UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Unidad Académica de los Ciclos Profesional y de Posgrado del Colegio de Ciencias y Humanidades

Instituto de Investigación en Matemática Aplicadas y Sistemas

UNA METODOLOGÍA DE ANÁLISIS Y DISEÑO CONDUCENTE AL MANTENIMIENTO FLEXIBLE DE PAQUETES EN PC

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

MARÍA DE LOS ÁNGELES SUMANO LÓPEZ

ABRIL DE 1996

a mi esposo
Juan Manuel Fernández Peña
por su total apoyo

a mis hijos
Pablo, Raúl y Diego

a mis padres
María Teresa y Salvador

a mis hermanos
Rita, Héctor, Tere, Lucy, Mario y Laura

a
Doña Eloisa Peña de Fernández

AGRADECIMIENTOS

Al director de este trabajo

Dr. Vladimir Estivil Castro

A mis sinodales

Dra. Hanna Oktaba

Dr. Christian Lemaitre León

M.C. Guadalupe Ibargüengoitia González

M.I. María de Luz Gasca Soto

CONTENIDO

LISTA DE FIGURAS

INTRODUCCIÓN.

La calidad de los productos de software es un tema que ha estado vigente por largo tiempo y que actualmente ha tomado mayor auge [PRE93]. En nuestro país son pocos los productos de software que se desarrollan para su comercialización masiva, y de ellos, muchos ofrecen niveles de calidad poco uniformes.

Por otro lado, dada la situación económica que atraviesa el país, la gran cantidad de egresados de las diferentes licenciaturas e ingenierías supera la oferta de empleo en las diversas empresas e instituciones.

Dados la situación anterior y el auge de las Computadoras Personales (PC) en todos los ámbitos, una opción podría ser desarrollar software de manera independiente, para lo cual es necesario adoptar una forma sistemática de desarrollo de software de buena calidad.

En el desarrollo profesional de la autora, se ha acumulado experiencia en la elaboración de sistemas tanto en el sector público como en el privado, utilizando desde macrocomputadoras hasta PC. La variedad de aplicaciones desarrolladas va desde sistemas de procesamiento de datos hasta Sistemas de Información Geográfica y Multimedios. Aunque al principio el desarrollo del software era más bien empírico, poco a poco se fueron viendo las bondades de la teoría aplicada a los problemas y también se fue logrando una combinación de intuición, teoría y práctica.

Aunque la aplicación de metodologías de desarrollo de software ha sido de gran utilidad para un mejor resultado en el producto, no se ha logrado completar un producto con un solo enfoque, sino más bien ha sido una combinación de varios de ellos lo que ha permitido cumplir con los retos de cada proyecto. Lo anterior hace sospechar que una metodología "clásica" por si sola es ineficaz, porque los diferentes enfoques de desarrollo han sido pensados para aplicaciones con ciertas características, generalmente para problemas muy grandes y con calendarios amplios para terminarlos, proyectos en los cuales es posible utilizar un modelo de desarrollo en cascada.[1]

[1] En diversos comentarios publicados en revistas como IEEE Software se aprecia la conveniencia de emplear diversas herramientas, ya que cada una tiene su enfoque, resultando que se complementan entre sí. Por otra parte, se han realizado algunos estudios que muestran que muchos de los métodos más publicitados se desarrollan en ambientes académicos, o al menos muy controlados, por lo que resulta difícil pasarlos a la realidad (ver [POT93]).

Ahora bien, la mayoría de las aplicaciones que se desarrollan en México, por lo menos en provincia, son para equipos pequeños[2] con necesidades muy variables y con clientes dispuestos a pagar poco, pero exigiendo cambios constantemente. Aunque podría argumentarse que ya existe un contrato formal y que hay que cumplirlo, la situación real de competencia hace que el desarrollador tenga que permitir la situación de indecisión del usuario. El trabajo aquí presentado mostrará que las metodologías actuales (Ejemplos: Flujo de Datos [YOU90], Orientación a Objetos [COA90a, COA90b]) resultan costosas en todo el ciclo de vida del sistema y por ello son inadecuadas para quien desea atender este tipo de mercado.

Por otro lado, el método propuesto para el desarrollo de software en PC busca que desde el principio del ciclo de vida del desarrollo del software se maneje y controle la calidad y que, además, se genere un producto de fácil mantenimiento.

Lo que se quiere es mostrar que, conjuntando la experiencia y diferentes conceptos de Ingeniería de Software e Inteligencia Artificial, se puede llegar a establecer una forma metodológica y sistemática de trabajo que permita al desarrollador realizar software flexible de calidad en un tiempo corto.

La estructura del trabajo es la siguiente:

Primero se listan los conceptos de calidad, estándares y la forma de medirlos; se muestra cómo uno de los métodos más utilizados hasta ahora (Flujo de Datos) no ha resultado precisamente el más adecuado en la búsqueda de la calidad en el tipo de proyectos del entorno del desarrollador de sistemas pequeños, pero de alta flexibilidad[3].

A continuación se explicarán una serie de conceptos de la teoría de Ingeniería de Software, Bases de Datos e Inteligencia Artificial que han permitido apoyar la metodología propuesta y constituyen elementos integrados en dicha metodología.

En el capítulo tres se explicará con detalle la metodología propuesta, que hace uso en el análisis de: la secuencia de pasos de Flujo de Datos, guiones (scripts) y pruebas de integración Orientada a Objetos. En el diseño utiliza Redes Semánticas, Diccionario de Datos y Entidad-Relación. En la implantación se dan sugerencias de cómo traducir en dos metodologías de programación. Y se explica cómo se llevan a cabo las pruebas y el mantenimiento.

---

[2] Se cuenta con equipos desde 286 con 512 Kb de RAM hasta un máximo de 486 con 4 Mb de RAM; son raros los casos con un equipo más grande y éstos, por lo general, los utilizan en aplicaciones de Ingeniería o Arquitectura.

[3] Sistemas como nómina, contabilidad, punto de venta o control de personal, desarrollados para empresas pequeñas y que requieren cambios constantemente.

Dado que la metodología propuesta es resultado de la experiencia en el desarrollo de software, se proponen, en el capítulo cuatro, dos tipos de experimentos para probar su efectividad:

- Estadístico.- Donde se muestra y compara la aplicación de los componentes de la metodología tomando en cuenta varios factores aplicados a una serie de proyectos realizados con anterioridad.
- Usabilidad.- En esta parte se aplicará la metodología en varios grupos de trabajo y se realizarán encuestas sobre la facilidad y efectividad del uso de la metodología propuesta.

Además, en éste mismo capítulo se propone una forma de prueba estadística a futuro que permitirá evaluar la metodología propuesta.

Finalmente, en el capítulo cinco, se realizará una evaluación de los resultados obtenidos para concluir sobre la efectividad de la metodología propuesta.

I. ESTADO ACTUAL DEL DESARROLLO DEL SOFTWARE.

El grado de madurez de las actividades de desarrollo de software es resultado de muchos factores, unos más establecidos que otros, algunos con estándares y otros sin ellos. En la Figura 1.1 se muestran varios elementos que influyen en el desarrollo del software.

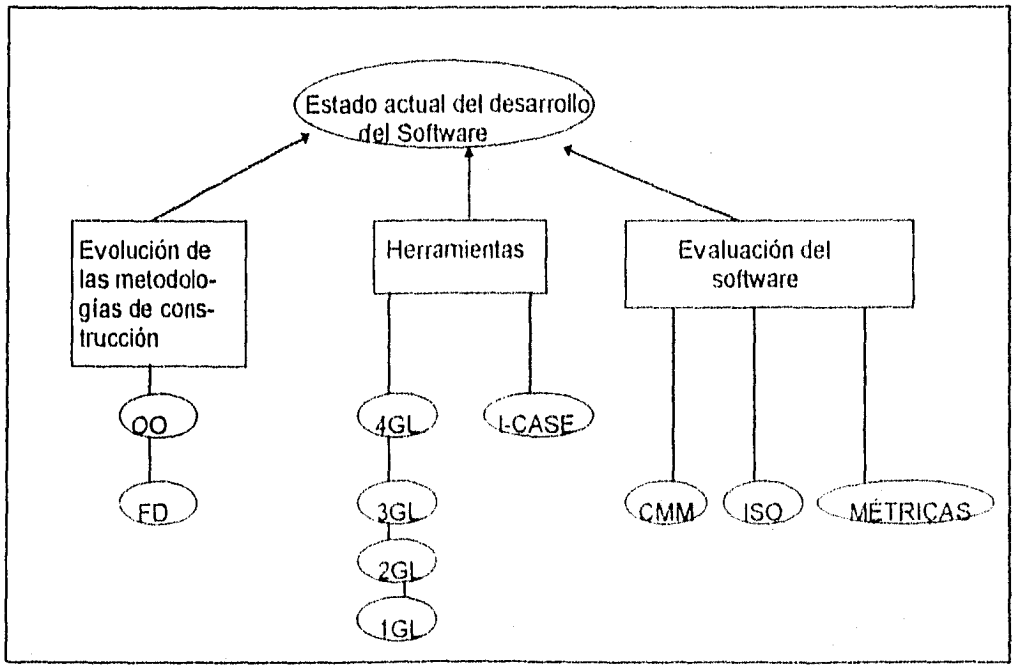


Figura 1.1 Esquematización de los elementos de desarrollo que afectan a la actividad de construcción del Software.

Así, se muestran tres grandes bloques:

Evolución de las metodologías.- Se refiere a aquellos elementos que ayudan a formar un modelo del problema del mundo real. Siendo los más representativos: actualmente el enfoque Orientado a Objetos (OO) y anteriormente el enfoque de Flujo de Datos (FD). Aúnque no se puede decir que una metodología a sustitido a la otra.

Herramientas.- Para poder implantar los modelos producidos por cualquier enfoque, es necesario recurrir a herramientas. Los lenguajes que han evolucionado desde los llamados Lenguajes de Primera Generación (1GL) que resultaban más cercanos a la máquina, hasta los más cercanos al lenguaje del ser humano o de cuarta generación (4GL).

Evaluación del software.- Tanto la elaboración del software como la revisión del producto final deben ser realizados para garantizar cierto grado de calidad deseado, y son los diferentes modelos (ISO, CMM) los que marcan pautas a seguir. Por otro, lado se tienen las métricas que ayudan a decidir el grado de calidad que un producto tiene [PRE93, MAC93].

En el marco de este trabajo, los elementos descritos son conformadores universales del estado actual de madurez de las actividades de construcción del software. Aúnque son varios, algunos han repercutido más en el medio mexicano, ya porque han tenido mayor aceptación o porque el número de personas que lo han tenido que acatar es mayor.

Aquí se revisarán algunos de estos elementos con el fin de situarse en un léxico común, en lo relativo a metodologías, herramientas, métricas del software. Otros elementos se verán por su importancia de frecuencia de aplicación como sucede con la metodología de Flujo de Datos. En lo posible nos situaremos siempre desde una perspectiva de la actividad de Ingeniería de Software en nuestro país.

1.1 El enfoque de Flujo de Datos.

Muchos de los sistemas computacionales de procesamiento de datos que están actualmente en uso se han elaborado con apoyo en el modelo de desarrollo orientado al Flujo de Datos (FD), ésto se debe al hecho de que esta metodología existe desde los finales de la década de los sesenta [SOM92]. Durante un par de décadas fue el método de desarrollo más natural, sobre todo para la etapa de diseño. Al llegar el modelo Orientado a Objetos (OO), a fines de los ochenta, se ha empezado a cuestionar la efectividad de FD con respecto a la calidad obtenida en los productos. Sin embargo, no cabe decir que FD ya no sirve; lo que sí se puede afirmar es la necesidad de una revisión del grado con que alcanza sus objetivos.

Por lo anteriormente expuesto, en esta sección se mostrará una serie de reflexiones surgidas de la práctica del enfoque FD. Previamente se desglosará de manera rápida en qué consiste, para poder discutir los temas sobre una base común. La tabla 1.1 resume los aspectos más importantes de este enfoque, mismos que se irán desarrollando en las secciones siguientes. Cabe aclarar que en el trabajo actual se ve con cierto detalle la metodología FD debido a que, los pasos que ésta utiliza son los que la metodología propuesta hereda.

TABLA 1.1 Actividades en el modelo de Flujo de Datos

ETAPA	ACTIVIDAD
ANÁLISIS	• DEFINIR OBJETIVOS: definición de la problemática usando Diagramas de Flujo de Datos (DFD) • ESTRATEGIA COMPUTACIONAL: definición de salidas propuestas, definición de entradas propuestas, nueva estrategia de trabajo con DFD. • DEFINICIÓN DE DICCIONARIOS DE DATOS
DISEÑO	• DETALLE DE LOS DFD subdividir el problema separando funciones muy grandes en subfunciones con sus propios DFD. • DIAGRAMA DE ESTRUCTURA para el sistema y cada subsistema hacer un diagrama jerárquico de Estructura • DETALLAR CADA PROCESO • LISTA DE PRUEBAS PARA CADA PROCESO • DESCRIPCIÓN DETALLADA DE ARCHIVOS • ESCOGER MÉTODO DE PRUEBAS DE INTEGRACIÓN

Fue a principios de los setenta cuando tomó más fuerza el enfoque de Flujo de Datos (FD) para el análisis y diseño de sistemas [PRE93]. La idea básica de este

es plasmar en diagramas especiales de qué manera fluye la información y se transforma al pasar por diferentes procesos o funciones (también conocida como orientación funcional) desde el momento en que es introducida hasta obtener resultados. Varios autores han contribuido al mejoramiento y adaptación del enfoque FD; se ha utilizado en diferentes tipos de problemas (administrativos, de investigación, de tiempo real) [PRE93], pero, básicamente, todos tienen la misma idea y la misma notación, que pueden verse en el diagrama de la Tabla 1.1 y se describen brevemente a continuación señalando los aspectos más relevantes en cuanto a las características que lo distiguen de otras metodologías.

1.1.1 El análisis en el enfoque de Flujo de Datos.

La etapa de análisis en FD consiste en obtener información del sistema actual (sea o no automatizado) y proponer alternativas de solución.

El analista (persona encargada de hacer el análisis del sistema), debe revisar el comportamiento del sistema actual y lo hace viendo cómo los datos, que se manejan en el sistema actual, van pasando por una serie de diferentes procesos, automáticos o manuales, hasta brindar los resultado. Tales procesos con que se obtuvo la información pueden resultar incómodos o costosos para el usuario. Entonces, ayudado por los Diagramas de Flujo de Datos (DFD), el analista debe plasmar la situación actual y proponer alternativas de solución. Los pasos del análisis en el FD son los siguientes:

- Definir objetivos.
- Establecer Estrategia Computacional.
- Desarrollar Diccionario de Datos.

A continuación se revisará cada una de estas etapas.

1.1.1.1 Definir objetivos.

Antes de precisar lo que se quiere hacer, es necesario conocer el contexto para el cual se quiere realizar el producto de software; para ello se realiza primero la definición del problema.

Para definir el problema se hace primero un Diagrama de Flujo de Datos (DFD) del estado actual del sistema. La realización de un DFD debe utilizar la siguiente notación:

Burbuja.- Una circunferencia que servirá para denotar un proceso o un evento del sistema (Figura 1.2). Se pone una burbuja por cada aclividad o evento que suceda en el sistema. A cada burbuja se le asocia una explicación por separado. Ejemplo: reunir, escribir, fin de día.

Almacén.- Consiste de dos rayas horizontales y paralelas con el nombre del almacén entre las dos (Figura 1.2) . Normalmente se refiere a los datos almacenados en medios magnéticos (conocidos como archivos), pero puede referirse también a almacenamientos físicos (como archiveros metálicos o cajas de cartón).

Terminador.- Se refiere a las unidades externas al sistema que se representan con un cuadro o rectángulo (Figura 1.2). Esto es, aquello que lo alimenta o lo que produce (documentos, gráficas).

Flecha.- Servirá para conectar los procesos con: procesos, almacenes o terminadores. Sobre las flechas se ponen los datos o paquetes de datos (ej. calificaciones, datos personales) que pasan entre los diferentes símbolos.

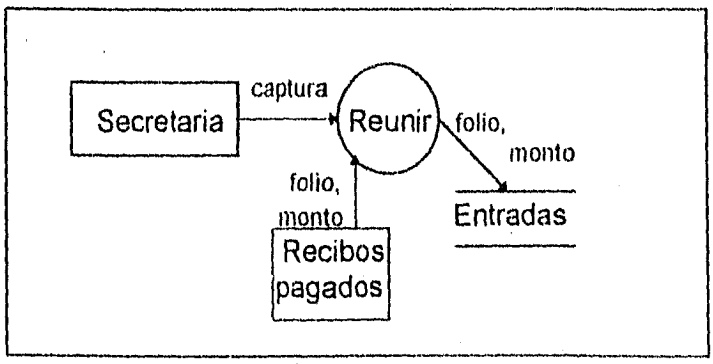


Figura 1.2 Ejemplo de diagrama de Flujo de Datos

La representación total del sistema se irá detallando en diagramas sucesivos evitando que cada uno exceda una hoja, a fin de visualizar mejor cada operación.

Al terminar los diagramas, el usuario señalará cuáles son los puntos fallos en el sistema actual y se hará una redacción de los aspectos que resulten ineficientes dentro del sistema.

Con la problemática aclarada se está en posición de establecer objetivos, metas y requisitos. Es posible que la solución al problema no involucre el uso de computadora y ésto debe aclarársele al usuario. Si se va a usar computadora, entonces se pasa a la siguiente fase.

1.1.1.2 Estrategia computacional.

Una vez aclarados los problemas actuales, los requisitos y metas del sistema, se procede a realizar una propuesta computacional, lo cual se puede hacer como sigue:

Definición de las salidas propuestas. Se hará una descripción de cada salida o resultado que emanará del sistema.

Se definen claramente cuáles serán las entradas.

Se hace una serie de DFDs para explicar la nueva estrategia. Debe hacerse igual que se hicieron los diagramas en la definición de objetivos.

1.1.1.3 Elaboración del Diccionario de Datos.

Lo que hasta ahora se ha realizado es una serie de diagramas que, de un primer vistazo, dan idea de lo que será el sistema. Para complementar los DFDs, es necesario realizar el Diccionario de Datos (DD). El DD contendrá las definiciones de los datos mencionados en los pasos anteriores. Habrá datos compuestos (datos que pueden ser subdivididos) y datos elementales (los que no pueden ser subdivididos). Los datos compuestos se definen en términos de sus componentes y los elementales por medio de los valores que pueden asumir. Un DD puede estar compuesto de procesos, archivos (datos almacenados) y datos usados en los procesos. Como ejemplos de lo que puede contener el DD se tienen los siguientes:

Almacenes.- Se pone el nombre del almacén con los datos que lo conforman. Ejemplo:

Alumnos	Nombre, matrícula, edad, fecha de ingreso, sexo, dirección
Materias	Nombre, clave, profesor, horario

Datos.- Cada dato y su descripción o fórmula que lo define. Ejemplo:

clave	se refiere al número de identificación de la materia
matrícula	año de la fecha de ingreso + número progresivo

1.1.2 Diseño de sistemas orientado al Flujo de Datos.

Para empezar el diseño de un sistema orientado al Flujo de Datos se realizan las actividades siguientes:

- Detallar el DFD del análisis.
- Realizar Diagramas de Estructura.
- Detallar cada proceso.
- Describir detalladamente los archivos.
- Definir el Diccionario de Datos.

Tales actividades no son necesariamente secuenciales; incluso, cabe esperarse que cada punto retroalimente a los otros, en mucho depende del modelo de desarrollo que se escoja.

1.1.2.1 Detallar el DFD propuesto en el análisis.

Para de obtener una serie de actividades claras e independientes, se van desglosando los almacenes y refinando los procesos, hasta lograr que éstos sean altamente cohesivos y que su acoplamiento sea bajo.

Lo anterior se refiere a que cada módulo debe comprender cuando más una sola función referente al sistema (alta cohesión) y que la comunicación entre los módulos diferentes sea mínima, ésto es, mediante parámetros o recurriendo a archivos de datos, eliminando así variables globales (bajo acoplamiento).

Otro punto importante será la forma en que queden los almacenes. Esto se irá logrando mediante diagramas sucesivos y conforme al proceso de normalización, que consiste en una serie de pasos para lograr que los datos queden funcionalmente juntos, es decir, cada atributo dependa (o sea función) de la llave.

1.1.2.2 Realizar diagramas de estructura.

La interacción de los módulos del sistema usando los DFD no es posible debido a que sólo se muestra cómo fluyen los datos y no cómo o cuándo un módulo decide llamar a sus subordinados; para ello se usa la notación de los diagramas de estructura que se muestra en la Figura 1.3.

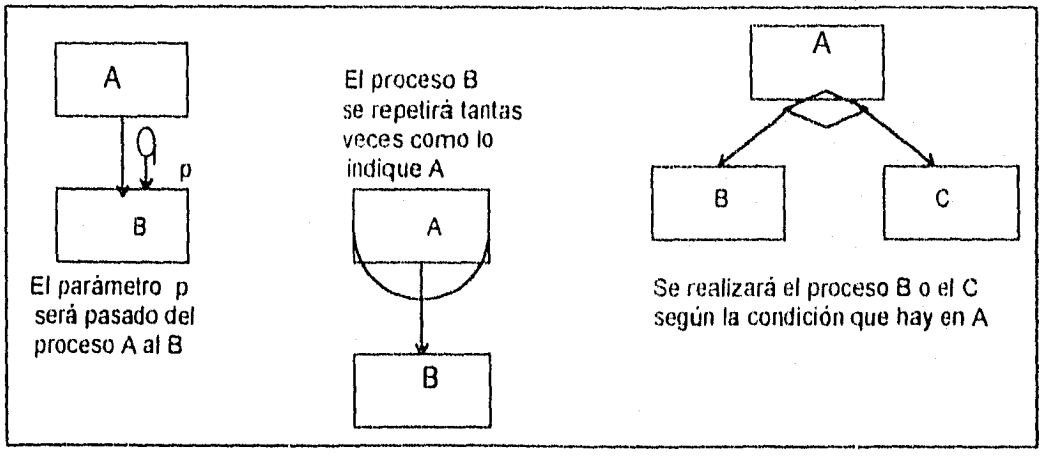


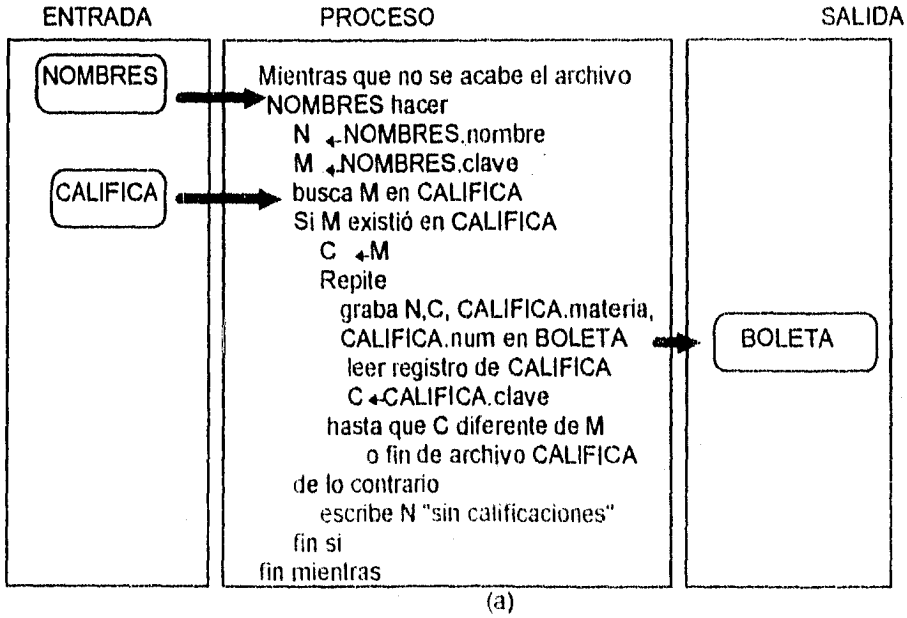
Figura 1.3 Notación para los diagramas de Estructura de la metodología de Flujo de Datos

Con los módulos identificados en los diagramas de Flujo de Datos se va formando una jerarquía, anotando cuál es llamado por cuál y en qué forma, ésto es, si solo cuando se cumpla una condición, de manera iterativa o si se pasarán parámetros.

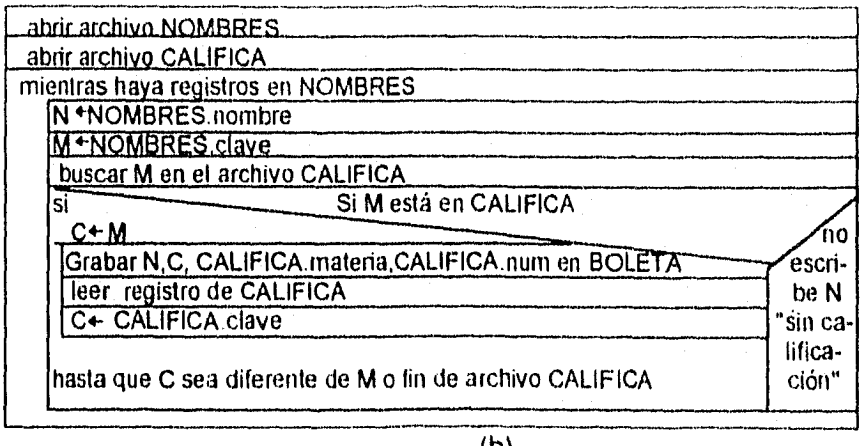
El diagrama jerárquico resultante puede ser extenso, así que las ramas pueden detallarse en diagramas aparte. En sistemas pequeños esta división resulta de gran utilidad.

1.1.2.3 Detalle de cada proceso.

Cada proceso o módulo debe especificarse en forma detallada. Para ello, se utilizan los diagramas de estructura del punto 1.1.2.2 o alguna otra técnica hasta que se logre una definición cuya implantación esté bien definida. Otras técnicas para la definición de un proceso son: español estructurado, diagramas HIPO o diagramas de caja o de Nassi-Schneiderman. La especificación del módulo debe ser del tamaño de una hoja carta como máximo. A continuación se muestra un ejemplo de un mismo módulo en HIPO (Figura 1.4a) y en diagrama de caja (Figura 1.4b).



(a)



(b)

Figura 1.4 Diagramas para el detalle de módulos

El proceso de ejemplo forma boletas de calificaciones guardándolas en el archivo (BOLETA) a partir de un almacén maestro de alumnos (NOMBRES) y un almacén de registro de calificaciones (CALIFICA).

1.1.2.4 Descripción detallada de archivos.

Esta fase tiene dos aspectos centrales:

1. Especificación detallada de archivos. En este paso deben definirse los campos que forman cada archivo, su método de organización lógica (secuencial, directo, indizado) y, en su caso, indicar qué campo o concatenación de ellos forman la llave primaria. Además, para cada campo se especificarán sus características, como pueden ser: su tipo, tamaño, si puede ser nulo o no.
2. Debe revisarse que la especificación final de archivos quede en tercera forma normal, cuando menos. Para ello se pueden seguir los pasos que se muestran en el diagrama de la Figura 1.5 [GAN79].

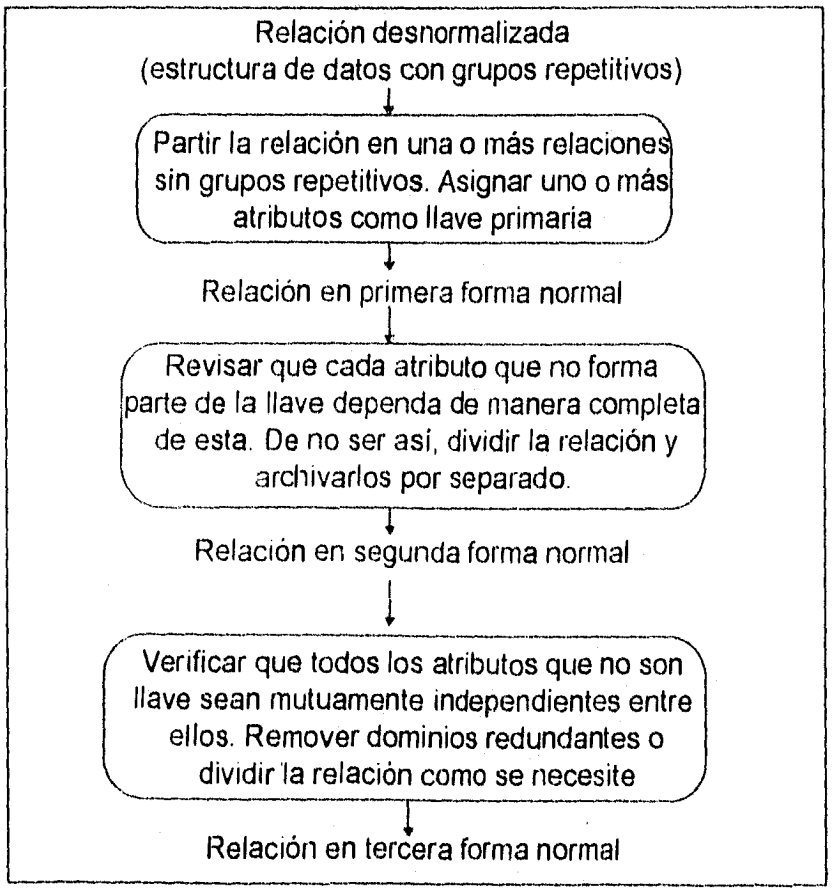


Figura 1.5 Proceso de normalización de los almacenes de datos

Al obtener los archivos en tercera forma normal, se tiene la ventaja de no producir anomalías de actualización [CAR83, KOR86].

1.1.2.5 Definición del Diccionario de Datos.

Se completa el Diccionario de Datos, agregando lo no previsto en el análisis y además, si se desea, las variable temporales de cada módulo.

Resulta importante mencionar que el papel del Diccionario de Datos es de complemento de Diagramas de Flujo de Datos y Diagramas de Estructura, detallando cada dato mencionado en los diagramas, ya sea con texto anexo o fórmula que defina la obtención del dato.

Ya en esta etapa también se añaden características de cada dato (numérico, carácter) y se señala si el dato es llave o parte de ella.

1.1.3 Algunos problemas con el enfoque de Flujo de Datos.

Existen varios aspectos dentro del FD que resultan ambiguos o incompletos, lo que lleva a la posibilidad de obtener productos de calidad insuficiente. Más aún, conducen a una práctica en la construcción de sistemas cuya administración y gestión controla pobremente el proceso de producción de software lo que implica no solo que el producto tiene problemas de calidad sino que, también, su construcción rebasa los pronósticos de esfuerzo, costo y tiempo. Entre los principales problemas se pueden citar:

- Al estar elaborando un DFD, es natural refinar cada vez más hasta lograr sólo módulos altamente cohesivos. Esta aparente conveniencia desemboca, sin embargo, en un conjunto de pequeños módulos sin sentido práctico y poco naturales que después costará mucho trabajo integrar.

- Las estructuras jerárquicas obtenidas en los diagramas de estructura sugieren dos tipos de pruebas de integración: de arriba hacia abajo (top-down) y de abajo hacia arriba (bottom-up) (existen combinaciones de ellos, pero se tratarán sólo top-down y bottom-up). Ambos métodos de prueba presentan inconvenientes que repercuten en la calidad y que a continuación se explican:

    - El método de prueba de integración top-down dice que se deben ir probando los módulos superiores e ir bajando cuando éstos hayan sido probados a satisfacción. Para lograr tal prueba es necesario construir lo que se conoce como "cabos" , que son pequeños módulos ficticios que hacen las veces de módulos de nivel inferior y que luego serán sustituidos por los verdaderos. El método tiene dos grandes inconvenientes: se pierde tiempo en realizar los cabos y los módulos correspondientes a las hojas se hacen al último y son, la mayoría de las veces, los que más problemas de programación tienen, originando reprogramación, recalendarización de las pruebas y que los proyectos no se entreguen a tiempo.

- El método bottom-up sugiere realizar los módulos de las hojas primero, ya que son los más complicados, e ir construyendo módulos falsos superiores que proporcionen datos. Ya que los módulos de nivel inferior se han completado, entonces de integran en el módulo de nivel superior. Esto obviamente trae complicaciones en el momento de la integración pues tal vez no se consideraron los problemas de espacio, paso de parámetros y uso de variables globales comunes.

- Independientemente del método de prueba que se haya planeado utilizar dadas las características del problema, existen las políticas de repartición de trabajo de la empresa, lo cual puede contradecir lo planeado. El resultado bien puede ser falta de naturalidad en las jerarquías ficticias impuestas por el método [JOR94].

1.2 Las herramientas.

Existen dos tipos de herramientas automáticas para el desarrollo de software: los lenguajes de programación y los CASE (del Inglés, Computer Aided Software Engineering). Los primeros están al alcance de cualquier desarrollador, mientras que los segundos solo de unos cuantos. En esta sección se trataran algunos aspectos relevantes de ambos que incumben al proceso de desarrollo del software.

1.2.1 Los lenguajes.

La herramienta fundamental para construir un producto de software es el lenguaje de programación. Este sirve para comunicar nuestras ideas a la máquina, mismas que tuvieron que ser convertidas a procedimientos formales.

Los lenguajes de programación han ido evolucionando con el tiempo; inicialmente eran un conjunto reducido de instrucciones (en los ensambladores) y ahora son formas mas amigables.

Aúnque existen lenguajes poderosos llamados de cuarta generación, en México la mayoría de las aplicaciones pequeñas en PC todavía se desarrollan en lenguajes de tercera generación como FORTRAN, COBOL y BASIC. Tal vez la razón principal sea que los recursos de hardware necesarios para dar apoyo a un lenguaje de cuarta generación son excesivos para la capacidad de empresas pequeñas.

Independientemente del lenguaje que se utilice para implantar el producto de software, lo principal en el proceso de codificación es el hecho de tener un diseño detallado correcto. Aúnque se tenga el mejor y más amigable de los lenguajes, un mal diseño llevará a una implantación incompleta y por ello sujeta a errores, fallas o problemas.

Diversos autores coinciden en señalar que no importa el lenguaje con que se codifique la aplicación; si tiene un buen diseño el resultado será bueno, mientras que si el diseño es malo, aúnque el lenguaje de programación sea muy bueno, la implantación será deficiente [PRE93].

