UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLAN"

" METODOS PARA EL ANALISIS Y DISEÑO DE SISTEMAS ORIENTADOS A OBJETOS"

T E S I N A

QUE PARA OBTENER EL TITULO DE:

LICENCIADA EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

MARIA ESTHER MORAN MEJIA

ASESOR: ANTONIO GAMA CAMPILLO

UNAM
MPUS ACATLÁN

DICIEMBRE 1998

Agradecimientos:

A Dios por permitir que siga adelante cada día.

A mis padres Andrea y Pepe por su amor y paciencia con esta hija que los ama.

Dedicatorias:

A Eduardo, mi esposo y mejor amigo, gracias por tu amor y apoyo en todo lo que decido llevar a cabo. Con tu ejemplo me motivas a superarme día a día.

A ti Dany, con todo amor y cariño, por ser mi esperanza en el futuro.

A mis hermanos Carlos, Memo, Pepe y Hugo porque sé que cuento siempre con ustedes.

A mis amigos Maritza y Antonio por darme ánimos y apoyo.

TEMARIO

INTRODUCCION.

Dada la gran competitividad que el mundo moderno presenta hoy en día para los egresados de la carrera de Matemáticas Aplicadas y Computación, en la U.N.A.M. se ha elaborado un plan de estudios que incorpora materias que manejan las últimas tendencias en el campo de la informática, por tal motivo el presente trabajo se desarrolla con el fin de servir como un material de consulta y apoyo didáctico para la asignatura Programación Orientada a Objetos, especificamente para los temas I (Principales modelos en el desarrollo de sistemas orientados a objetos) y III (Desarrollo orientado a objetos). Dicha materia se imparte con carácter obligatorio en el tercer semestre, teniendo como objetivo la introducción a las técnicas de análisis, diseño y programación orientados a objetos para el desarrollo de software de aplicación.

Al conjunto de teorías y reglas que se utiliza para representar el mundo o un problema de manera que sea la más parecida a la realidad se le llama Paradigma de Objetos, teniendo como base a los elementos llamados objetos y clases, lo que va a facilitar la comunicación entre los desarrolladores del sistema y los usuarios pues se maneja un lenguaje común.

La ventaja de este paradigma sobre otros, es que propone el desarrollo de software re-utilizable y de fácil mantenimiento, logrando de este modo una reducción de costos y de tiempo de desarrollo. La base para lograr esto es la modularidad de los sistemas que permite desarrollar cada modulo por separado, teniendo entre si una gran independencia que ayuda a la fácil detección, aislamiento y corrección de errores si se llegaran a presentar.

Al contar con software re-utilizable y de fácil mantenimiento se logra una reducción de tiempo en esta etapa, si se toma en cuenta que para los expertos en sistemas el setenta y cinco por ciento (75 %) del costo y esfuerzo de un sistema se encuentra en la etapa de mantenimiento, por lo que se logra desarrollar proyectos a un menor costo y un máximo aprovechamiento de la nueva tecnología en esta época en la que el software sé esta quedando atrás con respecto al desarrollo del hardware.

Desafortunadamente si no se pone la debida atención al análisis y diseño el resultado será un proceso de desarrollo un tanto desordenado y caótico. Desafortunado, pero comprensible por las discontinuidades que existen entre el análisis estructurado, el diseño estructurado y la programación estructurada que le han dado al análisis y al diseño una desfavorable reputación, pues algunas veces el diseño debe realizarse por segunda vez, lo que provoca que muchas veces el análisis y el diseño sean considerados como un proceso "burocrático" para llegar al objetivo principal que es la codificación del sistema.

El ciclo de desarrollo del software orientado a objetos es muy sencillo si es comparado comparamos con otros, el de cascada tradicional, por ejemplo (ingeniería del sistema, análisis, diseño, codificación, prueba y mantenimiento) pues solo se tienen tres etapas: análisis, diseño e implementación.

Los cambios entre una y otra etapa son muy pequeños, lo que permite una mayor continuidad y una menor perdida entre las transiciones, además de que es posible aplicar las teorías de prototipos entre cada una de las etapas. En resumen, al usuario se le provee un software que va a satisfacer todas sus necesidades, ya que el sistema final no sufrirá alteraciones al ser adaptado entre las etapas y se podrá ir ajustando durante todo el ciclo de desarrollo a los deseos y necesidades del usuario.

Los avances en el desarrollo del software orientado a objetos tiene complejas y profundas "raíces", pues el concepto de herencia que sé vera en el primer capítulo fue explorado por la inteligencia artificial a mediados de la década de los 70´s. Sin embargo para mucha gente, la orientación a objetos empieza y termina con Smalltalk, pero con la aparición de Objective-C y C++ se ha logrado cambiar esta idea. La programación orientada a objetos ha estado sujeta a muchos estudios que describen el principio de los avances en lo referente a la orientación a objetos y su incorporación a los lenguajes de programación como C++, Eiffel y Smalltalk, sin embargo se ha tenido muy poca atención para el estudio de los métodos orientados a objetos para el desarrollo de sistemas.

Existen muchos métodos de análisis y diseño orientados a objetos para el desarrollo de sistemas, algunos son muy conocidos porque se estudian en algunos libros y otros solo se describen en artículos de revistas especializadas en la programación orientada a objetos. Todos los autores de estos métodos defienden el suyo como el mejor y el más completo, sin embargo los usuarios son los que deben tener la última palabra, por tal razón, para no confundirnos con esta "guerra de métodos" lo primero es conocer cada uno de ellos y valorarlos por lo que son.

Al buscar información en algún libro sobre el análisis y diseño de sistemas OO se debe poner atención qué método está utilizando el autor, y ésta es una de las razones por la que se considera es importante hacer un estudio de los métodos de análisis y diseño orientados a objetos, algunos muy conocidos, algunos no tanto pero todos importantes. Algunas otras razones son las siguientes:

- Ayudar a elegir un método.
- Descubrir las diferencias entre los métodos.
- Descubrir si existe la compatibilidad entre los métodos.
- Descubrir los puntos débiles de cada método, si es que los tienen.

En él capítulo uno se dan las definiciones de los principales conceptos (abstracción, encapsulamiento definición de análisis y diseño, herencia y polimorfismo) que son la base del paradigma de objetos.

La definición de análisis y diseño orientados a objetos así como el significado de método y metodología se dan en el capítulo dos. En este capítulo también se describen las principales actividades de los métodos OOSA, HOOD, RDD Y AOO/DOO de Coad&Yourdon.

El Object Modeling Technique (OMT) es uno de los métodos más conocidos desarrollado por Rumbaugh, y tiene coma base el modelo entidad-relación, el modelado y diagramas de clases, herencia, comportamiento, agregación, generalización y multiplicidad. Este método se estudia en él capítulo tres.

El método OOD (Object-Oriented Design) fue desarrollado por Grady Booch en 1983 siendo uno de los pioneros en el desarrollo de métodos para el análisis y diseño O-O de sistema. Sin embargo, el método de Booch ha evolucionado y tenido mejoras (1986,1987) hasta obtener su última versión en el año de 1991. En él capítulo cuatro se explica detalladamente este método.

En el capítulo cinco se estudia el Método FUSION. En este método se incluye lo mejor de los métodos de Booch y Rumbaugh haciéndolo interesante para su estudio.

CAPITULO 1.

Términos y definiciones.

1.1. ¿Qué significa el término Orientado a Objetos?

El término Orientado a Objetos frecuentemente se utiliza para describir las técnicas de análisis, las técnicas de programación o una combinación de las técnicas de programación y las de diseño, sin embargo, la respuesta va a depender del sentido que se le dé a la pregunta, es decir, si se refiere a los lenguajes de programación, a una interfaz de usuario, a una aplicación, a una base de datos, o a un método de análisis y diseño. De manera general, se puede decir que "algo" es orientado a objetos si puede ser modificado para lograr su crecimiento y evolución a partir de sus componentes originales y a través de perfeccionar su comportamiento.

Con el paradigma orientado a objetos, el analista y el diseñador puede hacer uso de su intuición que sirve de guía en la interacción que se tiene con el mundo real que se percibe a través de ese "algo", es decir, los objetos y de la forma en que estos objetos se comportan lo que ayuda a tener una manera más natural de pensar acerca de los sistemas. Pero para poder comprender la orientación a objetos es necesario responder a la pregunta básica ¿qué son los objetos? y definir otros conceptos básicos referentes a este paradigma.

Un ***objeto*** es una parte del modelo de un problema del mundo real que puede ser una persona, una cosa o un lugar. Un objeto tiene características, es decir, sus ***atributos*** (estructura de datos) y un ***comportamiento*** (operaciones o servicios). Pero ¿qué es un modelo?

Rumbaugh da una clara definición de modelo: " Un modelo es una abstracción de algo para el propósito de comprenderlo antes de construirlo." [1]. Con la abstracción de los modelos sólo se consideran las características importantes de algo y se omiten los detalles no esenciales pues de esta manera resulta más fácil manipular ese algo. Como se puede ver este es el principal objetivo que se tiene al utilizar modelos, sin embargo los modelos también ayudan a realizar una evaluación de las características físicas antes de la construcción. En el pasado ya se utilizaban modelos para conocer la resistencia de las estructuras de las construcciones que se iban a desarrollar; también se han utilizado para la construcción de aviones, carros, etc., aunque en la actualidad se cuenta con la ayuda de la simulación de cosas y situaciones gracias a los avances en el campo de la computación, de cualquier manera ya sean modelos o simulaciones se logra reducir el costo de cualquier proyecto, además de que se logra tener una comunicación directa e intercambiar ideas y sugerencias con el cliente durante la fase del modelado del sistema.

Volviendo a los objetos, estos pueden ser agrupados en clases. Una ***clase*** de objeto se utiliza para describir a un grupo de objetos que se consideran del mismo tipo. En las clases está el formato general de los atributos y operaciones o servicios que se incluirán en cada uno de los objetos miembros de la clase. Aunque los objetos de una clase tengan los mismos atributos y operaciones, cada uno de los objetos es único, pues así como los seres humanos poseen los mismos "atributos" que pueden ser dos ojos, nariz, peso, estatura, etc. no todas las personas son iguales en peso, estatura, etc., por lo que se dice que todo objeto es una ***instancia*** o elemento individual de una clase en donde las instancias guardan su estado propio y sus datos internos separados de las otras instancias de su clase.

---

[1] Rumbaugh/Blaha. **Object Modeling and Design**. pag. 15.

Como se ha mencionado antes, un objeto tiene una estructura de datos y un conjunto de operaciones que describen de manera especifica como pueden ser manipulados o controlados los datos del objeto. Las operaciones al ser implementadas reciben el nombre de ***métodos,*** es decir, la forma en que se manipulan los datos. ¿Pero cómo se pueden comunicar entre si los objetos, si los métodos de un objeto sólo pueden y deben hacer referencia a las estructuras de datos de su objeto?.

Cuando un objeto desea utilizar la estructura de datos de otro objeto la comunicación se lleva cabo a través de un ***mensaje*** que contiene las operaciones que son requeridas y una lista de valores de los argumentos de entrada y salida (parámetros). Cada mensaje de un objeto esta asociado con uno o más métodos.

Los términos método y mensaje se usan comúnmente como sinónimos, sin embargo esto no es correcto, ya que el mensaje le dice a los objetos "qué" es lo que se necesita hacer, es la petición que se le hará a un objeto para que ejecute una acción o se comporte de alguna manera, y desde un punto de vista funcional es la operación que se hace sobre un objeto. El método le dice a los objetos "cómo" se debe hacer, es decir, la especificación de los pasos que se siguen en cada una de las operaciones. Los métodos son los mismos para cada instancia ya que la clase es la responsable de la implementación de los métodos y debe tener definido un método con el nombre de la operación que se le va a aplicar al objeto para que no se presenten errores.

El Object Management Group (OMG) da la siguiente definición de objeto: " Un objeto es una cosa. Esta es creada como la instancia de un tipo de objeto. Cada objeto tiene una identidad única que es diferente e independiente de cualquiera de estas características. Cada objeto ofrece una o más operaciones" [2]

1.2. Abstracción.

La abstracción es un concepto que se utiliza en la vida diaria con el que se ignoran aspectos que no son importantes para resolver un problema o una situación dada, ayudando a que se centre la atención en aquellos aspectos que si lo son.

En el caso especifico del paradigma O-O, la abstracción ayuda a determinar aquellos objetos que son importantes para el dominio del problema, así como los aspectos y características que tienen en común los objetos que interrelacionan en el sistema, ya que de otra manera solo se podrían ver las características que hacen que los objetos sean muy distintos entre si, por lo que se puede decir que la abstracción ayuda a manejar la complejidad de los objetos.

1.3. Encapsulamiento.

El encapsulamiento también llamado ocultamiento de la información es la acción de empaquetar datos y métodos (operaciones y servicios) con el propósito de ocultarlos de los demás objetos para que de esta forma se permita el acceso a sus datos mediante el uso de sus métodos propios. Con el encapsulamiento el usuario puede hacer uso de las operaciones pero no cómo se llevan a cabo.

Otro aspecto importante del encapsulamiento es que separa el comportamiento del objeto (operaciones) de su implementación, lo que permite tener una mejor evolución del modulo, ya que solo se modifica la codificación de un objeto sin que se tenga que modificar las aplicaciones que lo utilizan.

---

[2] Object Analysis and Design. volumen 1. pag. 32. Object Management Group. october 1. 1992.

1.4. Herencia.

La herencia define clases especializadas de una clase más general, en donde la clase especializada incluye la estructura de datos y las operaciones de la clase general además de poseer sus características propias. Esto se puede hacer debido a que algunas clases están organizadas de manera jerárquica y de esta forma logran relacionarse. Existen dos tipos de clases: las subclases, que son las que heredan de alguna otra clase, y las superclases que son de las que se hereda.

Algunos métodos orientados a objetos permiten que una subclase tenga solo una superclase, lo que se denomina "herencia simple". Otros métodos permiten que una subclase tenga más de una superclase, es decir, "herencia múltiple".

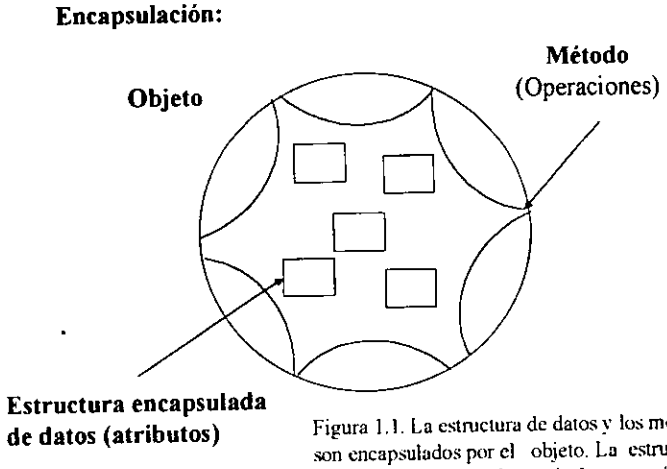

Figura 1.1. La estructura de datos y los métodos son encapsulados por el objeto. La estructura solo podrá ser accesada través de sus métodos propios.

Con la ayuda del concepto de herencia se puede describir al objeto estableciendo la clase general o clases a las que pertenece junto con aquellas características propias que lo hacen único.

La herencia contribuye con el modelado orientado a objetos de un sistema ya que da como resultado una red de objetos comunicados mediante mensajes donde cada objeto especifica sus datos y operaciones, y tal vez comparta propiedades de acuerdo a la clasificación jerárquica.

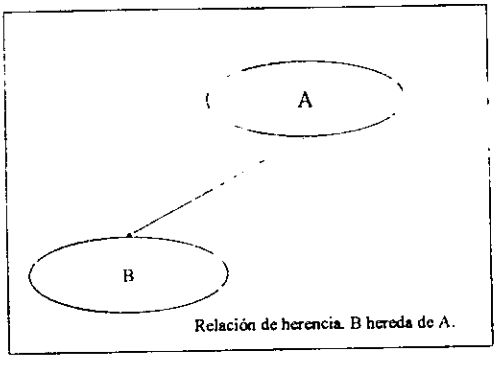

Figura 1.2. Herencia.

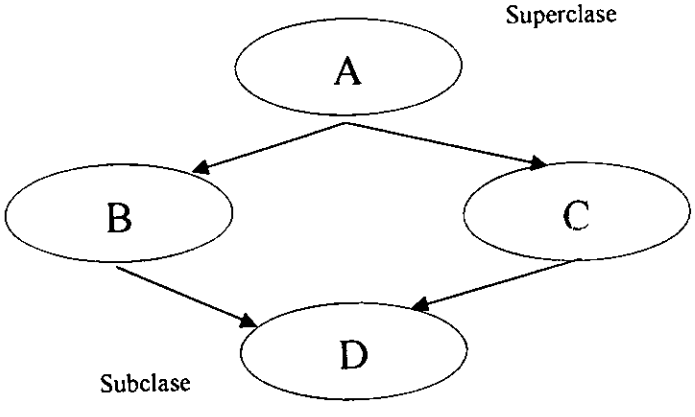

Figura 1.3. Herencia Múltiple.

1.5. Polimorfismo.

El polimorfismo es una característica del paradigma orientado a objetos que permite que un objeto se pueda comportar de diferentes maneras dependiendo de la información (tipo de dato) que se le proporcione a través de un mensaje, esto es, se puede tener un mismo nombre de operación con varias implementaciones. En Office se puede decir que una rutina de impresión es polimórfica, ya que esta rutina se utiliza para especificar una clase general de acciones, en este caso son el imprimir caracteres de texto, números y gráficas, y la rutina sabe la manera en la que va a imprimir tanto un texto como una gráfica dependiendo de la información que se le especifique.

CAPITULO 2.

Metodología para el desarrollo de sistemas orientados a objetos.

En este capítulo se describen las actividades de algunos métodos de análisis y diseño orientados a objetos, por lo que es importante definir primero que es el análisis y diseño orientados a objetos así como el significado de la palabra método y algunos conceptos relacionados con el desarrollo de software.

2.1. ¿Qué es el análisis orientado a objetos (AOO)?

Al análisis se le define en forma general como el proceso de especificar los requerimientos de uso y funciones del sistema independientemente de la implementación y de la descomposición física de los módulos o componentes.

En el mundo real se puede ver que muchas aplicaciones reales tienen componentes que son muy similares entre sí, sin embargo en su manera de funcionar (funcionalidad) son diferentes, pero con la ayuda del análisis orientado a objetos estos componentes pueden ser fácilmente modelados.

El análisis tradicional y el orientado a objetos no difieren mucho entre sí, ya que en ambos casos el objetivo es determinar cuales son los elementos que se deben considerar en un sistema y que es lo que debe de hacer el sistema.

Hablando específicamente del análisis orientado a objetos, el objetivo es desarrollar una serie de modelos que describen al sistema como el trabajo que satisface un conjunto de requerimientos que definidos por el usuario. Se identifican las clases y los objetos que forman parte de un sistema, por lo que se puede decir que el AOO es una actividad de clasificación.

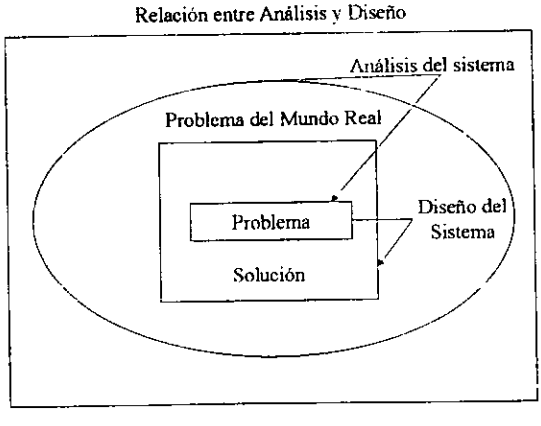

Figura 2.1. Relación entre Análisis y diseño

Para esta actividad de clasificación no hay que olvidar que se requiere del descubrimiento de las clases y objetos, aunque los analistas no tienen que ser unos expertos en el descubrimiento de las clases y los objetos ya que es usual que trabajen en equipo con los expertos en el dominio del problema.

Durante la etapa del diseño es importante la invención de nuevas clases y objetos que se derivan del dominio de la solución.

En el AOO se deben de hacer las siguientes preguntas:
- ¿ Cuál es el comportamiento deseado del sistema?
- ¿Cuáles son los roles y responsabilidades de los objetos que soportaran este comportamiento del sistema?

En un sistema se pueden considerar tres aspectos:
1) El dato u objetos o conceptos y sus estructuras.
2) Proceso o comportamiento local.
3) Control o comportamiento global del sistema.

Considerándolos de manera tradicional se puede ver que los datos y los modelos de entidades trabajan con el dato; los diagramas de flujo de datos y la descomposición funcional tratan con el proceso y los diagramas de transición de estados con el control.

Con el paradigma OO se combinan dos de esos aspectos (dato y proceso) encapsulando el comportamiento local con el dato por lo que el AOO va de lo particular (o clases) a través de lo individual (o instancias) a lo general (control).

2.2. ¿Qué es el diseño orientado a objetos (DOO)? ?

El diseño consiste en transformar la representación del sistema en una representación de la solución.

Al hablar del diseño tradicional y el DOO también se puede hablar de algunas diferencias que existen entre ellos. En el caso del diseño tradicional si no se utiliza alguna herramienta CASE, la actividad de pasar los requerimientos obtenidos a diagramas se convierte en una tarea difícil y va a depender más de la habilidad y experiencia de la persona que lo realiza que de una metodología. En caso contrario, con el DOO se siguen manejando los mismos términos en el mismo ambiente dejando de ser importante el hecho de que si se han utilizado o no las herramientas CASE.

El objetivo del DOO es transformar el modelo del análisis a un modelo de diseño que sirve como un detallado plan de acción para la construcción del software.

El DOO consiste en relacionar las clases y los objetos que se identificaron en la etapa del AOO para crear las jerarquías y mecanismos que logran el comportamiento deseado del sistema por ensamblar, lo que implica agregar todo lo antes realizado sin la necesidad de tener que rehacerlo todo, evitando de esta forma los posibles errores que se pueden originar al realizar la transición del AOO al DOO.

Durante el DOO se "inventan" las abstracciones y mecanismos que proveerán el comportamiento que el sistema requiere, entendiendo como mecanismo a cualquier estructura a través de la cual los objetos colaboran para proveer algún comportamiento que satisfaga los requerimientos del problema.

Beneficios del DOO:
- Los cambios de diseño que se lleguen a requerir son localizados fácilmente y las interacciones no anticipadas con otros módulos del programa son improbables.
- Las áreas de datos compartidos son eliminadas reduciendo de este modo la posibilidad de modificaciones no anticipadas.

Los métodos para el diseño orientado a objetos comparten algunos pasos básicos del diseño y solo varían un poco:

- Identificar los objetos del espacio de la solución y sus atributos así como el nombre de los métodos (servicios u operaciones)
- Establecer la visibilidad que tendrá el objeto en relación con los otros objetos.
- Determinar la interfaz de cada objeto y sus responsabilidades.
- La implementación y test de los objetos.

Durante la etapa del diseño es importante hacer ciertas preguntas relativas a la arquitectura del sistema:

- Que clases existen y como se relacionan.
- Que mecanismos se utilizan para regular la forma en que los objetos colaboran.
- Donde es recomendable que cada clase y objeto se declaren.

El AOO y el DOO tienen mucho en común. Tiene los mismos conceptos y técnicas, y las notaciones utilizadas en el análisis se aplican de igual manera en el diseño lo que hace que las herramientas para el desarrollo sean utilizadas en ambas actividades. Esta similitud hace que algunas veces sea difícil determinar qué actividad esta siendo soportada por la otra. Cuando se tenga esta confusión se puede decir que si la actividad tiene que ver con la especificación de alguna parte del problema y el usuario o experto tiene algún trato con dicha actividad, entonces se trata de la etapa del análisis; si la actividad tiene que ver con el desarrollo de parte de la solución entonces se trata del diseño.

2.3. Métodos para el análisis y diseño orientado a objetos.

2.3.1. Definición de "Método".

Antes de entrar al estudio del tema Métodos para el AOO y DOO es importante definir primero la palabra "método" ya que muchas veces se utiliza como sinónimo de la palabra metodología, lo cual es un gran error, pues se puede decir de manera general que la metodología es el estudio de los métodos.

Pero bien, ¿qué es un método? Para Grady Booch: " Un método es un proceso disciplinado para generar un conjunto de modelos que describe varios aspectos de un sistema de software bajo desarrollo usando una notación bien definida. Una metodología es una colección de métodos aplicados a través del ciclo de vida del desarrollo y unificado en forma general en un aproximamiento filosófico." [3]

Existen tres aspectos en el desarrollo de software que algunas veces son confundidos, que están relacionados entre sí pero que son diferentes: modelo de proceso, método y notación.

El *modelo de proceso* se define como una colección de estrategias, actividades y métodos que se organizan para llevar a cabo un grupo de objetivos.

El *método* es el conjunto de conceptos, técnicas y pasos que se utilizan para llevar a cabo una actividad del modelo de proceso.

La *notación* se utiliza para lograr un resultado a partir de la aplicación de un método. En la figura 2.2. se muestra la relación que existe entre los modelos de proceso, métodos y notaciones.

---

[3] Grady Booch. Object-Oriented Analysis and Design with Applications. pag. 18.

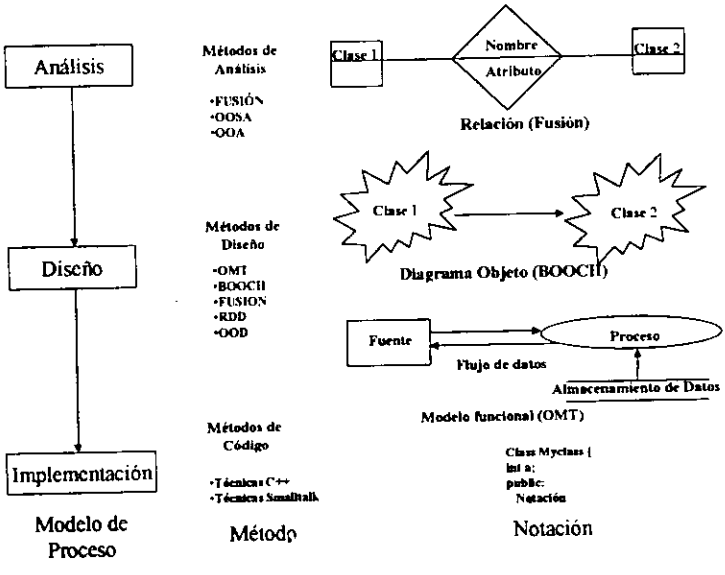

Figura 2.2. Relación entre modelo de proceso, métodos y notaciones. A la izquierda de la ilustración se muestra las posibles actividades de un modelo de proceso, en medio se dan algunos ejemplos de métodos que soportan estas actividades y a la derecha se muestra algunos ejemplos de notaciones que ayudan a capturar el resultado de los métodos.

