CAPITULO I

1. Conceptos Generales

1.1. Teoría de Análisis y Diseño Orientado a Objetos

1.1.1. Paradigma Orientado a Objetos.

Paradigma se define como el ejemplo a seguir o entidad ejemplar de un conjunto.

El paradigma Orientado a Objetos nace a finales de los años 60, pero recién a mediados de los años 90 comenzó a usarse como un modelo alternativo para solucionar problemas específicos para la producción de sistemas cada vez más complejos.

Las principales ventajas de la utilización para los niveles de dirección y técnicos de este paradigma son:

- Reutilización (de componentes de software) lo cual permite un desarrollo de software rápido y programas de mejor calidad.
- Mayor facilidad en cuanto al mantenimiento.
- Menores efectos colaterales al momento de hacer cambios, y;
- Mayor facilidad para adaptar y escalar.[1]

El paradigma Orientado a Objetos se puede definir como:

Una disciplina de ingeniería de desarrollo y modelado de software que permite construir de manera fácil sistemas complejos a partir de componentes individuales.

Una filosofía de desarrollo y empaquetamiento de software que permite crear unidades funcionales extensibles y genéricas, de forma que se las pueda aplicar según las especificaciones del sistema a desarrollar.

El paradigma Orientado a Objetos se está constituyendo en una metodología de análisis, diseño y desarrollo para producir software eficiente y barato.

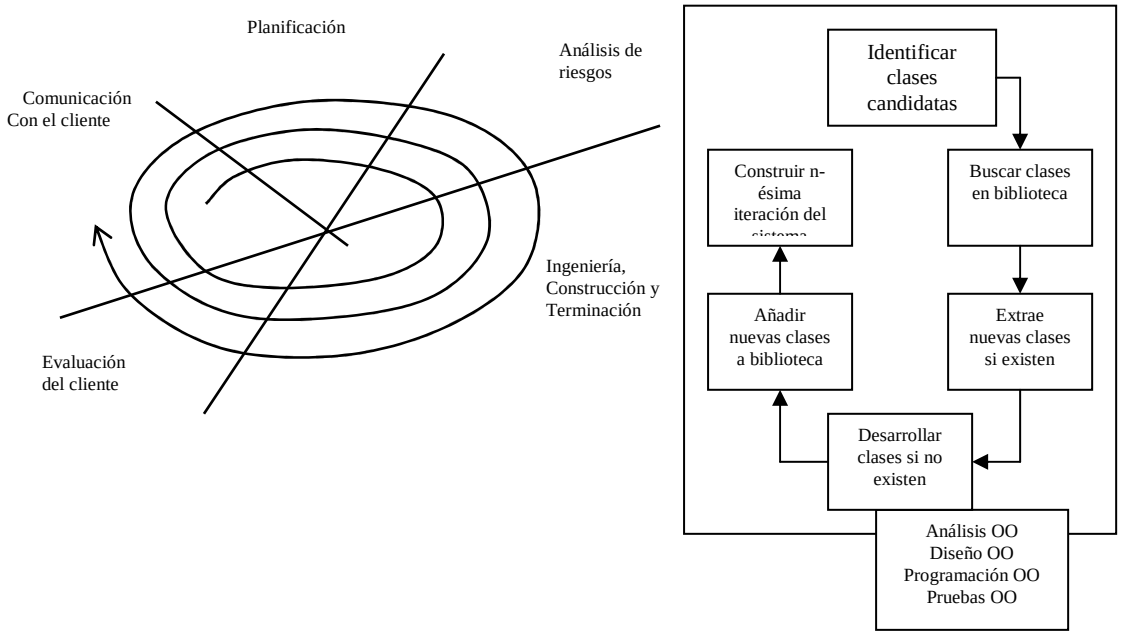
El mejor paradigma para Ingeniería de Software Orientado a Objetos es aquel que admite que los Sistemas Orientados a Objetos tienden a evolucionar con el tiempo.

El paradigma de Ingeniería de Software maneja varios modelos evolutivos, entre ellos el de Ensamblaje de componentes que es un modelo en espiral que se basa en la identificación y eliminación de problemas de alto riesgo a través de un proceso de diseño cuidadoso de tipo cíclico. Por ello un modelo evolutivo del Proceso acoplado con un enfoque que fomenta el ensamblaje de componentes, es la mejor opción. Por esto el proceso Orientado a Objetos se mueve a través de una espiral evolutiva que comienza con:

---

[1] Ingeniería de Software, Un enfoque práctivo, Roger Pressman pág 367

- Comunicación con el usuario.- Se define el dominio del problema y se identifican las clases básicas.
- La planificación.
- El Análisis de riesgos.
- Ingeniería, construcción y terminación.- Debido a que la Ingeniería de Software Orientada a Objetos hace incapié en la reutilización, las clases se buscan en una biblioteca de clases existentes, cuando esta no es encontrada, se procede a aplicar Análisis Orientado a Objetos (AOO), Diseño Orientado a Objetos (DOO), Programación Orientada a Objetos (POO) y Pruebas Orientada a Objetos (PROO) para crear la clase y los objetos derivados de la clase.
- Evaluación del cliente
- Comunicación con el cliente.



1.1.1.1 Modelos Orientados a Objetos.

"Un modelo es una simplificación completa y autoconsistente de la realidad, creado para comprender mejor un sistema."[2]

Los modelos orientados a objetos sirven para comprender problemas, comunicarse con los expertos en el manejo de una aplicación, modelar empresas, preparar documentación y diseñar programas y bases de datos.

El modelo de análisis ayuda a abstraer la parte más importante del dominio de la aplicación dejando a un lado la implementación.

[2] El lenguaje Unificado de Modelado, Grady Booch y otros, pág. 81

Los objetos del dominio de la aplicación constituyen el marco de trabajo del modelo de diseño, los cuales se implementan en términos de objetos que entiende la computadora. Adicionalmente, el modelo de diseño se implementa en un lenguaje de programación, base de datos o hardware.

"La esencia del desarrollo orientado a objetos es la identificación y organización de conceptos del dominio de la aplicación y no de su representación final en un lenguaje de programación tanto si este es orientado a objetos como si no lo es".[3]

Un modelo es una abstracción de algo, cuyo objetivo es comprenderlo antes de construirlo. La abstracción permite enfrentarnos a la complejidad.

Los objetivos de los modelos son los siguientes:

Probar una entidad física antes de construirla.- Los modelos son más baratos que construir un sistema completo y hace posible corregir desde un principio los errores.

Comunicación con el cliente.- Por medio de los modelos se puede imitar parte o todo el comportamiento externo de un sistema.

Visualización.- Un modelo permite tener una visión del sistema antes de implementarlo.

Reducción de la complejidad.- Los modelos permiten separar detalles no esenciales por lo que es más fácil la comprensión de un sistema.

1.1.1.2. Análisis Orientado a Objetos.

El Análisis Orientado a Objetos se basa en conceptos como son Objetos y Atributos, el todo y las partes, clases y miembros. El paradigma Orientado a Objetos ha tenido una evolución similar al paradigma de programación estructurada, es decir primero surgieron los lenguajes de programación estructurados que dieron paso al análisis estructurado. El paradigma Orientado a Objetos ha seguido el mismo camino: el uso de la Programación Orientada a Objetos modificó las técnicas de diseño para adaptarlas a los nuevos lenguajes y en la actualidad se ha empezado a usar las técnicas de análisis y diseño para esta forma de desarrollar software.

El Análisis Orientado a Objetos ofrece una nueva técnica para el análisis de requisitos. En lugar de considerar al software desde la perspectiva clásica de entrada/proceso/salida de los métodos estructurados clásicos, se basa en modelar el sistema mediante los objetos que forman parte de él y las relaciones estáticas (herencia y composición) o dinámicas (uso) entre estos objetos. Este enfoque pretende construir modelos que se ajusten mejor al problema real a partir del conocimiento del dominio del problema. La ventaja del Análisis Orientado a Objetos es que al utilizar objetos como abstracciones

[3] Modelado y diseño orientado a objetos, James Rumbaugh y otros, pág. 23

del mundo real permite centrarse en aspectos significativos del dominio del problema es decir en las características de los objetos y las relaciones que se establecen entre ellos.

Este enfoque se puede considerar como una extensión del modelado de datos Diagrama Entidad Relación (DER), que se utiliza en los métodos estructurados, DER está orientado al diseño de bases de datos y se centra en la identificación de los datos que maneja un sistema y en las relaciones estáticas que se establecen entre los datos, pero en el Análisis Orientado a Objetos, los objetos encapsulan tanto atributos como procedimientos (operaciones que se realizan entre los objetos), e incorpora conceptos como el polimorfismo o la herencia que facilitan la reutilización del código.

El uso de Análisis Orientado a Objetos puede facilitar mucho la creación de prototipos, los Objetos pueden ser reutilizables y se puede crear un catálogo de objetos que podemos usar en sucesivas aplicaciones.

1.1.1.3. Ventajas del Análisis Orientado a Objetos (AOO).

Dominio del problema.- Conduce a pensar acerca del problema en términos del mundo real en vez de pensar en términos de un ordenador.

Comunicación.- El AOO es más simple y está menos relacionado con la informática, permitiendo así una mejor comunicación entre el analista y el usuario.

Consistencia.- Los Objetos encapsulan tanto atributos como operaciones . Debido a esto el AOO reduce la distancia entre el punto de vista de los datos y el punto de vista del proceso, minimizando las inconsistencias entre los modelos.

Expresión de características comunes.- El paradigma Orientado a Objetos utiliza la herencia para expresar explícitamente las características comunes de una serie de objetos, poniendo énfasis en la reutilización lo cual impide duplicar y no por ello deja de resaltar las diferencias.

Resistencia al cambio.- Los cambios en los requisitos afectan a la funcionalidad del sistema, pero afectan en menor proporción cuando se trata de objetos.

Reutilización.- Aparte de la reutilización interna, basada en la expresión explícita de características comunes, el paradigma orientado a objetos desarrolla modelos mucho más próximos al mundo real, con lo que se aumentan las posibilidades de reutilización inclusive en otras aplicaciones.

1.1.2. Clases y Objetos.

1.1.2.1. Clases.

"Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica".[4]

[4] El lenguaje unificado de modelado, Grady Booch y otros, pág 42

Una clase es una abstracción de un conjunto infinito de objetos individuales. Cada uno de esos objetos es una instancia o ejemplar de dicha clase. Cada instancia de una clase tiene valores propios para sus atributos, pero comparte el nombre de estos atributos y las operaciones con el resto de instancias de su clase.

Cada clase debe tener un nombre que puede ser texto corto o expresión nominal inherente al sistema que se está modelando. Gráficamente una clase se representa por medio de un rectángulo.

1.1.2.2. Objetos.

Objeto se define como una cosa tangible o visible del mundo real que posee determinados atributos e identidad, diferenciándose así de los demás objetos.

Un objeto representa un item, una unidad o una entidad individual e identificable, sea real o abstracta con un papel bien definido en el dominio del problema.

Un objeto tiene estado, comportamiento e identidad, la estructura y el comportamiento de los objetos similares son definidos en sus clases.

Los términos instancia y objeto son sinónimos.

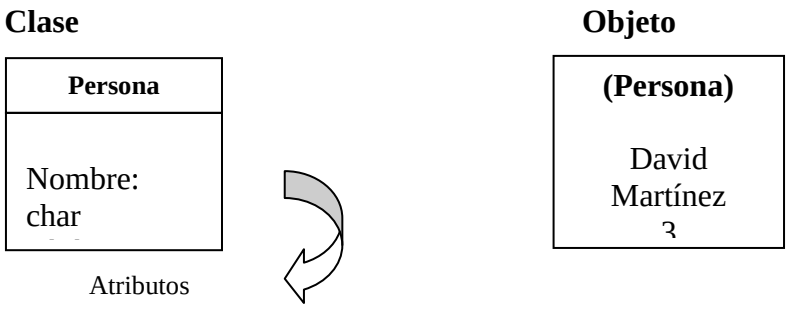
El estado de un objeto incluye todas las propiedades estáticas del objeto, más las propiedades dinámicas o valores actuales de cada una de esas propiedades.

El comportamiento es la forma como un objeto actúa y reacciona, en términos de sus cambios de estado y de paso de mensajes. El comportamiento de un objeto representa su actividad la que puede ser realizada por un objeto sobre otro o sobre él mismo.

Los objetos son entidades discretas y distiguibles que pueden ser concretas o abstractas y pueden tener su propia identidad. Dicho de otra forma, dos objetos son distintos incluso en el caso de que los valores de todos sus atributos coincidan.

1.1.2.3. Atributos

"Un atributo es una abstracción de un tipo de dato o estado que puede incluir un objeto de la clase."[5]



[5] Lenguaje Unificado de Modelado, Grady Booch y otros, pág 43.

