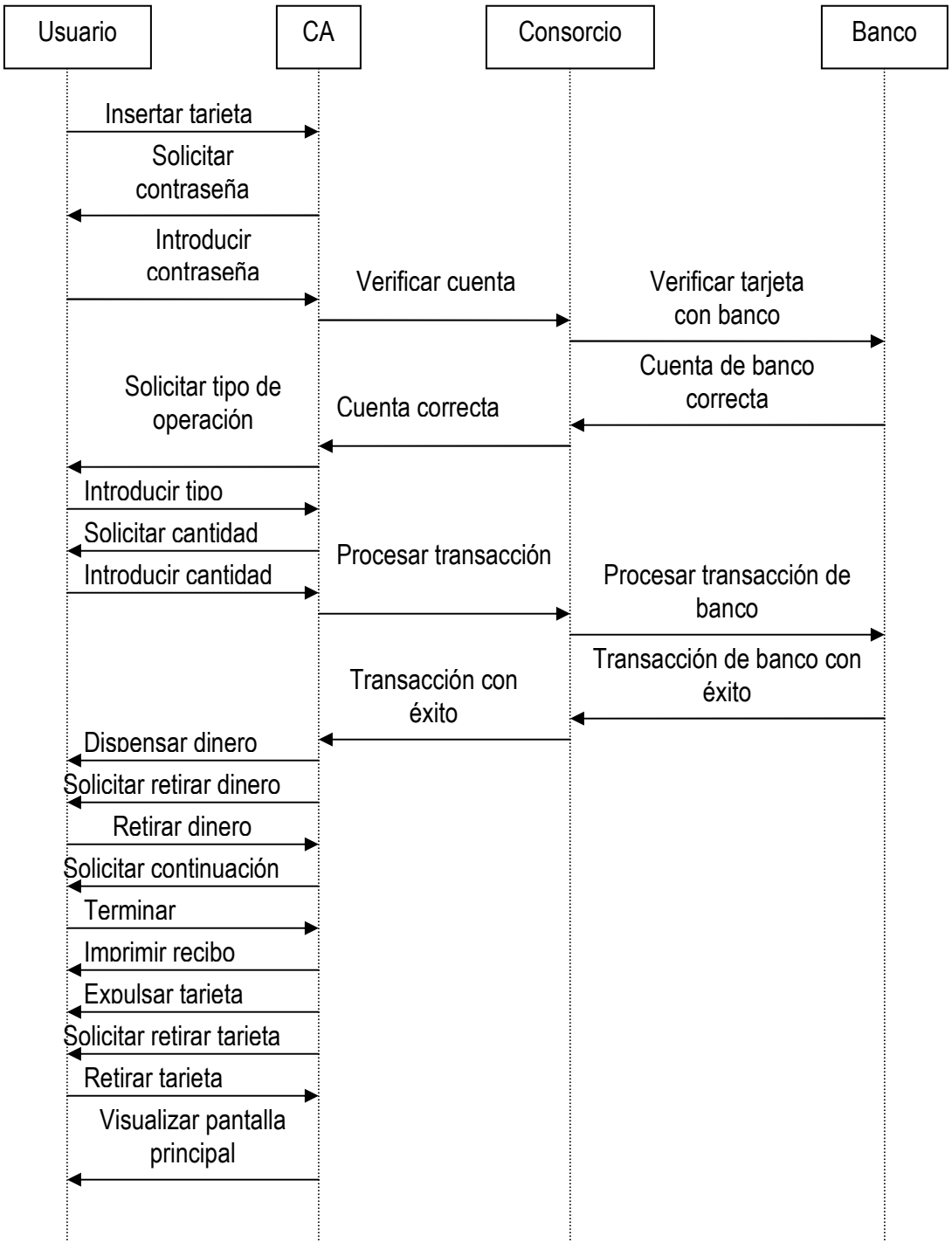
El CA pide al usuario que inserte una tarjeta; inserta una tarjeta de crédito.
El CA admite la tarjeta y se lee un número de serie.
El CA solicita la contraseña; el usuario escribe “9999”.
El CA verifica el número de serie y la contraseña con el consorcio, que los rechaza después de consultar con el banco adecuado.
El CA indica que la contraseña es incorrecta, y pide al usuario que vuelva a escribirla; éste usuario escribe “1234”, y la tarjeta es admitida por el consorcio tras verificar el CA.
El CA pide al usuario que seleccione la clase de transacción que desea; el usuario selecciona una retirada de fondos.
El CA pregunta la cantidad de dinero; el usuario cambia de opinión y pulsa “cancelar”.
El CA expulsa la tarjeta y pide al usuario la recoja, el usuario la recoge.
El CA pide a un usuario que inserte una tarjeta.