1.2.1.1 Lenguajes para PC.

La implantación de un software sobre PC es muy variada y los lenguajes de programación con que se cuenta también, pero existe un gran problema: la portabilidad[4]. Mientras que se puede implantar fácilmente un sistema en lenguajes como VisualBasic, las características del hardware y del software que éste requiere lo hacen difícilmente portable a máquinas con menos de 8Mb de memoria RAM.

[4] Se usa el término portabilidad en el sentido de implantar un programa que está corriendo en una computadora en otra diferente.

Así, la autora ha observado que los desarrolladores de software portable para PC se ven en la disyuntiva de utilizar una herramienta poderosa, elegante y poco portable; o una herramienta mas convencional como seria un compilador de lenguaje C.

Asimismo se ha observado que, para el caso de desarrollo de sistemas de Procesamiento de Datos, existe una familia de lenguajes basados en Dbase[5] que resultan bastante portables y que de hecho se han constituido en un estándar para el manejo de datos. La gran mayoría de software para PC en el mercado afirma ser compatible o importar y exportar datos de archivos con formato Dbase. Sin embargo, cuando las aplicaciones empiezan a crecer tanto en resultados como en cantidad de datos, estos lenguajes resultan lentos en sus operaciones; por lo tanto la alternativa vuelve a ser un lenguaje como C o Pascal.

Por otro lado, en las PC de menos de 4 Mb de memoria, la aplicación del enfoque orientado a objetos para aplicaciones de más de dos objetos y volúmenes de datos medianos (miles de registros) resultan prácticamente imposible de usar. Los lenguajes orientados a objetos sobre plataformas como la que se menciona son útiles para aplicaciones gráficas o de entretenimiento.

Sin embargo, si se tiene una PC de 8 MB o más de memoria, se tendrá mayor facilidad de uso de los lenguajes orientados a objetos y de su conjunto basto de bibliotecas, aunque todavía se tiene el problema del precio de éste software.

Lo dicho en el párrafo anterior se basa en el hecho de que tanto los compiladores como los depuradores orientados a objetos son más caros y requieren más recursos de memoria y disco. Por otro lado, los ejecutables resultan ser bastante más extensos y, por ende, requieren de más recursos para trabajar. En el momento de guardar los datos se guardan no sólo éstos, sino también su estructura, acarreando con ello que se requiera demasiado espacio en disco.

Aúnado a los recursos físicos necesarios se encuentra el hecho de la falta de madurez en la metodología orientada a objetos. Así lo muestran diferentes autores al no ponerse de acuerdo en que notación usar (Ejemplos: [BOO91 ,COA90a, LOZ94]).

Sin embargo, y debido a que las PC son cada vez más poderosas y la metodología OO está madurando rápidamente, se espera que en poco tiempo se puedan llevar a cabo desarrollos significativos usando éste enfoque.

[5] Dbase es un manejador de bases de datos para equipos pequeños que tiene tanto su lenguaje de definición de datos como el de manejo de datos y además un lenguaje para realizar aplicaciones [KOR86]. En el trabajo actual, al referirse al lenguaje Dbase, se estará hablando del lenguaje de aplicaciones.

1.2.1.2 Lenguajes para redes de PC.

El panorama en el caso de las redes de PC no es muy distinto. El software de red se instala en el servidor (en ocasiones se instalan además pequeños módulos en cada PC instalada en la red) y, para el manejo concurrente de datos, se suministran instrucciones nuevas o bien algunos suplementos de instrucciones.

La situación del desarrollador se agrava, pues tiene que controlar el manejo concurrente a los datos además de la programación normal. Sólo algunos paquetes como Informix, brindan éste acceso automáticamente.

Dado entonces que los programas y la mayoría de los datos se encuentran en el servidor de la red, los tiempos de acceso a los datos siguen siendo un "cuello de botella". Por lo anterior se hará necesario utilizar lenguajes de propósito general y llamadas a utilerías del Sistema Operativo huésped (Novel o Unix).

1.2.2 Las herramientas CASE.

El término CASE (Computer-Aided Software Engineering) se utiliza para identificar una herramienta computacional de apoyo a la labor de desarrollo de software. Está orientada especificamente al profesionista de la computación que por muchos años, paradójicamente, ha trabajado las primeras etapas del ciclo de vida de construcción de sistemas computacionales sólo con "papel y lápiz".

La idea fundamental del CASE es ir construyendo, a partir de los primeros datos que se le dan, pasos adelantados del desarrollo de software. Así, si se le dan los requisitos del sistema en un formato adecuado, el CASE construirá el primer esbozo de un diseño y si se refina el diseño, el CASE será capaz de generar código en un lenguaje dado.

Los CASE ayudan a sistematizar y automatizar labores tediosas y monótonas. Por añadidura reducen el número de errores. Las verificaciones típicas de un CASE son validaciones de consistencia. Por ejemplo, que todos los nombres en los diagramas de estructura aparezcan en el Diccionario de Datos (DD), que todo archivo tenga su descripción detallada, que todo dato mencionado en el Diagrama de Flujo esté descrito en el DD y que todo dato en el DD tenga origen en algún Diagrama de Flujo.

Los CASE han ido evolucionando a partir de algunas herramientas dispersas. Por ejemplo, de una herramienta gráfica a un constructor de etapas. Se encuentran en el mercado varios CASE de diferentes calidades: los que se conocen como Upper-Case que cubren las etapas de planeación y análisis, los Lower-CASE que sólo brindan ayuda a partir del diseño y el I-CASE que cubre todas las etapas con un diseño que permite la inserción de módulos de diferentes orígenes.

En ningún caso se puede pensar que un CASE puede sustituir la labor del informático; sería tanto como decir que una calculadora sustituye a un ingeniero. Pero sí facilita su labor al ser un ayudante paciente y detallista.

Un aspecto importante que se debe anotar es que aún los I-CASE no empiezan de cero el análisis; ésto es, necesitan que el analista humano dé los primeros pasos: entrevistas, especificación de objetivos y establecimiento de requisitos.

1.3 Evaluación del software.

En el proceso de desarrollo de software se hace necesaria una serie de evaluaciones orientadas tanto a la estimación de esfuerzos y recursos necesarios para el desarrollo de un producto, como a la medición de resultados. En otras palabras, es preciso saber cuáles son los logros, qué esfuerzo y cuáles recursos se consumieron para terminarlo.

Estas evaluaciones se requieren en todas las etapas de desarrollo de software y son de gran importancia para la administración del proyecto. Sólo cuando se monitorean indicadores del funcionamiento del proceso de construcción de software es posible iniciar una gestión hacia la calidad.

Los diferentes colegios de profesionales que existen en el área técnica, arquitectos o ingenieros, han tratado de establecer estándares y medidas de calidad. Los principales motivos para ello son: brindar un mejor desempeño del producto vendido, poder sistematizar la producción en tiempo y costo así como saber cuánto cobrar por un producto.

En el caso de la computación, no es fácil definir estándares ni métricas de calidad, pero se han realizado varios intentos. Al respecto, se tienen dos enfoques: uno busca asegurar la calidad a través de todo el desarrollo del producto y otro se orienta a evaluar el producto en sí. Como lo indica el trabajo de Pfleeger et al [PFL94], casi todos los esfuerzos se han concentrado en el primer enfoque, en contraste con otras ramas de la ingeniería. De hecho, cuando se habla de calidad en el software casi siempre es a través de la opinión de los mismos que lo producen, sin referencia a los usuarios finales y sus opiniones, y sin estándares relativos a las características de los productos finales.

En las secciones siguientes se tratarán los temas de calidad en el proceso de desarrollo de software, estándares y métricas.

1.3.1 Calidad del proceso.

Hace poco más de una década, las empresas involucradas en el desarrollo de software - preocupadas por el hecho de que sus productos siempre se terminaban en más tiempo y con un costo mayor del presupuestado, y que incluso la realización de un producto similar a otro construido con anterioridad presentaba grandes diferencias en esfuerzo, costo y tiempo - comenzaron a buscar soluciones. Al tratar de mejorar la calidad de sus productos a la vez que dejaban satisfechos a los administradores en lo relativo a tiempo y recursos gastados, notaron que carecían de elementos objetivos para estimar la calidad y con muy pocas métricas confiables para medir los esfuerzos requeridos. En general, tampoco se contaba con información histórica utilizable.

En forma más o menos convergente, diversas empresas grandes (Por ejemplo, Hewlett-Packard y NEC) comenzaron proyectos internos orientados a establecer estándares y definir métricas adecuadas, a la vez que se recopilaba información aprovechable para estudiar la calidad. En general estos esfuerzos, quizá por su mismo origen dentro de las empresas, se orientaron al proceso de desarrollo mismo, más que a los productos en sí.

Los partidarios de éste enfoque consideran que si todo el proceso se realiza con cuidado, siguiendo estándares, el resultado naturalmente será bueno y además se cumplirán propósitos de ahorro en el costo de desarrollo.

Aúnque de momento carece de suficientes bases objetivas, debido a la falta de estándares auténticos, perfectamente validados, orientarse al proceso ha permitido formar conciencia, en las organizaciones que usan software, de la importancia de cuidar el desarrollo de éste. Al respecto dos modelos importantes son ISO 9001 y el CMM.

1.3.1.1 Estándar ISO 9001.

La Comunidad Europea, en su esfuerzo por unificar su economía, ha debido enfrentar el problema de la diversidad de grados de avance en el desarrollo de los países que la forman. Mientras que algunos se encuentran en la vanguardia del desarrollo tecnológico, otros aún emplean métodos antiguos, con diferentes grados de modernización. Uno de los aspectos más importantes a resolver es el de normalizar la calidad en los diferentes países, para lo cual se implantaron una serie de estándares adaptados de los empleados en los países con mayor desarrollo.

El estándar ISO 9001, junto con otros complementarios, busca garantizar la calidad asegurando que la empresa se comprometa como un todo y de manera que ese compromiso se concrete en cada uno de los pasos necesarios para la elaboración del software que se produzca.

El estándar se resume, según algunos, en "Diga lo que hace (el producto) y haga lo que dice (el documento)".

Algunos puntos relevantes del estándar son:

- Requiere documentación escrita de todos los procesos.
- Debe haber un compromiso escrito acerca de la calidad y de cómo lograrla.
- El compromiso debe comenzar por la administración.
- Debe asegurarse que el personal y los recursos disponibles sean suficientes.
- Acerca del control de calidad, debe asegurarse que exista un plan.

- Si se subcontrata a otra empresa, debe haber un control que asegure que ésta reúna los requisitos de calidad, compromiso y documentación. Además debe contar con personal y recursos suficientes.
- Debe haber un control del diseño del producto.
- Debe haber un control de la documentación.
- Debe haber un control de los procesos.

Aunque este estándar es obligatorio únicamente para productos en el mercado europeo, muchas empresas en otros mercados lo están adoptando, incluso en nuestro país. Para hacerlo se debe pasar por un proceso de preparación y una serie de evaluaciones realizadas por los representantes autorizados.

1.3.1.2 Modelo CMM.

Hasta hace poco más de una década, la mayoría de los estándares y esfuerzos por comprender el desarrollo del software corrían en dos vertientes con muy poca comunicación entre sí: la académica, que a veces llevaba a resultados elegantes pero inútiles, y la pragmática, a nivel de empresa o desarrollador. A mediados de los 80, se comenzó un esfuerzo para reunir puntos de vista tanto de industria como de academia lo que llevó al modelo CMM (Modelo de madurez de capacidades) que es una propuesta del Instituto de Ingeniería de Software de la Universidad de Carnegie-Mellon en el cual participan académicos y representantes de la industria. El modelo también se orienta al logro de la calidad al asegurar la calidad del proceso de desarrollo de software. En forma semejante al ISO 9001, este modelo también pretende involucrar a toda la empresa, no sólo a los desarrolladores del software.

El modelo CMM considera que las empresas pueden estar en uno de cinco estados[6], en relación a la producción de software:

- **Inicial**: es un estado inestable, en que la calidad depende de la capacidad de los individuos y un poco de su acierto al elegir métodos de trabajo. Por lo tanto, cualquier cambio de personas o de tipo de trabajo llevan a resultados impredecibles. A veces se le llama "heroico", en honor a los desarrolladores. En este estado, costos y tiempos de entrega usualmente rebasan las metas, y presentan una enorme variabilidad.
- **Repetible**. En este estado se documentan aciertos y errores en los proyectos, por lo que planeación y administración se basan en proyectos anteriores. Se pueden trazar costos, funcionalidad, planeaciones. Más aún, se establecen y se cumplen los estándares necesarios, por lo que se puede decir que los procesos están disciplinados. En este estado la probabilidad de que costos y tiempos de entrega tengan una distribución

[6] Aunque en los escritos aparece la palabra "level" que literalmente se traduce como nivel, se escogió la palabra estado para significar el hecho de que se puede retroceder y no sólo avanzar dentro del modelo CMM

más centrada por lo que hace a las metas es mayor, pero aún presenta una variabilidad grande.

- **Definido**: En este estado ya se tiene información que permite establecer procesos típicos, consistentes, los cuales se documentan. Todo el desarrollo del software se integra en un proceso de software estándar. En este estado se reduce la variabilidad en estimaciones de costo y duración.
- **Administrado**: Este cuarto estado se caracteriza porque cuenta con una base de datos global. La empresa establece metas cualitativas y cuantitativas para productos y procesos, con medidas consistentes. Se puede medir calidad y productividad. Los procesos, de este modo, resultan predecibles. En este estado se continúa reduciendo la variabilidad de las estimaciones.
- **Optimizando**: Al llegar a este estado, la empresa entra en un ciclo de mejoras continuas a los procesos que ha establecido. Cabe suponer que para entonces la variabilidad en estimaciones será mínima y que la misma empresa irá fijando sus metas de superación.

El modelo CMM se halla documentado en archivos públicos vía Internet, de modo que se puede aplicarlos al gusto. Para obtener un reconocimiento oficial del nivel en que se halla una empresa, se debe pasar por un proceso de certificación semejante al del ISO 9001. Como este proceso involucra una auditoría, se requiere realmente el soporte de la empresa como un todo.

Algunos autores han realizado análisis comparativos entre el estándar ISO 9001 y el CMM, observando que tienen muchos elementos comunes pero también algunas diferencias debidas a su distinto objetivo, más comercial y normativo en el primer caso.

1.3.2 Los Estándares.

Los productos de software adolecen de lo que se conoce como la crisis del software la cual señala varios problemas entre los que se cuentan: demora en la entrega, presupuesto sobrepasado, insatisfacción del usuario, inconfiabilidad e inmantenibilidad.

Esta situación se debe a varios factores: el dominio de la aplicación es muy variado (por ello los productos de software deben ser flexibles), son demasiado caros (se requiere bajo costo), tardan mucho en llegar al usuario (se necesitan tiempos cortos de desarrollo) y, por añadidura, hay falta de aplicación de estándares que permitirían medir avances y calidad del software.

Antes de seguir es necesario saber lo que se entiende por estándar. Según el British Standards Institute [PFL94] se define como:

"Una especificación técnica u otro documento disponible públicamente, desarrollado con la cooperación y el consenso o aprobación general de todos los intereses afectados por él, basado en resultados consolidados de ciencia, tecnología y experiencia, orientados a la promoción y beneficios óptimos de la comunidad"

Muchos estándares de Ingeniería de Software no encajan en esa definición ya que no se cumplen los requisitos de cooperación y consenso, no se fundamentan en ciencia y tecnología ni miden objetivamente sus beneficios. Sin embargo, tanto el CMM como el ISO 9001 se han constituido como dos estándares de facto para la administración del proceso de construcción de software. El CMM en EEUU, principalmente porque el gobierno federal norteamericano y muchas organizaciones públicas de ese país demandan que un proveedor sea nivel tres para participar en cualquier contrato de licitación. En Europa, la presión por certificarse ISO 9001 proviene del cliente que, para certificarse el mismo, se ajusta al estándar que le exige que sus proveedores sean certificados.

Debemos considerar que, aún así, hay un gran número de estándares; 250 aproximadamente en el mundo para Ingeniería de Software (IS), según Pfleeger y sus coautores [PFL94], y eso motiva algunas preguntas:

- ¿qué prácticas deben estandarizarse?
- ¿los estándares no sirven o no se están utilizando y por qué?

Puede ser que muchos estándares no hayan sido bien investigados y establecidos; más bien son "a sentimiento".

Todas las empresas deberían establecer estándares de trabajo que les permitan llevar el seguimiento de sus proyectos y poder así mejorarlos. Pero antes de adoptar un estándar se deben contestar preguntas como:

- ¿Cuáles son los beneficios potenciales de usar estándares?
- ¿Puede medirse objetivamente la extensión de los beneficios que se obtendrán de su uso?
- ¿Cuáles son los costos necesarios para implementar el estándar?
- ¿Exceden los costos a los beneficios?

Para saber si un estándar es "bueno" habrán de revisarse tres aspectos:

- debe saberse cómo determinar si se cumple o no;
- debe conocerse el criterio de éxito;
- cuánto cuesta cumplirlo.

Los estándares pueden referirse a: el producto por dentro; el producto por fuera; el proceso; los recursos. En otras ingenierías los estándares son principalmente del producto; los estándares de IS, como se dijo anteriormente, casi

exclusivamente se refieren al proceso y luego a recursos y al producto en su interior.

Visto lo anterior, es importante notar que los modelos ISO 9001 y CMM resultan importantes porque obligan a toda aquella empresa que quiera competir con productos de calidad a adoptar algún estándar. Esto trae consigo un impacto en las prácticas cotidianas de construcción de sistemas; cada vez se necesita más disciplina y gestión orientada a la calidad en la construcción de sistemas y ésto pronto será un reto para el pequeño productor de software a la medida para PC en México.

1.3.3 Las métricas.

Para obtener una buena calidad en un producto de software es necesario adoptar una forma sistemática y eficiente de realización de actividades. Tal enfoque es conocido como el aseguramiento de la calidad del software o SQA (de sus siglas en inglés Software Quality Assurance). Las actividades principales en el SQA son siete según Pressman [PRE93]: (1) aplicación de métodos técnicos, (2) revisiones técnicas formales, (3) prueba de Software, (4) ajuste a los estándares, (5) control de cambios, (6) realización de informes y (7) las mediciones, esta última es de la que se tratará en esta sección.

La forma de medir la calidad del software es todavía bastante subjetiva pues son pocos los factores de calidad que se pueden medir objetivamente en un producto nuevo, más bien los factores de calidad estarán dados en función del comportamiento de productos similares. Usualmente, en vez de medir directamente la calidad, se eligen variables relacionadas, ya que resultan más sencillas de medirse; la relación se establece por la experiencia o formalmente, empleando métodos estadísticos, a las cuales se llama **métricas**. Las métricas no miden realmente la calidad, solo miden las manifestaciones de ésta. Entonces se puede decir que las métricas del software son medidas del éxito.

Antes de elegir un conjunto de métricas que resulten de utilidad para un proyecto, conviene considerar las bases siguientes:

- Las métricas requieren datos que sirvan para los propósitos de estudio. Cada una tiene sus peculiaridades y a veces requieren datos que no se tienen y son difíciles de estimar.
- Conviene contar con un plan de aplicación de las métricas; para el avance del proyecto, una lista de tareas para las cuales se llevará control de avance.
- Como los resultados se compararán contrastándolos con promedios históricos, se requiere la creación de una base de datos sobre proyectos desarrollados. Nótese que esta base de datos debe relacionarse con el tipo de datos mencionado en el primer punto.

- Existe un gran número de métricas propuestas en la literatura pero muchas carecen de utilidad práctica, ya sea por falta de validación o por ser difíciles de estimar. Una métrica famosa que es unánimemente rechazada es la de Halsted [SOM92].

En relación al desarrollo de proyectos de software, de acuerdo con R. Grady [GRA94], las métricas se pueden agrupar en cuatro áreas:

1. Estimación del proyecto y monitoreo del progreso.
2. Evaluación de productos del trabajo.
3. Mejoramiento del proceso empleado (análisis de fallas).
4. Validación experimental de las mejores prácticas.

En los proyectos grandes, es común que haya conflicto entre los administradores del proyecto y la administración de la empresa. Los primeros prefieren enfocar sus esfuerzos a las dos primeras áreas, mientras que la administración de la empresa observa con más cuidado las dos últimas.

En las subsecciones siguientes se tratan brevemente las cuatro áreas, indicando algunas métricas recomendables.

1.3.3.1 Estimación del proyecto y monitoreo del progreso.

Durante mucho tiempo, las métricas de estimación de proyectos, y aún de su avance, estaban centrados en la línea de código y en las horas-hombre. Sobre este tipo de conceptos se construyeron modelos como el COCOMO de Bohem (ver [PRE93]). Todos los modelos que emplean las líneas de código sufren del problema de convertir las unidades cuando se cambia de lenguaje, ya que no serán iguales 100 líneas de código en Cobol y 100 líneas en C++.

Para rastrear el progreso es útil revisar las líneas de código escritas y compararlas contra el tiempo en que fueron generadas; se pueden separar las codificadas y las examinadas, así como también las que corresponden a funciones completadas. Se recomienda una actualización semanal.

Actualmente se está popularizando el modelo de **puntos de función**, derivado del modelo que Albrecht desarrolló en IBM. Este modelo permite estimar proyectos antes de iniciarlos, a partir de características propias del proyecto, sin relación con el lenguaje en que se codificará. Diversas empresas reportan el uso de esta métrica para la que existe una asociación que administra el modelo y certifica a los que la emplean: The International Function Point Users Group. Para más información se puede consultar el libro de C. Jones [JON91].

Un buen paquete de métricas relacionadas con el monitoreo de avance son las conocidas como FURPS (Funcionalidad, Usabilidad, Confiabilidad, Rendimiento, Soportabilidad). FURPS es el de un conjunto de factores de calidad propuestos

por la compañia Hewlett-Packard y cuya explicación se da con más detalle a continuación [PRE93]:

- Funcionalidad.- Se obtiene como una evaluación del conjunto de características y de posibilidades del programa, la generalidad de las funciones que se entregan y la seguridad de todo el sistema.
- Usabilidad.- Se calcula considerando los factores humanos (psicológicos, sociales y ergonómicos), la estética global, la consistencia y la documentación.
- Confiabilidad.- Se calcula midiendo la frecuencia de fallos, su importancia y su recuperación, la eficiencia de los resultados de salida y que tanto prevé el programa los posibles errores.
- Rendimiento.- Mide la velocidad de proceso, tiempo de respuesta, consumo de recursos, rendimiento total de procesamiento y la eficiencia.
- Soportabilidad.- Se refiere a la posibilidad de ampliar y adaptar el programa, facilidad de prueba, compatibilidad, la posibilidad de configuración y la facilidad con que se pueden encontrar los problemas.

Para los administradores de la empresa importa más ir de acuerdo con el programa, lo que se traduce en la conveniencia de observar aspectos como:

- El corrimiento, que se define como el tiempo que se retrasa un proyecto hacia el futuro al requerirse una replanificación.
- Progreso promedio de proyectos, que se mide como:
  1-(Σcorrimiento(i) / Σtiempo transcurrido(i))
- El monitoreo de la gráfica de progreso (con ajustes de promedio móvil) ayuda a corregir los corrimientos.

En las etapas finales del proyecto se recomienda revisar la tendencia en defectos hallados y corregidos. Estas métricas y su tendencia resultan buenas para los administradores de la empresa y también para la administración del proyecto, ya que permite establecer criterios de liberación de productos.

Por ejemplo, si la razón

número de defectos / (1000 horas de examinación)

se mantiene dos semanas por debajo de una meta fijada con datos históricos, el producto puede liberarse.

1.3.3.2 Evaluación de productos de trabajo.

Otro grupo de métricas se refiere a los productos del trabajo. Se define un **producto de trabajo** como la porción del trabajo que se puede entregar o vender. Se debe buscar una métrica adecuada para cada uno de los posibles productos.

Algunas de éstas métricas, referidas a los programas, son las de complejidad, que ayudan a lograr productos más mantenibles. Dos de ellas son:

- Complejidad ciclomática. Esta métrica es ampliamente utilizada y se define como el número de instrucciones predicado (while, repeat, if, for) que existen dentro de un programa más uno. Para su aplicación se compara el valor obtenido, al calcularse la complejidad ciclomática, contra el número de actualizaciones necesarias y su costo, y de ahí se puede establecer una regla de decisión, como por ejemplo: "La complejidad ciclomática no debe tener un valor mayor de 14 por subrutina". Esta métrica se desarrolló para sistemas con énfasis en el uso de procesador, donde importa mucho el camino que sigue un algoritmo, y por ello es más útil en sistemas de control que en aquellos orientados al proceso de datos (se usa con diagrama de flujo de control).

- Cuadrado de llamadas (fan out). Esta métrica resulta buena para sistemas de proceso de datos. Existen una serie de variantes que tratan de mejorar su comportamiento y corregir algunos de sus puntos débiles (por ejemplo, no le da valor a los módulos terminales en la jerarquía, ya que no llaman a ningún otro). De cualquier forma, se ha encontrado que el cuadrado del fan out presenta una fuerte correlación con el número de defectos presentados. Esto se ha comprobado con sistemas que tienen vida más o menos larga, notándose que los módulos más modificados a lo largo del tiempo son aquellos que presentan los mayores valores con esta métrica. Esta métrica se usa junto con el diagrama de módulos. Uno de los usos que se le puede dar es poner un límite a la complejidad o al número de módulos con cierta complejidad, o pedir más documentación para tales módulos.

1.3.3.3 Mejoras al proceso mediante de análisis de fallas (failures).

Autores y analistas de la gestión de la construcción de software [GRA94] mencionan que otro aspecto práctico que conviene considerar en la selección de métricas es el relativo a las fallas que se van presentando, para lo cual se registran:

- Patrón de defectos del proyecto. Todos los defectos se van graficando por módulo en un histograma, y se atacan los que presenten mayores frecuencias.
- Patrones en el desarrollo de software. Las fallas se van organizando por origen (especificaciones, diseño, codificación), tipo (por ejemplo estándares, lógica) y modo. Para los de mayor frecuencia de referencia, se discute y se construye un diagrama

causa-efecto (Fish-bone); a partir de su análisis se sugieren mejoras. Debe notarse que las mejoras notables son progresivas y a veces no apreciadas a corto plazo por la administración de la empresa, por lo cual se debe considerar el panorama a largo plazo.

1.3.3.4 Validación experimental de las mejoras prácticas.

Se ha dicho anteriormente que muchas métricas carecen de validación práctica y aún de justificación teórica, aún cuando algunos continúen empleándolas. Para evitar caer en el uso de métricas inadecuadas, conviene validarlas experimentalmente. La validación es un proceso que no les gusta mucho a los desarrolladores, ya que sienten que se desvían de su objetivo principal. Sin embargo, según los comentarios que publican los desarrolladores expertos, rinde buenos resultados. Por ejemplo, para la examinación, se ha comprobado que el método de **lectura/inspección**[7] supera hasta cuatro veces a otros [GRA94].

[7] Prueba de software que se realiza cuando el desarrollador hace una lectura del programa y un conjunto de personas le van preguntando dudas, con lo cual se logran descubrir varios errores.

2. CONCEPTOS UTILIZADOS.

En este capítulo se verán aquellos aspectos teóricos en los que se fundamenta la metodología propuesta, siendo además los más relevantes y que surgieron como apoyo natural. En primer lugar se revisan los diccionarios de datos que forman la parte medular de la metodología, el modelo semántico de Bases de Datos se ve como parte del modelado de datos, después se revisan dos técnicas de representación del conocimiento: redes semánticas y guiones (scripts en inglés) y por último se muestran algunos conceptos de la orientación a objetos que fueron tomados en cuenta especialmente en la planeación de pruebas desde la etapa de análisis. Los diferentes temas se presentan más bien desligados entre sí, ya que su aprovechamiento conjunto surgió, inicialmente, de la práctica más que de un trabajo teórico. Su interacción y aplicación será tema de los capítulos siguientes.

2.1 Diccionario de Datos.

Un Diccionario de Datos (DD) es una estructura que contiene metadatos acerca del sistema. El papel de los DD ha ido evolucionado desde una visión estática, que corresponde a versiones antiguas del esquema de una base de datos (así lo ve, por ejemplo Wiederhold [WIE83]) hasta estructuras complementarias de las bases de datos, e incluso como descripción general de todo un sistema, como en los Lenguajes de Cuarta Generación y en las herramientas CASE.

Los DD cumplen tres tareas genéricas principales:

- Sirven de documentación y referencia del sistema, facilitando el entendimiento del mismo, según autores como Lucas [LUC81], Sommerville [SOM88] y Pressman [PRE88].
- Ayudan a lograr una auténtica independencia frente a los datos, ya que las diversas aplicaciones que los utilizan pueden referirse al diccionario para obtener de él las características de aquellos, al incluir la definición del esquema de la base de datos. Esta forma la incluyen autores como Lucas [LUC81] y Cárdenas [CAR83].
- Otra forma en la que puede verse un DD es como una Ontología, esto es como el conjunto de elementos primitivos, terminología y paradigmas que sirven para interpretar un dominio del discurso. De esta manera el DD (la Ontología) da un marco conceptual para hablar acerca de una aplicación del dominio y un marco de trabajo para la solución de problemas. Así, se puede particionar el contexto para que su solución resulte menos compleja [LEN90, WIE92].

El contenido de un DD varía en extensión según los diferentes autores. Una lista de su contenido, no exhaustiva, es la siguiente:

Entidades:

- Campos.
- Archivos.
- Esquema y subesquemas.
- Procesos.
- Personas.
- Documentos fuente.

Relaciones:

- Campos con registros.
- Registro con registro.
- Registros con archivos.
- Acceso a datos con personas.
- Datos con programas.

En cada metadato se puede almacenar una serie de atributos; algunos son generales, como el nombre o la descripción, y otros dependen del uso particular que se les da e incluyen las características físicas de los datos.

2.1.1 El papel actual de los Diccionarios de datos.

Los diseñadores de diccionarios de datos pretenden lograr una total integración, de manera que la única fuente de metadatos esté en él. Hasta ahora no se ha logrado. La mayoría de los DD en uso se han desarrollado sobre equipos grandes para servir a empresas también grandes. De esta manera son sistemas de gran tamaño que usualmente consumen una parte significativa de los recursos de cómputo. Sin embargo, versiones pequeñas de los mismos pueden correr en equipos chicos.

Por otra parte, los DD tradicionales son sistemas que pueden ser independientes del sistema concreto al que sirven, incluso del manejador de base de datos que se emplea. En el caso de los equipos personales, los manejadores traen módulos que ayudan a realizar sus funciones, aún cuando sea en forma muy limitada, como es el caso de las herramientas tipo Dbase.

La definición de un DD debe hacerse desde el momento del análisis de manera informal, como un simple glosario, hasta su formalización en el diseño detallado del sistema, quedando como apoyo para la documentación técnica [YOU90].

La existencia del DD permitirá una conexión entre analistas y programadores y un rápido entendimiento del sistema para cuando, en un futuro, se quieran hacer cambios. Esto es muy importante considerando que, en la mayor parte de su vida, un sistema requiere mantenimiento.

Para ilustrar la utilidad y flexibilidad del DD en el mantenimiento de un sistema se muestra el siguiente ejemplo:

> Supóngase que se ha desarrollado un sistema de control de empleados con metainformación en un DD en línea, esto es, el DD forma parte del sistema computacional, y que una vez puesto en operación se requiere modificar las características del campo edad que originalmente eran solo los años cumplidos y que ahora se quiere saber la fracción de año transcurrido.
>
> Usando el Diccionario de Datos, se puede obtener un listado de todos los módulos que usan el campo edad, se puede hacer el cambio de manera global dentro del DD y además estimar mejor el tiempo que se llevará hacer cambios y pruebas de corrección. Sin un DD se tendría que prometer una fecha, sin fundamento, para completar el mantenimiento y para realizarlo se tendría que revisar todo el código del sistema.

Como la metodología que aquí se propone es para sistemas de fácil mantenimiento, es de esperarse que el DD sea un instrumento central. Más adelante (capítulo 4) se ilustrará el valor del DD desde un punto de vista práctico, comparando los desarrollos que se han hecho usando o no el DD.

El uso de un DD en la elaboración de los sistemas de información resulta de gran utilidad en una empresa si se tiene un sistema integrado de información o si el sistema incluye más de cinco archivos entre catálogos, archivos maestros, históricos y de transacciones; sin embargo poco se ha utilizado si el sistema en cuestión es aún más pequeño.

2.1.2 El DD en la interoperatividad.

Además de las aplicaciones generales antes citadas, conviene agregar que los esfuerzos que se realizan tendientes a sistematizar la comunicación de datos entre bases distintas obliga a desarrollar sistemas basados en un Diccionario de Datos, aún cuando no se utilice este término. A manera de ejemplo, se puede citar el reporte del XDF Subgroup [ALK89], orientado a la estandarización del intercambio de datos entre bases biológicas. El reporte propone un lenguaje para definir las características de los datos en forma estándar y propone la realización de sistemas específicos que pasen los datos de una base dada al formato estándar y de regreso a la misma base o a una diferente.

El lenguaje propuesto descansa en una definición de los datos que es un verdadero diccionario. Así pues, si los datos a exportar o importar corresponden a un sistema con DD, la conversión al formato estándar será muy simple y automatizable, mientras que para un sistema sin esta ayuda deberá realizarse una buena parte del trabajo en forma manual.

2.2 El modelo semántico para Bases de Datos.

Existen varias deficiencias en los métodos tradicionales (jerárquico, reticular, relacional) para el desarrollo de sistemas y modelado de datos y estos faltantes son:

- Abstracción de datos.
- Herencia.
- Restricciones.
- Objetos no estructurados.
- Propiedades dinámicas de una aplicación.

Los modelos tradicionales son también conocidos como basados en el registro, ya que se preocupan mucho más de la forma de almacenamiento de los datos que de la conceptualización del mundo real. El modelo semántico, en cambio, trata de representar en forma clara la situación del problema que se está trabajando; por ello es considerado herramienta ideal en la transición entre análisis de requisitos y diseño detallado de los sistemas [HUL87].

Existen varios enfoques del modelo semántico y los más conocidos son: FDM (Functional Data Model), SDM (Semantic Data Model) y ER (Entity-Relationship); este último, por su importancia, se verá aquí y servirá para mostrar los conceptos más usados en el modelo semántico.

