CAPITULO II

MARCO TEORICO

2.1 METODOLOGÍA

Al desarrollar un software nos preguntamos en repetidas ocasiones que metodología usamos, de todas aquellas que hemos visto en el transcurso de la universidad; debemos escoger la más productiva, en cada caso, que cumpla con las facilidades de diseño y desarrollo del proyecto que vamos a realizar.

En nuestro caso, lo primordial en que nos hemos basado es en la flexibilidad de cambios y previsiones que se pueden dar en el transcurso del proyecto, con la finalidad de que no presente mayores problemas cuando se haga la creación o modificación de reportes que pida la empresa auspiciante.

Es importante que recalquemos en un buen diseño y funcionalidad del sistema, que refleje nuestro profesionalismo y responsabilidad sobre el proyecto encargado.

Gould, Boies y Lewis identifican cuatro principios elementales[1]:

a) Enfoque temprano en el usuario. Los diseñadores deben esforzarse en comprender las necesidades de los usuarios desde la primera fase del proceso de diseño.

1 Gould, Boies y Lewis (1991) identifican cuatro principios en el diseño centrado en el usuario

b) Diseño integrado. Los aspectos del diseño guardan una relación paralela más que secuencial. Es fundamental mantener el diseño interno en consonancia con las necesidades de la interfaz de usuario.

c) Pruebas tempranas y continuas. El único enfoque actual factible en el diseño de software es el empírico: el diseño funciona si el usuario final decide que funcione. La incorporación de pruebas de facilidad de uso en el proceso de desarrollo, ofrece al usuario la posibilidad de enviar comentarios sobre el diseño antes del lanzamiento del producto.

d) Diseño iterativo. Los grandes problemas son, a menudo, un cúmulo de pequeños errores. Los diseñadores y desarrolladores deben revisar el diseño de forma iterativa en tandas de pruebas.

2.2 ARQUITECTURAS Y METODOLOGÍAS EN LA ACTUALIDAD

Actualmente existen muchas metodologías de desarrollo de software, desde métodos muy pesados y burocráticos, métodos ajustables al proyecto y a las condiciones de desarrollo, hasta métodos ligeros que surgen como respuesta a los excesos formales de otros métodos.

Evidentemente, partiendo de los principios de tantas y diversas metodologías es muy difícil sacar una visión unificada sobre el diseño arquitectónico. Sin embargo sí que podemos destacar una serie de elementos comunes en aquellas que más se centran en este tema.

El primero es la existencia de una fase en la que se establece o diseña una arquitectura base y, el segundo, la altísima dependencia entre los casos de uso

y la arquitectura, definiendo un caso de uso como una interacción (secuencia de acciones) típica entre el usuario y el sistema.

2.3 METODOLOGIA OMT

Por todos sus beneficios se ha escogido OMT[2]. La metodología OMT fue creada por James Rumbaugh y Michael Blaha en 1991, mientras James dirigía un equipo de investigación de los laboratorios General Electric.

OMT es una de las metodologías de análisis y diseños orientados a objetos, más maduros y eficientes que existen en la actualidad.

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

OMT está construido alrededor de descripciones de estructura de datos, constantes, sistemas para procesos de transacciones.

Desde que la comunidad de programación orientada a objetos tuvo la noción de incorporar el pensamiento de que los objetos son entidades coherentes con

2 Object Modeling Technique

identidad estado y conducta, estos objetos pueden ser organizados por sus similitudes y sus diferencias, puestas en uso en herencia y polimorfismo.

OMT pone énfasis en especificaciones claras de la información, para capturar limpiamente los requerimientos, especificaciones imperativas para poder descender prematuramente en el diseño, declaraciones que permiten optimizar los estados; además provee un soporte declarativo para una directa implementación de DBMS[3].

2.4 PROCESO DE DESARROLLO DE OMT

Los pasos para desarrollar diseño OMT son:

2.4.1 CONCEPTUALIZACIÓN

El desarrollo empieza con el análisis de la empresa o negocio, o de cómo los usuarios conciben el sistema y formulan sus requerimientos. La conceptualización es a menudo reutilizada por la reingeniería de procesos de la empresa, es una observación crítica de los procesos de la empresa, y su impacto económico. En esta etapa se debe tener en cuenta las siguientes preguntas:

- ¿Cuál es la aplicación?
- ¿Qué problemas tendrán que ser resueltos?
- ¿Dónde será usado el sistema?
- ¿Cuándo será requerido el sistema?
- ¿Para qué es necesario el sistema?