Un escenario de CA con excepciones.

- Se identificar sucesos que actúen entre objetos.

Formato de la interfaz ATM

- Se prepara un seguimiento de sucesos para cada escenario.



Seguimiento de sucesos para un escenario de CA.

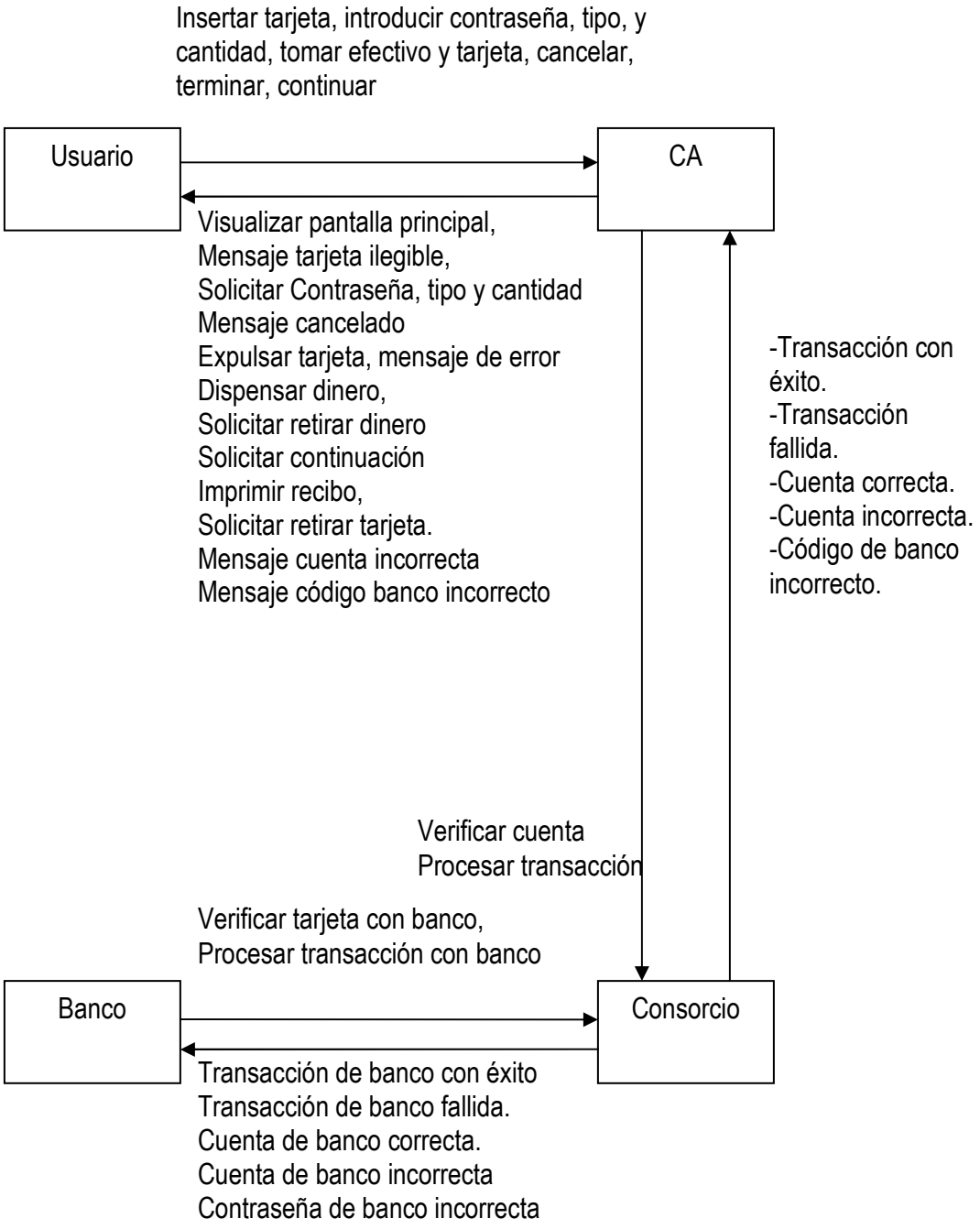


Diagrama de flujo de sucesos para el ejemplo de CA.