2.2.1 El modelo Entidad-Relación.

Propuesto por Chen en 1976 y modificado y ampliado por varios autores más [HUL87], el modelo Entidad-Relación (E-R) está considerado como una forma de trabajar el modelo semántico de Bases de Datos y es muy utilizado para el modelado de datos. De hecho, una vez propuesto el modelo de datos en E-R, existen métodos para convertirlo a cualquier modelo lógico de Bases de Datos existente (relacional, jerárquico, reticular) [KOR86].

Las partes que conforman el modelo E-R se describen a continuación [TEO86]:

Entidad.- Se expresa por medio de un rectángulo que en su interior lleva el nombre de un sustantivo, el cual puede representar persona, lugar, cosa o evento de interés en el sistema. Existen también las *Entidades Débiles*, las cuales se dibujan con un rectángulo de línea doble y no pueden existir por sí solas, esto es, dependen de una Entidad Fuerte. Ejemplo: si se tiene la Entidad vehículo, la entidad automóvil sería una entidad débil.

Atributos.- Se utilizan para detallar el contenido de las entidades. Se representan por óvalos unidos al rectángulo de la entidad a la que pertenecen por medio de una línea recta. Existen dos tipos de atributos: los descriptivos y los identificadores. Éstos últimos solo los tienen las Entidades Fuertes y sirven para

identificar un ejemplar de la entidad de otro. Los atributos identificadores van subrayados y son conocidos como llaves primarias o simplemente llaves.

Relación.- Se representa por medio de un diamante que lleva dentro el nombre de una asociación entre dos o más entidades. Una relación puede o no contener atributos.

Para establecer la relación semántica entre una entidad y otra se debe utilizar una relación (diamante) y la ocurrencia de esta conectividad (uno a uno, uno a muchos y muchos a muchos), se explícita poniendo números encima de las líneas de unión. Cabe aclarar que esta notación ha variado con el tiempo y una de las más utilizadas es la de poner punta de flecha a la línea que va del diamante a la entidad en lugar de ir un número uno sobre la raya.

Para aclarar la notación se utiliza el ejemplo que se ve en la Figura 2.1 y la cual representa las relaciones entre alumnos, las materias que llevaron y las calificaciones que obtuvieron. La letra n puesta sobre la raya representa el tipo de relación que guarda la entidad alumno con la entidad materia mediante la relación "llevó". Lo que quiere decir que varios alumnos pueden llevar una o más materias y que a su vez la materia puede ser llevada por muchos alumnos.

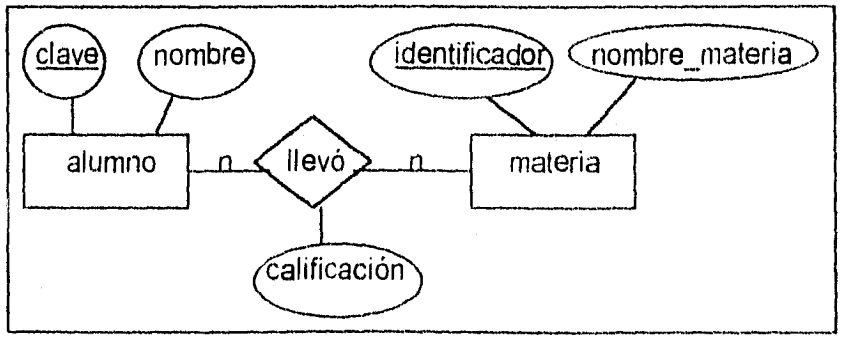


Figura 2.1 Ejemplo del modelo Entidad-Relación

Dentro de las extensiones que tiene el modelo Entidad-Relación están los conceptos de Jerarquía de Generalización (generalización) y Jerarquía de subconjunto (especialización), las cuales se describen a continuación:

Jerarquía de generalización.- Una entidad E es la generalización de las entidades E_1, E_2, ..., E_n si cada ocurrencia de E es también una ocurrencia de una y sólo una de las entidades E_1, E_2, ..., E_n. Ejemplo: La entidad EMPLEADO es generalización de SECRETARIA, TÉCNICO e INGENIERO.

Jerarquía de subconjunto.- Una entidad E_1 es un subconjunto de otra entidad E_2 si para toda ocurrencia de E_1 hay también una ocurrencia de E_2. Ejemplo: una entidad CUENTA puede incluir CUENTA_AHORRO, CUENTA_INVER como especializaciones que constituyen subconjuntos de la primera. Toda cuenta de ahorros o cuenta de inversión es también una cuenta.

2.3 Representación del conocimiento.

El problema central del analista y diseñador de sistemas consiste en estructurar y diseñar modelos de las aplicaciones en el mundo real que, eventualmente, puedan ser implantados en una computadora.

Las técnicas de Ingeniería de Software permiten, en primer lugar, la organización y el control del proyecto en desarrollo y, en segundo lugar, proveen las herramientas y notaciones estándares para la conformación y estructuración del modelo del sistema.

Las herramientas y notaciones de la Ingeniería de Software cubren muchos aspectos, pero los modelos obtenidos no siempre son completos. Entre más se logra capturar y formalizar en las notaciones y herramientas para los requisitos del sistema y para el funcionamiento ideal del mismo, más sistemática y robusta resulta la construcción y mejores resultados se obtienen en la calidad del producto final.

La elaboración de Sistemas de Procesamiento de Datos (SPD) es considerada una tarea rutinaria y que requiere menor esfuerzo intelectual que otras tareas dentro de la computación. Sin embargo, todavía no existe una herramienta automática (ni el CASE más refinado) capaz de sustituir en un cien por ciento la labor de un desarrollador de software.

Dado entonces que se requiere la inteligencia humana, es de esperarse que se piense que algunas técnicas de la Inteligencia Artificial (IA) puedan ayudar en el establecimiento de un mayor formalismo dentro de los SPD con la idea de lograr una mayor automatización.

En consecuencia, como las notaciones y formalismos de la IA para representar conocimiento son en sí mismos instrumentos de modelaje de aspectos complejos del mundo real y que resulta difícil modelar con los estándares notacionales de la Ingeniería de Software, se propone, en este trabajo, hacer uso de herramientas de representación de la IA para integrar otra parte de la metodología propuesta. A continuación se presentan una serie de elementos de la IA que participan, de una u otra forma, en la metodología propuesta.

2.3.1 Las redes semánticas.

Dentro de las diferentes formas de representación del conocimiento, las redes semánticas resultan ser una forma adecuada de representar datos y relaciones entre ellos dentro de los SPD, ya que se encuentran muy cerca del lenguaje natural, en el que se expresan tanto los requisitos como las observaciones acerca de los sistemas existentes y proyectados. Las redes semánticas son muy útiles en la etapa de análisis y pueden relacionarse fácilmente con los diccionarios de datos (cuya estructura interna es muy semejante).

"Una red semántica es una red formada por nodos que representan elementos del problema, y arcos etiquetados que representan relaciones entre ellos" [FER94]

Los nodos de una red (elementos de un problema) pueden ser objetos, entidades, acciones o eventos y restricciones. Los arcos que unen a los nodos representan el tipo de relación que guardan éstos. Existen varios tipos de relaciones; dos de las más utilizadas son: ES_UN y PARTE_DE.

La relación ES_UN se refiere a la especificación de un nodo con respecto a uno más general. Por ejemplo, si el nodo etiquetado "ROJO" es un ejemplar del nodo COLOR y se puede representar gráficamente como en la Figura 2.2.

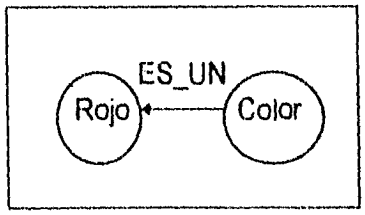


Figura 2.2 Ejemplo de la relación ES_UN en una Red Semántica

La relación PARTE_DE se emplea para definir que un nodo juega un papel de integrante de otro nodo. Por ejemplo, si el nodo etiquetado "DATO" forma parte de un "ARCHIVO", la Figura 2.3 muestra como se vería.

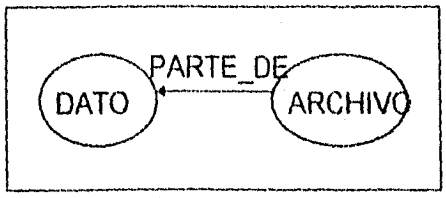


Figura 2.3 Ejemplo de la relación PARTE_DE en una Red Semántica

La forma de representar una red semántica varia según la aplicación y la herramienta con que se cuente. Así, puede programarse usando cálculo de predicados o usando tablas siempre que incluyan atributos para las relaciones que se quieren representar.

Existen sistemas enteros de auxilio al analista y diseñador establecidos sobre modelos de redes semánticas. Al respecto pueden citarse SYS-AIDE, desarrollado por Shemer [SHE86].

2.3.2 Las dependencias conceptuales.

Las redes semánticas resultan ser estructuras muy amplias de representación del conocimiento, lo cual hace que el programador defina el tipo de liga entre los diferentes nodos y aún el contenido de éstos. Un enfoque menos general lo dan las estructuras que se ven en esta sección y en la siguiente, y que son usadas tradicionalmente para el tratamiento del Lenguaje Natural. En la metodología aquí

presentada se eligió este tipo de representación, ya que es el Lenguaje Natural lo que usamos para entendernos con el usuario.

Para representar enunciados comunes que aparecen en el Lenguaje Natural se usa la teoría de las Dependencias Conceptuales (DC). Su objetivo principal consiste en tener una representación interna independiente del lenguaje natural utilizado e inferir de las frases originales el conocimiento explícito e implícito.

El uso de las DC se hace por medio de primitivas y cada una abarca diferentes acciones, como se puede ver en la Tabla 2.1. La combinación de primitivas usando símbolos gráficos (Tabla 2.2), da como resultado el enunciado y las dependencias entre las diferentes categorías conceptuales, que son los elementos involucrados en las acciones (Tabla 2.3).

Si se quiere representar una acción, hay que tomar elementos de la tabla de primitivas para representar el verbo, elementos de la tabla de categorías conceptuales para el objeto directo e indirecto y unirlos con los símbolos gráficos según lo que se diga en el enunciado original.

TABLA 2.1 SUBCONJUNTO DE PRIMITIVAS DE LA TEORÍA DE DEPENDENCIA CONCEPTUALES

Primitiva	Significado	Tipo de acciones que representa
ATRANS	transferencia de una relación abstracta	dar, regalar, pasar
PTRANS	transferencia de una localización física de un objeto	ir, conducir, viajar, transportar
PROPEL	aplicación de una fuerza física a un objeto	empujar, arrojar
MOVE	movimiento de una parte del cuerpo por su dueño	patear, sentar, parar
GRASP	un objeto es empuñado por un actor	asir, coger
INGEST	ingestión de un objeto por parte de un animal	comer, beber, tragar
EXPEL	expulsión de algo del cuerpo de un animal	llorar, escupir
MBUILD	construcción de información nueva a partir de la vieja	decidir, deducir
SPEAK	producción de sonidos	hablar, silbar
ATTEND	concentración de un órgano sensorial hacia un estímulo	escuchar, ver
MTRANS	transferencia de información mental	decir, enseñar

TABLA 2.2 CONJUNTO DE SÍMBOLOS GRÁFICOS UTILIZADOS EN LA TEORÍA DE DEPENDENCIAS CONCEPTUALES

SÍMBOLO	SIGNIFICADO
flechas	dirección de la dependencia
flechas con raya doble	tipos de enlace entre actor y acción
flecha con raya triple	atributos del actor o equivalencia entre dos seres
flecha con la punta invertida	iniciador de la acción

La teoría de DC fue propuesta por Shank y Abelson en 1975 [RIC94] y se ha utilizado en diversos sistemas de comprensión de Lenguaje Natural. Una ventaja en las DC es que permite tanto la representación global de los hechos como la construcción de trozos particulares de información. Sin embargo, resulta demasiado detallado el uso de la dependencia conceptual, de tal forma que Shank y Abelson proponen que sólo sea utilizada para enunciados sencillos [RIC94].

TABLA 2.3 CATEGORÍAS CONCEPTUALES

CATEGORÍA CONCEPTUAL	SIGNIFICADO
PP	Productores de imágenes o protagonistas: objetos físicos, seres vivos.
ACT	Acciones
LOC	Lugares. Donde la acción se desarrolla.
T	Tiempo. Puede ser un tiempo específico o un rango.
PA	Atributos o modificadores de los PP

Para ver como se conjuntan los diferentes elementos conceptuales, se da un ejemplo en la Figura 2.4 que muestra una forma de representar el enunciado "Juan comió el helado". Pueden notarse en la Figura 2.4 representados, además de otros, los siguientes hechos:

- Que el PP (Juan) realizó la ACT, en este caso INGEST (comer en el enunciado original) en el tiempo pasado (p sobre la doble flecha que une a Juan con INGEST, se aplicó la regla uno.
- Además, como lo marca la regla tres (ver la Figura 2.5 y su explicación), la acción ACT va acompañada de un objeto (O) que es PP (helado).

Aplicando la regla cinco, en la parte inferior derecha de la Figura 2.4 se ve que: el ACT (helado) cambió del lugar LOC original (no se sabe de antemano dónde estaba el helado, ya que el enunciado no lo especifica) en dirección D al lugar LOC que ocupa la boca de Juan.

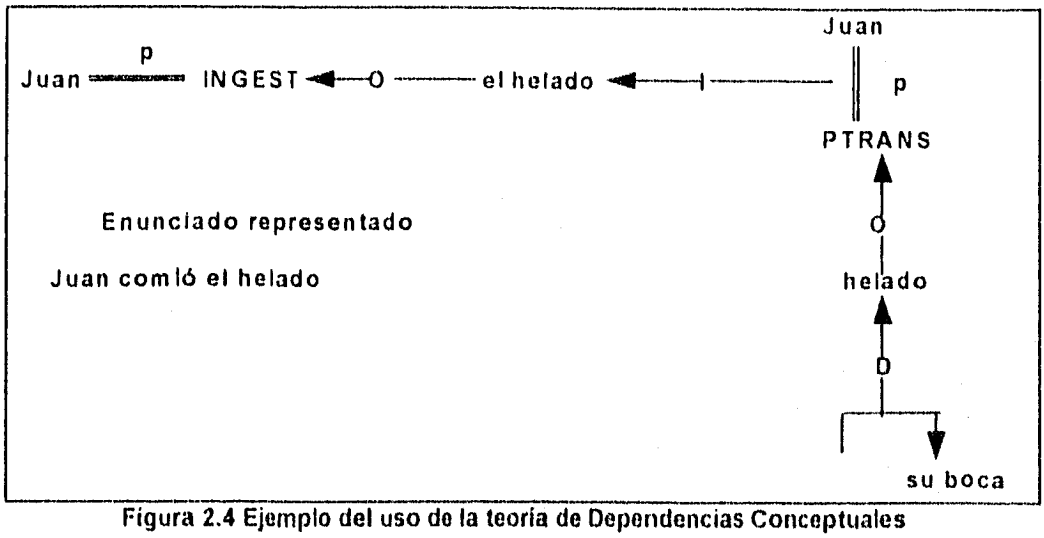


Figura 2.4 Ejemplo del uso de la teoría de Dependencias Conceptuales

Como se puede observar en la Figura 2.4, sobre algunas flechas aparecen otros símbolos; estos representan tiempos conceptuales que modifican la relación, expresando tiempo, modo y otros aspectos verbales. Una lista de tales modificadores se incluye en la Tabla 2.4.

TABLA 2.4 CONJUNTO DE TIEMPOS CONCEPTUALES

SÍMBOLO	TIEMPO CONCEPTUAL
p	Pasado
f	Futuro
t	Transacción
ts	Inicio de la transacción
tf	Fin de la transacción
k	Continuación
?	Interrogativa
/	Negación
nulo	Presente
δ	Atemporal
c	Condicional

Las categorías conceptuales listadas en la Tabla 2.3 pueden combinarse utilizando los símbolos gráficos mediante reglas llamadas de sintaxis conceptual y se muestran en la Figura 2.5. Las reglas de la once a la catorce sirven para representar la causalidad dentro de la DC. A continuación se explica el significado de cada regla:

1. Algún protagonista PP puede realizar cierta acción ACT.
2. El protagonista PP tiene atributos.
3. La acción ACT tiene un protagonista PP.
4. La acción ACT tiene dirección.
5. La acción ACT tiene recipientes.
6. La acción ACT tiene conceptualizaciones como instrumentos.
7. El protagonista PP puede ser descrito por conceptualizaciones en las cuales ellos aparecen.
8. La conceptualización tiene tiempo.
9. La conceptualización tiene localización.
10. Un protagonista PP puede ser equivalente a otro.
11. Una acción puede resultar en un cambio de estado.
12. Los estados pueden permitir acciones.
13. Los estados pueden producir cambios en el estado mental.
14. Los actos mentales pueden ser razones para acciones físicas.

Las dependencias conceptuales resultan de interés para el presente trabajo por su relación con los guiones, que se tratarán en la siguiente sección, y que constituyen un aspecto importante de la metodología desarrollada.

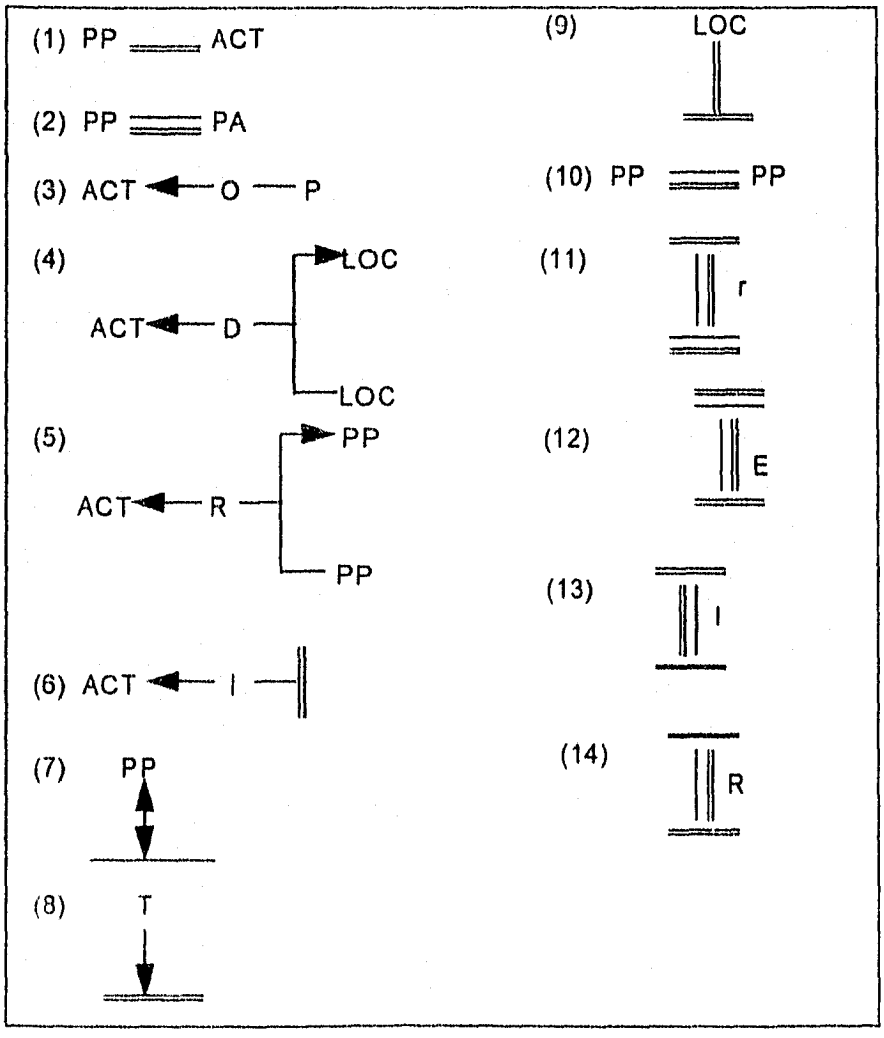


Figura 2.5 Reglas de Sintaxis Conceptual

2.3.3 Los guiones.

Los guiones (traducción de la palabra en inglés Scripts), conocidos también como marcos de situación, fueron propuestos por Roger Schank y Robert Abelson [HAR85] como estructura para el tratamiento de Lenguaje Natural, aduciendo que el ser humano guarda en su memoria conceptos completos y no sólo palabras.

En lugar de proponer una semántica orientada a palabras, se habla de una memoria episódica, la cual está formada por proposiciones que suceden juntas en un evento o intervalo de tiempo. Los elementos son definidos por su lugar en una secuencia de proposiciones que describen eventos asociados a un objeto o a un individuo.
Se dice que la forma en que se relacionan los objetos es a través de situaciones y no como elementos aislados. Un ejemplo: si se le pregunta a alguien, *¿quiénes eran sus amigos en 1968?*, la persona llegará a la respuesta recordando una

serie de acontecimientos que pasaron en esa época y no recorriendo mentalmente una lista de conocidos de toda su vida.

Para trabajar el lenguaje natural empleando guiones se agrupan conjuntos de acciones que desarrollan los objetos de interés dentro de una escena dada. El conjunto de acciones se inicia cuando ocurre un evento, definido por una colección de condiciones de entrada, las cuales no necesitan cumplirse todas.

A partir del evento, las acciones comienzan a ocurrir de acuerdo a un esquema establecido, el guión propiamente, hasta que finaliza la escena de una u otra forma. Al concluir la escena se contará con una serie de resultados o condiciones de salida.

Precisando las partes que conforman un guión, según Rich y Knigth [RIC94], se tiene:

- *Condiciones de entrada.*- Son aquellas que se deben satisfacer antes que se considere que ocurre un evento y puedan realizarse las actividades que se describen en el guión. No todas las condiciones deben cumplirse.
- *Resultados.*- Condiciones que se cumplirán después de que hayan ocurrido algunos acciones dentro del guión. No todas las condiciones se cumplirán.
- *Papeles o actores* .- Protagonistas que aparecen involucradas en los eventos del guión.
- *Utensilios.*- Objetos que aparecen explícita o implícitamente en los eventos del guión.
- *Escenas.*- Conjunto de eventos que forman una parte del guión.
- *Pista.*- Variación especifica de un guión más general. En algunas pistas pueden aparecer algunas componentes que no aparecen en otras pistas y viceversa.

Un ejemplo de guión se puede ver en la Figura 2.5, donde se describe una escena típica de un restaurante en su particularización de cafetería. En ella se puede notar:

1. Como los papeles pueden o no desarrollar las acciones del guión completo.
2. Que se deben cumplir algunas situaciones de entrada para poder "disparar" el guión.
3. Que al salir de él los papeles habrán de terminar en algunas de las salidas.

En el guión de la Figura 2.6, puede notarse cómo se emplean las Dependencias Conceptuales, tratadas en la sección anterior, para la representación de las diversas acciones que ocurren en la escena.

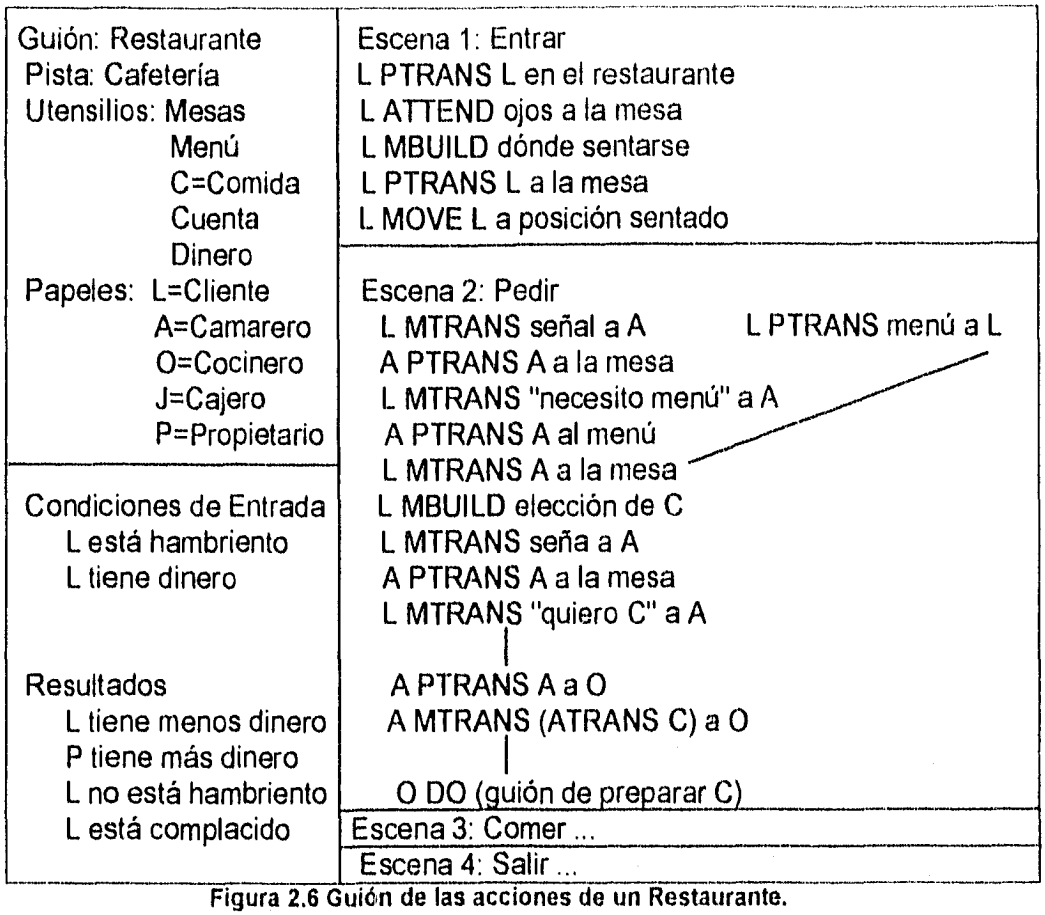


Figura 2.6 Guión de las acciones de un Restaurante.

En la escena de la Figura 2.6, correspondiente a un restaurante, se desarrollan las siguientes acciones, representadas como DC:

Tabla 2.5 Explicación de los eventos de la Escena 1 del Guión Restaurante

REPRESENTACIÓN CONCEPTUAL	SIGNIFICADO DE LA REPRESENTACIÓN
L PTRANS L en el restaurante	El cliente se translada a sí mismo dentro del restaurante.
L ATTEND ojos a la mesa	El cliente revisa con la vista las mesas para escoger una.
L BUILD dónde sentarse	El cliente construye mentalmente una ruta hacia dónde quiere sentarse.
L PTRANS L a la mesa	El cliente se translada a la mesa escogida.
L MOVE L a posición de sentado	El cliente se mueve en el mismo lugar cambiando su posición de parado a sentado.

2.4 Orientación a objetos

Dentro de la metodología Orientada a Objetos (OO) se encuentran resueltas varias de las preocupaciones que se anotaron anteriormente (sección 2.2), tales como la falta de abstracción, el manejo de la herencia, restricciones y actividades (métodos) propios del objeto.

La metodología OO brinda, teóricamente, casi todo los elementos de desarrollo que se necesitan pero, en la práctica actual, los lenguajes OO resultan lentos y consumen una cantidad de recursos muy grande. Además resulta una metodología compleja y aún excesiva para sistemas pequeños como los que constituyen la motivación de este trabajo.

A pesar de todas las limitaciones, posiblemente temporales, la OO nos brinda varios elementos para la planeación del sistema, previos a su implantación y durante las fases de prueba, que pueden ser acogidos fácilmente. Son estos puntos los que se describirán en este trabajo.

2.4.1 Conceptos básicos de Orientación a Objetos.

El objetivo de esta sección es brindar una rápida visión de cómo están conformados los objetos y de la forma en que se comunican o conectan estos para lograr un producto de software completo.

Una definición de objeto es la siguiente:

"Un objeto es una entidad que tiene un estado (cuya representación está oculta) y un conjunto definido de operaciones las cuales operan sobre el estado. El estado se representa como un conjunto de atributos del objeto. Las operaciones asociadas con el objeto dán servicios a otros objetos (clientes) los cuales solicitan estos servicios cuando se requiere algún cálculo" [SOM92].

Como se nota en la definición, las partes que conforman un objeto son:

- los atributos (campos) que lo definen;
- los métodos (procesos) que son propios del objeto; es decir aquellas acciones que puede realizar o que se le aplican para lograr un cambio o transformación de sus atributos o que brindan algún servicio a otro objeto.

Para realizar un sistema orientado a objetos es necesario que los objetos interactúen entre sí. Para ello se usan los mensajes que están formados por:

- Nombre del servicio requerido por el objeto que inicia la comunicación
- Copia de la información que se requiere para la ejecución del servicio solicitado.

Éste concepto de interacción entre los objetos es la que hace atractiva la orientación a objetos, ya que permite una capacidad de abstracción que es necesaria antes de entrar en detalles de diseño, evitando así el perderse en detalles de implantación.

2.4.2 Las pruebas de integración en la metodología orientada a objetos.

La idea principal de las pruebas de integración en la metodología OO es plantear rutas de trabajo [JOR94], mismas que estarán marcadas por los objetivos que se señalaron en el sistema. Los conceptos que se manejan son:

> Un Camino Mensajes-Métodos (**MM-Path**) (del inglés Method/Message Path) es una secuencia de ejecuciones de métodos ligados por mensajes.
>
> Una Función Atómica del Sistema (**ASF**) (del inglés Atomic System Function) se inicia en un evento del puerto[8] de entrada, seguido por un conjunto de MM-Paths, y terminado por un evento del puerto de salida.

Los dos conceptos anteriores se trabajan para la planeación de las pruebas de integración en la OO y resultan muy naturales, ya que las ASF no son otra cosa que los requisitos del sistema.

Los MM-Paths marcan la conexión entre los diferentes métodos de los objetos por medio de los mensajes. Probar un MM-path equivale a ver dónde empieza un evento interno, seguirlo por dónde pasa y verificar dónde termina. Todo lo anterior sirve para verificar que la conexión entre los objetos esté siendo efectiva, especialmente para aquellas rutas que no tienen una salida visible. Un MM-path se representa gráficamente con un círculo vacío donde empieza el mensaje, un círculo relleno a donde va el mensaje y una línea gruesa que los une.

Para entender más claramente cómo funcionan los planes de prueba de integración se considera el siguiente ejemplo:

> Se quiere realizar un pequeño sistema de control escolar que emitirá boletas de calificaciones en cualquier momento y promedios de aprovechamiento por materia y profesor.
>
> Los objetos que se van a considerar son: CALIFICACIÓN, ALUMNO, PROFESOR y MATERIA. Los métodos que cada uno realizará se muestran en la tabla 2.6.

[8] La palabra puerto se refiere en éste caso al punto de conexión externa con un objeto.

Tabla 2.6 Lista de objetos del Ejemplo de Control Escolar

OBJETO	ATRIBUTOS	MÉTODOS
ALUMNO	clavealu, nombrealu, dirección, teléfono, numgpo	nuevo_ingreso, lee_datos, modifica_datos_alumno
CALIFICACIÓN	clavealu, clavemat, calif	ingresa, muestra
MATERIA	clavemat, nombremat	crear, modificar
GRUPO	numgpo, clavemat, claveprof	crear_gpo, incluir_materia_profesor, lee_materias_gpo
BOLETA	cal1,cal2,cal3,cal4,alumno,mat1,mat2,mat3,mat4,facha,numgpo	guarda, imprime
PROFESOR	claveprof, nombre	nuevo_profesor, modifica_datos_profesor

Para este sistema se pueden definir las ASF siguientes:

1. Emitir Boletas.
2. Promedio de aprovechamiento de la materia y profesor dados.

El siguiente paso es dibujar las pruebas de integración para el sistema. A manera de ejemplo, en la Figura 2.7 se presenta la ASF boletas.

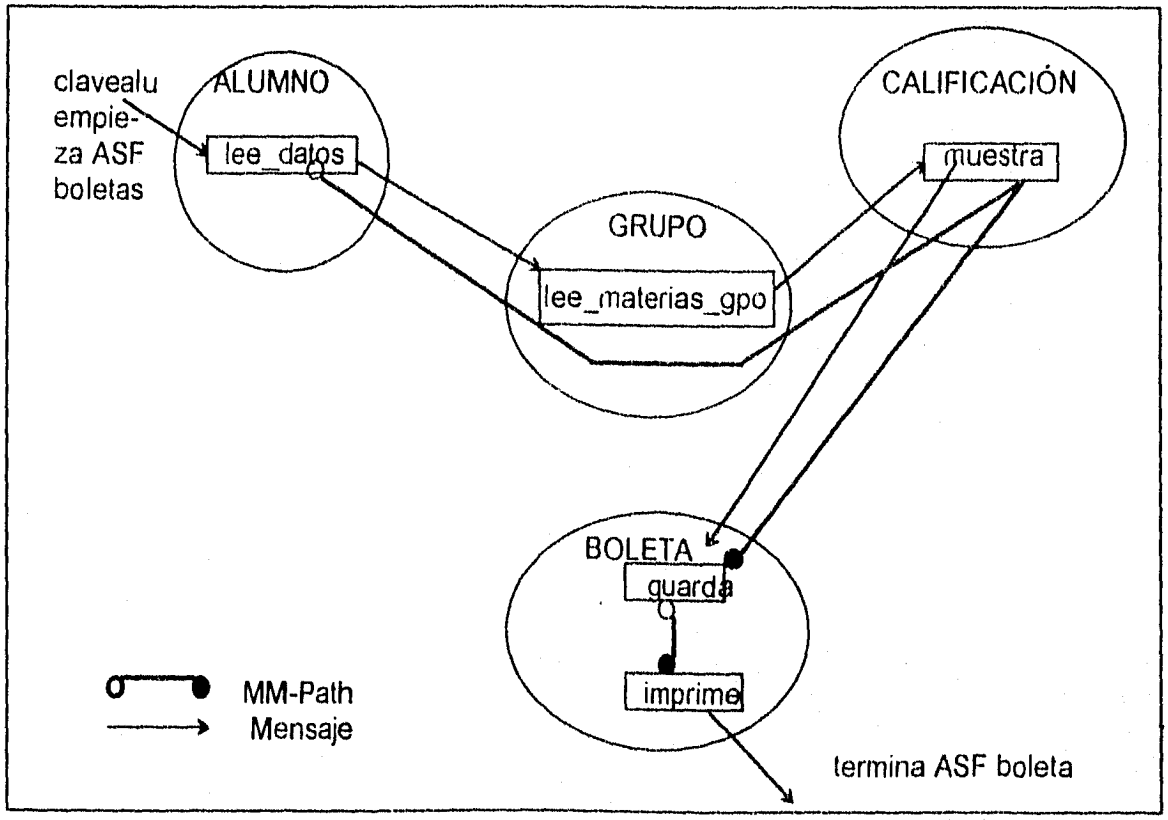


Figura 2.7 Ejemplo de una prueba de Integración Orientada a Objetos

3. METODOLOGÍA PARA EL DESARROLLO DE PAQUETES EN EQUIPOS PC.

Después de desarrollar una serie de productos de software, cada persona va creando su propio estilo de trabajo, fruto de la teoría y la práctica. La experiencia que cada persona va obteniendo puede resultar útil cuando se intenta formalizar y reportar de tal forma que pueda ser entendida y utilizada por otras personas. El resultado del esfuerzo de esta formalización puede resultar en una metodología de desarrollo de software, quizá completamente novedosa o bien formada por elementos viejos y nuevos.