1.1.2.2.1. Características de los objetos.

1.1.2.2.1.1. Identidad.

Significa que los datos están cuantificados en entidades discretas y distinguibles denominadas objetos. Los objetos pueden ser cosas concretas, conceptuales. Dos objetos son distintos a pesar de que los valores de todos sus atributos sean idénticos. Ejm: Un auto.

1.1.2.2.1.2. Clasificación.

Quiere decir que objetos con una misma estructura de datos (atributos) y comportamiento (operaciones) se juntan para formar una clase.

Dicho en otras palabras la clasificación permite agrupar objetos con la misma estructura de datos o atributos y el mismo comportamiento u operaciones en clases. Los valores de los atributos podrán ser distintos para cada uno de las clases mas todas comparten atributos y comportamiento.

Una clase es una abstracción que describe cosas importantes para una aplicación ignorando el resto.

Objetos auto **Clase auto**

Atributos

se abstrae

Capacidad
ocupantes
color
velocidad

Operaciones

Prender
Cambio de marcha

Cada objeto es una instancia de su clase.

Instancia

Atributos

Capacidad	=	94 kg
# ocupantes	=	5 personas
color	=	rojo
velocidades	=	5 velocidades

Toda instancia de la clase posee su propio valor para cada uno de los atributos pero comparte los nombres de los atributos y operaciones con las demás instancias de la clase.

Dominio.- Conjunto de valores permitidos.

1.1.2.2.1.3. Concurrencia.

Significa que en el dominio de la aplicación, los objetos pueden actuar en forma independiente y al mismo tiempo.

La concurrencia de objetos reales es un hecho que existe y su representación en software es realizada de forma adecuada usando una técnica de orientación a objetos.

1.1.2.2.1.4. Persistencia.

Significa que un objeto creado y operado durante el tiempo de ejecución de un programa, puede continuar existiendo inclusive después de que el programa que lo creó deja de operar.

La persistencia de un objeto en OO está relacionada con aplicaciones de bases de datos, donde los objetos creados son almacenados en las bases durante largos períodos de tiempo.

1.1.3. Abstracción.

Abstracción en Orientación a Objetos consiste en la concentración de los aspectos esenciales, propios de una entidad, ignorando sus propiedades accidentales, en base a un objetivo prefijado.

Significa concentrarse en lo que un objeto es y hace antes de decidir como el debe ser implementado.

El uso de la abstracción durante el análisis, significa lidiar apenas con el concepto del dominio de la aplicación y no tener que tomar decisiones sobre el diseño y la implementación de manera prematura, antes de ser comprendido el sistema.

El propósito de la abstracción es limitar el universo para poder entenderlo, porque no es posible que la abstracción permita describir una realidad en forma completa y absoluta.

1.1.4. Herencia.

Heredar significa compartir los atributos y las operaciones entre las clases en base a un relacionamiento jerárquico entre varias clases, es decir compartir atributos y operaciones de una clase padre (superclase) a una clase hijo (subclase).

Una clase puede ser definida en forma amplia y después ser refinada en varias subclases.

Cada subclase se incorpora o hereda todas las propiedades de la superclase e incrementa sus propias y exclusivas características.

Las propiedades de la superclase no necesitan ser repetidas en cada subclase.

1.1.5. Polimorfismo.

Es la característica que permite en la OO tener operaciones con el mismo nombre asociadas a diferentes objetos (clases), pero actuando de forma diferente, en otras palabras el polimorfismo permite que una misma operación pueda llevarse a cabo en forma diferente en clases diferentes.

Una operación es una acción o transformación que realiza o padece un objeto. La implementación específica de una operación determinada en una clase se denomina método.

Una operación es una abstracción de un comportamiento similar (pero no idéntico) en diferentes clases de objetos. La semántica de la operación debe ser la misma para todas las clases.

Polimorfismo significa también que el remitente de un mensaje a otro objeto (al hacerse un paso de mensaje), no necesita conocer la instancia (objeto) de la clase receptora. El objeto receptor puede pertenecer a una clase arbitraria.

El receptor del estímulo es quien determina como un estímulo deberá ser interpretado, no el remitente. El remitente solamente necesita saber que otra instancia puede desempeñar un determinado comportamiento y no a que clase la instancia pertenece y, por lo tanto, que operación realmente actúa la acción.

El polimorfismo de operaciones traslada la responsabilidad de decidir que implementación hay que utilizar dependiendo del código que hace la llamada a la jerarquía de clases.

1.1.6. Encapsulamiento.

El encapsulamiento (ocultamiento) de la información permite aislar los aspectos externos de los objetos de los detalles internos de la implementación. La encapsulación evita que al hacer cambios éstos no afecten en forma masiva al resto de la aplicación. La encapsulación tiene la capacidad de combinar estructuras de datos más comportamiento en una sola entidad.

Significa también que los datos y el código que los manipula son definidos juntos y que los datos no pueden ser separados o accesados separadamente del código asociado.

Aquellas porciones de un programa que no son parte de una definición dada no pueden accesar a cualquier dato dentro de la definición. El dato se considera así, encapsulado junto con el código.

CAPITULO II

2. Análisis de la Metodología Object Modelling Technique OMT

La Técnica de Modelado de Objetos (OMT, Object Modelling Technique) es una metodología que permite construir un modelo del dominio de una aplicación.

2.1. Fases de metodología OMT

2.1.1. Conceptualización

Es importante considerar la fase inicial de conceptualización donde se hace un análisis preliminar de requisitos de sistema y se definen los casos de uso, que es una técnica descrita por Jacobson en 1987 e incorporadas a la mayor parte de metodologías de Análisis Orientadas a Objetos.

2.1.1.1. Análisis de requerimientos

El análisis de requerimientos pretende establecer en primera instancia los requerimientos del usuario respecto al sistema que se va a construir. El análisis de requerimientos da como resultado el Documento Especificaciones de requerimientos de Software.

2.1.1.2. Casos de uso

Los casos de uso es una forma de describir los requisitos iniciales del usuario.

Un caso de uso está formado por una serie de interacciones entre el sistema y un actor. Los casos de uso se explicarán con mayor detalle en el capítulo No. 3.

2.1.2. Análisis

El modelo de Análisis permite elaborar una abstracción de lo que debe hacer un Sistema y no de su forma de implementación.

Los objetos del modelo deben ser conceptos del dominio de la aplicación. El modelo de análisis no debe contener ninguna decisión de implementación.

Dentro de la metodología OMT, estrictamente podemos dividir la etapa de análisis en tres fases: modelo de objetos, modelo dinámico y modelo funcional. No todos ellos tienen la misma importancia, y el desarrollar cada uno de ellos depende del dominio del problema.

2.1.3. Diseño del sistema

En esta etapa se toma decisiones de alto nivel acerca de la arquitectura global, además se organiza al sistema en subsistemas que se acoplen al análisis como a la arquitectura escogida.

Los pasos son:

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

2.1.4. Diseño de objetos

Se construye el modelo de diseño en base al modelo de análisis y se añaden los detalles de implementación. El diseño de objetos se centra en las estructuras de datos y los algoritmos necesarios para implementar cada una de las clases.

Los pasos a seguir son:

- Obtener las operaciones para el modelo de objetos a partir de los otros modelos
- Diseñar los algoritmos para implantar las operaciones
- Optimizar las rutas de acceso a los datos por medio de asociaciones o cálculos eficientes que deben ser guardados.
- Implantar el control del software introduciendo el esquema seleccionado durante el diseño de sistemas.
- Ajustar la estructura de clases para incrementar la herencia.
- Diseñar la implantación de las asociaciones.
- Determinar la representación de los atributos de los objetos.
- Empaquetar las clases y las asociaciones en módulos.

2.1.5. Implementación

Se traduce en un lenguaje de programación las clases de objetos y las relaciones desarrolladas durante el diseño. Durante la implementación se debe tener en cuenta los principios de la Ingeniería de Software.

2.2. Modelos de OMT

La Técnica de Modelado de Objetos transforma la definición del problema del usuario en tres modelos:

- Modelo de Objetos
- Modelo Dinámico
- Modelo Funcional

Los tres modelos en conjunto crean el modelo lógico requerido por los Estándares de Ingeniería de Software.

2.2.1. Modelo de Objetos

El Modelo de Objetos muestra la estructura estática de los objetos del sistema, identidad, relaciones con otros objetos, atributos y operaciones en el mundo real. El resultado del modelo de objetos es un diagrama de objetos que es un grafo con nodos donde se representan las clases de objetos y arcos que representan las relaciones entre clases. El procedimiento para construirlo es el siguiente:

- Identificación de las clases de objetos
- Identificación de las asociaciones entre clases
- Se añaden atributos de los objetos y de enlaces
- Escribir un diccionario de datos que contenga descripciones de clases, atributos y asociaciones
- Uso de herencia para organizar y simplificar las clases de objetos
- Se comprueban las vías de acceso usando escenarios e iterando los pasos anteriores cuando sea conveniente.

Los **diagramas de objetos** proporcionan una notación gráfica para el modelado de objetos, clases y relaciones. Hay dos tipos de diagramas de objetos: El diagrama de clases (que es un esquema o patrón para descubrir muchas instancias de datos posibles y estos describen clases de objetos) y el diagrama de instancias (describe las relaciones entre objetos siendo útiles para describir escenarios).

Un **diagrama de clases** es un esquema, patrón o plantilla para describir muchas instancias de datos posibles.

Un **diagrama de instancias** se refiere a las relaciones entre cierto conjunto de objetos.

El modelo de objetos además de **objetos, clases y atributos** puede contener los siguientes elementos:

2.2.1.1. Operaciones y Métodos

"Una **operación** es la implementación de un servicio que puede ser requerida a cualquier objeto de la clase para que muestre un comportamiento."[6]

En otras palabras una operación es una función que puede ser aplicada por los objetos de una clase. Por ejm: Las operaciones de la clase Persona pueden ser: cambiar de trabajo o cambiar de dirección.

---

[6] El lenguaje unificado de modelado, Grady Booch y otros, pag 44.

Un **método** es la implementación de una operación para una clase. Por ejm. La clase archivo puede tener una operación llamada imprimir. Los métodos de imprimir pueden estar mediante código diferente imprimir imágenes, imprimir archivos ascii.

Durante el modelado es bueno distinguir las operaciones que tengan efectos colaterales de aquellas que calculan un valor funcional sin provocar la modificación de un objeto. Este tipo de operación se denomina consulta las cuales se pueden considerar como atributos derivados.

2.2.1.2. Notación para las clases objetos

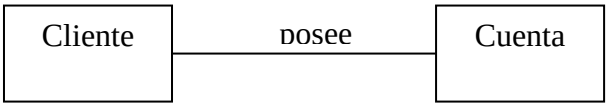
Una clase se representa mediante un cuadro que puede tener tres regiones: Nombre de la clase, Lista de atributos y lista de operaciones. El nombre de cada atributo puede ir seguido por detalles opcionales como lista de argumentos y tipo de resultado. Los atributos y operaciones pueden mostrarse o no, todo depende del grado de detalle.

Nombre de la clase
Nombre atributo 1 : Tipo dato 1=Valor omisión 1 Nombre atributo 1 : Tipo dato 1=Valor omisión 1
Nombre operación 1 (lista de argumentos 1) : tipo de resultado

2.2.1.3. Enlaces y Asociaciones

Los enlaces u asociaciones son mecanismos para establecer relaciones entre los objetos y las clases.

Enlaces.
"Un enlace es una instancia de una asociación."[7]
Un enlace es una conexión entre dos o más instancias de objetos.

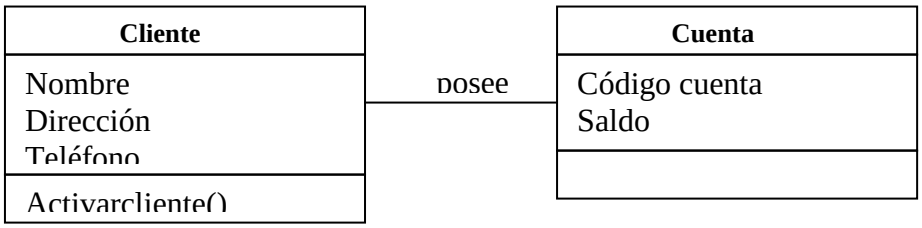


Asociaciones.

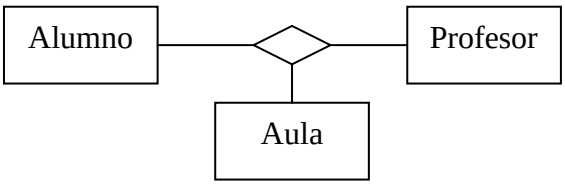
Una **asociación** describe un grupo de enlaces con una estructura común (todos ellos conectan instancias de objetos de las mismas clases) y una semántica común (todos los enlaces tienen el mismo significado). Una asociación describe un conjunto de enlaces

[7] Modelado y diseño orientado a objetos, James Rumbaugh, pág 53.

potenciales de la misma forma que una clase describe un conjunto de instancias potenciales.

