UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de un sistema que realice el proceso de asignación del concurso de ingreso a la educación media superior

TESIS

Que para obtener el título de

Ingeniero en Computación

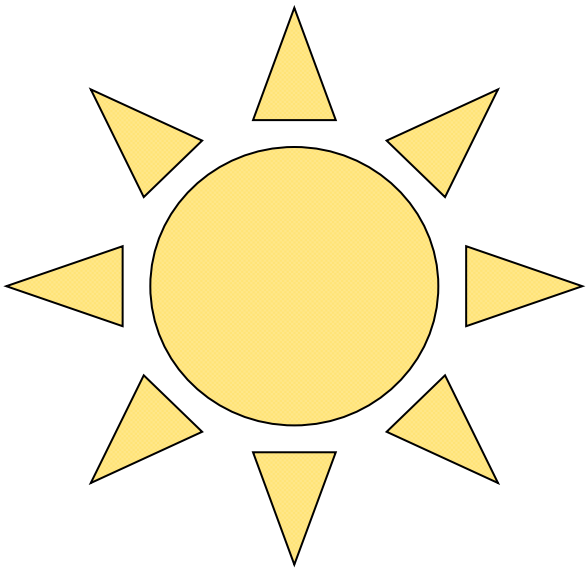
P R E S E N T A

Oliver Espinosa Colchado

DIRECTOR DE TESIS

Ing. Lucila P. Arellano Mendoza

Ciudad Universitaria, Cd. Mx., 2003



Prólogo

Prólogo

A partir de que en 1790 la revolución industrial, iniciada en Inglaterra, buscara la mecanización de los procesos de producción, el hombre ha venido creando máquinas o herramientas que le ayuden a realizar su trabajo más rápido, con un menor esfuerzo y con una mejor calidad.

Existe un grupo de máquinas especiales que realizan cálculos de una manera muy rápida y que desde su aparición han cambiado radicalmente la forma en la cual el mundo entero se comunica, controla, automatiza y sistematiza su trabajo, dichas máquinas son las computadoras.

Uno de los primeros ejemplos de la aplicación de una máquina que realizaba cálculos de una manera muy rápida fue la creada por Herman Hollerith, quien desarrolló una máquina de cálculo capaz de reducir el trabajo y el tiempo de entrega de los resultados del censo de los Estados Unidos de Norte América. A partir de ese momento y hasta nuestros días, este tipo de máquina ha evolucionado y se ha utilizado en casi todas las áreas del conocimiento del hombre, en nuestros hogares y en casi todas las industrias. En un sentido general puedo decir que son máquinas que han automatizado y sistematizado tareas diarias, máquinas que han ofrecido una alternativa que permite reducir esfuerzo, dinero, tiempo y que realizan su trabajo con una alta calidad.

En el siglo pasado la computadora fue llevada de los laboratorios de las universidades a los hogares de muchas familias en todo el mundo, esto cambió la manera en la cual las personas trabajan y se comunican. Desde la década de 1970, las computadoras se han conectado entre ellas y gracias a eso hoy en día el mundo cuenta con una red de alcance mundial llamada Internet que permite publicar y compartir información en diferentes formatos (sonido, texto y vídeo por ejemplo), vender casi cualquier artículo (comercio electrónico) y reducir distancias de comunicación (vídeo conferencias y correo electrónico).

Muchas de las tareas que las computadoras realizan día con día han sido programadas por el hombre, es decir, el hombre tiene que indicarle a la computadora la secuencia de instrucciones (o código de computadora) que ésta debe de seguir para que, por ejemplo, se cambien correctamente las luces de los semáforos en una intersección de calles o bien, se calcule la nómina de toda una empresa.

El presente trabajo de tesis trata sobre la creación de una herramienta (resultado de la unión de una computadora con secuencias de instrucciones o código) que permite realizar de una manera sistematizada el proceso conocido como “Asignación” y que consiste en asignarle un lugar a cada uno de los miles de jóvenes aspirantes a ingresar a las instituciones de educación media superior de a cuerdo a las bases del Concurso de Ingreso a la Educación Media Superior de la Zona Metropolitana de la Ciudad de México.

Esta tesis está estructurada de la siguiente manera:

1. Primera parte: Antecedentes, marco teórico y el concurso de ingreso

2. Segunda parte: Etapas del desarrollo del Sistema del Proceso de Asignación.

En la primera parte se presenta brevemente un poco de historia reciente relativa a las computadoras y también se mencionan los antecedentes referentes al Concurso de Ingreso. El marco teórico se enfoca exclusivamente a las ciencias encargadas del desarrollo de sistemas basados en computadora.

El objetivo de esta primera parte es ubicar al lector dentro del problema que estoy resolviendo y también decirle en qué ciencia de la ingeniería me he apoyado para resolverlo. Esta parte está compuesta por los siguientes capítulos:

Capítulo 1: Antecedentes

En este primer capítulo se presentan los hechos históricos que se vinieron presentando y que dieron origen a las computadoras como herramientas de trabajo y de sistematización de tareas. También se presenta el origen del Concurso de Ingreso y cómo es que éste permitió resolver algunos de los problemas que se generaban con las convocatorias de ingreso independientes y que las instituciones de educación media superior de la Zona Metropolitana de la Ciudad de México venían enfrentando de manera aislada.

Capítulo 2: Marco teórico

En este capítulo se presenta una vista general de las ciencias encargadas del estudio de los sistemas basados en computadora. Se habla acerca de los tipos de sistemas que existen y de su jerarquía. Finalmente se presentan los diferentes ciclos de vida de un proyecto de desarrollo de sistemas basados en computadora.

Capítulo 3: El Concurso de Ingreso

Este capítulo describe el objetivo y las etapas que componen al Concurso de Ingreso a la Educación Media Superior de la Zona Metropolitana de la Ciudad de México en su versión 2001.

En la segunda parte se presenta la metodología utilizada para sistematizar el proceso de asignación del Concurso de Ingreso, una a una son presentadas las tareas que permitieron sistematizar la asignación de los participantes del concurso. Esta segunda parte inicia con la creación de un modelo que describe la solución propuesta para el sistema de asignación y termina con las pruebas y las recomendaciones para versiones futuras del mismo.

Los capítulos que componen la segunda parte son:

Capítulo 4: Análisis del proceso de asignación

Este capítulo presenta la etapa con la que se inició la creación del sistema de asignación, en él se trata de responder en un principio a la siguiente pregunta ¿cuál es el problema que se desea resolver? Una vez entendido el problema a resolver, el capítulo presenta un modelo que describe el proceso de asignación y que servirá como punto de partida para el diseño del mismo.
Para ello, se presentan tanto las tareas propias del análisis de sistemas como las herramientas que son utilizadas en él, etapas y herramientas que el grupo de personas encargadas del desarrollo del sistema han seguido y utilizado, como ya se mencionó, para crear un modelo que describa el proceso de asignación y así responder a la pregunta antes presentada.

Capítulo 5: Diseño del proceso de asignación

Una vez que el proceso de asignación ha sido comprendido y representado de una manera abstracta en un modelo, el presente capítulo trata de responder a la siguiente pregunta ¿cómo resolver el problema que tenemos en nuestras manos? Es decir, este capítulo

presenta los conceptos y las técnicas de diseño de sistemas que permiten describir, en términos de programación, la solución propuesta para el sistema de asignación.
Los conceptos y las técnicas expuestas en este capítulo incluyen, a la programación estructurada, los diagramas de flujo y el lenguaje de diseño de programas.

Capítulo 6: Implantación del proceso de asignación

Este capítulo trata sobre los conceptos y las técnicas de programación utilizadas en la creación del sistema de asignación.
Conceptos y técnicas que permiten obtener un código de computadora de calidad, es decir, código con una estructura estándar, fácil de entender, fácil de revisar, fácil de corregir y que contiene las secuencias de instrucciones que corresponden a los modelos creados en los capítulos 4 y 5.
El capítulo presenta, entre otros, los conceptos de rutina y de módulo, así como también las técnicas para programar de una manera defensiva y algunos consejos para nombrar las diferentes rutinas y variables utilizadas por el sistema.

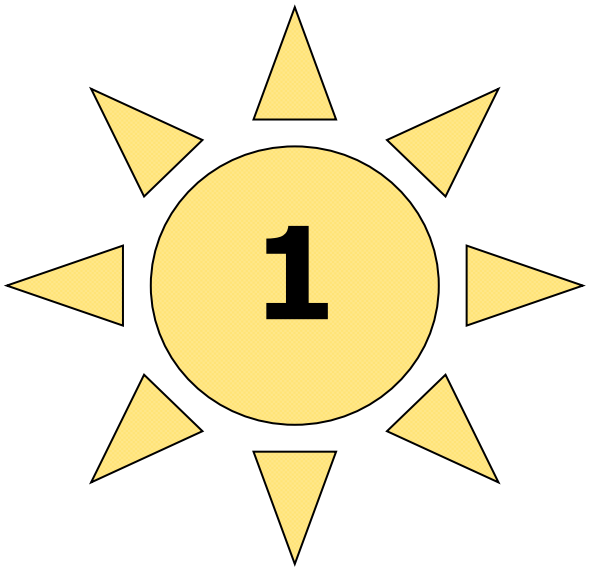
Capítulo 7: Pruebas realizadas al sistema de asignación

En este capítulo se mencionan las diferentes pruebas realizadas al sistema de asignación, también se presentan diferentes tablas con los resultados de las pruebas formales realizadas al sistema de asignación, a pesar de que son solo dos tipos diferentes de pruebas las presentadas en el capítulo, los resultados obtenidos de ellas arrojan datos muy interesantes.

Capítulo 8: Conclusiones y recomendaciones

En este capítulo se presentan las conclusiones que se obtienen del presente trabajo de tesis, también se presentan las recomendaciones que se pudieran aplicar para versiones futuras del sistema.

Al final de cada capítulo se incluye la bibliografía consultada.



Capítulo 1: Antecedentes

1.1 Las primeras computadoras

Desde su primera aparición, la computadora ha resultado ser una herramienta capaz de reducir el tiempo empleado en la realización de tareas que originalmente eran llevadas a cabo por los seres humanos.

En la década de 1950 fueron diseñadas y ensambladas en las universidades e institutos de los Estados Unidos de Norte América (EU) las primeras computadoras electromecánicas, estas primeras máquinas eran muy grandes (en algunas ocasiones ocupaban toda la área del laboratorio en donde se encontraban) y fueron utilizadas principalmente para la elaboración de complejos cálculos que en su momento redujeron en gran medida el tiempo de espera de resultados así como también el número de errores en los cálculos atribuibles a los humanos.

Pero mucho tiempo antes de esto ya se había dado una clara muestra de cómo es que una máquina capaz de realizar cálculos correctos puede reducir de una manera considerable el tiempo de procesamiento de información y la espera de resultados. En 1890 Herman Hollerith (quien más tarde fundara la International Business Machines o IBM) desarrolló una máquina que ayudó a llevar a cabo el levantamiento del censo nacional de EU, permitiendo realizar el conteo de 7,000 tarjetas en un solo día en lugar de sólo 700, con lo cual se redujo el proceso de conteo del censo de diez años a sólo tres.

Junto a la aparición de las primeras computadoras, surgen los lenguajes de programación, los cuales le permiten al operador indicar la o las tareas que desea que la máquina lleve a cabo; esto se logra a través de programas, los primeros programas constaban de una secuencia de instrucciones compuestas por unos y ceros, esto debido a que, estas primeras computadoras sólo entendían este tipo de lenguaje de programación (lenguaje binario).

En los años posteriores a 1950 el uso de las computadoras se había extendido a más universidades, institutos y al ejercito de EU, pero aún no estaba al alcance de todos, unas cuantas compañías en EU fabricaban computadoras en serie como por ejemplo, la IBM con su System/360 de 1964 el cual consistía en una familia de seis computadoras mutuamente compatibles y cuarenta periféricos que podían trabajar juntos. Por otro lado, a mediados de la década de 1960 Digital Equipment Corporation introduce una máquina llamada PDP 8 la cual se convertiría en la primer minicomputadora exitosa comercialmente hablando. Al mismo tiempo, Hewlett-Packard introduce la computadora de propósito general para negocios HP-2115 que soportaba los lenguajes de programación ALGOL, BASIC y FORTRAN, mientras que NEC construye en Japón a principios de 1960 la primer computadora electrónica, la NEAC 1101.

Por su parte, los lenguajes de programación también sufrieron cambios, ahora ya se podían escribir programas utilizando mnemónicos y números arábigos en lugar de solo cadenas de unos y ceros, estos tipos de lenguajes son conocidos como lenguajes ensambladores. Y mejor aún, como ya se mencionó anteriormente, algunos de los lenguajes de programación conocidos como de alto nivel tales como el FORTRAN (o FORmula TRANslation, desarrollado por John Bakus en la década de 1950) y el BASIC (o Beginner's All-purpose Symbolic Instruction Code, desarrollado por John Kemely y Thomas Kurtz en el Colegio Dartmouth en la década de 1960) ya eran utilizados por computadoras de negocios, teniendo como característica importante la de ser muy parecidos a los lenguajes naturales (los lenguajes utilizados por los seres humanos para comunicarse, como el Inglés, el Francés y el Alemán) permitiendo a las personas que trabajaban en las empresas, y que no eran expertas en computación, escribir sus propios programas y en consecuencia resolver problemas

referentes a sus actividades laborales y cotidianas con el uso de las computadoras como una herramienta más de trabajo.

1.2 La computadora como herramienta de trabajo

La penetración de las computadoras a las oficinas de negocios trajo como consecuencia que las aplicaciones de las computadoras se diversificaran. Ya no se utilizaron sólo para la realización de complejos cálculos matemáticos en las universidades y en el ejército. Ahora eran utilizadas en una variedad de industrias públicas y privadas que, incluían industrias del gobierno, industrias dedicadas a la investigación, defensa, medicina, exploración espacial y los negocios, en esta última industria, en sus oficinas las computadoras se comenzaron a utilizar para crear programas de computadora que permitían archivar de manera electrónica la información referente a los estados de cuenta de la empresa, la información de la nómina, los datos personales y laborales de los empleados, o mejor aún, se desarrollaron programas de computadora que permitían llevar a cabo el control del inventario o la nomina, por citar algunos ejemplos.

Es en este punto en donde surge la necesidad de modelar (representar de una manera abstracta un objeto o un conjunto de objetos, como se hace con los mapas, los globos terráqueos y los dibujos arquitectónicos) muchas de las tareas de negocios para posteriormente implantarlas ya no como simples programas aislados sino como sistemas de procesamiento de información, entendiendo como sistema al grupo de elementos que interactúan regularmente formando un todo.

Estos primeros sistemas de información trabajaban con archivos, por lo cual en realidad eran sistemas de procesamiento de archivos. Dichos archivos eran independientes unos de otros, no existía una relación entre ellos, lo que permitió la existencia de un archivo con los datos de los empleados, un archivo con los datos de los clientes, un archivo con la información de los distribuidores, y así otros más. Esta manera de trabajar generó los siguientes problemas:

- Duplicidad e inconsistencia en la información, el sistema de nómina tenia su propia copia del archivo de empleados y el sistema de recursos humanos la suya, en teoría ambos archivos deberían de ser iguales pero esto no siempre se cumplía, al final los datos en ambos archivos no eran los mismos ya que en uno se hacían las actualizaciones de información y en el otro no.
- Dependencia de los programas respecto al formato en el cual estaban almacenados los datos y a la manera en como acceder a ellos, no es lo mismo acceder y almacenar los datos utilizando un programa escrito en BASIC que en FORTRAN.
- Al tener sistemas desarrollados en diferentes lenguajes de programación, resultaba difícil conjuntar la información almacenada en los diferentes archivos (debido principalmente al formato particular de cada archivo) para la generación de reportes o para la realización de un análisis que involucrara a toda la información.

Para atacar este problema, las empresas comenzaron a centralizar sus datos claves, tales como los datos de las ordenes de trabajo, los datos del inventario, los datos de los clientes y los datos referentes a los estados de cuenta que manejaban, surgiendo con esto las primeras bases de datos a nivel organización. A medida que estos sistemas eran cada vez más utilizados el volumen de información se incrementó considerablemente, aparecieron entonces, sistemas cada vez más complejos y que en la mayoría de los casos eran bastantes lentos debido a la gran cantidad de información que debían procesar.

En 1970, E.F. Codd publico "A Relational Model Of Data for Large Shared Systems" en el cual se aplicaban los conceptos de una clase de matemáticas llamadas álgebra relacional al problema de almacenar grandes cantidades de información, este documento dio el inicio a la definición de lo que más tarde seria el modelo relacional de bases de datos. Este modelo ofreció, entre otras cosas, minimizar la duplicidad de información, la posibilidad de que los programas de los sistemas fueran independientes del formato en el cual estuvieran almacenados los datos, así como también, almacenar la información de una manera homogénea (en tablas), bajo estas nuevas circunstancias, resultó más fácil conjuntar la información de las diferentes áreas de la compañía para generar reportes o para realizar un análisis a la información almacenada en su conjunto y permitir la toma de decisiones en el ámbito empresarial.

En este momento muchas de las empresas comenzaron a migrar sus sistemas de procesamientos de archivos a sistemas de procesamientos de bases de datos que estuvieran implantados siguiendo el modelo relacional. Hasta la fecha, éste tipo de base de datos es las más utilizada en los sistemas de procesamiento de bases de datos.

Los sistemas de procesamiento de bases de datos están formados por la misma base de datos (compuesta por archivos con la información organizada en tablas, los índices que permiten el rápido acceso a la información y el diccionario de datos que es la descripción de la estructura de las tablas). Otro componente es el Sistema Administrador de Base de Datos o DBMS (por sus siglas en inglés) que funciona como el medio de comunicación entre los programas de aplicación y la base de datos física, es decir, el DBMS trabaja directamente con las tablas y los registros almacenados normalmente en un disco duro, y finalmente están los programas de aplicación que hacen las peticiones directas al DBMS solicitando algunos de sus servicios de administración; agregar, modificar, eliminar o simplemente consultar la información.

En la actualidad se tienen computadoras llamadas servidores y del tipo personal o PC (por sus siglas en inglés) con el mismo poder de procesamiento que las grandes máquinas de las décadas de 1970 y 1980, con mejores sistemas administradores de bases de datos ya que ahora muchos de ellos cuentan con una herramienta para la realización de reportes, otra para la creación de pantallas que permiten la administración de la información, un lenguaje de programación integrado, todo dentro de un ambiente de desarrollo gráfico. Todo esto a permitido reducir el tiempo de desarrollo de los sistemas y ha permitido también que sean instalados en servidores y en otros casos, dependiendo del tipo y tamaño del sistema, en las PC's.

1.3 ANUIES

La Asociación Nacional de Universidades e Instituciones de Educación Superior (ANUIES) fue fundada en 1950, es una asociación no gubernamental, de carácter plural, que agrupa alrededor de 140 universidades e institutos de educación superior con regímenes tanto públicos como privados, cubriendo así el 80% de la matricula de alumnos que estudian una licenciatura o un posgrado.

La ANUIES esta compuesta por órganos colegiados y por su secretaria general ejecutiva. La Asamblea General es el órgano supremo de la asociación y esta compuesto por todas las instituciones afiliadas. El Consejo Nacional es el órgano colegiado de dirección y articulación de la asociación, integrado por 14 titulares de las instituciones afiliadas. La Secretaria General Ejecutiva es la instancia operativa representada por el Secretario General Ejecutivo.

Entre los principales objetivos de la ANUIES están:

- Aportar soluciones a los problemas de la educación superior y opciones para su desarrollo con calidad en los ámbitos nacional, regional y estatal.
- Promover proyectos y actividades interinstitucionales que propicien la convergencia de intereses de las instituciones asociadas en los ámbitos nacional, regional y estatal.

De acuerdo a sus objetivos antes mencionados, la ANUIES se ha manifestado a lo largo de su historia por la creación y la realización de un examen previo al ingreso a la educación media superior (examen que debería ser presentado por todo aquel aspirante a un lugar), teniendo un carácter externo, debiendo evaluar los resultados académicos y las habilidades más fundamentales obtenidas hasta la terminación de la educación secundaria.

En abril de 1993 la ANUIES emitió la recomendación de contar con una instancia externa que coadyuvara en las acciones de evaluación de las instituciones de educación superior, independientemente de las funciones que en esta materia realizan las propias instituciones y las autoridades académicas.

1.4 El CENEVAL® y el EXANI I

En respuesta a la recomendación realizada en abril de 1993 por la ANUIES, a principios de 1994 se crea el Centro Nacional de Evaluación para la Educación Superior, A. C. (CENEVAL®) por resolución de la Coordinación Nacional de Planeación de la Educación Superior (CONPES), con la finalidad de realizar los exámenes nacionales de ingreso (EXANI) y los exámenes generales de egreso (EGEL). En ese momento se dijo que ambos exámenes se desarrollarían bajo los principios de:

- Equidad: … el trato justo e igualitario para todos los sustentantes e instituciones usuarias.
- Independencia de criterio en los consejos técnicos: … la libertad de estos cuerpos colegiados para que los contenidos, estructura y composición de la prueba sean decididos sin presión externa al consejo ya sea por parte de las otras instancias del CENEVAL® o de cualquier otra institución o persona ajena.
- Transparencia de los procesos: … todos los procesos de elaboración, aplicación y calificación de las pruebas están abiertos a escrutinio. Los reactivos de las pruebas no pueden ser consultados por personas ajenas al Consejo Técnico y al personal del CENEVAL® encargado de la prueba.
- Confidencialidad : … las instituciones y los sustentantes tienen el derecho a que los resultados de las evaluaciones no sean revelados sino a los propios sustentantes y a las instituciones de origen y destino, según sea el caso.

Partiendo de los acuerdos de la Asamblea General de la ANUIES, se establecieron las metas del CENEVAL® para sus primeros cinco años de vida de acuerdo a tres líneas de acción, de una de ellas se desprende la elaboración y puesta en operación del examen de ingreso a la educación media superior o EXANI I.

1.5 El Concurso de Ingreso a la Educación Media Superior

A principios de la década de 1990 la demanda de educación media superior en la Ciudad de México y su Zona Metropolitana se enfrentó a una serie de problemas que las instituciones educativas involucradas no podían enfrentar de manera aislada (como se venía haciendo tradicionalmente), entre estos problemas estaban:

- El aumento de la población del Distrito Federal y los 21 municipios conurbados del Estado de México.
- Incertidumbre en cuanto a una cantidad aproximada de la demanda de educación superior.
- Incertidumbre respecto al número demandado por modalidad de estudio por institución y por plantel.
- Falta de información de los aspirantes a la educación media superior con respecto a todas las opciones de educación existentes.
- Debido a la diversidad de convocatorias emitidas (cada una de las instituciones de educación media superior emitía la suya), a los métodos de selección y a los diferentes exámenes, se generaba un ambiente en el cual se favorecía a aquellos aspirantes que podían pagar y en consecuencia participar en varios concursos de selección.

En respuesta a estos problemas el Colegio de Bachilleres (COLBACH), el Colegio Nacional de Educación Profesional Técnica (CONALEP), la Dirección General de Educación Tecnológica Agropecuaria (DGETA), la Dirección General de Educación Tecnológica Industrial (DGETI), la Dirección General de Bachillerato (DGB), el Instituto Politécnico Nacional (IPN), la Secretaria de Educación, Cultura y Bienestar Social del gobierno del Estado de México (SECyBS), la Universidad Autónoma del Estado de México (UAEM) y la Universidad Nacional Autónoma de México (UNAM) convocaron en conjunto en el mes de abril de 1996 al Concurso de Selección para la Educación Media Superior.

Este hecho sin precedentes en el Sistema Nacional de Educación, tuvo las peculiaridades de convocar, registrar y examinar a toda la demanda de educación media superior de la Ciudad de México y su Zona Metropolitana de manera única y conjunta. El instrumento de evaluación fue elaborado en el CENEVAL®, siguiendo las especificaciones y los criterios convenidos por las nueve instituciones participantes, utilizando el banco de reactivos del EXANI I, además de que los representantes de las instituciones formaban parte del Consejo Técnico del EXANI I en aquel entonces.

Con la realización de este primer concurso de selección se sentaron las bases para mejorar los procesos de planeación de la oferta y la demanda de la matrícula en el ámbito de la educación media superior. Se logró conocer la demanda total de educación pública de nivel medio superior con una antelación mayor a como se venía obteniendo. El conocer la demanda con mayor antelación permitió ampliar la oferta de lugares disponibles que se había anunciado al inicio del concurso de selección. Por primera vez los aspirantes tuvieron la información de 636 opciones de estudio ofrecidas por las nueve instituciones involucradas.