Se acepta que la existencia de otras metodologías de desarrollo pueden cuestionar la validez de nuevas propuestas, pero la idea de formalizar todas las tareas para que se facilite la implantación en la computadora hace atractivo cualquier esfuerzo en pro de éstas.

La metodología aquí propuesta se orienta al desarrollo de software flexible[9] sobre plataformas pequeñas (PC o redes de éstas). La propuesta está fundamentada en elementos de Ingeniería de Software e Inteligencia Artificial y busca abarcar aspectos de calidad desde el momento de la definición del problema. Otro aspecto que se busca con esta metodología es el de, eventualmente, construir con base en ella un CASE, ya que brinda un seguimiento de todos los elementos que la constituyen desde el análisis hasta el mantenimiento.

Esta metodología lleva un orden secuencial para la presentación de pasos, pero puede haber retroalimentación y paralelismo entre ellos. Para llevar a cabo las fases de análisis y diseño es indispensable la presencia del usuario en todo momento.

[9] Sistemas que permiten hacer modificaciones a las estructuras de datos y en cierto grado a los objetivos estando ya en producción. También son conocidos como paquetes.

3.1 El ciclo de vida y la metodología.

La metodología propuesta evoluciona a partir de las fases de la metodología de Flujo de Datos (FD) explicadas en el Capítulo 1 pero complementando las técnicas de representación con guiones. Debe notarse que la metodología de Flujo de Datos resulta efectiva para obtener los productos de software en PC, pero se muestra débil en cuanto a la flexibilidad de los mismos.

La propuesta que aquí se realiza no sólo pretende salvar los problemas de construcción con calidad bajo FD, sino que también obtendrá sistemas fácilmente modificables.

La metodología puede desarrollarse siguiendo el ciclo de vida de cascada o bien puede aprovecharse su filosofía para manejar prototipos. En el esquema de la Figura 3.1 se muestran las relaciones entre el ciclo de vida y las metas que se deben ir cubriendo, de acuerdo a la metodología, para lograr la culminación de las diferentes etapas.

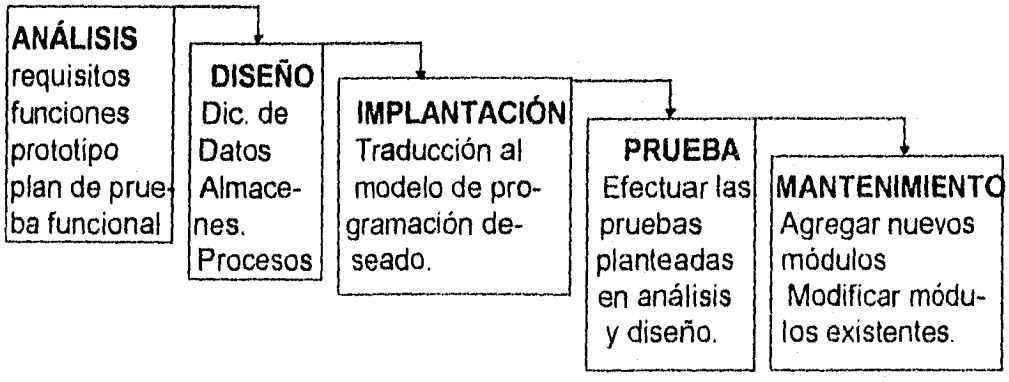


Figura 3.1 El ciclo de vida y las actividades de la metodología propuesta

En diversas versiones de este modelo de ciclo de vida se encuentran otras actividades en cada etapa, pero la metodología propuesta incluye sólo las que se consideran más relevantes e indispensables para el desarrollo de las diferentes etapas. Esta metodología no es excluyente de cualquier otra, así que es posible agregar los elementos faltantes de alguna otra forma.

Las metas mostradas en la Figura 3.1 son claras para una persona que conozca de desarrollo de sistemas . En cambio, los pasos necesarios para su logro muchas veces resultan, en la práctica, difíciles de entender y formalizar. Así, el método propuesto se concentra en precisar una serie de pasos para asegurar el logro de las metas de cada etapa. En la Tabla 3.1 se presentan las actividades recomendadas y su relación con las metas incluidas en las diferentes etapas.

TABLA 3.1 Relación de metas del ciclo de vida y actividades a desarrollar

META	ACTIVIDAD
Establecimiento de requisitos.	Redacción de objetivos generales Guión de la problemática Guión propuesto
Establecimiento de funciones	Se pasan las actividades (quintetas) propuestas en los guiones a una serie de enunciados completos.
Plan de prueba	Se construye el Diccionario de procesos agregando la forma en que estos se comprobarán y el tiempo estimado de realización.
Prototipo	Se realiza un manual de las funciones del sistema a partir de escenas y quintetas señaladas en los guiones. Se puede hacer un prototipo en la computadora.
Diccionario de Datos	Se llena una serie de tablas que pasarán a ser el Diccionario de Datos
Definición de procesos	Se realiza un diagrama de relación entre acciones de las escenas con la notación deseada. Se detalla cada proceso.
Definición de archivos	Se realiza el diagrama Entidad-Relación que utiliza como entidades los papeles, los utensilios y los resultados de las acciones. De ahí, se definen los archivos necesarios.
Traducción al modelo de programación deseado	Se escoge el lenguaje con el que se va a trabajar y se procede a realizar la codificación e inclusión del código en la máquina.
Pruebas	Se procede a realizar las pruebas especificadas en el análisis, diseño e implantación.
Mantenimiento	Se modifica el Diccionario de Datos, se corrige algún error en los módulos existentes o se agregan nuevos módulos.

Como puede verse en la Tabla 3.1, hay mucha relación entre esta metodología y otras formas de trabajo basadas en Flujo de Datos, pero contiene otros elementos que han mostrado su utilidad.

A continuación se describe el método propuesto, etapa por etapa, y se irá ilustrando cada una de ellas con el desarrollo de un ejemplo, mismo que corresponde a un sistema actualmente en funcionamiento. Antes de comenzar, se muestran los elementos importantes que se emplearán en la aplicación de la metodología.

3.2 Estructura general de la metodología propuesta.

Ésta metodología propone realizar aplicaciones (sistemas) sobre la base de un metasistema. El metasistema almacena en un Diccionario de Datos toda la información que define al sistema. En cualquier momento, a través del metasistema, el sistema puede modificarse.

Podría resultar innecesario contar simultáneamente con el sistema y el metasistema, por lo que la implantación puede considerar mantenerlas aisladas y eventualmente hacerlas interactuar.

En relación con las actividades mencionadas en la sección del ciclo de vida de la metodología se tiene una serie de elementos que caracterizan esta metodología las cuales se muestran gráficamente en la Figura 3.2.

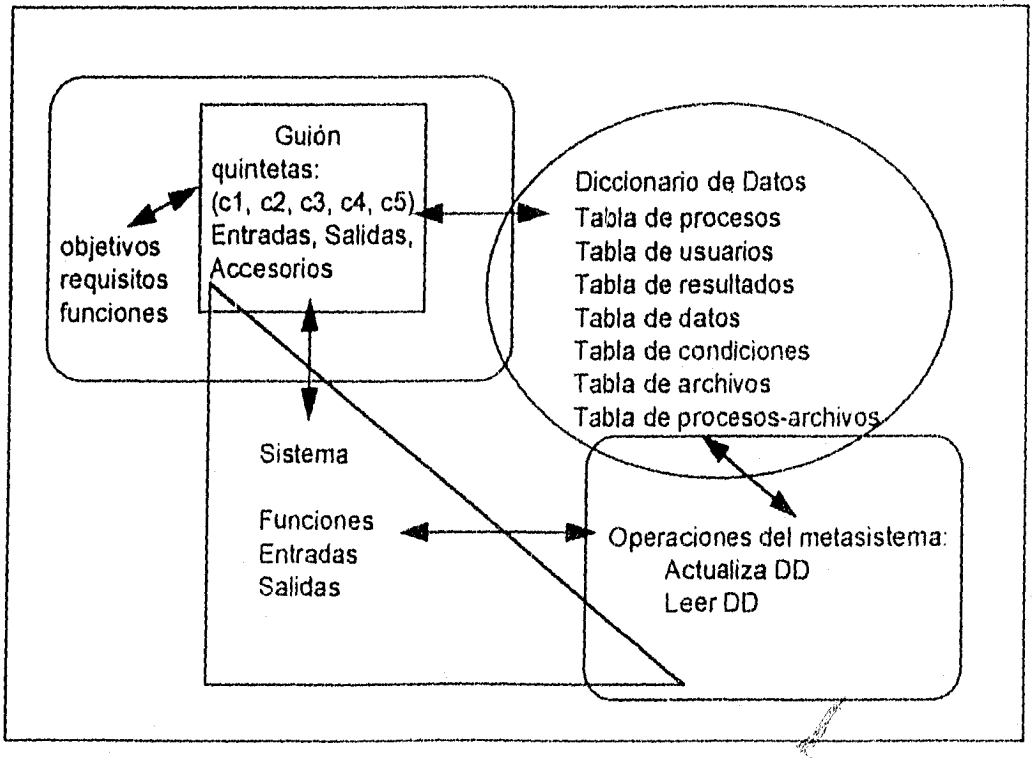


Figura 3.2 Estructura general de interacción de los elementos de la metodología propuesta.

Como se ve en la Figura 3.2, existen cuatro grandes bloques:

1. En el primero, de la esquina superior izquierda, se pueden encontrar elementos del análisis que, con ayuda de los guiones, llegan a establecer los objetivos, requisitos y funciones del sistema.
2. En un círculo se encerraron los elementos que conforman el Diccionario de Datos (DD), que fue iniciado en la etapa de análisis. El DD servirá como apoyo tanto al análisis como al metasistema.

3. El metasistema, esquina inferior derecha, estará formado por dos operaciones solamente: la actualización al DD y su lectura, que servirá para comunicar el DD con el Sistema.
4. Por último, el sistema realiza todas sus funciones utilizando los metadatos definidos en el DD mediante llamadas al metasistema. Las funciones, entradas y salidas del sistema serán el producto de la interacción de los elementos del análisis.

3.3 El análisis en la metodología propuesta.

La parte principal de un análisis es el establecimiento de objetivos, metas y requisitos. Con la metodología aquí propuesta, se cubren de manera natural y hasta entretenida, ya que sugiere manejar una serie de guiones que dan la idea al analista y al usuario de estar escribiendo un pequeña pieza teatral, en donde este último es protagonista mientras que el analista es el director de la obra, que mediante la observación, va acomodando las acciones de los diferentes participantes de la trama. Esta trama es lo que el usuario realiza actualmente y lo que hará con el nuevo sistema.

Entonces, ya que el objetivo principal de todo análisis es establecer la problemática de la situación que se está presentando y brindar una serie de posibles soluciones, en esta sección se van señalando los pasos que deben darse para lograrlo.

3.3.1 Definición de los objetivos preliminares del sistema.

Para comenzar el trabajo hay que redactar lo que, en principio, desea el usuario. Para ello será necesario realizar una serie de entrevistas que llevará al establecimiento de objetivos; durante las entrevistas se determinará el problema a resolver y se establecerán las fronteras del sistema y las responsabilidades del software.

Los objetivos deben presentarse utilizando el lenguaje del usuario, evitando tecnicismos que hagan perder la idea de para qué se va a desarrollar el análisis del sistema.

3.3.2 Definición del problema.

Es necesario establecer cuál es la problemática a la que se enfrenta el sistema actual, para ello se hará uso de la notación de guiones que se indica a continuación:

> Pista.- nombre del subsistema, el cual pertenece a un sistema más grande cuyo nombre es el del guión. Si el sistema es pequeño, se sugiere que la pista no se especifique[10].
> Papeles o actores.- Los usuarios del sistema, tanto los que lo operan directamente como aquellos que usan sus resultados (jefe, clientes).
> Accesorios o utensilios.- Son los documentos y medios físicos que se emplean en el sistema.
> Lista de entrada.- Aquellos accesorios que se necesitan para empezar el trabajo del sistema.

[10] Aunque se está tratando en éste trabajo de sistemas pequeños, no se descarta el que al menos la parte del análisis pueda manejarse en sistemas más grandes.

Lista de salida.- Los resultados, tanto en documentos como en pantalla u otro medio, que emite o produce el sistema.
Escena.- Explicación de cada etapa del sistema. No se utiliza la notación de dependencias conceptuales, ya que estos guiones serán mostrados y utilizados por el usuario. La notación de dependencias conceptuales es un poco obscura para las personas ajenas a la computación. La explicación de cada escena se debe hacer mediante una lista de *quintetas* (c1,c2,c3,c4,c5), donde

c1 = papel o actor que realiza las diferentes acciones (p. ej. Secretaria, Administrador),

c2 = verbo que describe la acción (p. ej. capturar, requerir),

c3 = utensilio resultante de la acción (p. ej. factura, nómina),

c4 = utensilio que se ocupa para realizar la acción (p. ej. computadora) puede aparecer o ser implícito,

c5 = tiempo entre ocurrencias de la acción (p. ej. 15 días), puede no aparecer.

El analista debe tener en cuenta que un guión es una secuencia estereotipada de acciones sobre un contexto específico; por lo que en este momento se trata de reflejar las acciones que conforman al sistema. El dibujo de la Figura 3.3 muestra como puede representarse el guión, de manera general.

Otros elementos que se muestran en el guión son tres tipos de líneas: línea continua representa una secuencia entre quintetas, línea punteada representa salto incondicional y línea doble se usa para salto condicional.

Al terminar de estructurar el guión, el resultado debe mostrarse y explicarse al usuario para que él identifique positivamente la representación y señale en dónde se encuentran los problemas del sistema actual. Esto permitirá identificar aquellos puntos donde se podrá mejorar computacionalmente la situación actual del usuario con respecto al sistema y aquellas partes en las que la computación no ayudará, ya que depende de los usos y costumbres de la empresa y que el analista sólo se limitará a señalar y dependerá del usuario realizar o no las modificaciones.

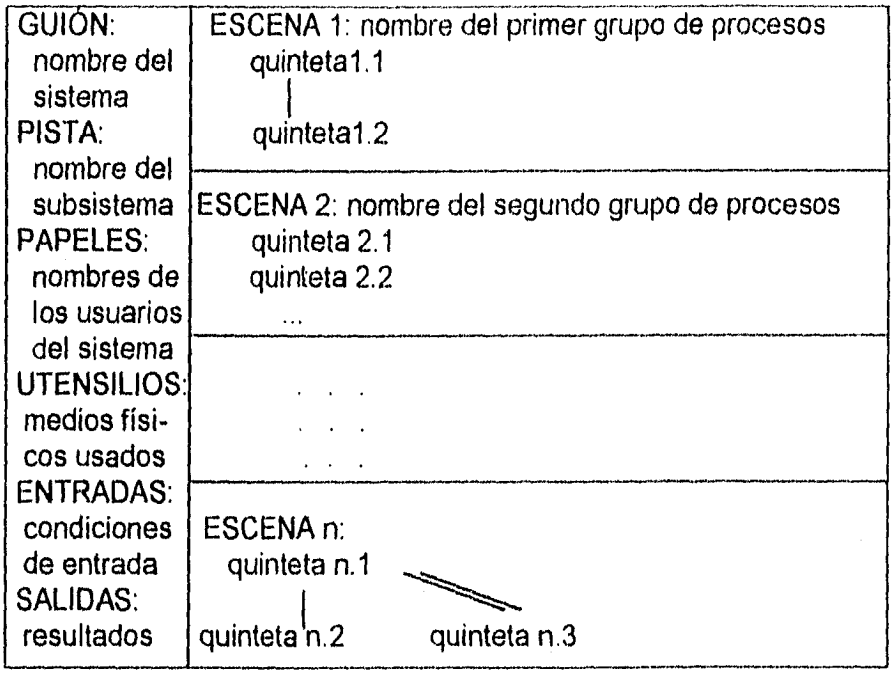


Figura 3.3 Esquema de un Guión

3.3.3 Estrategia computacional.

Una vez localizados los puntos problema del sistema actual, se procede a realizar el guión del sistema propuesto de la misma manera que el anterior, sólo que en esta ocasión reflejará la forma en que el usuario, el hardware y el software interactuarán.

3.3.4 Identificación de funciones.

La identificación de funciones se refiere a precisar las responsabilidades del hardware, del software y de los seres humanos dentro de un sistema automatizado. Esta parte resulta sencilla, ya que en el guión fueron descritas las escenas y en ellas se especificaron los actores de cada acción. Así, aquellas quintetas donde aparece el usuario indicarán sus acciones, las quintetas donde aparecen elementos del hardware indican el papel de éste dentro del sistema, por último el software tendrá como funciones generales cada una de las escenas especificadas en el guión. A manera de ejemplo:

> En el sistema de nómina de una compañía pequeña, se tienen las siguientes Escenas:
>
> Reunir manualmente los datos, Capturar, Revisar, Emitir y Conciliar.

Entonces, las funciones del software serán:

> *"El sistema brindará facilidades de captura y revisión de los datos. Emitirá la nómina y una vez que se ha pagado a los trabajadores, el sistema será capaz de conciliar lo pagado con lo emitido".*

Supóngase también que se tiene, las quintetas siguientes:

> (Secretaria, captura, categoría de cada persona, computadora)
> (Administrador, requiere, nómina, computadora, 15 días)

Entonces dentro de las actividades del usuario se tendrán:

> *"El usuario introducirá los datos necesarios (clave de personal, nombre, antigüedad) para que el sistema funcione. El administrador será el encargado de emitir la nómina cuando él lo quiera y cada vez será la correspondiente a la última quincena".*

Basándose también en las quintetas anteriores podremos definir que las funciones del hardware serán:

> *"El hardware será capaz de almacenar los datos de la nómina y de permitir la captura e impresión de los datos y listados de nómina necesarios".*

3.3.5 Establecimiento de requisitos.

El establecimiento de requisitos es una de las metas más importantes del análisis, ya que estos representan lo que el usuario desea, en forma comprobable. Son de utilidad también en el establecimiento del plan de pruebas y hasta para los acuerdos de aceptación del sistema terminado. De acuerdo a la metodología que aquí se describe, los requisitos se hallan incluidos en los guiones. Cada quinteta de las escenas que producen un resultado de la lista de salida será identificada como un requisito. Así, la quinteta

> (Administrador, requiere, nómina, computadora, 15 días)

pasará a ser el requisito.

> *"El administrador podrá requerirle a la computadora la emisión de la nómina cada 15 días"*

3.3.6 Plan de prueba.

Para probar el sistema, se elaborará una bitácora por cada quinteta identificada como requisito y se le agregarán tres columnas: tiempo estimado de realización, tiempo real de realización y lista de pruebas a realizar. La columna de tiempo real de realización se refiere al tiempo en que se terminó de hacer las pruebas funcionales y de estructura a cada Acción descrita en esta tabla. A la larga esta columna servirá para llevar un control del desarrollo de futuras aplicaciones, permitiendo así organizar mejor el tiempo.

Ejemplo:

Usuario	Acción	Docu-mentos	Medio	Tiempo desarrollo	Tpo. real	Lista de comproba-ciones
Secretaria	captura	datos	compu-tadora	un mes		Debe validar rangos de antigüedad y validar con catálogo la categoría del empleado.
Administra dor	requiere	nómina	compu-tadora	dos semanas		se revisará con nóminas emitidas en el sistema anterior.

3.3.7 Manual de operación.

El objetivo principal de incluir el manual de operación en la etapa de análisis se debe a que aclara de manera muy amplia, al usuario y al analista, la forma en que todos los elementos van a interactuar. Sirve para aclarar aquellos puntos que estaban "sobrentendidos". En general, los "sobrentendidos" resultan en confusiones y conflictos posteriores. Es claro que, el manual definitivo se completará después que se termine la programación.

Por otro lado el manual de operación sirve como el primer prototipo del sistema, y debe incluir:

- Una introducción en que se muestre una visión general del sistema.
- El índice general. Que será la lista de escenas.
- La explicación de cada escena se hará poniendo las pantallas o documentos que se utilizarán, los cuales vienen en cada quinteta. Para cada quinteta, redactar una explicación, anotando si la acción está restringida al actor que ahí aparece o no.
- Para cada accesorio, realizar un dibujo (ejemplo: boleta de venta) o fórmula (ejemplo: fecha de emisión=día+mes+año) de cómo está formado.

Debe aclararse que los puntos anteriores se refieren sólo al contenido del manual de operación. La forma de desarrollarlo será integrando los elementos de tal forma que resulten amenos al usuario, para lo cual a continuación se listan algunas recomendaciones tomadas, en su mayoría, del libro de Somerville [SOM88].

- Debe ser realista acerca de las capacidades del sistema, sin destacarse en demasía las características novedosas o "muy poderosas".
- Debe estructurarse como sigue:
    1. Una descripción funcional (general) sobre lo qué puede hacer el sistema.
    2. Una descripción de cómo instalar el sistema. Puede hacerse de manera tentativa.
    3. Una parte en que se explique de manera sencilla la funcionalidad del sistema.
    4. Una sección que sirva como manual de referencia en la cual se detallan claramente cada una de las funciones.
    5. Una guía que explique cómo reaccionar ante situaciones surgidas mientras el sistema se encuentra en uso. Es obvio que esta parte no puede hacerse completa en el análisis del sistema, pero puede iniciarse.
- Debe pedirse apoyo de escritores técnicos profesionales.
- Respecto a la redacción del manual, existen las siguientes recomendaciones:
    - Utilizar formas activas en lugar de pasivas. Es preferible "verá un cursor en la esquina izquierda superior ...", en lugar de "un cursor será visto en ...".
    - Deben evitarse las frases largas que incluyan varios hechos distintos.
    - La referencia a otras secciones debe hacerse con el nombre y número de éstas.
    - Si puede explicar su idea con cinco palabras, no use diez.
    - Los párrafos cortos en un manual son mejores.
    - Los títulos y subtítulos deben aparecer claramente.
    - Se debe cuidar la ortografía y las construcciones gramaticales sin abusar de los infinitivos, participios pasados o gerundios.

3.3.8 Llenado de Diccionarios en el análisis.

El Diccionario de Datos es un elemento que posteriormente servirá de enlace con el diseño. En este nivel se llenan tres tablas con los datos de las quintetas:

1. De actores. Lista de parejas (actor, acción).
2. De resultados. Lista de parejas (accesorio, lista de datos que lo componen).

3. De datos. Lista de parejas (nombre, descripción).

Las tablas se llenan con las columnas que se proponen entre paréntesis y servirán para aclarar las funciones de los elementos mencionados en el guión y los requisitos.

3.4 Análisis del Sistema Conserva.

En esta sección se mostrará un ejemplo de aplicación de la metodología propuesta y se hará con un sistema que está trabajando actualmente en varias instalaciones. No se detallará todo el sistema para evitar que se haga pesado el ejemplo y el lector pierda el seguimiento general.

El sistema **Conserva** fue realizado en 1990 para Conservation International con el fin de apoyar a los Centros de Datos que existen en varios Estados de la República Mexicana. A continuación se muestran las diferentes etapas por las que pasó en su desarrollo, en la forma que se propone en este trabajo. Cuando se hable en presente o se refiera a "la situación actual" se estará hablando del tiempo en que se desarrolló el sistema. De igual manera, cuando se hable en futuro se tratará del sistema terminado.

3.4.1 Objetivos preliminares.

Los Centros de Datos dependientes de Conservación Internacional tienen la necesidad de crear y mantener bases de datos de diferentes grupos de especies, de manera que, eventualmente, puedan combinarse. Cada centro tiene características especiales, que deben reflejarse en cierta variación de las bases de datos. Anteriormente, se emplearon diversos programas que realizaban parte de la captura, pero que resultaron inadecuados, por lo que se buscó contar con un sistema flexible, capaz de adecuarse a los diferentes centros.

El sistema **Conserva** brindará una solución al problema mencionado, permitiendo la definición, captura y explotación de datos para la conservación de especies.

3.4.2 Definición del problema.

A continuación se muestra cómo es el sistema actualmente, o mejor dicho cómo se trabaja actualmente en los diferentes centros de apoyo a la conservación de especies. Se hace el guión correspondiente, el cual se ilustra en la Figura 3.4.

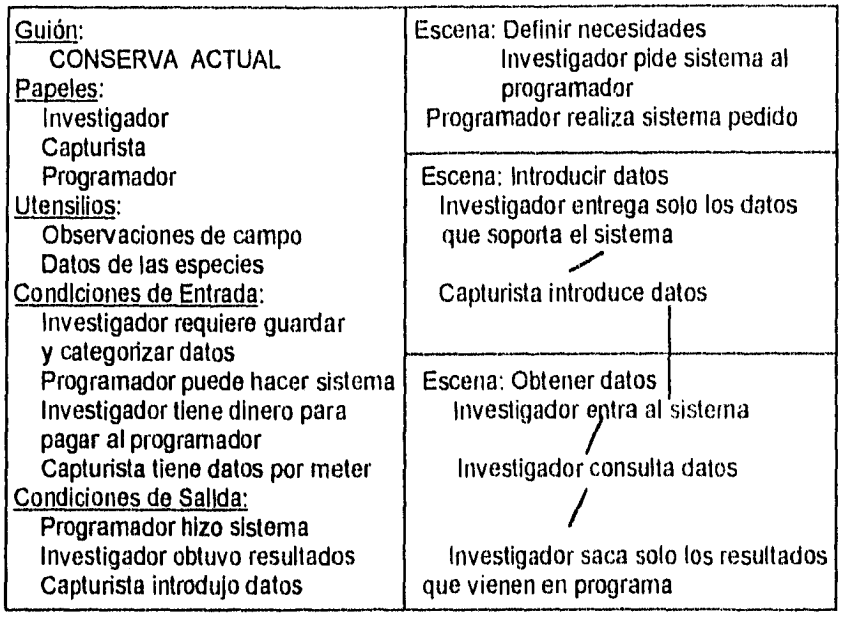


Figura 3.4 Guión correspondiente a la situación actual

3.4.3 Problemas principales.

Cada sistema realizado en cada centro cubre sólo las necesidades particulares del centro. En cuanto se necesita una modificación hay que volver a contratar programadores, que con frecuencia comienzan desde el principio o les cuesta mucho trabajo entender los programas viejos.

El formato de los archivos de datos de los diferentes sistemas es al gusto del programador y con frecuencia son incompatibles con los de otros centros.

Los formatos de captura resultan inflexibles, ya que si se quiere agregar un dato nuevo, esto no se logra.

3.4.4 Estrategia computacional

Dados los problemas descritos, se decidió realizar un paquete flexible que permita al usuario ir definiendo sus propios datos y metas. La forma de trabajar con un nuevo sistema computacional se muestra en el guión de la Figura 3.5.

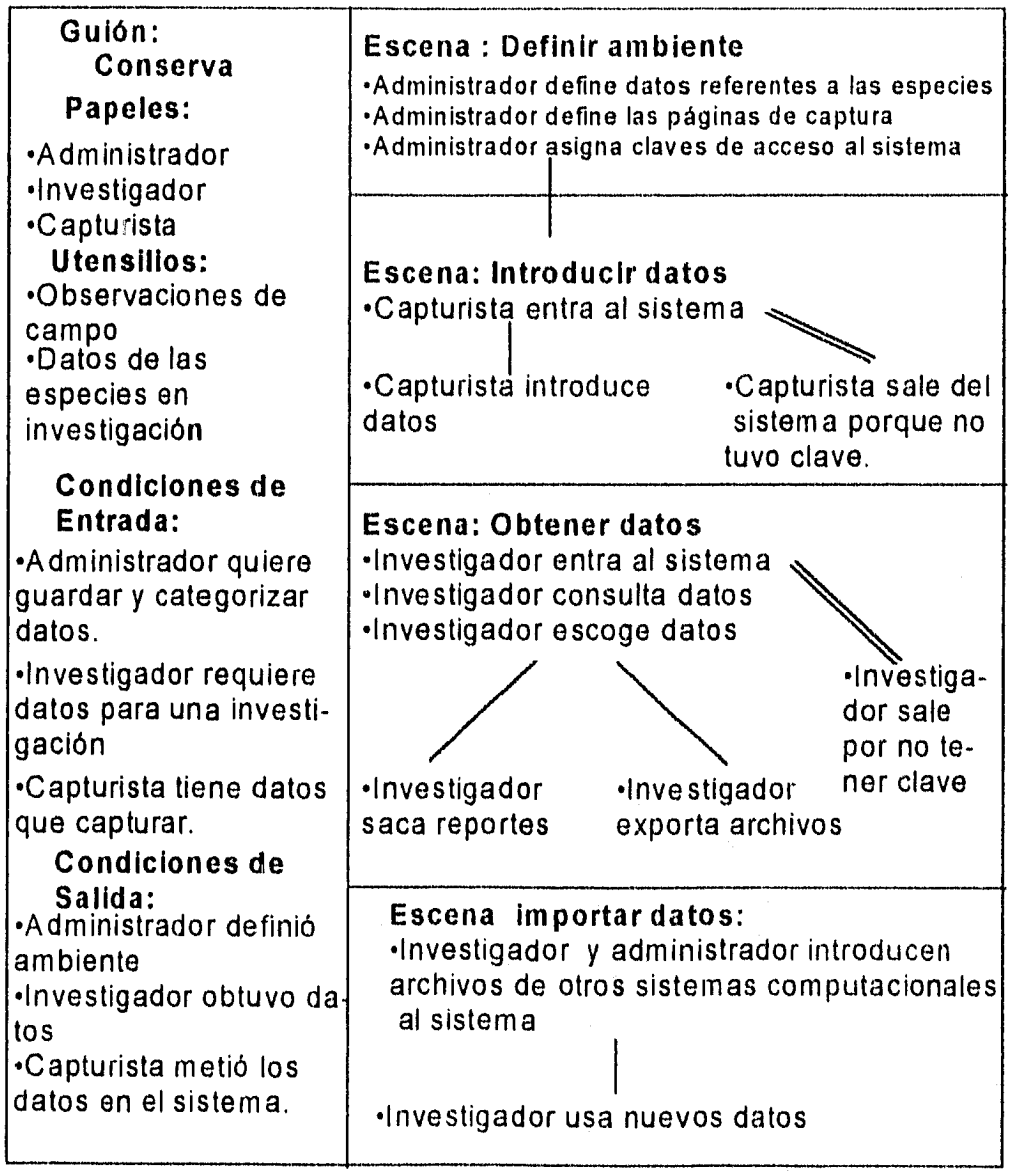


Figura 3.5 Estrategia Computacional para el sistema Conserva

Para que los guiones de las Figuras 3.4 y 3.5 queden más cercanos al usuario, no se utilizó una notación formal de las quintetas; es decir, no se empleó la notación entre paréntesis y separados los elementos con comas. Sin embargo para el analista sería conveniente que las pase a la notación formal para que identifique claramente los elementos que intervienen en el sistema. A continuación se ejemplifican algunas quintetas:

Escena 1, quinteta 1.- (Administrador, define, datos_especies).
Escena 1, quinteta 2.- (Administrador, define, páginas_captura).
Escena 3, quinteta 3.- (Investigador, escoge, datos, archivos).

3.4.5 Identificación de funciones.

El software.- Será capaz de recibir definiciones de datos, actualizaciones de archivos, importación de archivos creados con Dbase y exportación de archivos del **Conserva** a los formatos Dbase y texto.

El usuario.- Habrá tres tipos de usuarios con diferentes funciones y serán:

El administrador.- Definirá los datos y los accesos a ellos, además de poder realizar las funciones de los otros tipos de usuario.

El capturista.- Introducirá, revisará y corregirá los datos.

El investigador.- Consultará datos y los podrá imprimir o exportar a otros formatos.

El hardware.- Almacenará datos.

3.4.6 Establecimiento de requisitos.

A continuación se pone en lista la serie de requisitos de los usuarios que se establecieron para el sistema **Conserva.**

El administrador podrá definir los datos que manejará el sistema en tres bloques: (1) especies, (2) observaciones y (3) catálogos.

El administrador también podrá indicar el orden en que los datos deben aparecer en la pantalla de la captura.

El administrador podrá asignar claves de acceso a los usuarios del sistema, pudiendo cambiar o eliminar las que desee.

El capturista podrá entrar al sistema sólo después de identificarse mediante su clave.

El capturista podrá: introducir datos incompletos, corregir datos y obtener ayuda en línea durante la captura.

El investigador podrá entrar al sistema sólo después de identificarse mediante su clave.

El investigador podrá consultar los datos y escoger los que desea.

El investigador podrá sacar reportes de sus datos o exportarlos a otros paquetes.

3.4.7 Plan de prueba funcional.

Del guión de la Figura 3.5, de la experiencia del analista y de los acuerdos tomados con el usuario, queda el siguiente plan de prueba.

Actor	Proceso	Tiempo de realización	Tiempo real[11]	Lista de pruebas a realizar
Administrador	definir datos	dos meses		definiciones duplicadas
Administrador	definir captura	dos meses		respetar orden de captura especificado
Administrador	asignar claves de acceso	un mes		evitar duplicados
Capturista	identificar capturista	un mes		nivel de acceso a la captura y a los datos existentes
Capturista	introducir datos	dos meses		brindar ayuda, validar campos, incluir en cada registro la identificación del capturista, fecha y folio.
Investigador	identificar investigador	un mes		nivel de acceso sólo a los datos ya existentes
Investigador	consultar datos deseados	tres meses		permitir operaciones de selección y proyección sobre los datos con ayuda.
Investigador	exportar datos	un mes		los datos exportados estén en el formato

3.4.8 Llenado de diccionarios.

La lista de tablas que se deben llenar en la etapa de análisis se muestran a continuación:

De actores.