- Se construye un diagrama de estados.

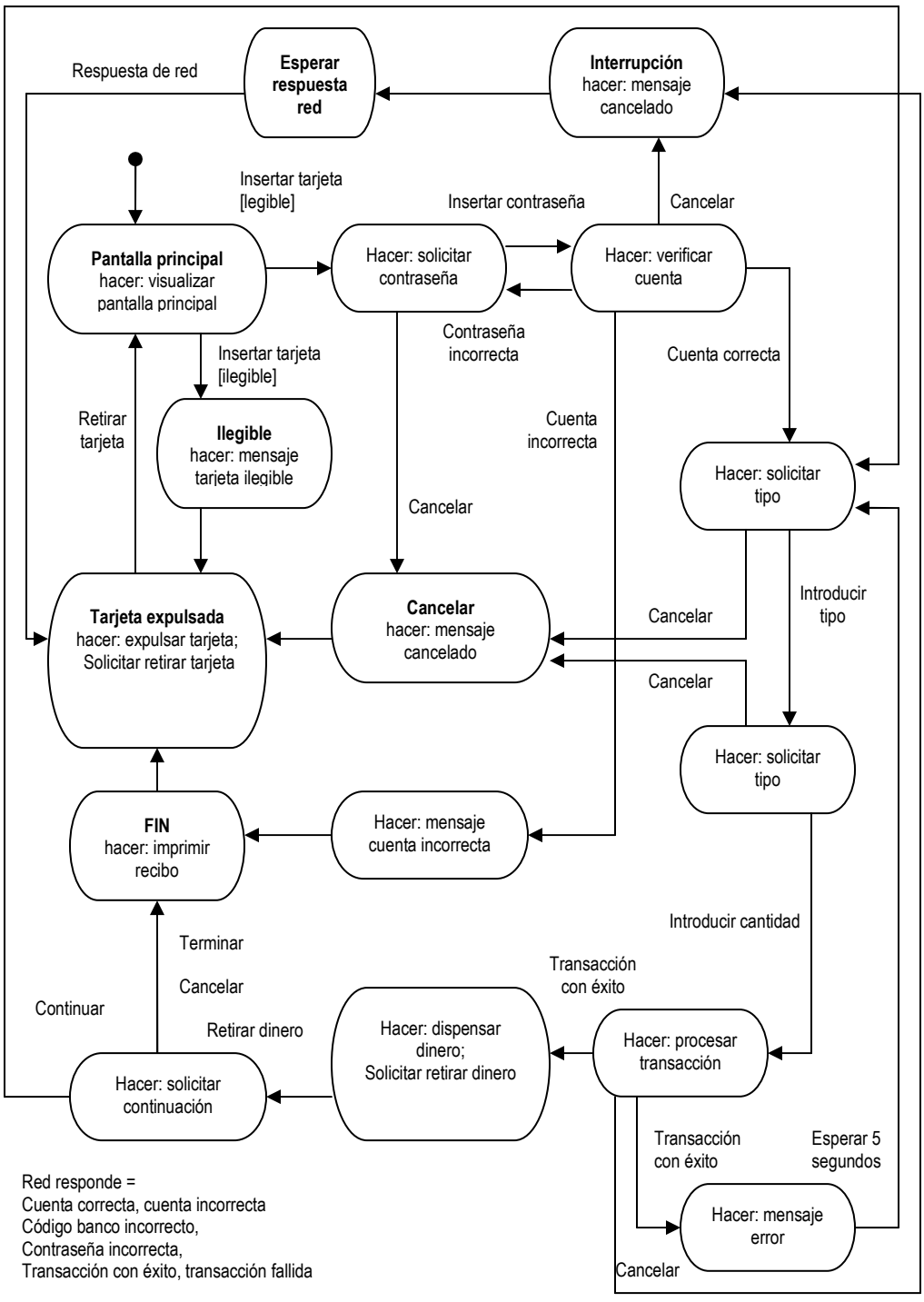


Diagrama de estados para la clase CA

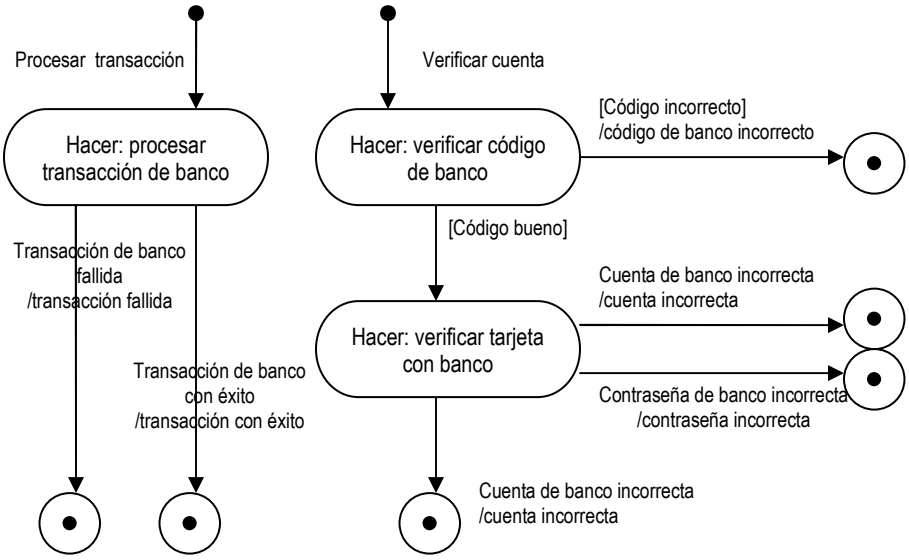