3 Sistemas Manejadores de Base de Datos

2.4.2 ANÁLISIS

Los requerimientos formados durante la conceptualización son revisados y analizados para la construcción del modelo real. La meta del análisis es especificar las necesidades que deben ser satisfechas.

Pueden existir diversas fuentes de información que pueden servir para el análisis, puede existir un lenguaje formal para describir el problema. Algunas veces los expertos del dominio pueden proveer escenarios, y casos de uso para un nuevo sistema.

Aquí es donde se determina el modelo de objeto, se hace una tentativa de clases para eliminar las clases irrelevantes, las posibles asociaciones entre las clases, luego se hace la refinación de asociaciones eliminando las redundantes o las que no tienen relevancia, posteriormente se hace una tentativa de atributos de objetos y enlaces.

Una vez obtenidos los objetos del sistema, se hace un refinamiento del modelo, posteriormente se busca un nivel de abstracción para modelar subsistemas, para buscar un sistema tangible y sólido.

Desarrollado el modelo, se introduce la noción de transacción, es una forma de modelar procesos o describir cambio de datos, movimiento de datos.

2.4.3 DISEÑO DEL SISTEMA

El diseño tiene un alto nivel estratégico y decisión para resolver los problemas.

Los problemas grandes se deben considerar desde el punto de vista de análisis y diseño; este sistema se divide en subsistemas, a su vez este subsistema puede ser dividido en otros subsistemas de manera que puedan ser manejados y cada componente pueda ser comprensible.

En esta etapa se deben crear estrategias, formular una arquitectura para el sistema y las políticas que deben guiarla, además un detalle del diseño.

2.4.4 MANTENIMIENTO

La documentación del desarrollo y seguimiento de los modelos a través del código facilita el posterior mantenimiento. La metodología OMT soporta múltiples estilos de desarrollo. Se puede usar OMT para conseguir un alto performance en la fase de análisis y diseño e implementación con una estricta secuencia de pasos, también adopta una estrategia de desarrollo iterativa.

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

- **Modelo de objetos.** Describe la estructura estática de los objetos del sistema (identidad, relaciones con otros objetos, atributos y operaciones). El modelo de objetos proporciona el entorno esencial en el cual se pueden situar el modelo dinámico y el modelo funcional. El objetivo es capturar aquellos conceptos del mundo real que sean importantes para la aplicación. Se representa mediante diagramas de objetos.

- **Modelo dinámico.** Describe los aspectos de un sistema que tratan de la temporización y secuencia de operaciones (sucesos que marcan los cambios, secuencias de sucesos, estados que definen el contexto para los sucesos) y la organización de sucesos y estados. Captura el control, aquel aspecto de un sistema que describe las secuencias de operaciones que se producen sin tener en cuenta lo que hagan las operaciones, aquello a lo que afecten o la forma en que están implementadas. Se representa gráficamente mediante diagramas de estado.

- **Modelo funcional.** Describe las transformaciones de valores de datos (funciones, correspondencias, restricciones y dependencias funcionales) que ocurren dentro del sistema. Captura lo que hace el sistema, independientemente de cuando se haga o de la forma en que se haga. Se representa mediante diagramas de flujo de datos

2.5 PROGRAMACION ORIENTADA A OBJETOS

Término de Programación Orientada a Objetos indica más una forma de diseño y una metodología de desarrollo de software que un lenguaje de programación, ya que en realidad se puede aplicar el Diseño Orientado a Objetos, a cualquier tipo de lenguaje de programación.

El desarrollo de la POO[4] empieza a destacar durante la década de los 80 tomando en cuenta la programación estructurada, a la que engloba y dotando al programador de nuevos elementos para el análisis y desarrollo de software.