Actor	Proceso
Administrador	definir datos
Administrador	definir captura
Administrador	asignar claves de acceso
Capturista	identificar capturista
Capturista	introducir datos
Investigador	identificar investigador
Investigador	consultar datos deseados
Investigador	exportar datos

[11] Esta columna deberá llenarse en el momento de la implantación del sistema.

Resultados.

Accesorio	datos que lo componen
Listado de catálogo	clave, nombre, otros
Listado de especies	nombre_científico, familia, otros
Listado de observaciones	nombre_científico, fecha, folio, capturista, otros
Archivo deseado	datos elegidos

Datos.

Nombre del dato	Descripción
Clave	En cada catálogo habrá una clave de identificación para cada registro. Los catálogos serán definidos por el usuario y contendrán datos referentes a la investigación. Ejemplo, catálogos de: familias, nombres científicos, usos de la planta,
Nombre	En cada catálogo habrá un nombre por cada clave de identificación para cada registro.
Nombre_científico	Se refiere al nombre en latín de la especie y cada investigador será responsable por su introducción. SI el nombre ya existe se usará la clave que antes se le asignó en el catálogo.
Familia	Se trata de la familia a la que pertenece la especie.
Folio	Número secuencial que se le asigna a cada observación de una especie en el momento de la captura. Será generado por el sistema.
Fecha	Fecha de captura de la observación
Clave_secreta	Contraseña secreta de cada usuario.
Iden_capt	Identificación (clave de acceso) de la persona que capturó

3.5 El Diseño en metodología propuesta.

El diseño de paquetes con la metodología propuesta se divide en dos subsistemas, mismos que se comunican entre sí a través de las estructuras que forman el Diccionario de Datos:

1. El del **metasistema,** que dará soporte y flexibilidad al paquete.
2. El del **Sistema**, que será el que se comunique con el usuario y saque los resultados usando los metadatos del metasistema.

La explicación de cada parte se hará por separado pero, en realidad, se llevan a cabo en paralelo con el apoyo en el modelo Entidad-Relación, el cual irá mostrando tanto las entidades o estructuras de diccionario como las entidades del sistema que serán necesarias y las relaciones entre ellas.

3.5.1 Diseño del Metasistema.

El metasistema para un paquete, desarrollado bajo esta metodología, contiene descripciones de diversos elementos: resultados, procesos, archivos, operadores, administradores condiciones y datos. Todos ellos se relacionan en la forma que describe la red semántica de la Figura 3.6.

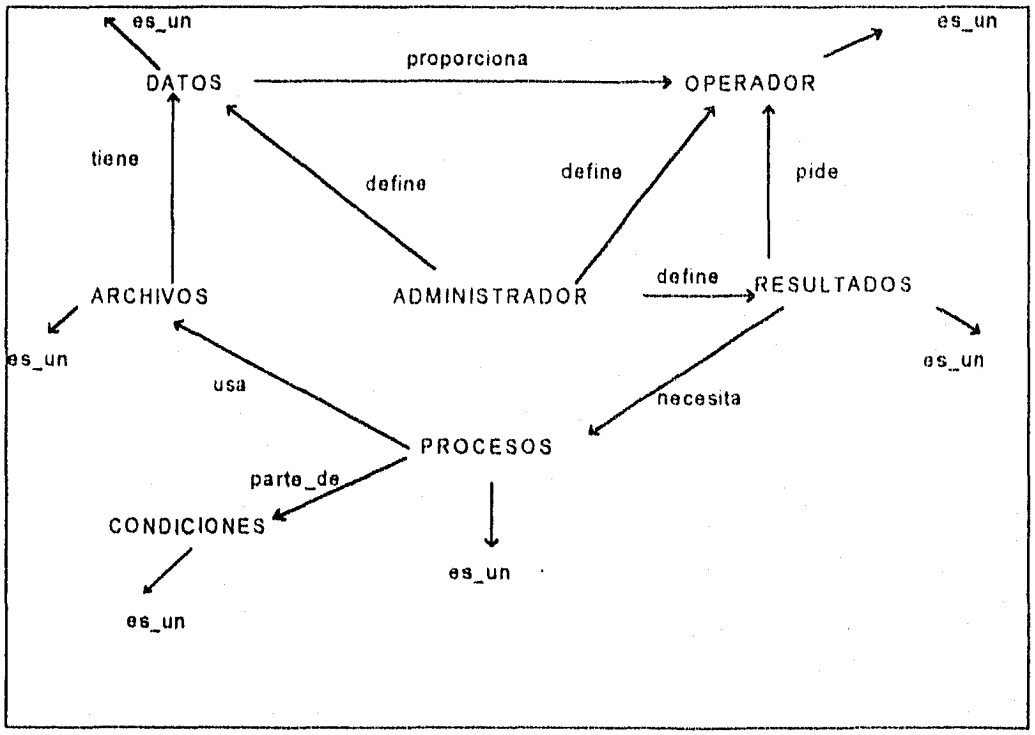


Figura 3.6 Red semántica para el Diccionario de Datos

El *administrador* es el usuario con privilegios para definir metadatos, nuevos usuarios (operadores) y nuevos resultados. No todos los sistemas permitirán tener un administrador con todos los privilegios antes mencionados, pero sí algunos. Sin embargo, cuando se quiera realizar una actualización a los procesos (modificación), el papel del administrador lo tomará el analista-diseñador encargado del mantenimiento del sistema.

Las relaciones **es_un** reúnen a los ejemplares de los nodos *datos, archivos, procesos, usuarios y resultados* que representan a un sistema particular.

3.5.1.1 La estructura del Diccionario de Datos.

La representación de la red semántica se hace mediante el Diccionario de Datos y su estructura de datos será una serie de tablas que pueden definirse como se sugiere en esta sección. En la Tabla 3.2 se listan las principales. Estas tablas se llenan a partir del guión y las tablas que se generaron durante la etapa de análisis.

Cada tabla tiene sus objetivos de interacción con el sistema y de refuerzo de la calidad de este, ya que con cada una de las tablas se pueden realizar acciones dentro del programa que obliguen a cubrir diferentes aspectos. En la Tabla 3.2, además de los objetivos, se indica la relación entre las tablas del metasistema y los factores de calidad[12] al que afectan positivamente.

Tabla 3.2 Relación entre Diccionario de Datos y Factores de Calidad.

TABLA	OBJETIVOS QUE CUBRE	FACTORES DE CALIDAD
Condiciones de Entrada	• Verificar que el hardware esté disponible • Verificar que existan los archivos intermedios que se van a utilizar	Funcionalidad Confiabilidad
Escenas y quintetas	• Presentación de menúes • Seguridad de acceso a las funciones	Funcionalidad Usabilidad
Resultados	• Emisión de resultados	Soportabilidad
Datos	• Modificación de las estructura • Listar las descripciones de los campos cuando sea necesario • Validar el contenido de los campos	Soportabilidad Funcionalidad Confiabilidad Usabilidad
Usuarios	• Controlar el acceso a los procesos.	Funcionalidad
Archivos	• Verificar que no hayan sido alteradas las estructuras en forma externa al sistema	Funcionalidad Soportabilidad
Procesos-archivos	• Verificar la existencia de los archivos antes de empezar un proceso	Funcionalidad Usabilidad

[12] Se considerarán los factores de calidad llamados FURPS (ver seción 1.3.3.1.).

Debe advertirse que no siempre será necesario crear todas las tablas. En algunos sistemas pueden sobrar algunas, o bien existir en otra forma en el software que acompaña al sistema en desarrollo. Por ejemplo, algunos lenguajes asociados a manejadores de bases de datos ya cuentan con mecanismos para conocer los campos de un archivo y sus características, al tiempo de ejecución.

A continuación, se mostrará la forma de cada tabla y la conexión que tiene con el análisis. La metodología obliga a que el diseño sea acorde con el análisis aceptado por el usuario.

Para ilustrar el llenado de tablas se usará el siguiente ejemplo: Suponga que se tiene un sistema hospitalario, dentro del cual se estableció el guión de la Figura 3.7.

GUIÓN: Control de historiales	Escena 1: Actualiza archivos Médico actualiza catálogo de enfermedades Administrador actualiza archivo de médicos
PAPELES: Enfermo Médico Enfermera Administrador ACCESORIOS: Computadora Forma de ingreso al hospital recetarios Control del enfermo ENTRADAS: Enfermo sufre enfermedad Médico sabe curarla SALIDAS: Enfermera registra al enfermo en el hospital Médico atiende al enfermo Enfermo adquiere receta	Escena 2: Ingresa Enfermo Enfermera revisa el médico que puede atender al enfermo Médico solicita historial del enfermo que se le asignó Médico diagnostica enfermedad y llena receta Enfermo se queda en el hospital — Enfermo sale del hospital Enfermera actualiza los datos del enfermo
	Escena 3: Emisión de reportes Administrador solicita historial administrativo del enfermo Médico solicita estadísticas de una enfermedad Administrador solicita estadísticas del médico

Figura 3.7 Guión de un sistema para el control de historiales médicos y estadísticas de enfermedades.

Tabla de Procesos.

Esta tabla se hará con base en las escenas y sus quintetas definidas en el guión de la estrategia computacional.

Núm. de escena	Núm. de quinteta	Nombre de la escena o la quinteta	clase del Operador	Apuntador a la primera condiciones de entrada

Ejemplo: de acuerdo al guión de la Figura 3.7, se tiene la siguiente tabla de procesos:

Núm. de escena	Núm. de quinteta	Nombre de escena o quinteta	Clase del operador	Apuntador a la primera condición
1	0	Actualizar archivos	2	1
1	1	Catálogo de enfermedades	2	2
1	2	Archivo de médicos	1	4
2	0	Ingresar Enfermos	3	6
2	1	Revisar médicos	3	8
2	2	Solicitar historial	2	9
2	2	Emitir diagnóstico	2	10
2	3	Actualizar datos de internos	3	12
3	0	Emisión de reportes	2	15
3	1	Historial administrativo	1	17
3	2	Estadísticas de enfermedades	2	20
3	3	Estadísticas de médicos	1	25

El campo *clase de operador* puede ser un número, como en el ejemplo. En ese caso, al ser uno indica que tiene acceso a todas las funciones del sistema y conforme va creciendo indica una disminución en sus privilegios.

Tabla de Condiciones de entrada a un proceso.

Un aspecto importante en el desarrollo de sistemas en PC es la validación que debe hacer el sistema en todo momento de contar con los elementos necesarios para su correcta operación. En otras metodologías no siempre queda explícito el ambiente necesario para trabajar. En la metodología descrita, por su herencia de los guiones, se insiste en la necesidad de precisar las condiciones de entrada. Así, la tabla de condiciones de entrada a procesos sirve para asegurar que al entrar a un proceso se tengan todos los elementos (hardware, archivos, permisos) necesarios para que se lleve a cabo. Su contenido es como sigue:

Nombre de la condición	proceso a la que pertenece	tipo de restricción (opcional, requerida)	condición siguiente

Ejemplo: La siguiente tabla de condición indica que el proceso Pedidos solo se podrá llevar a cabo si existe el archivo temporal.txt, se dió la contraseña correcta y opcionalmente tendrá que tener prendida la impresora, de lo contrario, sacará los resultados a pantalla.

CONDICIÓN	PROCESO	RESTRICCIÓN	APUNTADOR
tempo.txt	Pedidos	requerida	2
impresora	Pedidos	opcional	4
clientes.dat	Actualiza	requerida	5
contraseña	Pedidos	requerida	nil
transacc.dat	Actualiza	opcional	nil

Tabla de resultados.

Es la lista tanto de las salidas como de resultados de las acciones descritas en el guión y se utilizará para mostrar lo que puede hacer un proceso una vez elegido. El objetivo principal de ésta tabla es el dar apoyo para el agregado de nuevas funciones al sistema en la etapa de mantenimiento.

Nombre del resultado	Programa que lo realiza

Ejemplo: Suponiendo el guión de la Figura 3.7, la tabla quedaría:

Nombre del resultado	Programa que lo realiza
Catálogo de enfermedades actualizado	Historia.exe
Archivo de médicos actualizado	Historia.exe
Receta	Historia.exe
Historial médico	Historia.exe
Historial administrativo	Historia.exe
Estadísticas de enfermedades	Historia.exe

Ahora, si se desea agregar una nueva función al sistema, por ejemplo: estadísticas de pagos; sólo es necesario agregar el renglón (Estadísticas de pago, Estadis.exe). Claro que el programa Estadis.exe debe realizarse, pero no tiene porque unirse al código de Historia.exe.

Tabla de datos.

Durante la etapa del análisis no se mencionaron muchos nombres de datos; más bien se trató de los reportes y consultas deseadas. Ya en el nivel de diseño es primordial detallar cada dato, campo o atributo que existirá en el sistema, así como sus atributos. El contenido de la tabla puede ser:

Nombre del dato	descripción	tipo	lon-gitud	deci-males	rango inicial	rango final	nombre del catálogo	valor nulo

A continuación se explica cada uno de las partes que conforman la descripción de la tabla de datos:

Nombre del dato.- Un mnemónico que represente el dato. Ejemplo: *paremp.* Puede ser el nombre de campo que se use para "parte empleada de la planta".
Descripción.- Aquí se trata de poner una leyenda breve de lo que representa el dato. Ejemplo: "forma de uso de la planta".
Tipo.- Se refiere al tipo de campo, ésto es: numérico, carácter, fecha, cuerda, conjunto,
Longitud.- Es el tamaño del campo.

Decimales.- Solo se usa si el campo es un número de punto flotante.
Rango Inicial y rango final.- Se usan cuando el campo puede variar dentro de un rango de valores.
Nombre del catálogo.- Los catálogos son también conocidos como archivos de tablas; son permanentes, contienen datos de referencia utilizados cuando se procesan transacciones, se actualizan archivos maestros o se producen salidas [SEN92]. Si se tiene un campo cuyos valores varían dentro de un conjunto finito se usa un catálogo para validarlo.
Valor nulo.- Valor que puede aceptarse en el campo en caso de que el usuario no sepa que poner, entonces el valor nulo será cero o espacios. En casos para los cuales el valor nulo no es permitido, el campo quedará vacío.

La mayoría de los datos en ésta tabla debió aparecer en el análisis en el momento de realizar el manual de operación, donde debieron quedar reflejadas las características generales de cada dato. En la etapa de diseño se hará una definición más miniciosa de cada dato de tal manera que en su misma definición quede reflejado el plan de prueba de datos (rangos permitidos o lista de valores).

Ejemplo: Si se tiene el sistema para la recepción de enfermos en un hospital que se muestra en la Figura 3.7, se tendría la siguiente tabla.

NOMCPO	DESCRIPCIÓN	TIPO	LON.	DEC	RANGO INICIAL	RANGO FINAL	CATÁ-LOGO	VALOR NULO
clave_enf	Identificación del enfermo	NUM[13]	6		1	99,999		
nombre	nombre del enfermo	ALFA	40					
fechanac	fecha de nacimiento	DATE	8					00/00/0000
tipo	tipo de sangre	NUM	2				CATTI	
rh	RH	NUM	1				CATRH	
enferm	Enfermedad	NUM	5				CATEN	
peso	Peso del paciente	NUM	5	2	3	180		0
saldototal	Costo del tratamiento	NUM	5	2				0
numpagos	Número de pagos	NUM	2	0	1	12		0

Tabla de usuarios.

Para saber quiénes pueden hacer uso de qué resultados o procesos se utiliza la tabla de usuarios (actores en el guión hecho en el análisis). Esta tabla define los elementos que permiten garantizar la seguridad de acceso al sistema y sus elementos son:

[13] Se sugieren claves numéricas para que el sistema las agregue de forma automática.

Nombre del operador	Resultado o proceso	Clase de operador

Por ejemplo, si se toma el guión de la Figura 3.7, se tienen los siguientes usuarios.

Nombre del operador	Resultado o proceso	Clase de operador
Laura Pérez Prieto	Actualizar datos	2
Adam Williams García	Estadísticas médicas	1

Tabla de archivos.

Juntas ésta tabla y la de datos, definen las estructuras de almacenamiento del sistema y ayudan a revisar si los archivos han sido alterados en su estructura en forma externa al mismo y facilitan las operaciones en que es necesario relacionar archivos. Su contenido es el siguiente:

Nombre del archivo	Nombre del dato	categoría del dato (llave, relación, participante)

Ejemplo: tomando nuevamente el guión de la Figura 3.7 y la tabla de datos descrita arriba tendríamos la siguiente fracción la tabla de archivos:

Nombre del archivo	Nombre del dato	Categoría del dato
Maestro_enfermos	clave_enfer	llave
Maestro_enfermos	nombre	participante
Maestro_enfermos	enferm	relación con Maestro_enfermedades

Tabla de procesos-archivos.

Se utiliza esta tabla para garantizar que al entrar al sistema existan todos los archivos necesarios. También es conveniente para realizar soporte de datos (guardar los datos en memoria secundaria como discos flexibles o cintas), ya que en esta tabla se indicará cuales archivos deben guardarse mediante el campo *tipo de archivo* (si el archivo es temporal, no valdrá la pena hacer un soporte).

Nombre del archivo	tipo de archivo	Nombre del proceso

3.5.1.2 Las operaciones del metasistema

El metasistema debe ser tratado como un sistema aparte con sus archivos (las tablas) y procesos propios, los cuales se listan a continuación:

Actualización.- Estará restringida al administrador y al analista-diseñador.

Lectura.- El metasistema permitirá la lectura de sus elementos para que el sistema pueda presentar opciones y resultados.

Realizar o no un sistema especial para el metasistema dependerá de que se le permitirá al usuario alterar las estructuras del sistema. En el caso de no permitírselo, se pueden omitir las operaciones de actualización para realizarlas desde cualquier herramienta con la que se logre accesar y actualizar las tablas del diccionario.

La tabla de usuarios es un caso especial del Diccionario de Datos, pues es la liga natural entre sistema y metasistema. Dado que ambas partes la utilizan, podrá ser actualizada por las operaciones del metasistema o del sistema.

3.5.2 Diseño del Sistema.

Para la definición del diseño se puede hacer uso de la metodología del agrado del diseñador o que sea prudente utilizar. Lo que se debe tener siempre en mente es que deben hacerse conexiones constantes al metasistema.

Se recomienda que se utilice el modelo Entidad-Relación, como se explica en la sección 3.7.

Como recomendación general, Coad y Yourdon [COA90a] proponen no usar la metodología orientada a objetos si se tienen menos de tres objetos. A este respecto el analista puede decidir utilizar orientación a objetos de todas formas dado que: tal vez cuente con toda la infraestructura necesaria para solo unir partes sus diferentes bibliotecas y que cuente además con el hardware suficiente para soportarlo.

Independientemente de la metodología escogida, se pueden identificar las funciones del sistema y las entradas y salidas de ellos utilizando los guiones del análisis como sigue:

Cada acción en una quinteta será una función del sistema.
Los resultados de las acciones serán las salidas.
Los actores y los accesorios son las entradas.

Ejemplo: Si se tiene el guión de la Figura 3.7, se tendría que parte del Diagrama de Flujo de Datos sería el que se muestra en la Figura 3.8:

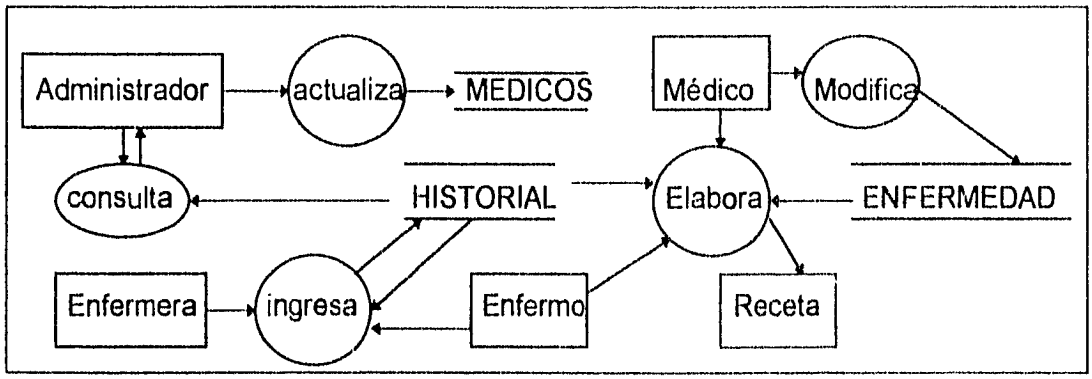


Figura 3.8 Fracción del Diagrama de Flujo de Datos correspondientes al Sistema de Historial Médico.

3.5.3 Relación del sistema con metasistema.

Como ya se había mencionado, el sistema será totalmente dependiente del metasistema. El sistema podrá realizar sus operaciones basándose en las diferentes partes del Diccionario de Datos como sigue:

Para la interfaz.- El sistema verificará el tipo de operador del que se trata y se tomarán los datos de la tabla de procesos (escenas) que le correspondan mostrándoselas al usuario; éste escogerá una y se verificará si corresponde a una operación o a un nuevo nivel de menúes, caso en el cual se volverá a desplegar.

Para la operación.- Una vez elegido el resultado que se quiere obtener, entonces se accesa la tabla de procesos, se verifica la existencia de los archivos que intervienen en él y que las condiciones de entrada sean las adecuadas, entonces se ejecuta.

Para los programas de actualización de los archivos maestros.

Mostrar la lista de descripciones de campos en una "pantalla enrollable" y permitir que el operador viaje por ellos hasta encontrar la deseada. La pantalla básica de actualización puede ser como se ve en la Figura 3.9. El objeto que tiene hacer la actualización de esta forma es evitar tener un tercer nivel de menú (Altas, Bajas, Cambios), que llega a aburrir al usuario y además permite tener más "a la mano" los datos de todo el archivo. Como se puede notar el hecho de que los nombres de datos salgan del Diccionario de Datos da flexibilidad al sistema, pues basta con agregar un campo en el diccionario para que, sin necesidad de reprogramar, aparezca como parte del archivo.

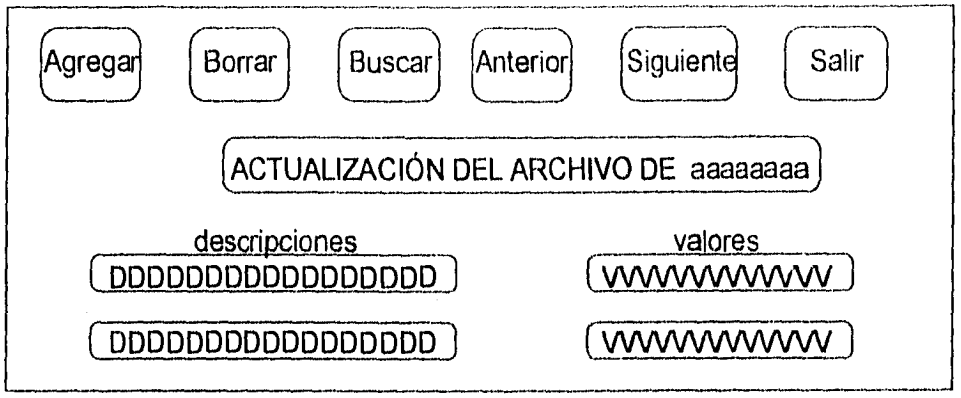


Figura 3.9 Pantalla para la actualización de archivos maestros

1. Si el campo elegido por el usuario está asociado a un catálogo, desplegar sus valores junto con la opción OTRO para que el usuario escoja el valor deseado.
2. Si se escoge un valor existente, se graba internamente la clave asociada a él (puede ser una clave numérica).
3. Si se escoge la opción OTRO, entonces se pide el nuevo valor del catálogo asociado y se actualiza este asignando una clave numérica en forma automática.
4. Si el campo no está asociado a un catálogo, entonces se debe consultar el diccionario para ver si tiene valores de rango inicial y final, para evitar la aceptación de algún valor que se salga de los límites.
5. Si no se tiene un catálogo o rango de valores asociado, entonces por lo menos debe cuidarse respetar el tipo del dato (numérico, carácter) y verificar si está o no permitido un valor nulo.

Para la consulta.- Mostrar la descripción del diccionario de resultados. Luego que el usuario escoja uno, se revisan los archivos que intervienen en ese resultado en el diccionario de relación Proceso-archivo. De ahí, se revisa cuáles son los campos *llave* y *relación* entre los archivos participantes y se pide al usuario indicar cuál de los campos se va a consultar; posteriormente se piden al usuario los valores, de la misma forma en que se hizo en la actualización.

3.6 Diseño del Sistema Conserva.

El sistema **Conserva** se desarrolla en Clipper [CLI90], por ello muchas de las tablas que conforman el diccionario no se utilizarán, ya que el lenguaje mismo nos da la información necesaria. Así pues, la tabla de datos no se necesitará y será sustituida por la instrucción AFIELDS() que será llamada en el momento en que se necesite saber los nombres de los campos.

3.6.1 El diseño del metasistema y el Diccionario de Datos de Conserva.

Las tablas que forman el Diccionario de Datos del sistema **Conserva** son:

DIRECTORIO.- que reune parte de la tabla de datos y de la de archivos.

Nombre del campo	categoría del campo	descripción del campo	página	número relativo	nombre del catálogo
10 caracteres	1 carácter	60 caracteres	3 caracteres	2 caracteres	8 caracteres

Nombre del Campo.- Se refiere a un mnemónico que servirá para el uso de los programas.
Categoría del Campo.- En este caso se trata de una letra que designa el papel del campo en el sistema y es como sigue:

E = archivo de Especies,
G =archivo Gacetero (observaciones),
A = ambos archivos, esto es campo de relación entre Especies y Gacetero.
C = nombre de un catálogo
H = campo obligatorio que servirá para la Historia de la captura (fecha, nombre capturista, folio)

Página.- La captura estará dividida en páginas lógicas (datos taxonómicos, datos de ambiente), así que se optó por asociar cada campo con un número de página.
Número relativo.- Se refiere a la posición relativa dentro de la página a la que pertenece el dato.
Nombre del catálogo.- En caso de que un dato tome valores fijos se asocia a un catálogo para apoyar a la consulta (ejemplo: nombres científicos, climas).

PÁGINAS.- Su función es la de manejar la interfaz de captura. En esta tabla se señala el orden en que se capturan los datos y los nombres de cada pantalla de datos.

Clave	Nombre	Archivo al que pertenece
3 numérico	40 caracteres	8 caracteres

Clave.- Debe coincidir con el campo página del directorio.
Nombre.- Descripción de la página, sirve como título en el momento de la captura y para que el operador diga cual parte quiere capturar.

Archivo al que pertenece.- Se agrega el nombre del archivo al que pertenece (Especies o Gacetero) y sirve para mostrar al usuario solo las páginas del archivo que desea usar.

USUARIOS.- Esta, como su nombre lo indica, sirve para saber que personas pueden hacer uso del sistema y los niveles de acceso que tienen a él.

Clave del usuario	Tipo de usuario	Comentario	Contraseña
10 caracteres	10 caracteres	40 caracteres	12 caracteres

Clave del usuario.- La clave de entrada al sistema
Tipo de usuario.- Es la forma de identificar el nivel de acceso a los datos y las operaciones.
Comentario.- Solo se pone para saber por qué está en el sistema cierto usuario
Contraseña.- Estará encriptada por un método especial y cuando sea tecleada no habrá eco en la pantalla y además cuando se despliegue esta tabla por otros medios, tampoco se entenderá lo que contiene.

3.7 El modelo Entidad-Relación dentro de la metodología propuesta.

Dado que el modelo Entidad-Relación (ER) proporciona elementos que permiten mostrar claramente la relación entre las entidades o componentes del mundo real, se usará entonces para aclarar más cómo, desde el análisis, se puede pasar paulatinamente al diseño. A continuación, se explica de qué manera se va a conformar el diagrama ER desde el análisis, mismo que durante el diseño puede sufrir modificaciones, aúnque mínimas.

Cada actor o papel dentro del guión será considerado una Entidad; cada acción que realiza el papel será una Relación y lo que produce tal acción será también una Entidad.

3.7.1 El modelo Entidad-Relación del Sistema Conserva y su relación con el metasistema.

Como se dijo anteriormente, los papeles, los resultados y los utensilios pasarán a ser entidades, mientras que las acciones desarrolladas por los papeles sus relaciones. En la Figura 3.10 se muestra el esquema E-R del sistema **Conserva**.

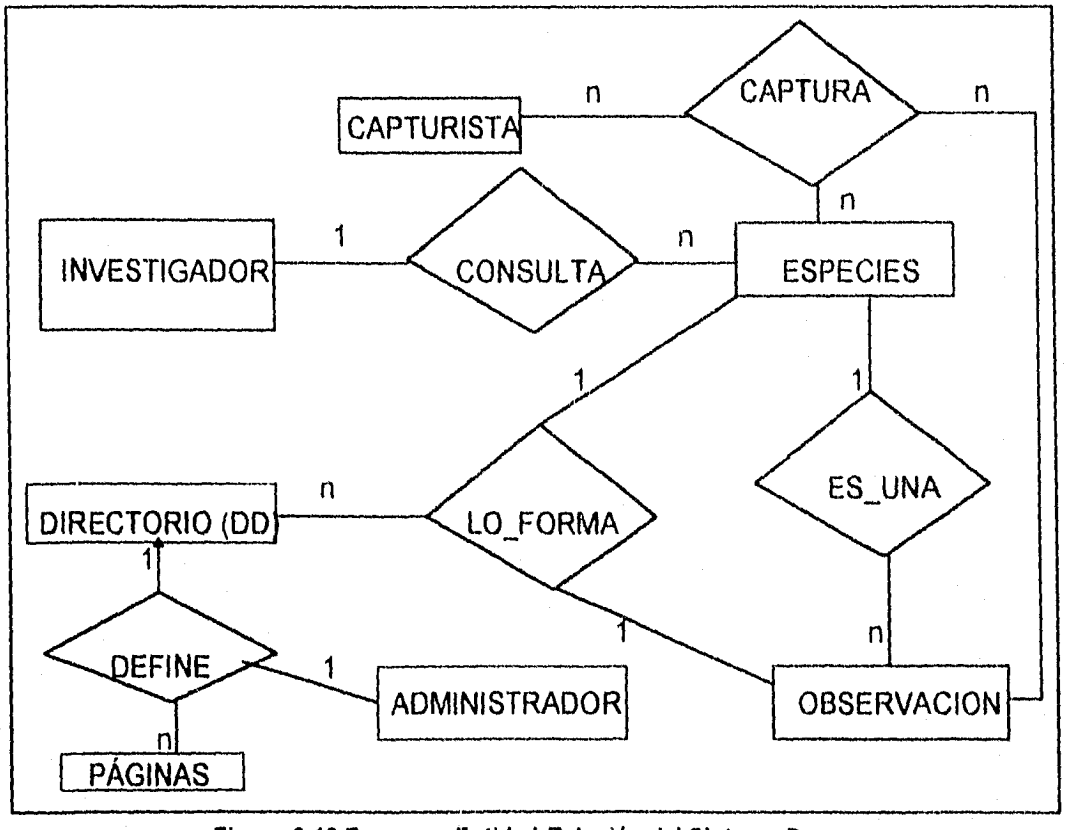


Figura 3.10 Esquema Entidad-Relación del Sistema Conserva

La forma en que las entidades se relacionan se marca sobre la raya. Ejemplos:

- varios capturistas puede capturar varias especies

- un investigador puede consultar varias especies.

3.8 La implantación.

En este punto se verá la forma en que se pueden implantar los sistemas en la computadora. No se tratará en detalle sobre algún modelo de programación específico, sino que se ejemplificará sobre dos de los más importantes por su frecuencia de uso: Orientación a Objetos y Programación Imperativa.

3.8.1 Implantación usando Orientación a Objetos.

Bajo este enfoque se puede considerar cada tabla que conforma el Diccionario de Datos (DD) como un objeto, cuyos atributos serían los elementos de la tabla y sus métodos serían al menos los mismos de la clase general llamada DD, a saber MODIFICAR y LEER.

Con respecto al sistema en sí, se formarían varios objetos que sirven de comunicación con el usuario. Por ejemplo: para la lista de operaciones del sistema se tendría la clase menú cuyos atributos serían nivel del menú, número de opciones a desplegar y lista de opciones; como métodos se tendrían llenar menú desde la tabla guardadada en un archivo, desplegar lista de opciones y leer la opción deseada. El código de este ejemplo en C++ se vería como se muestra en el listado de la Figura 3.11. El programa lee un archivo texto con renglones que contienen dos dígitos correspondientes a la escena y la quinteta y una cuerda con el nombre de la opción.

3.8.2 Implantación usando programación imperativa.

La forma en que se puede programar usando éste enfoque sería el uso de uno o más archivos que contengan el DD y al iniciarse la ejecución del sistema guardar en arreglos las diferentes tablas para su consulta. Así, por ejemplo se podrían tener la tabla de menúes (escenas y quintetas), la tabla para consulta, actualización y reportes (de los datos).