2.3.2. Principales características de algunos métodos O-O.

2.3.2.1. Método Object-Oriented Systems Analysis (OOSA) por Shlaer & Mellor.

El método Object-Oriented Systems Analysis fue desarrollado por Sally Shlaer y Stephen J. Mellor y su principal objetivo fue desarrollar un método con el que se pueda tener una visión previa acerca de las definiciones de objetos propuestas y poder examinar las implicaciones de esas definiciones. La principal técnica utilizada por Shlaer & Mellor es conocida como Modelado de la Información que sirve como ayuda para la formalización de toda la información.

La característica principal de este método es que se utiliza un conjunto de modelos que pueden ser ejecutados para su verificación y obtener un aproximamiento al diseño a través de una traslación de dichos modelos de análisis. Dicha aproximación ayuda al proceso a lograr una suave y predecible transición entre las fases del análisis, el diseño y la implementación.

La base de la estructura de este método es la identificación inicial de los principales sujetos independientes y separados llamados "dominios" (domains). Estos dominios pueden ser particionados en subsistemas y utilizar diagramas de alto nivel para facilitar el trabajo cuando se tienen proyectos grandes.

PROCESO DEL METODO.

Los pasos a seguir en el proceso de este método son los siguientes:

1. Partición del sistema en dominios.

2. Análisis del dominio de aplicación.
3. Verificación del análisis a través de la ejecución.
4. Especificación de la traslación de los modelos del análisis.
5. Construcción de los componentes de arquitectura.
6. Transformación de la aplicación en arquitectura
7. Transformación de la arquitectura en implementación

PASO 1. Partición del sistema en dominios.
Primero se divide el sistema en sujetos independientes y diferentes llamados dominios. Existen cuatro tipos de dominios que se distinguen por su propósito en el desarrollo.

1) ***Dominio de aplicación.*** Esta relacionado con el uso final del sistema, es decir, lo que el usuario desea que haga el sistema.
2) ***Dominios de servicio.*** Son dominios más generales que se requieren para soportar al dominio de aplicación y pueden ser interfaces de usuario, alarmas, etc.
3) ***Dominio de arquitectura.*** Provee los mecanismos y estructuras para el manejo de los datos y el control del sistema como un todo. Especifica la manera en la que se van a organizar y accesar los datos, cómo se van a manejar las líneas de control, etc.
4) ***Dominios de implementación.*** Provee las entidades conceptuales con las que se va a implementar el sistema, tales como: sistemas operativos, lenguajes de programación, redes y librerías de clases.

Los dominios se organizan en relaciones de cliente- servidor donde un dominio es visto como cliente y puede confiar en otro dominio servidor que le va a proveer los mecanismos y los servicios que necesita.

Esta relación cliente-servidor también ocurre entre dos clases cuando él módulo de una clase (clase A) invoca una operación publica de otra clase (clase B). La clase A que es la que lleva a cabo la invocación es llamada cliente y la clase que se invoca (clase B) se le llama servidor.

En la figura 2.3. se puede ver la representación gráfica del resultado que se obtiene al final de este paso en donde los óvalos representan a los dominios y las flechas las relaciones de cliente-servidor.

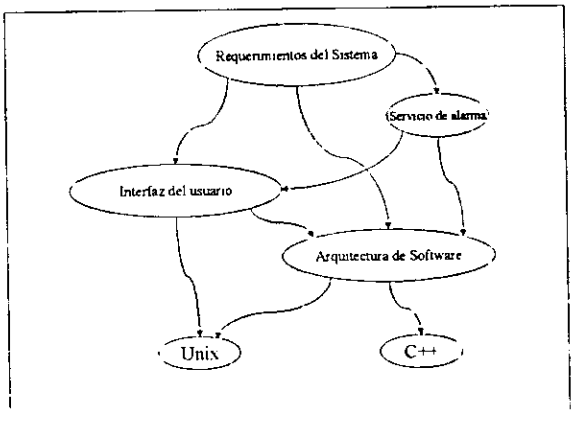

Figura 2.3. Relaciones entre los dominios.

El diagrama de los dominios es el primer paso en la partición del sistema donde cada dominio es considerado como un sujeto individual e importante que es analizado de manera separada de los otros dominios, pero se puede dar el caso de que el dominio sea demasiado extenso para ser analizado como unidad por un solo grupo de trabajo, por lo que es necesario que a su vez sea dividido en pequeños subsistemas que serán analizados por equipos pequeños de analistas.

PASO 2. Análisis del dominio de aplicación.
El siguiente paso es analizar el dominio de aplicación utilizando el análisis orientado a objetos de Shlaer-Mellor. Este análisis se describe con la ayuda de los siguientes tres modelos:

1. Modelo de Información.

2. Modelos de Estado.

3. Modelo de Proceso.

1. Modelo de Información.
En este modelo se definen los objetos (entidades) del dominio que son importantes en el mundo real y para el sistema; las asociaciones que existen entre los objetos se formalizan como relaciones que se basan en las políticas, reglas y leyes físicas que suceden en el mundo real.

Para la construcción de este modelo se llevan a cabo cuatro actividades:

Actividad 1. Identificación de los objetos.
Se identifican a los objetos y sus atributos para su clasificación.

Actividad 2. Definición de los identificadores.
Un identificador es el conjunto de uno o más atributos con el que se distingue cada instancia de un objeto. En el modelo de información para una universidad, por ejemplo, el número de cuenta sería el identificador pues es único y diferente para cada alumno.

Actividad 3. Descripción de los atributos.
Se hace la descripción de cada uno de los atributos. Las relaciones entre los objetos se modelan y se formalizan colocando los atributos en los objetos correspondientes.

Actividad 4. Especificación final de las relaciones.
Se determina y especifica que algunas relaciones son consecuencia de la existencia de otras relaciones, describiendo a los subtipos y supertipos que dan como resultado las relaciones de herencia.

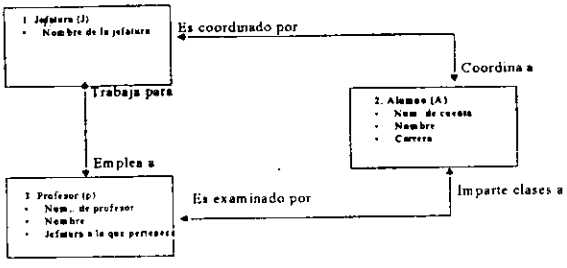

Figura 2.4. Modelo de Información.

2. Modelos de Estado.

Los modelos de estado describen el ciclo de vida (comportamiento) de cada uno de los objetos, construyendo un modelo de estado por cada uno de los objetos que fueron definidos en el modelo de información. No se construyen modelos de estado para objetos que no tengan un comportamiento dinámico y tal vez sea necesario construir un modelo de estado adicional si se manejan objetos con conflictos.

Los conceptos utilizados en estos modelos son:

- ***Acciones.*** Actividades asociadas a un estado que ocurren cuando un objeto arriva al estado.
- ***Estado.*** Indica la condición en que se encuentra un objeto.
- ***Eventos.*** Representan cualquier incidente que causa que un objeto se mueva de un estado a otro, por lo que se puede decir que un evento provoca que ocurra una acción.

El comportamiento de los objetos se describe en un diagrama de transición de estados por cada objeto con comportamiento dinámico.

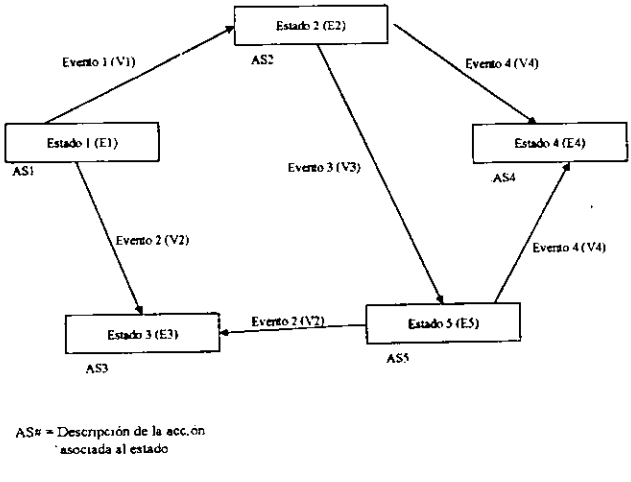

Figura 2.5. Diagrama de Transición de Estado.

Eventos / Estados	V1	V2	V3	V4	V5
Estado 1	E2	E3	*	-	*
Estado 2	-	-	E5	E4	-
Estado 3	-	*	-	-	-
Estado 4	*	-	-	-	*
Estado 5	-	E3	-	E4	-

* Evento ignorado

\- No puede ocurrir

Figura 2.6. Tabla de Transición de Estado.

En el modelado de la información se pone énfasis en que cada objeto, atributo y relación

Eventos	Descripción del evento	Origen	Destino
V1			
V2			
V3			
V4			
V5			

Figura 2.7. Lista de Eventos.

en el modelo se pueden representar de forma no gráfica, pues cada modelo de estado no consiste solo de un diagrama de transición de estado, sino también se puede hacer uso de una tabla de transición de estado que contiene las reglas de transición y una descripción de cada acción en el diagrama de transición de estado (requerida solo si la acción es muy grande para introducirla en el diagrama) y una lista de eventos que muestra a todos los eventos que han sido definidos para todos los modelos de estados.

3. Modelo del Proceso.
A este modelo también se le conoce como Modelo de Especificación de Acciones que ayuda a formalizar el procesamiento requerido en los modelos de estado.

Este procesamiento se divide en procesos que se analizan y se transforman directamente en operaciones que son parte del diseño orientado a objetos.

El propósito principal es obtener procesos fundamentales analizando las acciones de los modelos de estado. Estos procesos se representan en diagramas de flujo de acción de los datos(ADFD).

Para la construcción del modelo del proceso se llevan a cabo cuatro actividades:
Actividad 1. Las acciones se dividen en procesos.
Actividad 2. A estos procesos se les da nombre y una descripción.
Actividad 3. Se construye una tabla de procesos de estado que provee una lista de los procesos en el sistema y las acciones en las que serán utilizado.
Actividad 4. Se construye un Modelo de Acceso de Objetos que muestra la comunicación que ocurre al mismo tiempo entre los modelos de estado en el sistema.

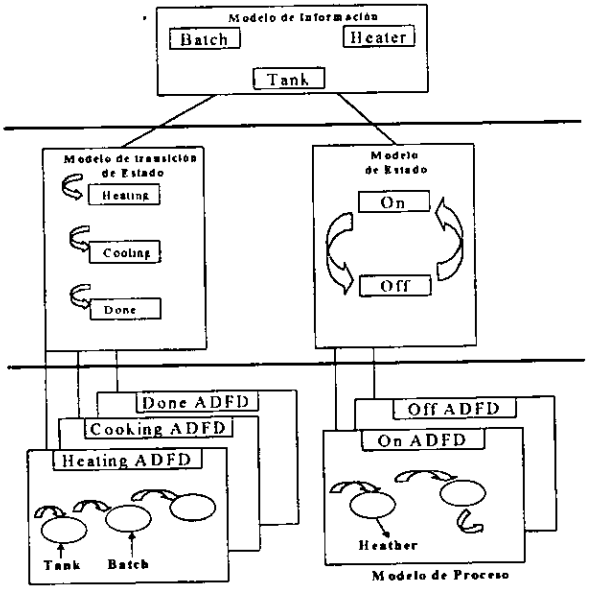

Figura 2.8. Modelos del dominio de aplicación.

Como se puede ver, el analista primero define los objetos al realizar el modelo de información para continuar con la descripción de su comportamiento y finalmente define el procesamiento. En la figura 2.8. se puede observar que este trabajo debe realiza en este orden ya que la decisión acerca de cual especificación de acción se va a construir es determinada por el estado en los modelos de estado, y a su vez la construcción de los

modelos de estado esta determinada por los objetos definidos en el modelo de información.

A partir de estos tres modelos fundamentales se derivan los siguientes modelos. Los primeros tres se refieren a las relaciones que existen entre los subsistemas y los demás se refieren a los subsistemas:

1. Modelo de relación de subsistemas (MRS).
Muestra las relaciones entre los objetos en diferentes subsistemas.

2. Modelo de comunicación de subsistemas(MCS).
Muestra las comunicaciones asincrónicas de eventos entre los objetos en diferentes subsistemas.

3. Modelo de acceso de subsistema(MAS).
Muestra el acceso sincrónico de datos entre los objetos en diferentes subsistemas. Los accesos se derivan a partir de las especificaciones de acción.

4. Modelo de comunicación de objeto.
Muestra la comunicación asincrónica de eventos entre los modelos de estado en el sistema y objetos en otros subsistemas. La información para su construcción se obtiene de la tabla de proceso de estado.

5. Modelo de acceso de objeto.
Proporciona una vista complementaria del modelo de comunicación de objeto y la comunicación sincrónica entre los modelos de estado.

6. Tabla de proceso de estado.
Es una lista de todos los procesos en todos los diagramas de flujo de acción de datos (ADFD). Se utiliza para identificar procesos comunes.

7. Diagrama de línea de control
Muestra el orden de los eventos y estados que son apropiados para las instancias en una línea de control. Una línea de control se define como una secuencia de acciones y eventos que ocurren en respuesta a la llegada de un evento particular no solicitado cuando el sistema esta en un estado en particular.

8. Diagrama de flujo de acción de datos.
Se utiliza para la representación de las unidades de procesamiento dentro de una acción y la comunicación entre esas unidades de procesamiento.

Con la ayuda de las herramientas CASE todos estos modelos se pueden construir.

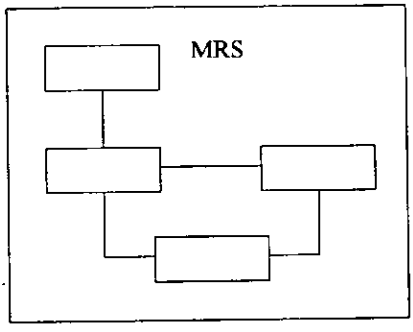

Figura 2.9. Modelo de relación de Subsistemas

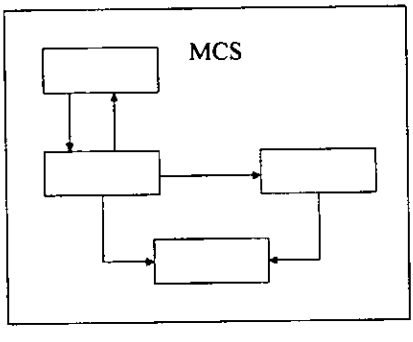

Figura 2.10. Modelo de comunicación de subsistemas. Cada rectángulo representa un modelo de comunicación de objeto de un subsistema en particular, donde las flechas indican en que subsistema se generó el evento y en cual subsistema lo recibe.

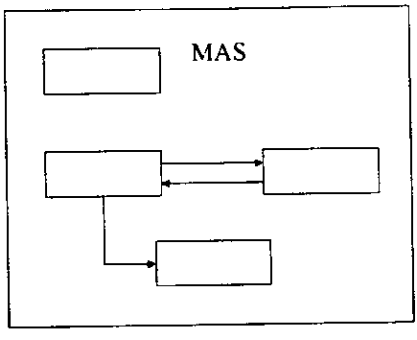

Figura 2.11. Modelo de acceso de subsistemas. Cada uno de los rectángulos representa al modelo de acceso de objeto.

PASO 3. Verificación del análisis a través de la ejecución.
Debido a la integración que existe en estos tres modelos del AOO es posible confirmar el comportamiento del sistema tanto de manera estática como dinámica.

Los tres modelos de manera individual son una unidad tal que cuando se integran definen el comportamiento, el dato y el procesamiento requerido por el dominio. Existe un conjunto de reglas para la sintaxis de los modelos que definen la consistencia del conjunto de modelos y son la base para una verificación estática. Cada una de estas reglas puede ser revisada y llevar a cabo una revisión cruzada entre los elementos que aparecen repetidamente en los modelos. Existen dos maneras de realizar esta revisión:

1. Se permite la entrada al usuario del conjunto de modelos a una herramienta CASE y realizar la revisión de cada regla posteriormente.

2. Solo permite la entrada a la herramienta CASE si se tienen los modelos construidos de manera correcta, lo que repercute en una mayor productividad del analista.

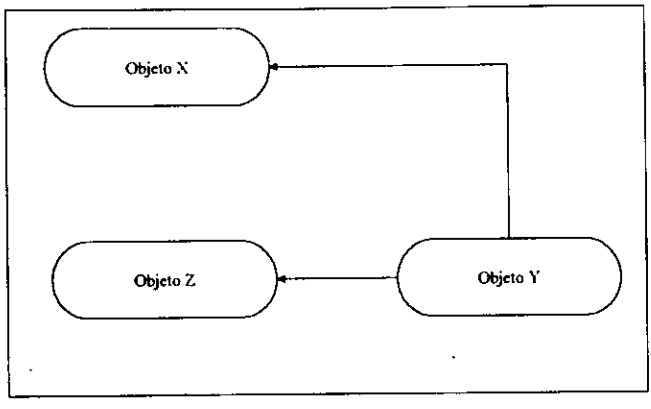

Figura 2.12. Modelo de comunicación de objeto.

Los modelos están de tal manera bien definidos que se puede llevar a cabo una ejecución. En esta ejecución, el dato que es procesado se define en el modelo de información, la secuencia en la que el procesamiento se realiza se especifica en los modelos de estado, y el procesamiento se describe en el modelo de proceso, de tal forma que se puede llevar a cabo una simulación a través del siguiente proceso:

1. Se establece en estado inicial deseado del sistema con valores de dato en el modelo de información.
2. Se inicia un comportamiento deseado con un evento enviado al modelo de estado.
3. Se ejecuta el procesamiento especificado por el modelo de proceso y como una secuencia a través de los modelos de estado.
4. Se evalúa el resultado obtenido y se compara con el resultado que se esperaba.

Cuando el resultado muestra que se ha conseguido el comportamiento deseado, se puede decir que la etapa del análisis orientado a objetos de Shlaer-Mellor ha terminado.

PASO 4. Especificación de la traslación de los modelos del análisis.
Para llevar a cabo esta especificación es necesario hacer una especificación del dominio de arquitectura, pues se van a determinar los mecanismos y estructuras para el manejo de los datos así como la función y control del sistema, además de definir la traslación de los modelos del análisis dentro de esos mecanismos y estructuras. También se especifican los requerimientos necesarios para el acceso a los datos y la transmisión de eventos, las multitareas, requerimientos de sincronización, distribución de plataformas, el tipo y el grado de la flexibilidad deseada.

En este método el reporte de la arquitectura se considera como una aplicación independiente en la que no se elaboran los modelos del análisis incluyendo información del diseño o una reorganización del análisis, sino que se realiza un reporte de los mecanismos y estructuras requeridas para el soporte del análisis orientado a objetos, por lo que la arquitectura debe proveer modos para el almacenamiento de datos, para la transmisión de eventos y para el acceso a los datos.

En el DOO se utilizan los siguientes diagramas:

- *Diagrama de Herencia.* Muestra las relaciones de herencia entre las clases.
- *Diagrama de Dependencia.* Describe las llamadas relaciones cliente- servidor y relaciones de "amigos" que existe entre las clases. Esta relación de amigos ocurre cuando un modulo de una clase (clase A) o cualquier otra invoca una operación interna de la clase B, o tiene acceso directo a los datos de otra clase. Entonces se dice, que la clase A es amiga de la clase B.
- *Diagrama de Clase.* Muestra la vista externa de una sola clase basada en la notación de Booch.
- *Gráfica de Estructura de Clase.* Muestra la estructura interna del código de la descripción de la clase.

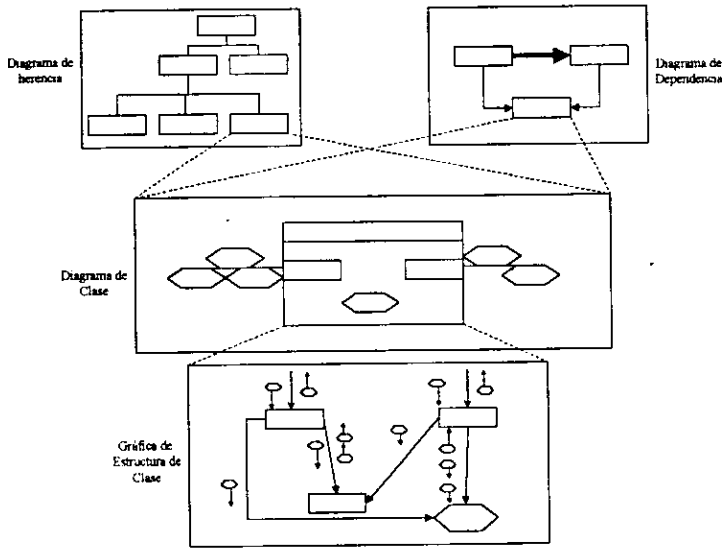

Figura 2.13. Relaciones entre los diagramas.

PASO 5. Construcción de los componentes de arquitectura.
El dominio de arquitectura se especifica con dos tipos de componentes: mecanismos y estructuras.

1. *Mecanismos:* Representan la capacidad especifica de la arquitectura que se debe proveer para realizar el sistema. Estos mecanismos se construyen a través del diseño e implementación del código que toman la forma de tareas y librerías de clase.

2. *Estructuras:* Estas estructuras se construyen a partir de una combinación del código y el dato a la que se le incluye elementos del dominio del cliente como declaración de variables, declaración de operación y código de operación. A esta construcción de estructuras se le llama ARCHETYPE. Estos archetypes son reutilizables y su construcción ofrece oportunidades para la automatización con la ayuda de herramientas CASE.

Shlaer y Mellor sugieren que los diagramas de clase y las gráficas de estructura de clase se deben representar en detalle solo como diagramas de archetype que indican en dónde

se debe ejecutar la especialización a través de una sustitución de nombre. A esta fase se la llama aplicación.

PASO 6. Transformación de la aplicación en arquitectura.
En el dominio de la aplicación que describen los diagramas de archetypes, un diagrama de clase se representa para cada clase y los segmentos de acción se representan con las gráficas de estructura de clase. Un diagrama de clase se considera como de tipo estructural ya que muestra la vista externa de una sola clase.

PASO 7. Transformación de la arquitectura en implementación.
En este paso los módulos de los diagramas archetype se expresan como código de archetype o patrones de código en un lenguaje de programación.

2.3.2.2 Hierarchical Object-Oriented Design (HOOD).

Esté método fue desarrollado por la Agencia Europea del espacio como un método de diseño para ser codificado en ADA aunque esta siendo extendido para codificarse en C++. Contrario a su nombre HOOD no soporta el concepto de herencia, por lo que no se le puede considerar verdaderamente como un método orientado a objetos, sino un método basado en objetos aunque se utiliza de manera comercial como una alternativa.

El proceso principal de este método es llamado el Paso de Diseño Básico donde utilizando técnicas de diseño O-O los objetos "hijo" se identifican a partir de un objeto "padre" dado. Las relaciones entre los objetos hijo con otros objetos son modeladas. Si la búsqueda para cualquier hijo se dirige a una clase "primitiva" o "terminal", entonces se genera el código.

Para soportar el diseño los objetos se clasifican como sigue:

1. *Objetos Activos.*
   Estos objetos pueden ejecutar en forma paralela y usar servicios de cualquier tipo de objetos.
2. *Objetos Pasivos.*
   Solo pueden ejecutar de forma secuencial y solo usan los servicios de otros objetos pasivos.
3. *Objetos de Ambiente.*
   Estos actúan como una interfaz con otros sistemas con los que el sistema debe interactuar.
4. *Objeto Nodo Virtual.*
   Este objeto representa un nodo en un sistema distribuido.
5. *Flujos de Control.*
   Son utilizados para mostrar cómo diferentes objetos de un nivel especifico se comunican.

PROCESO DEL METODO.

El método empieza con la identificación de un objeto para el sistema llamado Objeto Raíz. Este objeto raíz se divide en objetos internos que después serán especificados. Esta subdivisión se realiza de manera recursiva dentro de todos los objetos hasta que se alcanza un nivel que puede ser directamente implementado con paquetes de ADA. Esta estructura jerárquica fue considerada como una necesidad para distribuir el trabajo.

El Paso de Diseño Básico se divide en las siguientes fases y el orden del trabajo se ilustra en la figura 2.14.

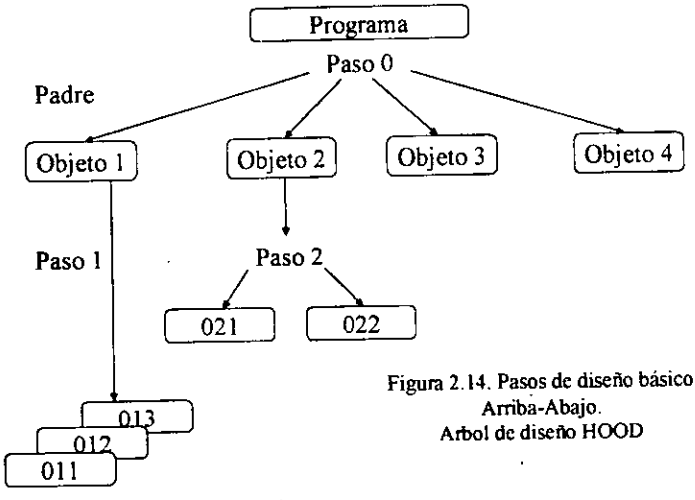

Figura 2.14. Pasos de diseño básico Arriba-Abajo.
Arbol de diseño HOOD

Fase 1. Definición del problema.

Fase 2. Elaboración de una estrategia informal de la solución.

Fase 3. Formalización de la estrategia.

Fase 4. Formalización de la solución.

Este proceso se realiza de manera recursiva dentro de cada objeto hasta que se alcanza el nivel de objeto más bajo, es decir el objeto terminal o final.

FASE 1. Definición del problema.
Se hace un reporte del problema para proveer un contexto para el objeto del nivel actual. Los requerimientos recibidos del objeto padre son analizados y estructurados. El propósito de este análisis es estar seguros de que el problema esta siendo bien entendido.

FASE 2. Elaboración de una estrategia informal de la solución.
Se elabora una solución del problema definido previamente pero a grandes rasgos utilizando el lenguaje natural y explicando el diseño para el nivel actual de abstracción. Esta descripción informal describe el diseño a través del significado de los objetos tomados del mundo real y sus acciones que desarrollan en el mundo real. Es recomendable utilizar un máximo de 10 sentencias.