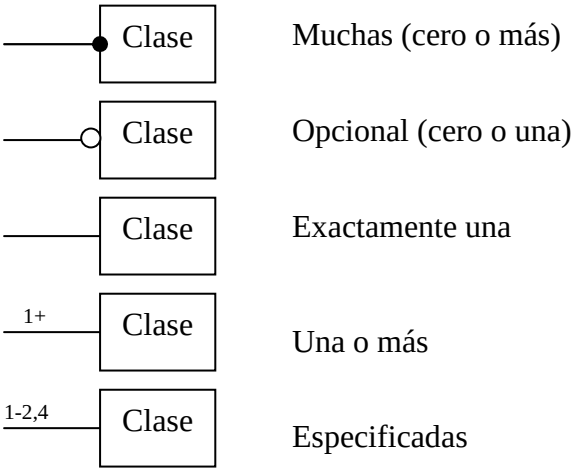


Tanto los enlaces como las asociaciones se representan mediante líneas etiquetadas con el nombre de la asociación. Las asociaciones pueden leerse en cualquier dirección.
Las asociaciones y los enlaces son los verbos que aparecen en la definición del problema.
Los enlaces y las asociaciones pueden conectar más de dos objetos y se representa mediante un rombo con conexiones a cada uno de los objetos.



La mayoría de las asociaciones conectan solamente dos clases de objetos por lo que se debe procurar usar asociaciones binarias porque son más sencillas de nombrar, entender e implementar.

2.2.1.3. Multiplicidad.



"La **multiplicidad** especifica el número de instancias de una clase que pueden estar relacionadas con una única instancia de una clase asociada."[8]
Para representar el número de instancias de cada clase que pueden participar en una asociación utilizaremos la siguiente notación en cada extremo de la asociación:

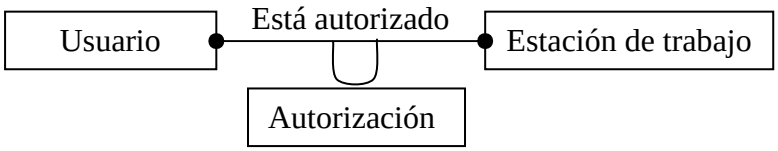
Círculo negro	•	Denota "muchos"	cero o más
Círculo blanco	o	Indica "opcional"	cero o uno
Línea sin símbolos			uno a uno

2.2.1.4. Atributos de los enlaces

"Un **atributo de enlace** es una propiedad de los enlaces de una asociación".[9]

La notación OMT para un atributo de enlace es un cuadro ligado a la asociación mediante un lazo donde pueden aparecer uno o más atributos de enlace en la segunda región del cuadro.

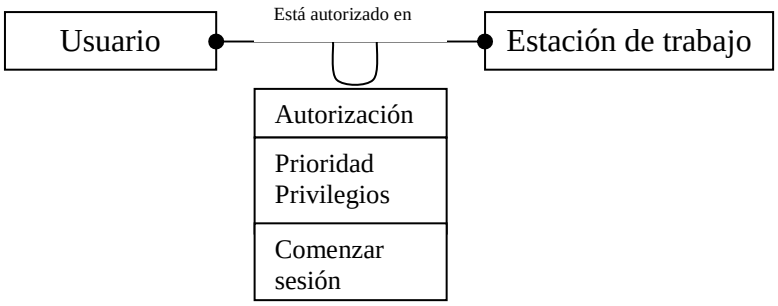
En la figura de abajo Autorización es un atributo de Está autorizado en.



En asociaciones muchos a muchos, los atributos de enlace no se pueden asociar a ninguno de los dos objetos individualmente sin perder información.

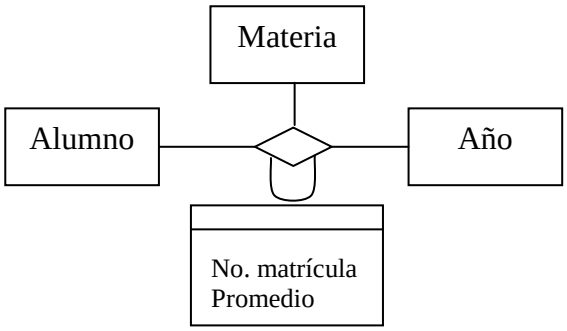
Diego Vega	Administrador	WS#1
José Hinojosa	Staff	WS#1
Diego Vega	Staff	WS#8

En ciertas ocasiones resulta útil modelar las asociaciones como clases. Cada enlace pasa a ser una instancia de la clase.



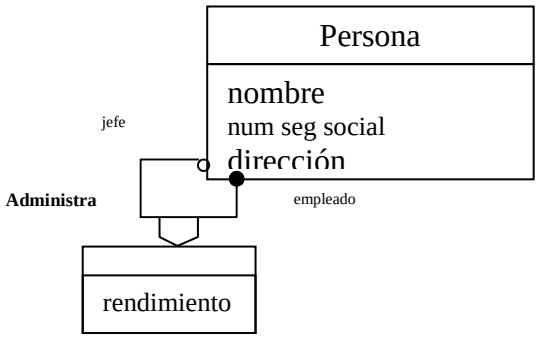
[8] Modelado y diseño orientado a objetos, James Rumbaugh, pág 56.
[9] Modelado y diseño orientado a objetos, James Rumbaugh, pág 59

Atributos de enlace para una asociación ternaria



2.2.1.5. Roles

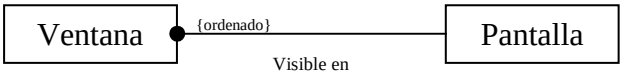
"Un **rol** es un extremo de una asociación"[10]. Los roles permiten visualizar una asociación desde un objeto hasta un conjunto de objetos asociados. Un rol es necesario para asociaciones entre objetos de la misma clase. Los nombres de roles deben ser únicos



En el ejemplo una los roles jefe y empleado distinguen a dos empleados que tienen la asociación administra.

2.2.1.6. Clasificación

La clasificación es una parte inherente de la asociación para indicar que los objetos están ordenados.

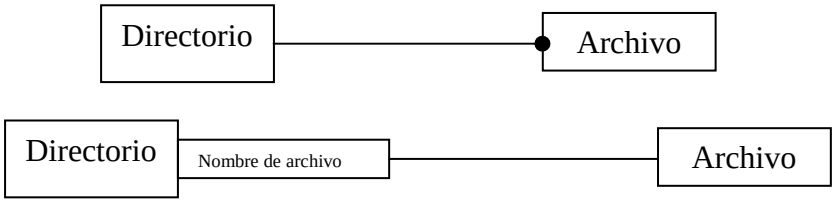


2.2.1.7. Cualificación

[10] Modelado y diseño orientado a objetos, James Rumbaugh, pág 62

Es una asociación que relaciona dos clases de objetos y un cualificador (atributo que reduce la multiplicidad). La cualificación puede darse en asociaciones muchos a muchos o uno a muchos, transformando la asociación a una de tipo uno a uno.

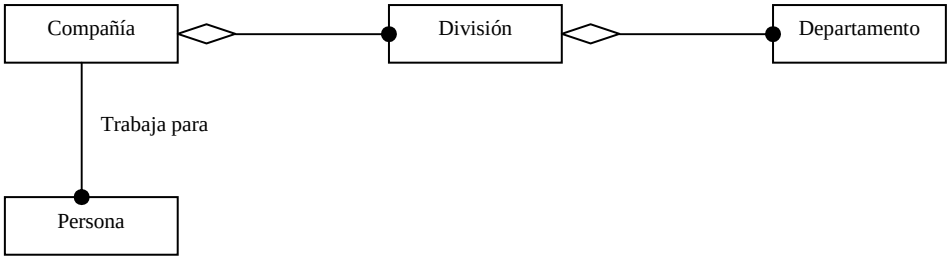
Asociación original: Un directorio tiene muchos archivos
Asociación cualificada: Un directorio con un nombre de archivo se combinan para identificar un archivo.



2.2.1.8. Agregación

"La agregación es un tipo especial de asociación"[11], en la cual una clase representa el "todo", que consta de otra clase con elementos más pequeños "las partes".

La agregación se dibuja con un rombo en el extremo del "todo".



2.2.1.9. Generalización y herencia.

"La generalización es la relación entre una clase y una o más versiones refinadas de esa misma clase. La que esta refinada se denomina superclase y cada versión refinada se denomina subclase".[12]

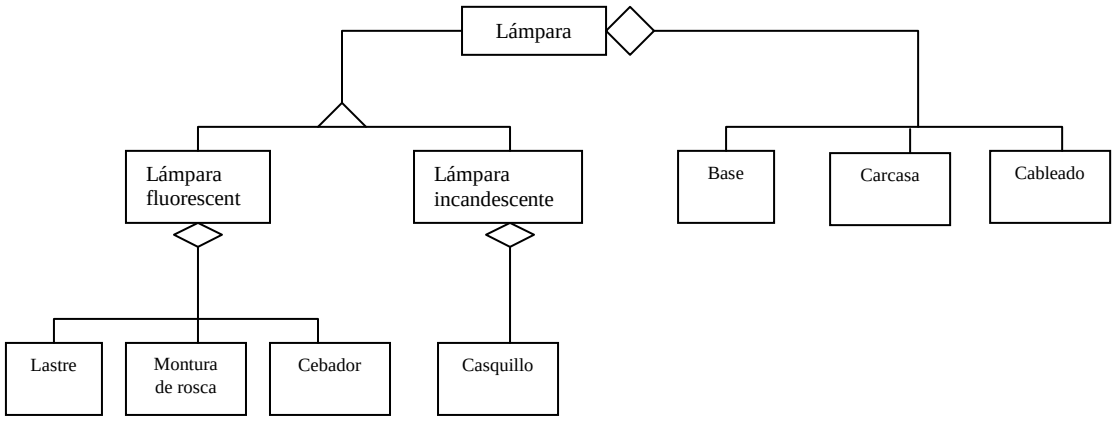
Las subclases heredan las características de sus antecesoras y además pueden tener sus propios atributos y operaciones.

[11] El lenguaje unificado de modelado, Grady Booch y otros, pág 58
[12] Modelado y diseño orientado a objetos, James Rumbaugh, pág 68

La generalización se representa mediante un triángulo que conecta la superclase con sus subclases. En cambio la superclase se conecta con una línea que va hacia el vértice del triángulo.

La generalización ayuda en el modelado estructurando las clases y ayuda a identificar lo que es similar y distinto respecto a las clases.

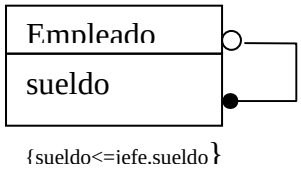
La especialización consiste en que las subclases refinan o especializan a la superclase.



2.2.1.10. Restricciones

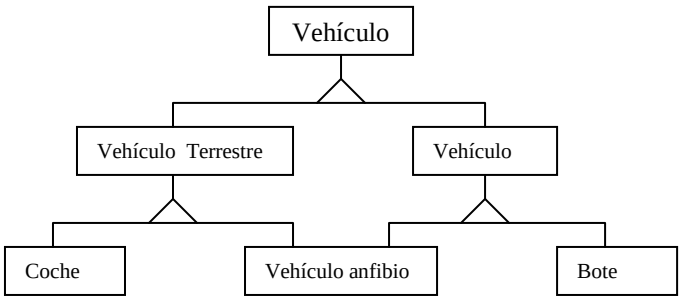
Las restricciones son relaciones funcionales entre entidades de un modelo de objetos. Una entidad incluye a los objetos, clases, atributos, enlaces y asociaciones. Las restricciones restringe los valores que puede tomar una entidad.

Las restricciones se grafican declarándola dentro de dos llaves.



2.2.1.11. Herencia múltiple.

La herencia múltiple permite a una clase tener más características de una superclase, y así heredar las características de todos sus padres. Esto complica a las jerarquías de herencia, que dejan de ser árboles para convertirse en grafos. La ventaja de la herencia múltiple es el incremento en las posibilidades de reutilización y su inconveniente es la pérdida de la simplicidad conceptual y de implementación.



2.2.1.12. Metadatos

Son datos que describen a otros datos. Las clases de objetos son metadatos por cuanto describen objetos. La realización de instanciación enlaza los objetos descriptores de clases con sus instancias. Los metadatos son conceptos útiles por dos razones: Aparecen en el mundo real y son una potente herramienta para implementar sistemas complejos.