Para aclarar un poco la forma de programar usando la metodología propuesta, se incluye el listado del programa en Clipper de formación y despliegue de un menú de opciones (ver Figura 3.12), dicho programa lee un archivo cuya descripción es la siguente:

NOMBRE DEL CAMPO	TIPO	LONG
NUMESC (número de escena)	numérico	1
NUMQUI (número de quinteta)	numérico	1
NOMOPC (nombre de la opción)	carácter	10

```
#include <conio.h>
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <string.h>
class menu
{
  struct opciones{int escena,quinteta;
          char opcion[20];} menar[20];
// métodos
 public:
 menu();
 int llena();
 int escoge(int limite);
 int despliega(int nivel, int mmax);
};
// función constructor de menú
menu::menu()
{
 int i,j;
 char *blancos="                    ";
 for (i=0; i <= 20; i++);
 {
   menar[i].escena=0;
   menar[i].quinteta=0;
   strncpy(menar[i].opcion,blancos,20);
   menar[i].opcion[20]='\0';
 }
}
// obtiene del archivo tipo texto menu.dat las
//  opciones de todos los menúes
int menu::llena()
{
 int i;
 char buf,cuerda[23];
 char *blancos="                    ";
 clrscr();
 i=0;
 strncpy(cuerda,blancos,22);
 ifstream archivo("menu.dat",ios::in);
 while (archivo.get(buf))
 {
 if (buf==10) {  archivo.get(buf);}
  menar[i].escena=buf;
  archivo.get(buf);
  menar[i].quinteta =buf;
  menar[i].escena-=48;
  menar[i].quinteta-=48;
  archivo.get(cuerda,22);
  strncpy(menar[i].opcion,cuerda,23);
  i++;
 }
 return(i);
}
```

```
// Método encargado de mostrar el menú, según
// el nivel que le sea indicado  (0=principal).
int menu::despliega(int nivel,int mmax)
{
 int i,cuantos;
 char opc;
 clrscr();
 cuantos=0;
 for (i=0; i <= mmax; i++)
  if (nivel==0)
  {
  if ((menar[i].escena != 0) && (menar[i].quinteta
                                          == 0))
   { printf("%d .- %s \r \n ", menar[i].escena,
         menar[i].opcion);
   cuantos++;
   } }
  else
  {
  if ((menar[i].escena == nivel) &&
                        (menar[i].quinteta != 0))
   { printf("%d .- %s\r\n",menar[i].quinteta,
         menar[i].opcion);
     cuantos++;
  } }
 return(cuantos);
}
// Encargada de obtener la opción deseada
int menu::escoge(int kua)
{
  int opcion;
  opcion = kua+1;
  while ((opcion > kua) || (opcion < 0))
  {
   cout << "Oprima un  número";
   cout << "de la lista o cero para salir\n";
   cin >> opcion;
  }
  return(opcion);
}
void main()
{
  menu principal;
  int opc,opcsub,niv,kua,zz,mm;
  niv=0;
  mm=principal.llena();
  kua=principal.despliega(niv,mm);
  opc=0;
  opc=principal.escoge(kua);
  if (opc!=0)
  { opcsub=0;
    kua=principal.despliega(opc,mm);
    opcsub=principal.escoge(kua);
  } }
```

Figura 3.11 Código en C++ correspondiente al llenado y desplegado de menúes.

```
Programa en Clipper que permite construir un menú desde un archivo.
Se supondrá que el archivo está ordenado físicamente por NUMESC y NUMQUI.
Se abre el archivo que contiene las escenas y quintetas.
USE ESCENAS
De esta forma se obtienen los elementos del menú principal.
COUNT FOR NUMQUI=0 TO NE
GO TOP
Se declara un arreglo llamado MENPRIN con el número de opciones del menú.
DECLARE MENPRIN[NE]
I=1
Se revisa todo el archivo hasta que ya no existan más opciones del primer nivel
DO WHILE (.NOT. EOF()) .AND. (I<=NE)
 MENPRIN[I]=NOMOPC
 COUNT FOR NUMESC=I TO NQ
 NQ=NQ-1
Ahora se obtienen el número de subopciones de la opción principal número I
 SUBMENU="SUBMENU"+STR(I,1)
Se declara un arreglo con el número de opciones del menú. Note que el
nombre del arreglo está guardado en una variable llamada SUBMENÚ.
 DECLARE &SUBMENU[NQ]
Se posiciona en la primera subopción de la opción i-ésima de nivel uno.
 LOCATE FOR NUMESC=I .AND. NUMQUI=1
 J=1
Recorre el archivo secuencialmente hasta que la opción de primer nivel cambie
 DO WHILE (I=NUMESC).AND.(.NOT. EOF())
  &SUBMENU[J]=NOMOPC
  J=J+1
  SKIP
 ENDDO
 I=NUMESC
ENDDO
Se despliega el menú principal y se escoge una opción con la función ACHOICE()
CLEAR
@ 0,0 TO NE,11
I=ACHOICE(1,1,10,10,MENPRIN)
Si hubo una elección diferente de salir (tecla ESC) se despliega el submenú en el renglón i-
ésimo.
IF I#0
 @ I,12 TO 21,21
 J=ACHOICE(I+1,13,20,20,&SUBMENU)
ENDIF
RETURN
```

Figura 3.12 Programa en Clipper 5.0 para desplegado de menúes.

3.9 La etapa de prueba en la metodología.

En este punto, que normalmente requiere el cuarenta por ciento del tiempo de desarrollo de software, la metodología propuesta ya lleva adelantada la planeación de varias pruebas del sistema, a saber:

1. Pruebas de funcionalidad.- En el diccionario de procesos realizado en el análisis se señalaron las pruebas a realizar y el tiempo esperado para las metas del sistema.
2. Pruebas de conjunto de datos .- En el diseño se plantearon los datos con que se debe probar el sistema al incluirlos en la tabla de datos (rangos y catálogos).

Todavía es necesario realizar pruebas de caja blanca, tales como caminos básicos, y de caja negra, como serían los de interfaz [PRE93]. Pero, dado que los sistemas hechos con esta metodología reutilizan el código de uso de diccionarios, gran parte del sistema ya habrá sido probada.

Cabe hacer notar que no se propone ningún tipo de prueba de volumen o de paralelismo, pero es obvio que deberán llevarse a cabo.

3.10 Mantenimiento.

Tal vez donde se noten más claramente las ventajas de esta metodología sea en esta etapa del ciclo de vida, que además es la más larga y tediosa (según James Senn [SEN92] se lleva del 60 al 90% del desarrollo del software), pero inevitable.

Las acciones que será necesario realizar en esta etapa cambiarán de acuerdo al tipo de mantenimiento que se necesite. Los tipos de mantenimiento que se manejarán [SEN92] son:

Correctivo.- Aquel en el que se realizan arreglos de emergencia y que tienen una frecuencia relativa del 20%.
Adaptativo.- Adaptación al cambio de datos y archivos y al software y hardware del sistema, cuya frecuencia relativa es del 20%.
Perfectivo.- Mejoras para el usuario. Perfeccionar la documentación, nueva codificación para lograr mayor eficiencia de cómputo. La frecuencia relativa de este tipo de mantenimiento es del 60%.

A continuación se inscriben algunas posibilidades en la Tabla 3.3. Las necesidades que se anotan pueden parecer obvias, pero resultan las más frecuentes y que ocasionan bastantes transtornos.

Tabla 3.3 Actividades de mantenimiento según la necesidad del usuario

NECESIDAD DEL USUARIO	TIPO DE MANTENIMIENTO	ACCIÓN CORRECTIVA
Cambiar nombres a listados	Perfectivo	Modificar la tabla de datos y de resultados
Agregar nuevos datos a los listados o consultas.	Adaptativo	Modificar la tabla de datos.
Agregar una nueva función al sistema	Perfectivo	Modificar tablas de procesos y archivos y agregar el código de la nueva función.

Debe notarse que aquellas actividades de mantenimiento denominadas **correctivas** deben realizarse como se hace tradicionalmente, esto es, modificando el segmento de programa que contiene el error, anotando los cambios hechos en el programa fuente y realizando todas las pruebas de corrida planeadas para el sistema.

Cuando ya se tiene un tiempo utilizando esta metodología, resulta poco frecuente el mantenimiento correctivo, ya que todas las operaciones del metasistema son reutilizables. Además, muchas de las operaciones del sistema resultan comunes en casi todos los sistemas de procesamiento de datos (captura, actualización), por lo que su código también se puede reutilizar.

4. PRUEBAS DE EFECTIVIDAD DE LA METODOLOGÍA PROPUESTA.

La aplicación de un conjunto de estándares para el desarrollo de productos de software debe, en general, reportar beneficios y éstos ser mayores que los posibles costos. Pero, ¿cómo saber que tal fenómeno se cumplirá? Existen varias formas de probarlo y en el trabajo actual se han escogido tres:

1. Mostrar el comportamiento del desarrollo de los productos de software en los que de alguna manera se fue introduciendo la metodología en el pasado.
2. Utilizar de manera ordenada la metodología propuesta con un grupo de desarrolladores de software y observar el comportamiento de ésta.
3. Proponer una serie de acciones que podrán ser aplicados a futuro para comprobar, con el tiempo, la efectividad de la metodología.

Para lograr lo anterior se muestra en la sección 4.1 el comportamiento histórico de los sistemas, una serie de tablas que muestran aspectos relevantes de éstos y las estadísticas que les fueron aplicadas. En la sección 4.2 se mostrará el experimento de usabilidad que se realizó y los resultados que de éste se obtuvieron. Por último en la sección 4.3 se propone brevemente una metodología de valor estadístico a futuro.

4.1 Comportamiento histórico.

A lo largo de su experiencia profesional, la autora ha participado en el desarrollo de un buen número de productos de software. En estos productos se emplearon diversos métodos, según se usaba en los lugares donde se desarrolló, según las tendencias del momento y según la experiencia de la autora. En muchos de estos desarrollos se fueron aplicando diversos elementos, mismos que ahora se proponen para formar la metodología, por lo cual conviene hacer una revisión e incluso una evaluación de sus bondades.

4.1.1 Conjunto de sistemas de muestra.

En la Tabla 4.1 se muestra el conjunto de sistemas al que nos referiremos en los análisis siguientes. Se incluyen los elementos más importantes para identificarlos y se incluye un mnemónico para relacionarlos con otras tablas. En el apéndice A se incluyen datos adicionales que no se emplearon en este capítulo.

Tabla 4.1 Conjunto de sistemas observados.

Mnemónico	Año y Dependencia[14]	Nombre del proyecto	Técnicas usadas [15]	Observaciones
1.- EVALPRIM	1974, CPE-SEP	Evaluación general de las primarias del D.F. y área Metropolitana	FD	Se emitían hojas para las respuesta de lector óptico, se enviaban a las escuelas y al regresar se empataban a través de un folio con las respuestas. Posteriormente se calificaban y se sacaban resultados estadísticos.
2.- ADMISECU	1974, CPE-SEP	Calificación y evaluación de los exámenes de admisión de las Escuelas secundarias Federales de toda la república.	FD	El volumen de datos de los sistemas era de cientos de miles de registros y los tiempos de proceso iban desde noches enteras para la calificación, hasta dos días para la impresión de resultados.
3.- CALITELE	1975, CPE-SEP	Calificación y evaluación de Telesecundarias	FD	En este se llevaba además las calificaciones por cada materia y al finalizar el tercer año se emitían certificados
4.- ADMINORM	1975 CPE-SEP	Admisión a la Escuela Normal	FD	Se emitían hojas de lector óptico para la aplicación del examen, se calificaba y se emitían resultados.
5.- ABIERTOS	1975-76 CPE-SEP	Proyectos piloto de primaria y secundaria abiertas	FD	Se emitían hojas de lector óptico para la aplicación del examen, se calificaba y se emitían resultados y certificados.
6.- PESCA	1976. IIB, UNAM	Simulación de la pesca del camarón	Simulación	Se empezaba el proceso de simulación desde que eran larvas en los esteros hasta que salían a alta mar
7.- TELEX	1976-1980. CIECE-SCT	Facturación del Servicio de Telex.	FD	Se recogía la información de las llamadas hechas a través de las repetidoras internacionales RCA, ITT y otras; se reunían y se emitían facturas a las diversas cuentas particulares y se hacía la relación de lo que se debía pagar a las compañías.

[14] Siglas empleadas:
CPE-SEP: Centro de Procesamiento y Evaluacion Dr. Arturo Rosenblueth de la SEP. IIB UNAM: Instituto de Investigaciones Biomédicas de la UNAM. CIESE SCT: Centro de Investigación Estadistica y Computacion Electronica. SCT. INIREB: Instituto Nacional Sobre Recursos Bióticos (ahora Instituto de Ecologia). UV: Universidad Veracruzana. UABC: Universidad Autonoma de Baja California.
[15] Metodos: FD: Flujo de Datos. DD: Diccionario de Datos. ER: Entidad Relación. PE: Programación estructurada

8.- INMUEBLES	1978 CIECE-SCT	Control de Bienes Inmuebles de la SCT	FD + DD + PE	Inventario de los Bienes Inmuebles de la SCT, que van desde una oficina hasta una draga. Su estado puede ser rentado, comprado o prestado y a su vez puede ser en forma total o parcial (una torre de Telecomunicaciones puede estar en un terreno particular). Los datos o características podían variar según el tipo de inmueble.
9.- ENCUESTAS	1978. Colegio de México	Captura de encuestas socioeconómicas	DD	Se hizo un programa general (en Fortran) que permitiera al usuario definir sus campos.
10.- PAGODIFE	1980. Dirección de Informática U.V.	Pago de diferencias al personal académico de la U.V.	FD	Sistema que tomaba la información de nóminas anteriores para el cálculo de algunas diferencias que se le adeudaban al personal académico de la universidad. Fue un sistema temporal
11.- CONTPERS	1981. Dirección de Informática U.V.	Control de personal	DD+FD	Se pretendía llevar un control del personal de la U.V. No se implementó por falta de interés del usuario
12.- RAYARECO	1986. Finca Cafetalera	Pago de raya a recolectores	E-R + FD	Se captura la información sobre los kilos de café cereza que corta cada recolector, los prestamos y anticipos que se le hacen, los kilos de café que se envían a los beneficios cafetaleros y se emiten reportes y etiquetas para los sobres de la paga a los recolectores.
13.- APLIRECU	1986. Finca Cafetalera	Aplicación de recursos	E-R + FD	Se va acumulando información sobre los gastos de manutención y operación de la finca cafetalera. El sistema ha sido adaptado para otros negocios del mismo propietario.
14.- CONTINVE	1986. Finca Cafetalera	Control de Inventario	E-R + FD	Se captura todos los bienes del propietario incluidos otros que no son los de la finca y se emiten reportes.
15.-CONTABILID	1988. Beneficio cafetalero.	Contabilidad	E-R + FD	Sistema que lleva la contabilidad basada en pólizas de un beneficio de café.
16.- CI/SIG	1987. INIREB con apoyo de Conservation International.	Sistema de Información Geográfica para computadoras de bajos recursos	E-R + FD + DD	El sistema corre actualmente en diversos centros de Conservación de América Latina y contiene modelos para la toma de decisiones en Ecología.
17.- CONSERVA	1988. Conservation International	CONSERVA. Sistema de apoyo a la conservación de especies	E-R + FD + DD	Sirve como un apoyo a biólogos que se dedican a la captura de datos para la conservación. Les permite definir y modificar sus estructuras de una manera restringida y definir sus resultados.

18.- AVES	1991. Conservation International.	Aves del Neotrópico.	E-R + FD + DD	Permite la consulta de más de 4000 aves mediante diversos atributos en equipos pequeños y se distribuirá junto con un libro sobre aves.
19.- ECOMED	1991. Fac. de Ciencias. UABC	Ecosistemas Mediterraneos	DD	Permite la consulta mediante varios atributos de los recursos naturales de la región mediterranea de Baja California
20.- ESFIVE	1992. Fac. de Ciencias. UABC.	Estudio Fitosociológico de la Vegetación	FD	Después de la captura de los datos relevantes a la vegetación de una zona, se realiza un estudio aplicando técnicas estádisticas que apoyan a las decisiones sobre Ecología.
21.- SIISEV	1993. Instituto de Salud Pública. U.V.	Sistema Integral Sobre Salud en el Estado de Veracruz	E-R + FD + DD	Sistema de apoyo a estudiantes e investigadores del área de la Salud para la creación de sus propias bases de datos y de la presentación de la cartografía asociada a los datos escogidos o creados por ellos.

En la Tabla 4.2 se muestran algunos datos cuantitativos en relación a los diferentes sistemas desarrollados. En los primeros once son estimaciones, ya que no se cuenta con registros de apoyo.

Tabla 4.2 Tamaño de los Sistemas

Mnemónico	Número aproximado de lineas de código	Número de archivos de datos	Número de transacciones por unidad de tiempo	Número de reportes distintos	Niveles de menúes
1.- EVALPRIM	2000	4	400,000 / año	5	0
2.- ADMISECU	2000	5	1'000,000 / año	5	0
3.- CALITELE	2000	4	500,000 / año	7	0
4.- ADMINORM	2000	4	100,000 / año	6	0
5.- ABIERTOS	2000	5	5000 / año	7	0
6.- PESCA	200	0		1	0
7.- TELEX	1000	4	100,000 / mes	5	0
8.- INMUEBLES	1000	4		variable	0
9.- ENCUESTAS	200	variable	?	5	0
10.- PAGODIFE	1500	2	2500/semana	2	2
11.- CONTPERS	5000	2	5000/año	variable	5
12.- RAYARECO	4800	22	1000 / semana	9	4
13.- APLIRECU	2500	13	100 / mes	6	3
14.- CONTINVE	5000	9	?	9	4
15.-CONTABILID	7000	?	?	5	5
16.- CI/SIG	?	variable	variable	mapas / tablas	5
17.- CONSERVA	5300	variable	variable	variable	5
18.- AVES	1700	39	no hay	variable	3
19.- ECOMED	1000	20	no hay	no hay	4
20.- ESFIVE	2500	10	?	?	5
21.- SIISEV	?	variable	variable	variable	5

En la tabla 4.3 se muestran algunos otros datos de los sistemas mencionados, que describen el esfuerzo empleado en su desarrollo.

Tabla 4.3 Esfuerzo en la elaboración del proyecto

Mnemónico	Tipo de proyecto [16]	Número de analistas	Número de programadores	Tiempo análisis[17]	Tiempo de construcción	Tiempo en que empezó a operar
1.- EVALPRIM	1	2	4	2	3	5
2.- ADMISECU	1	2	4	2	3	5
3.- CALITELE	1	2	4	2	3	5
4.- ADMINORM	1	2	4	2	3	5
5.- ABIERTOS	1	2	4	2	3	5
6.- PESCA	2	2	1	1	1	1
7.- TELEX	1	2	2	2	3	5
8.- INMUEBLES	1	1	1	2	4	6
9.- ENCUESTAS	2	1	1	1	1	desconocido
10.- PAGODIFE	1	1	2	0.25	0.25	0.25
11.- CONTPERS	1	1	1	2	2	desconocido
12.- RAYARECO	3	2	2	1	3	4
13.- APLIRECU	3	2	2	1	3	4
14.- CONTINVE	3	2	2	1	3	4
15.-CONTABILID	3	2	2	1	2	desconocido
16.- CI/SIG	2	5	8	6	18	24
17.- CONSERVA	2	2	2	3	6	12
18.- AVES	2	2	2	2	6	desconocido
19.- ECOMED	2	2	2	1	1	1
20.- ESFIVE	2	1	5	1	6	desconocido
21.- SIISEV	2	4	4	2	12	6

4.1.2 Clasificación de errores, fallas y problemas.

El propósito de observar los sistemas antes descritos es analizar el posible impacto que tuvo sobre ellos el uso de determinadas metodologías. Para tener elementos de comparación se describirán los errores que se presentaron en ellos, para luego relacionarlos con las metodologías.

Tratar de clasificar los errores que pueda haber en un sistema no es tarea fácil, comenzando por el concepto mismo de "error". Se buscó en varios textos de Ingeniería de Software y se encontró que el que más completo era el de Shooman [SHO87]. Por otro lado, el texto sobre Ingeniería de Software de Pressman [PRE93] menciona que: "En el contexto de cualquier disquisición sobre calidad y fiabilidad del software, fallo es cualquier falta de concordancia con los requisitos del software".

[16] Tipo de proyecto 1: institucional; 2: investigación; 3: particular; 4: educativo.
[17] El tiempo fue puesto en meses.

Dado que ninguno de los libros consultados tuvo las definiciones buscadas, se decidió hacer la siguientes definiciones y clasificaciones, que son aplicables a Sistemas de procesamiento de datos.

Error.- Todo aquello que un sistema de como resultado incorrecto.

- Tipos de errores:

1. - Inconsistencia en los almacenes magnéticos de datos.
2. - Resultados incorrectos.
3. - Velocidad de respuesta menor a la especificada en los requisitos.

Falla.- Es lo que provoca el error.

- Tipos de fallas:

1. - La interfaz del usuario. El diálogo (inconsistente e ininteligible), la entrada de datos (sin validar), la presentación de resultados (no entendibles o insuficientes).
2. - La interfaz con otros paquetes. Formato, capacidad de RAM (no se puede tener ambas aplicaciones abiertas en paralelo).
3. - Acceso concurrente no controlado a los datos.
4. - Capacidad de memoria.
5. - Caídas involuntarias (por parte de la aplicación) del sistema de cómputo.
6. - La configuración del hardware no es la esperada o tiene pequeñas variaciones.

Problema.- Lo que origina las fallas, que en algunas ocasiones son irremediables a corto plazo.

- Tipos de problemas:

1. - Falta de análisis o indefinición por parte del usuario de sus requisitos.
2. - Cambio constante de los requisitos o petición de nuevos resultados.
3. - Espacio reducido en disco y en memoria.
4. - Tiempo reducido para la presentación de resultados.
5. - Herencia de software viejo (generalmente mal hecho) para adaptar o conectar.
6. - Falta de capacitación.
7. - Falta de manuales.

En la tabla 4.4 se muestra un resumen de tipos de errores, fallas y problemas observados en los sistemas bajo estudio y un comentario sobre las consecuencias negativas que originaron.

Tabla 4.4 Impacto producido por los errores, fallas y problemas.

Mnemónico	Errores	Fallas	Problemas	Impacto
1.- EVALPRIM	1,2,3	4,5	1,2,3,4,5,6	Retraso en la entrega de resultados
2.- ADMISECU	1,2,3	4,5	1,2,3,4,5,6	Retraso en la entrega de resultados
3.- CALITELE	1,2,3	4,5	1,2,3,4,5,6	Retraso en la entrega de resultados
4.- ADMINORM	1,2,3	4,5	1,2,3,4,5,6	Retraso en la entrega de resultados
5.- ABIERTOS	1,2,3	4,5	1,2,3,4,5,6	Retraso en la entrega de resultados
6.- PESCA	2		1,2,6	Proyecto con resultados incompletos
7.- TELEX	2		1,2	Retraso en la entrega de resultados
8.- INMUEBLES			1,2	
9.- ENCUESTAS			1,2	Terminación del proyecto con resultados no probados
10.- PAGODIFE	1,2	1,4,5	1,2,3,6	Desvelo para la entrega de los resultados
11.- CONTPERS			1,2	Terminación del proyecto con resultados no probados
12.- RAYARECO	1,2,3	4,5	1,2,5,7	Modificaciones de emergencia
13.- APLIRECU	1,2,3	4,5	1,2,5,7	Modificaciones de emergencia
14.- CONTINVE	1,2,3	4,5	1,2,5,7	Modificaciones de emergencia
15.-CONTABILID	3		1,2	Modificaciones de emergencia
16.- CI/SIG	3		1,2,3,5	Retraso en la liberación del sistema
17.- CONSERVA	3	1,6	1,2,3	Un poco de desconcierto por parte del usuario
18.- AVES			1,2,3	Retraso en la liberación del producto
19.- ECOMED				
20.- ESFIVE			1,6	Terminación del proyecto con resultados no probados
21.- SIISEV			1,2,6,7	Retraso en la liberación del producto

Tomando los totales de tipos de error, de tipos de fallas y de tipos de problemas identificados en los sistemas bajo observación, se obtuvieron los resultados que se muestran en los histogramas de las Figuras 4.1 a 4.3. En ellas, una barra representan el número de sistemas que presentaron un cierto número de tipos diferentes de errores, fallas o problemas. La barra correspondiente al cero acumula los sistemas que no presentaron ninguno de los tipos de error, falla o problema descritos en este trabajo.

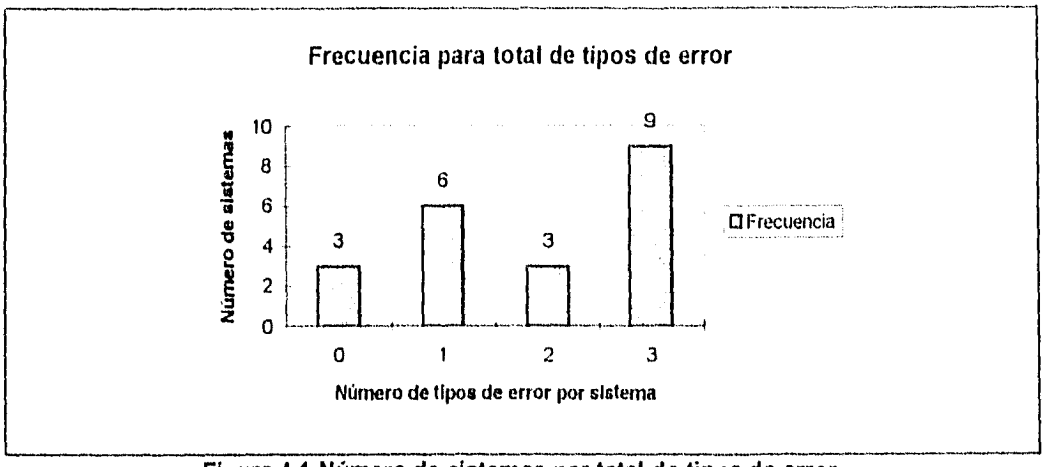


Figura 4.1 Número de sistemas por total de tipos de error.

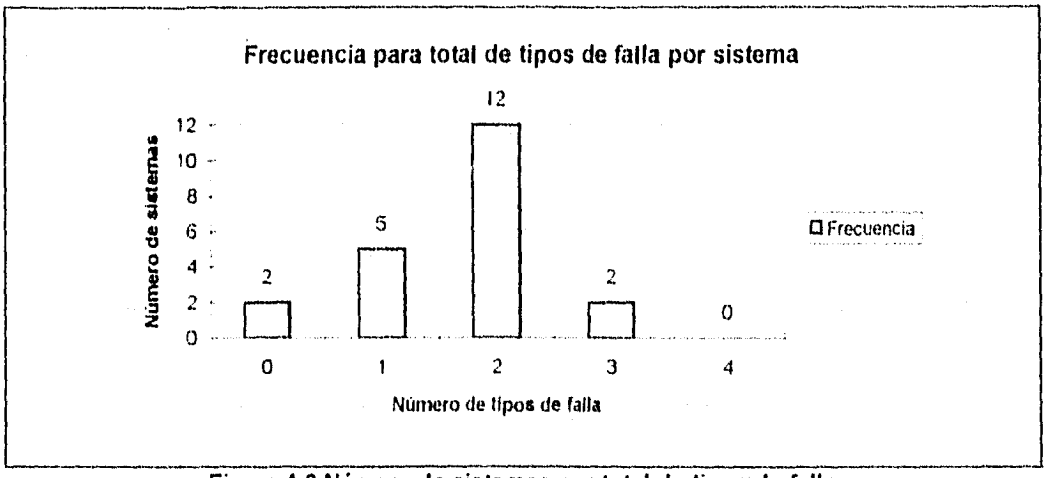


Figura 4.2 Número de sistemas por total de tipos de falla.

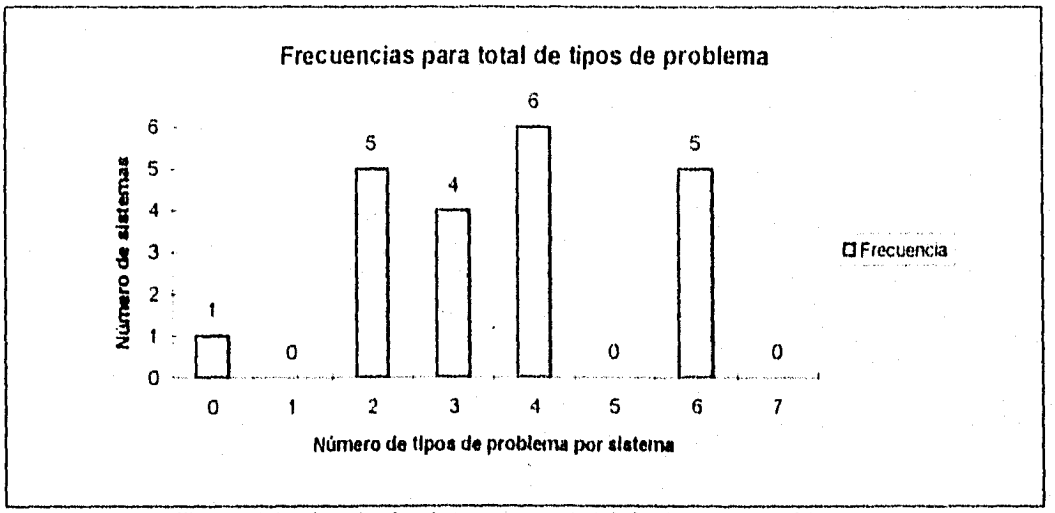


Figura 4.3 Número de sistemas por total de tipos de problemas.

4.1.3 Análisis estadístico.

Los datos contenidos en las diversas tablas (incluyendo las del apéndice A) se cargaron en una hoja de cálculo, convertidos todos a números, agrupados en cuatro grupos:

a) Datos descriptivos del ambiente de los proyectos y características de herramientas; casi son puros datos cualitativos, aún al volverlos numéricos.

b) Datos cuantitativos referentes al desarrollo del proyecto en cuestión.

c) Elementos importantes que caracterizan los avances metodológicos empleados; estos se manejaron como variables binarias.

d) Variables de resultado, que contienen básicamente los errores, fallas y problemas observados.

Con el fin de localizar posibles variables relacionadas, se obtuvo una tabla de correlación de todas las variables contra todas, aún las cualitativas. En la tabla 4.5 se muestra parte de estos resultados, donde se pueden observar algunos valores interesantes. Para eliminar algunos problemas en el cálculo de las correlaciones para las variables de errores, fallas y problemas (binarias) se empleó en su lugar el número de tipos diferentes de error, de falla y de problema detectados en cada sistema. Las columnas de la tabla corresponden a los renglones con el mismo índice.

Tabla 4.5 Coeficientes de correlación entre variables.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1. líneas de código	1.00													
2. número archivos	0.11	1.00												
3. niveles menúes	0.53	0.48	1.00											
4. analistas	0.75	0.21	0.30	1.00										
5. programadores	0.67	0.00	0.12	0.67	1.00									
6. tiempo análisis	0.75	0.08	0.15	0.71	0.67	1.00								
7. tiempo de construcción	0.76	0.25	0.44	0.84	0.73	0.82	1.00							
8. total de tipos error	0.05	-0.30	-0.21	0.00	0.36	-0.17	-0.10	1.00						
9. total de tipos falla	0.21	-0.21	0.21	-0.20	0.10	-0.13	-0.08	0.55	1.00					
10. total de tipos problema	0.20	-0.25	-0.46	0.25	0.47	0.21	0.09	0.74	0.42	1.00				
11. programación estructurada	0.31	0.53	0.85	0.11	-0.09	0.02	0.32	-0.36	0.17	-0.54	1.00			
12. Flujo de Datos	0.35	0.32	0.35	0.18	0.35	0.22	0.25	0.25	0.49	0.22	0.41	1.00		
13. Diccionario de Datos	0.17	0.30	0.36	0.21	-0.09	0.43	0.42	-0.77	-0.29	-0.53	0.41	-0.08	1.00	
14. entidad-relación	0.41	0.73	0.83	0.41	0.16	0.11	0.46	-0.13	-0.08	-0.36	0.75	0.31	0.23	1.00

A partir de la tabla de correlaciones se realizaron las observaciones siguientes:

a) Es muy marcado el descenso en los totales de tipos de error, fallas y problemas al usar Diccionario de Datos.
b) también se nota descenso con el uso del modelo Entidad-Relación, aúnque menor.
c) La programación estructurada ofrece resultados mixtos, indicando que disminuyen el número de tipos de errores y el de problemas, mientras que aumenta el de fallas.
e) Para el uso de Flujo de Datos francamente se asocia con un aumento en los totales de errores, fallas y problemas, aúnque la correlación es fuerte unicamente para los tipos de falla.
f) Los totales de problemas no parecen tener mucha correlación con las variables que no son resultados o tipo de metodología usada, salvo por algunas variables cualitativas, en las cuales puede deberse a la escala usada.
g) Para otras variables, como líneas de código, la mayoría de los resultados corresponden a lo que se esperarían intuitivamente.

En la Figura 4.4 se muestra un diagrama con la relación entre algunas variables, mostrando sus coeficientes de correlación.

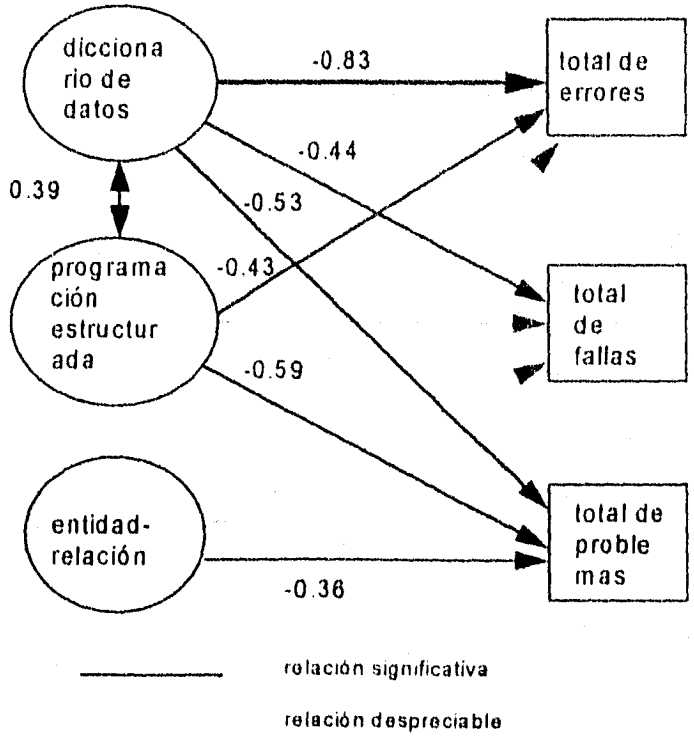
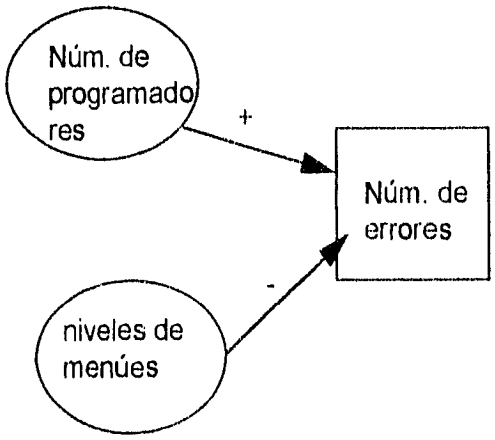


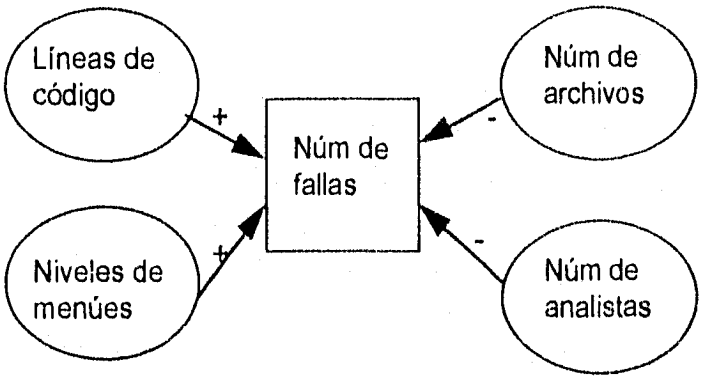
Figura 4.4 Interacción entre algunas variables.