A partir de 1996 y hasta el presente año se ha venido realizando el concurso de selección, que actualmente tiene el nombre de concurso de ingreso, sufriendo principalmente a lo largo de estos años, algunos cambios en cuanto a las reglas del mismo. Por citar un ejemplo, para el concurso de ingreso del año 2000 la UNAM diseñó, aplicó y calificó su propio examen de ingreso a todos aquellos aspirantes que eligieron alguno de sus planteles como primera opción dentro de su formato de registro al examen, mientras que las otro ocho instituciones continuaron aplicando el EXANI I del CENEVAL®.

1.6 La Comipems

Las principales instituciones públicas que ofrecen educación media superior en la Zona Metropolitana de la Ciudad de México (ZMCM) se integraron en una comisión, a la cual le dieron

el nombre de Comisión Metropolitana de Instituciones Públicas de Educación Media Superior (Comipems), las instituciones integrantes son:

- Colegio de Bachilleres o COLBACH
- Colegio Nacional de Educación Profesional Técnica o CONALEP
- Dirección General de Educación Tecnológica Agropecuaria o DGETA
  Centro de Bachillerato Tecnológico Agropecuario o CBTA. Plantel Texcoco
- Dirección General de Educación Tecnológica Industrial o DGETI
  Centros de Bachillerato Tecnológico, Industrial y de Servicios o CBTIS
  Centros de Estudios Tecnológicos, Industrial y de Servicios o CETIS
- Dirección General de Bachillerato o DGB
  Centros de Estudios de Bachillerato o CEB
- IPN
  Centros de Estudios Científicos y Tecnológicos o CECYT
  Centro de Estudios Tecnológicos o CET
- SECyBS
  Centros de Bachillerato Tecnológico o CBT
  Colegio de Bachilleres del Estado de México o COBAEM
  Colegio de Estudios Científicos y Tecnológicos del Estado de México o CECYTEM
  Escuela Superior de Comercio
  Preparatorias Anexas a las Escuelas Normales
  Preparatorias Oficiales
- UAEM
  Escuela Preparatoria (plantel Texcoco)
- UNAM
  Colegio de Ciencias y Humanidades o CCH
  Escuela Nacional de Estudios Profesionales Iztacala (carrera Técnico en Enfermería)
  Escuela Nacional Preparatoria o ENP
  Facultad de Estudios Superiores Zaragoza (carrera Técnico en Enfermería)

1.7 Las funciones de la Comipems

Como ya se mencionó, las nueve instituciones integrantes de la Comipems firmaron un convenio de colaboración en febrero de 1996, del cual surgió la convocatoria al Concurso de Selección en el mes de abril del mismo año. Hasta el año 2002 son ya 7 años consecutivos en los cuales el Concurso de Ingreso (nombre actual del mismo) se ha realizado.

El papel que juega la Comipems dentro de la educación media superior de la ZMCM en los últimos 7 años es el de ofrecer un proceso académico - administrativo unificado, sencillo y transparente para permitir el acceso a la educación media superior a todos los aspirantes que cumplan con los diferentes requisitos de ingreso establecidos por la misma comisión.

Para llevar a cabo este proceso la Comipems realiza las tareas de convocar, registrar, examinar, asignar, emitir los resultados del concurso de ingreso y posteriormente atender a los aspirantes que no fueron asignados.

Por lo tanto, la Comipems debe informar con la mayor claridad posible las características de todas las opciones públicas de educación media superior en la ZMCM ofreciendo opciones de bachillerato general, bachillerato tecnológico o una carrera profesional técnica, debe simplificar los trámites y reducir los costos del proceso de selección, debe también asegurar la igualdad de condiciones en el proceso de selección, incrementar las opciones de colaboración y coordinación entre las autoridades y directores de las instituciones públicas de educación media superior de la ZMCM y finalmente debe presentar elementos objetivos que permitan mejorar los procesos de planeación de la oferta de educación media superior en la ZMCM.

1.8 Perspectivas a futuro

El concurso de ingreso se ha mantenido vigente gracias a que favorece tanto a los aspirantes como al sistema educativo. Los aspirantes son evaluados en igualdad de circunstancias además de que con un solo registro los aspirantes pueden solicitar inscripción hasta en nueve diferentes instituciones. El sistema educativo obtiene información que le permite identificar el volumen de la demanda (total y por institución) que junto con otros elementos le permite orientar el trabajo educativo que desempeñan las escuelas secundarias.

Por todo esto, el concurso de ingreso se ha establecido como una manera equitativa y transparente de obtener un lugar en el sistema de educación media superior, lo cual permite pensar que este mecanismo de ingreso continuará en los próximos años.

Bibliografía

1. **Internet: The Computer Museum History Center,** "Timeline of Computer History", http://www.computerhistory.org/timeline/

2. **Internet: Asociación Nacional de Universidades e Instituciones de Educación Superior,** "¿Qué es la ANUIES?", http://www.anuies.mx/1.html

3. **Rafael Vidal, Yolanda Leyva, Agustín Tristán y Felipe Martínez Rizo**, Manual Técnico del CENEVAL®, publicado por CENEVAL A. C.; México; 2000; páginas 7 – 12.

4. **Comisión Metropolitana de Instituciones Públicas de Educación Media Superior,** "Instructivo del Concurso de Selección 1996", México; 1996; páginas 1 – 11.

5. **Comisión Metropolitana de Instituciones Públicas de Educación Media Superior,** "Concurso de Selección para la Educación Media Superior 1996, Informe final", México; 1996; páginas 4 – 16.

Capítulo 2: Marco teórico

2.1 Introducción

Antes de iniciar la construcción de una casa o de un edificio es necesario realizar un análisis para definir qué clase de construcción que se desea (número de niveles, número de ventanas, número de accesos, etcétera); una vez que se tiene claro el tipo de construcción que se desea, hay que proceder a diseñar en papel aquello que previamente se ha definido, finalmente se deben de elegir los materiales y las herramientas adecuadas al tipo de construcción.

Al igual que la construcción de una casa o un edificio, la construcción de un sistema para computadora necesita de un análisis y de un diseño previo a su elaboración, pues no se puede iniciar la construcción de un sistema sin tener en claro qué tipo de problema se debe resolver y, más aún, sin tener una idea clara de cómo es que internamente estará compuesto (su estructura).

La construcción de cualquier producto debe cumplir con una serie de etapas para lograr su éxito como tal, dichas etapas deben ejecutarse siguiendo una secuencia previamente establecida, los sistemas para computadora no pueden ser la excepción, el análisis y diseño antes mencionados son dos de las etapas que forman parte del ciclo de desarrollo de los sistemas para computadora.

2.2 Sistemas

¿Qué debemos entender por sistema? Existen muchas definiciones al respecto, en la vida diaria utilizamos en diferentes contextos la palabra sistema (sistema de calefacción, sistema de transmisión automática, sistema de consulta de saldos, sistema financiero, sistema respiratorio, etcétera). Y es que casi todo con lo que tenemos contacto día a día es un sistema en sí mismo, el cual en muchas ocasiones forma parte de un sistema más grande.

El diccionario Webster define un sistema como "un conjunto o disposición de cosas relacionadas de manera que forman una unidad o un todo orgánico … un conjunto de hechos, principios, reglas, etc., clasificadas y dispuestas de manera ordenada mostrando un plan lógico de unión de las partes … un método o plan de clasificación o disposición … una manera establecida de hacer algo; método; procedimiento". En mi opinión, un sistema es un conjunto de partes o elementos que están ordenados de tal manera que permiten alcanzar un objetivo específico.

En el mundo siempre han existido los sistemas naturales, como por ejemplo los sistemas geológicos, ecológicos y moleculares, el hombre por su parte ha desarrollado sus propios sistemas, como por ejemplo los sistemas económicos, sociales y numéricos.

Ahora bien, dentro de los sistemas desarrollados por el hombre está el caso de los sistemas computarizados también conocidos como automatizados o basados en computadora, no presentaré una definición de este tipo de sistemas, sino que diré que un sistema automatizado o basado en computadora (SBC) es aquel que tiene:

- Hardware: elementos mecánicos, electromecánicos y electrónicos
- Software: grupo de instrucciones especiales para computadora
- Personas: operadores o usuarios del sistema
- Datos: información concerniente y necesaria para el sistema, accedida a través del software
- Procedimientos: instrucciones que indican cómo debe de utilizarse cada elemento del sistema

Todos estos elementos están dispuestos y trabajan en conjunto de tal manera que logran resolver un problema en particular mediante el procesamiento de información; además de esto, debo remarcar que los sistemas basados en computadora pueden o no involucrar en su operación a más de una computadora.

A continuación presento las diferentes categorías de sistemas basados en computadora.

2.2.1 Sistemas en línea

Son aquellos sistemas basados en computadora en donde los datos que entran al sistema son proporcionados directamente por la o las personas que los crearon (por ejemplo a través de una terminal local o remota) y la salida o resultado del sistema es entregado directamente a la persona o personas que los necesitan. Ahora bien la información en este tipo de sistemas esta almacenada de tal forma que rápidamente pueda ser localizada una pieza de información sin la necesidad de recorrer toda la información de principio a fin.

2.2.2 Sistemas de tiempo real

Estos sistemas basados en computadora pueden considerarse como una variante de los sistemas en línea, la diferencia radica en el tiempo de respuesta que debe ofrecer un sistema de tiempo real ya que este debe de responder en mili o en micro segundos a una orden de un usuario u operador, la respuesta sirve para controlar o bien influir en un ambiente dado (por ejemplo un sistema de vigilancia de pacientes o un sistema de control de procesos utilizado para verificar y controlar un proceso químico). Y es que los sistemas de tiempo real en la mayoría de los casos trabajan con personas y con un ambiente que casi siempre es autónomo. Si un sistema de tiempo real no responde a la velocidad adecuada al ambiente o al proceso que controla, esté puede quedar fuera de control (perdida de información o fuga de un líquido por ejemplo).

2.2.3 Sistemas de Información

El incremento de las computadoras de escritorio en las empresas que se inicio en la década de 1980 y que continua hasta el día de hoy ha creado una explosión en la cantidad de información, información que se genera por la naturaleza misma de las tareas diarias de la empresa y que además crece muy rápidamente por lo que en la mayoría de los casos resulta difícil de administrar.

Para atacar este crecimiento de información en un principio los programadores creaban programas aislados para llevar a cabo el procesamiento de la información de una manera local (por área, departamento o dirección), después fueron integrando estos programas creando así sistemas empresariales, estos fueron cada vez más complejos dando origen a una nueva clase de sistemas basados en computadora; los Sistemas de Procesamiento de Información o simplemente Sistemas de Información.

2.2.4 Jerarquía de los Sistemas de Información

Sistemas operacionales

Como ya he mencionado, en respuesta al crecimiento de información se han desarrollado (los primeros en la década de 1960) muchos sistemas operacionales o de procesamiento de transacciones los cuales tienen la tarea de ayudar a realizar de una manera automatizada (dentro de lo posible) las

tareas diarias del negocio, estamos hablando aquí de los sistemas de nómina y de los sistemas de contabilidad por citar solo dos ejemplos de este tipo de sistema.

Sistemas de apoyo a la toma de decisiones

Este tipo de sistema ayuda tanto a los administradores como a los "trabajadores del conocimiento" a tomar decisiones inteligentes y documentadas referentes a la manera en que opera la organización ya que recuperan, presentan y analizan los datos en la mayoría de los casos de una forma gráfica. Hay que recalcar que estos sistemas no toman decisiones por sí mismos, pero si realizan análisis matemáticos y estadísticos contribuyendo a la formulación de una decisión de negocios.

Sistemas de planeación estratégica

Los sistemas de planeación estratégica "no son programas de computadora en sí, son complejas combinaciones de actividades y procedimientos, muchas de las cuales las llevan a cabo humanos utilizando información obtenida de fuentes externas y de datos internos de los sistemas operacionales de la organización y de los sistemas de apoyo a la toma de decisiones".

Estos tipos de sistemas ofrecen consejos acerca de la naturaleza del mercado en el que se desenvuelve la empresa, como por ejemplo, las preferencias de los consumidores. Son utilizados generalmente por los gerentes para evaluar y analizar la misión de la organización.

La figura 2.1 muestra la jerarquía de los sistemas de información.

Figura 2.1 Jerarquía de los sistemas de información

2.3 Ingeniería de Sistemas de Computadora, Ingeniería del Software e Ingeniería de Información

Una vez que se ha hablado sobre los diferentes tipos de sistemas es necesario preguntar ¿qué ciencia se encarga del estudio de los sistemas basados en computadora? Para responder a esta pregunta me apoyaré en la definición de Roger S. Pressman referente a la Ingeniería de Sistemas de Computadora (ISC) que dice:

"En lugar de solo concentrarse únicamente en el software, la ingeniería de sistemas de computadora se concentra en una variedad de elementos, analizando, diseñando y organizando esos elementos en un sistema que puede ser un producto, un servicio o una tecnología para la transformación de información o control de información".

En la definición anterior está la respuesta a la pregunta antes realizada, la ciencia encargada del estudio de los sistemas basados en computadora es la ISC. Debido a que la ISC analiza, diseña y organiza los elementos del sistema, la ISC es en sí misma un proceso de modelado.

La ISC crea modelos para:

- Definir las tareas que son necesarias para satisfacer las necesidades del sistema.
- Representar el comportamiento del sistema y los supuestos en los que se basa dicho comportamiento.

Si la ISC modela el sistema, entonces ¿quién desarrolla propiamente el sistema? La persona encargada de hacerlo es el Ingeniero de Software o como en nuestro país lo llamamos el Ingeniero de Sistemas apoyándose en la Ingeniería de Software (ISW), por lo tanto se dice que la ISW ocurre como una consecuencia de la ISC.

Pero ¿qué es la ISW? El Instituto de Ingenieros Eléctricos y en Electrónica (IEEE por sus siglas en inglés) en su IEEE93 la define como:

"La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software, es decir, la aplicación de la ingeniería del software".

Como la definición anterior hay muchas más, en mi opinión la ISW es el conjunto de conocimientos y técnicas de la ingeniería utilizadas en el análisis, diseño, construcción, pruebas y mantenimiento del software, ofreciendo métodos que indican cómo construir técnicamente el software, dichos métodos incluyen técnicas de análisis de requisitos, el modelado del sistema y de su almacén de datos por ejemplo. Otro punto importante de la ISW radica en que se propone crear software de calidad dentro del tiempo y del presupuesto establecidos para el sistema.

La figura 2.2 presenta la relación entre la ISC, los sistemas basados en computadora y la ISW.

Cuando el contexto de trabajo de la ISW es una empresa, recibe el nombre de Ingeniería de la Información (II) e intenta poner orden al desarrollo de sistemas basados en computadora colocando al software en su contexto, ya que le asigna un papel especifico para entonces establecer los enlaces con los otros elementos del SBC.

El objetivo de la II es el de definir la arquitectura de datos (la base de datos), de aplicaciones (sistema de programas, software) y la infraestructura de la tecnología (software y hardware

utilizados para dar soporte a las arquitecturas de datos y de aplicaciones) que permitan a las empresas utilizar la información eficazmente, trabajando en un plan para la implantación de dichas arquitecturas. Globalmente la II analiza los objetivos de la empresa, identifica las áreas involucradas en esos objetivos para entonces definir las necesidades de información de cada una de las áreas y del negocio, una vez realizado todo esto la II llega al nivel de los procesos y es en este nivel donde son creados los sistemas de información.

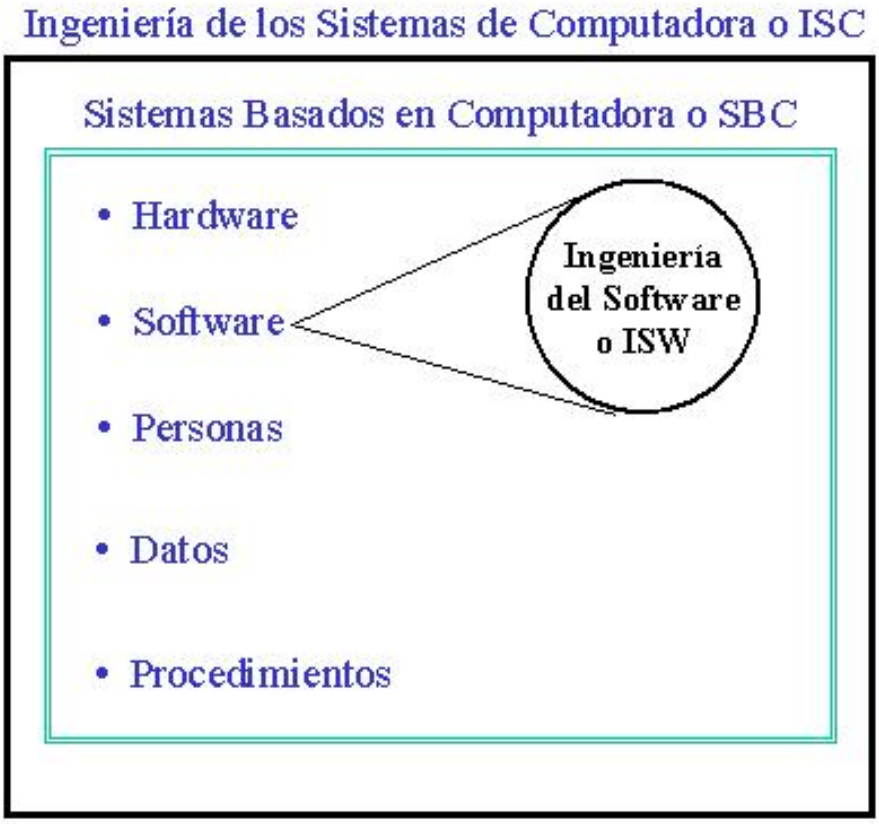


Figura 2.2 La ISC modela los elementos de un SBC, mientras que la ISW, se preocupa específicamente por crear software.

Cuando en el contexto de la ISW se requiere construir un producto para satisfacer las necesidades de un cliente o de un conjunto de capacidades definidas, la ISW recibe el nombre de Ingeniería del Producto (IP), dicho producto puede ser un sistema de comunicación, un sistema de mediciones eléctricas o un proceso único como lo es el Proceso de Asignación.

Al igual que la II la IP se encarga de crear software de calidad empleando métodos de ingeniería en todas sus etapas de desarrollo, poniendo también al software en su contexto y de igual manera estableciendo los enlaces con los demás elementos del SBC.

En la IP primero se deben de definir el o los objetivos y la o las metas del sistema en función de los requerimientos del cliente, después se modelan dichos requerimientos para entonces asignarles recursos de software, hardware, datos y personas.

2.4 Ciclo de vida de un proyecto

Como ya lo mencioné, en la introducción de este capítulo, antes de escribir la primera línea de código de cualquier software que forma parte de un SBC hay que realizar tanto un proceso de análisis como uno de diseño, no quiero decir con esto que sólo baste con la realización de estas dos actividades para crear software de calidad, en realidad ambos procesos (análisis y diseño) sólo son el inicio de lo que se conoce como el ciclo de vida del desarrollo de sistemas (SDLC por sus siglas en inglés).

El SDLC describe las etapas por las cuales un sistema de información debe pasar, es decir, índica o especifica las actividades que tanto el equipo de desarrollo del sistema como el usuario o usuarios finales deben de realizar. Debo aclarar que el usuario o usuarios finales solo participan en ciertas etapas del SDLC aportando sus necesidades y comentarios respecto a los procesos que debe de realizar el sistema en construcción.

A lo largo del tiempo han surgido diferentes ciclos de vida de desarrollo de software, tales como el ciclo de vida en cascada, el ciclo de vida estructurado, el ciclo de vida de la Ingeniería de Información y el propuesto por el IEEE conocido como el estándar IEEE/EIA 12270.

Los equipos de desarrollo de sistemas no se han puesto totalmente de acuerdo en cuántas y cuáles son las etapas o fases que deben de formar parte de los ciclos de vida en cascada y estructurado. Mientras que en el ciclo de vida de la Ingeniería de Información y en el propuesto por el IEEE parece que no hay tantas variaciones entre los estudiosos de la materia.

A continuación presento de una manera general cada uno de los SDLC antes mencionados.

2.4.1 Ciclo de vida clásico o en cascada

Probablemente el SDLC más antiguo es el conocido como “en cascada”. Se piensa que mucha de su filosofía tiene sus orígenes en las líneas de montaje de las industrias de manufactura. De acuerdo a Edward Yourdon las fases del SDLC en cascada son:

- Requerimientos del sistema
- Requerimientos del software
- Análisis
- Diseño del programa
- Codificación
- Pruebas
- Operaciones

Este SDLC recibe su nombre de cascada debido a su marcada progresión lineal y secuencial de sus fases ya que cada fase debe ser completada antes de que la siguiente se inicie; así por ejemplo, antes de iniciar la fase de análisis debe haberse terminado completamente la fase de requerimientos del software. Otra característica del ciclo de vida en cascada es su visión de desarrollo ascendente iniciando con los requerimientos del sistema y avanzando “hacia arriba” pasando por las fases de codificación para culminar con las operaciones del sistema.

Este SDLC ofrece las siguientes ventajas:

- Fácil administración: ofrece una descripción clara de en qué momento del ciclo de vida se encuentra el proyecto (fase de análisis o de diseño por ejemplo).
- Los límites o alcances de cada tarea son muy claros.
- Es muy ordenado el proceso de desarrollo.
- Permite la especialización de los elementos del equipo de desarrollo: dado que cada una de las fases están bien delimitadas, un individuo del equipo de desarrollo puede especializarse en alguna en especial, permitiendo que al finalizar su fase él se pueda integrar a otro proyecto.
- Si una fase está completamente terminada, en el mejor de los casos en lo sucesivo no se le debe de invertir más tiempo del proyecto.

Como desventajas tenemos:

- Si un error de análisis es detectado en las últimas fases del ciclo de vida, hay que regresar a la fase de análisis para realizar los ajustes necesarios, esto tiene un fuerte impacto en los costos del proyecto ya que detectar un error en las ultimas fases del ciclo de vida puede tener un impacto de hasta 25 veces más que si se detecta en las primeras.
- Nada está terminado hasta que todas las fases han sido concluidas, por lo que en la mayoría de los casos los usuarios finales tienen la sensación de que no hay avances en el proyecto y es que si a la mitad del SDLC el usuario desea ver algunas pruebas con datos reales, es muy poco probable que estas puedan llevarse a cabo.
- Cuando un proyecto es muy largo o complejo su misma naturaleza impide la descomposición por fases, el ciclo de vida en cascada no se puede ajustar de la mejor manera a este tipo de proyectos.

2.4.2 Ciclo de vida estructurado

En el ciclo de vida estructurado se habla de actividades y no de fases, se incluyen los elementos "terminadores" los cuales son los usuarios, los administradores del proyecto y el personal de operaciones del sistema quienes en su conjunto ofrecen las entradas al sistema y también son los principales beneficiarios del sistema.

Las actividades del ciclo de vida estructurado según Kenneth y Julie Kendall son:

- Identificación de problemas, oportunidades y objetivos
- Determinación de los requerimientos de información
- Análisis de las necesidades del sistema
- Diseño del sistema recomendado
- Desarrollo y documentación del software
- Prueba y mantenimiento del sistema
- Implantación y evaluación del sistema

El orden en que son listadas las actividades no establece a su vez el orden secuencial en el que se deben de realizar; es decir, a diferencia del SDLC en cascada, en donde se llevan a cabo en pasos separados las fases, aquí varias actividades pueden suceder simultáneamente además de que pueden ser repetidas tantas veces como sea necesario. Por lo tanto, las actividades pueden estarse traslapando y después disminuyendo en más de un momento del SDLC. Me parece que un aspecto muy importante de este SDLC es su naturaleza cíclica que es aplicada en todo momento, ya que si se descubre un error en cierta actividad el equipo de desarrollo podría estar obligado a regresar a la

fase que lo originó si es el caso y realizar las correcciones, lo que permite reducir los costos de los errores y permite crear software de mayor calidad.

Ahora bien, esto permite que exista tanto un enfoque radical como otro conservador. En un enfoque radical las siete actividades se llevan a cabo paralelamente desde el inicio del proyecto, mientras que en el enfoque conservador, la actividad número cinco debe de terminarse completamente antes de comenzar con la actividad número seis (lo que a mi parecer, en el peor de los casos nos lleva a un SDLC en cascada solo que con diferentes actividades).

Surge la pregunta ¿qué enfoque es el que se debe de utilizar? En general, se sugiere el enfoque radical para aquellos proyectos en donde el usuario final no esta muy seguro de lo que desea que el nuevo sistema realice, para cuando se ha especificado que para cierta fecha "algo" ya tiene que estar funcionando o bien para los casos en los cuales el usuario desea que sus requerimientos estén sujetos a posibles cambios. Por otro lado, se sugiere el enfoque conservador para aquellos proyectos grandes de una importancia trascendental en los cuales se requiere de un análisis y de un diseño muy detallados para así evitar en la medida de lo posible errores posteriores.