Diagrama de estados para la clase Consorcio.

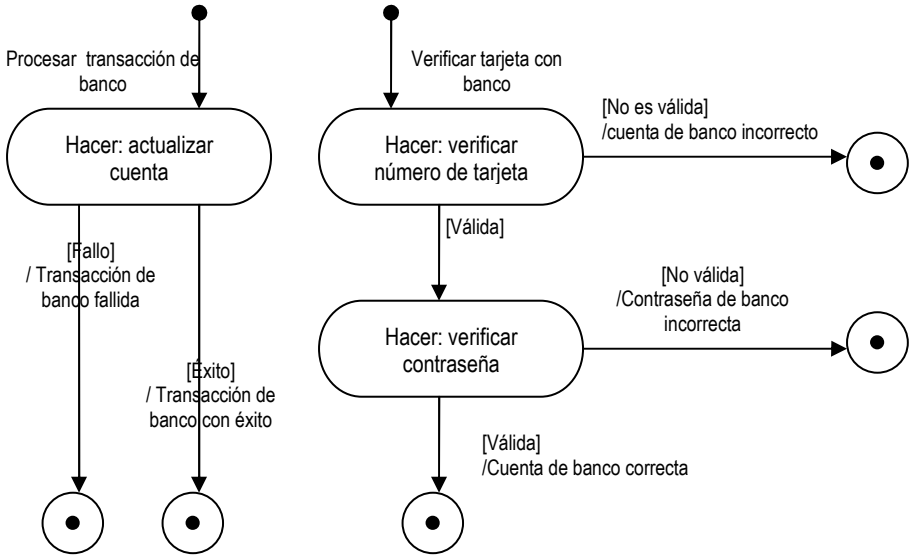
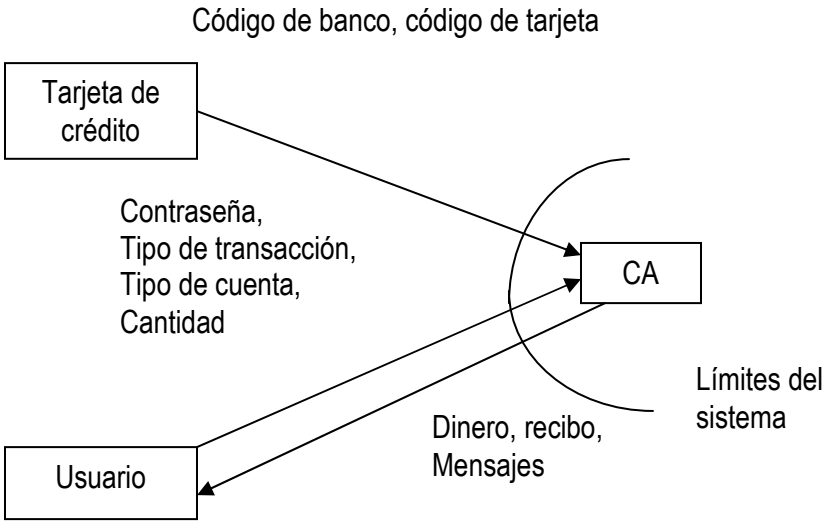


Diagrama de estados para la clase Banco.