Como las variables correspondientes al uso de metodologías corresponden a variables controlables, mientras que los errores, fallas y problemas son resultados observados, resulta razonable aceptar que la relación indicada en la Figura 4.4 indican causalidad. Por tanto, se sugiere la conveniencia de emplear DD para reducir la ocurrencia de errores y tambien de fallas y problemas. En forma similar se pueden revisar otras relaciones.

En la Figura 4.5 se muestran otras relaciones menos importantes.



a) Otras relaciones para Errores.



b) Relaciones para Fallas

+ correlación positiva - correlación negativa

Figura 4.5 Algunas relaciones no muy fuertes.

Realizando otros diagramas como éstos, se observa que hay pocas relaciones importantes que puedan atribuirse a una relación causa-efecto. Más bien se nota que las variables correlacionadas corresponden a lo esperado como, por ejemplo:

al crecer un proyecto requiere más analistas, más programadores, contiene más líneas de código.

En vista de la relación mostrada por la correlación entre el uso de ciertas metodologías y la ocurrencia de errores, fallas y problemas, conviene visualizar mejor la relación entre ambos grupos de variables. Así, se muestra una serie de histogramas (Figuras 4.6 a 4.17). Los histogramas se presentan en cuatro grupos de tres, correspondientes a las cuatro metodologías observadas y a los totales de tipos de error, falla y problema para cada caso. Cada uno de ellos compara los resultados de haber empleado o no una de las metodologías. Las barras representan el número de sistemas por total de tipos de error, falla o problema observado. Así, a manera de ejemplo, la figura 4-6 muestra que hubo 9 sistemas (69%) que presentaron tres tipos de error cuando no se empleó la metodología de Diccionario de Datos, mientras que no hubo sistemas con tres tipos de error cuando sí se empleó.

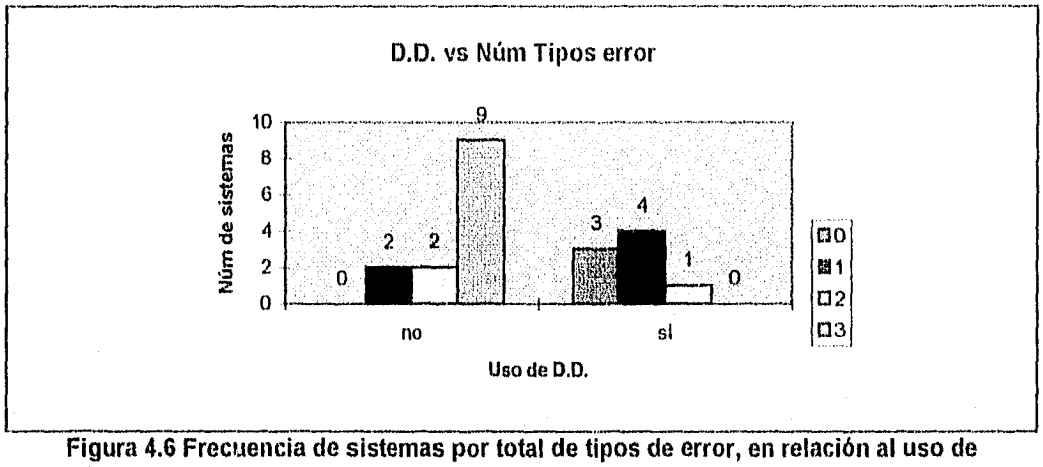


Figura 4.6 Frecuencia de sistemas por total de tipos de error, en relación al uso de Diccionario de Datos.

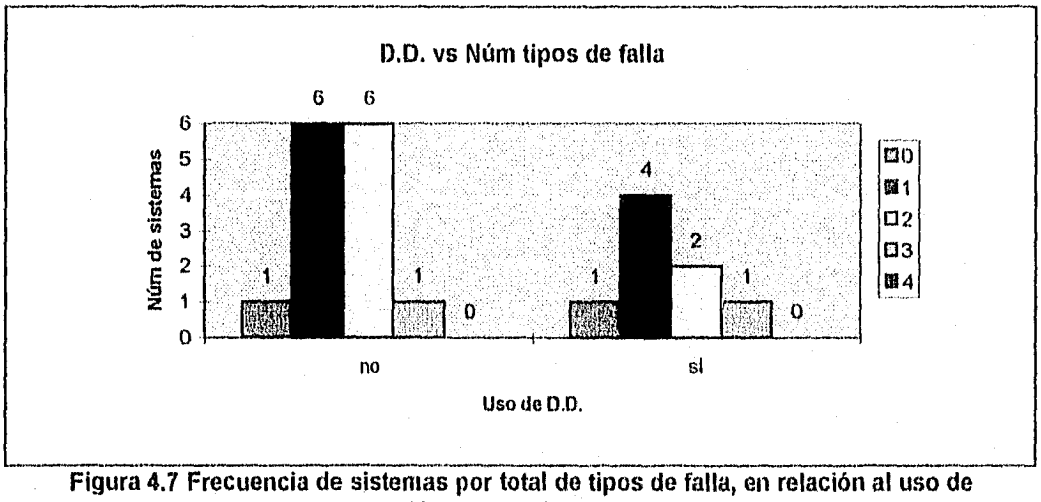


Figura 4.7 Frecuencia de sistemas por total de tipos de falla, en relación al uso de Diccionario de Datos.

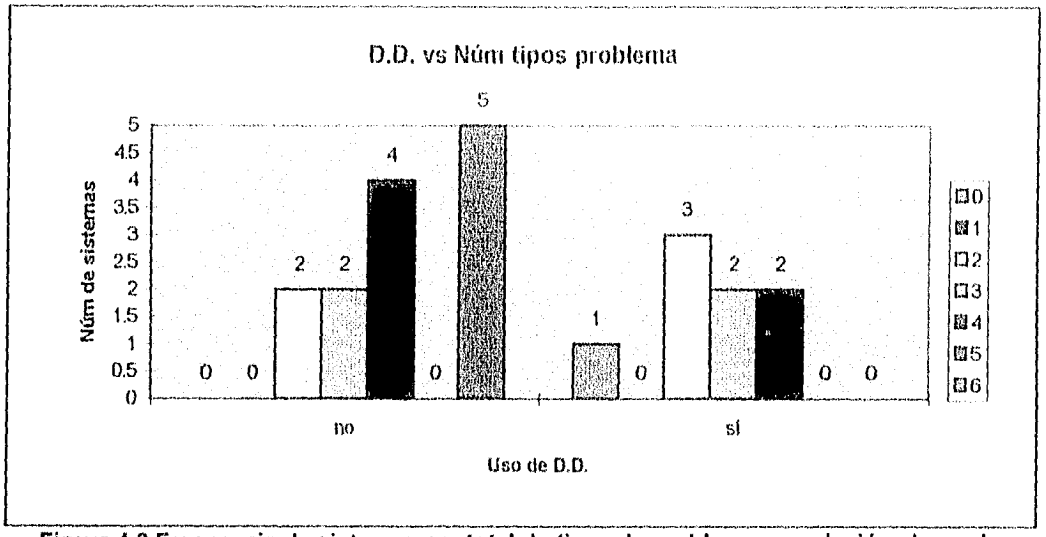


Figura 4.8 Frecuencia de sistemas por total de tipos de problema, en relación al uso de Diccionario de Datos.

En los histogramas anteriores, correspondientes al uso de Diccionario de Datos, se aprecia el descenso relativo en el número de tipos de error, falla y problema cuando se emplea esta metodología. Las medias pasan de 2.53, 1.5, 3.75 a 0.75, 1.37 y 2, respectivamente, confirmando la disminución.

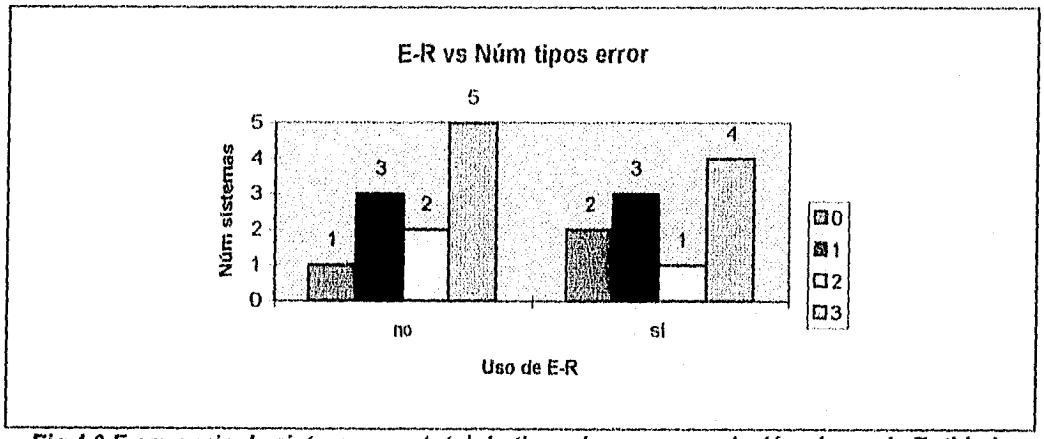


Fig 4.9 Frecuencia de sistemas por total de tipos de error, en relación al uso de Entidad-relación.

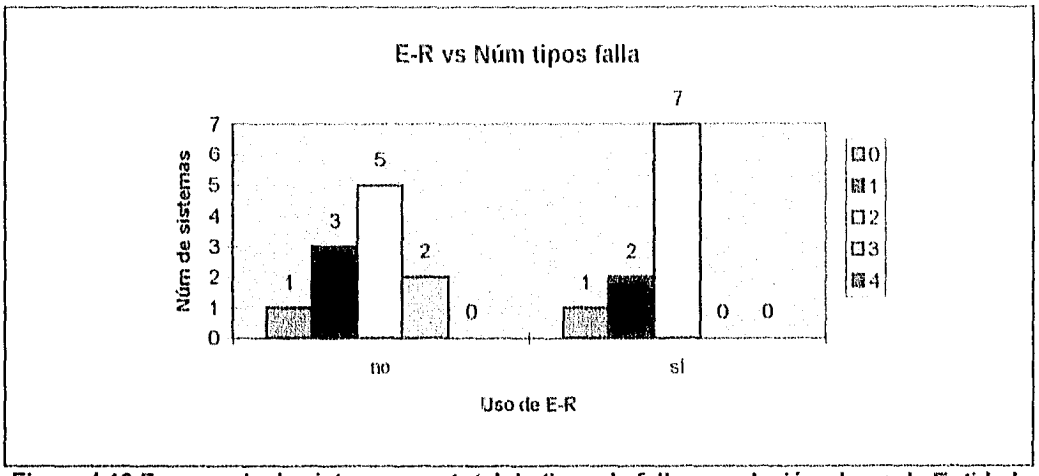


Figura 4.10 Frecuencia de sistemas por total de tipos de falla, en relación al uso de Entidad-relación.

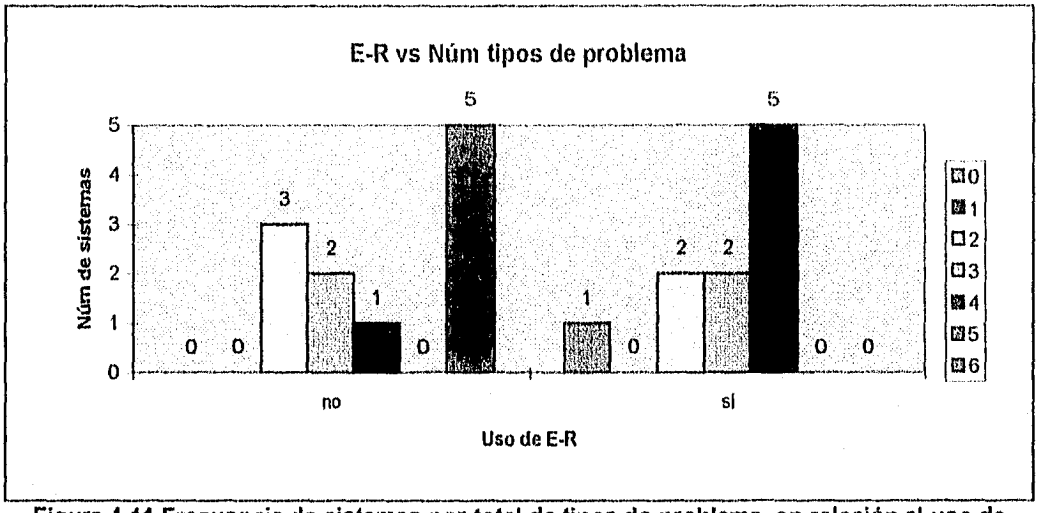


Figura 4.11 Frecuencia de sistemas por total de tipos de problema, en relación al uso de Entidad-relación.

Para el caso del uso de Entidad-relación no se nota una mejora tan clara como en el caso del uso de Diccionario de Datos, pero si una cierta disminución en el número de errores, fallas y problemas. En este caso las medias pasan de 2.0, 1.72, 4.18, a los valores 2.0, 1.6 y 3, respectivamente.

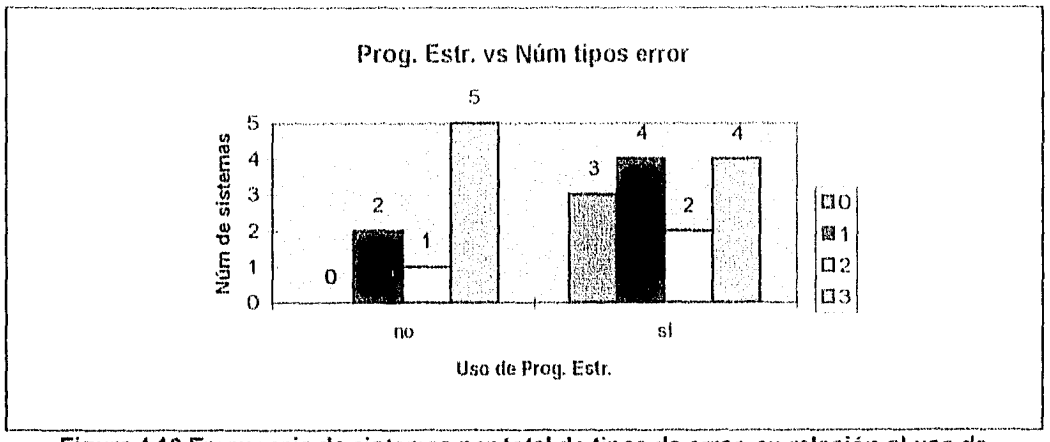


Figura 4.12 Frecuencia de sistemas por total de tipos de error, en relación al uso de Programación estructurada.

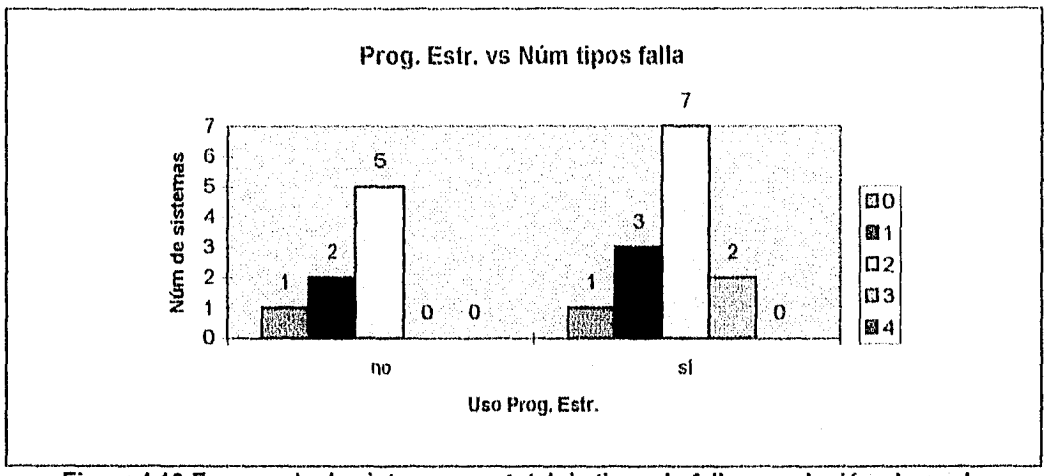


Figura 4.13 Frecuencia de sistemas por total de tipos de falla, en relación al uso de Programación estructurada.

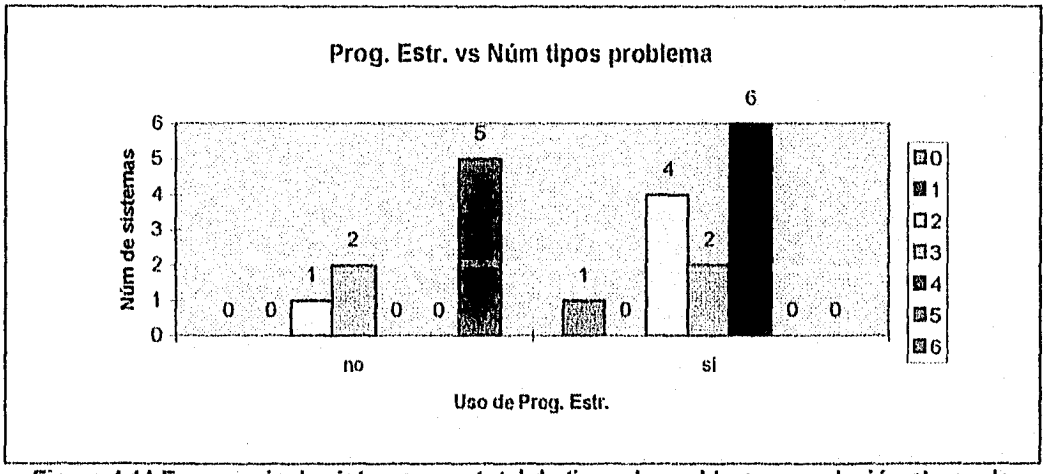


Figura 4.14 Frecuencia de sistemas por total de tipos de problema, en relación al uso de Programación estructurada.

Para el uso de Programación estructurada no es tan evidente que existan mejoras respecto al número de tipos de error, falla o problema. Sin embargo, las medias para el número de errores y de problemas disminuyen, pasando de 2.37 y 4.75 a 1.53 y 2.92, respectivamente.

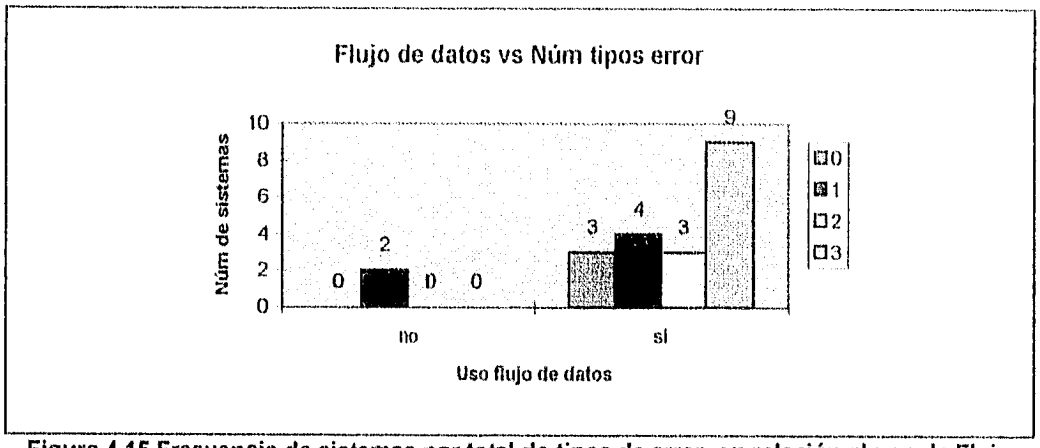


Figura 4.15 Frecuencia de sistemas por total de tipos de error, en relación al uso de Flujo de Datos.

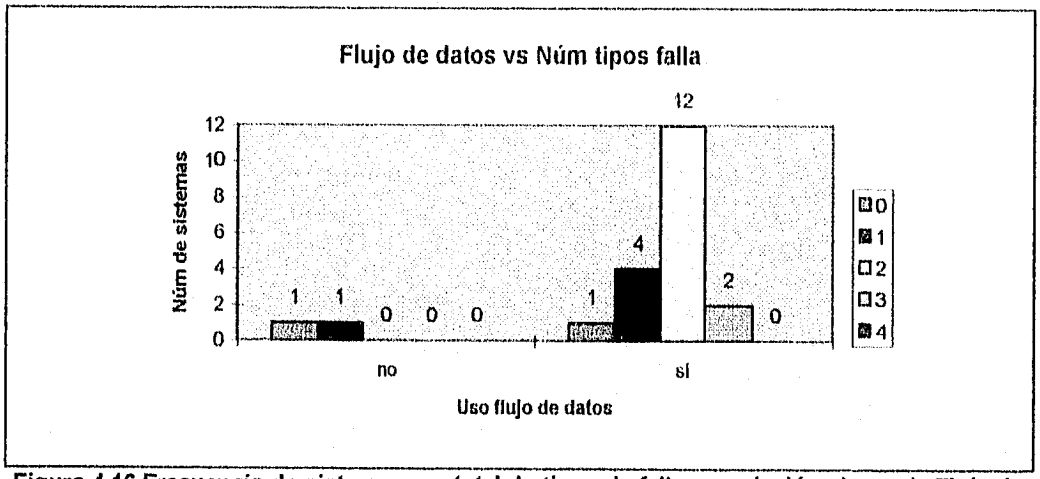


Figura 4.16 Frecuencia de sistemas por total de tipos de falla, en relación al uso de Flujo de Datos.

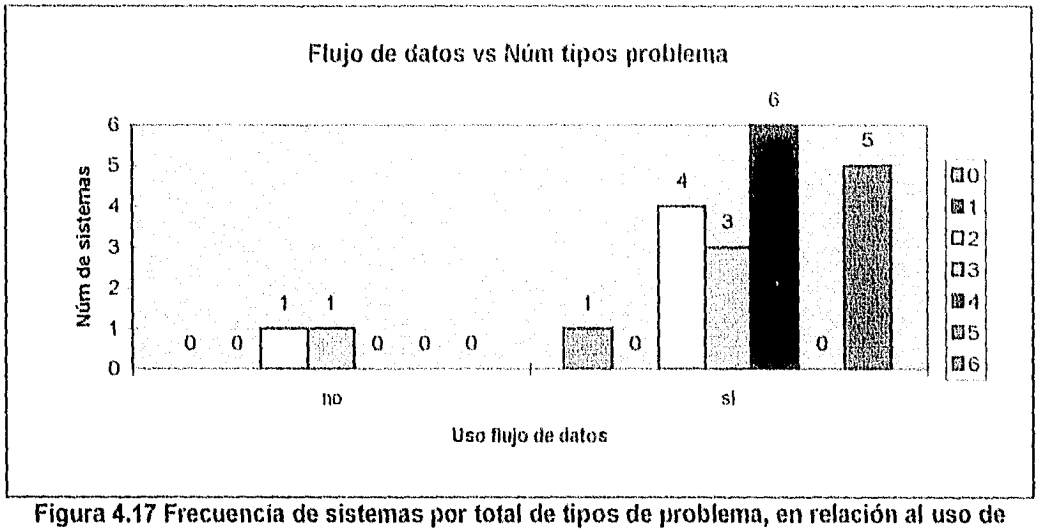


Figura 4.17 Frecuencia de sistemas por total de tipos de problema, en relación al uso de Flujo de Datos.

Los histogramas correspondientes al uso de Flujo de Datos muestran un empeoramiento de la situación, al aumentar los sistemas que presentan muchos tipos de error, falla o problema. Posiblemente se deba a que la gran mayoría (90.5%) de los sistemas observados hacen uso de esta metodología, por lo que realmente no se puede comparar contra el caso de no usarla.

Para profundizar en el efecto que las diferentes metodologías tienen en los errores, fallas y problemas encontrados, se revisó dicho efecto en cada error, falla y problema reportado. Como estas variables son de tipo binario, en vez de aplicar correlación se emplearon tablas de contingencia 2 X 2, como la de la Tabla 4.6[18]:

Tabla 4.6 Ejemplo de tabla de contingencia 2 X 2

	ausencia de variable B	**presencia de variable B**
ausencia de variable A	núm de sistemas con ausencia de A y B	núm de sistemas con ausencia de A y presencia de B
presencia de variable A	núm de sistemas con presencia de A y ausencia de B	núm de sistemas con presencia de A y B

Estas tablas se construyeron para las variables Diccionario de Datos, Entidad-relación, Programación estructurada y Flujo de Datos, contra las variables correspondientes a los diferentes errores, fallas y problemas. Una vez construidas, se aplicó la prueba exacta de Fisher[19] considerando como hipótesis nula que ambas variables son independientes. Si la prueba permite rechazar esta hipótesis, se tiene que esas variables muestran dependencia entre sí.

[18] Para más información acerca de este tipo de tablas, consultar un texto de Estadística, como puede ser el de Sprent [SPR93]

[19] Consultar Sprent [SPR93].

Los resultados de la prueba de Fisher para relacionar metodologías con errores, fallas y problemas se muestran en la Tabla 4.7, para aquellos casos en que se rechaza la hipótesis, es decir, donde sí existe una dependencia significativa. La tabla contiene los datos siguientes: las dos variables comparadas, la frecuencia observada en la celda 0,0 de la tabla de contingencia, que será la empleada para aplicar la prueba de Fisher; en la cuarta columna aparece la suma de probabilidades de obtener un número menor o igual al de la columna anterior, y el resultado (quinta columna) representa el grado de confianza con que se rechaza la hipótesis de independencia entre las variables, es decir, que se acepta que son interdependientes. Por tradición, se usan niveles de significancia del 1%, 5% y a veces hasta 10%. Con pruebas como la de Fisher, los resultados dan valores intermedios, pero puede seguirse la tradición, considerando muy buenos a los menores al 1%. Igualmente, resultados que rebasan el 10% no son necesariamente despreciables, pero no son tan buenos como fuera de desear. Por esta razón se agregaron algunos comentarios en la última columna de la tabla. Los casos que no aparecen definitivamente caen muy lejos de los límites aceptables.

La posible relación entre dos variables puede ser causal, es decir que la presencia de una origina la de la otra, o simplemente que varían juntas, posiblemente por causa de una tercera variable no visible.

Los resultados de esta prueba deben analizarse junto con otros resultados (como los antes presentados), para lograr una visión completa del significado de la dependencia o independencia entre variables.

Tabla 4.7 Resultados de la prueba de Fisher.

Variable A	variable B	val 0,0	Suma prob	Resultado
D-D	err1	3	0.09873949	significativo al 9%
D-D	err2	1	0.07142857	signif al 7%
D-D	falla 4	0	0.15384615	signif al 15% (no muy bueno)
D-D	prob 4	8	0.08301084	signif al 8%
D-D	prob 5	4	0.02863777	signif al 3%
D-D	prob 6	5	0.05789474	signif al 6%
E-R	err2	0	0.1000	signif al 10%
E-R	prob 4	6	0.02979876	signif al 3%
E-R	prob 6	4	0.07989998	signif al 8%
Pr. Est	err2	0	0.12307692	signif al 12%
Pr. Est	prob 4	6	0.02979876	signif al 3%
Pr. Est	prob 5	2	0.08490117	signif al 8.4%
Pr. Est	prob 6	2	0.03989045	signif al 4%
Flu Dat	prob 2	3	0.00361197	signif al 0.36% (muy bueno)

Los resultados de la Tabla 4.7 precisan qué tipos de error, de falla y de problema están más relacionados con alguna de las metodologías consideradas, para los sistemas que se observaron. Por ejemplo, el último caso, nos indica una fuerte relación entre el uso de Flujo de Datos y la aparición de problemas relacionados con el cambio constante de requisitos o petición de nuevos resultados (tipo 2), que es algo que se esperaría en sistemas desarrollados bajo ese enfoque.

Pasando los resultados a palabras, se tienen como pares de variables con mayor significancia en su dependencia (menos de 5%):

Flujo de Datos con problema de tipo 2
Diccionario de Datos con problema de tipo 5
Entidad-relación con problema de tipo 4
Programación estructurada con problema de tipo 4
Programación estructurada con problema de tipo 6

Como puede verse, todos los casos listados se refieren a problemas, es decir, la asociación se presenta hasta el nivel más profundo, donde se originan las fallas, que a su vez causarán los errores. Como se vió antes, la aplicación de las diversas metodologías -a excepción de FD- ayudaban a disminuir el número de errores, fallas y problemas. Así, reuniendo los resultados, puede decirse que estas metodologías son favorables para mejorar la calidad del software.

4.2 Aplicación de la metodología durante todo el ciclo de vida.

En la sección anterior se observaron varios sistemas donde se fueron aplicando una serie de elementos de la metodología propuesta; podría decirse que en ellos se fue gestando. El objetivo de esta nueva sección es comprobar la bondad de la guía propuesta al ser usada por varios equipos de personas durante el desarrollo de un paquete de software. Debe notarse que para ellas era la primera vez que la usaban y poca la práctica que tenían en el desarrollo de sistemas.

Los equipos de personas que se escogieron fueron alumnos del penúltimo semestre de la carrera de Licenciado en Informática de la Facultad de Estadística e Informática de la Universidad Veracruzana. El paquete que se desarrollo fue el de una Contabilidad Casera, cuyo guión se muestra en la Figura 4.18, y que se programó en herramientas Clipper, Visual Basic y Gupta. En cada una de las etapas se proporcionó primero una pequeña guía, se dió un cierto tiempo para su desarrollo, se aplicó un cuestionario sobre su usabilidad[20] y se entregó un documento que cubría la etapa.

GUION: Contabilidad Casera	Escena 1: Establecer ambiente EC define ingresos EC define gastos y sus tipos
PAPELES: EC = Encargado de la casa (esposa o esposo) H = Dependiente económico	Escena 2: Introducción de datos EC introduce ingresos EC introduce gastos H desea entrar al sistema H introduce gastos H no puede entrar
Condiciones de Entrada: Existen ingresos Existen gastos fijos Existen gastos variables Existen gastos imprevistos Se desea ahorrar	Escena 3: Planeación de gastos EC marca los gastos EC revisa ahorro sugerido EC acepta ahorro EC modifica ahorro
Condiciones de salida: Reportes de gastos Reportes de ingresos Reporte de ahorro Recordatorio de pagos	Escena 4: Reportes EC solicita reporte H solicita consulta H recibe consulta H sale por no tener autorización

Figura 4.18 Guión del sistema Contabilidad Casera

[20] Las preguntas utilizadas se basaron en el libro de Pressman [PRE93], y corresponden a criterios de calidad buscados durante las revisiones técnicas formales.

4.2.1 Resultados obtenidos durante el análisis.

Para la etapa de análisis se diseñó un cuestionario acerca de la usabilidad de la metodología propuesta, como se muestra en la Figura 4.19, obteniéndose los resultados que se muestran en la Tabla 4.8.

Para cada pregunta que se hace abajo, señale con una X la columna que usted considere que le corresponde.

	Buena	Mediana	Mala
1.- Expresa claramente lo que quiere en cada paso			
2.- Al final, los requisitos del usuario fueron claramente expresados			
3.- La forma de ir cubriendo las etapas del análisis resultan naturales			
4.- La forma en que se propone plantear las pruebas en la etapa del análisis resulta ...			
5.- Cómo resulta la secuencia de los pasos que se siguieron para el análisis.			

Conteste cada pregunta que sigue:

1.- ¿Considera que hicieron falta elementos para cubrir el análisis? _____
¿cuáles?_____

2.- ¿Después de terminado el análisis piensa que ya se tiene la suficiente información para pasar al diseño?_____¿por qué?_____

Figura 4.19 Cuestionario relativo al análisis.

Tabla 4-8 Usabilidad del estándar propuesto. Análisis.

	Buena	Mediana	Mala		Calificación
1. Expresa claramente lo que quiere en cada paso	9	16	0	25	6.8
2. Al final, los requisitos del usuario fueron claramente expresados	15	10	0	25	8
3. La forma de ir cubriendo las etapas del análisis resultan naturales	16	9	0	25	8.2
4. La forma en que se propone plantear las pruebas en la etapa de análisis resulta:	12	13	0	25	7.4
5. Cómo resulta la secuencia de los pasos que se siguieron para el análisis	18	6	1	25	8.4
Porcentaje promedio	56	43.2	0.8	125	**7.76**

Puede verse que los resultados son bastante aceptables, especialmente considerando que es la primera vez que aplicaban la metodología.

4.2.2 Resultados obtenidos durante el diseño.

Durante la etapa de diseño del sistema se aplicó un cuestionario como el de la Figura 4.20, obteniéndose los resultados que se muestran en la Tabla 4.9.

Para calificar si la metodología le ayudó a cubrir los siguientes puntos, ponga una X en la columna que usted considere adecuada.

	BIEN	REGULAR	MAL
1.- Siente que el sistema cubrió los requisitos especificados en el análisis.			
2.- Se ha conseguido que los programas sean independientes entre sí .			
3.- Se ha considerado la facilidad de mantenimiento.			
4.- Se han especificado la forma en que se comunicarán los diferentes programas del sistema.			
5.- Es adecuado el nivel de detalle que se llegó en el diseño.			
6.- El diseño resultó independiente de consideraciones de hardware.			
7.- El diseño resultó independientes de consideraciones de software.			
8.- Se han especificado la forma de cuidar la tolerancia a errores.			