2.2.2. Modelo dinámico

El modelo dinámico muestra el comportamiento del sistema en el tiempo y la secuencia de operaciones. Este modelo se utiliza para determinar o implementar los aspectos temporales, de comportamiento y de control del sistema. El modelo dinámico contiene diagramas de estados que es un grafo donde los nodos son los estados y los arcos son transiciones entre estados causados por eventos o sucesos. El procedimiento para construirlo es el siguiente:

- Identificar la secuencia de eventos dentro del ámbito del problema y documentarlo en el seguimiento de eventos.
- Construir un diagrama de transición de estados para cada objeto que sea afectado por los eventos, mostrando los mensajes que fluyen, las acciones que son realizadas y los cambios en el estado de los objetos que tienen lugar cuando ocurren los eventos.
- Verificar que los eventos compartidos entre diagramas de estado sean consistentes y correctos.

2.2.2.1. Diagramas de estados

El modelo dinámico de OMT es el correspondiente al modelo de comportamiento de las técnicas de análisis estructurado. En ambos casos el modelo se representa mediante diagramas de estado. En el caso de OMT, estos diagramas se utilizan para modelar el comportamiento común a todas las instancias de una clase.

Las instancias de una clase pueden encontrarse en estados distintos.

2.2.2.2. Estados y eventos(sucesos)

El estado de un objeto son los valores de los atributos y los enlaces que mantiene un objeto en un momento determinado. Los objetos interactúan unos con otros y como consecuencia de esas interacciones cambian de estado.
Los estados que figuran en los diagramas de estados son abstracciones de los valores de los atributos y enlaces de un objeto.

Al definir los estados, hay que ignorar los atributos que no afectan al comportamiento del objeto y agrupar todas las combinaciones de valores de atributos y enlaces en las que el objeto presente el mismo comportamiento.

La interacción entre objetos se realiza mediante eventos. Como consecuencia de un evento, un objeto puede realizar una serie de acciones y/o enviar eventos a otros objetos y/o cambiar de estado. Los eventos son el método de intercambiar información entre los objetos.

La relación o secuencia de estados, eventos, transacciones y acciones se representan en los diagramas de estados. El modelo dinámico es un conjunto de diagramas de estados, uno para cada clase de objetos que tengan un comportamiento no trivial. La relación entre los distintos diagramas de estados se establece mediante los eventos que emiten unos objetos y consumen otros.

2.2.2.3. Escenarios.

Un escenario es un seguimiento de eventos ocurridos durante la ejecución de un sistema. Por ejemplo, el escenario para una llamada telefónica que sólo afectan a la línea telefónica es el siguiente[13]:

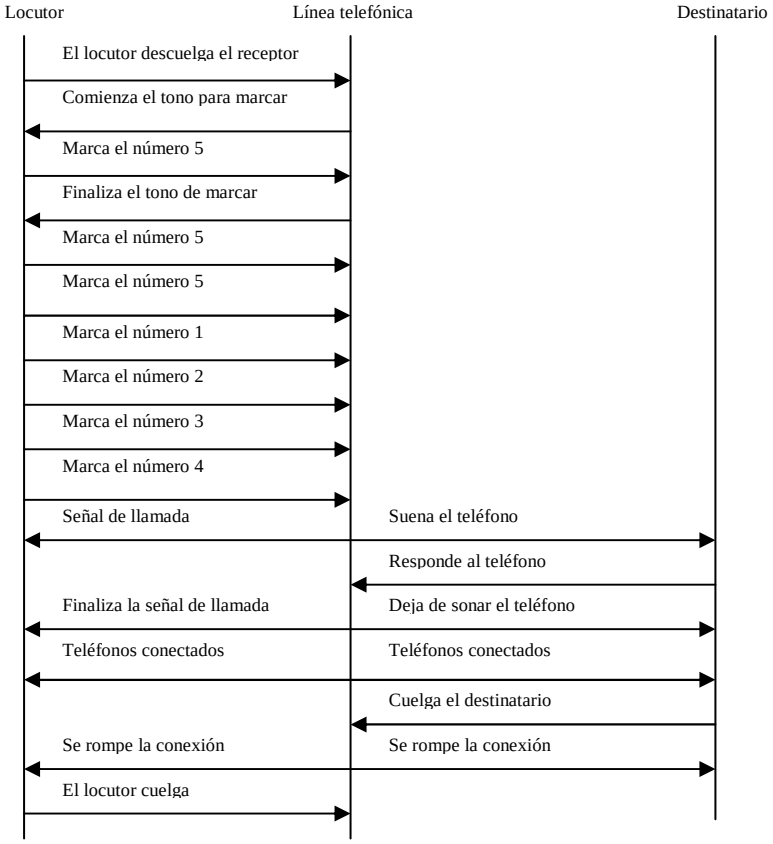
El locutor descuelga el receptor
Comienza el tono para marcar
El locutor marca el número 5
Finaliza el tono de marcar
El locutor marca el número 5
El locutor marca el número 5
El locutor marca el número 1
El locutor marca el número 2
El locutor marca el número 3
El locutor marca el número 4
El teléfono llamado comienza a sonar
Aparece la señal de llamada en el teléfono del locutor
El destinatario responde
El teléfono del destinatario deja de sonar
El tono de llamada desaparece en el teléfono del locutor
Los teléfonos se conectan
El destinatario cuelga

[13] Modelado y diseño orientado a Objetos, Rumbaugh y otros, pág 126

Los teléfonos se desconectan
El locutor cuelga

2.2.2.4. Trazas de eventos (sucesos)

La identificación de los objetos emisores y receptores para el ejemplo anterior son:[14]



En el ejemplo una vez que el receptor descuelga y antes de que se marque el primer número, la línea telefónica se encuentra en el estado Señal de marcar.

Una traza de eventos es un diagrama que muestra la secuencia de envíos y recepciones de eventos entre varios objetos. Sirven para mostrar cómo un sistema ha llegado a un escenario determinado, y para construir el modelo dinámico del sistema debido a que muestran una secuencia lógica de eventos que se puede utilizar para realizar los diagramas de estados de cada clase.

A partir de varias traza de eventos se puede construir el diagrama de estados de una clase.

2.2.2.5. Actividades y Acciones

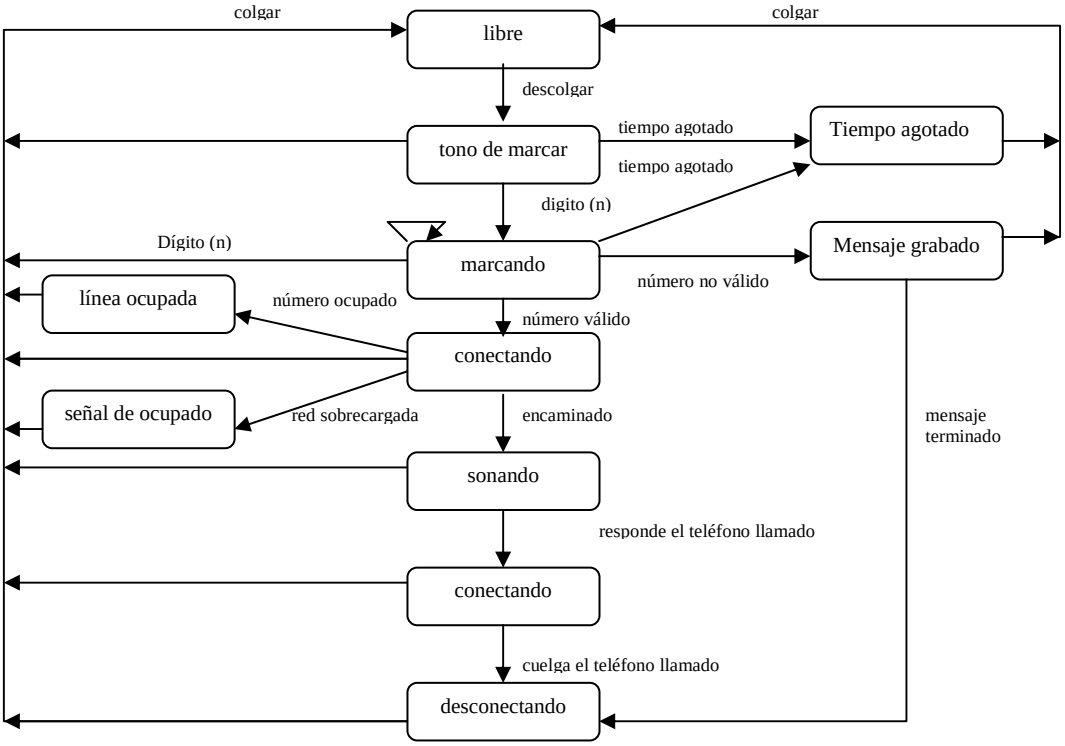
Son las respuestas de los objetos a los eventos producidos en el exterior o guardas que se producen internamente.

[14] Modelado y diseño orientado a Objetos, Rumbaugh y otros, pág 127

Guarda es una condición que se define sobre los atributos propios de un objeto. Al recibir un evento la transacción etiquetada con dicho evento se dispara si y solo si se satisface la guarda.

Cuando un objeto entra en un estado ejecuta la actividad correspondiente hasta que ésta finalice o hasta que el disparo de una transición permita abandonar ese estado. Por esto una actividad necesita un tiempo para completarse.

Una acción es una operación instantánea que idealmente no necesita tiempo para realizarse y se asocia al disparo de una determinada transición. Las acciones consisten en asignar valores a los atributos de un objeto o en generar eventos para que sean tratados por otro proceso. El diagrama de estados para una línea telefónica es el siguiente: [14]



2.2.2.5. Transición

Son los arcos entre los estados, es decir son las flechas desde el estado receptor hasta el estado destino, el rótulo de la flecha es el evento o suceso.

2.2.3. Modelo funcional

El modelo funcional describe los aspectos de un sistema que afectan a su transformación de valores, funciones, restricciones y dependencias funcionales. El

[14] Modelado y diseño orientado a Objetos, Rumbaugh y otros, pág 131

modelo funcional incluye qué es lo que hace un sistema, sin mencionar cómo o cuando es hecho.

El modelo funcional es representado con diagramas de flujo de datos. Los diagramas de flujo de datos muestran las dependencias entre valores y la obtención de valores de salida a partir de valores de entrada y funciones.

Las funciones son invocadas como acciones en el modelo funcional y son mostradas como operaciones sobre los objetos en el modelo de objetos.

El modelo funcional contiene diagramas de flujo de datos que es un grafo cuyos nodos son los procesos y cuyos arcos son flujos de datos.

El procedimiento para construirlo es el siguiente:

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

El modelo funcional se basa en el uso de Diagrama de Flujo de datos que distingue a OMT de otras metodologías, sin embargo su uso en recientes escritos de Rumbaugh es descartado y en su lugar incluye los casos de uso, diagramas de interacción de objetos y especificaciones de operaciones.

Modelo operacional de Coleman

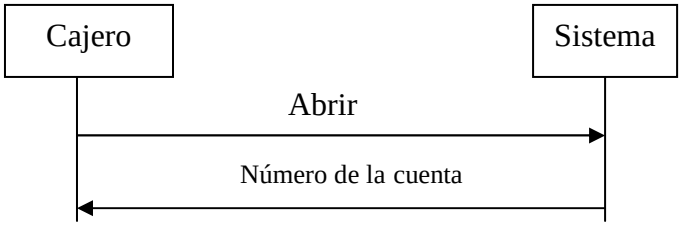
Permite especificar las operaciones del sistema declarativamente, definiendo sus efectos en términos de los cambios de estado y de los eventos de salida.

Las operaciones del sistema pueden ser:

Crear una nueva instancia de clase
Modificar el valor de un atributo de un objeto existente
Añadir o borrar una tupla de objetos de un relacionamiento
Enviar un evento a un agente
Usar precondiciones y poscondiciones. Las precondiciones son las condiciones sobre las cuales una operación puede ser invocada. Las poscondiciones describen la forma como el estado del sistema es modificado por una operación y cuales eventos son enviados a los agentes.

Ejm: Escenario para apertura de cuentas.

El cliente solicita al sistema la apertura de una cuenta, el sistema crea una nueva cuenta, genera un número único para la cuenta, inicializa el saldo con cero, y envía el número de cuenta al cliente.



Sintaxis:

Operación: Abrir cuenta
Descripción: Abre una cuenta nueva para un cliente.
Lee: nombre:nombre del cliente
Modifica: Banco
Envia: cuenta cliente: número de cuenta
Asume:
Resulta: Una cuenta es añadida al banco, saldo inicializado en cero, nombre inicializado con nombre, número de la cuenta fijado con un valor. El número de la cuenta es enviado al cliente.

CAPITULO III

3. Análisis del Lenguaje de Análisis y Diseño Unified Modelling Languaje UML

"UML es un lenguaje gráfico que sirve para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software"[15]. Cubre aspectos conceptuales como los procesos y funciones del sistema así como cosas concretas como clases, esquemas de bases de datos y componentes reutilizables.