Otra característica importante de este SDLC es que puede ser aplicado a casi cualquier tipo de sistema, desde proyectos muy grandes hasta los más pequeños y que las actividades de análisis y diseño son independientes del lenguaje de programación utilizado en el desarrollo del software.

Por último, quiero mencionar que éste es el método del SDLC que se comenzó a utilizar en la década de 1970 y que ha logrado mantenerse hasta el día de hoy como una de las mejores alternativas de desarrollo de sistemas, así como también que es el método que, en mi caso particular, me fue enseñado durante el estudio de mi licenciatura.

2.4.3 Ciclo de vida de la Ingeniería de Información

Como ya lo mencioné, cuando el contexto de trabajo de la Ingeniería de Software o ISW es una empresa recibe el nombre de Ingeniería de la Información o II. El SDLC de la II es una metodología que utiliza técnicas para la planeación, el diseño, y la construcción de sistemas de información dentro de una empresa.

La primicia de este enfoque es: "los datos son el centro de las organizaciones modernas de negocios". De tal manera que, primero se debe definir el modelo de datos que la empresa requiere para soportar el plan estratégico de la misma, los objetivos inmediatos y las metas a futuro; para entonces definir los sistemas de información requeridos para implantar el plan antes mencionado.

La empresa es modelada como un conjunto de áreas de negocios en lugar de sistemas aislados, el SDLC de la II ve el desarrollo de sistemas en términos de tres componentes:

- Los datos de negocios
- Las actividades de negocios
- La interacción entre las dos anteriores

De acuerdo al SDLC Handbook[1] las fases del SDLC de la II son:

- Planeación de la estrategia de información

[1] SDLC handbook, HB 500-07 October 1998

- Análisis del área de negocios
- Diseño del sistema de negocios
- Diseño técnico
- Programación o construcción
- Aceptación
- Transición o instalación
- Operación
- Evaluación

A diferencia del enfoque ascendente del SDLC en cascada, este enfoque es de arriba hacia abajo, puesto que se inicia desde el nivel más alto (toda la empresa) para después descender al área de negocios detectada y entonces ahí desarrollar el sistema o sistemas que son necesarios.

Aquí son frecuentemente utilizadas, para soportar las características de este ciclo de vida, más de una metodología de la II, así como también diferentes herramientas del tipo Software Asistido por Computadora (CASE por sus siglas en inglés). Como metodologías de ejemplo están: "Yourdon y DeMarco" o "James Martin", mientras que como herramientas CASE: "Composer de Texas Instruments" o "Ingeniería de Sistemas de Lermonth and Burchett Management Systems".

En mi opinión, este tipo de SDLC es la respuesta a la nueva era en la cual los negocios o empresas se han percatado de que la información que se genera en ellos debe de ser vista y tratada como un recurso, el cual a su vez debe de usarse de una manera estratégica para lograr una mejor competitividad.

2.4.4 Ciclo de vida IEEE/EIA 12270

La Organización Internacional para la Estandarización (ISO por sus siglas en inglés) en su especificación 12207 de 1995 conocida como ISO/IEC 12270 detalla los procesos del SDLC partiendo del hecho de la firma de un contrato en donde se define el desarrollo, el mantenimiento, y la operación de un sistema de software, por lo que este ciclo no aplica a la compra de software comercial. Como punto importante de este estándar se tiene el reconocimiento de los distintos papeles de un comprador y de un vendedor de software. Por lo que en mi opinión, este SDLC está dirigido al desarrollo de software de propósito especifico.

La Asociación de la Industria de la Electrónica (EIA por sus siglas en inglés) y el IEEE en colaboración con el Departamento de Defensa o DoD (por sus siglas en inglés) de los Estados Unidos de Norte América (USA por sus siglas en inglés) trabajaron en el desarrollo de un estándar para el SDLC para su uso en USA. Los principales objetivos de este trabajo fueron:

- Definir estándares que representen la mejor práctica comercial
- Crear estándares que sean adecuados a la aplicación de los complejos requerimientos de las adquisiciones que realiza el DoD
- El estándar debe ser compatible con el mercado emergente global del software

Como resultado de este trabajo en equipo se tiene el estándar IEEE/EIA 12207 publicado en marzo de 1998, dicho estándar esta compuesto de tres volúmenes:

- IEEE/EIA 12207.0 – ISO 12207
- IEEE/EIA 12207.1

- IEEE/EIA 12207.2

El IEEE/EIA 12207 es el estándar estratégico, el cual es en sí una adaptación del estándar ISO/IEC 12270, que presenta las bases para una amplia adopción de los procesos de software en las organizaciones y que puede ser utilizado en proyectos tanto comerciales como los del DoD permitiendo a los clientes domésticos y comerciales apoyarse en él.

El estándar IEEE/EIA 12207 ofrece algunas ventajas importantes sobre los estándares existentes:

- Cubre el ciclo de vida de una manera completa, en lugar de solo tratar los procesos de desarrollo.
- Ofrece un enfoque más flexible para grabar los datos de los procesos y de los productos con herramientas CASE en contraste con la dependencia tradicional de documentos en papel.
- Incorpora referencias a otros estándares de USA que pueden ser muy útiles en los procesos del SDLC al momento de ponerlos en practica.
- Es compatible con la especificación ISO 9000 referente a sistemas de calidad, calidad en la administración, y garantía de calidad.

Me parece de una gran importancia la aparición de este estándar en el continente americano ya que los Estados Unidos de Norteamérica tienen un peso muy importante en la industria del desarrollo del software mundial gracias a compañías como Microsoft, éste echo puede provocar la adopción y difusión del estándar, teniendo como consecuencia un cambio en la metodología del desarrollo del software en los próximos años.

2.4.5 Ciclo de vida orientado a objetos

Antes de hablar sobre el ciclo de vida orientado a objetos me gustaría hablar acerca de lo que se conoce como las tecnologías orientadas a objetos, dichas tecnologías son un enfoque diferente para la construcción de software.

El termino “tecnologías orientadas a objetos” se usa para hacer referencia a diferentes metodologías de análisis, diseño, lenguajes de programación, bases de datos y herramientas orientadas a objetos (generadores de código, herramientas de documentación, herramientas CASE orientadas a objetos, etcétera).

Se dice que las tecnologías orientadas a objetos a largo plazo resolverán algunos de los problemas clásicos y nuevos del desarrollo y del mantenimiento del software (por ejemplo: la creciente complejidad de los nuevos sistemas para computadora). También se dice que muchos ingenieros de sistemas han mejorado substancialmente su desempeño al crear sistemas utilizando las tecnologías orientadas a objetos (aunque no siempre hayan alcanzado todos los beneficios que esperaban).
En este sentido opino que, no es suficiente con programar utilizando un lenguaje de programación 100 % orientado a objetos para obtener todos los beneficios de las tecnologías orientadas a objetos, y es que, al igual que cuando utilizamos programación estructurada nos apoyamos en el análisis y diseño estructurado, al crear sistemas orientados a objetos debemos de apoyarnos en análisis y en el diseño orientados a objetos, así como también, diseñar bases de datos orientadas a objetos[2].

[2] Database Processing, David M Kroenke

Aunque el enfoque orientado a objetos tuvo sus inicios en la década de 1960, en un principio me parece que se aplicó y se habló de él sólo en el campo de la programación orientada a objetos, puesto que, sólo se hablaba de los lenguajes de programación que soportaban algunos de los conceptos sobre los cuales subyacen las tecnologías orientadas a objetos (objetos y clases, métodos, solicitudes, herencia y encapsulado). Entre los lenguajes de programación orientada a objetos clásicos y nuevos están Smalltalk, C++, Ada 95, recientemente tenemos el famoso Java de Sun Microsystems y el nuevo C# de Microsoft.

Ahora bien, el enfoque orientado a objetos ofrece:

- La posibilidad de volver a utilizar una mayor cantidad de código reduciendo el tiempo de desarrollo de nuevos sistemas
- Reducción de los efectos colaterales al realizar modificaciones
- Ocultación de los procedimientos entre los diferentes objetos existentes en un programa
- Creación de sistemas que resuelven de una mejor manera los problemas de negocios para los que fueron diseñados.

En la década de 1990 la ingeniería de software orientada a objetos tomo gran importancia entre los ingenieros de sistemas de todo el mundo, me parece que aunque llego con una gran fuerza no ha sido la solución a todo, ya que, no todos los ingenieros de sistemas aplican todas las técnicas orientadas a objetos además de los costos que implica en cambiar el código existente, los equipos, etcétera.
También debo mencionar que, existen muchas técnicas de análisis y de diseño[3] orientadas a objetos y que estás no son totalmente independientes del lenguaje de programación que se desea utilizar en la etapa de programación, por lo que creo que estas técnicas están en evolución y tal vez en los próximos años sean la solución que toda la industria del software está buscando.

A continuación presento de una manera general algunos de los conceptos básicos de la programación orientada a objetos:

Objeto : Es un conjunto de datos conocidos como los atributos del objeto, acompañado por las operaciones que se pueden aplicar al objeto (conocidas también como métodos).

Clase: Una definición de un objeto que describe como es que internamente está estructurado el objeto, es decir, una clase es la implantación de un tipo de objeto.

Herencia: Es una relación entre clases en donde una clase comparte su estructura con una (herencia simple) o con varias (herencia múltiple) clases. Esta relación genera una estructura jerárquica de "clase padre" o "clase superior" con una "clase hija" o "subclase". De esta forma, cada clase tiene su propia estructura más la estructura y las características que heredó de todas las clases que se encuentran arriba de ella en lo que también se conoce como "jerarquía de clases".

Encapsulado: La manera en la cual un objeto expone sus datos es a través de sus propios métodos, de esta manera un objeto puede "esconder" o "encapsular" sus datos de los demás ocultando así su información, es decir, "el encapsulado es el resultado (o acto) de ocultar los detalles de implantación de un objeto respecto de su usuario"[4].

---

[3] Software Tecnology Review, Cornegie Mellon Software Engineering Institute
[4] Análisis y Diseño Orientado a Objetos, James Martin James J Odell

Hasta este momento sólo he mencionado las técnicas orientadas a objetos y algunas de sus características, pero ¿cuál es el ciclo de vida orientado a objetos?

En la mayoría de la literatura orientada a objetos se habla siempre de análisis, de diseño y de programación orientada a objetos como las etapas del SDLC orientado a objetos, muy poco se menciona sí estás son todas las tareas que un ingeniero de sistemas debe de realizar para crear software cuando utiliza un enfoque orientado a objetos, sin embargo, un desarrollo orientado a objetos combina las tecnologías orientadas a objetos con la ingeniería de software en lo que se conoce como la Ingeniería de la Información Orientada a Objetos[4].

Por todo lo anterior, considero que el SDLC orientado a objetos es muy similar al SDLC de la ingeniería de información.

2.5 Elección del SDLC para el Proceso de Asignación

En función de lo que he presentado anteriormente en todo este capítulo, tomo la decisión de adoptar el SDLC estructurado para el desarrollo del Proceso de Asignación ya que como lo mencioné en su momento, es la metodología aprendida en la licenciatura, también es la utilizada en el departamento de sistemas donde desarrollaremos el sistema, además de que se ajusta al tipo de sistema que se desarrollará (de propósito especifico), pues no se trata de un sistema que se utilizará en los procesos diarios de la empresa.

Bibliografía

1. **Edward Yourdon,** Análisis Estructurado Moderno**,** editorial Prentice Hall Hispanoamericana, S. A.; México Edo. De México; 1993; capítulos 2 y 5.

2. **Roger S. Pressman,** Ingeniería del Software Un Enfoque Práctico, editorial McGraw-Hill; México D.F.; 1998; páginas 157 – 170.

3. **Kenneth E. Kendall y Julie E. Kendall,** Análisis y Diseño de Sistemas, editorial Prentice Hall Hispanoamericana, S. A.; México Edo. De México; 1997; Capítulo 1.

4. **James Martin y James J. Odell,** Análisis y Diseño Orientado a Objetos, editorial Prentice Hall Hispanoamericana, S. A.; México Edo. De México; 1994; páginas 17 – 21.

5. **David M. Kroenke,** Database Processing Fundamentals, Design & Implementation, editorial Prentice Hall; USA New Jersey; 2000; páginas 481 – 483.

6. **Internet: US Customs Service,** SDLC Handbook HB 500-07**,** october 1998, http://www.customs.ustreas.gov/contract/modern/sdlcpdfs/tocsdlc.htm

7. **Internet: Carnegie Mellon Software Engineering Institute**, Object-Oriented Analisys Software Technology Review, http://www.sei.cmu.edu/str/descriptions/

8. **Internet: Software Productivity Consortium**, IEEE/EIA 12207, http://www.software.org/quagmire/descriptions/us-12007.asp

9. **Internet: Joe More,** ISO 12207 and Related software Life-Cycle Standards, http://www.acm.org/tsc/lifecycle.html

10. **James W. Moore, Perry R. DeWeese y Dennis Rilling,** U.S. Software Lifecycle Process Standards, http://www.stsc.hill.af.mil/crosstalk/1997/jul/lifec

Capítulo 3: El concurso de ingreso

3.1 ¿Qué es el Concurso de Ingreso?

El Concurso de Ingreso a la Educación Media Superior (CIEMS) de la Zona Metropolitana de la Ciudad de México (ZMCM) tiene como objetivo: "identificar a los aspirantes que puedan ingresar a alguna de las opciones educativas que ofrecen las instituciones convocantes, tomando como base los conocimientos básicos y habilidades generales que se miden mediante un examen, así como las preferencias de cada participante por ciertas opciones educativas"[1].

Hay que señalar que dicho examen permite conocer el nivel de preparación académica de cada aspirante y que las instituciones convocantes son aquellas que forman parte de la Comipems.

El CIEMS está dirigido a todos los interesados en obtener un lugar en la educación media superior y que además "ya terminaron su educación secundaria o están cursando el tercer año de secundaria y pueden obtener el certificado correspondiente antes del 17 de julio de 2001"[1]. Hay que aclarar que, el concurso de ingreso "Es el procedimiento en el que debe tomar parte cada persona que desea inscribirse en alguna de las opciones educativas que ofrecen las instituciones que conforman la Comipems"[1].

La Comipems mediante el CIEMS ofrece tres opciones de educación media superior:

a) Educación profesional técnica, se ofrece una educación de carácter especializado en un gran número de carreras o profesiones de nivel medio superior, al terminar el egresado obtiene título y cédula profesional emitidos por la Dirección General de Profesiones (DGP) de la SEP.
b) Bachillerato general, se ofrece una educación de carácter general en diversas disciplinas, tales como matemáticas, español, física o biología por mencionar solo algunas de ellas, aunque algunas instituciones ofrecen capacitación de carácter técnico, este tipo de bachillerato prepara a sus estudiantes para iniciar la educación superior y al terminar el egresado obtiene certificado de bachillerato.
c) Bachillerato tecnológico, se ofrece una modalidad bivalente en donde se puede estudiar una carrera de técnico y un bachillerato al mismo tiempo, al igual que el bachillerato general se prepara al estudiante para iniciar la educación superior o bien el estudiante puede integrarse al campo laborar al terminar los estudios de este tipo de bachillerato, el egresado recibe certificado de bachillerato y la carta de pasante; posteriormente el egresado elige una opción de titulación para obtener su título y su cédula de la especialidad elegida, ambas registradas ante la DGP de la SEP.

Ninguna de las tres opciones educativas antes mencionadas es necesariamente terminal, ya que; "quienes cursan una carrera profesional técnica siempre tienen alguna forma de hacer los estudios complementarios para obtener el certificado de bachillerato"[1], lo que les permite posteriormente iniciar sus estudios de licenciatura.

3.2 Etapas del CIEMS

Las principales etapas del CIEMS son las siguientes:

I. Publicación de la convocatoria
II. Registro al concurso
III. Presentación del examen

[1] Instructivo del CIEMS 2001

IV. Procesamiento de los datos, calificación de los exámenes y asignación de los aspirantes
V. Publicación de los resultados

El presente trabajo de tesis se enfoca principalmente en la etapa número IV, en particular en el proceso conocido como Asignación.

A continuación se describen de una manera general las etapas previas y posteriores a la número IV y sólo esta etapa se describe de una manera más amplia.

3.2.1 Publicación de la convocatoria

El 25 de febrero de 2001 la Comipems publica en las escuelas secundarias y en los principales periódicos de circulación de la ZMCM la convocatoria al CIEMS 2001.

Con la publicación de la convocatoria se da inicio formalmente al CIEMS, la convocatoria le permite a los interesados en un lugar en la educación media superior el conocer todos y cada uno de los procedimientos que se deberán de seguir durante el mismo ya que: "La convocatoria es el marco legal y normativo del concurso y sus procedimientos"[1].

En la convocatoria se establecen las bases sobre las cuales se realizará el concurso, de las cuales las más importantes son:

a) La información indispensable que cada sustentante debe conocer para participar en el concurso.
b) Las fechas y los lugares en los cuales se realizarán las diferentes etapas del concurso según la Comipems.
c) Las obligaciones de la Comipems hacia los sustentantes y a su vez las de los mismos sustentantes durante las diferentes etapas del concurso.
d) Los instrumentos de evaluación y las reglas para la asignación de los sustentantes.

Hay que hacer notar que todo lo que no quede previsto en la convocatoria será analizado y resuelto única y exclusivamente por la Comipems.

3.2.2 Registro al concurso

A principios del mes de marzo de 2001 la Comipems distribuyó en las escuelas secundarias de la ZMCM los materiales básicos para participar en el CIEMS:

a) Ficha de depósito bancario
b) Hoja de datos generales
c) Instructivo
d) Solicitud de registro
e) Comprobante de pre-registro*

Hay que mencionar que los estudiantes foráneos (quienes estudian en una secundaria pública o privada fuera de la ZMCM), los egresados (quienes obtuvieron su certificado de secundaria antes del 28 de febrero de 2001), los provenientes del Instituto Nacional de Educación para los Adultos (INEA) y los provenientes de escuelas incorporadas a la UNAM, recibieron la documentación antes

* Solo a los alumnos de escuelas secundarias incorporadas a la UNAM, los aspirantes foráneos, los egresados y los provenientes del INEA.

mencionada en el **centro de registro 01 o Colegio de Bachilleres Plantel 02 - Cien Metros**, en el periodo comprendido entre el 28 de febrero y el 12 de marzo de 2001.

De acuerdo a las fechas y al horario establecido en la convocatoria del concurso, los aspirantes deben acudir personalmente al centro de registro que les corresponde para realizar formalmente su registro.

Los aspirantes deben presentar:

- Solicitud de registro debidamente llenada
- Hoja de datos generales debidamente llenada
- Copia de la ficha de depósito bancario, que conste que el pago fue realizado.

Los aspirantes egresados, los foráneos, los del INEA y los provenientes de escuelas incorporadas a la UNAM deben de registrarse en el **centro de registro 01**, para todos ellos es indispensable presentar su comprobante de pre-registro y, únicamente los aspirantes egresados deben presentar el original de su certificado de secundaria.

Una vez que cada uno de los aspirantes entregue la documentación antes descrita y que esta no contenga errores u omisiones, recibirá a cambio su comprobante-credencial el cual lo avala como aspirante del CIEMS, el comprobante-credencial es "El único documento oficial que sirve para comprobar que el aspirante ha sido registrado en el concurso de la Comipems"[1]. También recibirá en ese momento "La Guía del Examen".

Hay que resaltar que en el comprobante-credencial se encuentra un número de folio, formado por nueve dígitos, este número de folio es utilizado en las etapas posteriores del concurso para identificar a cada uno de los aspirantes.

3.2.3 Presentación del examen

La presentación del examen es el "24 de junio de 2001"[2].

Los aspirantes se presentan al examen en el lugar y la hora señalados en su comprobante-credencial. Al igual que en el concurso anterior, la UNAM aplica "su propio examen de ingreso a todos los aspirantes que elijan alguno de sus planteles en la primera opción dentro de su formato de registro. Las otro ocho instituciones seguirán aplicando el examen del CENEVAL® (EXANI I). Los exámenes serán diferentes, pero equivalentes en su estructura, contenido, número de preguntas y grado de dificultad"[1]. El EXANI I "es un instrumento técnico de evaluación elaborado por el CENEVAL®. Se trata de un cuestionario de opción múltiple, constituido por 128 preguntas"[1] y "no es un examen de acreditación que tenga como finalidad aprobar o reprobar"[1]. Tanto el examen aplicado por la UNAM como el examen del CENEVAL® son de opción múltiple y miden "las habilidades y conocimientos básicos relacionados con el plan y los programas de estudio oficiales de educación secundaria"[2].

Es indispensable para poder realizar su examen, que cada uno de los aspirantes presente su comprobante-credencial.

El día del examen a cada uno de los aspirantes se le entrega:

[1] Instructivo del CIEMS 2001
[2] Convocatoria al CIEMS 2001

- Un cuadernillo con las 128 preguntas de las que esta compuesto el examen
- Una hoja de respuestas

El tiempo utilizado para resolver el examen es de 3 horas y no está permitido introducir al salón de examen ningún libro, cuaderno, calculadora u otro material o documento, solo se permite lápiz del número 2½, goma blanda y sacapuntas de bolsillo.

Al finalizar cada aspirante su examen, éste debe entregar todo el material que le fue proporcionado así como también deberá firmar su hoja de respuestas para asentar que está de acuerdo con lo respondido en su examen.

3.2.4 Procesamiento de los datos, calificación de los exámenes y asignación de los aspirantes

En el presente trabajo de tesis se desarrolla el proceso conocido como "Asignación", para que este proceso se lleve a cabo es necesario que primero se procese la diferente información generada por:

- La Comipems, durante el registro se genera la información de las escuelas u opciones educativas solicitadas por cada uno de los aspirantes, es decir, la demanda de educación media superior para el 2001.
- La Comipems, el número de lugares disponibles por opción educativa, es decir, la oferta educativa de educación media superior para el 2001.
- La aplicación del examen, las respuestas de los aspirantes.
- Calificación de los exámenes, el resultado o puntuación global de cada uno de los aspirantes.

Tomando la oferta educativa, la demanda educativa y la puntuación global de cada sustentante se procede a la realización del proceso de asignación en donde cada uno de los aspirantes que cumpla con todos los requisitos establecidos en la convocatoria del CIEMS 2001 es asignado en alguna de las opciones educativas elegidas por el mismo, en función de su resultado de examen y del cupo de cada opción educativa.

Hay que mencionar que toda la información involucrada en el proceso de asignación tiene que pasar por una revisión exhaustiva que permita garantizar su veracidad.

Procesamiento de la Demanda y de la Oferta de Educación Media Superior

A mediados del mes de mayo de 2001 la SEP a través de la Dirección General de Evaluación (DGE) hace entrega de dos archivos electrónicos con formato dBase IV necesarios e indispensables para el proceso de asignación:

- Archivo de Registro de Aspirantes al Metropolitano o RAM
- Archivo de Opciones Educativas (conocido también como Oferta Educativa)

El archivo RAM contiene la información generada en la etapa de registro al concurso, como datos relevantes tenemos:

a) La información de todos y cada uno de los sustentantes (nombre, dirección, promedio de secundaria, número de folio, etc.)

b) La información del lugar donde cada sustentante debe presentar su examen.
c) Las claves de los planteles u opciones educativas elegidas por todos y cada uno de los sustentantes.

El archivo de Opciones Educativas contiene la información de los planteles que pertenecen a las 9 instituciones que forman la Comipems y que obviamente participan en el concurso, como datos relevantes tenemos:

a) La clave de la opción educativa y clave de la institución a la que pertenece.
b) Nombre y dirección completa de la opción educativa.
c) Cupo o número de lugares disponibles en la opción educativa.

Ambos archivos son revisados, se revisa que los datos relevantes sean correctos, también se busca la existencia de información duplicada, es decir, que los datos de algún aspirante no aparezcan dos veces o más en el archivo RAM o bien que la información de una opción educativa no aparezca dos veces o más en el archivo de opciones educativas.