FASE 3. Formalización de la estrategia.
Esta formalización se realiza en etapas para la identificación y descripción de los objetos y

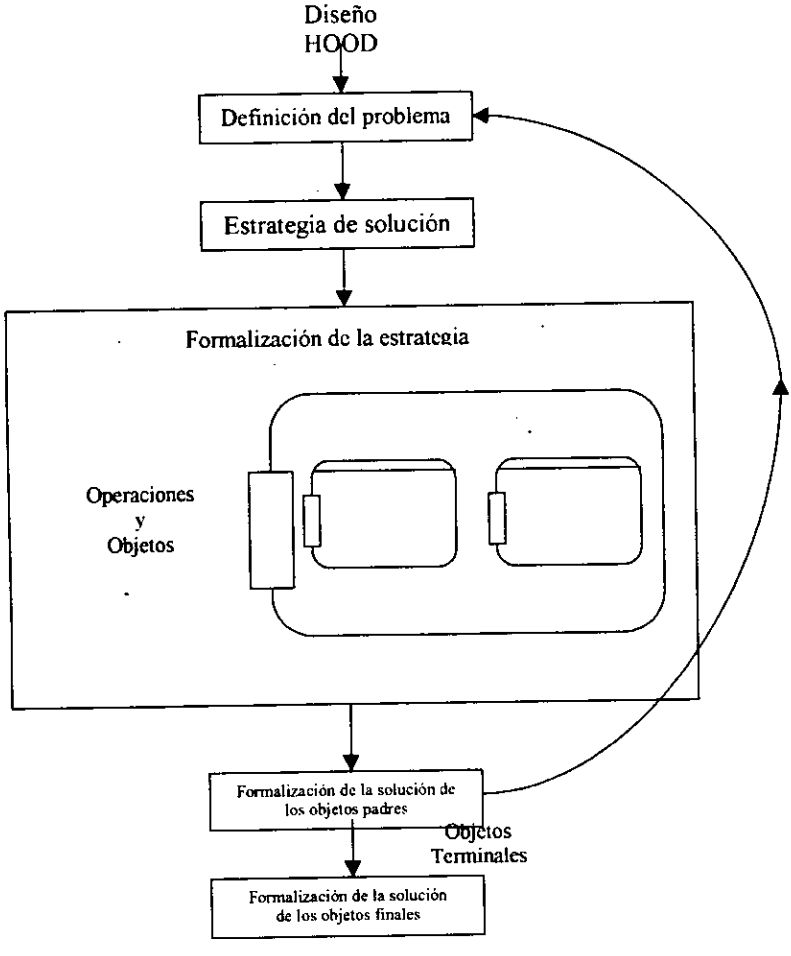

Figura 2.15.Paso de diseño básico del HOOD

sus operaciones, agrupándolos y elaborando diagramas jerárquicos que muestran las relaciones padre-hijo y las operaciones, así como el uso de jerarquías "implementación by" de encadenamientos y flujos de datos.

El desarrollo de esta formalización se divide en cinco pasos:

1) *Identificación de objetos.* Se realiza mediante la extracción de los sujetos de la estrategia informal de la solución.

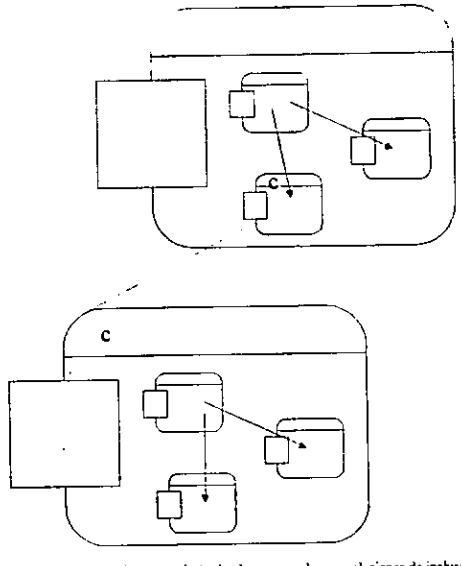

Figura 2.16.HOOD la descomposicion de objetos con el uso y relaciones de inclusion
Las flechas pueden ser identificadas como 'usos' y la inclusion de un paquete
en el diagrama denota que este es parte de objetos incluidos

2) *Identificación de operaciones.* Se hace a partir de la extracción de los verbos tomados de la estrategia informal de la solución.

3) *Agrupación de objetos y operaciones.* Implica la conexión de cada operación con un objeto apropiado.

4) *Descripción gráfica a través del desarrollo de diagramas de datos.* Es la descripción gráfica de los objetos y operaciones utilizando gráficas que siguen el estilo de las gráficas de Booch. En la figura 2.16. se muestra la típica notación que indica la composición jerárquica descrita a través de incluir iconos.

5) *Justificación de las decisiones del diseño.* Esta la elabora el diseñador con el propósito de explicar las razones que tuvo de tomar ciertas decisiones en donde no resultan obvias. En este paso es recomendable hacer un registro del encadenamiento de los requerimientos originales para ser utilizado en caso de que sea necesario un rastreo posterior. Es necesario explicar cierta terminología, ya que en el HOOD a los métodos se les llama "operaciones" y la "implementacion by" se refiere a la necesidad de que las operaciones en los objetos padre invoquen a las operaciones en los objetos hijos.

En el ejemplo ilustrado en la figura 2.17. la operación Start del Objeto-Padre invoca a través de implemented_by a la operación Begin del Objeto-Hijo que va a llevar a cabo la función de la operación Start.

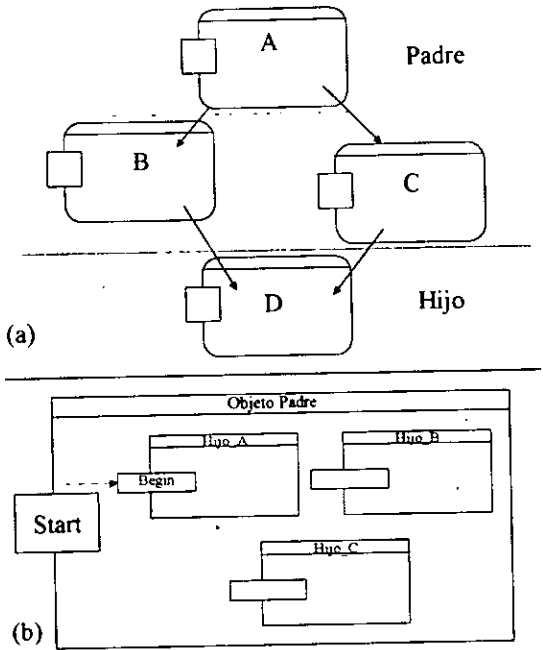

Fig. 2.17.HOOD utiliza relaciones e implementa cadenas (a) La jerarquía padre con el uso de relaciones: (b) El Comienzo (Start) del método padre se dice que será 'implementado por' el método (Begin) comienza de Hijo_A el cual provee la funcionalidad de (Start). Las cadenas in HOOD son mostradas con lineas punteadas . Otras relaciones no son mostradas.

FASE 4. Formalización de la solución.
Se desarrolla un modelo para cada uno de los objetos identificados. Este modelo se le llama *Esqueleto de Descripción de Objetos (ODS)* y contiene las interfaces, las estructuras de control de objetos, las operaciones y comentarios informales. El perfeccionamiento del ODS de los objetos padre se realiza a través de la definición de tipos, constantes y datos, y el de los objetos terminales o finales a través de la definición de su estructura de control interna (OPCS). También en este paso se hace una revisión para localizar inconsistencias, se elabora cierta documentación, se genera y evalúa el código ADA.

Como se dijo antes este proceso se realiza recursivamente hasta que se llega a los objetos finales.

En todos los niveles jerárquicos cada uno de los objetos es documentados en un *Capítulo HOOD* que se divide en secciones de acuerdo a las fases del proceso antes descritas, las que dan una descripción detallada del objeto que tal vez puede ser revisada y perfeccionada más adelante. La documentación también se organiza de manera jerárquica, teniendo la documentación de los objetos raíz con su propia documentación raiz.

La descripción formal del objeto que se realiza durante la fase de la formalización de la solución se escribe utilizando el esqueleto de descripción de objeto (ODS) que es una sección de capitulo HOOD. Este ODS se escribe utilizando un Lenguaje de Diseño de

Programación especial (PDL) que tiene una sintaxis similar a ADA y el ODS también se utiliza para generar código y posiblemente código C++. En el ODS el pseudocódigo o ADA tal vez se utilice para especificar cada una de las operaciones o cada una de las estructuras de control. Durante el perfeccionamiento, en el código ADA que se generó el diseñador desarrolla las operaciones del objeto, y cada objeto genera un paquete en código ADA donde cada objeto hijo se incluye en su objeto padre mediante la utilización de la instrucción WITH en ADA. La transformación de ODS a código ADA puede ser automatizada utilizando herramientas CASE.

2.3.2.3. RESPONSIBILITY-DRIVEN DESIGN (RDD)

Este método también es llamado Designing Object-Oriented Software y aunque el título sugiere que solo cubre la etapa del diseño como se podrá ver más adelante cubre tanto la etapa del diseño como la del análisis.

Los autores de este método consideran a los objetos como "agentes" que colaboran y tienen responsabilidades siendo a su vez clientes y servidores donde la interrelación que existe entre ellos es descrita a través de contratos.

El primer paso en este método es buscar los objetos para describir a continuación sus responsabilidades, pues en RDD el software es considerado como una "cosa viviente" en donde el mundo puede ser modelado como un sistema en el que los objetos colaboran.

RDD soporta los conceptos básicos de la orientación a objetos tales como objetos, clases y herencia. Cada clase tiene sus *responsabilidades* que se refieren a los roles que tienen los objetos y las acciones de los objetos. Estas responsabilidades mas tarde son perfeccionadas y agrupadas dentro de contratos.

Los *contratos* se refieren al conjunto de requerimientos que los objetos de una clase pueden soportar. Estos contratos son perfeccionados dentro de los *protocolos* que muestran la firma especifica de cada operación. Para facilitar el diseño se introduce el concepto de subsistemas.

En este método existe la *relación de herencia* llamada ***relación "is-kind-of"*** que describe la relación de superclase/subclase entre las clases. El método soporta la herencia múltiple y la simple.

Existen algunos tipos de relaciones que no son consideradas como relaciones realmente en el método RDD, pero que se examinan para encontrar responsabilidades y colaboraciones adicionales:

- ***Relación "is-part-of"***, también conocida coma relación *Whole-Part*, que se examina para determinar posibles lugares donde puedan ser colocadas las responsabilidades, tanto en la clase "Part" como en la clase "Whole". También se utiliza para encontrar colaboraciones.
- ***Relación "is-analogous-to"***, es examinada para encontrar superclases faltantes, cuando dos clases son semejantes y posiblemente tienen responsabilidades en común.
- ***Relación "has-knowledge-of"*** para determinar colaboraciones adicionales entre las clases.

El proceso de este método comprende dos fases:

1) *Fase Exploratoria (Análisis en el ciclo de vida del software).* En esta fase se determina a los objetos que tiene un papel en el mundo real, sus responsabilidades y la colaboración de estos objetos.

2) *Fase de Análisis Detallado (Primera parte del diseño en el ciclo de vida del software).* El resultado que se obtuvo de la fase exploratoria se perfecciona y se realiza una especificación completa.

PROCESO DEL METODO.
Para lograr tener una mejor comprensión de este método es importante tener en cuenta tres aspectos importantes acerca de RDD:

1. Los resultados que se obtienen al aplicar este método nunca son resultados finales, ya que la reiteración constituye una parte importante del método.

2. El método proporciona unas guías que no se deben considerar de manera rígida y formal.

3. Durante el proceso se sugiere que la validación se puede llevar a cabo a partir de la ejecución en ciertos partes del diseño a través de atajos (walk-through). El hablar de atajos se refiere a que muchas rutas de ejecución validas e invalidas del mundo real se pueden realizar de una manera más sencilla por "un camino más corto " ya sea por el analista o diseñador para verificar que el diseño sí tiene concordancia con los requerimientos deseados.

FASE EXPLORATORIA.

Actividad 1. Identificación de clases.
Para esta identificación de clases, primero se lee la especificación del problema para comprender los requerimientos. La especificación del problema es una descripción en lenguaje sencillo y natural de cómo debe funcionar el sistema.

De ésta especificación se subraya cada nombre, sujeto o cláusula nominal para hacer una lista de las clases candidatas. A continuación se localizan las clases abstractas y se establece el propósito de cada una de las clases.

Una *clase abstracta* es aquella clase que no crea objetos (instancias), sino que solo va a especificar el comportamiento o un modelo de comportamiento para sus subclases.

Una vez que se han identificado las clases se puede proceder a identificar y nombrar las candidatas a superclases abstractas a través de la agrupación de clases que tengan atributos en común. Para facilitar la tarea de nombrar a las superclases se puede hacer una lista y enumerar los atributos que comparten para que a partir de estos se pueda determinar el nombre. Si al realizar esto aun no se puede determinar el nombre, entonces se debe de descartar el grupo de clases.

A continuación cada una de las clases y su descripción se registra en tarjetas CRC. Estas tarjetas CRC (Class-Responsibility-Collaboration) se utilizan para indicar las clases con sus superclases, subclases, responsabilidades y colaboraciones.

Se debe de tener en cuenta que esta actividad es preliminar y tendrá que realizarse en más ocasiones, por lo que se recomienda al analista que no descarte a la primera alguna clase candidata (es mejor mantenerla en consideración y posiblemente más adelante sí sea necesario descartarla).

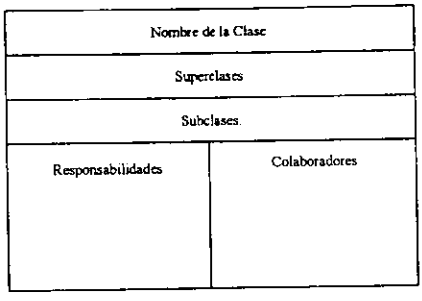

Figura 2.18. Formato de las tarjetas CRC

Actividad 2. Identificación de responsabilidades de las clases.
El término *responsabilidad* se refiere a las acciones que un objeto puede llevar a cabo.

Primero se revisa la especificación del problema para localizar los verbos y frases verbales que indican las acciones(operaciones) de los objetos en el sistema.

Las responsabilidades se identifican a partir de:
- Los verbos y frases verbales que se extraen de la especificación del problema.
- De los propósitos que tienen las clases candidatas.

Las responsabilidades candidatas se asignan a las clases a las que pertenecen aplicando la siguiente guía:
- Se establecen las responsabilidades de manera general.
- Se debe mantener un comportamiento con información relacionada, por ejemplo, si un objeto guarda información, también debe llevar a cabo las operaciones sobre esta información.
- Se debe almacenar información acerca de algo en un lugar. Pero si esto no es posible se puede hacer lo siguiente:

(a) Crear un nuevo objeto que va a servir para almacenar la información
(b) Asignar la responsabilidad al objeto cuya principal tarea es mantener la información.
(c) Agrupar en un solo objeto a los objetos que requieren la información.

Si resulta difícil determinar a que clase se debe de asignar alguna responsabilidad se puede recurrir al walk-through para que las responsabilidades sean valoradas y examinadas.

Al examinar las relaciones antes mencionadas se logra encontrar responsabilidades adicionales:
- Relación "is kind of", permite identificar responsabilidades compartidas por subclases.

- Relación "is analogous to", para identificar a subclases faltantes con sus respectivas responsabilidades.
- Relación "is part of", para identificar las responsabilidades que pertenecen a la clase "whole" o a la clase "part", y encontrar las responsabilidades que la clase "whole" debe tener con respecto a la clase "part".

Una vez hecha esta identificación de responsabilidades se registra en las tarjetas CRC.

Actividad 3. Identificación de colaboración entre las clases.
Cuando un objeto se relaciona con otro objeto para satisfacer sus propias responsabilidades se dice que existe *colaboración.*

Cada una de las clases es capaz de llevar a cabo sus responsabilidades ya sea través de un desarrollo computacional o con la colaboración de otras clases. Para lograr la identificación de estas colaboraciones es importante hacer las siguientes preguntas para cada una de las clases:

- ¿Esta clase es capaz de llevar a cabo por sí misma sus responsabilidades? Si la respuesta es negativa, entonces se hace la pregunta ¿ con cuál clase requiere colaborar para sus llevar a cabo sus responsabilidades?
- ¿Qué otras clases requieren el resultado?
- ¿Qué necesita para poder hace uso de las responsabilidades de esta clase?

También haciendo uso de las siguientes relaciones se puede encontrar algunas colaboraciones adicionales:

- Relación "has knowledge of", que implica la colaboración entre la clase que "tiene el conocimiento" y la otra clase.
- Relación "is parte of", en donde una clase "whole" tal vez requiera colaboración(es) con sus clases "part".

Las clases que no tienen colaboración con otras clases deben descartarse. Una vez realizada la identificación de colaboraciones se hace su registro en las tarjetas CRC.

Se sugiere que para comprobar que la elección de las colaboraciones ha sido la correcta se aplique un "walk-through".

Al terminar esta actividad se concluye la fase exploratoria y el resultado obtenido es un diseño preliminar que se transforma en un diseño confiable en la siguiente fase.

FASE DEL ANALISIS DETALLADO.

Actividad 1. Identificación de jerarquías entre las clases.
Paso 1. Se representan las relaciones de herencia entre las clases con la ayuda de Gráficas de Jerarquía.

Los diagramas de Venn se utilizan para mostrar las responsabilidades compartidas entre las clases y las subclases que heredan de las superclases.

La siguiente guía se aplica para construir una correcta Jerarquía de Clase:

- Se hace un modelo de jerarquía "kind-of" (herencia) donde la subclase va a heredar las responsabilidades de sus superclases.
- Las responsabilidades en común tienen que ser colocadas tan alto como sea posible en la jerarquía de clase con lo que tal vez se requiera crear nuevas superclases.
- No se debe permitir que las clases abstractas hereden de las clases concretas.

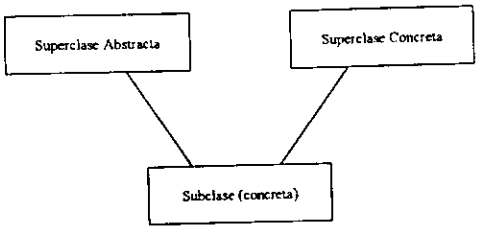

Figura 2.19. Jerarquía de Clases.

Paso 2. Identificación de contratos. Entendiendo por *contrato* a la colaboración que se lleva a cabo a través de una relación de cliente-servidor donde el cliente requiere una responsabilidad de otro objeto(servidor). Un contrato es un conjunto de responsabilidades que un cliente puede requerir y lo lleva a cabo un servidor.

Aplicando la siguiente guía se puede realizar la identificación de contratos, empezando con los contratos definidos para las clases en lo más alto de la jerarquía de clase e ir bajando a través de la jerarquía a las subclases:

- Los contratos se identifican en los niveles de la jerarquía a través de la agrupación de las responsabilidades que son utilizadas por los mismos clientes.
- Se sugiere que una clase debe soportar un conjunto de contratos.
- Se debe tener un mínimo de contratos, esto se logra moviendo responsabilidades similares que pueden ser generalizadas en un nivel alto en la Jerarquía de Clase.

Esta actividad termina con el registro de los contratos en las tarjetas CRC y la asignación de un número a cada uno.

Actividad 2. Identificación de los subsistemas.
Un *subsistema* es un conjunto de clases (e incluso pueden ser subsistemas) que colaboran para satisfacer uno o más contrato y que al ser identificados simplifican los patrones de colaboración.

La identificación de los subsistemas inicia con la construcción de una Gráfica de Colaboración que muestra la manera en que las diferentes clases y subsistemas colaboran para llevar a cabo sus responsabilidades. Este tipo de gráfica también se utiliza para trazar todas las rutas posibles entre las clases dentro del sistema.

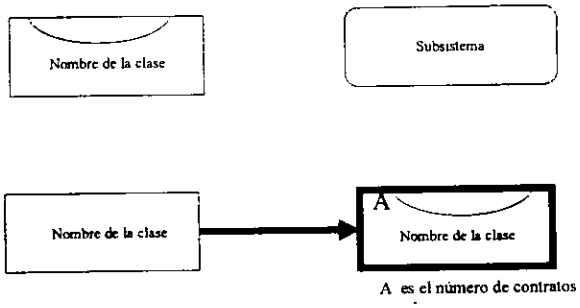

Figura 2.20. Notación para las Gráficas de Colaboración

Para identificar los subsistemas se aplica la siguiente guía:
- Las clases en un subsistema deben colaborar para que soporten un grupo pequeño de responsabilidades.
- Las clases que forman un subsistema deben ser fuertemente interdependientes.

Para llevar a cabo la simplificación de los patrones de colaboración se aplica lo siguiente:
- Minimizar el número de colaboraciones que una clase tiene con otras clases o subsistemas.
- Minimizar el número de clases y subsistemas a los que se les delega responsabilidades.
- Se minimiza el número de diferentes contratos que una clase o subsistema soporta.

Los subsistemas se anotan en Tarjetas de Subsistemas y las tarjetas CRC se modifican ya que la colaboración entre las clases se ha convertido en una colaboración entre clases y subsistemas.

Los cambios que se hicieron en las clases, colaboraciones y contratos tienen que registrarse tanto en las gráficas de Jerarquía como en las tarjetas CRC.

***Actividad* 3.** Identificación de Protocolos.
Los *protocolos* son las firmas para el método que cada clase va a implementar, en donde una firma es el nombre del método, los tipos de los parámetros y el tipo del objeto que el método regresa en respuesta al mensaje.

En esta actividad se realiza un diseño de los protocolos para cada una de las clases aplicando la siguiente guía:
- Se utiliza un nombre para cada una de las operaciones.
- Se asocia una sola operación con cada nombre de método.

- Se especifica de manera explícita en la Jerarquía de Herencia si las clases satisfacen la misma responsabilidad.
- Los protocolos tienen que ser útiles.

Por último se escribe una especificación de diseño para cada una de las clases, subsistemas y contratos para que puedan ser directamente implementados.

Esta especificación de diseño consiste en:
- Una gráfica de jerarquía para cada clase.
- Una gráfica de colaboración para cada subsistema.
- Una especificación de cada clase.
- Una especificación de cada subsistema.
- Una especificación de los contratos soportados por cada una de las clases y subsistemas.

2.3.2.4. METODO OOA/OOD por Coad y Yourdon

Para estos autores aplicando este método se obtiene una reducción del problema y de las responsabilidades del sistema, además de que ofrece ciertos beneficios que están muy por encima de los que se pueden obtener al utilizar los métodos de análisis y diseño tradicionales. Algunos de estos beneficios son: las clases y los objetos se representan de una manera común y explícita, se construyen especificaciones que son resistentes a los cambios, los resultados obtenidos con el análisis y diseño se pueden reutilizar en caso necesario y se obtiene una representación básica para el análisis y el diseño.

Coad y Yourdon basan su método en ciertos principios generales que son muy útiles para manejar la complejidad de los problemas y que solo pueden ser aplicables utilizando la Orientación a Objetos, que a su vez esta basada en una representación básica y uniforme de las clases y objetos que van a aportar ciertos beneficios tales como: no tener una mayor diferencia entre las notaciones utilizadas para el análisis y el diseño, no va a existir una transición del análisis al diseño y va a haber una representación uniforme para el OOA y el OOD, así como para la POO.

Este método consiste de cinco elementos y cuatro componentes.

Los cinco elementos que constituyen el Modelo de AOO describen la funcionalidad del sistema:

1. Elemento Tema (subject).
   Como un mecanismo de partición.
2. Elemento Clase-&-objeto.
   Para capturar las clases y los objetos.
3. Elemento Estructura.
   Para capturar las estructuras o relaciones de Herencia y Whole-Part.
4. Elemento Atributo.
   Para capturar los atributos y las conexiones de instancia entre los Clase-&-Objeto.
5. Elemento Servicio.
   Para capturar los métodos y conexión de mensajes entre los Clase-&-Objeto.

Los cuatro componentes que son diseñados durante la etapa del diseño son:

1. Componente del Espacio del Problema.
2. Componente de la Interacción Humana.
3. Componente del Manejo de Tareas.
4. Componente del Manejo de Datos.

Estos elementos y componentes se implementan con la ayuda de cuatro tipos de diagramas:

1) Diagrama de Objeto o Modelo de Objeto.
2) Diagrama de Estado.
3) Diagrama de Transición de Estado.
4) Diagrama de Interacción de Objeto.

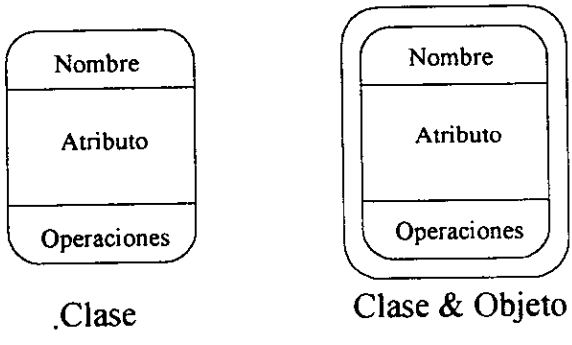

Figura 2.21.Notación de OOA/OOD

Los tipos de relaciones que soporta este método son:

- ***Relación Whole-Part*** entre los objetos.

En esta de relación el objeto que representa el todo (Whole) se descompone en otros objetos (Parts). Esta relación tiene tres variantes:

a) Partes. Un objeto se compone de ciertas partes, por ejemplo una computadora tiene monitor, CPU, teclado, etc.
b) Contenido. Por ejemplo, un closet contiene ropa, zapatos, ropa de cama, etc.
c) Miembros. Por ejemplo, en una empresa dedicada al desarrollo de sistemas tiene un número de analistas, diseñadores, programadores, etc.

- ***Relación de asociación entre los objetos (conexión de instancia).***

En esta relación un objeto necesita de otros objetos para poder llevar a cabo sus responsabilidades. Por ejemplo el objeto Profesor trabaja en el objeto Universidad.

- ***Relación de Generalización-Especialización entre las clases (herencia).***

Esta relación Gen-Spec entre clases define una jerarquía de herencia en la que una clase (superclase) se divide a su vez en otras clases (subclases) que van a tener un comportamiento de especialización de la clase general.

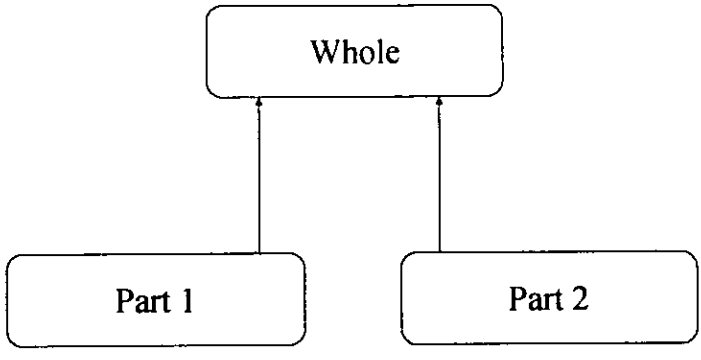

Figura 2.22. Estructura Whole - Part

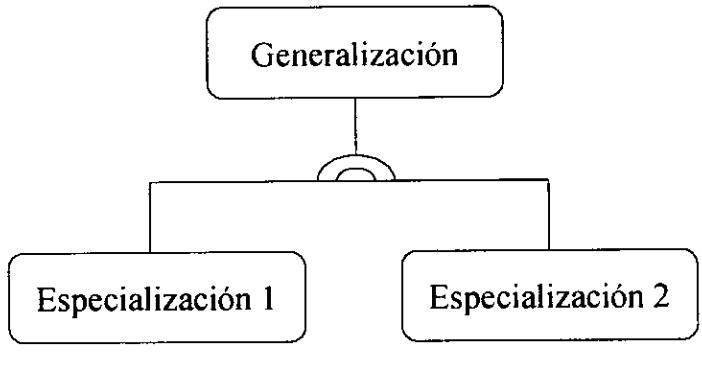

Figura 2.23. Estructura Generalización - Especialización
(Gen - Spec)

ANALISIS.