3.1. Breve historia de UML

Los lenguajes de modelamiento orientados a objetos aparecieron entre mediados de los 70's y fines de los 80's, debido al surgimiento de lenguajes orientados a objetos, y aplicaciones cada vez más complejas.

Entre los métodos orientados a objetos se destacaron tres: el de Booch, el OOSE (Object-Oriented Software Engineering) de Jacobson, y el OMT (Object Modeling Technique) de Rumbaugh. Otros métodos importantes fueron: Fusion, Shlaer-Mellor y Coad-Yourdon.

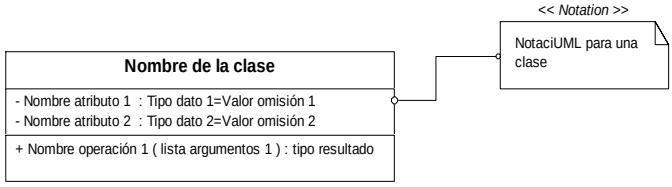
A mediados de los 90's cada autor de los tres métodos más reconocidos (Booch, Jacobson y Rumbaugh) empieza a reunir sus ideas. En 1994 Rumbaugh se une a Booch, y sacan la versión 0.8 de Unified Method, el cual es publicado en 1995. Luego Jacobson se une a Rational, y sacan la versión 0.9 de UML, en junio de 1996. Posteriormente los tres sacan la versión 1.0 en enero de 1997, con la aprobación de un grupo de empresas llamado OMG (Object Management Group). Este lenguaje y sus posteriores versiones se han adoptado como un estándar en la industria del desarrollo de software.

3.2. Elementos de UML

UML tiene cuatro tipos de elementos:

Elementos estructurales.- Son las partes estáticas de un modelo y representan las cosas materiales o conceptuales.
Elementos de comportamiento.- Son las partes dinámicas de los modelos de UML, son los verbos de un modelo y representan el comportamiento en tiempo y espacio.
Elementos de agrupación.- Son las partes organizativas de los modelos UML.
Elementos de anotación.- Son las partes explicativas de los modelos UML.

3.2.1. Elementos estructurales

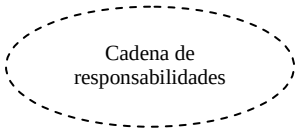


[15] El lenguaje Unificado de Modelado, Booch y otros, pág 11

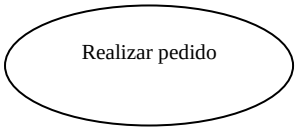
Clase.- Conjunto de operaciones que comparten los mismos atributos, operaciones, relaciones y semántica.

Interfaz.- Es una colección de operaciones que especifican un servicio de una clase o componente. Una interfaz define un conjunto de especificaciones de operaciones, pero no un conjunto de implementaciones de operaciones.

Colaboración.- Define a una interacción. Una colaboración es una sociedad de clases, interfases, roles y otros elementos que colaboran para proporcionar un comportamiento cooperativo mayor que la suma de los comportamientos de sus elementos.



Caso de uso.- Es una descripción de un conjunto de secuencias de acciones los cuales son usados para estructurar aspectos de comportamiento en un modelo.



Clase activa.- Es una clase cuyos objetos tienen uno o más procesos o hilos de ejecución las cuales pueden dar origen a actividades de control.

Componente.- Es una parte física y reemplazable de un sistema. Puede contener elementos lógicos como clases, interfases y colaboraciones.

Nodo.- Es un elemento físico que aparece en tiempo de ejecución y representa un recurso computacional que tiene memoria y capacidad de procesamiento.

3.2.2. Elementos de comportamiento

Interacción.- Es un comportamiento que comprende un conjunto de mensajes intercambiados entre un conjunto de objetos. Una interacción permite especificar el comportamiento de objetos u operaciones.

Máquina de estados.- Es un comportamiento que especifica las secuencias de estados por las que pasa un objeto.

3.2.3. Elementos de agrupación

Paquete.- Es un mecanismo de propósito general que permite organizar elementos en grupos, aquí se pueden incluir elementos estructurales, elementos de comportamiento. Un paquete solamente existe en tiempo de desarrollo.

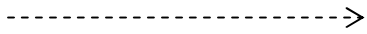
3.2.4. Elementos de anotación

Notas.- Son comentarios donde se pueden aplicar para describir, clarificar, y hacer observaciones sobre cualquier elemento de un modelo.

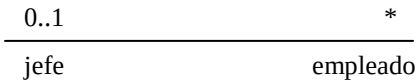
3.3. Relaciones en UML

Hay cuatro tipos de relaciones en UML:

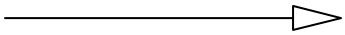
Dependencia.- Es una relación semántica entre dos elementos, en la cual un cambio de un elemento independiente puede afectar a la semántica del elemento dependiente.



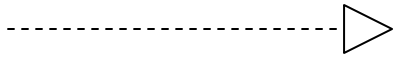
Asociación.- Es una relación estructural que describe un conjunto de enlaces, los cuales son conexiones entre objetos. La agregación es un tipo especial de asociación, que representa una relación estructural entre un todo y sus partes. Puede incluir multiplicidad y nombres de rol.



Generalización.- Es una relación de especialización /generalización en la cual los objetos del elemento especializado pueden sustituir a los objetos del elemento general.



Realización.- Es una relación semántica entre clasificadores, en donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá.



3.4. Arquitectura

"La arquitectura tiene que ver con la estructura y el comportamiento, el uso, la funcionalidad, el rendimiento, la capacidad de adaptación, la reutilización, la capacidad de ser comprendido, las restricciones económicas y tecnológicas y aspectos estéticos."[16]

La arquitectura de un sistema puede describirse de mejor manera por medio de cinco vistas interrelacionadas:

[16] El lenguaje Unificado de Modelado, Booch y otros, pág 26

La vista de casos de uso.- Comprende los casos de uso que describe el comportamiento del sistema percibido por los usuarios finales, analistas y encargados de las pruebas. Con UML, los aspectos estáticos de esta vista se capturan en los diagramas de casos de uso.

La vista de diseño.- Comprende las clases, interfases y colaboraciones que forman el vocabulario del problema y su solución. Con UML, los aspectos estáticos de esta vista se capturan en los diagramas de clases y de objetos.

La vista de procesos.- Son los hilos y procesos que forman los mecanismos de sincronización y concurrencia del sistema. Esta vista cubre el funcionamiento, capacidad de crecimiento y rendimiento del sistema y se hace énfasis en las clases activas. Con UML, los aspectos estáticos de esta vista se capturan en los diagramas de clases y de objetos.

La vista de implementación.- Son los componentes y archivos que utilizan para ensamblar y hacer disponible el sistema físico. Con UML, los aspectos estáticos se capturan con los diagramas de componentes.

La vista de despliegue.- Son los nodos que forman la topología de hardware sobre la que se ejecuta el UML, los aspectos estáticos se capturan en los diagramas de despliegue.

En las vistas anteriores, con UML, los aspectos dinámicos se capturan con los diagramas de interacción, de estados y de actividades.

3.5. Términos y conceptos

En UML además de conceptos tratados anteriormente como Clases, Atributos, Operaciones, Relaciones de Dependencia, Generalizaciones y Asociaciones (Rol, Multiplicidad y Agregación) y restricciones, hay otros que complementan los conceptos básicos que maneja UML:

Responsabilidades.- Son un contrato o una obligación de una clase. Una clase bien estructurada tiene al menos una responsabilidad y a lo mucho unas pocas.

Una responsabilidad se dibuja al final de los compartimentos de la clase.

Interfases.- Se usan cuando se quieren separar la implementación de una clase de su especificación.

Notas.- Son símbolos gráficos para representar restricciones o comentarios asociados a un elemento a un conjunto de elementos.

Estereotipos.- Es una extensión del vocabulario de UML que permite crear nuevos tipos de bloques de construcción(clases, interfases, colaboraciones, componentes y otros) parecidos a los existentes pero ceñidos al problema que se está modelando.

Valor etiquetado.- Permiten añadir más información a la especificación de un elemento. Uno de los usos más comunes es para especificar propiedades relevantes a la generación de código.

3.6. Diagramas UML

"Un diagrama es la representación de los elementos que constituyen un sistema el que se grafica como un grafo conexo de nodos (elementos) y arcos (relaciones)."[17] UML incluye nueve diagramas, cuatro diagramas estructurales y cinco de comportamiento:

3.6.1. Diagramas Estructurales

3.6.1.1. Diagrama de clases(vista estática).

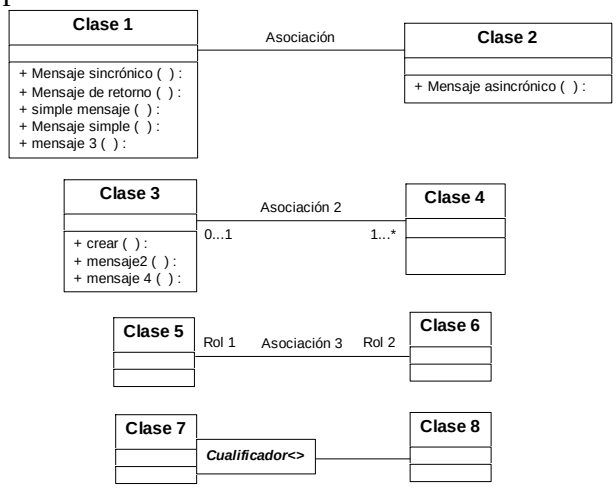
Muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Gráficamente un diagrama de clases se representa por medio de nodos (clases) y arcos (relaciones).

Elementos

- Clases
- Interfases
- Colaboraciones
- Relaciones de dependencia, generalización y asociación.
- Notas y restricciones.

Los diagramas de clases se utilizan para modelar:

El vocabulario de un sistema.- Determinar qué abstracciones son parte del sistema.
Colaboraciones simples.- Las cuales se logran identificando una función o comportamiento resultante de una interacción de clases.
Esquema lógico de la base de datos.- Debido a que los sistemas modelados tienen objetos persistentes es decir tiene objetos que pueden ser almacenados en una base de datos para ser recuperados cuando sea necesario.



[17] EL lenguaje Unificado de Modelado, Booch y otros, pág 81

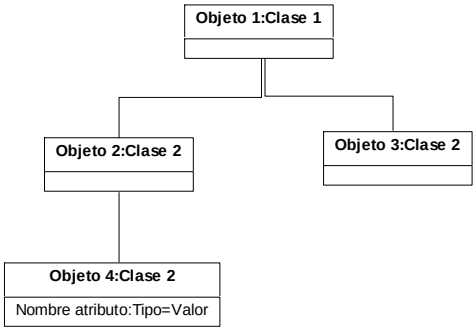
3.6.1.2. Diagrama de objetos(vista estática).

Muestra el conjunto de objetos (instancias) y sus relaciones. Se utilizan para describir estructuras de datos. El diagrama de objetos es un modelado de una instantánea del sistema en un momento concreto.

Elementos
- Objetos
- Enlaces
- Notas y restricciones

Los diagramas de objetos se usan para modelar:

Estructuras de Objetos.- Es tomar una instancia de un diagrama de clases en un momento determinado.



3.6.1.3. Diagrama de componentes(vista estática).

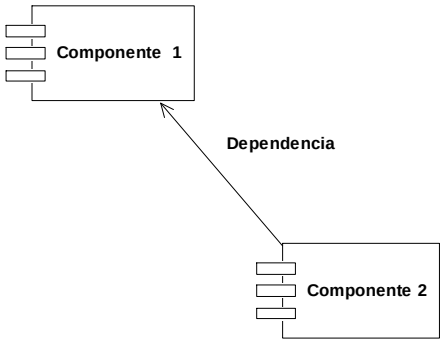
Muestra la organización y relaciones entre un conjunto de componentes. Los diagramas de componentes se relacionan con los diagramas de clases en que un componente se corresponde con una o más clases, interfases o colaboraciones.

Sirven para modelar las cosas físicas que residen en un nodo tales como ejecutables, bibliotecas, tablas, archivos y documentos.