- Se comparan los sucesos intercambiados entre objetos para verificar la congruencia.

MODELO FUNCIONAL:

- Identificar los valores de entrada y salida.



Valores de entrada y salida para el sistema CA.

- Construir diagramas de flujo de datos que muestren las dependencias funcionales.

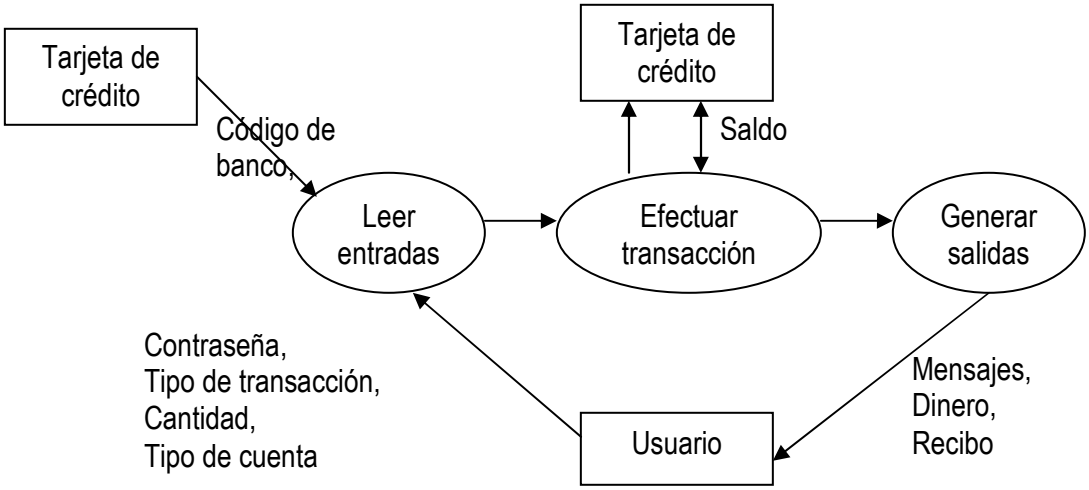


Diagrama de flujo de datos del más alto nivel para el CA

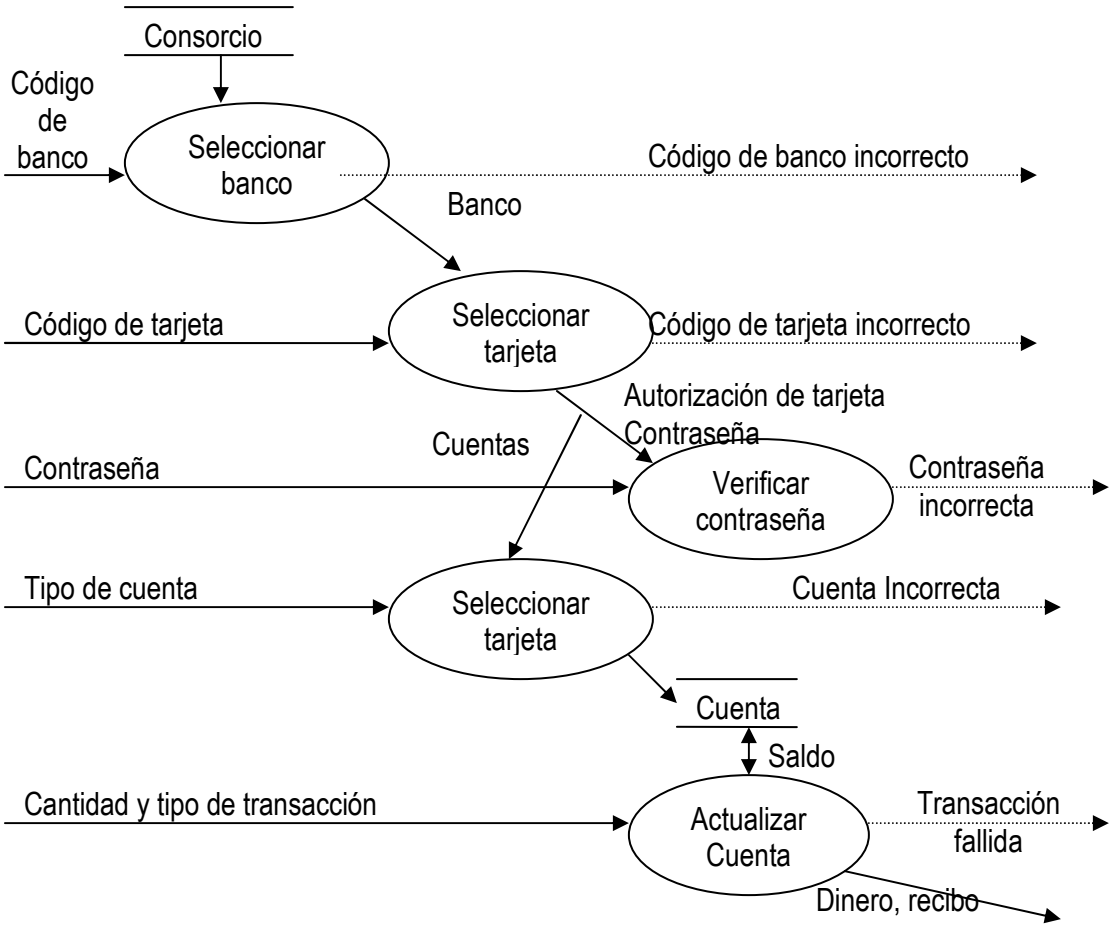


Diagrama de flujo de datos para el proceso efectuar transacción en un CA.

- Describir funciones.

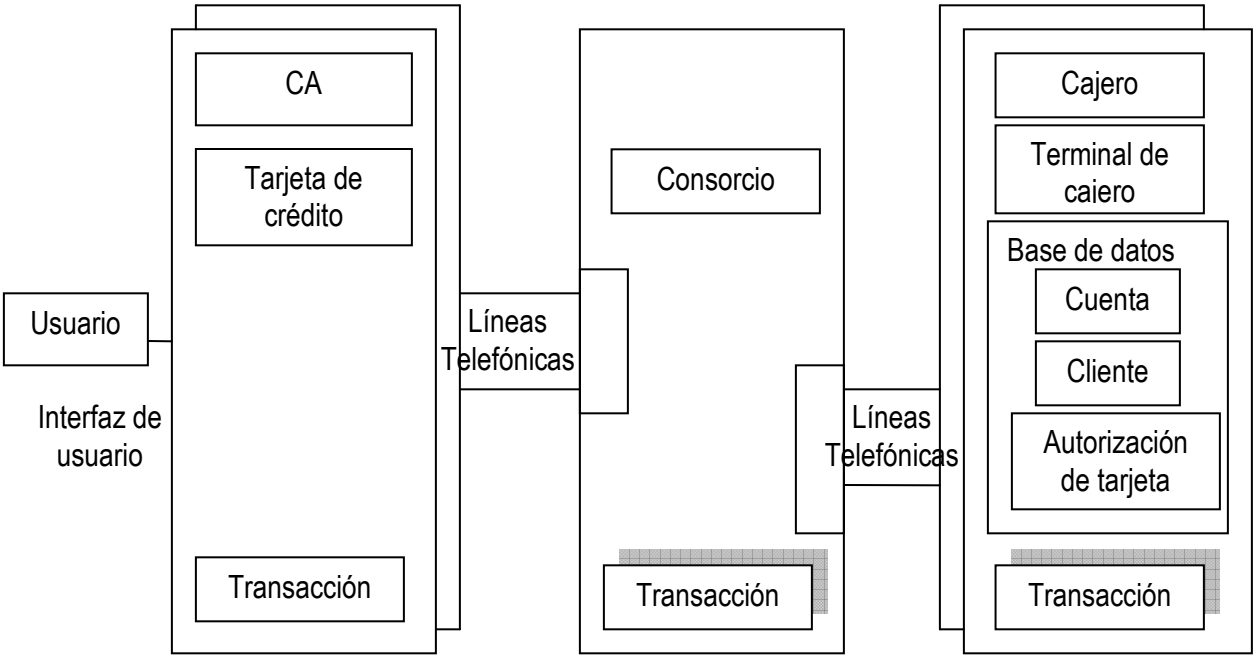
Actualizar cuenta (cuenta, cantidad, tipo-de-transacción) -> dinero, recibo, mensaje
Si la cantidad que se intenta retirar supera el saldo disponible,
    Rechazar la transacción y no entregar ningún dinero.
Si la cantidad que se intenta retirar no supera el saldo disponible,
    Cargar el importe y dispensar el efectivo solicitado
Si la transacción es un ingreso,
    Abandonar el importe y no dispensar efectivo.
Si la transacción es una petición de saldo
    No dispensar efectivo.
En todo caso,
El recibo muestra el número del CA, fecha, hora, número de cuenta, tipo-de-transacción, importe (si lo hubiere) y nuevo saldo.