El proceso del método de AOO se divide en cinco pasos principales:
1. Identificación de los Clase-&-Objetos.
2. Identificación de Estructuras.
3. Identificación de Temas.
4. Identificación de Atributos.
5. Identificación de Servicios.

Estas cinco actividades no deben necesariamente ser realizadas en el orden secuencial que se menciona arriba, sin embargo en proyectos que son complejos por su tamaño se recomienda empezar con la actividad de Identificación de Clase-&-Objeto.

Paso 1. Identificación de Clase-&-Objeto.
Para realizar la identificación de Clase-&-Objetos se recurre a una revisión del espacio del problema para determinar que clases-&-objetos se requieren para describir una solución. Si al identificar los clase-&-objetos resulta que son importantes para la implementación de la solución se deben descartar ya que son parte del espacio de la solución(diseño).

Para facilitar la tarea de encontrar posibles clase-&-objetos se puede considerar las siguientes formas en las que se localizan:
- En estructuras que definen clases de objetos, por ejemplo un triciclo (clase) es una estructura que se compone de Clase-&-objetos tales como ruedas, asiento, pedales, etc.
- Cosas o sucesos (eventos importantes) que forman parte del espacio del problema.
- Papeles que juegan las personas que forman parte del sistema que se está analizando.
- Lugares (locaciones fisicas) que describan el espacio del problema y el funcionamiento del sistema.

Una vez identificados los posibles clase-&-objetos se evalúan para que una vez aprobados se incluyan en el modelo del AOO. Para hacer esta evaluación existen ciertos criterios:
- Solo será útil si la información que contiene es recordada durante el análisis.
- Debe contener operaciones que hagan posible el cambio en los valores de sus atributos.
- Debe de tener más de un atributo.
- Debe de tener conjuntos de atributos y operaciones que puedan aplicarse durante todas las ocurrencias que tenga.

Paso 2. Identificación de Estructuras.
Para identificar las estructuras Gen-Spec cada clase se considera como una generalización y se hacen las siguientes preguntas:
- ¿Existe dentro del espacio del problema?
- ¿Esta incluida dentro de las responsabilidades del problema?
- ¿Va a participar dentro de alguna relación de herencia?

A continuación se le considera como una especialización realizando las preguntas anteriores nuevamente.

Para identificar las estructuras Whole-Part se debe tener en cuenta que en esta estructura cada clase es considerada como un todo (whole) y que esta constituido por varias partes que su vez pueden ser consideradas y definidas como clase-&-objetos.

Por último se determinan las estructuras múltiples que pueden estar formadas por varias combinaciones de estructuras whole-part, gen-spec o ambas y se incluyen en el modelo del AOO.

Paso 3. Identificación de Temas.
Los temas son mecanismos por medio de los cuales se particiona un modelo que es muy grande y complejo ayudando de esta manera a la organización de paquetes de trabajo del modelo.

Esta identificación se hace particinando el modelo de Clase-&-objeto donde un tema es un grupo de clase-&-objetos. Para facilitar este trabajo se puede utilizar a las estructuras antes identificadas, por ejemplo una estructura gen-spec puede ser agrupada dentro de un tema.

Una vez construidos estos temas se pueden incluir en el modelo del AOO.

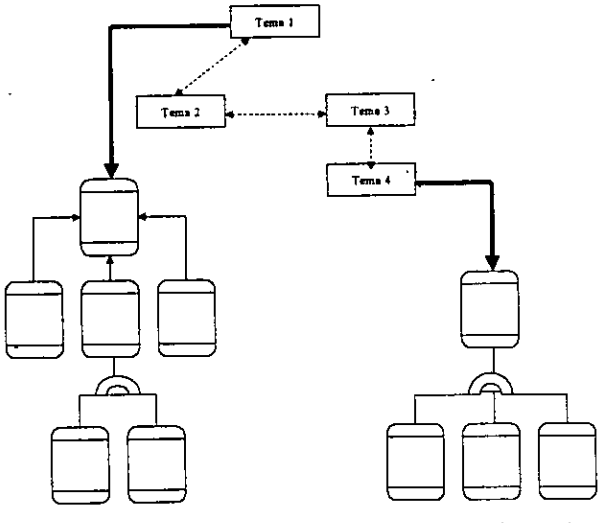

Figura 2.24. Modelo de análisis con referencia temática.

Paso 4. Determinación de Atributos.
Para llevar a cabo esta determinación de atributos se deben realizar las siguientes preguntas para cada clase-&-objeto:

- ¿Cómo se describe éste clase-&-objeto de manera general?
- ¿Cómo se describe en el espacio del problema?
- ¿Cómo se describe dentro del contexto de las responsabilidades del sistema?
- ¿Qué es lo que el clase-&-objeto necesita conocer?
- ¿Qué información tiene que recordar o almacenar todo el tiempo?

En este paso también se definen las conexiones de instancia que indican las relaciones que existen entre los clase-&-objectos de una clase a través de líneas. Los tipos de líneas para la conexión de instancias se muestran en la figura 2.25.

Por ultimo los atributos identificados se incluyen el modelo del AOO.

Paso 5. Identificación de Servicios.
Se definen las operaciones de las clases mediante la identificación de los estados del objeto y la definición de servicios(crear, accesar, conectar, eliminar, operaciones que realizan un cálculo, etc.) que se logra describiendo los estados y transiciones en un diagrama de estado de objetos.

Una vez determinados estos servicios se pueden definir las conexiones de mensajes en donde cada flecha indica un intercambio de mensajes entre objetos en el modelo del AOO.

Los mensajes se mueven a través de los caminos de conexión de instancias, por lo que se puede decir que las conexiones de mensajes se derivan directamente de estas conexiones de instancias.

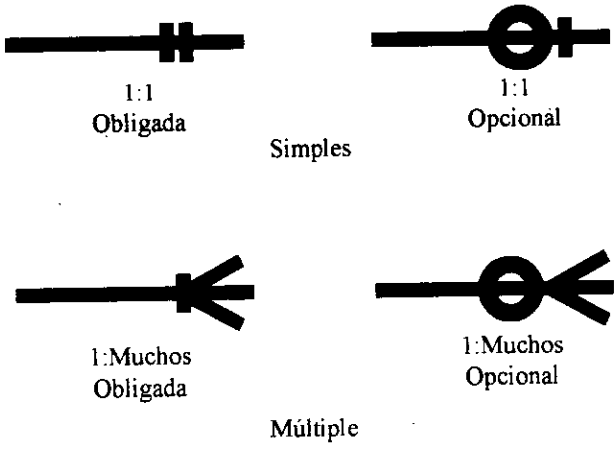

Figura 2.25 Tipos de conexión de instancias.

Para hacer una correcta identificación de las conexiones de mensajes se pueden realizar las siguientes preguntas para cada uno de los clase-&-objetos:
- ¿Qué clase-&-objetos necesitan servicios de éste clase-&-objeto?
- ¿De qué clase-&-objetos necesita servicios éste clase-&-objeto?

Paso 6. Elaboración de la documentación.
Durante este paso se hace la recolección de la documentación.

El diagrama del modelo del AOO se dibuja y se hace la especificación de todos los clase-&-objetos con su nombre, atributos y operaciones. Como actividad adicional se puede incluir diagramas o tablas de los servicios y estados.

DISEÑO.

Paso 1. Componente del Espacio del Problema.

Este componente contiene objetos que directamente corresponden al problema que esta siendo modelado. Se dice que estos objetos son " neutrales" ya que tienen poco o ningún conocimiento acerca de los objetos contenidos en los demás componentes.

Durante el diseño de este componente los resultados obtenidos en el AOO se mejoran y se incrementan. Se realiza una búsqueda de diseños previos y de clases que puedan ser reutilizadas lo que implica realizar pequeños cambios en el AOO. A continuación se hace una especificación de clases que puedan ser agrupadas y se les coloca en una clase raiz que sirve como mecanismo para la agrupación de clases.

Al considerar el lenguaje de programación que se va utilizar para la implementación se hace necesario realizar cambios en las estructuras Gen-Spec del AOO para hacer un acomodo de los niveles de herencia que el lenguaje de programación ofrece.

El soporte del componente del manejo de datos se puede realizar de dos maneras:
1) Cada objeto puede almacenarse a sí mismo.
2) El objeto puede ser guardado por el componente del manejo de datos.

Para ambos casos los atributos y los servicios o nuevas clases tienen que agregarse al componente del espacio del problema. También se puede recurrir al uso de un Sistema de Manejo de Base de Datos Orientado a Objetos (OODBMS) que se hace cargo del almacenamiento de los objetos.

Por último se hace una revisión y evaluación de todo lo agregado a los resultados del AOO.

Paso 2. Componente de la Interacción Humana.
Este componente contiene objetos que proveen una interfaz entre los objetos del espacio del problema y los usuarios como ventanas y reportes.

El diseño de este componente se logra realizando una clasificación de los usuarios del sistema, teniendo en cuenta algunos aspectos como el nivel de habilidades y conocimientos de los futuros usuarios, etc. A continuación cada una de estas categorías de usuarios se describen por medio de criterios como su edad, gustos, necesidades, lo que no les agrada, habilidades, conocimientos, etc.

Para el diseño de las interacciones entre el sistema y los usuarios se debe considerar ciertos puntos: debe haber un mínimo de pasos, darle la importancia que el usuario requiere, lo importante que es la presentación del sistema para el usuario, etc.

Para la evaluación de esta actividad se requiere realizar un Prototyping. Como actividad final se lleva a cabo el diseño de las ventanas, de las gráficas, etc.

Paso 3. Componente del Manejo de Datos.
Contiene los objetos que proveen una interfaz entre los objetos del espacio del problema y una base de datos o un sistema de manejo de archivos.

El diseñador debe elegir entre utilizar un sistema de manejo de base de datos relacionales o un sistema de manejo de base de datos orientado a objetos, también debe hacer una elección de las herramientas de manejo de datos. Por último realiza el diseño del componente del manejo de datos para el sistema de manejo de base de datos y las herramientas que eligió que consiste en el último diseño del dato y sus servicios.

PASO 4. Componente del Manejo de Tareas.
El componente del manejo de tareas contiene objetos que proveen una interfaz entre los objetos del espacio del problema y otros sistemas, por lo que primero se debe hacer una revisión para determinar si se requiere tener tareas en el sistema.

Algunos de los casos en que se requiere tener tareas son:
- Si existe una interacción del usuario con múltiples ventanas que corren de manera simultanea.
- Sistemas de multiusuario.
- Sistemas de un solo procesador que necesitan de las tareas para la comunicación y coordinación.
- Sistemas de multiprocesador.
- Sistemas que necesitan comunicarse con otros sistemas.

Existen muchos tipos de tareas: tareas de manejo de eventos (cuando se oprime el mouse), tareas de manejo del reloj en un intervalo especifico de tiempo, etc. Cada una de las tareas se especifica determinando que tarea es, como se comunica y como se coordina.

CAPITULO 3.

Método OMT (Object Modeling Technique) por Rumbaugh.

3.1. Conceptos en el modelado.

En este método de análisis y diseño orientados a objetos Rumbaugh introduce conceptos referentes al Modelado que se utiliza como una técnica de diseño. Los conceptos y las notaciones que involucra el Modelado Orientado a Objetos se utilizan durante el análisis y el diseño.

Este método cubre las siguientes etapas:

1. ANALISIS.

El propósito del análisis es comprender el problema, es decir, los requerimientos del proyecto, así como el desarrollo de modelos que representan lo que el sistema debe hacer. Estos tres modelos son: modelo del objeto, modelo dinámico y modelo funcional que representan a los objetos y sus relaciones, el flujo de control dinámico y la transformación funcional del sistema respectivamente.

2. DISEÑO DEL SISTEMA.

Durante la etapa del análisis se determina que es lo que la implementación debe hacer, la cual se representa con la ayuda del modelo dinámico, de objeto y funcional. Durante la etapa del diseño del sistema se determina la estrategia a seguir para resolver el problema y construir una solución. Si se trata de algún proyecto muy grande es aquí donde los subsistemas se identifica y particionan de manera ordenada para que el desarrollo sea más manejable y sencillo.

3. DISEÑO DEL OBJETO.

Se construye un modelo de diseño que se basa en el modelo del análisis, pero en el que se incluyen algunos detalles de la implementación que deben de ir de acuerdo con la estrategia que se eligió en la etapa anterior. Durante esta etapa se sigue considerando importantes las clases de la etapa del análisis y se decide que algoritmos y estructuras de datos son los que se necesitan para llevar a cabo la implementación de estas clases, es decir, se determina la definición completa de las clases y las asociaciones que serán usadas durante la implementación, tales como las interfaces y los algoritmos utilizados para la implementación de las operaciones. También se incluyen nuevos objetos, atributos de los objetos y las operaciones. El diseñador debe elegir la manera de implementar los objetos para lograr obtener el mínimo de tiempo de ejecución y de memoria requerida.

La técnica del modelado orientado a objetos involucra y combina tres tipos de modelos que se utilizan para hacer una descripción del sistema:

A: Modelo del Objeto.

Este modelo describe a los objetos (sus atributos y operaciones) a los que algo les ocurre y las relaciones con otros objetos en el sistema con la ayuda de diagramas de objetos, en donde los nodos representan los objetos y los arcos que los unen representan las relaciones que existen entre ellos.

B. Modelo Dinámico.

El modelo dinámico muestra el tiempo y la secuencia de operaciones (eventos que se realizan, la secuencia de estos eventos y la organización de eventos y estados) que dependen del sistema y a los objetos en él. Este modelo dinámico cuenta con diagramas de estado donde los nodos representan a los estados y los arcos representan a las transiciones entre los estados ocurren como consecuencia de un evento.

C. Modelo Funcional.
Este modelo representa la manera en que se transforman los datos(valores) del sistema (procesos de computo). Consta de diagramas de flujo de datos donde los nodos representan los procesos y los arcos el flujo de datos que se da entre los procesos. Este modelo contesta a la pregunta ¿qué es lo que el sistema hace? sin importar el cómo o el cuándo.

Estos tres modelos están relacionados entre sí, pero pueden ser estudiados de manera individual como se hará más adelante.

Las interrelaciones que existen entre estos tres modelos son las siguientes:

- El modelo del objeto describe las estructuras que se utilizan durante el modelo dinámico y funcional.
- Las operaciones del modelo del objeto, en el modelo dinámico son los eventos y en el modelo funcional son las funciones.
- El modelo funcional describe las funciones que se invocan en el modelo del objeto por las operaciones y en el modelo dinámico por las acciones.
- Estas funciones operan con los valores de datos que se especifican en el modelo del objeto.

3.2. Análisis.
En el análisis se realiza la construcción de un modelo formal acerca del problema en el mundo real que consiste de los modelos del objeto, dinámico y funcional,

Actividad 1. Establecimiento de los requerimientos del problema.
Para la construcción de los modelos del análisis como primera actividad hace el establecimiento del problema que tal vez no sea muy preciso pero con la ayuda del análisis esto se irá logrando. En este reporte se registran todos los requerimientos del cliente acerca de lo que quiere que el sistema haga, sin importar la manera en la que se van a llevar a cabo esas necesidades.

Actividad 2. Construcción del modelo del objeto.
Una vez que se establecieron los propósitos y requerimientos del problema se procede a la construcción del modelo del objeto.

Esta actividad empieza con la ***identificación de clases y sus objetos***. Para la identificación de posibles objetos hay que tener en cuento que muchos objetos corresponden a conceptos y cosas tangibles. Para esta identificación de clases y objetos se debe ir al reporte de los requerimientos ya que frecuentemente los sujetos, nombres o cláusulas nominales corresponden a posibles clases. Lo importante de esta actividad es considerar todos los objetos y clases que parezcan importantes y necesarios, aún si no sé esta seguro de que son los correctos, pues con la ayuda de ciertos criterios se pueden detectar los objetos y clases correctos:

- Descartar clases redundantes, esto es, que si dos clases expresan la misma información se debe elegir aquella que tenga el nombre más descriptivo, por ejemplo, en el caso de un hospital se puede considerar a la clase "usuario" y "paciente", sin embargo se debe elegir "paciente" ya que es más descriptiva para este caso.
- Descartar clases irrelevantes que tengan poco o nada que ver con el problema.
- Descartar clases que no son muy precisas, por lo que tal vez se le pueda incluir en otras clases.
- Tener cuidado de no considerar a atributos como clases.
- Cuando un nombre describe a una operación no se debe de considerar como clase.

- A los roles que juegan los objetos en una asociación se les debe descartar como clases.
- Descartar a las construcciones que forman parte del dominio de la implementacion, como subrutinas, estructuras de datos (arboles, listas encadenadas, etc.), algoritmos, etc.

Una vez identificadas las clases de objetos se ***prepara un diccionario de datos***, en el que se incluye cada una de las clase identificadas con su descripción, atributos y operaciones.

A continuación se realiza la ***identificación de asociaciones.*** Si una clase hace referencia a otra o existe una relación de dependencia, existe una asociación. Estas asociaciones pueden ser identificadas como verbos o frases verbales en el reporte del problema:

- Locaciones fisicas: cerca de, parte de, contenida en, etc.
- Acciones de dirección: maneja a, dirige a , etc.
- Acciones de comunicación: habla con, se comunica con, etc.
- Pertenencia: tiene, posee, esta formado por, etc.
- Que satisface alguna condición: trabaja para, esta casado con ,etc.

Para hacer una identificación de aquellas asociaciones correctas y necesarias se aplican los siguientes criterios:

- Descartar asociaciones entre clases que han sido eliminadas.
- Se eliminan asociaciones irrelevantes o que sean asociaciones del dominio de la solución(diseño).
- No se deben modelar acciones, es decir eventos pasajeros.
- Se deben evitar las asociaciones terciar tratando de descomponerlas en asociaciones binarias.
- Descartar asociaciones que son asociaciones derivadas, es decir, que se definen en términos de otras ya que son asociaciones redundantes.

Una vez identificadas las asociaciones se perfeccionan:

- Se le da un nombre a cada asociación para entender mejor a que situación se refiere.
- Se incluyen nombres de roles a las asociaciones si esto es necesario.
- Se especifica la multiplicidad de las asociaciones.
- Se incluyen asociaciones que faltaban por identificar.

El siguiente paso de esta actividad es ***identificar los atributos.***

Los atributos son caracteristicas que describen a un objeto, tales como color tamaño, peso, etc., y es frecuente identificarlos como sujetos o nombres seguidos de adjetivos o frases posesivas. Es importante dar un nombre significativo a cada uno de los atributos. Se recomienda evitar atributos derivados, pero si no es posible se deben etiquetar de manera clara y distinguirlos de su atributo base. También se identifican los atributos encadenados, que son una propiedad del encadenamiento entre dos objetos.

Para una correcta identificación de atributos se utiliza la siguiente guía:

- No se deben identificar objetos como atributos. Si se tiene alguna duda se debe tener en cuenta que en el caso de los atributos lo que importa es el valor que toma, y en el

caso de los objetos lo importante es su existencia como una entidad en el modelo del análisis.

- No se deben modelar objetos identificadores ya que estos son incorporados directamente por los lenguajes orientados a objetos.
- Descartar atributos que describan valores internos de un objeto que no puedan ser captado por otros objetos.
- Se deben omitir atributos con poca importancia y que es poco probable que afecten a muchas operaciones.
- Cuando se tiene un atributo que no tiene ninguna relación con los demás, indica que es necesario incluirlo en una clase nueva.

Las clases se organizan en una jerarquía de clases (herencia) de manera que puedan compartir sus estructuras comunes. Las clases pueden ser generalizadas dentro de superclases si las clases tienen aspectos en común tales como atributos, asociaciones u operaciones; las clases existentes van a ser especializadas dentro de subclases.

Se recomienda evitar en lo posible la herencia múltiple ya que resulta difícil su implementación.

Para determinar si al diagrama del modelo del objeto se le han incluida clases y asociaciones innecesarias, no se han identificado e incluido algunas clases y asociaciones necesarias o se han colocado en el lugar equivocado, ***se hace una evaluación trazando patrones de acceso a través del diagrama del modelo del objeto***.

Agrupar a las clases dentro de hojas y módulos es el último paso que se realiza en la construcción del modelo del objeto. Un módulo es un conjunto de clases (una o más hojas) que capturan algún subconjunto del modelo. Para realizar esta partición del modelo en módulos se debe observar con cuidado todos los puntos de intersecciones entre las clases, pues se corre el riesgo de cortar en algún lugar la red que constituye el modelo del objeto. También se debe procurar tener el mínimo de puentes que conecten cada clase de un módulo con otra clase de otro módulo si no es posible localizar un solo punto de intersección.

Actividad 3. Construcción del Modelo Dinámico.
En esta actividad se hace la ***preparación de los escenarios*** para los diálogos entre los usuarios y el sistema.

Pero ¿qué es un escenario? Un *escenario* es una lista de eventos que describen el comportamiento del sistema.

Primero se consideran los casos normales de esta interacción entre el usuario y el sistema en donde no se presentan entradas inusuales o condiciones de error. A continuación se toman en cuenta los casos especiales como omitir secuencias de entrada, tener valores máximos y mínimos y valores repetidos. Por último se preparan los escenarios para los casos de error de usuarios como tener valores inválidos y fallas de respuesta. En cualquier caso durante esta etapa no es importante el formato de presentación sino el flujo e intercambio de la información y el control

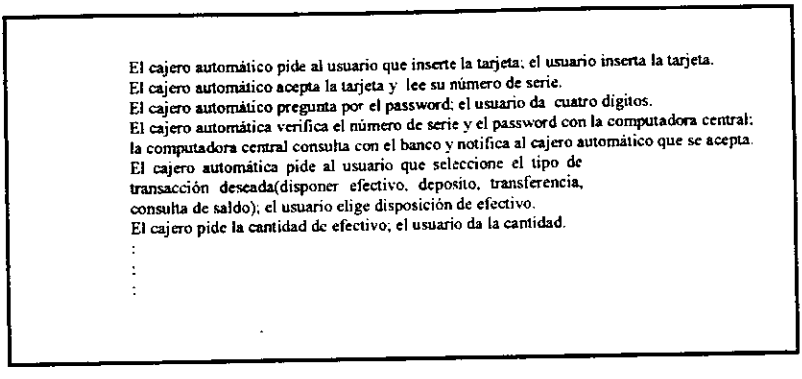

Figura 3.1. Escenario para disponer de efectivo en un cajero automático.

Una vez que se han preparado los escenarios se examinan para hacer la ***identificación de eventos entre objetos*** teniendo en cuenta lo siguiente:

- Un evento es toda acción a través de un objeto que transmite información.
- Cuando existe una interacción objeto a objeto la mayoría de las veces es un evento.
- Eventos pueden ser señales, entradas, decisiones, interrupciones, transiciones y las acciones de un usuario o para un usuario.

Una vez identificados los eventos se agrupan bajo un mismo nombre siempre y cuando tengan el mismo efecto sobre el flujo de control aunque los valores de los parámetros no sean iguales. En el ejemplo del cajero automático "Introduce password" es una clase de evento ya que el valor del password no va a afectar el flujo de control y se puede agrupar con el evento "Tomar tarjeta" que tampoco afecta el flujo de control. Si algún evento afecta el flujo de control se debe agrupar aparte como es el caso del evento "Password incorrecto" que afecta el flujo del control.

Cada uno de los escenarios se representa en un diagrama de trazo de eventos en el que a cada objeto le corresponde una columna y se muestra el flujo de eventos entre estos objetos, donde un evento de entrada es para el objeto que lo recibe y un evento de salida para el objeto que lo manda como se muestra en la figura 3.2.

Los eventos entre todas las clases de objetos se pueden representar con un diagrama de flujo de eventos en el que se incluye todos los eventos de todos los escenarios aun los eventos de errores.

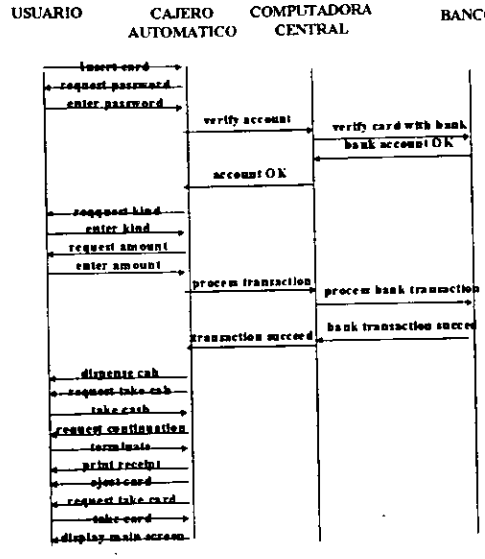

Figura 3.2. Diagrama de trazo de eventos.

El siguiente paso es la ***construcción de los diagramas de estados*** para cada una de las clases con comportamiento dinámico. Estos diagramas muestran los eventos que los objetos de cada clase mandan y recibe; para su construcción se utilizan los diagrama de trazo de eventos.

De los diagramas de trazo de eventos se toman solo los eventos que afectan a la clase de objeto de la que se construye el diagrama de estado, donde el intervalo que hay entre dos eventos representa un estado al que se le da un nombre de preferencia significativo y descriptivo.

Cuando se han cubierto todos los escenarios y eventos que afectan a un objeto de la clase se puede decir que esta construcción ha terminado.

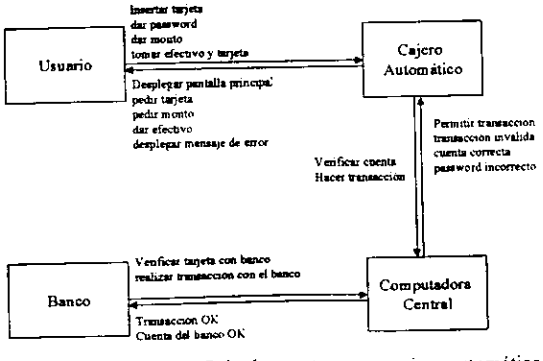

Figura 3.3. Diagrama de flujo de eventos en un cajero automático.

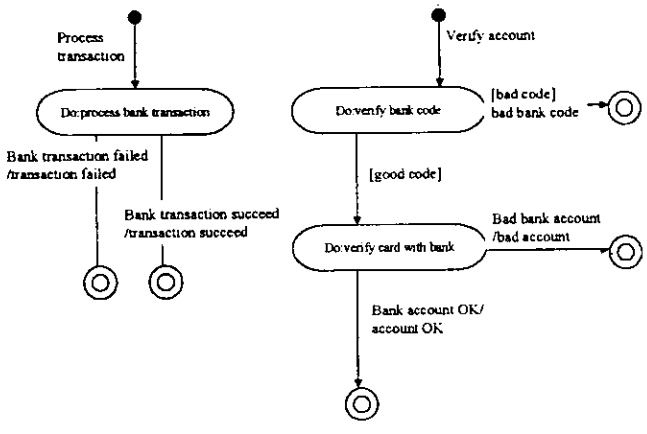

Figura 3.4. Diagrama de estado para la clase Computadora Central.