Principalmente se revisan en ambos archivos los datos utilizados en el proceso de asignación:

a) Valores no esperados en el número de folio, cualquier valor que no este en el intervalo de 0 a 9.
b) Promedio de secundaria, cualquier valor que este fuera del intervalo de 0 a 10.
c) Que todos los aspirantes hayan solicitado por lo menos una opción educativa.
d) Que el cupo o número de lugares disponibles de cada una de las opciones educativas no sea igual a cero.
e) Que cada una de las opciones educativas tenga una clave de plantel asignada.
f) Se comprueba si es que las opciones educativas solicitadas por los aspirantes en el archivo RAM están contenidas en el archivo de Opciones Educativas.
g) Que todos los planteles que aparecen en el Instructivo del CIEMS 2001 estén contenidos en el archivo de Opciones Educativas

En el caso de encontrar información no esperada u omitida se notifica inmediatamente a la DGE de la SEP para que esta a su vez responda por la anomalía y entregue una nueva versión del archivo con las correcciones correspondientes.

Procesamiento de las hojas de respuestas del examen

Como ya se mencionó anteriormente, la aplicación del examen es el 24 de junio de 2001. Ese día es presentado el examen por los aspirantes en dos turnos, matutino y vespertino. Tanto la UNAM como la Comipems aplican por separado y a quien les corresponde su examen.

Esa misma tarde - noche son recolectadas todas las hojas de respuestas de los aspirantes que se presentaron al examen (hojas provenientes de las diferentes sedes de aplicación, se conoce como sede de aplicación al lugar en donde es aplicado el examen) creándose por decirlo de una manera simple, un gran paquete de hojas de respuestas de los aspirantes de la UNAM y otro gran paquete de hojas de respuestas de los aspirantes examinados por la Comipems.

La UNAM toma las hojas de respuesta de los aspirantes examinados por ella y los pone bajo su custodia. La Comipems por su parte, clasifica y ordena en un lugar común y seguro y de acuerdo a la sede de aplicación las hojas de respuestas que le corresponden, para su posterior procesamiento, todo ante la presencia de notarios públicos que certifican la transparencia de este proceso.

Una vez que la información ha sido resguardada por la Comipems, se procede a la realización de los procesos conocidos como Lectura óptica y Micro filmación de las hojas de respuestas.

Lectura Óptica y Micro filmación

De acuerdo al turno y a la sede de aplicación son procesadas, o dicho de otra forma, son "leídas" una a una todas las hojas de respuestas de los aspirantes con la ayuda de equipos electromecánicos especiales llamados "lectores ópticos". Este proceso es conocido como "Lectura Óptica".

Los lectores ópticos pueden trabajar con computadoras personales a través del puerto serie de éstas últimas. De tal manera que, utilizando como “interface” o medio de comunicación un programa de computadora propio del fabricante del lector óptico, se puede controlar el proceso de lectura.

En general el proceso de lectura óptica se desarrolla de la siguiente manera: una vez que todo un paquete de hojas de respuestas es leído, se genera un archivo con un formato ASCII (Código Estándar Americano para el Intercambio de Información, por sus siglas en inglés) conteniendo en cada uno de sus renglones, entre otros datos que son utilizados como de control, el número de folio del aspirante y cada una de sus respuestas. Así por ejemplo, si "son leídas" 500 hojas se tendrá un archivo ASCII con 500 renglones, cada uno de estos renglones corresponde a cada una de las hojas de respuestas.

Es importante mencionar que la calidad de la "lectura" esta en función de las marcas realizadas en los alvéolos de la hoja, ya que el lector óptico tiene la característica de ser calibrado para así detectar las marcas realizadas en la hoja de respuestas con un lápiz de grafito del número 2½, esto se debe a que este tipo de lápiz es el que deja las marcas que mejor se pueden "leer".
Ahora bien, los lectores ópticos no "captan" correctamente las marcas realizadas con pluma, color, algún otro tipo de lápiz diferente al antes especificado o aquellas marcas que son demasiado tenues. Finalmente hay que mencionar que cuando el lector óptico encuentra dos marcas en la misma respuesta, invalida ese valor, colocando en esa posición un asterisco (*).

El proceso de lectura óptica es realizado en dos ocasiones, por lo que al final se tiene un archivo correspondiente a la "primer lectura" y otro que corresponde a la "segunda lectura", ambos en consecuencia contienen la información referente al número de folio y las respuestas de todos los aspirantes que se presentaron al examen.

Validación de la Lectura Óptica

Durante la realización de la primera y segunda lectura óptica es realizado un proceso de validación, este proceso analiza la información y presenta un diagnóstico. Dicho diagnóstico determina la existencia de folios mal codificados (por un error del aspirante), la falta de versiones en los exámenes (omitidas por los aspirantes) y las inconsistencias de versiones por turno (en el turno matutino solo se pueden presentar las versiones 1,2 y 3 mientras que, en el turno vespertino las versiones 4,5 y 6).

Después de que se ha terminado la segunda lectura óptica se realiza una comparación de las dos lecturas con la intención de buscar y determinar si es que existen inconsistencias en la información referente a las respuestas de los aspirantes.
Este proceso consiste en comparar la cadena de respuestas (se le llama cadena de respuestas al total de respuestas) de cada uno de los aspirantes de la primer lectura contra su equivalente de la segunda lectura, respuesta por respuesta. Las anomalías detectadas permiten identificar aquellos aspirantes

con posibles problemas, menciono posibles problemas debido a que la calidad de la lectura óptica esta en función de la calibración del lector, y es que por ejemplo, puede ser que una hoja a la cual llamare Hoja A es leída en la primer lectura por un lector, el Lector A pero en la segunda lectura esta misma hoja es leída por otro lector al que llamare Lector B, el cual puede tener un nivel de calibración ligeramente diferente al del primer lector, provocando que:

- Un alvéolo que si fue leído ahora no lo sea (por estar marcado muy tenuemente)
- Una doble respuesta (el lector capta dos marcas y coloca en esa posición un asterisco)
- Una respuesta donde antes no la había (el lector en esta ocasión captó una marca muy tenue, cuando en un principio no fue captada)

Las hojas de respuestas con anomalías son ubicadas y extraídas de sus paquetes para su revisión, son revisan las respuestas donde se han encontrado diferencias para así determinar cual respuesta es la correcta.

Para éste proceso de validación de lectura se ha desarrollado un sistema para computadora llamado Sistema de Validación de Lectura Óptica del Metropolitano o SIVALOM.

Ya que, el hecho de comparar las dos lecturas y determinar que estas son iguales garantiza una correcta lectura, resulta muy importante este proceso de comparación entre las dos lecturas ópticas.

Micro filmación

De una forma paralela al proceso de lectura óptica es realizado el proceso conocido como Micro filmación, el propósito de este proceso es el de conservar una imagen de todas las hojas de respuestas de los aspirantes en rollos de micro filmación, tanto las hojas de respuestas de los aspirantes que presentaron el examen como las de los que no lo presentaron. Este proceso consiste, como su nombre lo dice, en micro filmar mediante cámaras especiales cada una de las hojas de respuestas de los aspirantes por ambos lados. Los rollos utilizados son posteriormente revelados, obteniendo así las imágenes o fotografías de cada una de las hojas de respuestas.

Calificación de los exámenes

Una vez que el proceso de lectura óptica queda concluido se genera un archivo con formato ASCII conteniendo en cada uno de sus renglones:

- El número de folio del aspirante
- La versión del examen que presentó el aspirante
- La cadena de respuestas del aspirante

Dicho archivo contiene sólo a los aspirantes que si presentaron el examen.

Esta información es introducida en un programa para computadora el cual realiza el proceso de calificación. El programa unos momentos antes de iniciar el proceso de calificación es alimentado ante la presencia de notarios públicos con un segundo archivo en formato ASCII con las claves de respuestas correctas para cada una de las versiones del examen.

El proceso de calificación genera un archivo con formato ASCII con todos los sustentantes que presentaron el examen, dicho archivo contiene:

- El número de folio
- Versión del examen
- Número de aciertos obtenido en el examen

Hay que mencionar que el proceso de calificación se limita sólo a la realización del conteo del número de aciertos de cada aspirante de acuerdo a la versión del examen y a las claves de respuestas correctas correspondientes.

Asignación

El proceso de asignación tiene como único objetivo colocar a todos aquellos "aspirantes que hayan presentado el examen de selección, que hayan obtenido por lo menos 31 aciertos y que hayan concluido la educación secundaria"[1] en alguna de las opciones educativas que ellos mismos eligieron en función tanto de su puntuación obtenida en el examen, como del "número de lugares disponibles y los requisitos establecidos por cada institución"[1], el orden en que son atendidos los aspirantes es de acuerdo a su puntuación y es iniciado con los aspirantes de mayor puntuación y terminado con aquellos que obtuvieron sólo 31 aciertos.

Para lograr el objetivo del proceso de asignación es necesario combinar la oferta de educación media superior (proporcionada por la Comipems), el número de aciertos del examen de cada uno de los sustentantes y las opciones educativas solicitadas por ellos mismos.

El proceso de asignación debe de atender en igualdad de circunstancias a todos y cada uno de los aspirantes que cumplan con los requisitos establecidos en la convocatoria del CIEMS.

Debido a la gran cantidad de información involucrada en el proceso de asignación, resulta necesario utilizar una computadora como herramienta de procesamiento de los datos, por lo que hay que desarrollar un sistema para computadora que permita realizar la asignación de los aspirantes.

3.2.5 Publicación de los resultados

La publicación de los resultados del CIEMS 2001 es el 3 de agosto de 2001, su difusión se realiza mediante:

- La publicación impresa de la Gaceta de Resultados
- Los módulos de orientación
- Internet, en la dirección: http://www.sep.gob.mx
- Orientatel y Telsep.

"En la Gaceta aparecerán los resultados obtenidos por todos los aspirantes registrados en este concurso, identificados y ordenados según el número de folio del comprobante-credencial"[2]

En los módulos de orientación se puede consultar el resultado de cada uno de los aspirantes, con atención y orientación personal de gente que forma parte de la Comipems.

En Orientatel y Telsep se puede obtener, vía telefónica, el resultado individual de cada participante del concurso.

---

[1] Instructivo del CIEMS 2001
[2] Convocatoria al CIEMS 2001

Tanto en la Gaceta de Resultados como en Internet se puede obtener la información de la ubicación de los planteles, las opciones educativas que después del concurso aún tienen lugares disponibles[1], las fechas de inscripción, entre otros datos informativos.

Bibliografía

1. **Comipems,** "El Concurso de Ingreso a la Educación Media Superior de la Zona Metropolitana de la Ciudad de México 2001"

2. **Comipems,** "Convocatoria al Concurso de Ingreso a la Educación Media Superior de la Zona Metropolitana de la Ciudad de México 2001"

Capítulo 4: Análisis del proceso de asignación

4.1 Herramientas del análisis estructurado

A continuación presento el análisis realizado al proceso conocido como "Asignación" el cual como ya mencioné, forma parte del CIEMS 2001[1]. Cabe recordar que la fase de análisis es una actividad de modelado en la cual se recurre a diferentes "técnicas de modelado" para sistemas basados en computadora, técnicas que le permiten al analista del sistema representar las necesidades que debe de satisfacer el sistema que se desea construir.

Existe más de un método de análisis de sistemas basados en computadora, tales métodos han surgido y evolucionado a lo largo de los últimos 40 años. Los más importantes, en mi opinión son el método del análisis estructurado y el método del análisis orientado a objetos, el primero por ser el método clásico para analizar sistemas y el segundo porque desde su aparición ha creado grandes expectativas en torno a él mismo al ofrecer la posibilidad de modelar los sistemas de una manera más realista al permitir representar a través de código "objetos" del mundo real, además de presentar la posibilidad de volver a utilizar una mayor cantidad de código a través de las "clases" y la "herencia". Debo aclarar que de estos dos métodos de análisis, en este capítulo se presenta el método de análisis estructurado[2].

Al utilizar el método de análisis estructurado se deben crear modelos (representaciones gráficas, descripciones textuales, etcétera) de nuestro sistema, modelos que nos permitan representar su información, la o las transformaciones que sufre dicha información y el flujo que la misma información tiene tanto interna como externamente de una manera clara, para lograrlo se han desarrollado desde la década de 1960 diversas herramientas de modelado para la fase de análisis estructurado. Dichas herramientas son:

- Diagrama de flujo de datos o DFD, esta herramienta permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por "conductos" y "tanques de almacenamiento" de datos[3]. Este enfoque sólo presenta las *funciones* del sistema.
- Diccionario de datos, un diccionario de datos es un listado organizado de todos los datos pertenecientes al sistema, con definiciones precisas y rigurosas para que tanto el usuario como el analista tengan un entendimiento común de todas las entradas, las salidas, los componentes de almacenes de datos y los cálculos intermedios[3] realizados a los datos.
- Especificación del proceso, se trata de una descripción narrativa, tablas de decisiones, lenguaje estructurado o diagrama de flujo que describe lo que realiza cada uno de los procesos especificados en el DFD.
- Diagrama de entidad relación o DER, los diagramas de entidad relación son una notación gráfica que es utilizada para modelar los datos que son utilizados por el sistema, en la mayoría de los casos se utiliza como una herramienta de modelado de la base de datos del sistema. A diferencia del DFD que se enfoca en las funciones del sistema el DER se enfoca en los *datos* del sistema.
- Diagrama de transición de estados o DTE, se enfoca en el *comportamiento dependiente del tiempo* del sistema, por lo general esta herramienta de modelado es utilizada en los sistemas de tiempo real.

---

[1] Sección "Asignación" del capítulo "El concurso de ingreso" de esté trabajo de tesis

[2] Sección "Elección del SDLC para el proceso de asignación" del capítulo "Antecedentes" de esté trabajo de tesis

[3] Análisis estructurado moderno, Edward Yourdon

Estas herramientas son utilizadas en mayor o menor medida de acuerdo a las preferencias del equipo de desarrollo y a la naturaleza misma del problema, ya que, una herramienta puede resultar más adecuada para un problema que otra.

4.2 Análisis estructurado del proceso de Asignación

El éxito del software se funda en un buen análisis de requisitos (de los requisitos que tiene el cliente), pues no importa que tan bien diseñado o codificado se tenga un sistema sí es que el problema a resolver no fue bien analizado y se llegó a una solución diferente a la esperada por el cliente. En este caso el cliente se sentirá defraudado y muy probablemente el software nunca será utilizado, y es que, en muchos casos se crea una solución para el problema equivocado al no cumplir con las expectativas y los requisitos del cliente. De aquí que yo considere de vital importancia el realizar una buena fase de análisis del proceso de Asignación, porque de lo contrario, el fracaso del sistema estará muy cercano.

Durante el proceso de análisis estructurado se debe:

- Modelar y entender la información que es utilizada por el problema a resolver
- Definir las tareas que el software debe realizar evitando el caer en ambigüedades
- Entender qué la respuesta del software ante un estimulo externo debe modelarse
- Dividir los modelos creados para el software, presentando los detalles de cada uno de ellos de una manera jerárquica o por capas, para así reducir la complejidad del modelo
- Iniciar el modelado del software a partir de su información esencial, para después avanzar hasta terminar con el detalle de los procesos internos que realizarán las tareas del sistema
- Ofrecer la posibilidad de que el lector del modelo desarrollado pueda predecir el comportamiento del mismo

Los modelos que se crearán en esta etapa ayudan al equipo de desarrollo a:

- Enfocarse en las partes del sistema que son de mayor importancia
- Realizar cambios en los requerimientos del usuario antes de que se haya escrito una sola línea de código, reduciendo el costo de las modificaciones en tiempo y en dinero
- Verificar que se ha representado de una manera confiable el ambiente del usuario, un ambiente que le permita al usuario visualizar lo que éste desea
- Escribir el código del sistema más fácilmente

En el análisis estructurado se parte funcionalmente el sistema a construir, es decir, se va descomponiendo o dividiendo el sistema en pequeños procesos los cuales a su vez son descompuestos en otros procesos hasta llegar a un punto en el cual todas las funciones o tareas que el sistema debe de realizar para lograr su o sus objetivos ya se han especificado. Esto implica que se deben de modelar los requisitos de datos, de flujo de información y de control, así como también hay que modelar el comportamiento operativo del sistema.

A continuación presento el análisis realizado al proceso de asignación, los objetivos que persigo son dos; el primer objetivo es tener una idea clara y escrita de cuál es el objetivo del proceso de asignación del CIEMS 2001, el segundo objetivo es tener los modelos necesarios que describan de una manera gráfica el objetivo del proceso de asignación del CIEMS 2001.
Para lograrlo me apoyaré tanto en la convocatoria al CIEMS 2001 como en su instructivo ya que ahí se especifican las reglas que debe de seguir el proceso de Asignación de los sustentantes a cada una

de las opciones educativas de su elección, es decir, tanto el instructivo como la convocatoria al CIEMS 2001 son la base sobre la cual se realizará el análisis del proceso de Asignación pues en ellos están especificados los requerimientos del mismo.

De acuerdo con la convocatoria al CIEMS 2001 que dice: La UNAM diseñará y calificará su propio examen y lo aplicará a aquellos aspirantes que la elijan en la primera opción de sus preferencias[4]. Se desprende la siguiente afirmación: **El presente análisis se aplica a todos aquellos aspirantes del CIEMS 2001 que no son atendidos por la UNAM**.

El sistema del proceso de Asignación a desarrollar encuentra sus requisitos en la Convocatoria al CIEMS 2001, en específico en el apartado número siete titulado "Reglas para la asignación de los lugares".

Apartado número 7 de la convocatoria al CIEMS 2001.

7.1 La asignación de lugares se realizará sólo entre los aspirantes que cumplan con los siguientes requisitos:

- Sustentar el examen el domingo 24 de junio en el lugar y hora señalados en el comprobante-credencial que se entregará al momento del registro;
- Obtener más de 30 aciertos en el examen (por lo menos 31);
- Contar con su certificado de educación secundaria (o la constancia a que se refiere la base 6.2) a más tardar el 17 de julio de 2001, y
- Cumplir con los requisitos particulares que fija cada institución, según se informa en el instructivo del Concurso.

7.2 El único criterio para asignar un lugar a un aspirante será el número de aciertos obtenido en el examen respectivo.

Por lo tanto, los aspirantes serán ordenados según el número de aciertos obtenido en el examen (de mayor a menor). En ese orden, a cada aspirante le será asignado un lugar en la opción educativa de su más alta preferencia que disponga de cupo, conforme a la jerarquización de preferencias que aparece en su comprobante-credencial. El Instructivo del Concurso contiene una explicación más detallada del procedimiento.

7.3 Los aspirantes con igual puntaje en el examen tendrán el mismo derecho de asignación.

Para ampliar un poco más el punto 7.1, a continuación presento el punto 5 de la etapa VII del Instructivo del CIEMS 2001 titulada "Calificación de los exámenes y procesamiento de los datos":

"Antes de realizar las rutinas antes mencionadas, la computadora separa a los aspirantes que "no se presentaron al examen" (NP), a los que no obtuvieron 31 o más aciertos (<31) y a los que adeudan alguna asignatura de la secundaria o por alguna razón no tienen el certificado respectivo (S/C). También "elimina" a quienes fueron dados de baja en el Concurso por alguna irregularidad (por ejemplo, un intento de fraude o suplantación)."

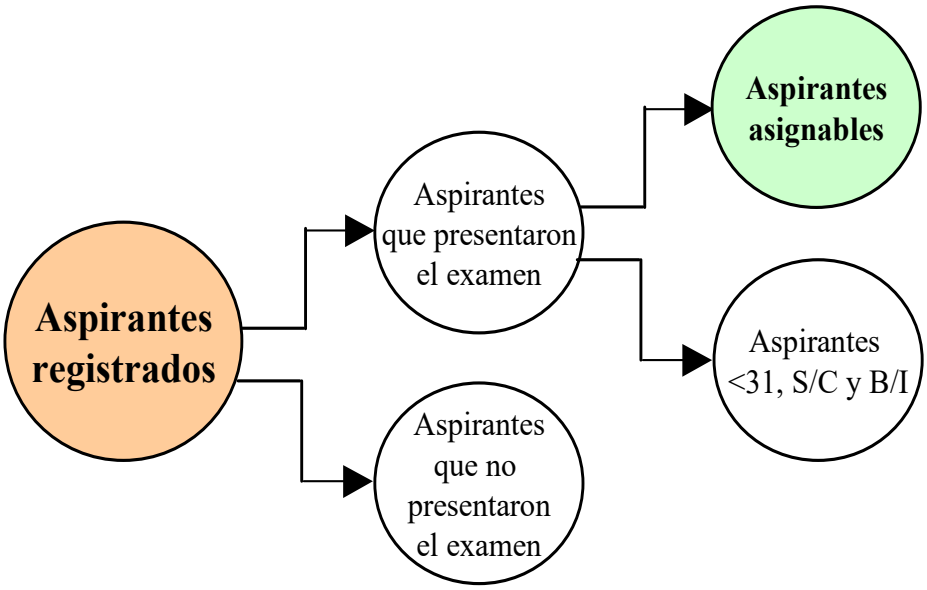
Analizando toda esta información puedo escribir la especificación del proceso de asignación:

El proceso de Asignación debe atender a todo aquel aspirante que:

[4] Instructivo del CIEMS 2001

a) Presente el examen el 24 de junio de 2001.
b) Obtenga más de 30 aciertos en el examen (por lo menos 31 aciertos).
c) Cuente con su certificado de secundaria a más tardar el 17 de julio de 2001.
d) No haya incurrido en una falta que provoque su baja del concurso.

Al grupo de aspirantes que cumpla con los puntos antes mencionados se le conoce como "asignables". Por lo tanto en el grupo de los asignables se encuentran todos los aspirantes que competirán por obtener un lugar en alguna de las opciones educativas de su preferencia cumpliendo con los requisitos particulares establecidos por cada institución y de acuerdo a lo especificado en los apartados 7.2 y 7.3 de la Convocatoria al CIEMS 2001. La figura 4.1 presenta las diferentes poblaciones del CIEMS.



Donde:
<31 se refiere a los aspirantes con menos de 31 aciertos en el examen.
S/C se refiere a los aspirantes sin certificado.
B/I se refiere a los aspirantes dados de baja del concurso.

Figura 4.1 Diferentes poblaciones del CIEMS

A continuación presento el análisis estructurado del proceso de asignación.

4.2.1 Diagramas de flujo de datos (DFD) del proceso de asignación

En los DFD's que presento a continuación no estarán aquellos de más bajo nivel (aquellos que muestran el detalle de cómo se resolverá el proceso de asignación), en cambio presentaré los DFD's que le permitan al lector de esta tesis el entender de una manera general cómo se desarrollará el proceso de asignación.

El DFD de la figura 4.2 es el diagrama de contexto del proceso de asignación, en él se muestran las entidades externas, los flujos de información de entrada y de salida del proceso. El objetivo de este

diagrama es el de ofrecer una idea general o global de los actores internos y externos del proceso (personas, empresas, dependencias, etcétera) así como también el mostrar los principales flujos de entrada y de salida (resultados) de información.

El DFD de la figura 4.3 presenta el primer nivel del proceso de asignación, en él se muestran uno a uno los procesos necesarios para lograr que el proceso de asignación se lleve a cabo, debo aclarar que aunque las tareas tienen un número asignado, éste no especifica necesariamente el orden en el que se llevarán a cabo. En este diagrama se pueden observar los flujos de información y la transformación que sufren durante todo el proceso de asignación, los flujos son representados por una flecha que entra o sale de un proceso, también se presentan los almacenes de datos, estos son representados con la figura generada al unir un cuadrado y un par de líneas paralelas. Es importante distinguir que el flujo de información sirve para presentar datos en movimiento, mientras que el almacén de datos sirve para presentar datos en reposo.