Descripción de la función actualizar cuenta.

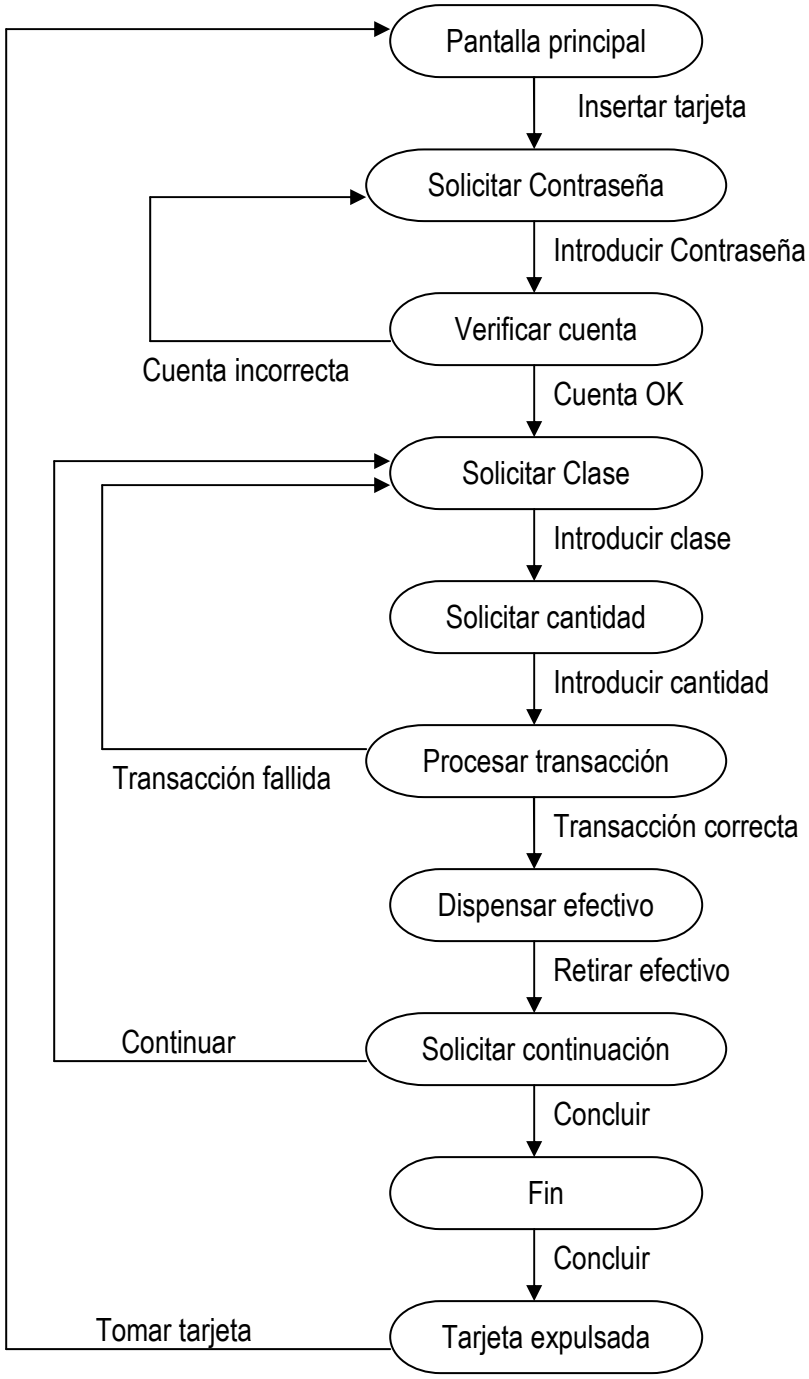
- Identificar las restricciones.