Por último se ***verifica la consistencia de cada uno de los diagramas*** revisando que cada uno de los eventos tenga quien lo envíe y quien lo reciba, así también los estados deben tener un predecesor y un sucesor a menos que el estado sea un estado de inicio o de terminación.

Actividad 4. Construcción del Modelo Funcional.
Como primer paso de esta actividad ***se identifican los valores de entrada y los valores de salida*** que corresponden a los parámetros de los eventos que ocurren entre el sistema y el mundo exterior. Para el ejemplo del cajero automático:
Valores de entrada: password, tipo de transacción, tipo de la cuenta, monto del retiro de efectivo.
Valores de salida: mensajes y efectivo.

A continuación se ***construyen los diagramas de flujo de datos*** que muestran cómo cada uno de los valores de salida ha sido procesado a partir de un valor de entrada. Estos diagramas de flujo de datos se pueden construir en forma de placas superpuestas donde se muestran diferentes niveles de detalle de estos diagramas de flujo de datos. Estos diagramas contienen los procesos, los actores, los almacenes de datos y el flujo de datos entre los procesos.

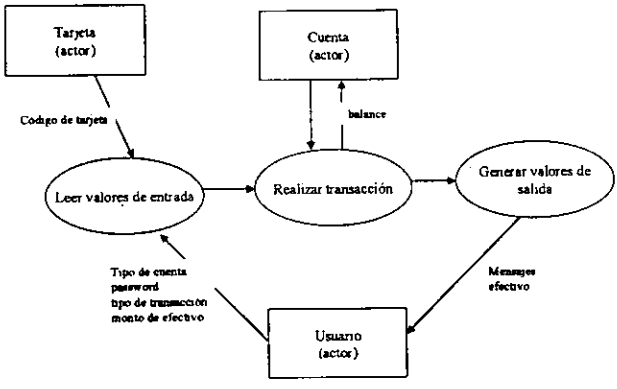

Figura 3.5. Diagrama de flujo de datos.

Una vez terminados y perfeccionados los diagramas, se ***describen las funciones*** de manera declarativa o en forma de procedimientos utilizando el lenguaje común, ecuaciones matemáticas o pseudocódigo.

La *descripción procedural* implica utilizar algún algoritmo cuyo propósito es especificar que es lo que hace la función.

La *descripción declarativa* es mucho más fácil de realizar por lo que es preferible su utilización. Un ejemplo de una descripción declarativa de una función en un cajero automático puede ser:

Actualización de cuenta(cuenta, monto, tipo de transacción)→efectivo, recibo, mensaje
Si la cuenta en un retiro excede el balance actual de la cuenta
    Rechazar transacción y no dar efectivo
Si el monto de un retiro no excede el balance actual de la cuenta
    Dar el débito a la cuento y dar la cantidad requerida
Si la transacción es un depósito
    Dar crédito a la cuenta y no dar efectivo
Si la transacción es una solicitud del estado de la cuenta
    No dar efectivo
En cualquier caso
El recibo debe mostrar el numero del cajero automático, la fecha , la hora, el no. de cuenta el no. de la transacción , el tipo de transacción, el monto de la transacción(si existió) y un nuevo balance.

Hay que recordar que en este paso lo importante es describir que es lo que las funcionen hacen y no la manera en la que se implementan.

El siguiente paso es ***identificar las posibles restricciones para los objetos.*** Algunas de estas restricciones pueden ser precondiciones, es decir, restricciones que los valores de entrada deben cumplir, o postcondiciones que son restricciones que los valores de salida deben sostener.

Actividad 5. Refinamiento y documentación de los tres modelos.
El primer paso es ***incluir las operaciones en el modelo del objeto.*** Algunas de estas operaciones corresponden a:

- Preguntas acerca de atributos o asociaciones.
- Eventos en el modelo dinámico.
- Funciones en el diagrama de flujo de datos.

Las operaciones simples como leer o escribir valores de los atributos no se deben incluir en el modelo del objeto.

Algunas acciones de los diagramas de estados pueden ser funciones y se les puede incluir en el modelo del objeto.

Se ***revisa cada una de las operaciones en el modelo del objeto*** para verificar si existen operaciones similares o muchas variaciones de una sola operación, haciendo una definición amplia de cada operación para que abarcar las posibles variaciones que pueda tener la operación.

También se puede utilizar la herencia durante la identificación de operaciones para reducir el número de operaciones individuales o separadas de las demás, teniendo cuidado de colocar cada una de las operaciones en el nivel correcto de la jerarquía de clases.

Como último paso de la etapa del análisis se hace el ***perfeccionamiento del modelo del análisis*** que se logra revisando varias veces cada una las actividades para verificar que el modelo del análisis cumple con los requerimientos dados. El autor recomienda que se consulte al experto en el dominio del problema para que verifique que el modelo del análisis esta modelando correctamente al problema de acuerdo al contexto del mundo real.

3.3. Modelos del Análisis.

Modelo del Objeto.

Para la construcción de este modelo es necesario definir algunos nuevos conceptos de la orientación o objetos y recordar otros, así como la notación gráfica de los conceptos involucrados.

En el modelo del objeto primero se determinan los objetos y las clases con sus atributos. A continuación se especifican las ***asociaciones*** que es la forma en la que se establecen las relaciones existentes entre las clases y los objetos.

A la conexión entre las instancias de los objetos se le llama ***encadenamiento*** (link). Un encadenamiento es la instancia de una asociación por lo que se puede decir que una asociación describe a un grupo de encadenamientos.

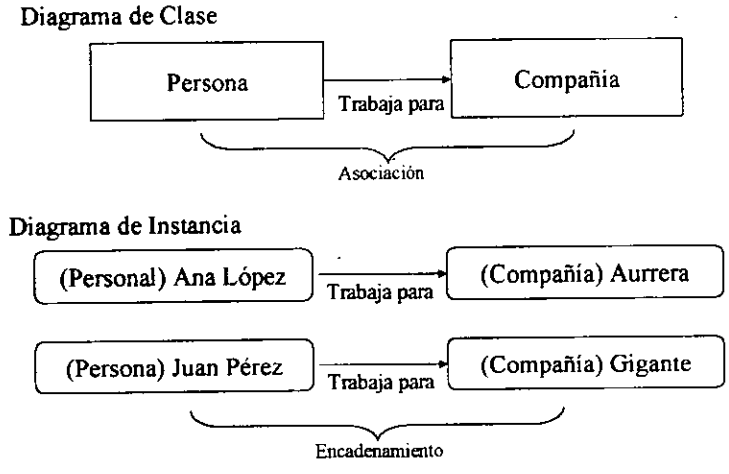

Figura 3.6. Asociación y Encadenamiento

Como se observa en la figura 3.6. estas asociaciones son bidireccionales y en ambos casos la dirección es importante y se refiere a la misma asociación.

Las asociaciones pueden ser binarias o terciarias, aunque se recomienda evitar las terciarias debido a que es muy difícil su representación gráfica y su implementación.

Con la ayuda de la ***multiplicidad*** se puede especificar la forma en la que muchas instancias de una clase se relacionan con una instancia de una clase asociada limitando el número de estos objetos relacionados. La multiplicidad se especifica con un número o un intervalo, por ejemplo:

1 = uno exactamente,
1+ = uno ó más de uno,
3+ = tres o más de tres,
4-6 = de cuatro a seis, etc.

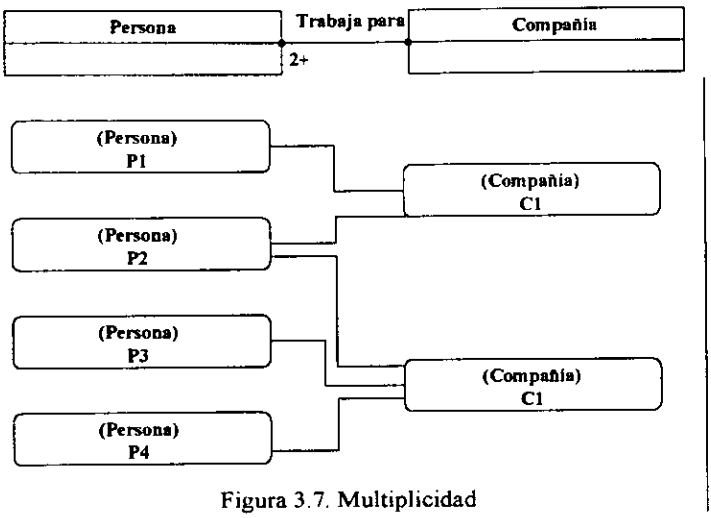

Figura 3.7. Multiplicidad

En la figura 3.6. se muestra una multiplicidad uno-a-uno, y en la figura 3.7. una multiplicidad mucho-a-mucho, donde el punto negro indica `mucho` y significa cero o más y una instancia de la clase Compañía tal vez tenga asociadas dos o más instancias de la clase Persona. La compañía C1 tiene asociadas dos instancias de Persona(P1 y P2) y C2 esta asociada con dos o más instancias de Persona(P2, P3 y P4). Cuando se utiliza el símbolo (o) significa cero o uno e indica opcional.

Figura 3.8.Roles

En las relaciones de asociación existen ***roles*** como se muestra en la figura 3.8.En este caso Persona tiene el rol de "empleado" respecto a Compañía y Compañía tiene el rol de "empleador" respecto a Persona.

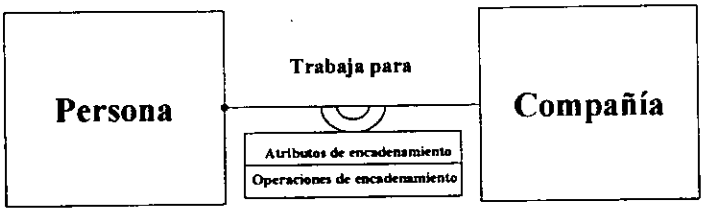

Figura 3.9. Atributos de encadenamiento

Para construir el modelo algunas veces se considera a una asociación como una clase, en donde el encadenamiento va a tener ciertos atributos (salario y puesto, por ejemplo) y tal vez operaciones, lo que evita que estos atributos y operaciones puedan ser utilizados por otros objetos que no forman parte del encadenamiento pues se corre el riesgo de perder información.

Existe un caso especial de asociación llamado ***agregación*** o ***relación Whole-Part*** donde se relaciona a una clase con uno de sus componentes por lo que se puede decir que los objetos van a formar parte de otro objeto. Esta asociación tiene dos características principales:

1. Es transitiva, es decir, si A es parte de B y B es parte de C, entonces A es parte de C.
2. Es asimétrica, si A es parte de B, B no es parte de A.

En la figura 3.10. se ilustra una relación de agregación que consiste de varios niveles donde el rombo indica la agregación.

Otro caso especial de asociación entre clases/objetos es la ***generalización*** o ***herencia,*** ya sea simple o múltiple.

Aunque los términos herencia y generalización son utilizados como sinónimos pues los aspectos que manejan se refieren a la misma idea, debe quedar claro que la herencia se refiere a la forma de compartir los atributos y operaciones utilizando una relación de generalización, la que a su vez se refiere a la relación entre las clases.

Pero ¿qué diferencia existe entre agregación(whole-part) y generalización(herencia)? La agregación se refiere a una relación entre instancias en la que se involucran los objetos y uno de ellos es parte de otro objeto y la generalización es la relación entre clases para lograr la descripción de un objeto.

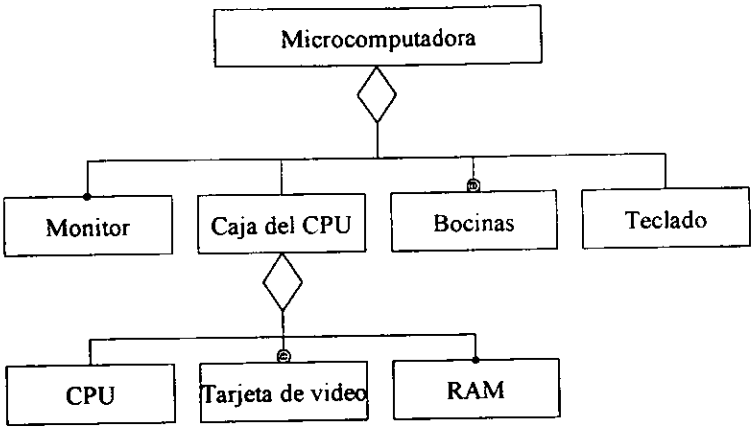

Figura 3.10. Agregación

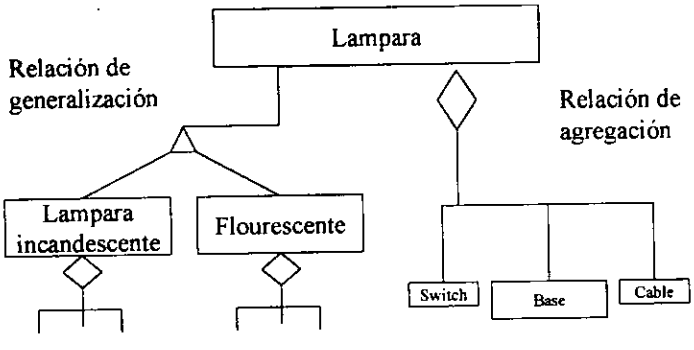

Figura 3.11. Relación de Agregación y de Generalización

Modelo Dinámico.

Este modelo describe el comportamiento dinámico del sistema, esto es, los cambios que sufren los objetos y sus relaciones en el tiempo. Se construye con la ayuda de los diagramas de estado que muestran los estados, eventos y las transiciones de estado, entendiendo por ***estado*** el valor de los atributos y encadenamientos de un objeto.

Un ***evento*** es algún suceso que ocurre a un objeto en un tiempo determinado y cuando este evento actúa sobre otro objeto cambiando el estado del objeto se dice que existe una ***transición de estado***.

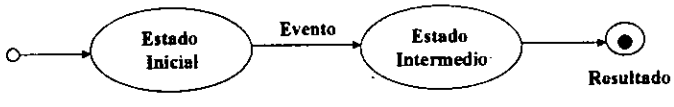

Figura 3.12. Estados inicial y final.

Para la construcción del modelo dinámico primero se deben definir algunos conceptos y notaciones referentes a los diagramas de estado:

1) Acciones de entrada/salida.
Estas acciones ocurren cuando se está entrando o saliendo de algún estado.

Figura 3.13 Acciones de Entrada/Salida

2) Condiciones de Guardia.
Estas condiciones de guardia son funciones booleanas que se prenden al ocurrir un evento solo si la condición es verdadera.

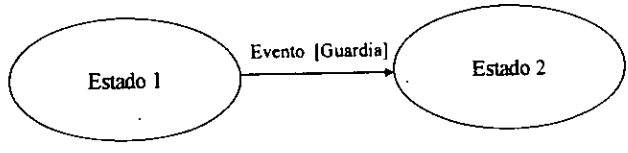

Figura 3.14. Condiciones de Guardia.

3) Acción en una transición.
Una acción es el resultado de un evento y representa una operación instantánea.

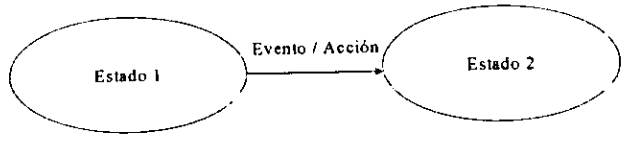

Figura 3.15. Acción en una transición.

4) Acción Interna.
Cuando ocurre un evento no siempre debe causar una acción que lleve a cabo un cambio de estado.

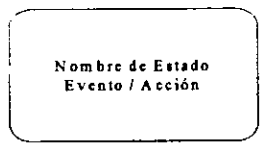

Figura 3.16. Acción Interna.

En los diagramas de estado también se pueden utilizar las relaciones de agregación y generalización con los estados para representar en detalle todo el comportamiento dinámico del sistema.

Al utilizar la agregación se obtienen subestados que solo pueden interactuar entre ellos de forma limitada.

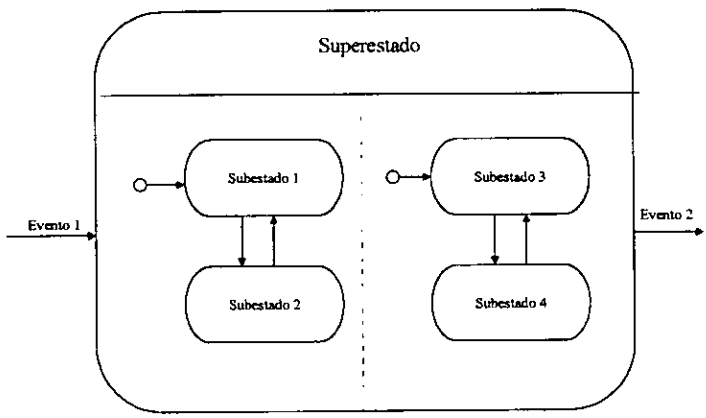

Figura 3.17. Agregación de estado.

En el caso de la generalización los estados pueden ser particionados en diagramas de subestados en donde cada uno de estos hereda las transiciones de estado del superestado. En el caso de los eventos, el superevento hereda sus características a sus subeventos.

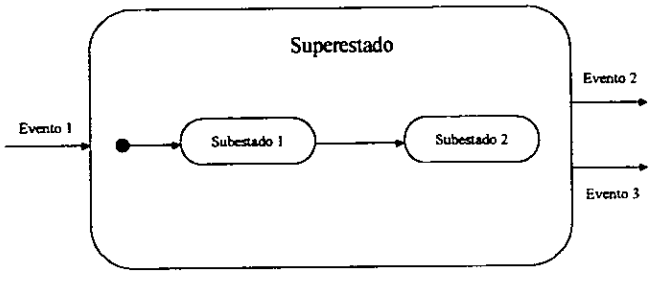

Figura 3.18. Generalización de Estado.

Modelo Funcional.

Este modelo muestra los procesos de computo de los valores de datos sin tener en cuenta cómo, cuándo y porqué estos valores se procesan.

El modelo funcional esta formado por los Diagramas de Flujo de Datos que representan el flujo de los valores de los datos saliendo de los objetos-fuente a través de los procesos que los transforman hasta su destino en otros objetos.

Para comprender mejor estos diagramas de flujo de datos es necesario explicar algunos conceptos importantes para su construcción.

1) Proceso.
Un proceso es aquello que transforma datos y se representa mediante una elipse que contiene el nombre descriptivo de la transformación.

En la figura 3.19. cada una de las flechas que entra o sale de un proceso representa el acarreo de un valor de un tipo dado.

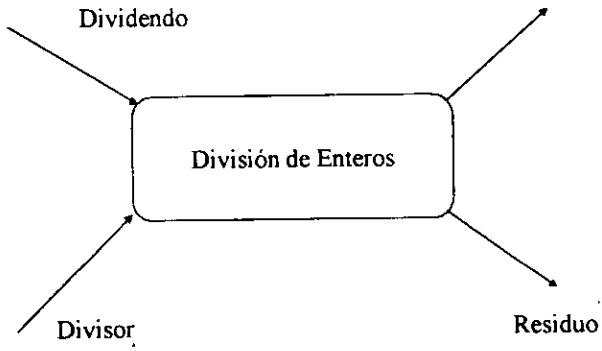

Figura.3.19. Proceso

2) Flujo de Datos.
Mediante el flujo de datos la salida del proceso u objeto se conecta con la entrada del otro proceso u objeto. Este flujo de datos se representa utilizando flechas entre el productor y el consumidor de los valores de los datos etiquetadas con la descripción del dato que puede ser un nombre significativo o el tipo de dato.

3) Actor.
Un actor es todo objeto que consume o produce valores de datos y se representa mediante un rectángulo etiquetado con el nombre del actor.

4) Almacenamiento de Datos.
El almacenamiento de datos es un objeto que tiene un comportamiento pasivo en un diagrama de flujo de datos cuya principal tarea es almacenar datos para accesos posteriores, lo que permite tener acceso a los valores en un orden diferente al que fueron generados. Este almacenamiento se representa con dos líneas horizontales paralelas conteniendo el nombre del almacenamiento de datos y las flechas que llegan a él representan las operaciones o información que lo modifican como: modificar, eliminar elementos o incluir nuevos elementos al almacenamiento.

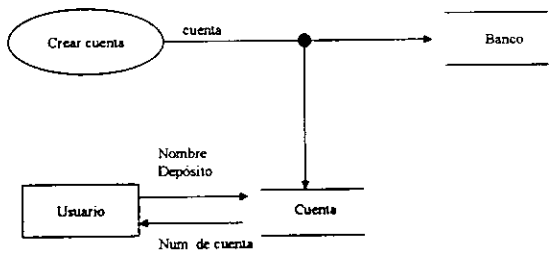

Figura 3.20. Diagrama de flujo de datos para abrir una cuenta nueva en un banco.

3.4. Diseño del Sistema.

Actividad 1. Se organiza el sistema en subsistemas.
Para facilitar el diseño se organiza el sistema en subsistemas, ya que cada uno de estos puede ser diseñado de manera independiente a los demás sin que se vean afectados.

Un subsistema se define como un paquete de clases, eventos y restricciones que están interrelacionados y que tiene una interfaz con los demás subsistemas. A los subsistemas se les puede identificar dependiendo del servicio que proveen, es decir, el conjunto de funciones que tienen un propósito común, por ejemplo realizar operaciones aritméticas.

Así como el sistema se puede dividir en subsistemas estos también puede ser divididos en módulos.

La forma en la que se relaciones los subsistemas es de dos tipos:

1. Relación cliente-proovedor.
El cliente se comunica con el proveedor para solicitar un servicio, sin embargo sólo el cliente conoce la interfaz del proveedor ya que los clientes son los que siempre se van a comunicar para solicitar un servicio.

2. Relación igual-a-igual.
Los subsistemas se pueden llamar unos a otros, sin embargo esta relación es más complicada debido a que los subsistemas necesitan conocer la interfaz de los otros subsistemas.

La descomposición del sistema en subsistemas puede ser en capas (layers) o en particiones:

- Capas.

Cada una de las capas se construye en base a las capas inferiores a ella y es a su vez la base para la construcción de sus capas superiores, lo que implica que los subsistemas sólo pueden tener información acerca de los subsistemas a partir de los cuales fue construido y no de los que son superiores o están encima de él (por ejemplo Windows).

Esta arquitectura en capas a su vez puede ser de dos tipos:
1) *Arquitectura cerrada:* el subsistema solo tiene conocimiento acerca del subsistema inferior inmediato a él, lo que permite que la dependencia entre los subsistemas sea más reducida.
2) *Arquitectura abierta:* el subsistemas puede tener conocimiento acerca de todos los subsistemas debajo de él por lo que no se puede manejar el ocultamiento de la información, aunque se reduce la necesidad de redefinir las operaciones en cada un de los subsistemas. En todo caso el diseñador es el que tendrá que valorar la eficiencia para decidir cual de estos dos tipos es el que le conviene

- Particiones.

Esta organización del sistema en subsistemas es en forma vertical donde los subsistemas se encadenan por parejas. Los subsistemas tienen muy poco conocimiento acerca de los otros subsistemas y cada uno va a proveer un servicio.

Para la organización de un sistema en subsistemas de manera combinada se puede utilizar la descomposición en capas y particiones.

Una vez que se identificados los subsistemas se ***elige la topología del sistema*** utilizando un diagrama de flujo de datos para mostrar el flujo de la información entre los subsistemas, ya que todos los subsistemas pueden interactuar entre ellos o un subsistemas maestro controla las interacciones entre ellos.

Actividad 2. Se identifica la concurrencia.
Cuando dos objetos son concurrentes es porque reciben eventos al mismo tiempo sin que exista una interacción. Para identificar la concurrencia se observan los diagramas de estado. De la misma forma se identifican aquellos subsistemas que son independientes para que sean asignados a diferentes unidades de hardware ya que se obtiene un ahorro al no haber costos de comunicación.

Los objetos que son concurrentes pueden ser controlados juntos a través de lineas de control que al ser implementadas reciben el nombre de tareas.

Una *línea de control* es una ruta trazada a través de un grupo de diagramas de estado en donde un objeto es activado a un tiempo. Es por esto que primero se deben examinar las líneas de control en el sistema para que se pueda hacer la definición de las tareas concurrentes.

Actividad 3. Se asignan los subsistemas y tareas a los procesadores.
El diseñador debe considerar las necesidades de desarrollo que se tienen para elegir la manera en que se van satisfacer estas necesidades. Va a decidir si se utiliza un solo procesador o múltiples procesadores dependiendo del volumen de información que se va a procesar y de la velocidad del procesador.

También se determina que subsistemas van a ser implementados en software y cuáles en hardware. Al implementar en hardware debe de tener en cuenta algunos aspectos

como el costo, la flexibilidad, la compatibilidad, posibles actualizaciones y cambios en el diseño sin olvidar que día con día la industria del hardware está evolucionando.

Se recomienda que los subsistemas que interactuan sean asignados a un mismo procesador para ahorrar costos de comunicación y los que son independientes sean asignados en procesadores diferentes.

A continuación ***se asignan las tareas a los procesadores*** ya que son requeridas en diferentes localidades fisicas para tener control del hardware, para que se permitan operaciones concurrentes e independientes y para incrementar la velocidad de respuesta del sistema.

El siguiente paso es ***elegir la topología de la conectividad*** entre las unidades teniendo en cuenta que las asociaciones y las relaciones cliente-proveedor son con frecuencia conexiones fisicas, de cualquier manera el diseñador debe de ver la manera de que sea optimo el costo de estas conexiones.

También se especifica la topología de las unidades que son iguales y que se repiten, pues en algunos casos se requiere que sea de esta forma por algunas razones de desarrollo. Los canales de conexión y los protocolos de comunicación aquí también se determinan.

Actividad 4. Se elige el manejo del almacenamiento de datos.
El diseñador decide cómo se va a manejar el almacenamiento de datos, ya sea por medio de archivos o por medio de un sistema manejador de base de datos (DBMS).

Actividad 5. Se identifican las fuentes de información y el acceso a ellas.
Estas fuentes de información pueden ser el botón del mouse, la pantalla de una estación de trabajo, un procesador, un satélite de comunicaciones, el nombre de una clase o una base de datos, y es en esta actividad que se identifican y se elige la manera en la que se podrá tener acceso a ellos.

Actividad 6. Se hace la elección de la implementación del control.
En está actividad el diseñador decide cómo se va a implementar el control del software. Para hacer esta elección se considera el control externo y control interno.

- El *control externo* es el flujo de eventos externos y visibles a través de objetos que puede ser manipulado de tres formas:

1) *Sistemas de manejo de procedimientos.* El control reside dentro del código del programa, por ejemplo los lenguajes de programación orientados a objetos.
2) *Sistemas de manejo de eventos.* Un subsistema, un sistema operativo o un lenguaje va a proveer un despachador o monitor que va a tener el control como en el caso de X-Windows.
3) *Sistemas concurrentes.* El control reside en varios objetos independientes donde cada uno de estos tiene una tarea, por ejemplo ADA que soporta tareas concurrentes dentro del lenguaje.