4 Programación Orientada a Objetos

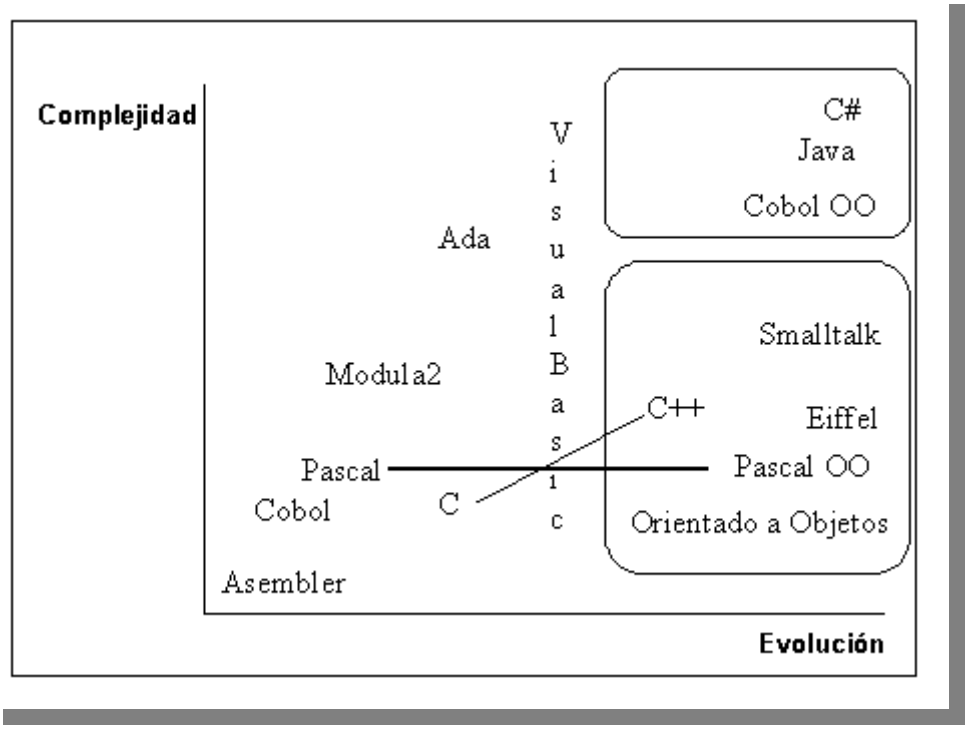

Figura No.1 (Evolución y Complejidad de la programación orientada a objetos)

En la programación orientada a objetos tenemos dos elementos fundamentales, las CLASES y los OBJETOS. Podemos definir a la clase como la generalización de los objetos y los objetos como la concreción de la clase.

Esta metodología, como se mencionó antes, gira en torno a términos tales como abstracción, encapsulamiento, herencia y polimorfismo.

2.5.1 ESTRUCTURA DE UN OBJETO

Un objeto puede considerarse como una especie de cápsula dividida en tres partes:

1 - RELACIONES

2 - PROPIEDADES

3 - METODOS

Cada uno de estos componentes desempeña un papel totalmente independiente:

Las **relaciones** permiten que el objeto se inserte en la organización y están formadas esencialmente por punteros a otros objetos.

Las **propiedades** distinguen un objeto determinado de los restantes que forman parte de la misma organización y tiene valores que dependen de la propiedad de que se trate. Las propiedades de un objeto pueden ser heredadas a sus descendientes en la organización.

Los **métodos** son las operaciones que pueden realizarse sobre el objeto, que normalmente estarán incorporados en forma de programas (código) que el objeto es capaz de ejecutar y que también pone a disposición de sus descendientes a través de la herencia.

2.5.2 BENEFICIOS QUE SE OBTIENEN DEL DESARROLLO CON PROGRAMACION ORIENTADA A OBJETOS

Día a día los costos del hardware decrecen. Así surgen nuevas áreas de aplicación cotidianamente: procesamiento de imágenes y sonido, bases de datos multimediales, automatización de oficinas, ambientes de ingeniería de software, etc.

Lamentablemente, los costos de producción de software siguen aumentando; el mantenimiento y la modificación de sistemas complejos suele ser una tarea trabajosa; cada aplicación, (aunque tenga aspectos similares a otra) suele encararse como un proyecto nuevo.

La introducción de tecnología de objetos como una herramienta conceptual para analizar, diseñar e implementar aplicaciones permite obtener aplicaciones más modificables, fácilmente entendibles y a partir de componentes rehusables. Esta rehusabilidad del código disminuye el tiempo que se utiliza en el desarrollo y hace que el desarrollo del software sea más intuitivo porque la gente piensa naturalmente en términos de objetos más que en términos de algoritmos de software.

