22

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

UNA VARIANTE DE LA METODOLOGIA DE YOURDON APLICADA AL DESARROLLO DE UN GENERADOR AUTOMATICO DE REACTIVOS.

T E S I S

QUE PARA OBTENER EL TITULO DE:

A C T U A R I O

P R E S E N T A :

LUCIO GERARDO CHAVEZ HEREDIA

DIRECTOR DE TESIS:

M. EN C. JOSE ANTONIO FLORES DIAZ

MEXICO, D. F. 2001

FACULTAD DE CIENCIAS
SECCION ESCOLAR

UNIVERSIDAD NACIONAL AVPNºMA DE MEXICO

M. en C. Virginia Abrín Batule
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
P r e s e n t e

Comunicamos a usted que hemos revisado el trabajo de Tesis:

UNA VARIANTE DE LA METODOLOGIA DE YOURDON APLICADA AL DESARROLLO DE UN GENERADOR AUTOMATICO DE REACTIVOS

realizado por Lucio Gerardo Chávez Heredia

con número de cuenta 7609002-8 , pasante de la carrera de Actuaría

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis
Propietario M. en C. José Antonio Flores Díaz

Propietario M. en C. Ma. Guadalupe E. Ibarguengoitia González

Propietario M. en A. P. MA. del Pilar Alonso Reyes

Suplente Act. MA. Guadalupe Tzintzún Cervantes

Suplente Act. Ana Mireya Pareja Rendón

Consejo Departamental de Matemáticas

M. en A. P. MA. del Pilar Alonso Reyes

RECONOCIMIENTOS

A Rocio y a mis hijos Gustavo, Ariadna y Karen, así como a mi madre, Doña Esperanza.

A mis hermanas Virginia, Blanca y Lucía, así como a mis sobrinos y sobrinas, tios, tias, primos y primas.

A mis muertos: Don Lucio, mi hermana Patricia y mi hermano Gerardo.

A todo mi profesorado, particularmente al de mi Alma mater.

AGRADECIMIENTOS

A José Antonio por su infinita paciencia y amistad.

A Pilar por todo.

A Lupita por su amistad y enseñanzas.

A Guadalupe y Mireya por su valioso tiempo.

A mi Preparatoria 7 y a mi Facultad de Ciencias de la UNAM, por formarme académicamente... y un poco más en otras áreas.

A la Dirección General de Personal de la UNAM y mis compañeros de batalla. Ahí también me forme en lo profesional.

A todas aquéllas personas que directa o indirectamente, positiva o negativamente, influyeron para que terminara este trabajo.

UNA VARIANTE DE LA METODOLOGÍA DE YOURDON APLICADA AL DESARROLLO DE UN GENERADOR AUTOMÁTICO DE REACTIVOS

Lucio Gerardo Chávez Heredia

CONTENIDO

* 1. EXTRACTO

Este documento establece un diagnóstico sobre la generación de exámenes o tareas en la UNAM, a partir del cuál se plantea la construcción de un **Generador Automático de Reactivos** (GAR). Se espera que dicha aplicación de cómputo coadyuve al profesorado en el mejor desempeño de sus funciones, al proporcionarles una herramienta que les permita incrementar la calidad de los reactivos usados en los exámenes o tareas, necesarios para la evaluación del aprendizaje de los alumnos.

El documento establece un marco teórico relativo a la metodología de Yourdon para establecer una **variante** de la misma, contemplando al efecto técnicas de RAD y de Warnier; dicha variación puede emplearse en organizaciones amplias y, particularmente, facilitó el desarrollo de la aplicación de cómputo en cuestión.

Con base en lo anterior, se explica la aplicación de la variante de la metodología de Yourdon al GAR, acerca del cuál se plantean sus antecedentes, su marco y su diseño conceptual, presentando también una descriptiva de la aplicación de cómputo y documentación técnica de la misma, así como las conclusiones del caso.

* En esta tesis textos en **negrilla** además de indicar títulos, señalan conceptos importantes que se usan por primera vez. *Cursivas* corresponden a latinismos o palabras extranjeras; *"cursivas"* se refieren a citas textuales; 'texto' indica conceptos del área de sistemas de cómputo que incluso correspondan a términos considerados en castellano como barbarismos (*v.g.* 'implementar'). "Texto normal" hace referencia a frases coloquiales o empleadas propiamente por el autor. Por lo anterior, mis más amplias disculpas a nuestra lengua y a sus usuarios.

2. OBJETIVOS

Proponer una metodología que permita a organizaciones amplias administrar la elaboración de sistemas y/o aplicaciones de cómputo como además, definir recursos, **'actividades'** e instancias organizacionales involucradas en la materia, persiguiendo integrar a los usuarios en su análisis e **'implementación'**, así como homogeneizar dichos sistemas y/o aplicaciones a través de la **'estandarización'** de su documentación, **'funcionalidad'**, forma y presentación, de manera tal que se facilite su control, coordinación y enlace, generando así flujos de información uniformes y consistencia informática organizacional.

Mostrar como un caso particular de la aplicación de la metodología propuesta la construcción del **'prototipo'** de un generador automático de reactivos, que permita elaborar de manera automática y sistemática instrumentos de evaluación dirigidos al aprendizaje de los alumnos.

Facilitar un medio que permita al profesorado homologar los procesos para el diseño y elaboración de exámenes o tareas, así como homogeneizar los criterios para evaluar cursos.