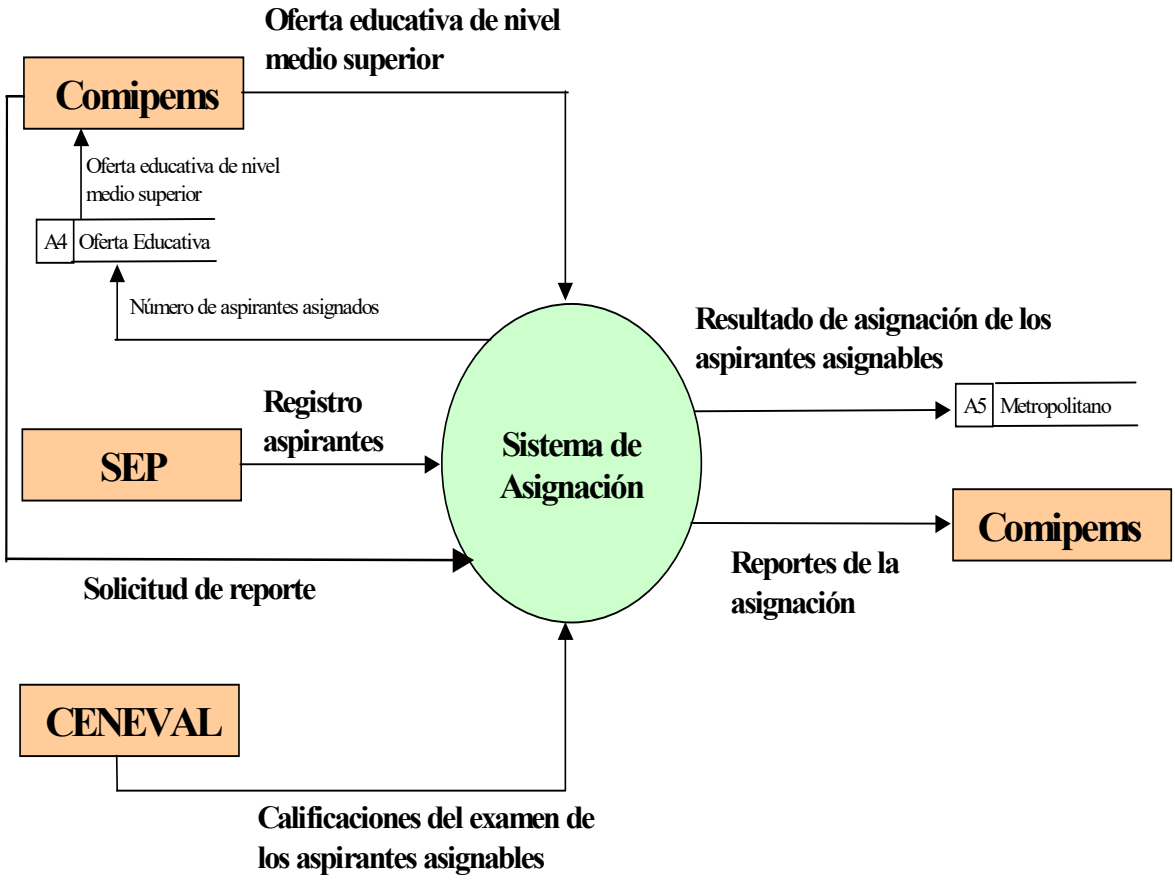


Figura 4.2 DFD de nivel contextual del Sistema de Asignación

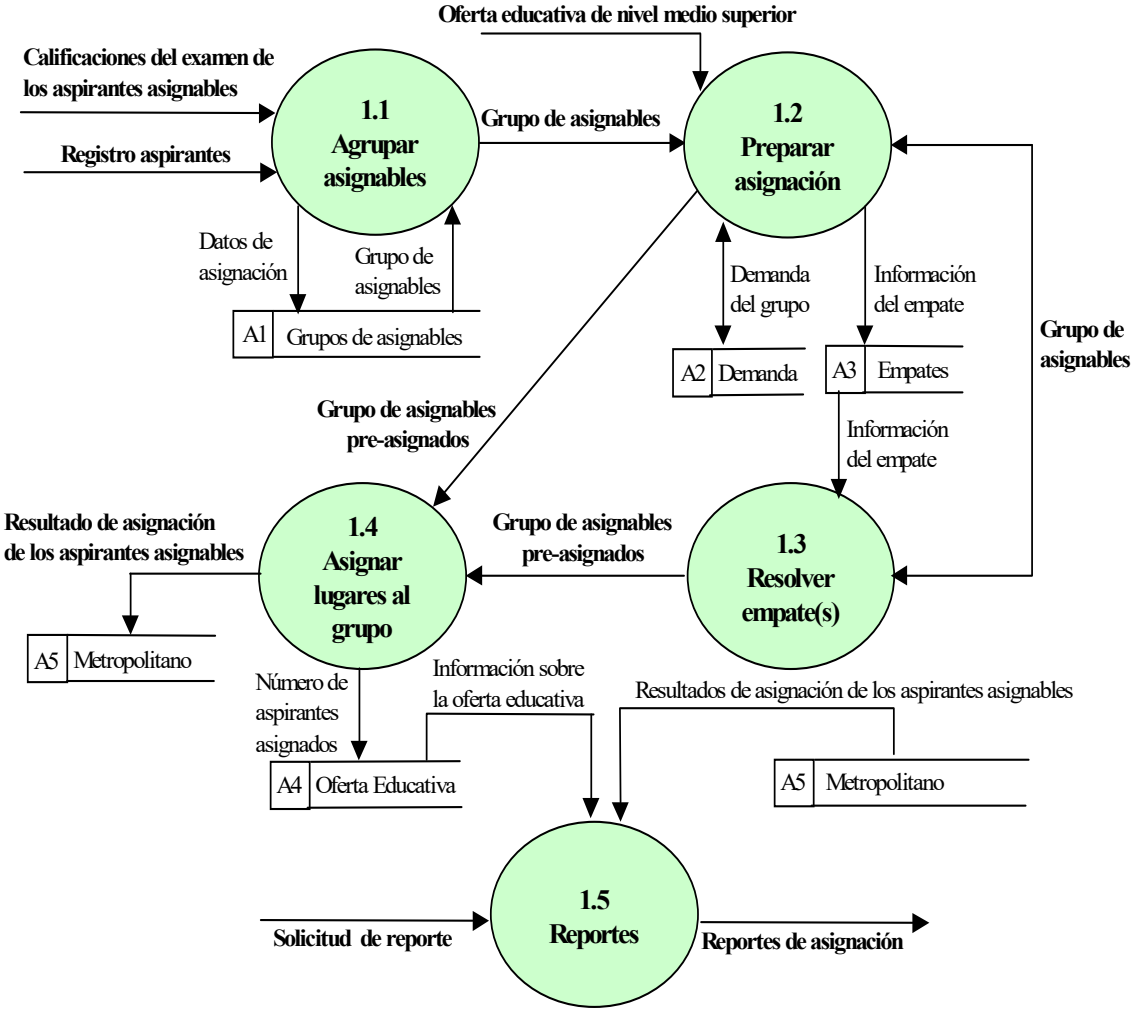


Figura 4.3 DFD nivel 1 del Proceso de Asignación

4.2.2 Especificación de procesos

En esta sección presento las descripciones narrativas de cada uno de los procesos de los DFD's del proceso de asignación, es decir, la descripción de que es lo que sucede en cada uno de los procesos.

Frecuentemente es utilizado como alternativa a las descripciones narrativas de procesos el Lenguaje Estructurado (LE), el cual es un lenguaje con estructura, con restricciones en el tipo de palabras que se pueden utilizar y en la manera en que dichas palabras se pueden combinar para formular las oraciones. También es utilizado un grupo pequeño de verbos orientados a describir una acción (Validar, Buscar, etcétera) y es permitido combinar frases tomadas de la programación estructurada tales como SI-ENTONCES-OTRO, HACER-MIENTRAS, etcétera.

En teoría las descripciones de procesos se realizan sobre las burbujas de más bajo nivel pero como dichas burbujas no se incluyen en este trabajo de tesis, presentaré en su lugar las descripciones narrativas de procesos de la figura 4.3.

Con la intensión de presentar las dos opciones de especificación de procesos, primero utilizaré LE en el proceso 1.1, mientras que en los procesos restantes utilizaré descripciones narrativas.

Debo aclarar que además de las descripciones narrativas también se desarrollaron diagramas de flujo, dichos diagramas corresponden al nivel más bajo del análisis del proceso de asignación y por lo tanto no se presenta en este trabajo de tesis.

Especificación en LE del proceso 1.1 Agrupar asignables

COMIENZA PROCESO
REPITE HASTA que se haya atendido a todo el **Registro de aspirantes**
TOMAR el número-de-folio
BUSCAR el número-de-folio en las **Calificaciones del examen de los aspirantes asignables**
SI encuentra el número-de-folio ENTONCES
COLOCAR **datos de asignación** del aspirante en el grupo de asignables que le corresponda de acuerdo a su calificación
FIN_SI
FIN_REPITE

Especificación narrativa del proceso 1.2 Preparar asignación

El proceso recibe como información de entrada:

- El grupo de asignables y
- La oferta educativa de nivel medio superior

Se obtiene la siguiente opción educativa[5] válida de cada uno de los aspirantes contenidos en el grupo de asignables (una opción educativa se considera como válida, si es que aún cuenta con lugares y el sustentante cumple con el requisito de promedio necesario para ingresar a la opción, si es el caso) iniciando en la opción educativa número uno sí es que es la primera vez que se obtiene su opción educativa válida, en caso contrario, se inicia en la siguiente a su última opción educativa válida calculada. Si a cualquier aspirante se le agotan sus opciones educativas, es decir, no se le puede obtener su siguiente opción válida, esté debe de ser marcado como CDO[6] (Con Derecho a otras Opciones).

Después se contabilizan todas las opciones educativas válidas de los asignables para generar así la demanda de lugares por opción educativa, dicha demanda es guardada en el almacén llamado Demanda. A continuación se toma una a una la demanda por opción educativa para compararse con su correspondiente en la oferta de educación media superior, el resultado de todas estas

[5] Opción educativa es cualquiera de las opciones educativas elegidas por el sustentante en su hoja de solicitud de registro al CIEMS 2001.
[6] Un aspirante es marcado como CDO cuando no alcanzó el número de aciertos suficientes o no cumplió con el requisito de promedio establecido por la institución y fue superado entonces por los aspirantes que solicitaron las mismas opciones educativas y ganaron todos los lugares disponibles.

comparaciones puede generar cero empates[7], o bien, puede generar uno o varios empates, si hay uno o varios empates la información correspondiente al empate o a los empates se guarda en el almacén llamado Empates.

Si durante la comparación de oferta contra demanda se generan cero empates, el proceso entrega al grupo de asignables ahora como pre-asignados al proceso 1.4, en caso contrario se entrega el grupo de asignables al proceso 1.3.

Especificación narrativa del proceso 1.3 Resolver empate(s)

El proceso recibe como información de entrada:

- El grupo de asignables
- La información del (los) empate(s) y
- La demanda del grupo

Uno a uno se resuelven los empates eligiendo una de las siguientes dos opciones:

a) Ampliación del cupo de la opción educativa, en este caso todos los aspirantes empatados son aceptados.
b) Cerrando la opción educativa, en este caso ningún aspirante empatado es aceptado.

Si por lo menos un empate es resuelto "cerrando la opción educativa", se debe calcular otra vez la demanda de lugares por opción educativa del grupo de asignables. Esto implica obtener la siguiente opción válida de cada uno de los aspirantes que se vieron afectados por el cierre de la opción educativa, y es que para ellos y para el resto de los aspirantes asignables[8], la opción educativa cerrada ya no es una opción válida. Cuando sucede esto se debe enviar al proceso Preparar Asignación de nueva cuenta el grupo de asignables. En el caso de que se resuelvan los empates sin cerrar ninguna opción educativa el proceso entrega al grupo de asignables ahora como pre-asignados al proceso 1.4.

Especificación narrativa del proceso 1.4 Asignar lugares al grupo

El proceso recibe como información de entrada al grupo de asignables pre-asignados y marca como "Asignados" en el almacén de datos Metropolitano a todos los aspirantes que no hayan sido identificados como CDO's en el proceso Preparar Asignación, de igual manera marca como CDO's a los aspirantes restantes. Después resta el número de aspirantes asignados por opción educativa en el almacén Oferta Educativa, con lo que la oferta de las opciones educativas es disminuida.

Es importante aclarar que en el proceso de asignación no todos los aspirantes asignables obtienen un lugar, ya que existen los aspirantes marcados como CDO's. Es por esta razón que el proceso 1.4 entrega el flujo de información llamado "resultado de asignación de los aspirantes asignables" y no un flujo llamado "aspirantes asignados".

Especificación narrativa del proceso 1.5 Reportes

[7] Un empate ocurre cuando el número de aspirantes que desean ingresar a una opción educativa determinada es mayor que el número de lugares disponibles en dicha opción educativa, se dice que los aspirantes están empatados puesto que todos en este momento tienen la misma oportunidad de ser aceptados por la institución.
[8] Incluyendo también a los grupos de asignables con puntajes menores al actual o que aún no han sido atendidos.

El proceso recibe como información de entrada la petición de generar un reporte, dicho reporte puede ser alguno de los siguientes:

- Reporte global del número de aspirantes asignados por institución educativa
- Reporte del número de aspirantes asignados en cada una de las opciones educativas
- Reporte de instituciones que después del proceso de asignación cuentan con lugares disponibles

Y entrega como resultado, de acuerdo a la petición realizada, el reporte en pantalla con la opción de imprimirse en papel.

4.2.3 Diagrama de entidad relación (DER) del proceso de asignación

A continuación presento una herramienta gráfica para modelar datos, esta herramienta es independiente al proceso o procesos que transforman la información en un sistema. Se trata del Diagrama de Entidad Relación o DER el cual permite desarrollar el modelo de datos del sistema, enfatizando las relaciones entre almacenes de datos.

En teoría el DER permite identificar objetos de datos y sus relaciones mediante una notación gráfica. Pero aquí debido a que ya se han identificado los objetos de datos (los almacenes de datos del DFD de nivel 1) en esta parte solo estoy enfatizando las relaciones que existen entre ellos.

Para poder presentar el DER del sistema de asignación primero explicare de una manera resumida sus elementos.

Los objetos de datos o **entidades** son cualquier cosa que puede identificarse en el ambiente de trabajo del usuario, o bien, cualquier grupo de datos que definan un objeto y que el sistema deba de entender. Así por ejemplo los siguientes datos definen a un cliente: Nombre, Número de cliente y dirección, otros ejemplos de entidades son: una persona, un automóvil, un departamento perteneciente a una organización, un almacén, una orden de pedido, una factura, etcétera.

Los **Atributos** se refieren a las propiedades que describen las características de las entidades, características que permiten relacionar una entidad con otra. Como ejemplos tenemos: nombre de un empleado, número de ruedas de un automóvil, el código de área de un número telefónico, el color de un archivero, etcétera.

Los **Identificadores** son los atributos que identifican a las entidades, en la mayoría de los casos son únicos y sirven para encontrar una instancia[9] de una entidad. Uno o varios atributos se pueden definir como un identificador, así por ejemplo Clave Lada más Número local identifica a un distribuidor, otro ejemplo sería Número de motor identifica a un automóvil.

Las **Relaciones** permiten describir las asociaciones existentes entre las entidades. Una relación puede incluir muchas entidades, el número de entidades involucradas determina el grado de la relación, por ejemplo, una relación entre dos entidades tiene un grado 2. Cuando el grado de la relación es dos se pueden presentar 3 tipos de relaciones:

[9] Una instancia es la representación particular de una clase de entidad, por ejemplo para la entidad Cliente una instancia puede ser Juan Miguel Fernández.

- Relación 1 a 1 (1:1), sucede cuando solo una instancia de una entidad A esta relacionada con solo una instancia de una entidad B y de igual manera para B. Por ejemplo, un estudiante tiene asociado sólo un número de cuenta y sólo un número de cuenta le corresponde a un estudiante, la figura 4.4 presenta este tipo de relación.

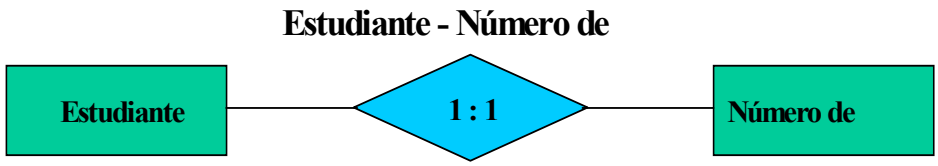


Figura 4.4 Relación 1 a 1

- Relación 1 a muchos (1:N), sucede cuando sólo una instancia de una entidad A está relacionada con una o con muchas instancias de una entidad B, pero sólo una instancia de B está relacionada con sólo una instancia de A. Por ejemplo, un estudiante puede ser dueño de muchos libros de matemáticas, pero cada libro de matemáticas sólo puede tener como dueño a un estudiante, la figura 4.5 presenta este tipo de relación.

Figura 4.5 Relación 1 a muchos

- Relación muchos a muchos (M:N), sucede cuando una instancia de A está relacionada con una o muchas instancias de B y una instancia de B está relacionada con una o muchas instancias de A. Por ejemplo: un estudiante asiste a M salones de clases y en un salón de clases N estudiantes toman clase, la figura 4.6 presenta este tipo de relación.

Figura 4.6 Relación muchos a muchos

Los valores 1:1, 1:N y M:N son conocidos como la máxima cardinalidad de la relación e indican el número máximo de entidades que pueden ocurrir en cada uno de los extremos de la relación. Puede suceder que en una relación una de las entidades puede o no existir, es decir, cuando no es necesario que esté la instancia en la relación se dice que tiene una cardinalidad mínima de cero. Por el contrario, si es necesario que la instancia exista (o debe de existir) en la relación, se dice que tiene una cardinalidad 1. Por ejemplo, un dormitorio tiene asociado por lo menos un estudiante (cardinalidad mínima de 1) y un máximo de N estudiantes (cardinalidad máxima), mientras que un estudiante puede no estar asociado a ningún dormitorio (cardinalidad mínima de cero) o estar asociado como máximo a un dormitorio (cardinalidad máxima).

Existen **Subtipos de entidades** y ocurren cuando algunas entidades contienen atributos opcionales. Por ejemplo, tenemos la entidad Cliente con los atributos número de cliente, nombre del cliente y cantidad adeudada. Puede suceder en el entorno de la empresa que un cliente sea una persona, un socio de negocios o una compañía y que obviamente de acuerdo al tipo de cliente resulte necesario almacenar información adicional, de tal manera que se deban de definir tres subtipos de entidades relativas a cliente: cliente-persona, cliente-socio y cliente-compañía, la figura 4.7 presenta estos subtipos de entidades.

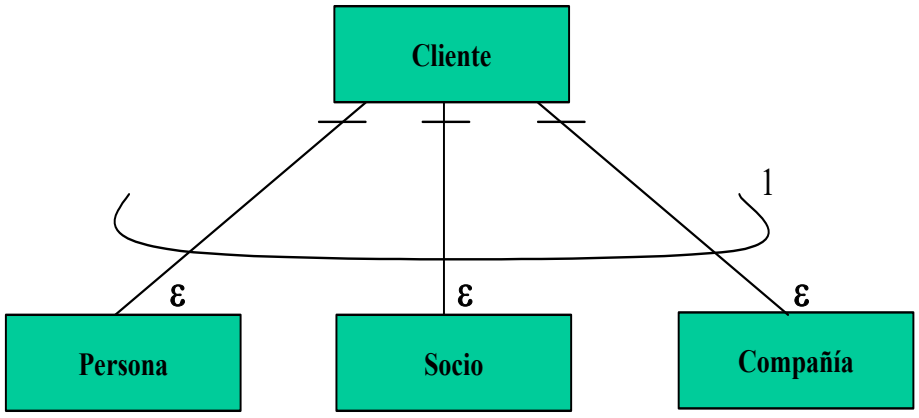
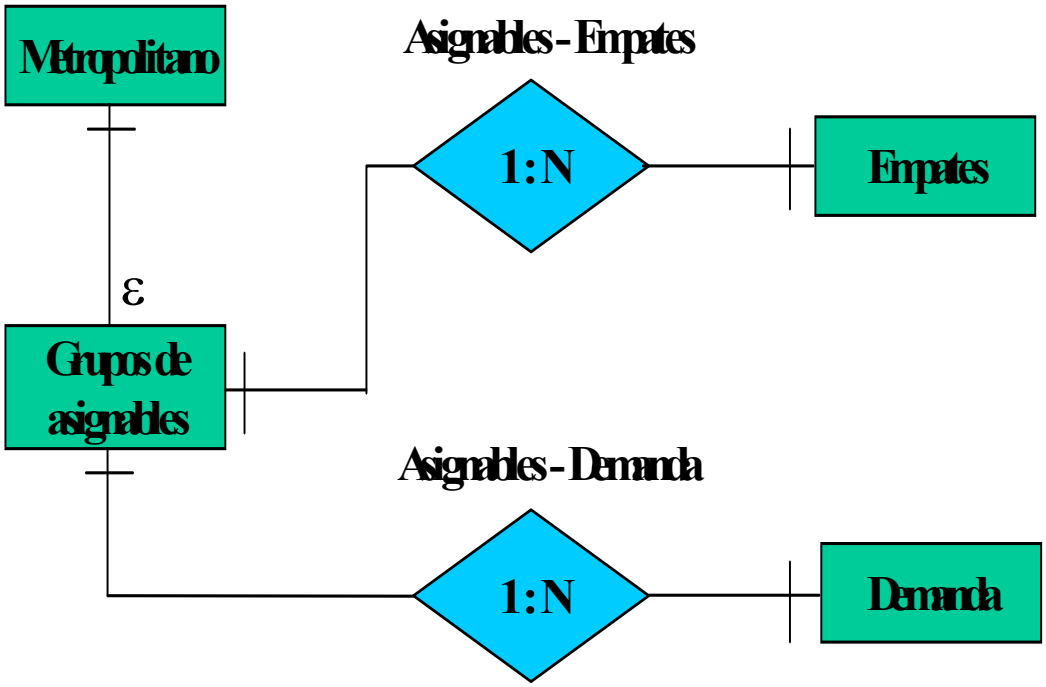


Figura 4.7 Subtipos de la entidad Cliente.

En la figura 4.7, el símbolo ε denota que es un subtipo de la entidad cliente. La línea curva con el número 1 indica que la entidad cliente esta asociada a uno y solo a un subtipo, esto quiere decir que los subtipos son mutuamente excluyentes y que solo uno de ellos es requerido. El guión establece que la relación inicia desde un cliente (cardinalidad mínima) y obviamente termina en un subtipo de cliente (cardinalidad máxima).

Una vez presentados los elementos de un DER, en la figura 4.8 presento el DER del Sistema de Asignación.



4.8 DER del Sistema de Asignación

Del diagrama podemos observar que:

Hay una relación de subtipo entre el almacén Metropolitano y los Grupos de asignables, cada grupo de asignables debe ser del metropolitano.
A un grupo de asignables se les puede relacionar con uno o varios empates y un empate se puede relacionar con un solo grupo de aciertos.
A un grupo de asignables se les puede relacionar con una o varias demandas y una demanda se le puede relacionar con solo un grupo de asignables.

Con esto queda concluido el análisis del proceso de asignación.

Bibliografía

1. **Edward Yourdon,** Análisis Estructurado Moderno**,** editorial Prentice Hall Hispanoamericana, S. A.; México Edo. De México; 1993; capítulos 4, 9, 10, 11 y 12.

2. **Roger S. Pressman,** Ingeniería del Software Un Enfoque Práctico, editorial McGraw-Hill; México D.F.; 1998; páginas 183 – 192 y 201 – 221.

3. **Kenneth E. Kendall y Julie E. Kendall,** Análisis y Diseño de Sistemas, editorial Prentice Hall Hispanoamericana, S. A.; México Edo. De México; 1997; Capítulo 9, páginas 345 - 352 .

4. **James Martin y James J. Odell,** Análisis y Diseño Orientado a Objetos, editorial Prentice Hall Hispanoamericana, S. A.; México Edo. De México; 1994; capítulo 5.

5. **David M. Kroenke,** Database Processing Fundamentals, Design & Implementation, editorial Prentice Hall; USA New Jersey; 2000; páginas 49 – 66.

Capítulo 5: Diseño del proceso de asignación

Las fases de análisis y de diseño de los sistemas basados en computadora o SBC están muy relacionadas entre sí. El método de análisis crea la base sobre la cual posteriormente se diseñará el software, por lo que si es utilizado un método de análisis estructurado entonces se debe utilizar un método de diseño estructurado.

5.1 ¿Qué es el diseño de software?

El diseño de software es la primera de tres actividades técnicas – diseño, codificación y prueba – necesarias para construir y verificar el software[1]. Hay que seguir una serie de pasos (que son repetidos tantas veces como sea necesario) que le permiten al diseñador el especificar todos los puntos del software a crear. Recordando la analogía con la construcción de una casa, el diseño equivale a los planos de la misma, por esto el modelo del diseño ofrece vistas generales y a detalle del software a construir, estas diferentes vistas del modelo permitirán construir posteriormente el software.

En la fase de diseño, el equipo de desarrollo debe traducir los requisitos del cliente en software, esto es, un proceso repetitivo que tiene como objetivo crear un modelo de diseño del sistema, este modelo toma como base el modelo del análisis.

La relación directa entre la fase de análisis y de diseño la presento a continuación:

Fase de análisis		***Fase de diseño***
Especificación de procesos	→	Diseño procedimental
Diagrama de flujo de datos	→	Diseño de interfaz
Diagrama de flujo de datos	→	Diseño arquitectónico
Diagrama entidad-relación, diccionario de datos	→	Diseño de datos

De acuerdo a la ingeniería del software de Pressman[1], los diferentes modelos del diseño se colocan siguiendo la forma de una pirámide en donde primero se debe crear una base muy sólida con el diseño de datos, encima de esta base se coloca en la región media el diseño arquitectónico y de interfaz para finalmente en la parte superior ubicar el diseño procedimental.