2.6 VISUAL BASIC NET

2.6.1 INTRODUCCION A VISUAL BASIC

Hace 10 años, el proceso de construir una simple aplicación basada en Microsoft Windows se convertía en un acto complicado, difícil y largo. Construir estas aplicaciones ricas en gráficos no era un proceso trivial. Con la aparición de Visual Basic 1.0 en mayo de 1991, los programadores podían, por primera vez, implementar aplicaciones de Windows en un ambiente intuitivo y gráfico, simplemente arrastrando controles sobre un formulario. Visual Basic conllevó un renacimiento del desarrollo de aplicaciones basadas en Windows. En los

últimos 10 años, la comunidad de Visual Basic se expandió hasta convertirse en la mayor comunidad de desarrolladores de software del mundo.

Hoy, se continúa ampliando las posibilidades del desarrollador en Visual Basic. Con Visual Basic .NET, se posibilita a los desarrolladores en Visual Basic con niveles de control y productividad sin precedentes. A través de objetos-orientados de primera-clase, herencia, manejo estructural excepcional, y construcciones con parámetros. Con el acceso completo al marco del NET de Microsoft, se puede, por primera vez, conseguir ventaja directa de la rica plataforma de Microsoft y construir aplicaciones tradicionales basadas en Windows, aplicaciones Web de pequeños clientes, los servicios de nueva generación de Web de XML, y software para móviles.

2.6.2 ¿QUE ES .NET?

Es un entorno de desarrollo independiente del lenguaje, que permite escribir programas de forma sencilla, e incluso permite combinar código escrito en diferentes lenguajes.

2.6.3 ¿QUÉ ES EL .NET FRAMEWORK?

Se define como: “NET Framework [5] es un entorno para construir, instalar y ejecutar servicios Web y otras aplicaciones.

[5] En el libro de **Microsoft .NET Framework**

2.6.3.1 Ventajas:

- Gestión de memoria y seguridad
- Libera al programador de muchas tareas
- Permite que se concentre en la lógica del programa

2.6.3.2 Seguridad

Proporciona esquemas de autorización y autenticación predeterminados para aplicaciones Web. Resulta fácil eliminar, agregar o reemplazar estos esquemas dependiendo de las necesidades de la aplicación.

2.7 REQUERIMIENTOS PARA VISUAL STUDIO .NET

Los requerimientos varían según las combinaciones si se incluye una versión de MSDN Library, se necesita:

Tabla 2.1 (Requerimientos para Visual .Net)

Procesador	Computadora personal (PC) con un procesador Pentium II, 450 MHz
Sistema operativo	Windows 2000, Windows NT, Windows Server 2003, Windows XP
Memoria	Microsoft Windows® XP Professional 160 megabytes (MB) de RAM Windows 2000 Professional 96 MB de RAM Windows 2000 Server 192 MB de RAM Windows NT 4.0 Workstation 64 MB de RAM
Disco duro	Professional y Enterprise Edition. 3.5 GB en la unidad de instalación, que incluye 500 MB en la unidad de sistema
Unidades	Unidad de CD-ROM o DVD-ROM
Video	Monitor Super VGA (800 x 600) con 256 colores o de alta resolución
Mouse	Microsoft Mouse o dispositivo compatible

2.8 BASE DE DATOS

El concepto básico en el almacenamiento de datos es el registro. El registro agrupa la información asociada a un elemento de un conjunto, y está compuesto por campos.

El modelo relacional, basado en tablas, tiene en la actualidad una difusión mayor. Las búsquedas pueden ser mucho más flexibles, basadas en cualquier campo.

El gestor de base de datos ordena, diseña, define y utiliza los registros, ficheros y formularios.

2.9 SQL

SQL[6] es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos informática.

"El SQL trabaja con estructura cliente/servidor sobre una red de ordenadores. El ordenador cliente es el que inicia la consulta; el ordenador servidor atiende esa consulta. El cliente utiliza toda su capacidad de proceso para trabajar; se limita a solicitar datos al ordenador servidor, sin depender para nada más del exterior. Estas peticiones y las respuestas son transferencias de textos que cada ordenador cliente se encarga de sacar por pantalla, presentar en informes tabulados, imprimir, guardar, etc., dejando el servidor libre"[7]

6 *Structured Query Languaje* (Lenguaje de consultas estructurado)

7 *Curso 95/96 Universidad de Navarra*

2.9.1 QUÉ ES Y PARA QUÉ SIRVE EL SQL

Como su propio nombre indica, SQL es un lenguaje informático que se puede utilizar para interaccionar con una base de datos.