- *Control interno.* El flujo de control es dentro de un proceso que es visto como el llamado de un procedimiento en un lenguaje de programación.

Actividad 7. Manejo de algunas condiciones limitantes.
Como parte del diseño se debe describir la manera en la que se van a manejar ciertas condiciones limitantes como:

- Inicialización, es decir la manera en la que va a inicializar el sistema pues ciertos datos como constantes, parámetros, variables globales, tareas, etc. sin olvidar que solo una parte del sistema va a funcionar durante su inilización.

- Terminación del sistema, esta fase es más sencilla ya que solo se tendrá que abandonar a los objetos internos y en caso de sistemas concurrentes basta con que una tarea notifique a las demás que va a finalizar su función.
- Fallos, es deseable que en un sistema no se presenten fallos pero un buen diseño debe considerar esta posibilidad.

Actividad 8. Se establece el intercambio de prioridades.
En esta actividad el diseñador toma la decisión de cuáles son las prioridades del sistema, ya que no siempre se pueden llevar a cabo todas las metas que se tenían durante el establecimiento del problema o todos los deseos del cliente que tal vez sean incompatibles con estas prioridades. Esta actividad es muy importante pues el éxito o fracaso del producto final tal vez dependa de la buena o mala elección que tuvo el diseñador al elegir las prioridades para el buen funcionamiento del sistema.

3.5. Diseño del Objeto.

Actividad 1. Combinación de los modelos del objeto, dinámico y funcional.
Debido a que tal vez el modelo del objeto no muestra las operaciones, en esta actividad el diseñador las define convirtiendo las acciones y actividades del modelo dinámico y los procesos del modelo funcional en dichas operaciones.

En el modelo dinámico se puede asociar cada evento que un objeto recibe con una operación, es decir, cada evento tal vez sea convertido en una operación de un objeto. Lo mismo sucede con el modelo funcional, donde cada proceso tal vez sea una operación de un objeto. El diseñador va a incluir estas operaciones al modelo del objeto para su implementación.

Actividad 2. Diseño de los algoritmos para las operaciones.
Se ***diseñan o eligen los algoritmos*** para implementar las operaciones. Existen ciertos criterios para la elección del mejor algoritmo:

- La complejidad del algoritmo para procesarlo. Se debe tener en cuenta la forma en que se incrementa el tiempo de ejecución o cuanta memoria se requiere con los datos de entrada.
- La facilidad para implementarlo y comprenderlo
- La flexibilidad del algoritmo.
- Si el algoritmo es el apropiado para el modelo del objeto.

También ***se eligen las estructuras de datos*** apropiadas que se utilizan durante la implementación como listas, arrays, arboles, etc.

Se ***definen clases nuevas*** que no se especificaron en los requerimientos del cliente pero que se necesitan para guardar resultados intermedios y que son detalles de implementación totalmente internos para el sistema.

Se incluyen nuevas operaciones de bajo nivel para que a partir de ellas se puedan definir operaciones que complejas. Estas operaciones de bajo nivel algunas veces pueden ser identificadas a través de las operaciones que se determinaron en la etapa del análisis.

A continuación ***se asignan las responsabilidades a las operaciones y estas operaciones a las clases correctas***, lo que implica tener ciertos conocimientos de implementación ya que se ven envueltas clases internas que son incluidas para la implementación pero que no corresponden a objetos del mundo real haciendo más difícil la tarea de asignar las operaciones.

Cuando en una clase existe más de un objeto resulta un poco difícil decir cual objeto es el que dirige la operación, pero se puede recurrir a una pequeña guía que sirve de ayuda en estos casos:

- Se tiene que ver cual objeto es cambiado por una operación, mientras los otros objetos sólo son requeridos por la información que contienen. El que se modifica es el que importa para la operación y se le denomina *objeto examinado.*
- Se asocia la operación con el objeto examinado (sobre el que se realiza una acción) y se ignora al objeto que lleva a cabo la acción.
- Para localizar una clase que es la meta para la operación se examina el modelo del objeto localizando una clase que está rodeada por un grupo de clases y sus asociaciones.

Actividad 3. Optimización de las rutas de acceso (diseño de optimización).
En esta actividad se adicionan ciertos detalles al modelo del análisis para optimizarlo lo que va a repercutir en una implementación más eficiente.

Primero se ***adicionan asociaciones redundantes.*** Aun cuando en el análisis se evito tener precisamente esta redundancia en las asociaciones ya que no aportaban una información adicional, en la implementación si hay que considerarlas ya que hacen más eficiente el acceso a ciertos objetos y se mejora el tiempo de respuesta. Para llevar a cabo esta optimización se puede tener en cuenta los siguientes puntos:

- Se examina cada una de las operaciones y asociaciones que son atravesadas para obtener su información.
- Se hace un registro de las veces que es llamada una operación y el costo que esto implica.
- Se estima cual es el costo que implica cubrir una ruta.

A continuación se ordena nuevamente la forma de llevar a cabo la ejecución de las operaciones en el algoritmo, pues muchas veces existen rutas que es mejor modificar para obtener una mayor eficiencia.

Se ***almacenan los atributos derivados*** lo que evita su procesamiento en forma repetida para tener una mayor eficiencia del sistema. Para este almacenamiento tal vez se requiera crear objetos y clases nuevas que tendrán que ser actualizadas cada vez que un objeto que pertenece a cualquiera de ellas sufra un cambio.

Un atributo derivado debe ser actualizado cuando cambian los valores base a partir de los cuales fue definido. Esta actualización se puede realizar de tres maneras:

1) *Actualización Explícita.* Como cada atributo derivado se define a partir de un objeto base, el diseñador realiza la actualización determinando que atributos derivados son afectados al cambiar los atributos base codificando la actualización en la operación de actualización del objeto base.
2) *Reprocesamiento Periódico.* En este tipo de actualización los atributos derivados se agrupan y su actualización no se tiene que realizar cada vez que un valor base cambia, sino que se actualiza periódicamente aunque tiene el inconveniente de que se actualiza todo el grupo de atributos derivados y no solo los que lo requieren.
3) *Valores Activos.* La actualización se lleva a cabo a través de una operación que actualiza un valor activo que activa a su vez las actualizaciones para todos los valores dependientes, ya que un valor activo contiene un grupo de valores dependientes y operaciones de actualización.

Actividad 4. Implementación del control.
Durante la etapa del diseño del sistema en la actividad seis, se eligió una estrategia para implementar el modelo dinámico, en esta actividad el diseñador del objeto tiene que

perfeccionar y completar la estrategia elegida para implementar el control. La manera en que se realizara la implementación depende de la estrategia que se eligió anteriormente:

- Para el sistema de manejo de procedimientos se utiliza una localidad dentro del sistema para guardar el estado.
- Para el sistema de manejo de eventos se implementa directamente un mecanismo de máquina de estado que puede ser creada con la ayuda de un lenguaje de programación orientado a objetos si es que no se cuenta con un paquete de máquinas de estado.
- Y por último, para el sistema concurrente se pueden utilizar tareas concurrentes, en donde un objeto se incluye como una tarea en un lenguaje OO o en un sistema operativo, aunque hay que tener en cuenta que la mayoría de los lenguajes OO aún no soportan la concurrencia.

Actividad 5. Mejoramiento e incremento de la jerarquía de herencia.
En esta actividad se perfecciona la estructura de las clases y la definición de las operaciones para incrementar la jerarquía de herencia ordenando nuevamente las clases y operaciones. El objetivo es definir una operación que pueda ser heredada por diferentes clases a partir de aquellas operaciones que son similares pero no idénticas y que cuentan con la misma interfaz y semántica.

La interfaz o firma es el número y tipo de los argumentos y resultados de una operación, por lo que todas las operaciones deben contener el mismo número de argumentos y resultados.

Para igualar el número de argumentos en las operaciones que tienen un número menor de estos se les incluye los argumentos que les faltan aunque vayan a ignorarlos.

Las operaciones que tienen un número menor de argumentos debido a que son un caso especial de una operación generalizada pueden llamar a su operación general que tiene el número de argumentos que se requiere.

A los atributos con diferentes nombres pero que son similares y definidos en diferentes clases se les puede dar el mismo nombre y definirlos en una superclase para que de esta manera la operación tenga un acceso mejor a ellos.

Las operaciones se pueden definir en una superclase y en las subclases que no van a hacer uso de ellas se definen como No-Operaciones.

Se busca un comportamiento común en las clases del modelo del objeto ya que si existe en un grupo de operaciones y/o atributos y se repite en más de dos clases, puede ser que estas sean en realidad especializaciones de la misma cosa, por lo que se puede crear una superclase que contenga estas especializaciones. Por lo regular la nueva superclase es abstracta ya que no tiene instancias directas pero comparte el comportamiento común con todas las instancias de sus subclases.

Actividad 6. Diseño de las asociaciones.
En esta actividad el diseñador elige la estrategia para la implementación de las asociaciones mostradas en el modelo del objeto. Esta estrategia puede ser la misma para todas las asociaciones o una diferente para cada una, pero antes de tomar la decisión debe analizar de que manera se utilizan estás asociaciones y su multiplicidad.

Actividad 7. Determinación de la representación de los objetos.
El diseñador determina cuando va a tener que utilizar tipos de datos primitivos (tipo entero y string) para representar a los objetos en la implementación, o si va a utilizar grupos de objetos relacionados, es decir, se van a adicionar clases para capturar a los atributos.

Actividad 8. Empaquetar clases y asociaciones en módulos (Empaquetamiento Físico).
El empaquetamiento de las clases y sus asociaciones es útil ya que permite que varias personas trabajen en un mismo proyecto si este es muy grande. El empaquetamiento depende del lenguaje de programación elegido, del ocultamiento de la información y de la coherencia entre las entidades.

Ocultamiento de la información.
Como se recordara el ocultamiento de la información permite la implementación de clases que se pueden modificar sin que se modifique el código de sus objetos, y es en esta actividad que el diseñador decide y especifica si una clase va a ser pública o privada y a que atributos se podrá tener acceso desde fuera de su clase, registrando esta decisión en el modelo del objeto con el propósito principal de tener el mínimo de dependencias en el sistema.

El diseñador tiene que limitar el espacio al que tendrán acceso las operaciones, ya que entre menos conocimiento tenga una operación acerca de algo se verá menos afectada si es que se realiza algún cambio en el modelo del objeto y entre menos conocimiento tenga acerca de los detalles acerca de una clase, será más fácil realizar los cambios en la clase. Llevando a cabo las siguientes recomendaciones de diseño se puede reducir el espacio de oportunidad de acción para las operaciones:

- A cada una de las clases se le provee la información relacionada con la asignación de sus responsabilidades de llevar a cabo las operaciones definidas.
- Utilizar una operación para tener acceso a los atributos de un objeto de otra clase.
- Evitar recorrer asociaciones que no están conectadas con la clase requerida en el momento.
- Se busca ocultar a los objetos externos definiendo clases que van a actuar como intermediarios entre el sistema y los objetos externos sin procesar (Clases abstractas de interfaz).

Para lograr un empaquetamiento correcto se busca la coherencia entre las entidades(clases, operaciones y módulos), es decir, que estas entidades sean las apropiadas y que estén organizadas para lograr llevar a cabo un objetivo.

La definición de los módulos se tiene que hacer nuevamente ya que la organización que se realizo en etapas anteriores puede que no sea la correcta pues se han estado incluyendo clases nuevas. Se utiliza el modelo del objeto observando las asociaciones que se muestran sin olvidar definir correctamente las interfaces entre los módulos, es decir, las asociaciones que existen entre las clases de un módulo con las clases de otro módulo, así como las operaciones que tendrán que atravesar los límites entre los módulos, ya que dichas operaciones son las que determinan las relaciones cliente-proveedor que fueron definidas en el modelo funcional.

Actividad 9. Documentación de las decisiones de diseño.
Como última actividad se elabora la documentación detallada de todas las decisiones tomadas por el diseñador en esta etapa para evitar que existan confusiones posteriores; la documentación de los detalles que se adicionaron al modelo del objeto se debe realizar tanto de manera gráfica como textual.

CAPITULO IV.

Método OOD(Object-Oriented Design) por Grady Booch.

Grady Booch es considerado pionero en el desarrollo de métodos para el diseño orientados a objetos. En 1991 público su primer método en el libro Object- Oriented Design with Appilications (1991) pero al ir evolucionando amplió el proceso de desarrollo del método, el cual fue publicado en el libro Object-Oriented Analysis and Design with Applications (1994).

Este método no se debe considerar como un conjunto de pasos a ser desarrollados, sino como un método en el cual el desarrollo es iterativo y se incrementa poco a poco, donde el proceso termina cuando ya no se tienen abstracciones primitivas o cuando las clases y los objetos pueden ser implementados a partir de componentes del software ya existentes.

Para Booch dos aspectos muy importantes para llevar a cabo el diseño de un sistema es la iteración del proceso y la creatividad del diseñador pues considera que la parte más difícil de este proceso es la identificación de las clases y de los objetos.

Booch considera que un solo modelo no es suficiente para capturar todos los detalles que implica un sistema complejo, por lo que utiliza cuatro modelos diferentes para capturar las decisiones y estrategias que se realizan durante las etapas del análisis y el diseño que son la base para llevar a cabo la implementación del proyecto.

En la figura 4.1. se muestra que se utiliza dos dimensiones para especificar la estructura

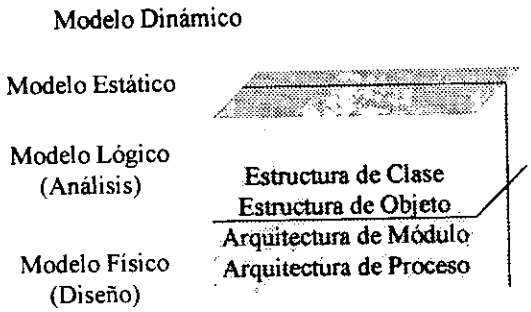

Figura 4.1. Modelos del análisis y diseño

y el comportamiento de un sistema orientado a objetos. Para capturar las decisiones del análisis y el diseño se toma en consideración a las clases y objetos y sus colaboraciones de acuerdo a dos dimensiones, es decir, desde un punto de vista lógico y físico, así como desde un punto de vista dinámico y estático.

Desde el punto de vista estático se construyen los siguientes diagramas:

- Diagrama de objeto. Muestra relaciones o interacciones entre las instancias de clase(objetos) que son definidas en el diagrama de clase.
- Diagrama de clase. Muestra las clases con sus responsabilidades, roles y relaciones con otras clases que determinan el comportamiento del sistema.
- Diagrama de módulos. Indica cómo se particiona la arquitectura del sistema colocando clases y objetos en módulos para el diseño físico del sistema.
- Diagrama de proceso. Muestra la conexión física entre los procesadores y terminales que se requieren para que se ejecute el sistema

Desde el punto de vista dinámico se construyen:

- Diagrama de transición de estados. Para mostrar los eventos que causan el cambio de un estado a otro(transición) y las acciones que resultan de este cambio.
- Diagrama de interacción. Se utiliza para trazar la ruta de ejecución de los escenarios que indican paso a paso lo que sistema debe hacer.

La notación que se utiliza para mostrar el aspecto lógico se muestra en la figura 4.2.

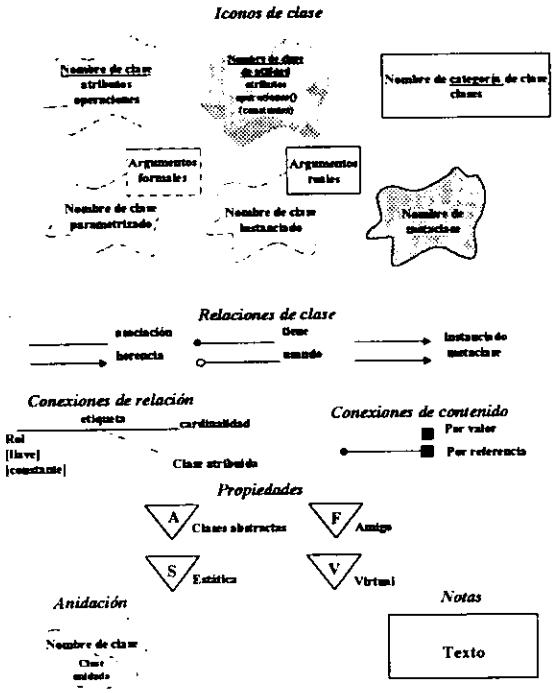

Figura 4.2.Notación para los diagramas de clases.

El proceso del método se divide en un ***microproceso*** y en un ***macroproceso.*** El microproceso consta de cuatro pasos y se utiliza en cada una de las cuatro etapas del macroproceso.

Microproceso:

Paso 1. Identificación de las clases y objetos.
Paso 2. Identificación de la semántica de las clases y objetos.
Paso 3. Definición de las relaciones entre las clases y entre los objetos.
Paso 4. Implementación de las clases y objetos.

Paso 1. Identificación de las clases y objetos.
Se identifican las clases y objetos para que se establezcan los límites del problema. Durante la etapa del *análisis* este paso ayuda a descubrir las abstracciones del espacio del problema en particular, es decir, se decide que es importante y que no lo es para el desarrollo del sistema.

En la etapa del *diseño* se determinan aquellas abstracciones que son importantes y que forman parte del espacio de la solución.

En este paso se construye un diccionario de datos que al principio solo es una lista de las clases y objetos identificados nombrados de manera significativa, pero poco a poco con la iteración del proceso este diccionario de datos es perfeccionado y ampliado, y algunas veces modificado ya que a lo largo del proceso se puede observar que se tienen clases u objetos que en realidad no son importantes para el proyecto.

En este paso se contemplan tres actividades:

Actividad 1. Como actividad inicial se determina las cosas tangibles y el papel que desarrollan utilizando el método del análisis para descubrirlas. Dicho método consiste en extraer de la descripción del problema cada nombre o cláusula nominal considerando sus sinónimos ya que estos determinan posibles clases y objetos. Si se requiere que la clase u objeto sea implementado como una solución, entonces es parte del diseño, pero si es necesario solo para describir una solución es parte del análisis. Utilizando este procedimiento a las abstracciones se les denomina de primer orden y a partir de las cuales se determinan as abstracciones llamadas de segundo orden.

Actividad 2. En el diseño se identifican aquellas abstracciones que están directamente relacionadas al funcionamiento del sistema (implementación).

Actividad 3. En el macroproceso se generan escenarios que al principio del ciclo solo describen el comportamiento del sistema de manera general, pero en esta actividad se explora cada uno de los detalles de estos escenarios más a fondo.

Para este paso se pueden utilizar las tarjetas CRC mencionadas en el capítulo dos para registrar las clases y objetos identificadas. Si se quiere conocer más acerca de este tema se recomienda consultar la referencia bibliográfica número 18.

Paso 2. Identificación de la semántica de las clases y objetos.
El propósito de este paso en la etapa del *análisis* es establecer el comportamiento(operaciones) y los atributos para cada una de las clases y objetos identificadas en el paso anterior. En la etapa del *diseño* se utiliza para llevar a cabo una separación de las partes relacionadas en la solución.

Actividad 1. Se construyen diagramas de objetos y de interacción de objetos con el propósito de capturar las semánticas de los escenarios generados en el macroproceso, capturando de manera formal el storyboarding de cada uno de los escenarios.

¿ Pero qué es y para qué sirve un escenario? Un escenario provee un plan de actividades que muestra algún comportamiento del sistema que es el resultado de la colaboración de varias clases u objetos que documenta los requerimientos y diseños, proveyendo de esta manera un punto central para la comunicación de las abstracciones del sistema que sirve como un plan detallado para la implementación del sistema.

Para comprender mejor la utilización de escenarios y storyboards se puede tomar como ejemplo la realización de una película: como primer paso se determina las escenas, los personajes y la historia de la película para realizar un script. Para la realización de una película es muy importante el desarrollo o flujo de las acciones, por lo que se utilizan los storyboards, ya que un storyboard lleva a cabo la continuidad de la acción que va a ser mejorada escena por escena de manera simultanea con el script de la película, es decir, va a representar una ruta a través del flujo de la película.

Sin embargo la analogia entre el proceso de desarrollar una película y el desarrollo del software aquí termina, ya que en el desarrollo del software no es común tener solo una ruta de comportamiento como sucede con las películas, pues raramente un escenario se encuentra solo, así como las clases y objetos raramente permanecen solos, pues colaboran entre ellos mismos ya que en un sistema que esta bien estructurado cada uno de los escenarios involucra a un pequeño grupo de clases y objetos.

La manera en la que se representan estos escenarios puede ser como tarjetas CRC, diagramas de interacción (similares a los Diagramas de Trazo Eventos del método OMT) y diagramas de objetos.

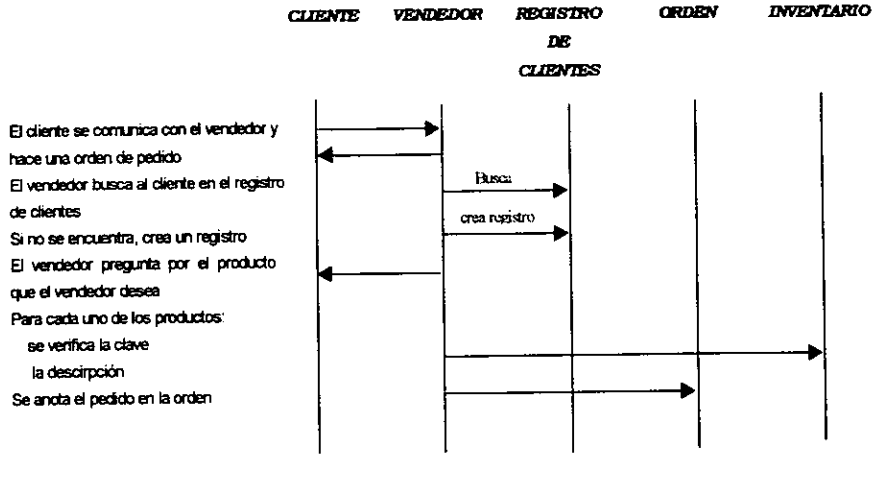

Figura 4.3. Diagrama de Interacción.

Es importante aclarar que los escenarios por si solos no van a definir una arquitectura pues para esto se requieren de dos elementos:

1) *Dimensión Estática,* que consiste de un grupo de clases ordenadas en una jerarquia de herencia.

2) *Dimensión Dinámica*, que consiste en un conjunto de mecanismos que definen las colaboraciones entre ciertas clases y objetos, y es aquí en donde los escenarios ya sean como diagramas de interacción o diagramas de objeto se utilizan para comunicar esta dimensión dinámica.

Actividad 2. Se realiza el diseño de una clase única que aún no determina la jerarquía de herencia de las clases, solo se hace un diseño de cada una de las clases para asignarles sus operaciones y no un diseño arquitectural. Para llevar a cabo esta actividad se hace lo siguiente:

- Para cada una de las clases se determina sus responsabilidades y sus roles.
- Se determinan las operaciones que van a cubrir esas responsabilidades y roles, tratando de utilizar repetidamente aquellas operaciones que pueden satisfacer roles y responsabilidades similares.
- Se definen las operaciones para la construcción, eliminación y copiado de objetos ya que es mejor determinar una política para el comportamiento de las clases, aunque esta actividad no se realiza inmediatamente sino más adelante en el ciclo del desarrollo conforme se va iterando.
- Se adicionan aquellas operaciones que no son requeridas al momento por los clientes, pero que pueden ser requeridas por futuros clientes.

No se debe olvidar que este es un proceso iterativo por lo que más adelante se va a aplicar este paso de identificación de la semántica de igual manera pero con una jerarquía de clases, por lo que se ve la necesidad de crear clases abstractas o de colocar algunas operaciones en una superclase.

Actividad 3. Se identifican los patrones de comportamiento común con el fin de buscar una reutilización pues responsabilidades y roles comunes pueden ser agrupados en clases abstractas.

Paso 3. Definición de las relaciones entre las clases y entre los objetos.
Para la etapa del *análisis* se identifican las asociaciones que existen entre las clases y los objetos considerando algunas relaciones de herencia y de agregación. Se construyen diagramas de clase donde se establecen las asociaciones entre las abstracciones y se incluyen las operaciones y los atributos. También se construyen diagramas de objetos considerando nuevas interacciones de las clases y los objetos, localizando patrones de interacción que tal vez se encontraban ocultos.

Durante la etapa del *diseño* se buscan las colaboraciones que forman los mecanismos de la arquitectura y se perfeccionan los diagramas de clases mostrando las decisiones que se toman acerca de las relaciones de herencia, agregación, instanciación y uso. En esta etapa se utilizan los diagramas de objetos junto con detalladas maquinas de estado finito para mostrar el comportamiento dinámico. También se van a construir un conjunto de diagramas en donde se van a describir las colaboraciones.

En este punto es importante describir las relaciones que soportan las clases y los objetos en este método:

<u>*Relaciones entre objetos:*</u>

- Colaboración. Un objeto colabora con otros objetos a través de los encadenamientos. donde un encadenamiento muestra una asociación especifica a través de la cual un objeto (cliente) utiliza los servicios de otro objeto (proveedor).Un objeto puede comportarse de tres maneras:

1) Actor. Este objeto puede operar sobre otros objetos pero no se puede operar sobre él (usualmente objetos activos).

2) Servidor. Es un objeto que solo puede ser operado por otros objetos.
3) Agente. Es aquel objeto que puede ser operado por otros objetos y también operar sobre otros objetos.

Cuando un objeto le pasa un mensaje a otro objeto a través de un encadenamiento se dice que estos objetos están sincronizados.

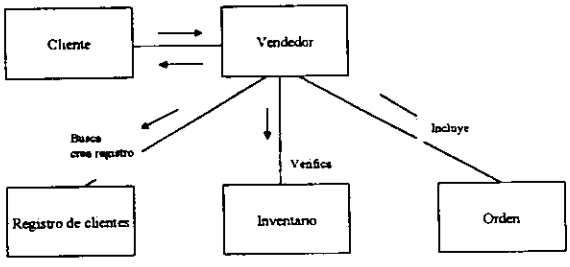

Figura 4.4. Diagrama de Colaboración.

Agregación o relación Whole-Part. Se relaciona un objeto (whole) con sus partes o componentes.

Relaciones entre las clases:

- Asociación. Esta relación indica la dependencia de la semántica (comportamiento y atributos) entre dos clases.

Utilizando la cardinalidad se puede indicar cuando existe más de una asociación entre un par de clases:

1	exactamente uno.
N	donde N es cualquier número.
0..1	cero o más.
1..N	uno o más.
0..1	cero o uno.
3..7	rango específico de 3 a 7.

- Relación de Herencia (Generalización-Especialización). Muestra la manera en que una subclase hereda las operaciones y atributos de su superclase.

- Agregación (Whole-Part). Similar a la relación de agregación entre los objetos.