Elementos
- Componentes
- Interfases
- Relaciones de dependencia, generalización, asociación y realización
- Notas y restricciones

Los diagramas de componentes sirven para modelar:

El código fuente
Las versiones ejecutables
Bases de datos físicas



3.6.1.4. Diagrama de despliegue(vista estática).

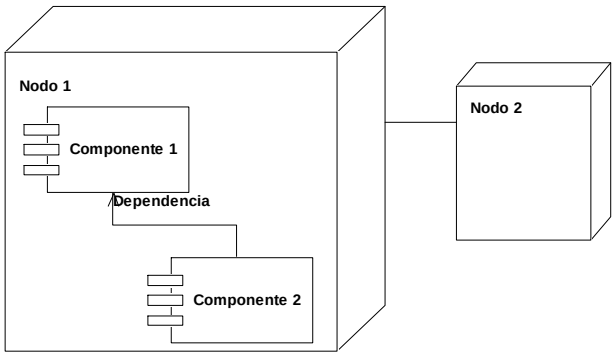
Muestra la configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

Elementos

- Nodos
- Relaciones de dependencia y asociación
- Notas y restricciones

Los diagramas de despliegue sirven para:

Modelar sistemas empotrados.- Los sistemas empotrados son hardware con gran cantidad de software como motores, pantallas accionados por sensores, cambios de temperatura.
Modelar sistemas cliente/servidor.- Son sistemas que tienen la interfaz del usuario (cliente) y los datos persistentes (servidor).
Modelar sistemas distribuidos.- Son los que incluyen varios niveles de servidores.



3.6.2. Diagramas de comportamiento

3.6.2.1. Diagrama de casos de uso(vista dinámica).

Muestra un conjunto de secuencias de acciones, actores(un tipo especial de clases) y sus relaciones. Ayudan a validar la arquitectura y a verificar el sistema mientras evoluciona

a lo largo del desarrollo. Es decir ayudan a que los usuarios finales y los expertos del dominio se comuniquen con los desarrolladores.

Actor.- Representa a un rol que puede ser ejecutado por una persona, un dispositivo de hardware u otro sistema.

Caso de uso.- Especifica qué hace un sistema y no cómo lo hace, Es un documento narrativo que describe la secuencia de eventos cuando uno o más actores utilizan un sistema para llevar a cabo un proceso.

Este documento narrativo de alto nivel tiene el siguiente formato:

Caso de uso.- Nombre del caso de uso
Actores.- Lista de actores
Descripción.- Breve descripción del caso

Para un documento narrativo con mayor detalle es mejor describir una secuencia típica de eventos con el siguiente formato:

Acción de los actores	**Respuesta del sistema**
Acciones de los actores enumeradas	Descripción de las respuestas de sistema enumeradas

La enumeración de las acciones de los actores así como las respuestas del sistema siguen un orden secuencial según ocurren en el tiempo.

Relaciones de un caso de uso

Existen tres tipos de relaciones representadas por flechas que van del elemento dependiente al independiente.

Comunica (communicates).- Es la relación entre un actor y un caso de uso, refleja la participación del actor en el caso de uso.

Usa (uses).- Es la relación entre dos casos de uso, muestra la inclusión del comportamiento de un escenario en otro.

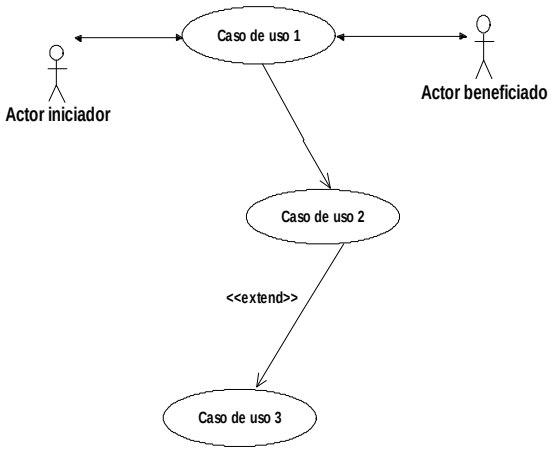
Extiende (extends).- Relación entre dos casos de uso, indica cuando un caso de uso es una especialización de otro.

Elementos

- Casos de uso
- Actores
- Relaciones de dependencia, generalización y asociación
- Notas y restricciones.

Los diagramas de casos de uso se usan para:

Modelar el contexto de un sistema.- Involucra dibujar una línea alrededor del sistema y determinar qué actores quedan fuera del sistema.
Modelar los requisitos de un sistema.- Se debe especificar lo que debe hacer el sistema.



3.6.2.2. Diagramas de interacción

"Constan de un conjunto de objetos y relaciones y los mensajes que fluyen entre ellos. Sirven para visualizar un sistema en ejecución a lo largo del tiempo"[18]. Hay dos tipos de diagramas de interacción: Los Diagramas de secuencia y los diagramas de colaboración.

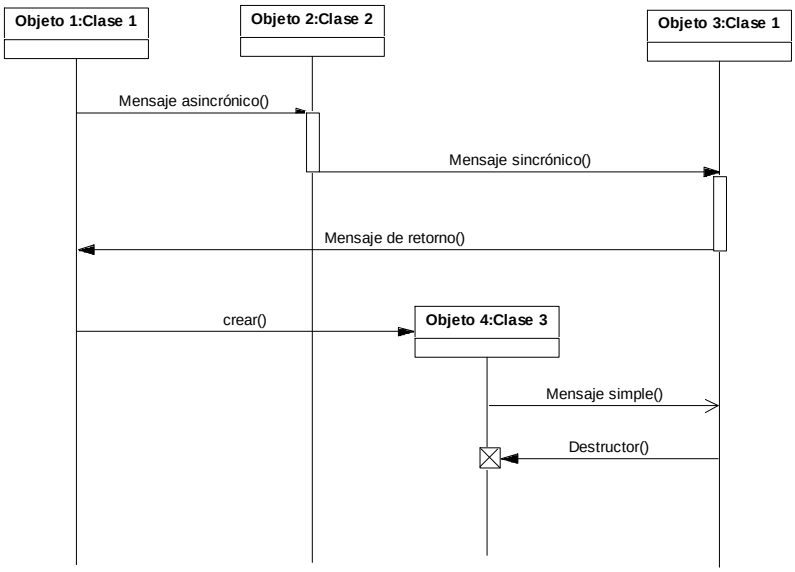
3.6.2.2.1. Diagrama de secuencia(vista dinámica).

Es un diagrama de interacción que consta de un conjunto de objetos y sus relaciones con los mensajes que pueden ser enviados entre ellos. El diagrama de secuencia resalta la ordenación de los mensajes. El diagrama de secuencia presenta un conjunto de objetos y los mensajes enviados y recibidos por ellos.

Elementos
- Objetos
- Enlaces
- Mensajes
- Notas y restricciones

[18] EL lenguaje Unificado de Modelado, Booch y otros, pág 213

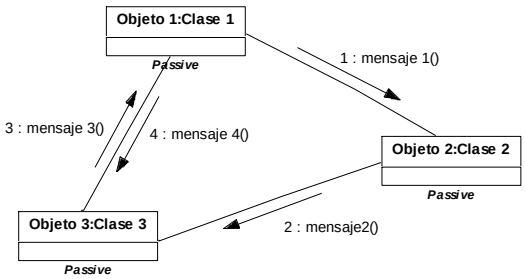
Se representa mediante una tabla de objetos dispuestos a lo largo del eje X y mensajes ordenados según suceden en el tiempo a lo largo del eje Y. Generalmente a la izquierda se coloca el objeto que inicia la interacción y a continuación los objetos subordinados.



Los diagramas de secuencia tienen dos características:

Línea de vida.- Es una línea vertical que representa la existencia de un objeto en un periodo de tiempo.
Foco de control.- Es un rectángulo delgado y estrecho que representa el tiempo durante el cual un objeto ejecuta una acción.

3.6.2.2.2. Diagrama de colaboración(vista dinámica).



Es un diagrama de interacción que hace énfasis en la organización estructural de los objetos que envían y reciben mensajes.

Elementos

- Objetos

- Enlaces
- Mensajes
- Notas y restricciones

El diagrama de colaboración es una colección de nodos (objetos) y arcos (enlaces). Los enlaces se adornan con mensajes que envían y reciben los objetos.

Los diagramas de colaboración tienen dos características:

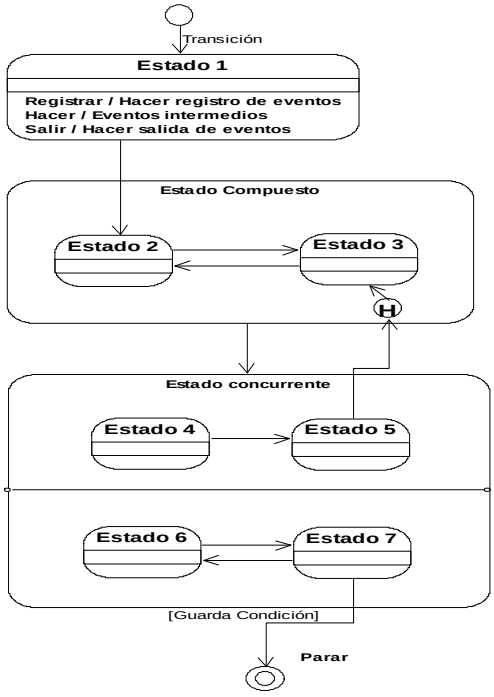
El camino.- Para indicar cómo se enlaza un objeto a otro.
El número de secuencia.- Para señalar la ordenación temporal de un mensaje.

3.6.2.2.2. Diagrama de estados(vista dinámica).

Un estado es una condición o situación en la vida de un objeto en respuesta a un evento.

Los diagramas de estados sirven para el modelado del comportamiento de una interfaz, una clase o una colaboración y están constituidos por estados, transiciones, eventos y actividades. Estos resaltan el comportamiento dirigido por eventos de un objeto.

Un evento es la especificación de un acontecimiento significativo que ocupa un lugar en el tiempo y el espacio.



Elementos

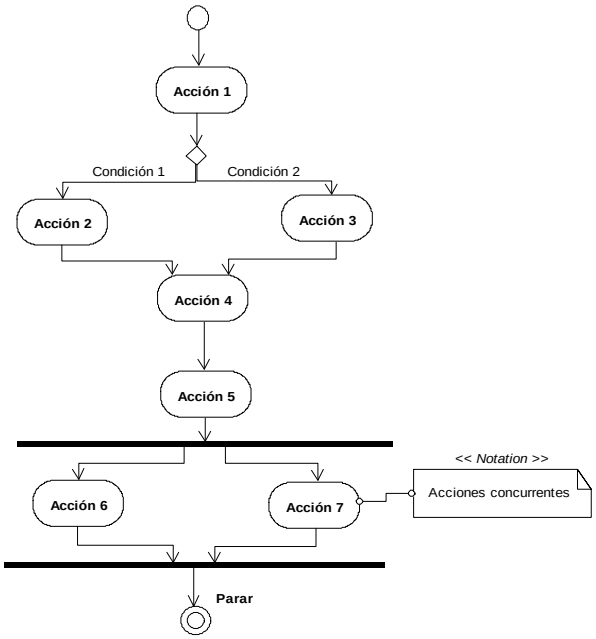
- Estados simples y compuestos
- Transiciones incluyendo eventos y acciones

Los diagramas de estados sirven para modelar objetos reactivos (objeto dirigido por eventos).

3.6.2.2.3. Diagrama de actividades(vista dinámica).

Es un caso especial del diagrama de estados, pues muestra el flujo de actividades dentro de un sistema.

Una actividad es una ejecución no atómica en curso. Modelan el funcionamiento (función) de un sistema y resaltan el flujo de control entre objetos.



Elementos

- Estados de actividad
- Estados de acción
- Transiciones
- Objetos

Los estados de actividad o de acción ejecutan una actividad o acción la cual al finalizar el control pasan a la siguiente actividad o acción.

Transición.- Es el flujo de control que hace que un estado de actividad o de acción pase a otro. Se representa mediante una línea.

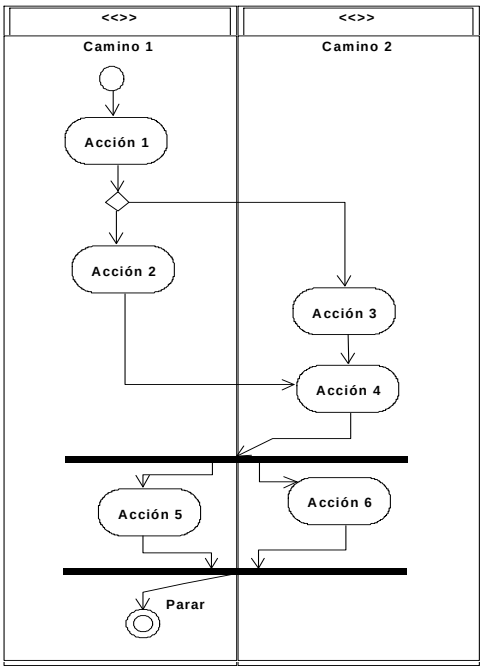
Bifurcaciones.- Una bifurcación puede tener una transición de entrada y dos o más de salida. En cada transición de salida se coloca una expresión booleana que se evalúa a la entrada de la bifurcación. Una bifurcación se representa mediante un rombo.