En la actualidad existen numerosas bases de datos y lenguajes de programación que el programador puede utilizar, pero la comunicación entre estos sería demasiado complicada por esto se ha creado los estándares que permiten realizar operaciones básicas de manera que se puedan entender entre diversos lenguajes y bases de datos.

2.10 MICROSOFT SQL SERVER 2000 DESKTOP ENGINE (MSDE 2000)

Es la versión de SQL Server que se distribuye gratuitamente, se puede utilizar en aplicaciones cliente o pequeñas aplicaciones Web de hasta 25 usuarios simultáneos.

Si bien tiene algunas restricciones, como un tamaño de base de datos máximo de 2 GB y un límite en la cantidad de sentencias procesadas en paralelo, es completamente compatible con Microsoft SQL Server 2000. Incluso soporta características como la creación de procedimientos almacenados y desencadenadores, y ofrece las mismas opciones de seguridad.

2.10.1 ESCENARIOS DE USO DE MSDE

Se usa para dos escenarios, MSDE 2000 con aplicaciones de escritorio para usuario final y pequeños sitios Web.

Los sitios Web también pueden utilizar MSDE 2000 como motor de base de datos. MSDE puede ser instalado con sus aplicaciones sin pago de regalías, pero existen algunas restricciones:

- Tamaño de base de datos máximo de 2GB.
- Máximo de 8 sentencias procesadas en paralelo.
- Máximo de 16 instancias por máquina.
- No incluye herramientas de administración.

2.10.2 REQUISITOS DEL SISTEMA

Tabla 2.2 (Requerimientos para MSDE)

Procesador	Intel Pentium o compatible a 166 MHz o superior
Sistema operativo	Windows 2000, Windows 98, Windows ME, Windows NT, Windows Server 2003, Windows XP
Memoria	128 MB de memoria RAM (Windows XP), 64 MB de memoria RAM (Windows 2000) 32 MB de memoria RAM para los demás sistemas operativos
Disco duro	2.5 GB
Unidades	Unidad de CD-ROM o DVD-ROM
Video	Monitor Super VGA (800 x 600) con 256 colores o de alta resolución
Mouse	Microsoft Mouse o dispositivo compatible

2.11 CONEXIONES DE SQL SERVER

SQL Server crea un programador diferente para cada una de las CPU del sistema. Cuando los clientes conectan con el servidor, se les asigna el programador que tenga el menor número de conexiones. Una vez conectado, un cliente nunca cambia de programador, sino que sigue con el programador asignado hasta que se desconecta.

Si una aplicación no tiene un diseño adecuado o no distribuye uniformemente el trabajo entre sus conexiones, es posible que algunas de las conexiones de la aplicación tengan que luchar de manera innecesaria por los recursos de la CPU, mientras que otras permanecen prácticamente desocupadas.

2.11.1 TIPOS DE CONEXIONES

ODBC

- Conectividad para Bases de Datos Abiertas
- Proveedor con más éxito hasta la fecha
- se crea para definir y promover estándares de acceso a datos.
- Objetivo Poder cambiar la BD sin cambiar el código
- Proporciona serie de funciones que permiten controlar el acceso a datos.
- Ofrece un conjunto de instrucciones para diferentes sistemas independiente de la plataforma, proveedor, BD y lenguaje que se emplee.

DAO

- Es el primer modelo de datos en Visual Basic.
- Apareció con Visual Basic 3.0 (1992)
- Es una tecnología diseñada entorno al JET capaz de acceder a BD Access y base de datos ISAM(dBase, PAradox, FoxPro) y ODBC.
- Funcionaba muy bien accediendo a BD en el mismo PC, y no tanto en acceso a BD remotas

OLE DB

- API de Microsoft para el acceso a datos tanto BD relacionales como no relacionales.
- Se basa en el modelo de componentes .COM
- Se divide en dos tipos de componentes: consumidores y proveedores, Consumidores Utilizan los datos, Proveedores Hablan con las fuentes de datos y suministran los datos a los consumidores

RDS

- Servicio de datos remota.
- Similar a ADO pero diseñado para proporcionar tecnología OLE DB para aplicaciones en la Web.