- Relación de Uso. Esta relación indica los encadenamientos entre las instancias de las clases en donde existe una relación cliente/proveedor. La clase cliente debe invocar operaciones de la clase proveedor para llevar a cabo ciertos servicios.

En el diagrama de clases de la figura 4.5. se muestran las relaciones entre un grupo de clases. Existe una relación de asociación entre la Clase 1 y la Clase 2. La Clase 2(Whole) es a su vez una agregación en donde la Clase 3(Part) tiene una instancia y la Clase 4(Part) un número cualquiera de instancias. La Clase 5 es una clase abstracta que tiene como subclases a la Clase 3 y Clase 4 así como una relación de uso con la Clase 6.

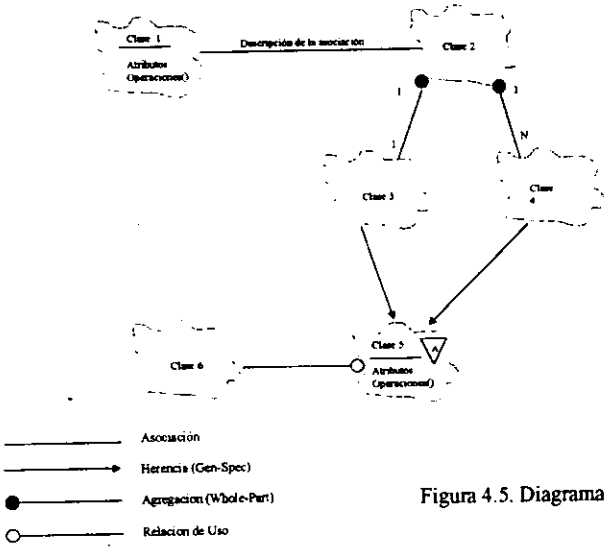

Figura 4.5. Diagrama de Clase.

- Relación de Instanciación. Con esta relación se crean clases (clases instanciadas) con instancias que son objetos de otra clase (clase parametrizada) con operaciones y atributos definidos independientemente de los parámetros formales de la clase parametrizada, en donde la clase instanciada requiere utilizar una relación de uso con otra clase concreta que va a proveer sus parámetros actuales. Los lenguajes de programación Eiffel, ADA y C++ definen clases parametrizadas.

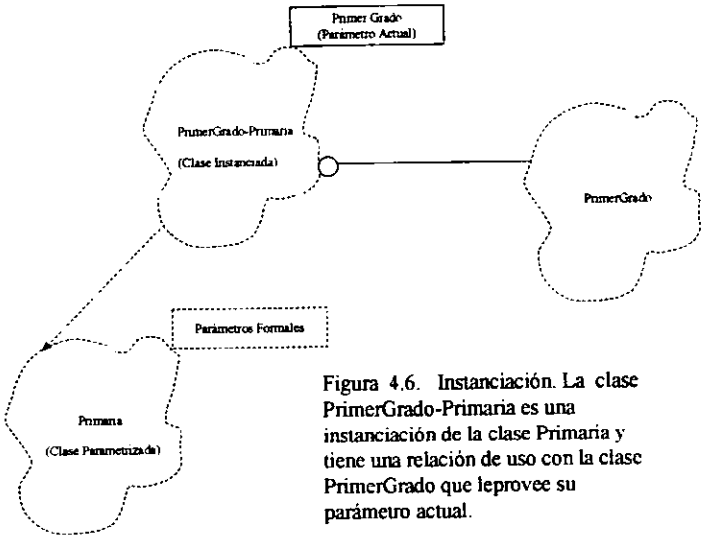

Figura 4.6. Instanciación. La clase PrimerGrado-Primaria es una instanciación de la clase Primaria y tiene una relación de uso con la clase PrimerGrado que leprovee su parámetro actual.

- Metaclase. Una metaclase es una clase de una clase que es tratada como un objeto estático que contiene operaciones que se utilizan para crear nuevas instancias de su clase. En los lenguajes de programación C++ y Smalltalk es posible definir metaclases.

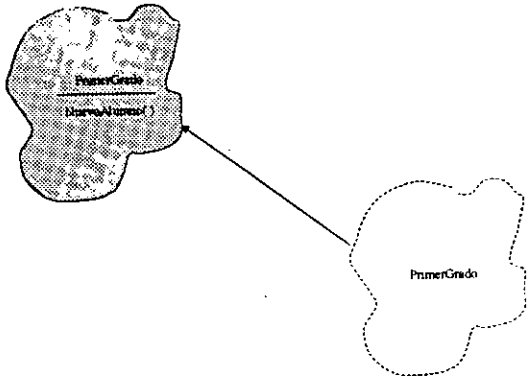

Figura 4.7. Metaclase. La metaclase PrimerGrado tiene la operación NuevoAlumno() que se utiliza para crear nuevas instancias de la clase PrimerGrado.

En este paso se desarrollan tres actividades principales:

Actividad 1. Especificación de las asociaciones.
Esta actividad es principalmente un análisis y diseño previo para capturar los detalles importantes acerca de las relaciones entre las abstracciones. Se construye un diagrama de las clases que muestra las operaciones y atributos necesarios para el modelo determinando las asociaciones entre estas clases. Si existen dependencias indirectas se crean abstracciones nuevas que sirven como agentes o intermediarios. Para asegurarse de que la identificación de las asociaciones es la correcta se hace uso de los escenarios para validar estas decisiones.

Actividad 2. Identificación de colaboraciones.
Es una actividad de diseño donde la identificación de colaboraciones depende de la etapa del macroproceso en la que se trabaja.

Se definen los mecanismos de arquitectura identificados y donde se vea la posibilidad de concurrencia se especifican los actores, agentes y servidores, así como el tipo de sincronización entre ellos. Se colocan las clases en una jerarquía de herencia para representar la generalización/especialización.

En el diseño de la arquitectura se agrupa a las clases dentro de categorías y se determinan los *subsistemas* que están formados por módulos y posiblemente otros subsistemas con el propósito de particionar el modelo físico del sistema.

Una vez definidos los subsistemas se pueden construir los diagramas de módulos. En la figura 4.8. se muestra la notación para los diagramas de módulos y la forma en que se indica que el módulo 1 depende de la especificación del módulo 2.

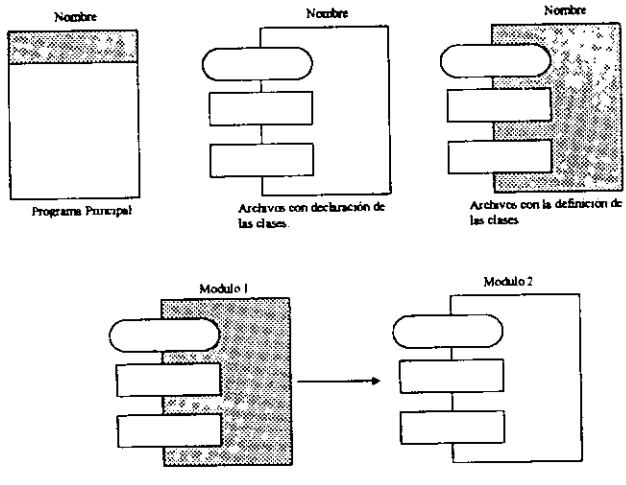

Figura 4.8. Notación para los diagramas de módulos. El módulo 1 depende del módulo 2.

Actividad 3. Perfeccionamiento de las Asociaciones.
Esta es una actividad tanto de análisis como de diseño. En el *análisis* se busca involucrar ciertas asociaciones dentro de otras para se vea representado de manera más clara y precisa el problema. Durante el *diseño* se hace un detallado plan de acción para la implementación mediante una transformación de estas asociaciones e incluir nuevas relaciones.

Paso 4. Implementación de las clases y de los objetos.
Durante el *análisis* se busca identificar nuevas clases y objetos que van a servir para el siguiente nivel de abstracción y que van a ser consideradas en el siguiente paso de este proceso iterativo, así como perfeccionar aquellas abstracciones ya definidas con anterioridad.

En el *diseño* se crean representaciones tangibles de las abstracciones existentes para que se pueda obtener un modelo de las abstracciones que pueda ser ejecutable.

Este paso solo considera una actividad: la *selección de las estructuras y de los algoritmos* que proveerán las semánticas de las abstracciones del modelo. Se determinan las operaciones más importantes y las que se requiere que sean optimizadas y los algoritmos necesarios para implementar cada una de las operaciones.

El ***macroproceso*** es una estructura que esta controlada por el microproceso y consta de cuatro etapas:
Etapa 1. Conceptualización.
Etapa 2. Análisis.
Etapa 3. Diseño.
Etapa 4. Implementación.

4.1. Conceptualización.
En esta etapa se establecen los requerimientos esenciales para el sistema sin olvidar nuevas ideas que se tengan acerca de una aplicación, ya que en la conceptualización se busca establecer una visión general acerca de cada idea y validar cada una de las suposiciones que se tenga acerca de estas ideas para llevar a cabo una definición completa de cada una de ellas.

Para llevar a cabo la validación de las suposiciones acerca de estas ideas se hace uso de Prototipos evaluando los resultados obtenidos y tomar decisiones para el desarrollo del producto, además de tomar en consideración los posibles riesgos o decidir si es necesario hacer una exploración más detallada.

Considerando lo anterior, se debe tener en cuenta que la conceptualización es más que nada una actividad que involucra la creatividad de los analistas por lo que no se deben seguir reglas rígidas para el desarrollo de esta etapa.

4.2. Análisis.
En el análisis se construye un modelo que muestra el comportamiento del sistema mediante la representación de las clases y de los objetos así como sus responsabilidades, roles y colaboraciones.

Puesta que en esta etapa se centra la atención en el comportamiento del sistema, se identifican las funciones que va a llevar a cabo el sistema capturándolas a través de los escenarios. Cada uno de estos escenarios captura una función especifica utilizando los

principales para mostrar comportamientos clave y los secundarios para aquellos que están bajo alguna condición excepcional.

Esta etapa consta de dos actividades principales:

1) ***Análisis del dominio***. En esta actividad se identifican las clases y objetos que forman parte del espacio del problema y se hace una revisión de los sistemas que ya estén en uso para ver si es posible su reutilización o de alguna de sus partes para la construcción del proyecto actual.
2) ***Planeación de los escenarios***. Las funciones se agrupan de acuerdo a su funcionalidad en el comportamiento del sistema. Para llevar a cabo el storyboard de los escenarios se hace uso de las tarjetas CRC. Los diagramas del objeto se utilizan para mostrar la semántica de los escenarios de una manera más precisa ya que estos muestran a los objetos que colaborar para llevar a cabo una función del sistema, así como la manera en que interactúan a través del paso de mensajes, es decir, el proceso de colaboración entre objetos. También se construyen los diagramas de clases para mostrar las asociaciones existentes entre las clases del sistema. Si se requiere mostrar el ciclo de vida de algún(os) objeto(s) se hace uso de los diagramas de transición de estado o máquinas de estado finitas. Si se quiere profundizar en el funcionamiento de estas máquinas de estado finitas se puede recurrir a la bibliografía referente a Grady Booch.

4.3. Diseño.

En esta etapa se crea la arquitectura necesaria para llevar a cabo la implementación haciendo uso de diagrama de objetos y de clases para mostrar sus estructuras.

Para la *arquitectura lógica* lo importante es mostrar el agrupamiento de clases en categorías de clases y para la *arquitectura física* el agrupamiento de módulos en subsistemas.

También se describen las políticas necesarias para algunos casos como: la detección de errores, errores de mano, el manejo de memoria, el manejo del almacenamiento de datos, y todo lo relacionado al control, capturando estas políticas con la ayuda de los escenarios.

En el diseño se hacen las siguientes preguntas relacionadas con la arquitectura del sistema:

- ¿Qué clases existen y de que manera se relacionan?
- ¿Qué mecanismos se van a utilizar para regular la manera en la que los objetos colaboran?
- ¿Dónde es recomendable que cada objeto y clase sean declarados?
- ¿A que procesador es recomendable que se le asigne un proceso, y para un procesador dado de que manera se le van a asignar determinados procesos?

Esta actividad esta compuesta por tres actividades:

Actividad 1. En esta actividad el sistema entero es dividido en capas y particiones, llevando a cabo una descomposición lógica mediante la representación del agrupamiento de las clases, y una descomposición física utilizando los módulos.

Las funciones que fueron agrupadas en el análisis se colocan en capas o en particiones. Si las funciones se construyen teniendo como base otra función se colocan en diferentes capas, pero si colaboran para proveer un comportamiento común se asignan a particiones que representen servicios iguales.

Actividad 2. Como se menciono anteriormente en esta etapa es de vital importancia definir las políticas comunes necesarias para el buen funcionamiento del sistema, por lo

que en esta actividad se desarrolla un escenario para cada una de estas políticas para describir su comportamiento que será capturado utilizando un prototipo ejecutable.

Actividad 3. Se elabora un plan de acción detallado y formal en el que se identifican los vacíos que se tengan acerca de la arquitectura, las tareas del equipo del trabajo y los riesgos evaluados en el análisis con el propósito de organizar la evolución de la arquitectura.

CAPITULO 5.

Método FUSION.

5.1. Conceptos.

FUSION es considerado por sus desarrolladores como un método de segunda generación debido a que dentro de su proceso se incluyen y amplían algunos aspectos que forman parte de otros métodos orientados a objetos y de los métodos formales.

Durante la etapa del análisis de FUSION se puede observar que existe una gran influencia del método OMT descrito en el capítulo cuatro.

El Modelo Funcional de OMT es análogo al Modelo de Operación de FUSION, aunque para los autores de FUSION los diagramas de flujo de datos de OMT no son lo suficientemente apropiados y son poco prácticos, por lo que en este método se hace uso de Precondiciones y Postcondiciones que forman parte de los métodos formales y que describen formalmente lo que el sistema hace.

Para la construcción del Modelo del Ciclo de Vida se tiene la influencia del Diseño de Sistemas de Jackson utilizando expresiones regulares para la descripción de los ciclos de vida.

En la etapa del diseño se ve la influencia del método de Booch y de CRC (Class Responsability Collaborator) que no es en sí un método sino una técnica que da la idea principal para la construcción de gráficas de Interacción de Objetos en lugar de utilizar las ya conocidas tarjetas CRC, ya que con estas tarjetas no se logra tener la información relacionada a la comunicación de una manera visible debido a que se tiene la información de cierta manera un poco dispersa en las tarjetas.

Para la construcción de las gráficas de Visibilidad se toma la idea de los diagramas de Objeto del método de Booch.

El proceso del método FUSION y las dependencias que existen entre los diferentes modelos que se construyen se muestra en la figura 5.1.

5.2. Análisis.

Como se puede observar en la figura 5.1. establecer y documentar los requerimientos del sistema no forman parte del proceso del método ya que se da por hecho la existencia de un reporte por parte del cliente de los requerimientos necesarios para el desarrollo del sistema.

En la fase del análisis se especifica cada una de las tareas que el sistema tiene que llevar a cabo.

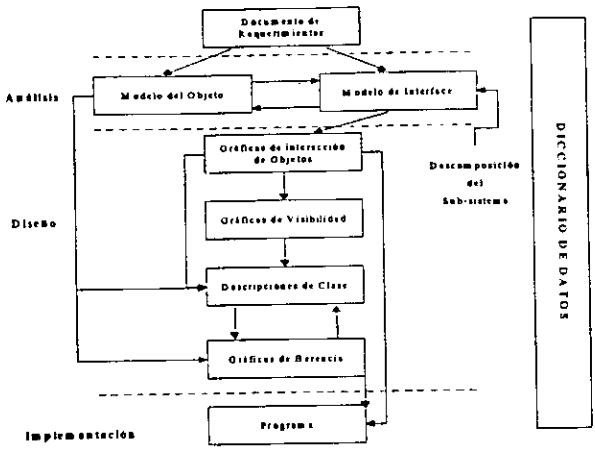

Figura 5.1. Proceso del Método Fusión

En el análisis se realizan cuatro actividades de manera iterativa para obtener el resultado deseado:

Actividad 1. Desarrollo del Modelo del Objeto.
El modelo del objeto se utiliza para representar la estructura estática del sistema mostrando las clases y asociaciones que existen entre los objetos que pertenecen a dichas clases del espacio del problema.

Como primer paso ***se hace una lista de las posibles clases y relaciones*** que se tenga la idea pertenecen al dominio del problema. Se recomienda que esta lista sea revisada repetidas veces para que la lista final solo contenga clases que son esenciales para lograr una explicación clara del problema y eliminar las clases que son innecesarias buscando siempre guardar aquellas clases que sean más generales.

Una vez que se tiene una lista lo más completa posible se puede llevar a cabo la ***elaboración del Diccionario de Datos*** que contiene a las clases y relaciones identificadas junto con una descripción de cada una de ellas.

Se hace ***la construcción del modelo del objeto*** en el que se describen las relaciones sin olvidar que en este modelo se muestran las clases y no los objetos, por lo que estas relaciones dan la idea de las asociaciones que existen entre los objetos que pertenecen a las clases en el modelo. La manera en la que se representan las relaciones se muestra en la figura 5.2.

Para tener ciertas restricciones en el número de asociaciones que un objeto puede tener con cualquier otro se utilizan *restricciones de cardinalidad:*

* = denota cero o más,
\+ =denota uno o más,
1..4 = es un rango de uno a cuatro, por ejemplo, o
3 = un número, por ejemplo tres.

Figura 5.2. Relación de enseñanza

- = se utiliza cuando en una relación deben participar todos los objetos de una clase involucrada, ya que no siempre se requiere que cada uno de los objetos de una clase participen en una relación.

Al igual que en OMT una relación puede ser tratada como un objeto y tener sus propios atributos y cada una de clases puede ser etiquetadas con el nombre del rol que juega en la asociación.

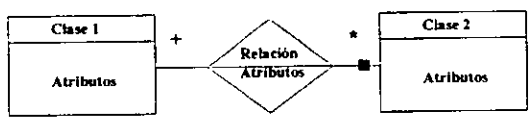

Figura 5.3. Relación con Atributos

Para la estructuración del modelo del objeto se cuenta con un mecanismo llamado **agregación** que permite tener una clase agregada a partir de atrás clases que son componentes de la clase agregada. La notación para las clases agregadas se muestra en la figura 5.4. en donde se indica la cardinalidad de cada uno de los componentes que participan en la agregación, de lo contrario se toma por default una cardinalidad de uno.

En la figura 5.5. se puede observar que en algunos casos existe una relación entre los componentes de una agregación. La clase Examen participa en una relación de asociación con la clase Salón, por lo que Examen cuenta con un atributo denominado Duración. La clase Evaluación participa en una relación con la clase Curso.

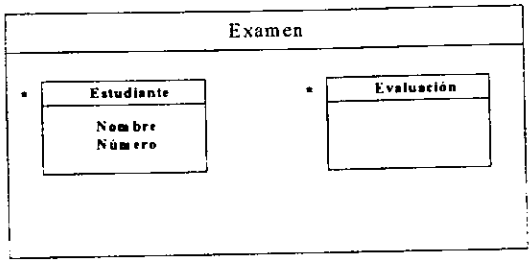

Figura 5.4. Examen es una Agregación de Estudiante y Evaluación

En FUSION se consideran las relaciones de generalización y de especialización:

- ***Generalización***. En esta relación existe una clase llamada supertipo que hereda sus atributos a todos sus subtipos, además de que los subtipos pueden tener sus propios

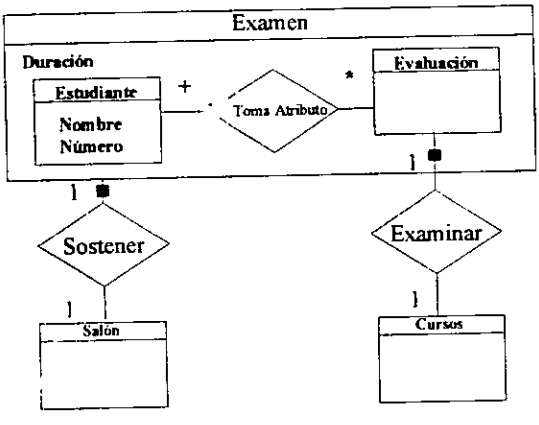

Figura 5.5. La Agregación puede ser tratada como una clase

atributos y tal vez participar en otras relaciones. Su principal caracteristica es que todos los objetos de un subtipo también pertenecen al supertipo. En la figura 5.6. se muestra un ejemplo simple de generalización en donde el triángulo negro indica que los subtipos son independientes, ya que cuando es un triángulo vacío se indica que los subtipos tienen algo en común.

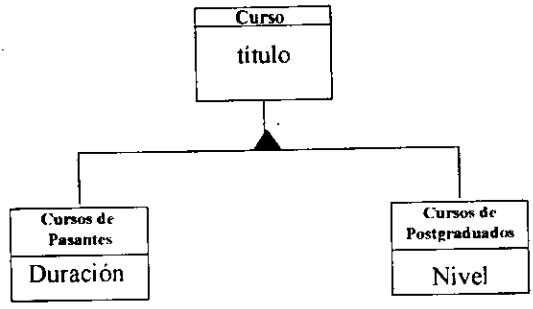

Figura 5.6. Generalización

- ***Especialización.*** Un subtipo es una versión especializada de uno o más supertipos (especialización múltiple) por lo que una subclase hereda todos los atributos y relaciones de sus superclases, esto es, la especialización múltiple es la manera en la que una subclase es separada en dos o más superclases.

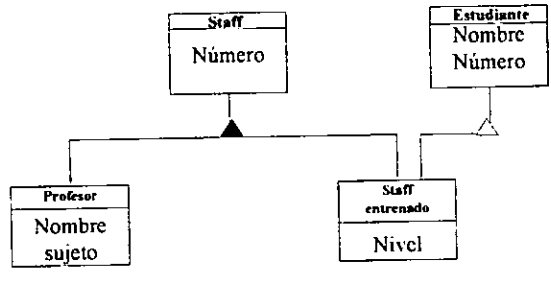

Figura 5.7.Especialización Múltiple

Actividad 2. Determinación de la Interfaz del Sistema.
En esta actividad se determina la interfaz del sistema que es el conjunto de operaciones del sistema a las que se les puede responder y los eventos que se pueden producir (eventos de salida).

" ..un sistema es modelado como una entidad activa que coopera con otras entidades activas, llamadas agentes. El sistema y los agentes se comunican a través de mandar y recibir eventos. Cuando los eventos **son** recibidos por el sistema pueden causar un

cambio de estado y se van a producir eventos (de salida). Un evento de entrada y sus efectos asociados es conocido como una operación del sistema" [4]

Para determinar la interfaz del sistema primero ***se identifican los agentes*** involucrados que invocan una operación de sistema, cada una de ***las operaciones del sistema*** y ***los eventos de salida,*** se registran en el diccionario de datos para que se elaboren los escenarios.

Un *escenario* se utiliza para mostrar la manera en que se lleva a cabo el flujo de los eventos entre los agentes y el sistema. Se indican los agentes, la tarea que debe de realizar el sistema y la secuencia de la comunicación que se ve involucrada para este fin. Los escenarios en FUSION son similares a los diagramas de trazos utilizados en OMT y a los diagramas de interacción del método de Booch.

Se construye un Modelo del Sistema del Objeto que es un Modelo del Objeto pero indica sus límites dependiendo del servicio que provee el conjunto de clases, relaciones y eventos. Este modelo es similar a la actividad de organizar el sistema en subsistemas del método OMT y en el de Booch la tares de agrupar las clases en categorías de clases.

Actividad 3. Desarrollo del Modelo de la Interfaz.
Con el modelo de la interfaz se describe el comportamiento del sistema ya que con el modelo del objeto solo se describe la estructura del sistema. El modelo de la interfaz hace uso de dos modelos para lograr capturar los diferentes aspectos del comportamiento del sistema: modelo del ciclo de vida y modelo de operación:

- ***Modelo del ciclo de vida.*** La forma en que el sistema interactua con los agentes (ambiente) no puede ser de manera arbitraria por lo que se requiere que exista una secuencia permitida para esta comunicación llamada ciclo de vida. Este modelo se utiliza para mostrar la secuencia permitida de la comunicación entre el sistema y su ambiente(agentes).
- ***Modelo de Operación.*** Este modelo sirve para mostrar los efectos que cada una de las operaciones del sistema causa, así como los eventos de salida que se generan. Algunas de las operaciones del sistema de las que se realizan sus esquemas pueden ser: crear una instancia nueva de una clase, cambiar el valor de algún atributo de un objeto ya existente, incluir o eliminar un par de objetos involucrados en una relación y mandar un evento a un agente.

Modelo del Ciclo de Vida.
Para la construcción del modelo del ciclo de vida se utilizan *expresiones del ciclo de vida,* que definen el patrón a seguir para que se lleve a cabo la comunicación entre el sistema y su ambiente.

Como primer paso ***se generaliza a los escenarios***, es decir, se forman las expresiones del ciclo de vida.

La sintaxis y la semántica de las expresiones son las siguientes:

Nombre:	Cualquier nombre de evento de entrada o de salida puede ser utilizado en una expresión. A los eventos de salida se les da el prefijo #
Operadores:	En donde x y y son expresiones del ciclo de vida.

[4] Coleman. Dereck/Arnold. Patrick. **Object-Oriented Development:The FUSION Method,** pagina 45.

Concatenación:	x , y donde x es seguida por y.
Alternancia:	$x \mid y$, donde x o y ocurren.
Repetición:	x^* donde x ocurre cero o más veces.
	x^+ donde x ocurre una o más veces.
Opcional:	$[\, x \,]$ donde x es opcional.
$x \mid\mid y$	donde se van a ordenar de manera alternada los elementos de x y y.
Substitución:	Una expresión puede ser nombrada en una substitución; el nombre tal vez sea utilizado para otras expresiones pero las substituciones no deben ser recursivas. Nombre = Expresión del ciclo de vida.

A partir del modelo de objeto de un sencillo sistema de un banco se crearon las siguientes expresiones del ciclo de vida para llevar a cabo la apertura de una cuenta nueva, una transacción y la consulta de saldo:

Apertura = abrir_cuenta. #Num_cuenta. ConsultaSaldo' . deposito

Para realizar una transacción:

Transacción =retirar_efectivo . (#dar_efectivo | #fondos_insuficientes) | deposito

Para realizar una consulta de saldo:

ConsultaSaldo = revisar_balance. #balance_actual

Con estas expresiones se puede llevar a cabo la construcción de un modelo del ciclo de vida para el sistema de un banco:

Lifecycle Sistema_Banco = Apertura. (Transacción | ConsultaSaldo)'

Apertura = abrir_cuenta . #num_cuenta . ConsultaSaldo' . deposito

Transacción = retirar_efectivo . (#dar_efectivo | #fondos_insuficientes) | deposito

ConsultaSaldo = revisar_balance . #balance_actual

Modelo de Operación.

Para la construcción de este modelo de operación se utilizan precondiciones y postcondiciones con las que se especifican las operaciones del sistema. Estas precondiciones y postcondiciones son predicados lógicos que pueden ser falsos o verdaderos. Estos predicados están formados por una lista de cláusulas que tienen que ser verdaderas para que el predicado sea verdadero. Si cualquiera de las cláusulas es falsa, entonces el predicado es falso.