- Especificar los criterios de optimización

DISEÑO



Arquitectura del Sistema CA



Control de un CA

Pseudocódigo

```
Hacer para siempre
    Mostrar pantalla principal
    Leer tarjeta
```

```
Repetir
        Pedir contraseña
        Leer contraseña
        Verificar cuenta
Hasta que la verificación de cuenta sea correcta
Repetir
        Repetir
                Preguntar clase de transacción
                Leer clase
                Leer cantidad
                Comenzar transacción
        Esperar que acabe
Hasta que la transacción sea correcta
Dispensar efectivo
Esperar a que lo tome el cliente
Preguntar si continúa
Hasta que el usuario quiera terminar
Expulsar tarjeta
Esperar hasta que el cliente tome la tarjeta
```

CONCLUSIONES

OMT pone énfasis en la importancia del modelo y uso del modelo para lograr una abstracción, en el cual el análisis esta enfocado en el mundo real para un nivel de diseño, también pone detalles particulares para modelado de recursos de la computadora. Esta Tecnología puede ser aplicada en varios aspectos de implementación incluyendo archivos, base de datos relacionales, base de datos orientados a objetos. OMT esta construido alrededor de descripciones de estructura de datos, constantes, sistemas para procesos de transacciones.

Es muy fácil de aprender ya que para el 90% de casi todos los proyectos se ocupan casi todo el mismo subconjunto de notaciones, además debido a su sencillez se ha extendido a casi todo los niveles de ingeniería de software, pero esta simplicidad del método hace posible que en algunos casos (sobre todo complejos) no se puedan modelar con este sistema.

Recomendamos esta metodología como base para aprender métodos más modernos y profesionales como pueden ser UML u Objectory por mencionar algunos.

Esta investigación dejó en nosotros la importancia y las facilidades que brindan el análisis y diseño orientado a objetos usando una metodología en particular, la cual en nuestro caso fue OMT, la cual debido a la poca práctica con este nuevo paradigma no lo hemos asimilado del todo pero con la práctica de nuestra tercera parte del proyecto quedarán comprendidas todas estas ideas.

OMT 2

En 1994 una revisión de OMT apareció con la introducción formal de los casos de uso. OMT2 declara que los casos de uso están limitados a la etapa de análisis de OMT. Esto requiere una extensión sobre que está definido en OMT añadiendo 2 nuevos modelos a la etapa de análisis:

- ✓ Modelo de dominio. Este modelo es creado explorando el dominio general y adquiriendo conocimiento de las tareas que serán efectuadas
- ✓ Modelo de aplicación. Este modelo es construido sobre el modelo de dominio examinando los casos de uso del dominio.

En esta revisión, los casos de uso son un método para examinar las interacciones de los actores del sistema desde el punto de vista de un usuario.

Además OMT 2 introduce cambios en el modelo de objetos para hacerlo compatible con UML.:

- ✓ Notación de objetos: Objeto como un rectángulo con la siguiente sintaxis nombreObjeto : nombreClase
- ✓ Multiplicidad: La multiplicidad muchos añade información al símbolo de círculo relleno, incorporando información textual adicional
- ✓ Atributos de la relación: Un atributo de la relación es una caso particular de la clase asociación.
- ✓ Clase asociación: La notación es ahora una línea discontinua que une la clase asociación con la asociación.
- ✓ Generalización: La representación aquí es una flecha cuyo extremo “toca” la clase más genérica. Puede organizarse en árbol.
- ✓ Símbolo de cruce: Se introduce para evitar ambigüedades.

OMT 2 está publicado en las páginas blancas de www.rational.com

Bibliografía

Modelado y diseño orientados a objetos Metodología OMT
James Rumbaugh, Michael Blaha, William Premerlani, Frederick Hedí y William Lorensen
Editorial Prentice Hall 1996 Primera reimpresión.

Metodologías orientadas a objetos (Revisión comparativa)
Instituto Tecnológico de Morelia 1999
Monografía presentada por: Helio Bernandino Hernández Ponce

http://fciencias.ens.uabc.mx/~metprog2/cap1.htm
http://fciencias.ens.uabc.mx/~metprog2/cap2.htm
http://fciencias.ens.uabc.mx/~metprog2/cap3.htm
http://fciencias.ens.uabc.mx/~metprog2/cap4.htm

http://fciencias.ens.uabc.mx/~metprog2/cap5.htm
http://fciencias.ens.uabc.mx/~metprog2/cap6.htm
http://www.mcc.unam.mx/~cursos/Objetos/Omt/omt.html
http://www.monografias.com/trabajos6/meto/meto.shtml
http://www.lafacu.com/apuntes/informatica/inge_soft/isw5/default.htm
http://www.csn.ul.ie/~mrmen/UseCaseReport/UseCaseFinalDocument.html
http://exa.unne.edu.ar/depar/areas/informatica/anasistem1/public_html/Temas/Temas_08.pdf