División y unión.- Para especificar la división y unión de flujos de control se utiliza la barra de sincronización.

Calles o Swimlanes.- Sirven para dividir los estados de actividad de un diagrama de actividades en grupos, donde cada uno representa la parte de la organización responsable de esas actividades.

Los diagramas de actividades se utilizan para:

Modelar un flujo de trabajo.- Son los procesos del negocio que tiene el sistema que se está desarrollando.
Modelar una operación.- Son los detalles de una computación. Se incluyen los parámetros de la operación así como sus objetos locales.



CAPITULO IV

4. Desarrollo del Sistema

4.1. Análisis

4.1.1. Especificación de Requerimientos de Software (SRS)

Sistema de Admisiones y Facturación para el centro de Idiomas de la Academia Cotopaxi (SISADMINFAC)

4.1.1.1. Introducción

Propósito

Este documento técnico recoge los requerimientos recopilados para desarrollar el Sistema de Admisiones y Facturación del Centro de Lenguaje de la Academia Cotopaxi (SISADMINFAC).

Aquí se determina las funcionalidades que debe tener el Sistema de Admisiones y Facturación para apoyar a la Administración de las admisiones, matriculas y facturación que se ejecutan en el Centro de Lenguaje y será el documento de soporte a las actividades de entrega recepción del sistema.

Tendrán acceso a este documento los usuarios del sistema en el Centro de Lenguaje, el Departamento Financiero y el Departamento de Tecnología.

Alcance

Este Sistema de Admisiones y Facturación para el centro de Lenguaje de la Academia Cotopaxi tendrá la siguiente funcionalidad:

Inscribir aspirantes es un proceso en el que se registra los datos de los aspirantes y se genera los rubros necesarios para el cobro de la inscripción.
Matricular alumnos consiste en asignar a un aspirante un número de identificación único, asignar curso y generar los rubros necesarios para el cobro de la matrícula.
Generar cobros permite registrar los abonos, descuentos o ajustes sobre los rubros generados.
Aprobar alumnos significa registrar las notas logradas en un curso, determinar el promedio e indicar si pasa de nivel.
Imprimir Facturas permite emitir las facturas de los abonos realizados.
Generar Reportes para Contabilidad.

Definiciones, Acrónimos y Abreviaturas

SISADMINFAC.- Sistema de Admisiones y Facturación del Centro de Idiomas de la Academia Cotopaxi.

SPYRAL.- Sistema de Contabilidad que utiliza la Academia para sus procesos Financieros.
IEEE.- Institute of Eelectrical and Electronics Engineers
SRS.- Especificación de requerimientos de software.

Referencias

Standard de Recomendación práctica IEEE para la especificación de requisitos de software (Std. IEEE 830-1993). Anexo1.
Los siguientes documentos se adjuntan en el Anexo #2.
Formato de hoja de datos manual, Ene/2002, Centro de Lenguaje, Anexo2.1.
Formato de factura manual, Ene/2002, Centro de Lenguaje, Anexo 2.2.
Formato de recibo provisional, Ene/2002, Centro de Lenguaje, Anexo 2.3.
Formato de letra de cambio, Ene/2002, Centro de Lenguaje, Anexo 2.4.
Formato de registro de cobros, Ene/2002, Centro de Lenguaje, Anexo 2.5.

Visión General

Este documento está basado en la Recomendación práctica IEEE para la Especificación de Requisitos de Software (Std. IEEE 830-1993) que consta de cuatro capítulos, estos son:
Introducción, Descripción General, Requisitos Específicos y Apéndices.

4.1.1.2. Descripción General

Propósito del sistema

SISADMINFAC será un software que dará apoyo al Centro de Idiomas de la Academia Cotopaxi en sus procesos de Admisiones, Matrículas y Facturación.

Diagrama de casos de uso

El diagrama de casos de uso, en el contexto del desarrollo de este trabajo se encuentra como parte del modelo funcional del sistema. Ver Anexo #5.

Requerimientos mínimos de software

Sistema Operativo: Windows NT 4.0 y Windows 9X.
Herramientas de Desarrollo: Power Builder 6.5
Base de Datos: Sql Server 6.5
Herramienta de diagramación Orientada a Objetos: Gdpro 5.0

Requerimientos mínimos de hardware

Servidor

Servidor Nefinity 3500

Pentium X86
Ram 128 MB
Disco Duro 1 GB
Unidad de drive 3 ½
Unidad de CD

Estación

Computador Personal
Clon
Pentium celeron 500 o más
Ram 64 MB
Disco Duro 1 GB
Unidad de drive 3 1/2
Unidad de CD

Requerimientos mínimos de comunicaciones

Tarjetas de red ethernet 10/100
Cable UTP nivel 5
Conectores
Hub de 8 puertos
Switch

Descripción de los casos de uso

CASO DE USO: Administrar códigos
ACTORES: Admisiones
TIPO: Primario

DESCRIPCION: Es el ingreso de los datos básicos a las tablas que manejará el sistema.

Acción de los Actores	Respuesta del Sistema
1.El representante de admisiones ingresa la información correspondiente a: Tipo de cursos Tipo de programas Periodos Profesores Paralelos Status Cuentas Pagadores Tipo de rubros Tipo de retiros Cursos	

Paralelos por curso Pagadores	2.El sistema valida la información ingresada y permite actualizar, borrar o buscar esta información.

CASO DE USO: Inscribir aspirante
ACTORES: Aspirante, Admisiones, Cobranzas
TIPO: Primario

DESCRIPCION: Es el proceso mediante el cual el aspirante es admitido.

Acción de los Actores	Respuesta del Sistema
1.El aspirante se acerca al Centro de Lenguaje y solicita inscribirse en un curso de Idiomas. 2.Admisiones le entrega al aspirante una hoja de datos. 3.El aspirante llena la hoja de datos y la entrega al representante de admisiones. 4.El representante de admisiones registra los datos del aspirante.	
	5. El sistema genera el Número de aspirante. 6. El sistema registra los datos del aspirante
7.El Aspirante se acerca a Cobranzas a pagar el valor de la inscripción y prueba. (Ver sección generar cobro). 8.El aspirante se acerca a admisiones con el recibo del pago de la inscripción y prueba de admisión para seguir con el proceso. 9.El representante de admisiones aplica la prueba de admisión correspondiente al aspirante.	
	10.El sistema registra el resultado de la prueba, registrando el nivel que aprobó. 11. El sistema registra al aspirante como activo.

Actividades alternas:

Línea6.- Si el aspirante decide no pagar, la transacción se cancela.

CASO DE USO: Matricular Alumno
ACTORES: Matrículas, Cobranzas, Alumno.
TIPO: Primario

DESCRIPCION: Es el proceso donde se promueve a un aspirante como estudiante para matricularlo en un curso o para tomar un alumno antiguo y matricularlo en un curso.

Acción de los Actores	Respuesta del Sistema
1.Si es alumno antiguo debe presentar documento de pase de nivel y no se ejecuta el paso #7. 2. Si es alumno nuevo (aspirante), entrega el # de aspirante. 3.El alumno manifiesta el curso de su preferencia. 4.El representante de matrículas verifica el cupo.	
	5.El sistema valida el cupo.
6.El representante de matrículas anuncia al alumno la los datos del curso en el que se matriculó.	
	7. El sistema genera un # de matrícula para el alumno. 8.El sistema registra el curso en el que se matriculó el alumno.
9.El representante de matrículas envía al alumno a pagar el costo total de la matrícula en cobranzas. (Ver sección Generar cobro).	

Actividades alternas:
Línea7.- Si el alumno decide no pagar, la transacción se cancela.

CASO DE USO: Generar cobros
ACTORES: Alumno, Pagador, Admisiones, Matrículas, Cobranzas, Servicio de Autorización de tarjeta.
TIPO: Primario

DESCRIPCION: Es el proceso mediante el cual se registran los cobros a los alumnos. Puede haber ajustes.

Acción de los Actores	Respuesta del Sistema
1.El alumno se acerca a cobranzas a pagar. 2.El alumno escoge la razón de su pago. • Si el alumno escoge Inscripción y prueba, ver sección Pago por inscripción. • Si el alumno escoge matrícula, ver sección pago por matrícula.	

Sección pago por inscripción

Acción de los Actores	Respuesta del Sistema
1.El aspirante manifiesta al representante de cobranzas que el pago que va a realizar es por inscripción. 2.El aspirante indica el representante de cobranzas quien será el pagador de la cuenta.	
	3.El sistema registra el rubro.
4.El pagador manifiesta la forma de pago. • Si selecciona pago total por adelantado, ver sección pago total por adelantado. • Si selecciona pago plan Diners ver sección pago Diners. • Si selecciona Plan pago con cheque, ver sección pago con cheques. 5.El representante de cobranzas manifiesta el total al pagador.	
	6.El sistema emite recibo de pago y factura.
7. El pagador realiza el pago.	

Sección pago por matrícula

Acción de los Actores	Respuesta del Sistema
1.El alumno manifiesta al	

representante de cobranzas que el pago que va a realizar es por matrícula. 2.El alumno indica el representante de cobranzas quien será el pagador de la cuenta.	
	3.El sistema registra el rubro.
4.El pagador manifiesta la forma de pago. • Si selecciona pago total por adelantado, ver sección pago total por adelantado. • Si selecciona pago plan Diners ver sección pago Diners. • Si selecciona Plan pago con cheque, ver sección pago con cheques. 5.El cobrador verifica si el alumno tiene descuento. (Ver sección descuentos)	
	6.El sistema registra el descuento.
7.El representante de cobranzas manifiesta el total al pagador. 8. El pagador realiza el pago.	
	9.El sistema emite recibo de pago y factura y envía a Spyral el asiento contable.
10.El representante de cobranzas entrega la factura al pagador. 11. El representante de cobranzas entrega al alumno una letra de cambio para que la firme. 12.El alumno devuelve firmada la letra de cambio	
	13. El sistema registra que el alumno firme la letra de cambio.

Sección pago total por adelantado

Acción de los Actores	Respuesta del Sistema
1.El pagador entrega al representante de cobranzas el efectivo. 2.Cobranzas recibe el efectivo.	
	3.El sistema registra el ingreso de abono.
4.Cobranzas entrega el vuelto al	

pagador.	

Sección pago plan Diners

Acción de los Actores	Respuesta del Sistema
1.El pagador comunica a cobranzas la información de la tarjeta con la que va a pagar.	
	2.El sistema genera la plantilla de datos para registrar datos de la tarjeta.
3.El representante de cobranzas realiza la llamada al Servicio de Autorización de tarjeta para preguntar la validez del crédito 4.El servicio de autorización de tarjeta aprueba el crédito.	
	5.El sistema registra el ingreso de abono.

Actividades alternas:
Línea4.- Si el Servicio de autorización de tarjetas desaprueba el crédito, cobranzas sugiere escoger otro tipo de pago.

Sección pago con cheque

Acción de los Actores	Respuesta del Sistema
1.El pagador escribe el cheque 2.El representante de cobranzas verifica la información del cheque.	
	3.El sistema genera la plantilla del datos para registrar los datos del cheque. 4.El sistema registra el ingreso de abono.

Sección descuentos

Acción de los Actores	Respuesta del Sistema
1. Cobranzas verifica si tiene descuento.	
	2.El sistema registra el descuento.

CASO DE USO: Aprobar alumno
ACTORES: Alumno, matrículas, Profesor
TIPO: Primario

DESCRIPCION: Permite conocer si un alumno pasa de nivel o lo repite.

Acción de los Actores	Respuesta del Sistema
1.El profesor entrega al representante de matrículas las notas obtenidas por el alumno.	
	2.El sistema hace el cálculo respectivo. Si el promedio es 75/100 el resultado es aprobado
3.Si el alumno aprobó, continua con el proceso de matrícula al siguiente nivel, si no repite el nivel.	

CASO DE USO: Retirar alumno
ACTORES: Alumno, Matrículas, Pagaduría
TIPO: Primario

DESCRIPCION: Es cuando un alumno decide retirarse del Centro de Lenguaje.

Acción de los Actores	Respuesta del Sistema
1.Luego de haber cancelado el total de la matrícula, el alumno decide no continuar el curso dentro del los 10 primeros días de iniciado el curso, solicita al representante de matrículas su retiro.	
2.El representante de matrículas calcula el total a devolver.	
	3.El sistema registra el monto
	4.El sistema genera una orden de devolución.
5.La orden de devolución se envía a Pagaduría.	
	6.El sistema cambia el estado del alumno a retirado.
7. Pagaduría elabora cheque y devuelve al alumno (esto no pertenece a este sistema).	

Características de los usuarios

Jefe de admisones.- Deberá tener conocimientos básicos de Sistema Operativo Win95/98 o WinNT y manejo de herramientas visuales.