Para cada una de las operaciones del sistema ***se establece un esquema*** que consiste en un texto estructurado cuya sintaxis y semántica se muestra a continuación:

Operation: **Nombre.**
Identificador de la operación.

Description: **<Texto>**
Descripción de la operación.

Reads: **<Supplied elementos>**
En donde *elementos* es una lista de todos los valores (objetos y atributos) a los que la operación puede tener acceso pero no los cambia; la palabra *supplied* indica que el identificador es un parámetro de la operación.

Changes: **<New elementos>**
Elementos es una lista de valores (objetos y atributos) a los que la operación puede tener acceso y tal vez los cambie; la palabra *new* indica que la operación del sistema crea un objeto nuevo en el estado del sistema.

Sends: **AgenteyEventos**
Se refiere al nombre de un agente y todos los eventos que tal vez la operación le manda al agente.

Assumes: **Precondición**
La precondición va a definir las características bajo las cuales se invoca a la operación.

Result: **Postcondición**
La postcondición describe la manera en la que la operación cambia el estado del sistema y los eventos que se mandan a los agentes durante la operación.

Las precondiciones se mantienen antes y durante la invocación de una operación y solo pueden hacer referencia a los parámetros y partes del sistema definidos en Reads y Changes.

Si la precondición es falsa no se conoce el efecto que se tendrá al invocar una operación; si la precondición es verdadera tal vez se decida omitirla, por lo que se puede observar que una operación puede ser invocada desde cualquier estado de la precondición.

Coma ***primer paso*** para el desarrollo de este modelo de operación se describen las cláusulas que forman parte de Result utilizando para esto el modelo del ciclo de vida que determina que eventos son de salida.

Como ***segundo paso*** se revisa cada una de las cláusulas para evitar que se obtengan valores no deseados.

El ***tercer paso*** es determinar las cláusulas para Assumes y las cláusulas para Result que pudieran haber sido omitidas.

Como ***cuarto paso*** se verifica que las cláusulas sean las apropiadas.

El ***quinto paso*** es actualizar el diccionario de datos con las operaciones del sistema y los eventos.

El ***sexto paso*** es extraer las cláusulas Sends, Reads y Changes a partir de Assumes y Result. El propósito de las cláusulas sends, reads y changes es establecer que objetos y agentes se verán involucrados durante la operación. Se puede decir que el esquema construido es el correcto si todos los valores iniciales satisfacen la cláusula Assumes y los valores finales que existen satisfaces la cláusula Result.

El esquema para la operación de la apertura de una cuenta nueva en un banco podría ser el siguiente:

Operation:	Abrir_cuenta
Description:	Abrir una cuenta nueva para un cliente
Reads:	supplied nombre: nombre_cliente (el agente provee un valor)
Changes:	banco new cuenta (se crea un nuevo objeto)
Sends:	cliente : {num_cuenta} (solo el agente cliente recibe el mensaje)
Assumes:	
Result:	Se ha adicionado una cuenta al sistema balance_cuenta ha sido ajustado a cero cuenta.nombre ha sido asignado a nombre A cuenta.num_cuenta se le ha asignado un valor único El num_cuenta ha sido asignado a cliente.

Actividad 4. Revisión de los modelos del análisis.
Como se ha mencionado en capítulos anteriores, se puede decir que la fase del análisis ha sido concluida cuando los modelos construidos son apropiados para ser utilizados en la etapa del diseño. Pero antes de pasar a la etapa del diseño es necesario hacer una revisión y evaluación a cada una de los modelos para verificar que han sido desarrollados de la manera más completa posible y revisar su consistencia.

Al hablar de un modelo completo se refiere a que se ha tomado en cuenta cada una de las abstracciones que forman parte del dominio del problema recurriendo a la especificación de los requerimientos para una revisión cuidadosa y en caso de alguna duda se debe recurrir al cliente.

A continuación se verifica que cada uno de los escenarios ha sido considerado para la construcción del modelo del ciclo de vida, que a cada una de las operaciones se le ha construido su esquema, que en el modelo del objeto se ha considerado toda la información de carácter estático y ha sido incluida en el diccionario de datos.

La consistencia se refiere a que no existan contradicciones entre los modelos así como la consistencia interna de cada uno de éstos.

Los autores de FUSION dan una guía para verificar la consistencia de los modelos del análisis:

" Todas las clases, relaciones y atributos mencionados en el modelo de operación deben aparecer en el modelo del sistema del objeto....

El límite del modelo del sistema del objeto es consistente con el sistema de la interfaz dado por el modelo del ciclo de vida.

Todas las operaciones del sistema en el modelo del ciclo de vida tienen un esquema.

Todos los identificadores dados en los modelos del análisis están registrados en el diccionario de datos.

...la salida de los eventos en el modelo del ciclo de vida y el modelo de operación deben ser consistentes. El esquema para una operación debe generara los eventos de salida que siguen en los escenarios en el modelo del ciclo de vida.

...elegir ejemplos de escenarios y definir el cambio de estado que cada una de ellos causa.

Entonces se ejecutan los escenarios utilizando el esquema para definir el comportamiento de cada uno de las operaciones del sistema. Verificar que el resultado es el esperado" [5]

5.3. Diseño.

En la etapa del diseño se define la implementación de cómo va a funcionar el sistema para obtener un grupo de objetos que interactúan para que se puedan ejecutar las operaciones del sistema descritas en el modelo de operación.

Durante el diseño se construyen cuatro modelos a partir de los que se puede hacer la codificación, evaluación y mantenimiento del sistema:

1. Gráficas de Interacción de Objetos.
2. Gráficas de Visibilidad.
3. Descripciones de Clase.
4. Gráficas de Herencia.

1. Gráficas de Interacción de Objetos.

En estas gráficas se muestra los objetos involucrados para realizar una operación y la manera en que se comunican a través de mensajes para llevar a cabo la función especificada en el modelo de operación, por lo que se construye una gráfica de interacción de objetos para cada una de las operaciones descritas en el modelo de operación.

Estas gráficas se construyen con los llamados *Objetos de Diseño* que están conectados por medio de flechas que representan el paso de los mensajes.

Cada una de estas gráficas va acompañada de una descripción de la operación que esta siendo implementada y una definición de los mensajes mostrados.

Los *objetos de diseño* tienen un nombre que los identifica y pueden tener valores de atributos además de un método que se invoca por medio de mensajes con el fin de que el método lleve a cabo una tarea.

En la figura 5.8. se muestra el paso de mensajes entre los objetos de diseño donde el objeto controlador tiene el nombre *d* con el que se le identifica y *Directorio* es la clase a la

[5] idem. pag. 60.

que pertenece. El objeto colaborador es el archivo *f* y colabora con *d* para llevar a cabo la operación.

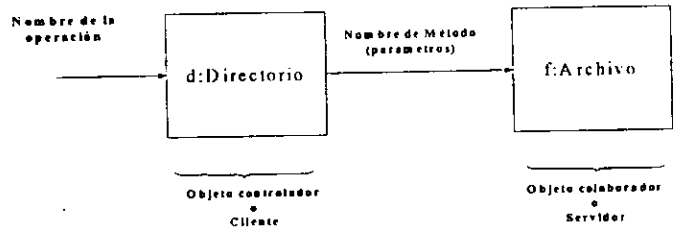

Figura 5.8. Paso de Mensajes entre los objetos del diseño

En la figura 5.8. solo se muestran objetos pero también se pueden definir colecciones de objetos de la misma clase. En la figura 5.9. se puede ver que la colección de objetos tiene el nombre *archivos* donde cada uno de los objetos de la colección pertenece a la misma clase llamada *File*. Cuando se manda un mensaje a esta colección de objetos todos los objetos reciben el mensaje y cada uno de ellos ejecuta el método.

El mensaje se etiqueta con un nombre y los parámetros actuales, donde un parámetro consiste de un tipo y un nombre opcional. El tipo se refiere al tipo de los argumentos que se requieren en el mensaje y el nombre se requiere solo para identificar un valor en particular. Cuando se manda un mensaje y se obtiene un resultado, el tipo del valor que retorna el método se indica después de la lista de parámetros actuales:

Nombre del mensaje(lista de parámetros):tipo del valor retornado

Figura 5.9 Colección de Objetos de la clase File

En algunas ocasiones cuando se manda un mensaje al servidor para invocar un método, el método retorna un valor que se indica por medio de una flecha que tiene la dirección servidor-cliente y su explicación es incluida en la descripción de la operación como se muestra en la figura 5.10.

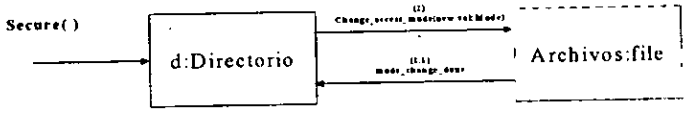

Figura 5.10. Paso de mensajes entre un objeto y una colección de objetos.

Cuando se invoca un método, posiblemente este método también va a mandar mensajes a otros objetos, por lo que estos métodos invocados tienen que ser llevados a su fin antes de que la invocación inicial sea completada. Para indicar la secuencia de los mensajes se etiquetan con un número dentro de paréntesis como se muestra en la figura 5.10. donde (1) y (1.1) indican que la invocación del método que resulta del mensaje (1.1) debe completarse antes de que regrese el control al método invocado por el mensaje (1).

Para indicar que se crea un objeto nuevo se utiliza la palabra *new*, y para su inicialización se manda un mensaje especial etiquetado con el nombre *create* junto con los parámetros de invocación apropiados. Si lo anterior no se lleva a cabo a este objeto nuevo no se le podrá mandar mensajes ya que se dice que se encuentra en un estado indefinido.

La construcción de la gráfica de interacción del objeto para cada una de las operaciones involucra cuatro actividades:

Actividad 1. Se identifican los objetos involucrados en el proceso.
Para esta identificación se recurre a los esquemas del modelo de operación.

Actividad 2. Se establece el papel que juega cada uno de los objetos.
Se identifican los objetos que harán el papel de controlador (clientes) para que invoquen a

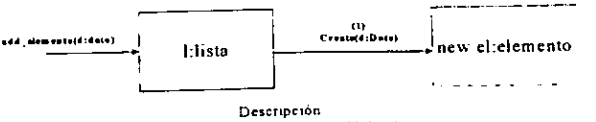

Figura 5.11.Creación de un objeto dinámico.

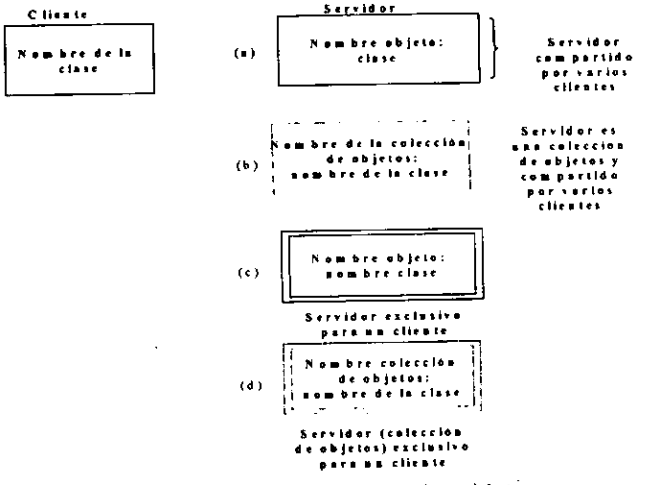

Figura 5.12. Notación para los objetos cliente y servidor

las operaciones del sistema.

También se identifican los colaboradores(servidores) que cooperan con los controladores para que se puedan implementar las operaciones del sistema.

Actividad 3. Se decide que mensajes van a pasar entre los objetos involucrados, esto es, se describe la forma en que todos colaboran para que se lleve a cabo la operación.

Actividad 4. En el diccionario de datos se registra la manera en que los objetos colaboran incluyendo una descripción de cada método.

La consistencia que estas gráficas de interacción del objeto tienen con los modelos del análisis se verifica a través de una revisión de los efectos funcionales de los objetos de acuerdo a la especificación de las funciones de la operación del sistema en el modelo de operación.

2. Gráficas de Visibilidad.
En las gráficas de visibilidad se muestra la manera en que se lleva a cabo los patrones de comunicación entre los objetos, ya que en las gráficas de interacción de objetos solo se indica el paso de mensajes.

Para que la comunicación entre objetos se pueda llevar a cabo es necesario que el objeto cliente haga referencia al objeto servidor. La referencia de un objeto a otro indica que debe existir una identificación y un direccionamiento de objetos.

En las gráficas de visibilidad se hace *visible* la forma en la que un objeto cliente hace referencia a un objeto servidor para mandarle un mensaje, en otras palabras, se va a hacer *visible* al servidor para que se le puedan mandar mensajes.

La notación para los objetos clientes y servidores (ya sea un objeto o una colección de objetos) se muestra en la figura 5.12.

La referencia se hace a través de flechas de visibilidad y puede ser dinámica o permanente como se muestra en la figura 5.13.

La manera de utilizar las flechas de visibilidad depende del tipo de relación de visibilidad que el diseñador decida utilizar. Las diferentes relaciones de visibilidad que soporta este método son:

Figura 5.13.Flechas de Visibilidad

A) Referencia de Vida.
Cuando un cliente manda un mensaje a un servidor para la invocación de un solo método se hace a través de una referencia de visibilidad dinámica.

Pero cuando a un servidor se le van a enviar mensajes para una invocación continua de métodos se hace una referencia de visibilidad permanente.

B) Servidor Visible.
Cuando un objeto servidor es utilizado exclusivamente por un cliente se representa con el tipo de servidor del inciso (c) de la figura 5.12., y cuando es compartido por más de un cliente se denota con el tipo de servidor del inciso (a).

C) Servidor Restringido.
La vida de un objeto inicia cuando es creado y termina cuando se le borra. Cuando un servidor es borrado al borrar el cliente con el que está relacionado, se dice que el servidor es un servidor restringido. En la figura 5.14. se muestra este tipo de relación de visibilidad donde el servidor(es) se coloca dentro del cliente.

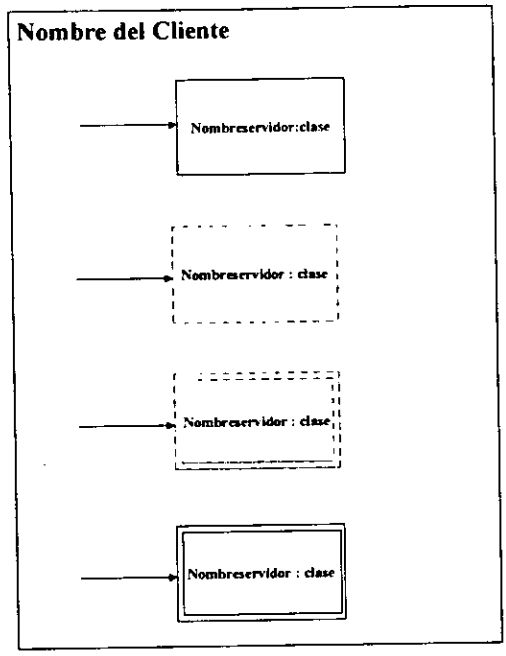

Figura 5.14. Servidores Restringidos

D) Referencia Mutable.
Cuando una referencia puede ser reasignada después de que ha sido inicializada, se dice que es mutable. Pero si la referencia no se asigna después de su inicialización, se dice que la mutabilidad de la referencia es constante, lo que significa que los servidores no podrán ser cambiados (sus atributos si podrán cambiar) y se indica colocando la palabra *constant como* prefijo del nombre del servidor.

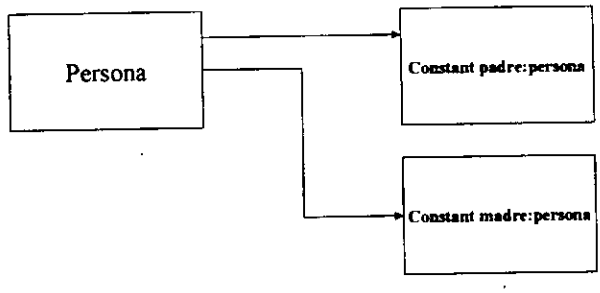

Figura 5.15. Mutabilidad de referencia constante

Las gráficas de visibilidad se construyen a partir de la información descrita en las gráficas de interacción de objetos, ya que para cada flecha que va de un objeto controlador al colaborador(es) se determina el tipo de relación de visibilidad de referencia adecuada.

Una vez terminada la tarea de elaborar las gráficas se hace una revisión de los modelos, verificando que exista consistencia con los modelos del análisis. Se revisa que las relaciones entre las clases identificadas en la etapa del análisis se mantengan en la elaboración de estas gráficas de visibilidad. También se comprueba que los servidores exclusivos no tengan más de un cliente y que cada paso de mensajes definidos en las gráficas de interacción de objetos haya sido tomado en cuenta en las gráficas de visibilidad.

3. Descripción de Clase.
Una vez construidas las gráficas de visibilidad, el siguiente paso es la elaboración de una descripción de clase para cada una de las clases identificadas en el análisis; dichas descripciones de clase son la base para llevar a cabo la codificación del sistema en la etapa de la implementación.

La información requerida para esta descripción de clase es la contenida en el modelo del sistema del objeto, en las gráficas de interacción del objeto y en las gráficas de visibilidad.

La descripción de clase consiste en una descripción de la estructura interna de la clase, esto es:

- De las gráficas de interacción del objeto se toman los métodos y parámetros, es decir, la interfaz de la clase.
- El modelo del sistema del objeto y el diccionario de datos provee los atributos de datos, ya que los atributos de las clases definidos durante el análisis se implementan a través de los atributos de datos definidos en la etapa del diseño.

- De las gráficas de visibilidad se toman los atributos de los objetos, ya que las referencias permanentes se implementan a través de estos atributos de objeto, en donde el atributo se refieren al objeto y el tipo es la clase a la que pertenece el objeto en las gráficas de visibilidad.
- También se incluye la información de las dependencias de herencia de cada clase, pero esta actividad será al final ya que primero se tienen que elaborar las gráficas de herencia.

La sintaxis para la descripción de cada clase es la siguiente:

class <*nombre de la clase*> **is a** <*nombre de su superclase*>

//para cada atributo

{Visibilidad del servidor}
attribute <*nombre del atributo*> : / **shared** | **exclusive** / < *tipo* >

{Servidor restringido}
attribute <*nombre del atributo*> : / **bound** | **unbound** / < *tipo* >

.
.
.

//para cada método

method < *nombre del método* > (*lista de argumentos*)

.
.
.

endclass

4. Gráficas de Herencia.
En el análisis se identificaron las relaciones de los subtipos abstractos entre las clases, estas relaciones son las de generalización y especialización que se utilizan como un mecanismo para mostrar las relaciones entre las clases en el modelo del dominio del problema, lo que significa que solo se utilizan de una manera descriptiva, por lo que no son utilizadas para el diseño del sistema (dominio de la solución) ni para la implementación del mismo.

Sin embargo, en la etapa del diseño se lleva a cabo la construcción de las gráficas de herencia que se utilizan para llevar a cabo la implementación de estas relaciones de herencia entre las clases.

La notación para estas gráficas de herencia es la misma que se utiliza para la construcción del modelo del objeto en la etapa del análisis.

La información necesaria para la elaboración de estas gráfica se obtiene de las gráficas de interacción de objeto, de las de visibilidad, del modelo del sistema del objeto y de la descripción de clase de la siguiente manera:

- Del modelo del sistema del objeto se considera y se trata de conservar las relaciones de especialización y generalización de los subtipos.

- De las gráficas de interacción de objetos se busca la especificación de la interfaz de las clases (métodos y parámetros) mostradas en estas gráficas. En la descripción de clase también se documento esta especificación de la interfaz de clase.
- La descripción de clase se utiliza para identificar los casos en los que exista una funcionalidad común haciendo una revisión cruzada entre estas descripciones de clases con el fin de crear clases abstractas nuevas para definirlas en la estructura de herencia.
- También las gráficas de visibilidad se revisan de manera cruzada para determinar si existen estructuras de referencia en común que pueden ser definidas creando clases abstractas nuevas.

Una vez que el diseñador termina la construcción de las gráficas de herencia se actualizan las descripciones de clase incluyendo las clases abstractas nuevas y las relaciones de herencia definitivas.

CONCLUSIONES.

El propósito de este trabajo es exponer los principales métodos de análisis y diseño orientados a objetos con el fin de conocer las diferentes opciones con las que se cuenta al querer llevar a cabo el desarrollo de software orientado a objetos y no hacer una comparación entre los métodos aquí tratados, ya que la elección de un método depende de las necesidades y criterios personales del desarrollador del software.

Ahora bien, la elección de un método puede ser de dos maneras:

1) Elegir alguno de los métodos disponibles para realizar el proyecto completo.

2) Hacer una combinación de los métodos tomando lo mejor que cada método aporta para el desarrollo del proyecto, sin embargo esta elección puede implicar un riesgo, ya que se debe ser muy cuidadoso para lograr una buena consistencia en el proyecto. El mayor riesgo sería hacer una combinación de los métodos en cada etapa del proyecto, además de que sería una tarea muy laboriosa, aunque esto podría ser la base para un proyecto para desarrollar un método nuevo para el análisis y diseño orientado a objetos, como es el caso del método FUSION. Se correría un riesgo menor al tomar un solo método para desarrollar una etapa y adaptar los resultados obtenidos de tal manera que se puedan utilizar en el desarrollo de la siguiente etapa con un método diferente.

Algunos de los criterios que se deben tener en cuenta para la elección de un método son los siguientes:

- El método debe proveer una guía que ayude a llevar a cabo el desarrollo del sistema paso a paso a partir de los requerimientos hasta la codificación. Los métodos expuestos en este trabajo dan por sentado que el analista cuenta con un documento en el que ya se han establecido correctamente todos los requerimientos y necesidades del usuario.

- Es importante que de alguna manera se pueda verificar que se ha concluido correctamente cada una de las actividades o etapas (análisis y diseño) para que se pueda continuar con el proceso.

- Respecto a los errores, se debe tener en cuenta que existen métodos como el OMT por ejemplo, que propone una guía para prevenir que se lleguen a cometer errores, y otros métodos como el método FUSION que da una guía para verificar si se han cometido algunos errores. La elección de cuál conviene al desarrollador es de él.

- Es recomendable que de alguna manera se pueda llevar a cabo un registro de las decisiones importantes tomadas a lo largo del proceso, ya que puede ser de gran utilidad para realizar fácilmente el mantenimiento del sistema. Este registro podría ser a través de tarjetas CRC o diccionarios de datos como lo hacen algunos métodos aquí tratados.

- Es de una gran utilidad que existan herramientas disponibles que ayuden al soporte del método, pero si no se cuenta con estas herramientas una opción sería que el método elegido cuente con una notación que sea fácilmente elaborada a mano. Sin embargo no se recomienda elegir un método sin contar con las herramientas necesarias para soportarlo, ya que el análisis y el diseño requieren de un gran manejo de información que implica la construcción diagramas complejos.

Es importante recordar que no importa qué método se elija, ya que obtener el resultado final deseado implica que los desarrolladores pongan a trabajar su habilidad y creatividad y no sólo seguir al pie de la letra las actividades del método.

La creatividad es de vital importancia dentro del paradigma orientado a objetos, pues representar el problema de acuerdo a la forma en que se comportan los objetos en el mundo real da como resultado un buen modelado de la solución, que es la base para la codificación optima del sistema.

Haber desarrollado este trabajo ha sido de vital importancia para actualizarme en los avances realizados referentes al desarrollo del software bajo el Paradigma Orientado a Objetos, lo que en la vida laboral de cualquier alumno y egresado de la carrera de Matemáticas Aplicadas y Computación puede ser relevante para su buen desempeño profesional y laboral pues actualmente los lenguajes orientados a objetos son los más utilizados para el desarrollo de software.

Métodos / Características	OOSA	HOOD	RDD	Coad & Yourdon	OMT	Booch	FUSION
Identificación de:							
Clases	si	no	si	si	si	si	si
Objetos	si	si	si	si	si	si	si
Atributos	si	no	si	si	si	si	si
Operaciones	si	si	si	si	si	si	si
Herencia	si	no	si	si	si	si	si
Generalización	no	no	no	si	si	si	si
Agregación(Whole-Part)	no	no	si	si	si	si	si
Otras relaciones	si	si	si	no	no	si	no
Representación Gráfica:							
Clases y Objetos	**	**	**	***	***	***	***
Atributos	**	*	no	***	***	***	***
Relaciones	***	**	**	***	***	***	***
Trazo de Eventos	no	no	no	no	***	***	***
Transición de Estado	***	no	no	***	***	***	***
Flujo de Datos	***	no	no	no	***	***	***

*** Buena
** Regular
* Mala

BIBLIOGRAFIA.

1. **Object – Oriented Analysis,**
   Coad, Peter/Yourdon, Edward.
   Prentice Hall, 1993.

2. **Object – Oriented Information Systems. Planning and Implementation,**
   Taylor, Davis.
   WILEY, 1993.

3. **Software Enginnering,**
   Sommerville, Ian.
   Addison – Wesley, 1993.

4. **Object – Oriented Analysis and Design with Applications,**
   Booch, Grady.
   Benjamin/Cumming, 19994.

5. **Object – Oriented Modeling and Design,**
   Rumbaugh/Blaha,
   Prentice Hall, 1991.

6. **Object – Oriented Analysis and Design,**
   Martin, James/J. Odell, James.
   Prentice Hall, 1994.

7. **Principles of Object – Oriented Analysis and Design,**
   Martin, James/J. Odell, James.
   Prentice Hall, 1990.

8. **Object Lifecycles: Modeling the World in States,**
   Shlaer, Sally/J.Mellor, Stephen,
   Prentice Hall. 1992.

9. **Designing Object – Oriented Software,**
   Wirfs-Brock, R./Wilkerson, B.,
   Prentice Hall, 1990.

10. **Object – Oriented Development:The FUSION Method,**
    Coleman, Derek/Arnold, Patrick,
    Prentice Hall, 1994.

11. **Succeeding with Objects. Decision Frameworks for Project Management,**
    Goldberg, Adela/S. Rubin, Kennet,
    Addison Wesley, 1995.

12. **Object – Oriented Systems Analysis and Design,**
    J. Norman, Ronald,
    Prentice Hall, 1996.

13. **Mainstream Objects, an Analysis and Design Approach for Business,**
    Yourdon, Ed.
    Prentice Hall, 1995.

14. **Object – Oriented Software Construccion,**

Meyer, Bertrand,
Prentice Hall, Second Edition, 1997.

15. **Object – Oriented Software Engineering,**
Jacobson, Ivar,
Addison Wesley, 1992.

16. **Best of Booch; Designing Strategias for Object Technology,**
Booch, Grady, Edited by Ed Egkholt,
Sign Book & Multimedia, 1996.

17. **What Every Programmer Should Know about O-O Design,**
Page-Jones, Meilier,
Dorset House Publishing, 1995.

18. **Using CRC Cards, an informal Approach to Object – Oriented Development,**
Wilkinson, Nancy,
Sign Books, 1996.