Lo anterior sugiere que el diseño de datos sea una base sólida y amplia sobre la cual se colocará el resto del diseño del software, esta superficie que sirve como cimiento es la base de datos.
El diseño arquitectónico describe la estructura jerárquica del software, es decir, las relaciones y el acomodo de los diferentes módulos.
El diseño de la interfaz de usuario consiste en la utilización de técnicas para la elaboración de formas y pantallas que permitan capturar los datos necesarios para el sistema, también incluye el diseño del formato de la información de salida ya sea en pantalla como en forma de impresiones en papel. Una interfaz es un medio que permite al usuario comunicarse con el software (puede tratarse también del teclado, el ratón, un menú de comando, etcétera).
El diseño procedimental como su nombre lo indica se refiere a la lógica de los diferentes procedimientos de los módulos del software.

[1] Ingeniería del software un enfoque práctico, Roger S. Pressman

Ahora que he expuesto la relación entre las fases de análisis y de diseño, debo aclarar que se debe utilizar alguno de los diferentes métodos de diseño que existen para lograr esta transición. Un método de diseño debe cumplir con:

- Un mecanismo para la transformación de un modelo de análisis en una representación del diseño
- Una notación para representar componentes funcionales y sus interfaces
- Heurísticas para el refinamiento y la partición
- Consejos para validar la calidad del modelo

Finalmente diré que, en la fase de diseño el equipo de desarrollo del software debe de asegurar que los requerimientos especificados en la parte del análisis se puedan implantar con la tecnología actual. Si se realiza una buena fase de diseño se tendrá al final un sistema o software de alta calidad (libre de errores), esto permitirá en un futuro hacer cambios al software fácil y rápidamente.

5.2 Conceptos sobre el diseño de software

Existen conceptos de diseño fundamentales que todo diseño debe en teoría cumplir, dichos conceptos son Abstracción, Refinamiento, Modularidad, Arquitectura del software, Jerarquía de control, Partición estructural, Estructura de datos y Ocultamiento de información.

5.2.1 Abstracción

Debemos de aislar o considerar por separado cada uno de los niveles desde los que se puede observar una solución modular de un problema, es decir, podemos abstraernos en el mayor nivel de nuestra solución y ver a nuestro software solo en términos del problema que estamos resolviendo, así por ejemplo para este caso particular puedo decir:

“El sistema de asignación intenta colocar a cada uno de los aspirantes en alguna de sus preferencias de acuerdo a su puntaje y a los lugares disponibles.”

Si seguimos descendiendo en los niveles de abstracción de nuestro problema entraremos en la parte en donde los detalles de los procedimientos son especificados combinando la problemática con la solución propuesta en la fase de análisis. Continuando con nuestro ejemplo y ya en un nivel diferente de abstracción puedo afirmar:

“El módulo marca como NP a todo aquel aspirante que no haya presentado el examen el día y a la hora señalados en su comprobante credencial.”

Conforme se avanza en los niveles de abstracción, se van creando abstracciones de procedimientos y de datos, la primera se refiere al conjunto de sentencias que tienen un objetivo específico y bien delimitado, mientras que la segunda se refiere a la colección de datos del sistema (para este caso por ejemplo el número de folio, la información relativa a las opciones educativas, las calificaciones del examen, etcétera). También se crean abstracciones de control las cuales se refieren a los mecanismos de control de los programas pero sin presentar los detalles internos.

Para concluir, en el nivel más bajo de abstracción se describe de manera directa la solución que se desea tomar o realizar y es especificada con la construcción de las diferentes rutinas y el programa principal del sistema.

5.2.2 Refinamiento

La estructura de un programa se desarrolla repetidamente para ir revisando y obteniendo cada vez más y más detalles del procedimiento hasta llegar a las instrucciones del lenguaje de programación, es decir, se va refinando nuestro programa partiendo de un enunciado que especifica de una manera general que se debe hacer, para después, proporcionar detalles con cada nuevo refinamiento que se hace, hasta llegar a las instrucciones de programa.

5.2.3 Modularidad

El sistema a construir se debe descomponer en diferentes secciones, partes o módulos para ser estudiados por separado, reduciendo la complejidad de todo el sistema y entendiendo a cada módulo como una entidad.

5.2.4 Arquitectura del software

La arquitectura del software se enfoca en "la estructura global del software y a las maneras en que esa estructura proporciona integridad conceptual a un sistema"[1], es decir, la arquitectura del software presenta la jerarquía de módulos, su forma de interactuar y la estructura de datos utilizada por ellos. La visión de la jerarquía del sistema le permite al diseñador tomarla como punto de partida para después concentrarse (este es un proceso de abstracción) en cada uno de los módulos y ahí con más detalle, continuar con el diseño.

5.2.5 Jerarquía de control

La estructura del sistema lleva consigo asociada una jerarquía de control. La jerarquía de control no ofrece los detalles sobre la secuencia que deben tener los procesos internos del sistema y si es que se deben o no de repetir un número determinado de ocasiones. En lugar de eso, la jerarquía de control permite ver el número de módulos que son llamados desde otro módulo, los datos involucrados en dichas llamadas permitiendo saber cuales módulos sí están conectados unos con otros y cuales no lo están, esto a la vez indica cual es la comunicación entre módulos, comunicación que como ya mencioné involucra transferencia de información. Normalmente se utilizan diagramas de árbol para representar la jerarquía de control.

5.2.6 Partición estructural

Se refiere a la manera en como se partirá o se dividirá el software. Esta partición puede ser tanto en forma vertical como en forma horizontal.

Cuando se parte el software de una manera horizontal se hace en secciones paralelas de izquierda a derecha donde cada una de ellas tiene una función principal del software, en cada una de estas secciones tenemos en el nivel más alto los módulos de control los cuales coordinan la comunicación y la ejecución de las diferentes funciones del programa, después hacia abajo, se tienen los módulos de entrada de datos del software, en el nivel intermedio se encuentran los módulos de procesamiento de información (también llamados módulos de transformación de información), mientras que en el nivel inferior se encuentran los módulos de salida o de presentación de resultados.

Este enfoque permite crear software que es más fácil de probar, de mantener y de ampliar permitiendo minimizar los efectos secundarios creados al realizar el mantenimiento del software. Por otro lado el realizar la partición horizontal provoca que se pasen muchos datos entre los diferentes niveles lo cual hace complicado seguir el flujo del programa, además, si se desea que el sistema responda con rapidez, éste debe esperar a que en "cascada" los datos avancen de nivel a nivel hasta llegar al nivel más bajo el cuál presentará los resultados.

La partición vertical se parece a la manera en que la mayoría de las organizaciones de negocios operan ya que en la parte superior de una partición vertical se encuentran los módulos que están encargados de realizar las tareas de control y estos módulos muy pocas veces realizan tareas de procesamiento de información, mientras que en la parte inferior de la estructura se encuentran los módulos encargados de la entrada, el procesamiento y la salida (o resultados) de la información.

La partición vertical al concentrar los procesos de entrada, de procesamiento y de salida de información en los módulos inferiores, trae como consecuencia que los cambios realizados al software tengan menos efectos colaterales o secundarios, ya que, es más probable que se realicen ajustes a ciertos procesos de información (ubicados en los módulos inferiores) que al objetivo general del software (módulos superiores), esta ventaja facilita el mantenimiento del software puesto que se sabe que los ajustes se realizarán a los módulos inferiores y no en todos los módulos.

5.2.7 Estructuras de datos

Se refiere a la representación de las relaciones lógicas de los datos individuales. La estructura de datos dicta la organización de los métodos de acceso, la capacidad de asociación y el procesamiento de la información.
Creo que el concepto antes descrito se aplica al tipo de software en donde el tiempo de acceso a los datos es crucial para la aplicación o cuando se desea que el software administre la información utilizando listas enlazadas y pilas, por ejemplo.
En la actualidad la mayoría de los lenguajes de programación populares cuentan con un mecanismo para acceder a los datos almacenados por alguno de los administradores de bases de datos existentes (por ejemplo, un mecanismo de acceso a datos se puede realizar a través de la conectividad abierta de bases de datos u ODBC por sus siglas en inglés). Esto no quiere decir que en el software que estamos diseñando no utilicemos las estructuras de datos fundamentales como lo es un vector de 1 dimensión (un arreglo de tipo numérico) o un vector de 2 dimensiones, es decir, una matriz.

5.2.8 Procedimiento del software

Mientras que la jerarquía de control únicamente define la organización de los módulos del software, el procedimiento del software se emplea para estudiar y desarrollar los detalles de la tarea asignada a cada uno de los módulos, por lo tanto cuando se diseña el procedimiento del software en un módulo en particular, se debe de incluir una indicación de los módulos que se comunican con él. Es importante mencionar que el procedimiento del software emplea el concepto de abstracción ya que se debe de realizar por capas o niveles.

5.2.9 Ocultamiento de la información

Individualmente un módulo debe encapsular u ocultar a cada uno de sus procedimientos y los detalles necesarios para alcanzar su objetivo, para que de esta manera, cuando se encuentre activo alguno de sus procedimientos ningún otro módulo pueda acceder a él. Este concepto dice que

colectivamente los módulos deben de intercambiar entre ellos sólo la información necesaria para lograr el objetivo del sistema.

El encapsular los procedimientos y los datos permite que los errores introducidos al realizar cambios al software no afecten a los demás módulos.

5.2.10 Ventajas al utilizar módulos

Todos los conceptos antes presentados apoyan el diseño y la construcción de sistemas modulares.

Un diseño modular ofrece:

- Reducción de la complejidad del software. Los módulos son más fáciles de entender ya que son subsistemas autocontenidos, es decir, que el código del módulo describe su función, garantizando su comprensión.
- Un fácil mantenimiento del sistema, una corrección a nuestro sistema deberá de involucrar a uno y en el caso extremo a unos cuantos módulos y no a todo el código del software.
- Un desarrollo de los módulos en paralelo.
- La eliminación de la duplicidad de código.
- La posibilidad de esconder o encapsular las secuencias de instrucciones de procesamiento de información.
- La oportunidad de volver a utilizar código, en ocasiones desarrollado para otros sistemas.

Pero ¿qué debemos entender por módulo?.

5.3 Módulo

Un módulo es una colección de datos y de las rutinas que actúan sobre esos datos. Un módulo también puede ser una colección de rutinas que ofrecen un conjunto cohesivo de servicios aunque no haya datos comunes envueltos en estos servicios.

Existen conceptos tales como Independencia funcional, Cohesión y Acoplamiento que sirven para crear módulos de calidad que permiten hacer más fácil su implantación.

5.3.1 Cohesión

Las instrucciones que forman parte de un módulo son únicamente las necesarias para completar una sola tarea bien definida. Es decir, no se deben de fragmentar los procesos esenciales en uno o más módulos, ni se deben de incluir instrucciones que realicen tareas que no tengan que ver con el objetivo del módulo.

5.3.2 Acoplamiento

Cada módulo debe comunicarse de una manera sencilla con otro u otros módulos, se deben de compartir el mínimo de datos entre módulos, se debe de cuidar que un módulo no modifique los datos y la secuencia lógica de instrucciones de otro módulo.

5.3.3 Independencia funcional

Los módulos del software deben crearse teniendo en mente que cada uno de ellos realizará una función única y que en la medida de lo posible, se debe evitar su interacción con los demás módulos, es decir, debemos "diseñar software de manera que cada módulo trate una subfunción específica de los requisitos"[1].

Hay que considerar lo siguiente al construir módulos:

a) Mantener cada módulo de un tamaño manejable, idealmente incluyendo una sola función. Cuando un módulo por su naturaleza es muy grande, éste debe partirse en submódulos, se recomienda también que un módulo no debe tener más de 6 submódulos con la intensión de reducir la complejidad, un módulo con demasiados submódulos puede resultar difícil de leer y de entender.
b) Poner particular atención a las interfaces críticas, es decir, hay que tener mucho cuidado con los datos y las variables de control que son pasados a otros módulos.
c) Minimizar la cantidad de módulos que necesita el programador modificar cuando se tengan que realizar cambios.
d) Mantener la jerarquía de módulos, especificada, en la jerarquía de control.

5.4 Métodos de diseño

Como ya mencioné el diseño es la etapa en la cual se deben de transformar los requisitos del software en modelos del diseño que sinteticen la estructura de los datos, la estructura de los programas, las características de la interfaz de usuario y los detalles de los procedimientos que se implantaran.

Existen diferentes métodos que permiten realizar esta transición entre requerimientos y la construcción de los diseños de los programas a desarrollar.

5.4.1 Diseño de datos

El diseño de datos es la base sobre la cual descansará todo nuestro diseño, de ahí su gran importancia. En este punto presentaré don enfoques o actividades referentes al diseño de datos; las estructuras de datos y las bases de datos.

Cuando hablo del diseño de las estructuras de datos me refiero a la correcta selección de las representaciones lógicas de los objetos de datos que han sido identificados en la fase de análisis, hay que elegir por ejemplo entre matrices, listas, pilas, colas y las operaciones sobre ellas que nos permitan tener el diseño más eficaz, por ejemplo, si un módulo debe pasar como parámetros a otro módulo una cantidad considerable de valores, entonces se puede elegir un vector o arreglo como la estructura de datos que almacene todos los parámetros y así hacer más fácil el paso de valores entre los módulos.

Por otro lado, al hablar acerca de la base de datos me refiero al almacén de datos utilizado por el software, almacén que tiene su origen en el diagrama entidad - relación y en el diccionario de datos desarrollados en la fase de análisis.

El concepto base de datos tiene en la actualidad un uso muy amplio y puede ser entendido de diferentes maneras, por ejemplo, una secretaria que tenga en fichas de cartón los teléfonos y las

direcciones de los clientes de su jefe puede argumentar que ella cuenta con una "base de datos de clientes", de igual manera, en la actualidad podemos encontrar en las bibliotecas los catálogos (en fichas de cartón) de los libros que ahí se encuentran y con ellos fácilmente encontrar el libro que buscamos, aquí también se puede hablar de una "base de datos de libros".
Como estos ejemplos hay muchos más en donde nosotros coloquialmente utilizamos el termino base de datos para referirnos a un conjunto de información que tiene la misma naturaleza y que está organizada de una manera que nos permite acceder a un elemento de ella de una manera rápida y fácil.

Por otro lado en el campo de la informática o en el del procesamiento de información por computadora, entiendo como base de datos a: una colección de registros integrados que cuentan con una sección que describe su estructura, la base de datos es una representación de la "visión" que tiene nuestro usuario de la información que él maneja cotidianamente y que en este momento nosotros estamos diseñando.

Una base de datos se compone de:

- Archivos compuestos por registros, estos archivos contienen la información del software.
- Diccionario de datos o metadatos, describe la estructura de la base de datos.
- Índices, permiten representar relaciones entre datos así como también ofrecen un mejor desempeño al acceder a los datos.

La manera en la cual se transforma el modelo entidad – relación en un diseño de base de datos es el siguiente:

a) Primero cada entidad es transformada en una tabla de dos dimensiones compuesta por renglones (que permiten representar las instancias de la entidad) y columnas (que permiten representar los atributos de la entidad).
b) Después se realiza un proceso conocido como normalización en el cual se revisa la estructura de cada una de las tablas y si es necesario se divide en dos o más tablas.
c) También los diferentes tipos de relaciones (uno a uno, uno a muchos y muchos a muchos) existentes en el modelo entidad – relación se representan mediante tablas, incluyendo en cada una de las tablas relacionadas un campo común el cual regularmente en alguna de las dos tablas es la clave principal.

5.4.2 Diseño arquitectónico

Hay que recordar que el objetivo del diseño arquitectónico es el de crear una estructura modular del sistema y representar con esto las relaciones de control entre los mismos módulos. Dicha estructura modular se puede crear utilizando el enfoque orientado al flujo de datos ya que casi todo el software se puede analizar y modelar mediante diagramas de flujo de datos, el presente diseño se ajusta a este enfoque, por lo tanto a continuación enuncio los pasos para la transformación de un diagrama de flujo de datos a un diseño arquitectónico del software.

1) Establecer el tipo de flujo de información.
2) Definir los limites del flujo de información.
3) Transformar el diagrama de flujo de datos en la estructura del sistema.
4) Definir la jerarquía de control.
5) Refinar la estructura creada o generada a partir de los cuatro puntos anteriores.

5.4.3 Diseño de interfaz

El diseño de interfaz se concentra en:

Diseño de interfaces entre los módulos del sistema

El diseño de interfaces entre módulos se realiza en función de los datos que se pasan entre ellos y de las características del lenguaje de programación que se utilizará.
Debido a que estamos transformando nuestro análisis en un diseño, debo mencionar que, las flechas que representan las entradas y salidas de información de los procesos de un DFD son transformadas aquí en un diseño de interfaz para el módulo involucrado en la transformación de dicha información.

Diseño de interfaces entre el sistema y las entidades externas

El diseño de interfaces del sistema con el mundo exterior se refiere a que se deben tomar cada una de las entidades externas representadas en el diagrama de flujo de datos identificando sus requisitos de datos y de control para posteriormente diseñar las interfaces correspondientes.

Diseño de la interfaz entre el usuario y el software

El diseño de interfaz entre el usuario y el software se refiere al diseño de pantallas, menús, botones, etcétera, que le permiten al usuario ejecutar las diferentes tareas del sistema.

Cuando se diseña la interfaz, se deben de considerar tres diferentes modelos.
El modelo del usuario, es creado para presentar el perfil de los usuarios del software, para lograr esto se debe de saber a quien va dirigido el software.
El modelo de la percepción del usuario, presenta la percepción o bien la imagen que tiene en su cabeza el usuario final del sistema.
El modelo de la imagen del sistema, es aquel que resulta de la mezcla del "aspecto que tiene el software" y la información (manuales, libros, etcétera) del sistema. Juntos "describen la sintaxis y la semántica del sistema"[1]. Cuando la imagen del sistema y la percepción que tiene el usuario coinciden, se obtiene como resultado, un usuario que esta cómodo y que utiliza eficazmente el software.

Otro aspecto importante del diseño de interfaz es el análisis y el modelado de todas las tareas que comúnmente realizan los futuros usuarios del software, posteriormente estas tareas serán clasificadas en función de su similitud para ser implantadas en la interfaz del usuario y el software.

Ahora hablaré de la entrada y de la salida de información del software. La salida puede ser un reporte, una impresión, una pantalla o un sonido. Si la información de salida o resultados está incompleta o no es clara el usuario pensará que la aplicación no funciona correctamente y tal vez decida no utilizar el software, de ahí la importancia del buen diseño de la salida del sistema.

De acuerdo a Kendall y Kendall[2] existen objetivos que la salida debe cumplir:

- Diseñar la salida para que sirva al propósito deseado
- Diseñar la salida para que se ajuste al usuario
- Entregar la cantidad adecuada de salida

[2] Análisis y diseño de sistemas, Keneth y Julie Kendall

- Asegurarse de que la salida se encuentra donde se necesita
- Entregar la salida a tiempo
- Seleccionar el método de salida adecuado

Otros aspectos importantes de la salida son:

- Presentar sólo la información relevante, de acuerdo al proceso que es llevado a cabo.
- Permitir que la salida pueda ser entendida rápidamente por el usuario, para lograrlo hay que presentar la información necesaria.
- Cuando se genera un error presentar su tipo, una breve descripción de él y evitar que la aplicación termine su ejecución en ese momento, es decir, ante la ocurrencia de un error el sistema debe permitir que se interrumpa la ejecución del software.
- Utilizar divisiones para agrupar a diferentes tipos de información, por ejemplo, hay que indicar en una factura la área referente a los datos generales del cliente, la área referente al o a los productos que él esta comprando y la área donde está el cálculo de la cantidad a pagar.
- Presentar la información lo más parecido a como la encontramos en el mundo real, por ejemplo, un reporte de ventas se representa comúnmente en forma tabular.

Al hablar de la entrada de información me refiero a las diferentes formularios o pantallas utilizados para introducir los datos al software. Si tenemos buenos medios de entrada de datos, entonces muy probablemente tendremos una mejor salida de datos.

Existen lineamientos para la creación de pantallas:

- Pantallas sencillas con sólo la información necesaria
- Presentación consistente de la pantalla (por ejemplo, el nombre y la dirección de un cliente deben estar juntas en una pantalla de captura de datos generales)
- Crear pantallas atractivas que no presenten datos “amontonados”, en lugar de esto hay que usar varias pantallas, si estamos trabajando en un ambiente gráfico de ventanas, se pueden usar efectos tridimensionales. Evitar el uso de colores fuertes que a la larga cansen la vista del usuario.
- La pantalla debe reducir la cantidad de escritura de datos por parte del usuario.
- La pantalla debe proporcionar valores predeterminados cuando sea posible (por ejemplo, ceros después del punto decimal al introducir unidades de medida o el valor mínimo de una cantidad)
- Las pantallas deben permitir que se introduzcan los datos por medio del ratón o del teclado.

5.4.4 Diseño de procedimientos

Teóricamente la especificación de procedimientos debe hacerse en un lenguaje natural como lo es el español o el inglés, pero dado que el diseño de procedimientos tiene como objetivo el especificar los detalles de los procedimientos sin “ambigüedades” el uso de un lenguaje natural no es apropiado ya que en muchas ocasiones el significado de las cosas depende del contexto en el que se encuentre.

Por la tanto, hay métodos para realizar las especificaciones de procesos, tales como la Programación estructurada, la Notación gráfica de diseño, la Notación tabular de diseño y el Lenguaje de diseño de programas. A continuación presento brevemente las técnicas que conozco en la práctica.

5.4.4.1 Programación estructurada

La programación estructurada propone un grupo de construcciones lógicas con las que podría formarse cualquier programa, la manera en que se va desarrollando el programa es de arriba hacia abajo. Las construcciones estructuradas son fragmentos lógicos que nos permiten identificar los elementos de procedimiento sin tener que leer todo para comprender una rutina o un módulo, de igual manera pueden estar anidadas unas dentro de otras lo que permite crear esquemas lógicos que a su vez son complejos.

Las estructuras lógicas a las que he hecho referencia son:

- La secuencia, esta estructura realiza los pasos indispensables del procedimiento
- La condición o selección, esta estructura ofrece la posibilidad de decidir entre grupos de secuencias sobre la base de una condición lógica que se debe evaluar
- La repetición, esta estructura permite como su nombre lo índica, repetir un grupo de secuencias.

La base de la programación estructurada es la idea de que un programa sólo debe utilizar estructuras de control tanto de entrada sencilla como de salida sencilla. Dichas estructuras constan de un bloque de código que sólo tiene un lugar en donde él debe de iniciar y sólo otro lugar en donde él debe terminar, no teniendo otras entradas o salidas.

A continuación presento algunas de las ventajas obtenidas al utilizar la programación estructurada:

- Mejora la productividad, donde se obtienen los mejores resultados es en proyectos grandes con severas restricciones y con un grado alto de complejidad.
- Mejora la comprensión del código, lo que ayuda a tener un programa con calidad.
- Un programa estructurado progresa de una manera ordenada y disciplinada de arriba hacia abajo, en lugar de estar “saltando” dentro de las diferentes partes del código de una manera casi impredecible.

Quisiera señalar algunas cosas que considero importantes; la programación estructurada no es un método para dar formato al código de programación (tabulaciones o sangrías), la programación estructurada únicamente es aplicada en el nivel de un proyecto donde se realiza el proceso de codificación del software, y finalmente, la programación estructurada no es una técnica para reducir el tiempo de programación.

5.4.4.2 Diagramas de flujo

Otra forma de describir los detalles de un procedimiento son los diagramas de flujo. Un diagrama de flujo es la representación gráfica de un procedimiento, o bien, un diagrama de flujo muestra gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema. Se dice que un diagrama de flujo representa la esquematización gráfica de un algoritmo (entiendo por algoritmo al conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema en especial).

En un diagrama de flujo se contemplan los siguientes elementos:

- Inicio del diagrama de flujo.
- Especificación de los datos de entrada.
- Operaciones a realizar con los datos o decisiones a tomar.
- Especificación de la salida (resultados).
- Fin del diagrama de flujo.

Un diagrama de flujo permite representar las tres construcciones estructuradas vistas en la sección anterior, es decir, permite representar la secuencia, la condición y la repetición.

Ventajas al utilizar un diagrama de flujo:

- Permite en forma gráfica planear las operaciones y/o decisiones de un programa antes de escribirlo.
- Representa ayuda visual para observar las correlaciones que se presentan entre operaciones o decisiones de un programa.
- Facilita la interpretación.
- Ayuda a la comunicación entre programadores.
- Forma parte de la documentación de un programa.
- Es independiente del lenguaje de programación a emplear.