Asistente de Matrículas y Cobranzas.- Deberá tener conocimientos básicos de Sistema Operativo Win95/98 o WinNT y manejo de herramientas visuales.

Restricciones

El Sistema no manejará:

Tipos de pruebas a ser aplicadas a los alumnos.
Automatización de horarios.

FUNCIONES GENERALES DEL PRODUCTO

FUNCIONES
Administrar códigos
Inscribir aspirantes
Matricular alumno
Generar cobros
Aprobar alumno
Retirar alumno

4.1.1.3. Requisitos específicos

El sistema SISADMINFAC manejará las siguientes clases:

NOMBRE DE LA CLASE	**DESCRIPCION**
AD_PARALELOS	Contiene los paralelos existentes en el Centro de Lenguaje. Puede también referirse a los Nombres de las empresas donde se dictan clases.
AD_TIPO_CURSO	Se refiere a la categoría de cada curso, es decir si son: Cursos regulares, One to One.
AD_TIPO_PRO	Son los tipos de programas existentes: Adultos o Niños.
AD_PROFESOR	Almacena la nómina de profesores que trabajan en el Centro de Lenguaje.
AD_PERIODO	Se refiere al año escolar vigente.
AD_PARA_CURSO	Contiene los paralelos por curso.
AD_STATUS	Tabla que contiene el Status de distintos objetos, por ejemplo: T = Activo, E = Estudiante.
AD_ASPIRANTE	Se almacena los datos correspondientes a los aspirantes.
AD_ESTUDIANTE	Se almacena los datos correspondientes a los estudiantes.
AD_AJUSTES	Tabla que contiene los ajustes correspondientes a un rubro, los cuales pueden ser descuentos o cambio de valor del rubro (cuando se retira un

	alumno).
AD_ESTU_DET_MATRICULAS	Detalle donde se almacena información tanto de los aspirantes, estudiantes y rubros.
AD_TIPO_RUBROS	Tabla que almacena el Tipo de rubro, por ejemplo: Pago por inscripción, Costo por laboratorio.
AD_RUBROS	Contiene los rubros generados.
AD_ABONOS	Tabla que almacena los datos correspondientes a los abonos que hace un aspirante o alumno. En esta tabla se encuentran todos los datos necesarios para realizar el Asiento contable.
AD_ABODEMAT	Tabla que contiene detalles del abono.
AD_PAGADORES	Contiene los datos de la persona representante de un espirante o alumno para realizar el pago.
AD_ESTU_HIST_ESCOLAR	Se almacena la información del récord escolar del alumno.
AD_ESTU_RETIRO	Posee los datos de los retiros de los alumnos.
AD_TIPO_RETIRO	Almacena las razones de los retiros.
AD_CUENTAS	Tiene los datos para indicar si un aspirante o estudiante son de La Academia, Individuales o Empresas.

Las operaciones del SISADMINFAC son las siguientes:

Insertar	Ingresar datos en una tabla.
Eliminar	Borrar datos de una tabla siempre y cuando no viole integridad referencial.
Guardar	Grabar datos en una tabla.
Registrar pago aspirante estudiante	Es la fase inicial para el proceso de cobros.
Generar rubros	Proceso mediante el cual se crea una plantilla para registrar los cobros.
Registrar detalles aspirante matricula	Permite almacenar los detalles de la inscripción o matrícula de un alumno.
Registrar detalle abonos	Es un proceso mediante el cual se registran los pagos cancelados por el alumno.
Registrar pago aspirante estudiante	Es la fase inicial para el proceso de cobros.
Matricular estudiante curso	Permite asignar a un alumno un curso.
Reactivar estudiante	Es cambiar el estado cuando un alumno se ha retirado.
Promover aspirante a estudiante	Es asignar un número de matrícula a un aspirante.
Aprobar estudiante	Es actualizar a un estudiante de si aprobó o no el curso.

En el Anexo # 7 correspondiente al Manual Técnico se encuentra la descripción de las columnas de las tablas del sistema.

4.2. Diseño del Sistema y de Objetos

4.2.1. Modelo de Objetos.

Representa la estructura estática del sistema.

Diagramas del modelo de Objetos:

- Diagrama de Clases
- Diagrama de Objetos

Ver los diagramas en el Anexo # 3.

4.2.2. Modelo Dinámico.

Representa los aspectos temporales y de comportamiento.

Diagramas del modelo Dinámico:

- Diagramas de secuencia
- Diagramas de colaboración
- Diagramas de estados
- Diagramas de actividades

Ver los diagramas en el Anexo # 4.

4.2.3. Modelo Funcional

Representa los procesos de transformación

Diagrama del modelo funcional:

- Diagrama de casos de uso

Ver el diagrama en el Anexo # 5.

4.2.4. Implementación

Diagramas de la implementación

- Diagramas de componentes
- Diagramas de despliegue

Ver los diagramas en el Anexo # 6.

4.3. Pruebas

Ver los casos de pruebas en el CD adjunto a este trabajo en la carpeta Pruebas del Sistema.

4.4. Manual Técnico del Sistema y Manual de Usuario

Ver el Manual Técnico en el Anexo # 7 (Creación de la Base de datos), ver el resto del manual en el CD adjunto en la carpeta Manual Técnico.

Ver el Manual de Usuario en el CD adjunto a este trabajo en la carpeta Manual de Usuario.

CAPITULO V

5. Conclusiones y Recomendaciones

5.1. Conclusiones

- La metodología Orientada a Objetos proporciona una mejor visión de las aplicaciones a desarrollarse ya que esta se basa en el dominio de la aplicación y no en los requisitos funcionales de un problema.

- El uso de la metodología orientada a objetos permite el poder compartir información dentro de una aplicación, ofreciendo la posibilidad de reutilizar diseño y código en otros proyectos.

- Los tres modelos de OMT son partes claves para la descripción de un sistema en forma completa, pero el modelo más importante de la metodología utilizada en el presente trabajo es el modelo de objetos ya que éste nos proporciona la información necesaria para determinar los actores de la aplicación antes de describir cuándo o cómo funcionan dichos actores.

- La base de datos que utilicé SQL Server no es orientada a objetos, por lo que se debió traducir el modelo de Objetos a un Modelo Entidad Relación.

5.2. Recomendaciones.

- Para una mejor comprensión de lo que un sistema debe hacer se recomienda seguir los siguientes pasos:

    - Un levantamiento de información con los usuarios involucrados. A cada uno de ellos se le debe preguntar cómo hacen su trabajo. Cada uno de los usuarios constituye un actor.
    - A continuación se debe identificar los procesos desempeñados por cada actor. A los procesos identificados se les denomina casos de uso. Con esta información se puede escribir el documento de Casos de uso y dibujar el Diagrama de Casos de Uso.
    - Seguidamente se debe realizar el Diagrama de Clases, para ello nos basamos en el documento Casos de Uso, identificamos en esta narrativa los sustantivos quienes pasan a ser las clases, los verbos que pasan a ser las operaciones y las responsabilidades que pasan a ser las funciones de cada clase. Adicionalmente se debe dibujar las relaciones entre clases.

  Estos tres pasos serían suficientes para comenzar con el desarrollo de la aplicación, debido a que el modelo de objetos es el más importante de metodología orientada a objetos.

- La herramienta utilizada para diseñar los diagramas fue Gdpro 5.0, ésta no permitió realizar la generación directa de la base de datos a partir del modelo de clases, debido a que la herramienta no cuenta con esta característica y es por ello que se recomienda utilizar una herramienta más completa como Rational Rose. La Academia Cotopaxi no pudo realizar la adquisición de Rational Rose por lo cual se utilizó Gdpro 5.0.

Bibliografía

Libros:

Rumbaugh James, Blaha Michael, Premerlani William, Eddy Frederick, Lorensen William (1996), Modelado y Diseño Orientado a Objetos, Prentice Hall International, España.

Booch Grady, Rumbaugh James, Jacobson Ivar (1999), Lenguaje Unificado de Modelado, Addison Wesley Iberoamericana, Madrid.

Pressman Roger (1998), Ingeniería del Software, un enfoque práctico. Editorial McGraw-Hill. España.

Links del internet:

http://www.mcc.unam.mx/~cursos/Objetos/Omt/omt.html
http://www.lafacu.com/apuntes/informatica/inge_soft/isw5/default.htm
http://www.dcc.uchile.cl/~cc61j/

INDICE

RESUMEN EJECUTIVO

CAPITULO I 1
1. Conceptos Generales 1
1.1. Teoría de Análisis y Diseño Orientado a Objetos 1
1.1.1. Paradigma Orientado a Objetos. 1
1.1.1.1 Modelos Orientados a Objetos. 2
1.1.1.2. Análisis Orientado a Objetos. 3
1.1.1.3. Ventajas del Análisis Orientado a Objetos (AOO). 4
1.1.2. Clases y Objetos. 4
1.1.2.1. Clases. 4
1.1.2.2. Objetos. 5
1.1.2.3. Atributos 5
1.1.2.2.1. Características de los objetos. 6
1.1.2.2.1.1. Identidad. 6
1.1.2.2.1.2. Clasificación. 6
Objetos auto Clase auto 6
1.1.2.2.1.3. Concurrencia. 7
1.1.2.2.1.4. Persistencia. 7
1.1.3. Abstracción. 7
1.1.4. Herencia. 7
1.1.5. Polimorfismo. 8
1.1.6. Encapsulamiento. 8
CAPITULO II 9
2. Análisis de la Metodología Object Modelling Technique OMT 9
2.1. Fases de metodología OMT 9
2.1.1. Conceptualización 9
2.1.1.1. Análisis de requerimientos 9
2.1.1.2. Casos de uso 9
2.1.2. Análisis 9
2.1.3. Diseño del sistema 9
2.1.4. Diseño de objetos 10
2.1.5. Implementación 10
2.2. Modelos de OMT 10
2.2.1. Modelo de Objetos 11
2.2.1.1. Operaciones y Métodos 11
2.2.1.2. Notación para las clases objetos 12
2.2.1.3. Enlaces y Asociaciones 12
2.2.1.3. Multiplicidad. 13
2.2.1.4. Atributos de los enlaces 14
2.2.1.5. Roles 15
2.2.1.6. Clasificación 15
2.2.1.7. Cualificación 15
2.2.1.8. Agregación 16
2.2.1.9. Generalización y herencia. 16
2.2.1.10. Restricciones 17

2.2.1.11. Herencia múltiple. 17
2.2.1.12. Metadatos 18
2.2.2. Modelo dinámico 18
2.2.2.1. Diagramas de estados 18
2.2.2.2. Estados y eventos(sucesos) 19
2.2.2.3. Escenarios. 19
2.2.2.4. Trazas de eventos (sucesos) 20
2.2.2.5. Actividades y Acciones 20
2.2.2.5. Transición 21
2.2.3. Modelo funcional 21
CAPITULO III 24
3. Análisis del Lenguaje de Análisis y Diseño Unified Modelling Languaje UML .. 24
3.1. Breve historia de UML 24
3.2. Elementos de UML 24
3.2.1. Elementos estructurales 24
3.2.2. Elementos de comportamiento 25
3.2.3. Elementos de agrupación 25
3.2.4. Elementos de anotación 26
3.3. Relaciones en UML 26
3.4. Arquitectura 26
3.5. Términos y conceptos 27
3.6. Diagramas UML 28
3.6.1. Diagramas Estructurales 28
3.6.1.1. Diagrama de clases(vista estática). 28
3.6.1.2. Diagrama de objetos(vista estática). 29
3.6.1.3. Diagrama de componentes(vista estática). 29
3.6.1.4. Diagrama de despliegue(vista estática). 30
3.6.2. Diagramas de comportamiento 30
3.6.2.1. Diagrama de casos de uso(vista dinámica). 30
3.6.2.2. Diagramas de interacción 32
3.6.2.2.1. Diagrama de secuencia(vista dinámica). 32
3.6.2.2.2. Diagrama de colaboración(vista dinámica). 33
3.6.2.2.2. Diagrama de estados(vista dinámica). 34
3.6.2.2.3. Diagrama de actividades(vista dinámica). 35
CAPITULO IV 37
4. Desarrollo del Sistema 37
4.1. Análisis 37
4.1.1. Especificación de Requerimientos de Software (SRS) 37
4.1.1.1. Introducción 37
4.1.1.2. Descripción General 38
4.1.1.3. Requisitos específicos 46
4.2. Diseño del Sistema y de Objetos 48
4.2.1. Modelo de Objetos. 48
4.2.2. Modelo Dinámico. 48
4.2.3. Modelo Funcional 48
4.2.4. Implementación 48
4.3. Pruebas 49
4.4. Manual Técnico del Sistema y Manual de Usuario 49

CAPITULO V 50
5. Conclusiones y Recomendaciones..... 50
5.1. Conclusiones..... 50
5.2. Recomendaciones. 51
Bibliografía..... 52