Conteste las preguntas que siguen:

1.- ¿Considera que le hace falta algo al diseño siguiendo esta metodología? ____¿qué?____

2.- ¿Se sintió a gusto usando esta metodología para la etapa de diseño?_____ ¿por qué?__

Figura 4.20 Cuestionario relativo al diseño

Tabla 4.9 Usabilidad de la metodología. Diseño

	Buena	Mediana	Mala		Calificación
1. Siente que el sistema cubrió los requisitos especificados en el análisis	9	13	0	22	7.05
2. Se ha conseguido que los programas sean independientes entre sí	8	14	0	22	6.82
3. Se ha considerado la facilidad de mantenimiento	7	15	0	22	6.59
4. Se han especificado la forma en que se comunicarán los diferentes programas del sistema	5	16	1	22	5.91
5. Es adecuado el nivel de detalle a que se llegó en el diseño	8	12	2	22	6.36
6. El diseño resultó independiente de consideraciones de hardware	10	12	0	22	7.27
7. El diseño resultó independiente de consideraciones de software	11	11	0	22	7.50
8. Se ha especificado la forma de cuidar la tolerancia a errores	4	13	5	22	4.77
Porcentaje promedio	35.2	60.2	4.5	176	6.53

Como puede verse, también en esta etapa los resultados son aceptables, con muy pocas opiniones en contra.

4.2.3 Resultados obtenidos durante la implantación.

Como en las etapas anteriores, se aplicó la encuesta de la Figura 4.21 y en la Tabla 4.10 se muestran los resultados obtenidos.

	BUENO	REGULAR	MALO
1.- El estilo del lenguaje utilizado para la implantación es adecuado.			
2.- La codificación resultó adecuada a lo especificado en el diseño.			
3.- Se ha hecho un uso adecuado de las convenciones del lenguaje.			
4.- La validación de datos especificada en el diseño fue adecuada.			
5.- La forma en que quedó el sistema permitirá un mantenimiento rápido.			
6.- Las pruebas planteadas en las etapas anteriores fueron suficientes para garantizar la confiabilidad del sistema.			

OBSERVACIONES:

Figura 4.21 Encuesta aplicada para la implantación del sistema

Tabla 4-10 Usabilidad del estándar propuesto. Implantación.					
	Buena	Mediana	Mala		Calificación
1. El estilo del lenguaje utilizado para la implantación es adecuado.	12	4	0	16	8.75
2. La codificación resultó adecuada a lo especificado en el diseño.	6	10	0	16	6.875
3. Se ha hecho un uso adecuado de las convenciones del lenguaje.	13	3	0	16	9.0625
4. La validación de datos especificada en el diseño fue adecuada.	8	8	0	16	7.5
5. La forma en que quedó el sistema permitirá un mantenimiento rápido.	15	1	0	16	9.6875
6. Las pruebas planteadas en las etapas anteriores fueron suficientes para garantizar la confiabilidad del sistema.	12	4	0	16	8.75
Porcentaje promedio	68.75	31.25	0	96	**8.44**

4.3 Metodología de validación estadística a futuro.

Para evaluar el impacto de esta metodología a futuro, se seguiría el esquema de todo programa de calidad es decir, deben definirse medidas y métricas del proceso de construcción de sistemas y, a partir de éstas, entrar en un ciclo de mejoramiento continuo.

El tipo de medidas y métricas que deberían registrarse en un período de uno a tres años en proyectos usando esta metodología son:

Costo.- Que se refiere al costo total de desarrollo y sus componentes como son: costo de programación y costo de instalación.

Esfuerzo.- Se debe medir en términos de horas analista, horas documentador, horas diseñador y horas programador.

Tiempo.- Que puede ser puesto en días de desarrollo, midiendo cada fase: análisis, diseño, implantación y prueba.

Respuesta al mantenimiento.- Promedio de la corrección de acuerdo a si fue: error, falla o problema.

Calidad.- Uso de métricas cuantitativas, como sería: número de fallas sobre tamaño del sistema.

Para lograr lo anterior se propone seguir enseñando la metodología en nivel licenciatura, para que al egresar los alumnos utilicen la metodología en su práctica profesional y mantener contacto con ellos para realizar encuestas, quizá cada tres meses.

Como medida comparativa con otras metodologías, se propone realizar un sondeo entre diferentes desarrolladores y aplicar el mismo tipo de encuestas.

5. EVALUACIÓN DE LA METODOLOGÍA.

En el presente trabajo se ha llegado al punto en que se hace necesario realizar una evaluación de la metodología propuesta; para ello, primero se plantea la forma en que se llevará a cabo y posteriormente se procederá a realizarla. Además, al final de éste capítulo, se habla un poco del comportamiento de la metodología frente a los estándares ISO 9001 y CMM.

5.1 Criterio de evaluación.

Una metodología debe evaluarse desde dos puntos de vista [MAC93]:

- La metodología como tal.
- La calidad de los productos realizados con la metodología.

Como metodología en sí deben revisarse los siguientes aspectos:

- Fundamentación, evolución y experimentación.
- Adaptación.
- La orientación al usuario.
- Soporte para el manejo de proyectos.
- Aseguramiento de la calidad.
- Modelado.

Mientras que los productos que se desarrollen con la metodología pueden revisarse con alguno de los criterios de calidad conocidos. Uno de ellos, el FURPS, fue descrito en el capítulo uno.

Ahora bien, como la metodología aquí propuesta en su conjunto es nueva, la calidad de los productos resulta muy difícil de medir. Por ello, en el capítulo tres, se hicieron pruebas estadísticas de algunas partes de la metodología y se usó en un medio académico para comprobar su usabilidad que. Aún cuando las pruebas anteriores no son suficientes para obtener resultados sobre la calidad, permitirán vislumbrar algunos aspectos.

Así pues, en este capítulo se enfatizará la revisión de la metodología como tal. En las secciones que siguen, se irá aclarando cada uno de los puntos que debe contener una metodología a la vez que se señalará el comportamiento de la metodología propuesta. Los puntos que se desarrollarán fueron tomados, en su mayoría, del trabajo de L. A. Macaulay [MAC93].

5.2 Fundamentación, evolución y experimentación.

En esta parte de la evaluación se revisarán cuatro puntos que debe cumplir una metodología:

Basarse en principios sólidos e hipótesis válidas.- O sea, que la confiablilidad de la metodología se debe a principios y consideraciones teóricas subyacentes en ella.

> La metodología propuesta tiene una fundamentación variada y previamente probada en diferentes contextos (como: Ingeniería de Software e Inteligencia Artificial). Lo que en este trabajo se hizo fue conjuntarlos en una sola metodología siguiendo el ciclo de vida de un sistema. Es conveniente destacar que tanto la metodología de Entidad-Relación como la de Diccionario de Datos son herrarnientas de diseño muy utilizadas y probadas como eficientes. El uso de Guiones en el análisis surgió de manera natural al ser éstos una herramienta de Tratamiento de Lenguaje Natural y en el análisis lo que principalmente se debe manejar es el lenguaje natural. Para probar el sistema nuevamente se utilizó una herramienta natural: las pruebas de integración orientadas a objetos.

Permitir la evolución de ella misma y crear soluciones modificables .- Aquí se habla de que la metodología debe poder adaptarse a la organización donde será utilizada y debe permitir modificaciones iterativas de soluciones, esto es retrabajar los sistemas conforme estos requieran crecer.

> Uno de los principales objetivos de la metodología propuesta fue precisamente permitir la flexibilidad en el mantenimiento de sistemas creados por ella. Así, si vemos la pregunta tres de la encuesta de diseño (se ha considerado la facilidad de mantenimiento) que tuvo un promedio de 6.59 y la pregunta seis del cuestionario de implantación (la forma en que quedó el sistema permitirá un mantenimiento rápido) que obtuvo un promedio mayor a nueve. Por otro lado, el uso de DD mostró en las pruebas estadísticas una tendencia clara a la disminución de errores, mismos que, en su mayoría, son causados por la constante petición de cambios en los sistemas. Así pues, se puede concluir que la metodología muestra una disposición a la modificación de los sistemas, aúnque ésta no resulte tan sencilla o evidente como se esperaba.

Haber sido experimentada .- Una metodología es más creíble si está basada en éxitos y fracasos en un medio ambiente real (aplicaciones comerciales), en lugar de solo estar probada en un medio académico.

Definitivamente en este aspecto la metodología propuesta satisface este aspecto de esta parte de evaluación, ya que el nacimiento de la metodología se debió a las necesidades de los diferentes sistemas en que la autora se vió involucrada y que en la actualidad sigue trabajando.

Ser robusta.- Una metodología debe poder aplicarse para producir sistemas con diferentes tipos de usuarios finales, distintas organizaciones, varios tamaños de sistemas, diversas aplicaciones y aplicable a través de diferentes plataformas.

Dentro de los sistemas que fueron dando vida a la metodología propuesta se pueden destacar varios puntos que la hacen robusta como: los sistemas han sido empleados por usuarios de varios tipos, las aplicaciones han sido variadas, aúnque todas dentro del contexto de procesamiento de datos; se ha trabajado en diferentes tipos de empresas y en distintas plataformas. El único aspecto que no cubre es el tamaño de los sistemas ya que, en general, son considerados pequeños o medianos salvo el caso del Sistema de Información Geográfica CI/SIG. Sin embargo, debe recordarse que la orientación de la metodología es precisamente la de los sistemas pequeños y medianos.

En conclusión, con respecto al criterio que se está revisando puede considerarse buena aúnque aplicable solo a sistemas pequeños o medianos en plataformas variadas y de preferencia en ambientes PC.

5.3 Adecuación.

Una metodología debe tener ciertas características para que pueda ser aceptada en una organización o por un individuo; esto es, debe adecuarse al desarrollador, para lo cual debe cumplir con los puntos que siguen:

Una metodología puede cubrir todo el proceso de desarrollo o solo una parte de éste pero, en el segundo caso, debe poder ser retomada por alguna otra metodología estándar. Cada estado de la metodología debe proveer salidas que después serán retomadas por los estados siguientes. El hecho de que no exista una forma de comunicar las diferentes etapas generalmente es reconocido como la mayor causa de fracaso de los sistemas.

En este aspecto la metodología propuesta se comporta bien tratando de cubrir varias de las etapas del ciclo de vida, y provee salidas en cada una de ellas que podrán ser tomadas para cubrir varias otras. Por ejemplo en el análisis se construye un plan de prueba funcional que servirá en la etapa de implantación y se realizan también diferentes tablas que servirán como inicio del diseño.

Dentro de la encuesta de usabilidad del análisis, la pregunta 2 (ver la Figura 4.8) indica que el 88% respondieron que sí se tiene suficiente información para pasar al diseño.

De los dos párrafos anteriores se desprende pues que la metodología contempla la comunicación entre diferentes etapas del desarrollo del software.

Una metodología debe adecuarse a las prácticas comunes de una organización debido al soporte de conceptos establecidos para la Administración de los Sistemas. Este aspecto se convierte en más crítico en sistemas grandes. Una práctica importante es la modularización, pues debe existir un proceso de jerarquización de las funciones.

Ya que la metodología propuesta está pensada para el desarrollo de paquetes[21] que serán desarrollados por personas o empresas independientes para su posterior venta, el punto de adecuación empresarial quedaría fuera de la evaluación. Sin embargo, si se considera a los elementos que forman la metodología, el hecho es que han sido utilizados en diferentes organizaciones y su uso resultó adecuado.

La realización de prototipos es considerada una práctica deseable dentro de una metodología, ya que estos ayudan a establecer los requisitos de los sistemas.

En la metodología propuesta se tiene contemplado la elaboración de prototipos. Un ejemplo sería el manual de operación del sistema en la etapa del análisis. Además, como apoyo a éstos se tiene la elaboración de los guiones, que son una dramatización de la estrategia propuesta por el analista basada en sus pláticas con el usuario y la situación actual del sistema.

5.4 La orientación al usuario.

En el caso de una metodología, el usuario no es aquel que va a utilizar los sistemas sino quien los va a desarrollar. Entonces, los puntos que se tratan a continuación serán con respecto a la aceptabilidad que pueda tener la metodología para un desarrollador.

El desarrollador acepta una metodología siempre que sea flexible, ésto es, debe adaptarse a su forma de trabajar y ser fácil de aprender y usar.

Todo lo anterior lo abarca la metodología propuesta puesto que: en la encuesta de usabilidad del análisis reporta en la pregunta uno, ("expresa claramente lo que se quiere en cada paso") que se reporta de forma

[21] Software que cubre una función específica y que se adapta a cada organización [LUC81]

mediana fue entendido el uso de ella y la pregunta tres que dice "la forma de ir cubriendo las etapas fueron naturales" obtuvo una calificación de 8.2. Por otro lado en la encuesta de diseño se hizo la pregunta "¿Se sintió a gusto usando esta metodología para la etapa de diseño?" y hubo un 81.81% de personas que expresaron que sí les gustó, pero que les costó trabajo entenderla. Con lo que se puede concluir que más bien fue fácil de usar y que cuesta un poco de trabajo entenderla.

La metodología debe investigar las consecuencias sociales, éticas, estéticas, económicas y prácticas de las soluciones.

Cuando se empezó a utilizar parte de esta metodología se encontraron algunos problemas estéticos (falta de claridad en las interfaces) y ahora se trataron de cubrir haciendo algunas recomendaciones a los desarrolladores. Por otro lado desde el punto de vista práctico, el hecho de permitir el mantenimiento fácil y de poder adaptarse a cualquier lenguaje resulta bueno. En lo que respecta a los problemas éticos, se cuida especialmente el acceso a operaciones y datos a través de las tablas de actores o usuarios y de condiciones, aúnque en gran medida los problemas éticos que puedan aparecer serán responsabilidad del desarrollador de aplicaciones.

Debe permitir y animar la comunicación, el trabajo y los puntos de vista interdisciplinarios. Es decir deberá permitir que haya interacción con el usuario desde el principio y que profesionistas de otras ramas (ejemplo: diseñadores gráficos) dén sus puntos de vista y ayuden a la solución del problema.

En lo que respecta a este punto la metodología propone manejar como parte central de los sistemas un Diccionario de Datos que permitirá ir modificando la solución conforme se vaya necesitando y nunca se aparta de que la interfaz del usuario sea diseñada como parte independiente del software medular. Por otro lado promueve el uso de guiones para dar una claridad visual al usuario y de manuales como prototipo para dejar por escrito todos los puntos relevantes desde un principio.

5.5 Soporte para la administración de proyectos.

Las metodologías deben cubrir aspectos que permitan llevar el control administrativo de los proyectos realizados con ella, los puntos que se deben cubrir son: secuenciación, organización del equipo, estimación de costos y estrategia de planeación. La metodología debe aumentar la productividad por razones financieras, ya que éstas constituyen los principales criterios que la mayoría de las instituciones usan como criterio para adoptar o cambiar a una nueva metodología.

La metodología propuesta carece de varios aspectos de apoyo a la administración de proyectos. Sin embargo, abarca algunos: la tabla de prueba funcional que contiene una columna de tiempo de desarrollo y otra de tiempo real de realización, los cuales ayudarán al control administrativo y a la planeación de nuevos proyectos. Por otro lado la construcción de un metasistema prevé que el código de éste sea completamente reutilizable para otros proyectos, quedando por realizar solo aquellas funciones de presentación de resultados. Con esto la metodología estará ahorrando tiempo, dinero y esfuerzo.

Una metodología debe proporcionar parámetros de rendimiento y permitir la comparación de las soluciones contra los parámetros que se supone debe cubrir.

La metodología propuesta supone una serie de factores a cubrir en cada parte del desarrollo de las soluciones (ver tabla 3.2). Estos elementos sirven como guía para cuantificar el avance y así medir el rendimiento.

5.6 Modelado de requisitos.

Como último aspecto a revisar se tiene el que habla de la forma de presentar los requisitos de los productos. Lo que una metodología debe cubrir para modelar dichos requisitos son los características siguientes:

Alto nivel de abstracción .- Se debe hablar en el lenguaje del usuario.

Legibilidad .- Cualquier persona enterada del problema del usuario debe poder leer las especificaciones del sistema.

Preciso .- Debe suministrarse un lenguaje de especificaciones de alto nivel que no de cabida a ambigüedades.

Especificaciones completas.- Debe existir una forma de presentar las especificaciones completas (Ejemplo: No debe olvidarse poner el tiempo en sistemas de tiempo real).

Permitir el mapeo a etapas posteriores.- La metodología debe proporcionar un método formal de pasar los requisitos del sistema a la fase de diseño.

Para la metodología propuesta se utilizan las quintetas dentro de los guiones; las quintetas pasan a cada etapa transformando su forma de presentación a requisitos y posteriormente a tablas del Diccionario de Datos, que serán utilizadas por la solución del problema. Así, se logra un modelado ligero a la vez que formal.

5.7 La metodología frente a ISO 9001 y CMM.

La metodología propuesta lleva naturalmente al desarrollador por el camino de la documentación. Como todo su desarrollo parte de la documentación de los guiones, donde participa el usuario, va dejando huellas documentales en todo el ciclo de vida. Además, como muchos tramos son automatizables, se puede reforzar el cumplimiento de la metodología y con ello de su documentación. Así, se cumple uno de los aspectos más importantes del ISO 9001.

Ahora bien, como esta metodología no incluye toda la empresa donde se utilice, habría que asegurar el cumplimiento del estándar en otros departamentos. Pero esto rebasa el alcance de cualquier metodología. Por otra parte, por el tipo de sistemas a que está enfocada, es más probable que sea empleada por un subcontratista o productor independiente de software, donde sí se podrá vigilar el cumplimiento de la norma.

Puntualizando, en relación a los puntos listados en la sección 1.3.1.1, la metodología cumple los puntos uno, dos, cinco, siete y nueve. Los puntos tres, cuatro y seis son aspectos administrativos que rebasan el alcance de una metodología. El punto ocho (control de documentación) es fácilmente automatizable sobre las estructuras sugeridas en la metodología.

En resumen, dentro de una empresa comprometida a satisfacer los requisitos del estándar ISO 9001, esta metodología resultaría de gran utilidad.

En relación al modelo CMM, nuevamente debe considerarse que una metodología no es toda una empresa. Ahora bien, en el nivel dos se habla de documentar y apegarse a estándares, aspectos que esta metodología ayuda a cumplir. En el nivel tres se habla de procesos estandarizados, que esta metodología también ayuda a reforzar; de hecho, el manejo de Diccionario de Datos será un proceso estándar por sí mismo, y será fácil establecer otros conforme se aplique a diversos proyectos. Así pues, esta metodología ayuda a una empresa a lograr el nivel dos y el tres. En lo relativo al nivel cuatro, debido a sus limitaciones de tipo administrativo, aún hace falta desarrollarla más.

Para terminar, debe considerarse nuevamente el enfoque de esta metodología hacia sistemas medianos y pequeños, probablemente en un ambiente de productor de software en pequeña escala. En ese tipo de empresa tal vez resulte poco realista hablar de los niveles cuatro y cinco del modelo CMM, al menos en nuestra realidad actual.

6. CONCLUSIONES.

La principal motivación para realizar este trabajo fue la de conseguir una metodología completa para el desarrollo de software en equipos PC que permitiera un fácil mantenimiento. Para lograrlo se conjuntaron y formalizaron varias prácticas e ideas de desarrollo de software, muchas de ellas aprendidas de la experiencia.

Para un analista que escucha a un usuario en una entrevista, su principal vehículo de comunicación es el lenguaje natural. Además cuenta con documentos formales e informales como son:

oficios,
notas en cuaderno,
grabaciones,
recuerdos en la mente del analista,
informes,
listados de datos proporcionados por el usuario.

La labor del analista consiste en construir con todo ello una descripción precisa, el modelo conceptual del sistema, que sea comprensible para:

El usuario.- A fin de que le permita validar el producto antes de su construcción, evitando así retrocesos costosos en el ciclo de vida.

El desarrollador.- Para que la implantación del producto sea correcta.

El analista debe contar con una serie de herramientas que le permitan lograr las descripciones que constituyen un modelo conceptual:

textos e información estructurada,
diagramas,
prototipos.

La Ingeniería de Software brinda algunas de estas herramientas pero deja algunos huecos entre el inicio del proceso de adquisición del conocimiento del sistema y el modelado de los requisitos. La Inteligencia Artificial (IA), por otro lado, brinda un apoyo de formalización del conocimiento que apoya vigorosamente el proceso de modelado.

Más adelante, al pasar a la realización de una solución, el desarrollador de la implantación del sistema necesita una forma de trabajar que le permita generar soluciones rápidas y de calidad. La Ingeniería de Software brinda actualmente una amplia gama de herramientas para lograr la elaboración de productos de calidad.

El mantenimiento es la etapa del ciclo de vida más larga y, además, la más tediosa, razón por la cual resulta necesario reducirla y volverla fácil. Para ello la Ingeniería de Software brinda una herramienta poderosa conocida como Diccionario de Datos (DD). La información relativa a un sistema puede guardarse en un DD y tomarla de ahí para realizar las modificaciones pertinentes y apenas tocar el código de sistemas que ya estaban corriendo, evitando con ello generar errores en cascada.

La metodología propuesta en este trabajo aprovechó, por un lado, la representación del conocimiento que brinda la Inteligencia Artificial y por otro, los DD y metodologías de desarrollo que proporciona la Ingeniería de Software. De la conjunción de ellas surge un resultado que satisface en gran parte el objetivo principal.

Como resultado de este trabajo se obtuvo una metodología bien cimentada, que facilita el mantenimiento, que deja bien claros los requisitos y que puede adaptarse con facilidad a cualquier lenguaje. A pesar de sus aciertos, tiene algunos faltantes, entre los que destaca el soporte para la administración de proyectos.

Durante el desarrollo del trabajo se pudo notar que:

> La aplicación de una nueva metodología no es tarea fácil.
> A los desarrolladores no les gusta leer manuales.
> El hecho de que la "práctica hace al maestro" resulta cierta, pero es mejor si "el maestro" se encuentra dispuesto a conocer y aplicar nuevas ideas.

La elaboración de este trabajo da pie a trabajos futuros, tales como la elaboración de un CASE a partir de la especificación en Lenguaje Natural del estado actual y las necesidades del sistema deseado. Los CASE actuales ya contienen muchos de los elementos que la metodología propone y no parece difícil manejar un programa que dibuje los guiones. Cabe mencionar que la mayoría de los CASE comerciales para PC (Ejemplo: System Architect [SYS90]) se basan en lo que llaman la Enciclopedia, que es precisamente un Diccionario de Datos.

Otra investigación que puede llevarse a cabo, es la de extender ésta metodología a problemas más grandes de los tratados aquí. No se ve que sea imposible lograrlo, pero sí se necesitaría tomar en cuenta más elementos, tales como el uso concurrente de los datos y los tiempos de respuesta a consultas.

APÉNDICE A

CARACTERÍSTICAS ADICIONALES DE LOS SISTEMAS OBSERVADOS.

Tabla A-1 Características del ambiente de cada Sistema

Mnemónico	Plataforma	Arquitectura	Herramienta
1.- EVALPRIM	UNIVAC 1106	CENTRALIZADA	COBOL-ALGOL
2.- ADMISECU	UNIVAC 1106	CENTRALIZADA	COBOL-ALGOL
3.- CALITELE	UNIVAC 1106	CENTRALIZADA	COBOL-ALGOL
4.- ADMINORM	UNIVAC 1106	CENTRALIZADA	COBOL-ALGOL
5.- ABIERTOS	UNIVAC 1106	CENTRALIZADA	COBOL-ALGOL
6.- PESCA	BURROUGHS 6500 / MCP	CENTRALIZADA	DYNAMO
7.- TELEX	Cyber 7000 / SCOP	CENTRALIZADA	FORTRAN
8.- INMUEBLES	Cyber 7000 /SCOP	CENTRALIZADA	COBOL
9.- ENCUESTAS	UNIVAC 1106	CENTRALIZADA	FORTRAN
10.- PAGODIFE	PDP1134 / RSX-11M	CENTRALIZADA	COBOL
11.- CONTPERS	PDP1134 / RSX-11M	CENTRALIZADA	COBOL
12.- RAYARECO	PC/MS-DOS	ÚNICA	CLIPPER
13.- APLIRECU	PC/MS-DOS	ÚNICA	Dbase III+
14.- CONTINVE	PC/MS-DOS	ÚNICA	Dbase III+
15.-CONTABILID	PC/MS-DOS	ÚNICA	Dbase III+
16.- CI/SIG	PC/MS-DOS	ÚNICA	C-FORTRAN-ROOTS-METAWINDOWS
17.- CONSERVA	PC/MS-DOS	ÚNICA	CLIPPER
18.- AVES	PC/MS-DOS	ÚNICA	CLIPPER
19.- ECOMED	PC/MS-DOS	ÚNICA	CLIPPER y CI/SIG
20.- ESFIVE	PC/MS-DOS	ÚNICA	Dbase III+ y C
21.- SIISEV	RED PC/ NOVEL MS-DOS	CENTRALIZADA	C-IDRISI-CLIPPER

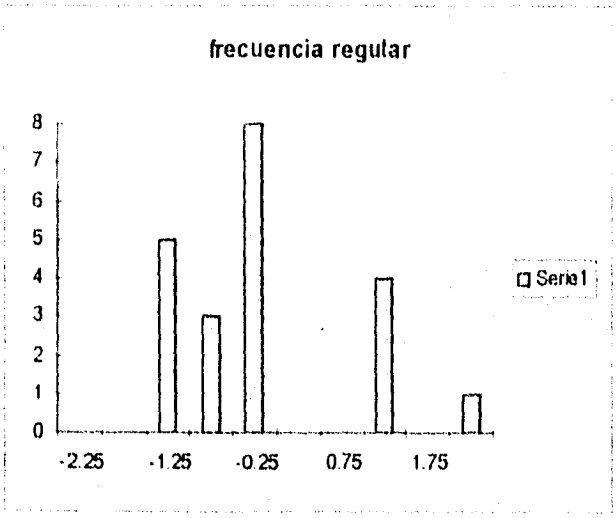
Tabla A-2 Características de los usuarios

Mnemónico	Experiencia del usuario que lo encargó	Experiencia del usuario operador	Actividades del usuario
1.- EVALPRIM	Ninguna	Experto	Programación, operación
2.- ADMISECU	Ninguna	Experto	Programación, operación
3.- CALITELE	Ninguna	Experto	Programación, operación
4.- ADMINORM	Ninguna	Experto	Programación, operación
5.- ABIERTOS	Ninguna	Experto	Programación, operación
6.- PESCA	Conocedor	Conocedor	Programación, operación
7.- TELEX	Ninguna	Experto	Programación, operación
8.- INMUEBLES	Ninguna	Experto	Programación, operación
9.- ENCUESTAS	Ninguna	Experto	Captura
10.- PAGODIFE	Ninguna	Experto	Programación, operación
11.- CONTPERS	Ninguna	Experto	Programación, operación
12.- RAYARECO	Ninguna	Ninguna	Captura, operación
13.- APLIRECU	Ninguna	Ninguna	Captura, operación
14.- CONTINVE	Ninguna	Ninguna	Captura, operación
15.-CONTABILID	Ninguna	Ninguna	Captura, operación
16.- CI/SIG	Experto	Ninguna	Administración, captura, operación
17.- CONSERVA	Ninguna	Ninguna	Administración, captura, operación
18.- AVES	Experto	Ninguna	Operación
19.- ECOMED	Poca	Poca	Operación
20.- ESFIVE	Poca	Poca	Captura, operación
21.- SIISEV	Poca	Poca	Administración, captura, operación

Tabla A-3 Vida del Sistema

Mnemónico	Total de vida útil (años)	Frecuencia de Solicitudes de mantenimiento	Velocidad de Respuesta (días)
1.- EVALPRIM	6	constante	7
2.- ADMISECU	6	constante	7
3.- CALITELE	6	constante	7
4.- ADMINORM	6	constante	7
5.- ABIERTOS	6	constante	7
6.- PESCA	1	constante	1
7.- TELEX	6	constante	2
8.- INMUEBLES	6	constante	1
9.- ENCUESTAS	no se sabe	no hubo	
10.- PAGODIFE	0.25	no hubo	
11.- CONTPERS	no se usó	no hubo	
12.- RAYARECO	7 (en uso)	semestral	2
13.- APLIRECU	7 (en uso)	semestral	2
14.- CONTINVE	1	semestral	2
15.-CONTABILID	1	no hubo	
16.- CI/SIG	7 (en uso)	desconocida	
17.- CONSERVA	6 (en uso)	semestral	1
18.- AVES	no se sabe	no ha habido	
19.- ECOMED	1	no hubo	
20.- ESFIVE	no se sabe	no hubo	
21.- SIISEV	2 (en uso parcial)	semestral	1
22.-LECTOMEDIA	en desarrollo	en desarrollo	

Como un aspecto exploratorio se normalizó el tamaño de los programas de la muestra observada, medida en líneas de código, obteniéndose una distribución como la que se muestra:



BIBLIOGRAFÍA.

[ALK89]	Allkin, Robert White, Richard J.	XDF a languaje for definition and exchange of biologicaldata sets. Description & Manual version 3.3. Computer Section, Royal Britanic Gardens, United Kingdom, Noviembre 1993
[BOO91]	Booch, G.	Object-Oriented Design with Aplications Benjamin/Cummings Publishing, 1991
[CAR83]	Cárdenas, Alfonso F.	Sistemas de Administración de Bancos de Datos Limusa, 1983
[COA90a]	Coad, P., Yourdon, E.	Object Oriented Analysis Prentice-Hall, 1990
[COA90b]	Coad, P., Yourdon, E.	Object Oriented Design Prentice-Hall, 1990
[CLI90]		Clipper 5.0. Reference. Nantucket Corporation, 1990
[EST94]	Estivil, Vladimir Fernández, Juan M. Sumano, Angeles	Ingeniería de Software Apuntes del diplomado en Sistemas y Arquitecturas Avanzadas de Informática del LANIA, 1994
[FER94]	Fernández, Juan M.	Los Sistemas Expertos y su relación con la Investigación de Operaciones análisis de su naturaleza y aplicaciones Tesis de maestria, UNAM, 1994
[GAN79]	Gane, Chris Sarson, Trish	Structured Systems Analysis: tools and techniques Prentice-Hall Inc., 1979
[GRA94]	Grady, R. B	Successfully applying software metrics IEEE Computer, sep. 1994, pp. 18-25.

[HAR85] Harris, Mary Dee
Introduction to Natural Language Processing
Reston Publishing Co., Inc. Prentice-Hall, 1985

[HUL87] Hull, Richard
King, Roger
Semantic Database Modeling: Survey, Applications, and Research Issues
ACM Computing Surveys
Vol. 19, Núm. 3, Septiembre 1987

[JON91] Jones, C.
Applied Software Measurement.
McGraw-Hill, 1991.

[JOR94] Jorgensen, Paul C.
Erickson, Carl
Object-Oriented Integration Testing
Communications of the ACM
Vol. 37, Núm. 9, septiembre 1994

[KOR86] Korth, Henry F.
Silberchatz, Abraham
Database System Concepts
McGraw-Hill, 1986

[LEN90] Lenat, Douglas B.
Guha, Ramanathan
Pittman, Karen
Pratt, Dexter
Shepherd, Mary
CYC: Toward Programs with Common Sense
Communications of ACM
Agosto 1990

[LOZ94] Lozano, Rafael
Diseño de un Sistema Clasificador Genético Utilizando la Metodología Orientada a Objetos
Tesis de Licenciatura en Informática
Universidad Veracruzana, Diciembre 1994

[LUC81] Lucas, Henry C. Jr.
The Analysis Design, and Implementation of Information Systems
segunda edición, McGraw-Hill, 1981

[MAC93] Macaulay, Linda A.
Evaluation Criteria for a Requirements Method
UMIST, Manchester, Reino Unido, 1993

[PAU93] Paulk, Mark
Curtis, Bill
Chrissis, Mary Beth
Capability Maturity Model, version 1.1
IEEE Software, julio 1993

[PEC88] Peckham, Joan
Maryanski, Fred
Semantics Data Models
ACM Computing Surveys
Vol. 20, Núm. 3, 1988

[PFL94]	Pfleeger, S.L., Fenton, N. Page, S.	Evaluating Software Engineering Standars IEEE Computer, sept 1994. pp 71-79
[POT93]	Potts, C.	Software-Engineering research revisited. IEEE Software, sep. 93, pp 19-28.
[PRE88]	Pressman, Roger	Ingeniería de Software, un enfoque práctico segunda edición, McGraw-Hill, 1988
[PRE93]	Pressman, Roger	Ingeniería de Software, un enfoque práctico tercera edición, McGraw-Hill, 1993
[RIC94]	Rich, Elaine, Knight, Kevin	Inteligencia Artificial Segunda Edición, McGraw-Hill, 1994
[ROD83]	Rodríguez, G. Frausto, J.	Diseño Conceptual de Bases de Datos para programas de Control de Energía Memorias de la Conferencia Mexicon 83 IEEE sección México, nov. 1983
[SEN92]	Senn, James A.	Análisis y Diseño de Sistemas de Información Segunda Edición, McGraw-Hill, 1992
[SHE86]	Shemer, I.	SYS-AIDE: An expert aide to system analysis. The Israel Institute of Business Research, Tel Aviv University, working paper 887/86, january 1986.
[SOM88]	Sommerville, Ian	Ingeniería de Software Addison-Wesley Iberoaméricana, 1988
[SOM92]	Sommerville, Ian	Software Engineering cuarta edición, Addison-Wesley, 1992
[SPR93]	Sprent, P.	Applied nonparametrical statistical methods Segunda Edición Chapman & Hall, 1993
[SYS90]		System Architect. User Guide. Popkin Software & Systems Incorporated 1987-90

[TEO86] Teorey, Toby J.
Yang, Dongqing
Fry, James P.
A Logical Design Methodology for Relational Databases Using then Extended Entity-Relationship Model
ACM Computing Surveys
Vol. 18, núm. 8, 1986

[WIE92] Wiederhold, Gio
Wegner, Peter
Ceri, Stefano
Toward Mega Programming
Communications of the ACM
noviembre 1992

[YOU90] Yourdon, Edward
Análisis Estructurado Moderno
Prentice Hall, 1990