5.4.4.3 Lenguaje de diseño de programas o PDL

El lenguaje de diseño de programas o PDL (por sus siglas en inglés) es conocido también como lenguaje estructurado o pseudocódigo, es un lenguaje rudimentario en el sentido de que utiliza vocabulario de un lenguaje natural (regularmente el inglés) y la sintaxis de otro, el cual regularmente es el lenguaje estructurado de programación. Originalmente fue desarrollado por la compañía Caine, Farber & Gordon y ha sido modificado substancialmente desde su primera publicación en 1975[3]. Debo comentar que en un PDL también se usa texto descriptivo para explicar sus sentencias.

Todo PDL debe cumplir con:

- Sintaxis fija para las palabras clave.
- Sintaxis libre para el texto descriptivo.
- Permitir declarar fácilmente tanto las estructuras de datos simples (un vector) como las complejas (un árbol, una pila o una lista).

Al escribir un PDL se deben considerar los siguientes puntos:

- Utilizar sentencias en el lenguaje Inglés que sean precisas al describir una operación especifica.
- Eliminar la utilización de los elementos sintácticos propios del lenguaje de programación que se vaya a utilizar en la codificación.
- Se debe escribir el PDL en un nivel bajo que permita generar el código del procedimiento de una manera automática. Si el PDL recién creado tiene un nivel alto,

[3] Code Complete, Steve Mc Connel

entonces se debe refinar, obteniendo cada vez más y más detalles hasta que sea fácil su cambio al lenguaje de programación utilizado.

Bibliografía

1. **Roger S. Pressman,** Ingeniería del Software Un Enfoque Práctico, editorial McGraw-Hill; México D.F.; 1998; capítulos 13 y 14.

2. **Kenneth E. Kendall y Julie E. Kendall,** Análisis y Diseño de Sistemas, editorial Prentice Hall Hispanoamericana, S. A.; México Edo. De México; 1997; páginas 485 - 490.

3. **David M. Kroenke,** Database Processing Fundamentals, Design & Implementation, editorial Prentice Hall; USA New Jersey; 2000; páginas 14 – 16.

4. **Steve McConnell,** Code Complete: A Practical Handbook of Software Construction, editorial Microsoft Press; USA Washington; 1993; capítulos 5, 7 y páginas 54 – 57.

5. **Ross Nelson,** Guía Completa de Visual Basic para Windows, editorial McGraw-Hill; México D.F.; 1994; páginas 349 – 356.

Capítulo 6: Implantación del proceso de asignación

6.1 Construcción de software

La tarea que le sigue al análisis y al diseño de una casa es la construcción de la misma. De igual manera en la creación de un sistema, la programación, construcción o implantación de todas las rutinas y del programa principal se llevan a cabo una vez que las etapas de análisis y de diseño ya lo permiten, menciono esto último debido a que la implantación incluye un poco de diseño detallado. Esta actividad se encuentra en el centro del proceso de desarrollo del software, después de las etapas de análisis y diseño y antes de la etapa de pruebas del sistema.

La programación es una de tres etapas técnicas necesarias para construir software, esta actividad típicamente consume de un 30 a un 80 % del tiempo del proyecto de un sistema, en ella también se genera el 75 % de los errores en pequeños sistemas, un 50 a 70 % de errores en sistemas medianos a grandes[1]. Comúnmente la gente encargada de crear sistemas siempre inicia con esta actividad olvidándose de las etapas de análisis y de diseño, lo cual considero que no es correcto pues se pasa por alto la metodología, muchas veces esto se debe a la necesidad de crear rápidamente un sistema.

Debido a que en el proceso de construcción del sistema se invierte una gran cantidad de tiempo, de que en él intervienen más de un programador y de que es aquí en donde se genera una cantidad considerable de errores, debemos de preocuparnos por aprender a construir software de la mejor manera. Para lograrlo debemos apoyarnos en técnicas que nos permitan crear código de la más alta calidad, técnicas que permitan a un grupo de programadores trabajar en equipo, que ante la salida de uno de los elementos del equipo otro pueda continuar con la o las rutinas que se le habían encargado, que después de un tiempo se pueda regresar a revisar el código y que éste sea claro, que contenga los comentarios suficientes en los lugares adecuados para que así nuestro programa sea detallado, correcto e informativo.

Entre las muchas tareas que componen la construcción del software están:

- Diseñar y construir rutinas y módulos.
- Crear tipos de datos y nombrar variables.
- Elegir estructuras de control y organizar bloques de instrucciones.
- Encontrar y corregir errores.
- Dar formato y realizar comentarios que mejoren la comprensión de nuestro programa.
- Integrar componentes de software que se hayan construido de manera separada.
- Ajustar los programas para hacerlos más pequeños y rápidos.

El proceso de construcción del software debe ser consistente con el diseño ya que éste es su guía principal, también debe ser consistente de una manera interna, en este sentido existen reglas o guías para nombrar variables, para nombrar rutinas, para aplicar formato al código y para realizar los comentarios, es decir, se deben unificar los criterios utilizados en la creación del código logrando una integridad de bajo nivel para evitar que al final nuestro software sea un conjunto de rutinas mal coordinadas con muchos estilos diferentes de codificación.

6.2 ¿Qué es una rutina?

[1] Code Complete, Steve McConnell

Una rutina es una función o un procedimiento que puede ser llamado para que realice una tarea específica.

Pero, ¿qué es una función y qué es un procedimiento? Tanto una función como un procedimiento son secciones de código con un propósito bien definido, es decir, deben de realizar una serie de procesos internos para alcanzar el objetivo para el cual fueron creados, ambos pueden recibir parámetros (valores que ellos utilizan internamente). La diferencia que encuentro entre ellos es que una función debe regresa un valor el cual puede ser un entero, un valor lógico (verdadero o falso) o un caracter por ejemplo, mientras que un procedimiento no regresa ningún valor (aunque puede presentar algún mensaje en pantalla o puede realizar la impresión de un reporte).

Pero ¿porqué el grupo de programadores debe crear rutinas? A continuación presento algunas buenas razones.

- Reducen la complejidad de los programas o sistemas, sin rutinas un sistema podría llegar a ser un listado de muchas páginas y se tendría que leer de principio a fin para entenderlo, al dividir un problema en pequeños subproblemas reducimos su complejidad (aplicamos aquí un proceso de abstracción), también el uso de rutinas ayuda al mantenimiento del sistema y reduce el tamaño de nuestro código.
- Eliminan la duplicidad de código, si creamos una rutina que nos permita abrir un archivo, el código asociado se creará una sola vez y podrá ser llamado para abrir cualquier archivo en todo nuestro sistema.
- Reducen el número de modificaciones que se deben realizar, si una rutina es utilizada en 10 lugares diferentes de nuestro sistema, basta con hacer los cambios en la rutina para que, los 10 lugares en donde utilizamos la rutina se actualicen y reflejen los cambios realizados.
- Con las rutinas resulta más fácil probar el software, podemos revisar de una manera asilada nuestras rutinas para comprobar que estas realizan correctamente su trabajo y no tener que esperar a que todas las rutinas estén terminadas para ejecutar todo el código y llevarnos la sorpresa de que nuestro sistema no trabaja como lo esperábamos.
- Reducen los efectos colaterales al realizar cambios, si tenemos aisladas nuestras secciones de código en rutinas, un cambió en una de ellas en teoría debe de tener un alcance local, es decir, sólo debe de afectar a una o a un número reducido de rutinas (aquellas rutinas que trabajan con nuestra rutina).
- Aumentan el desempeño, si el código de una rutina se ha optimizado y es utilizado a lo largo de nuestro software, todas las partes que lo utilizan se verán beneficiadas pues estarán utilizando un código que ha sido creado para ejecutarse más rápido y más eficientemente.
- Esconden estructuras de datos, resulta muy conveniente esconder los detalles de las estructuras de datos utilizadas en la implantación de una rutina, ya que no es necesario que todo el sistema las conozca.
- Promueven el hecho de volver a utilizar código, una rutina para abrir un archivo puede utilizarse en más de un sistema.

6.3 Pasos para crear una rutina

Los principales pasos para crear una rutina son:

Diseñar la rutina, el primer paso para crear una rutina es diseñarla. Antes que nada se debe revisar si es que el propósito de la rutina esta bien definido y si ésta encaja en la arquitectura, se debe

revisar si la rutina es necesaria para cumplir directa o indirectamente con alguno de los requisitos del sistema.
Después, se debe determinar si el detalle en la especificación de la rutina es suficiente para iniciar a codificarla, el diseño debe indicar:

- La información que la rutina debe ocultar
- Las entradas a la rutina
- Las salidas de la rutina, incluyendo las modificaciones a las variables globales
- Cómo es que la rutina manejara los errores que se puedan generar en su interior

Nombrar la rutina, el siguiente paso en la creación de una rutina es darle un nombre correcto, el nombre no debe ser ambiguo, si tenemos problemas al establecer el nombre de la rutina tal vez el propósito de la rutina no ha quedado claro.

Se recomienda nombrar un procedimiento utilizando un verbo seguido de un objeto, el nombre debe describir que es lo que el procedimiento hace y una operación sobre un objeto, por ejemplo: ImprimirTablaResAsignacion() o RevisarInformacionDeOrden().

Para el nombre de una función se recomienda utilizar la descripción del valor que la función debe regresar, por ejemplo: Cos(), NumLugaresDisp(), ClaveDeOpcionEducativa() o ImpresoraLista().

Se recomienda que la longitud del nombre de una rutina esté entre 20 y 30 caracteres.

Determinar la forma en la cuál la rutina se validará, es importante determinar cómo es que se validará la rutina una vez que se codifique, una forma de hacerlo es ejecutando las instrucciones línea por línea, otra es, llamando a la rutina junto con un conjunto de datos que nosotros alimentaremos (datos similares a los que la rutina utilizará cotidianamente).

Buscar los mejores algoritmos, antes de iniciar con la creación de un nuevo algoritmo es importante buscar en la literatura correspondiente si es que no existe ya un algoritmo que realice la tarea que estamos por programar, si el algoritmo ya existe, es muy probable que sea un algoritmo probado y optimizado, en este caso es recomendable tomar dicho algoritmo en lugar de invertir nuestro tiempo en desarrollar uno propio.

Transformar el algoritmo en un Lenguaje de Diseño de Programas o PDL[2]. Una vez que los pasos anteriores se han completado, se puede proceder a escribir el PDL de la rutina evitando utilizar las características específicas de un lenguaje en especial, como lo son, el manejo de apuntadores, llamadas a funciones y comandos propios.

Se debe iniciar con la descripción de la tarea que debe realizar la rutina, después se debe convertir cada uno de los elementos del algoritmo en una o varias sentencias de PDL.

Revisión del diseño de la rutina, el PDL terminado debe revisarse más de una ocasión para hacer las correcciones necesarias. Una técnica que podemos utilizar para revisar nuestro PDL es pensar que debemos de explicarlo a otra persona. Si en el proceso de revisión se encuentran errores, se deben corregir.

Codificación de la rutina, los pasos para codificar una rutina son:

[2] En el capítulo Diseño del proceso de asignación se explica qué es un PDL.

- **Se inicia escribiendo la declaración de la rutina**, se debe crear la declaración de la rutina de acuerdo al lenguaje de programación que estemos utilizando, por ejemplo en Visual Basic se hace esto con ayuda de las palabras “Function” o “Procedure”. El comentario principal que indica que es lo que realiza la rutina se debe pasar a comentario y se debe de colocar enseguida de la declaración de la rutina.
- **Escribir la primera y la última línea de la rutina,** todo el PDL debe quedar atrapado en medio de la nueva rutina y se debe pasar totalmente a comentarios. Por ejemplo, en C se indica el principio de la rutina con el signo { y el final con el signo } y se utilizan los caracteres /* y */ para colocar todo el PDL como un comentario.
- **Convertir el PDL,** línea por línea el PDL se transforma en las líneas de código o bloques de código que permiten realizar las tareas que se piden. En este momento también se crean o declaran las variables que la rutina necesita.
- **Revisar el código,** de una manera informal se revisa el código para tratar de encontrar errores en el diseño de la rutina, un problema en el diseño de la rutina puede aparecer o hacerse evidente hasta que ésta ha sido escrita, o bien, puede ser que durante la codificación se haya cometido algún error en la escritura de las diferentes sentencias de la rutina.
- **Revisar la calidad de la rutina**, la revisión se realiza para determinar si es que:
  a) Todos los parámetros de la rutina fueron utilizados.
  b) La rutina realiza solo una tarea y la realiza bien.
  c) No existen nombres de variables inadecuados y datos o variables que nunca fueron utilizados.
  d) La rutina cuenta con los suficientes comentarios que describen su funcionamiento sin ambigüedades o que indiquen porque se uso alguna instrucción o un conjunto de instrucciones en especial.

Los pasos antes presentados se deben repetir las veces que sea necesario puesto que el proceso de creación de una rutina es iterativo.

Revisar el código de la rutina de una manera formal, se debe revisar la rutina una vez escrita, se puede hacer lo siguiente:

- Si la rutina es pequeña, se pude ejecutar mentalmente alguna línea o alguna sección de código que sea de interés o que pensemos que no entregará el resultado que esperamos, este proceso se puede realizar también en papel, lo que nos lleva a realizar una “prueba de escritorio”.
- Otra manera de revisar nuestro código es compilando la rutina, la compilación de la rutina nos permite identificar variables que no fueron declaradas, código mal escrito (errores de sintaxis), inconsistencia en la utilización del nombre de una variable, etcétera.
- Una vez que se compile la rutina y que no se genere ningún error, es decir, una vez que pidamos a nuestro ambiente de desarrollo que revise nuestra rutina y que nos indique que todo se encuentra escrito correctamente, se puede abrir la rutina en el “depurador” (herramienta que forma parte del ambiente de trabajo del lenguaje de programación) para determinar visualmente y línea por línea si es que nuestro código se ejecuta de la manera en que nosotros esperamos.

6.4 Ejemplos de rutinas del sistema de asignación

La tabla 6.1 presenta el nombre y la descripción de algunas de las rutinas utilizadas en el sistema de asignación.

Rutina	Descripción
GgenerarGposDeAciertos	Función que genera los archivos por grupos de aciertos sobre la base del resultado del examen de cada sustentante.
GobtenerSiguienteOpcionValida	Función que obtiene la siguiente opción válida de todos los sustentantes del grupo de aciertos que es atendido al momento de realizar la llamada a la función.
GidentificarEmpates	Procedimiento que se encarga de identificar las opciones educativas que tienen problemas de cupo al atender al grupo de aciertos actual.

Nota: la letra "g" en el nombre de las rutinas indica que el alcance de la rutina es global, es decir, la rutina se encuentra en un módulo en donde todas las rutinas pueden llamarse desde cualquier parte del sistema.

Tabla 6.1 rutinas del sistema de asignación.

6.5 Características de las rutinas de alta calidad

En la sección anterior presenté una serie de pasos para la creación de rutinas, ahora hablare acerca de las características de las rutinas que las convierten en rutinas de alta calidad.

6.5.1 Fuerte cohesión

El concepto de cohesión se refiere a la fuerza con que están ligadas o relacionadas las diferentes operaciones internas de una rutina, operaciones que trabajan en conjunto para lograr un objetivo en especial.
Una rutina con fuerte cohesión es aquella que ha sido diseñada y programada para realizar una sola tarea, mientras que, una rutina con poca cohesión es aquella que realiza dos o más tareas. Entre más bajo sea el nivel de cohesión de una rutina más difícil será depurarla y modificarla, en este caso es mejor volver a escribir una nueva rutina que cumpla con un sólo objetivo.

6.5.2 Poca relación entre rutinas

Este concepto se refiere al nivel de comunicación entre rutinas, una vez que programemos una rutina con una fuerte cohesión, debemos lograr que las conexiones entre ella y las demás rutinas sean lo más simples posibles.
Por lo tanto, debemos preocuparnos por crear rutinas que dependan poco o muy poco de otras, en la medida de lo posible las rutinas deben ser independientes, esto se puede lograr si la relación entre rutinas es mínima.
Entre menos parámetros reciba una rutina tendrá menos relación con otras rutinas y su comunicación con ellas también será más clara o transparente.
Como ejemplo puedo mencionar a un rutina llamada Sen(), esta rutina únicamente recibe como parámetro un valor que indica el ángulo del cual se desea conocer su valor seno, esta rutina tiene muy poca relación con las demás ya que no depende su funcionamiento de otras ni tampoco ella interfiere en el trabajo de las demás.
Este ejemplo también nos habla del número de parámetros que una rutina debe recibir. Entre más parámetros reciba una rutina por parte de otra más relación tendrá con ella, esto las hará dependientes y muy probablemente sólo podrán trabajar juntas, evitando que otra rutina trabaje con

una de ellas sin tener que involucrar a la otra. Se recomienda que el número de parámetros que una rutina recibe no debe de ser mayor a 7[1].

Otra consideración existente para tener una rutina independiente es, lograr una comunicación directa y clara entre rutinas, esto se puede alcanzar si dos rutinas se comunican a través de parámetros, el caso contrario es que dos rutinas se comuniquen a través de variables globales, bajo esta situación la comunicación entre las dos rutinas no es ni directa ni clara.

Finalmente mencionaré un concepto conocido como "flexibilidad de la rutina", este concepto se refiere a la creación de rutinas que puedan ser llamadas fácilmente por otras, si lo logramos, nuestras rutinas serán flexibles ya que se podrán utilizar en diferentes situaciones o también en diferentes sistemas. Por ejemplo, la función Sen() puede llamarse desde diferentes sistemas ya que realiza un proceso en especial y requiere un sólo parámetro.

6.5.3 Tamaño de una rutina

Existen argumentos a favor de rutinas pequeñas, así como también, a favor de rutinas grandes, teóricamente el mejor tamaño de una rutina es aquel que no rebasa las 2 cuartillas, entre 66 y 132 líneas. Como dato histórico puedo mencionar a la compañía IBM que en los años 70's limitó el tamaño de las rutinas a 50 líneas[1]. En mi opinión, una rutina grande es aquella que tiene más de 200 líneas de código.

Una rutina no debe de ser ni demasiado pequeña ni demasiado grande, salvo excepciones que se deben a la naturaleza misma de la rutina.

Un programa con muchas rutinas muy pequeñas normalmente fragmenta demasiado la solución del mismo dificultando su comprensión. En el otro extremo, un programa que está compuesto tal vez de un par de rutinas de más de 200 líneas de código también resulta difícil de comprender.

6.6 Programación defensiva

Cuando programamos debemos tener claro que los usuarios finales de nuestro sistema pueden hacer cualquier acción diferente a la esperada como introducir valores que no son correctos, o bien, tratar de continuar con la ejecución del sistema sin terminar de especificar todos los datos necesarios para hacerlo.
Si el usuario realiza algo peligroso, el sistema debe estar preparado para que su ejecución no se interrumpa, o bien, para que no continúe sin haber detectado la acción peligrosa del usuario.

El concepto de programación defensiva nos dice que una rutina no debe verse afectada en su funcionamiento cuando recibe parámetros incorrectos, aunque los datos incorrectos sean proporcionados "correctamente" por una segunda rutina o por el programa principal.
Las rutinas deben estar protegidas de los datos que aparentemente no causarán un gran problema cuando sus valores sean diferentes a los esperados. Como por ejemplo, un valor igual a cero en el dividendo de una operación de división de dos números.

Pero ¿qué podemos hacer para programar defensivamente? Una alternativa es utilizar funciones que evalúen ciertas suposiciones hechas en el sistema a través de una expresión lógica y que en caso de ser falsas presenten un mensaje de alerta, o bien, terminen la ejecución del sistema.

Este tipo de función regularmente tiene dos parámetros; una expresión lógica que describe el estado de la suposición que se cree que es verdadera y un mensaje que se presenta o imprime en el caso de que dicha suposición sea falsa.
Las funciones de este tipo son útiles durante la etapa de programación y la etapa de mantenimiento del código. Durante la etapa de desarrollo evidencian suposiciones contradictorias, condiciones no esperadas y valores incorrectos pasados a las rutinas. Durante la etapa de mantenimiento, indican si las modificaciones realizadas a las rutinas han afectado su código y obviamente su funcionamiento.

Otra alternativa para tener una programación defensiva se logra si las rutinas manejan excepciones, esto se puede obtener si el código de una rutina puede responder ante la presencia de datos no esperados o de excepciones en los valores esperados. Cuando se detecta la presencia de datos no esperados, la rutina no debe terminar su ejecución, en lugar de eso debe continuar. Por ejemplo, si se desea que una rutina copie en un disco flexible un archivo de texto y el usuario no inserta el disco flexible en la unidad correspondiente, la rutina debe "atrapar" la excepción y pedir al usuario que inserte el disco para intentar copiar de nuevo el archivo.

El sistema de asignación cuenta con rutinas que manejan excepciones, por ejemplo:

La rutina que genera los grupos de aciertos está preparada para continuar su ejecución cuando el número de sustentantes en el grupo de aciertos es cero, por ejemplo, cuando se intenta generar un grupo de aciertos con los sustentantes que obtuvieron 128 aciertos y de la población de sustentantes asignables no hay ninguno con esta calificación, la rutina no detiene ahí su ejecución, por el contrario, continúa con el siguiente grupo de aciertos.

Otro ejemplo de programación defensiva del sistema de asignación se puede ejemplificar con ayuda de la rutina que obtiene la siguiente opción educativa válida (clave de plantel/carrera) de cada uno de los sustentantes asignables. La rutina comprueba que la opción obtenida existe en la tabla de opciones educativas que participan en el concurso, si la comprobación antes mencionada no es exitosa y la opción educativa no se encuentra dentro del grupo de opciones válidas, la rutina presenta un mensaje al usuario especificando el tipo de error que se ha generado y enseguida termina su ejecución al tiempo que envía como valor de regreso a la rutina que la llamó un código de error que corresponde a un error crítico del sistema.

6.7 Cómo usar los parámetros de una rutina

A continuación presento algunos consejos sobre la forma en que se deben utilizar los parámetros de una rutina:

- Cuando se pasen los parámetros a una rutina, éstos deben tener los mismos tipos de datos que los establecidos en la definición de la misma. Por ejemplo, sí la función Sin() en su definición tiene un sólo parámetro del tipo flotante (entero con fracciones), entonces no se le debe pasar nunca un número de tipo caracter.
- Los parámetros en la definición de la rutina deben escribirse siguiendo la secuencia de operaciones que una rutina en lo general realiza; datos de entrada, datos que sufren cambios y datos que son enviados cómo resultado.
- Si dos o más rutinas usan una serie de parámetros similares, entonces, éstos deben de colocarse en un orden consistente.

  Por ejemplo, si una función regresa un número determinado de caracteres de una cadena de texto a partir del caracter situado más a la izquierda y, una segunda función regresa también un número determinado de caracteres de una cadena de texto

pero a partir del caracter situado más a la derecha, entonces, ambas declaraciones de rutinas deben ser consistentes en el número de parámetros y en el orden en que ambas los reciben, a continuación presento las posibles definiciones de estas dos rutinas:

```
CadCarIzq(lcCadenaDeTexto,lnNumCaracteres)
CadCarDer(lcCadenaDeTexto,lnNumCaracteres)
```

En ambos casos lcCadenaDeTexto es la cadena de la cual se desea obtener un número determinado de caracteres y lnNumCaracteres indica el número de caracteres que se desean obtener.

- No modificar los parámetros que son pasados a una rutina, para evitarlo se puede utilizar variables locales. Esta recomendación se hace porque los parámetros deben de conservar su valor original durante toda la vida de la rutina, para que puedan ser utilizados en todo momento, esto evita que se generen errores al tratar de utilizar un parámetro pensando que conserva su valor original después de que éste ha sido modificado previamente.

6.8 Uso de nombres correctos para las variables de una rutina

Un aspecto muy importante en la programación es la elección correcta de los nombres de las variables que utilizamos en las rutinas. Los nombres deben ser:

- Entendibles
- Fáciles de recordar
- Apropiados al contexto de la rutina

Tal vez la consideración más importante acerca de los nombres de las variables es que éstos deben de describir adecuadamente la entidad que representan. Por ejemplo, en NumeroDeFolio y ClaveDeOpcionEducativa, resulta muy fácil identificar que es lo que cada variable representa.

Debemos tener cuidado de no utilizar en los nombres de variables palabras que expresen una acción sobre ellas, en lugar de eso, debemos crear nombres de variables pensando en su función dentro del problema que estamos resolviendo. Por ejemplo, una variable con un nombre orientado a una acción es ClaveDeSiguienteOpcionEducativa, en lugar de utilizar ese nombre podemos llamar a nuestra variable ClaveDeOpcionEducativa, este último nombre de variable no incluye una acción sobre la misma.

El tamaño del nombre de la variable es muy importante, una variable con un nombre demasiado largo resulta difícil de recordar y de escribir repetidamente en nuestra rutina, en el otro extremo un nombre demasiado pequeño en ocasiones no dice nada acerca de la entidad que representa, además se puede confundir fácilmente con otra entidad.
Por ejemplo, un nombre de variable muy largo es NumeroDeOpcionesEducativasDelConcurso2001, mientras que, NoOpEd es muy pequeño, un nombre adecuado podría ser NumOpcionesEdu.

Si en nuestras rutinas tenemos nombres demasiado largos o cortos, debemos revisarlos para encontrar mejores nombres, nombres que nos permitan entender fácilmente las entidades que las variables representan.

El nombre de una variable debe ser tan largo o corto como se requiera. Existe una recomendación respecto al número de caracteres del nombre de una variable; el tamaño debe encontrarse dentro del rango de 8 a 20 letras[1].

Cuando el nombre de la variable resulta demasiado largo se puede reducir su tamaño:

- Eliminando palabras que no son necesarias
- Utilizando acrónimos o sinónimos
- Definiendo más de un criterio para abreviar palabras, como por ejemplo, eliminar las vocales de una palabra, excepto cuando la palabra inicia con una de ellas, manteniendo las primeras y las últimas letras de una palabra o eliminando sufijos que no son necesarios.

Es muy importante tener convenciones para nombrar las variables, esto ayuda a que un grupo de programadores pueda trabajar separadamente siguiendo los mismos criterios para nombrar las variables de sus rutinas y al final, ellos pueden conjuntar sus diferentes rutinas para formar un nuevo sistema, dicho sistema será estructurado y consistente facilitando su futuro mantenimiento.

Hay que evitar utilizar nombres de variables en donde:

- Las abreviaciones utilizadas se confundan con palabras reservadas o con el nombre de una función propia del lenguaje de programación, por ejemplo: los nombres de variables Sum (abreviación de suma) o False (abreviación de Falla en la segunda entrega).
- Los nombres de dos variables sean similares, por ejemplo: los nombres de variables CP (abreviación de Código de Página) y CodPost (abreviación de código postal).
- Los nombres de variables sean muy similares pero representen a entidades diferentes, por ejemplo: los nombres de variables NumReg (abreviación de número de registro) y NumRegi (abreviación de número de región).
- Al utilizar números para nombrar variables se llegue al dilema de tratar de entender qué es lo que se está realizando, por ejemplo:

  If NumReg1 > NumReg2 Then

  ¿Cuál es la diferencia entre NumReg1 y NumReg2? Lo único que cambia es el último caracter, deben de existir un par de mejores nombres para estas dos variables.

Las convenciones permiten que un grupo de programadores distinga entre variables que almacenan información de alcance local, modular o global en una rutina. También ayudan a identificar el tipo de dato que almacena la variable, permitiendo distinguir entre variables de tipo numéricas, caracter, lógicas, etcétera.
Por ejemplo, tenemos la variable llamada giNumOpcionesEducativas, el primer caracter o letra g indica que se trata de una variable de alcance global, mientras que el segundo caracter, indica que el valor que la variable almacena es un número entero (la i proviene de integer).
Cada lenguaje de programación tiene guías que indican las convenciones que se recomiendan para nombrar variables, así que es altamente recomendable leer la literatura correspondiente.

Para ejemplificar lo antes expuesto, a continuación presento un fragmento de código de la rutina **"***gIdentificarEmpates"*

****-- Se abre el archivo de demanda de opciones educativas del grupo de aciertos actual con el "Alias" DEM***

```
        gAbrirArchivo(gaParSistema[lnBaseDeDatos] + [DEMANDA],[DEM],.F.,.F.,5)
          Go Top In [DEM]
*--       Se repite mientras no se alcance el fin de archivo de la tabla DEMANDA
          Do While .Not. Eof([DEM])
*--               Obtenemos la clave de la opción educativa actual, así como también, su demanda total
                  lnDemandaOpcionEducativa  = DEM.Tot_Dema
                  lcCveOpcionEducativa          = DEM.Cve_Opc
*--               Buscamos esta opción  en la tabla de opciones educativas
                  Select [OpcEdu]
                  If Seek(lcCveOpcionEducativa) Then
*--                       Obtenemos la oferta o lugares disponibles actuales  de la opción educativa
                           lnOfertaEducativa  = OpcEdu.Cup_Disp
*--                       Si la demanda de la opción educativa es mayor que su oferta, entonces,
*--                       hay un empate
                            If lnDemandaEducativa > lnOfertaEducativa Then
*--                                 Se agrega un registro más a la tabla de empates del grupo actual de
aciertos
                                    gGeneraEmpate(lcCveOpcionEducativa,
                                    lnDemandaEducativa,lnOfertaEducativa)
                  EndIf
                  Else
*--                       Se presenta al usuario la descripción de la excepción ocurrida,  se regresa
*--                        un valor  lógico igual a falso y se termina la ejecución de la rutina
                          lcMensaje = [No es posible obtener la oferta educativa de la clave]
                                    +  lcCveOpcionEducativa
                          MessageBox(lcMensaje,0+16,[Error])
                          Return .F.
                  EndIf
          EndDo
```

En el segmento de código presentado se puede observar lo siguiente:

- Los nombres de variables y funciones de alcance global inician con una letra g, mientras que, las variables de alcance local (dentro de la rutina) inician con una letra l. Esto permite identificar fácilmente el alcance de cada variable dentro de todo el sistema.
- Los tipos de datos de las variables se indican con una letra al principio del nombre de la misma, por ejemplo, las variables de tipo caracter tienen una letra c al inicio.
- El formato del código ofrece claridad al apoyarse en tabuladores, las palabras reservadas inician siempre con una letra mayúscula, se combinan mayúsculas y minúsculas en los nombres de variables y palabras reservadas, también se utilizan saltos de línea y comentarios en secciones importantes de la rutina.
- En caso de presentarse una excepción, el sistema informa al usuario y regresa un valor lógico igual a falso para que entonces el sistema realice las acciones pertinentes.
- La comunicación entre rutinas es clara, como se observa en la llamada a la rutina ***gGeneraEmpate*** en donde se utiliza un número de parámetros aceptable.

Es muy importante adoptar una serie de convenciones cuando programamos nuestros sistemas, convenciones referentes a cómo nombrar las variables de nuestras rutinas, las rutinas, cuáles son las características del formato que utilizaremos, etcétera.

Se debe generar un documento con todas estás convenciones, dicho documento debe ser del conocimiento de todos los programadores que intervengan en el desarrollo del nuevo sistema o de aquellos que tengan encargada la tarea de darle mantenimiento a uno existente.

Bibliografía

1. **Steve McConnell,** Code Complete: A Practical Handbook of Software Construction, editorial Microsoft Press; USA Washington; 1993; capítulos 4, 5, 6 y 9.

2. **Ross Nelson,** Guía Completa de Visual Basic para Windows, editorial McGraw-Hill; México D.F.; 1994; páginas 357 – 371.

3. **Robert L. Kruse,** Estructura de Datos y Diseño de Programas, editorial Prentice-Hall Latinoamericana, S. A.; México Edo. De México; páginas 8 – 20.

Capítulo 7: Pruebas realizadas al sistema

7.1 Pruebas generales

El objetivo de realizar pruebas es el de determinar si nuestro sistema trabaja de acuerdo a lo planeado, durante esta etapa se pueden introducir valores similares a los que normalmente utilizará nuestro sistema, o bien, se pueden introducir valores no esperados para observar cómo es que responde el sistema, si es que detecta los datos incorrectos y ofrece opciones para continuar con su ejecución, obviamente sin dejar de funcionar.

Existen diferentes momentos en los que se realizan las pruebas al sistema, las primeras se llevan a cabo cuando se terminan de programar las diferentes rutinas, de manera independiente una a una son probadas. Después cuando se tiene un grupo de rutinas que trabajan entre ellas, se prueban para observar si es que en conjunto logran alcanzar los objetivos (intermedios) para los cuales fueron programadas. Finalmente cuando el sistema está completo, se realizan pruebas que procesarán de principio a fin toda la información, se observa el comportamiento completo del sistema y se revisan los resultados intermedios y finales.

Las primeras pruebas al sistema de asignación iniciaron desde la etapa de implantación o programación de las diferentes rutinas y módulos, después cuando se concluyó el sistema se realizaron otras pruebas finales y de puesta a punto. Entre las pruebas realizadas están:

- Ejecución aislada de las rutinas utilizando valores creados y proporcionados por el equipo de desarrollo.
- Ejecución de las rutinas utilizando valores diferentes a los esperados durante la ejecución normal del sistema.
- Repetición del proceso de asignación del año 2000.
- Ejecución del sistema en paralelo (en diferentes equipos) para determinar en cuál se lograba un tiempo de respuesta menor.
- Ejecución de las rutinas que resolvían el mismo problema, pero que se habían implantado siguiendo diferentes algoritmos, esto permitió determinar cuál era la mejor solución (menor cantidad de líneas, fácil depuración o seguimiento y mejor tiempo de respuesta).
- Ejecución de los diferentes reportes emitidos después de concluido el proceso de asignación, aquí se hace una revisión de los datos calculados, ortografía y formato.

Además de estas pruebas se realizaron 2 más que a mi punto de vista arrojaron datos muy interesantes.

Una prueba consistió en “ampliar” el cupo o lugares de las opciones educativas cuando en un cierto grupo de sustentantes (en donde todos tienen el mismo resultado de examen), la demanda de lugares rebasa a la oferta de la opción educativa.
La segunda prueba consistió en no ampliar ningún cupo de las opciones educativas ante la misma situación descrita anteriormente.

La falta de lugares en la oferta de las opciones educativas se presenta cuando el sistema se encuentra atendiendo a un número determinado de sustentantes (que tienen la misma calificación de examen) y una parte de ellos intenta obtener un lugar en la misma opción educativa, sólo que en ese preciso momento, ya no hay suficientes lugares para permitir que todos ingresen sin excepción a dicha opción educativa.

Al grupo de sustentantes que intentan ingresar a la opción educativa, se le llama demanda mientras que, al número de lugares disponibles al momento de atender a los sustentantes, se le llama oferta. Debo hacer notar que esta oferta no es el total de lugares que originalmente la opción educativa ofreció, ya que seguramente antes de llegar a este punto otros sustentantes han logrado ingresar a ésta misma opción provocando una disminución en su oferta original.

A continuación presento los resultados de las dos pruebas antes mencionadas.

La distribución de las poblaciones en ambas pruebas se presenta en la tabla 7.1.

Población	% del total de sustentantes	Participan en la prueba al proceso de asignación
ASIG	54.04	Sí
<31	01.82	No
NP	03.35	No
SC	12.14	No
UNAM	28.65	No
Total	100.00	No

Tabla 7.1 Distribución de poblaciones para la etapa de pruebas al sistema.

En donde:

ASIG son los sustentantes que cumplieron con todos los requisitos establecidos en la convocatoria del concurso.
< 31 son los sustentantes que no obtuvieron 31 o más aciertos en el examen.
NP son los sustentantes que no presentaron el examen.
SC son los sustentantes que no obtuvieron su certificado de secundaría antes de la fecha establecida en la convocatoria y en el instructivo del concurso.
UNAM son los sustentantes que presentaron el examen de la UNAM.

7.2 Resultado de la prueba “ampliando” todos los cupos ante la falta de lugares en la oferta de las opciones educativas

En esta prueba, cuando se presentaba un empate automáticamente se ampliaba el cupo de la opción educativa involucrada para permitir que todos los aspirantes empatados obtuvieran un lugar en ella.

La tabla 7.2 presenta el porcentaje total de sustentantes asignados y no asignados a una opción educativa con respecto a la población total de sustentantes que participaron en el proceso de asignación.

Población	%
OUL	95.89
NUL	04.11
Total	100.00

Tabla 7.2 Sustentantes asignados y no asignados, prueba ampliar siempre

Donde:

OUL son los sustentantes que obtuvieron un lugar en alguna de las opciones educativas.
NUL son los sustentantes que no obtuvieron un lugar.

La tabla 7.3 presenta el porcentaje de ocupación de la oferta educativa:

Oferta educativa[1]	%
Utilizada	73.62
Disponible	26.38
Total	100.00

Tabla 7.3 Porcentaje de ocupación de la oferta educativa, prueba ampliar siempre

La tabla 7.4 presenta el número de escuelas y la cantidad de lugares que tuvieron que aumentar en su oferta educativa, para permitir que todos los sustentantes empatados obtuvieran un lugar en ellas.

Número de lugares ampliados	Número de opciones educativas que ampliaron su cupo	% del total de opciones educativas
10	51	8.70
20	17	2.90
30	11	1.88
40	8	1.37
50	6	1.02
70	3	0.51
90 a 95	1	0.17
Más de 95	0	0.00

Tabla 7.4 Porcentaje de escuelas que tuvieron que ampliar lugares

La tabla 7.5 presenta la distribución después de la prueba de los sustentantes por institución educativa.

Institución	% de ocupación[2]
COLBACH	19.5
CONALEP	14.8
DGB	01.3
DGETA	00.2
DGETI	23.8
IPN	14.6
SECYBS	25.2
UAEM	00.6
Total	**100.0**

Tabla 7.5 Distribución de poblaciones por institución, prueba ampliar siempre

A partir de los resultados antes presentados puedo afirmar que:

[1] Esta oferta educativa no incluye las opciones educativas de la UNAM.
[2] Respecto al total de la oferta de todas las instituciones que participaron en la prueba

- El sistema funciona correctamente cuando amplia lugares en las opciones educativas.
- Un alto número de sustentantes obtiene un lugar en la educación media superior.
- El porcentaje de lugares disponibles después del proceso de asignación, es mayor que el porcentaje de sustentantes que no obtuvieron un lugar.
- El número de opciones educativas que tuvo que ampliar más de una clase de 50 alumnos, es apenas del 1.02 %.
- La distribución de la población de sustentantes asignados y que capto cada institución corresponde con el comportamiento histórico.

Sin embargo hay que remarcar que en esta prueba la UNAM no fue incluida.

7.3 Resultado de la prueba "cerrando" todos los cupos ante la falta de lugares en la oferta de las opciones educativas

En esta prueba, cuando se presentaba un empate automáticamente se cerraba la opción educativa involucrada, esto provocaba que se tuviera que obtener la siguiente opción válida de todos los aspirantes involucrados en el empate de dicha opción educativa y que se volviera a calcular la demanda del grupo actual de aciertos**.**

La tabla 7.6 presenta el porcentaje total de sustentantes asignados y no asignados a una opción educativa, respecto a la población total de sustentantes que participaron en el proceso de asignación en ésta prueba.

Población	%
OUL	94.85
NUL	05.15
Total o ASIG	**100.00**

Tabla 7.6 Sustentantes asignados y no asignados, prueba cerrar siempre

La tabla 7.7 presenta el porcentaje de ocupación o utilizado de la oferta educativa:

Oferta educativa[1]	%
Utilizada	72.82
Disponible	27.18
Total	**100.00**

Tabla 7.7 Porcentaje de ocupación de la oferta educativa, prueba cerrar siempre

La tabla 7.8 presenta el porcentaje de opciones con cierto número de lugares disponibles resultado de cerrar ante el empate (en cierto momento del proceso de asignación la demanda fue mayor a su oferta).

Lugares disponibles	% de opciones educativas
1 a 20	35.67
21 a 50	02.56
51 a 100	00.85
Más de 100	00.00
Total	**39.08**[3]

Tabla 7.8 Lugares disponibles y porcentaje de instituciones que cerraron ante la falta de lugares.

La tabla 7.9 presenta la distribución de los sustentantes por institución educativa

Institución	% de ocupación[2]
COLBACH	19.9
CONALEP	15.4
DGB	01.3
DGETA	00.2
DGETI	24.2
IPN	13.8
SECYBS	24.6
UAEM	00.6
Total	**100.0**

Tabla 7.9 Distribución de poblaciones por institución, prueba cerrar siempre

De las tablas antes presentadas rescato los siguientes datos:

- El sistema funciona correctamente cuando cierra las opciones educativas ante un empate.
- El porcentaje de opciones educativas que tuvieron que cerrar ante la falta de lugares fue de 39.08 %.
- Únicamente el 2% de las opciones educativas no completó su último grupo de clase.

En general los resultados de ésta segunda prueba no cambiaron sustancialmente con respecto a la primera.

Con esto queda concluido el proceso de pruebas realizadas al sistema de asignación.

[3] El 60.92 restante de opciones educativas no se vieron involucradas en cerrar

Bibliografía

1. **Steve McConnell,** Code Complete: A Practical Handbook of Software Construction, editorial Microsoft Press; USA Washington; 1993; páginas 68 – 70.

2. **Robert L. Kruse,** Estructura de Datos y Diseño de Programas, editorial Prentice-Hall Latinoamericana, S. A.; México Edo. De México; páginas 20 – 26.

Capítulo 8: Conclusiones y recomendaciones

8.1 Conclusiones generales

A continuación presento las conclusiones a las que he llegado después de terminar el presente trabajo de tesis:

- El desarrollo de un sistema de información como el Sistema del Proceso de Asignación requiere de un verdadero enfoque metodológico para su creación (técnicas de análisis, de diseño, de pruebas, etcétera). No opino que sea correcto pensar que con sentarse e iniciar la programación del software se logrará tener un sistema con la calidad que se necesita, ya que el sistema debe de satisfacer las expectativas educativas, políticas y sociales del concurso de ingreso.

- Los primeros pasos que se dieron en la creación del sistema fueron muy importantes, realizar un buen proceso de análisis y de diseño contribuyó a tener una fase de programación rápida y congruente con los requerimientos del proyecto, evitando retrasos y resolviendo el problema acertadamente. Lo anterior me ayudó a reafirmar las enseñanzas que me fueron transmitidas en la facultad de ingeniería, como por ejemplo, "nunca se debe de menospreciar ni dejar de realizar en cualquier proyecto importante de desarrollo de sistemas de información, las actividades previas a la programación (análisis y diseño)", he podido comprobar que dichas actividades son los cimientos del sistema, es tiempo invertido que al final se ve recompensado con un sistema que trabaja y resuelve el problema para el cual fue creado de una manera correcta, libre de errores y que es desarrollado dentro del tiempo y con el presupuesto asignado a él.

- En el nivel más bajo del desarrollo del sistema están las rutinas, el programa principal y la base de datos del sistema, todos estos elementos deben de ser bien analizados, diseñados, desarrollados y probados antes de ser puestos a trabajar.
  En este sentido he comprendido que no sólo hemos trabajado con datos y líneas de código, la información y las rutinas para mí han cobrado una importancia muy grande ya que, si el sistema de asignación no trabaja correctamente, el resultado no sólo se debe ver como datos incorrectos y rutinas que no trabajan bien, el resultado se traduce en afectar el futuro académico de miles de jóvenes de la Zona Metropolitana de la Ciudad de México, resulta entonces evidente la magnitud de la alta responsabilidad que tenemos en nuestras manos como equipo de desarrollo, por eso cada rutina, cada dato, cada línea de código es sumamente importante ya que es mucho lo que esta en juego.

- Cuando se trabaja con por lo menos otra persona escribiendo el código de un sistema, resulta muy importante tomar acuerdos, seguir pequeñas reglas y nomenclaturas para nombrar rutinas, módulos y variables, así como también, aplicar un formato estándar a las diferentes líneas de código; aplicando las mismas reglas a tabulaciones, saltos de línea y comentarios.
  Lo anterior provoca que todo el grupo de desarrollo trabaje siguiendo las mismas reglas facilitando a su vez el trabajo en equipo, la integración final de los diferentes elementos y el mantenimiento del sistema.

- Aunque en muchas ocasiones se dice que las técnicas de análisis y de diseño clásico, como el enfoque estructurado, tienen muchos puntos desfavorables, he podido comprobar que estas técnicas son muy poderosas ya que permiten resolver problemas tan diversos e importantes como el proceso de asignación, independientemente del lenguaje de programación en el que se desee implantar el sistema final. Hay que pensar en el número de sistemas que desde la

aparición de estas técnicas, y hasta el día de hoy, se han desarrollado con éxito siguiendo sus conceptos.

- Actualmente las características de las computadoras y de las herramientas de desarrollo de programación ofrecen la posibilidad de que una computadora personal o PC (por sus siglas en inglés) pueda ejecutar el proceso de asignación, evitando instalaciones o configuraciones de hardware especiales.
  Debo aclarar que esta PC tiene una memoria de acceso aleatorio o RAM mayor a la que de fabrica se incluye, que cuenta con un microprocesador que tiene una velocidad cercana a 1 GHz y que además la herramienta de desarrollo utilizada es muy completa y robusta, respaldada por Microsoft, una de las empresas más exitosas en el mundo del desarrollo de software.

- Personalmente siento que el participar en este proyecto ha contribuido con mi formación profesional, misma que se inicio en la facultad de ingeniería. Puedo decir que, en la facultad de ingeniería recibí la teoría acerca del ciclo de vida de un proyecto de información, técnicas de programación y de la creación de bases de datos, entre otras muchas cosas que al final, me permitieron afrontar exitosamente un proyecto de esta magnitud.
  Mucha es mi ganancia después de combinar la teoría con la práctica, de obtener experiencia laboral (tan necesaria para escalar posiciones dentro de una empresa), así como de recibir también la experiencia de las personas que estuvieron a mi lado.

- Aunque creo que la facultad de ingeniería no me dio toda la experiencia necesaria para un proyecto como este, si me dio lo suficiente para afrontarlo, permanecer en él y ser una pieza importante.

- Al participar en este proyecto me parece que cumplo con uno de tantos objetivos de la facultad de ingeniería que dice más o menos así; “formar ingenieros que sirvan y contribuyan al desarrollo del país”.

8.2 Recomendaciones

- El sistema no cuenta con un mecanismo que permita regresar a su estado original una solución de empate, es decir, una vez que el usuario ha elegido y confirmado la solución de un empate, ésta no se puede cancelar y regresar a su valor anterior. Una mejora en el sistema se puede lograr si es que se incluye un grupo de rutinas que permitan regresar a su estado original una solución de empate que ha sido tomada incorrectamente, ofreciendo con esto, mayor flexibilidad al sistema.

- Es recomendable extender la nomenclatura de los nombres de variables y de las funciones para lograr nombres más claros, por ejemplo, la segunda letra con la que inicia el nombre de cada una de las variables indica el tipo de dato que ésta almacena, actualmente no se puede distinguir una variable que almacena un número entero sin decimales de otro que si lo hace, ya que en ambos casos se utiliza la letra “n” que indica que se trata solamente de un número.

- En general más de un algoritmo se puede mejorar, como por ejemplo, toda la rutina *gGeneraGruposDeAciertos* se puede reemplazar por código que genere en memoria uno a uno los grupos de aciertos y no crear un archivo “temporal” para cada grupo de aciertos.

- Recomiendo para la siguiente versión del sistema, aprovechar las características que ofrece el administrador de bases de datos utilizado en la versión actual, características tales como; vistas

de datos, procedimientos almacenados, “desencadenantes” o “triggers”, comandos del Lenguaje Estructurado de Consulta o SQL (por sus siglas en inglés), reglas de integridad referencial y un mejor manejo de errores, errores generados por una acción realizada sobre la base de datos. Todo esto con la finalidad de ayudar a reducir las líneas de código y al mismo tiempo mejorar el tiempo de respuesta del sistema.

- Para ofrecer una mejor apariencia en el sistema recomiendo generar una aplicación que maneje una pantalla principal del tipo Interfaz de Múltiples Documentos o MDI (por sus siglas en inglés) como la que es utilizada por Microsoft Word.
  Dicha interfaz debe incluir una barra de menús, para que con ayuda de ella se pueda acceder a cada una de las funciones o características del sistema, al trabajar de esta manera se presentarán dentro de la pantalla principal cada uno de los formularios del sistema con lo que se logrará una presentación más profesional del mismo.

Recomendaciones para mejorar el diseño y la interacción del usuario con los formularios del sistema:

- Una mejor combinación de los colores de los formularios (incluyendo texto e imágenes).
- Eliminación de datos que no resultaron útiles para los usuarios en los reportes del sistema.
- Utilizar una sección determinada del formulario principal para que muestre el porcentaje de avance del calculo de la demanda del grupo de aciertos que se atiende actualmente.
- Incluir algunas de las opciones de impresión de los programas de Windows, al formulario de impresión de reportes, para que por ejemplo, el usuario pueda especificar el número de copias del reporte que desea imprimir.
- Incluir un formulario que nos permita consultar por opción educativa el porcentaje de cupo alcanzado en ella al momento de realizar la consulta.