Cap. able
a 1 cap. a l neuso
l pamael

3. DIAGNÓSTICO DE LA GENERACIÓN DE EXÁMENES Y TAREAS EN LA UNAM

Con base en múltiples entrevistas con profesores de la UNAM, se establecieron las siguientes conclusiones diagnósticas en la materia:

* Inexistencia o insuficiencia de documentación sobre la elaboración de exámenes o tareas dirigidas a la evaluación del aprendizaje de los alumnos.

* Falta de uniformidad en la forma y presentación de los exámenes o tareas dirigidas a la evaluación de los alumnos que deriva, junto con otros factores, en una alta hetereogeneidad en la forma de evaluar el aprendizaje (grupos "con bajo rendimiento" y "con alto rendimiento").

* Uso generalizado muy limitado de herramientas de *software* en la materia, pues en la mayoría de los casos sólo se emplean **'procesadores de textos'** y en algunas ocasiones se combinan con alguna otra herramienta de cómputo al efecto (*e.g.* manipuladores de imágenes).

* Bajo nivel de motivación y de formación profesional para elaborar instrumentos de evaluación del aprendizaje.

* En algunas asignaturas, sobretodo en aquéllas donde la bibliografía es limitada o la investigación prácticamente nula, alta repetición de reactivos en exámenes o tareas -incluso en grupos con profesorado diferente-.

4. MARCO TEÓRICO

La metodología expuesta a continuación es conocida como **ciclo de vida del sistema** y consiste de un conjunto de especificaciones y métodos formales para su construcción; fue publicada por Edward Yourdon en 1982 y está orientada al desarrollo de sistemas en **'forma estructurada'**. A pesar de que se le pueda considerar como una metodología ya muy vista y tal vez trillada en comparación con otras más en boga, esto no invalida su utilidad. Se seleccionó como marco teórico entre otras metodologías tales como las de Yourdon y Coad -orientada a **'objetos'**-, Warnier -para **'programación estructurada'**- o Board -'prototipos'-, con base en su actual amplia difusión tanto en México como en aquel de origen para los productos comerciales de *software* que se importan a fin de desarrollar aplicaciones de cómputo y, fundamentalmente, porque obliga a generar documentación de los sistemas, indispensable para fines de mantenimiento en organizaciones e instituciones amplias. Dicha metodología tiene como influencia principal la de Tom de Marco (1978) y contempla la posibilidad de integrar métodos y herramientas ampliamente difundidas tales como los **'diagramas de flujo'**. Según el propio autor, es recomendable su uso cuando se trabajan simultáneamente diversos proyectos "simples" o "difíciles", de hasta 9,999 **'líneas de programación'** o de entre 10,000 y 100,000, respectivamente, o bien, cuando se tiene uno "complejo" de entre 100,000 y hasta 1'000,000 de esas 'líneas'*.

4.1 HERRAMIENTAS GRÁFICAS Y NOTACIÓN

Las herramientas gráficas que se mencionan enseguida son las más relevantes para generar la documentación de sistemas en la metodología de Yourdon y son empleadas particularmente en la variante de dicha metodología expuesta en la presente tesis (Cap. 5). Son usadas primordialmente en un contexto de 'actividades' de análisis, sin excluir su empleo en otros

* Al respecto me permito comentar que "sistema complejo aterrizado" implica "alguna metodología se aplicó en el desarrollo del sistema".

casos. La documentación generada con base en herramientas gráficas puede contener textos o narrativa de soporte al efecto, con el fin de detallar cuanto sea necesario para lograr entendimiento común, primordialmente, entre el usuario y el analista, así como entre otras instancias involucradas en el desarrollo de un sistema o aplicación de cómputo. Dichas herramientas facilitan la lectura y comprensión de la documentación en cuestión, mediante los siguientes atributos que la caracterízan:

* gráfica -más que transcrita-,
* **'modular'** o particionada -en vez de constituir un ente completo prácticamente indivisible-,
* independiente de su 'implementación' -con lo que es viable lograr un modelo abstracto del sistema-,
* ***top-down*** o **'descendente'** -lo que permite presentar una descriptiva del sistema en niveles progresivos de detalle-.

Las herramientas en cuestión se describen enseguida a través de su aplicación a un supuesto "Sistema de inventario de equipos".

4.1.1 Diagramas de flujo de datos (DFD -*Data Flow Diagrams*-)

Representación gráfica mediante la cual es posible representar o "modelar" sistemas, sean manuales o automatizados o incluso mezcla de ambos, a través del uso de **'flechas de flujo de datos'**, **'terminadores'** (rectángulos), **'procesos'** (círculos o **'burbujas'**) y **'almacenes de datos'** o ***data stores*** (líneas paralelas). Las 'burbujas' mencionadas pueden **'explotarse'**, es decir, ser descompuestas en términos de otras a efecto de ser examinadas en mayor detalle, generando así diferentes niveles de DFD; de esta forma, cualquier sistema de cómputo puede ser disgregado para efectos de su comprensión y análisis o diseño en términos de diversos niveles de DFD, donde invariablemente debe de iniciarse con un **'diagrama de contexto'** o **'de nivel 0'**, que contiene una sola 'burbuja', misma que representa al sistema en su **medio ambiente**, *i.e.* con otros sistemas y fuentes de datos con los que tiene que ver, para

continuar con niveles subsecuentes de DFD, de mayor detalle. Todo lo anterior se ejemplifica a continuación para el mencionado "Sistema de inventario" (figuras 1 a 3):

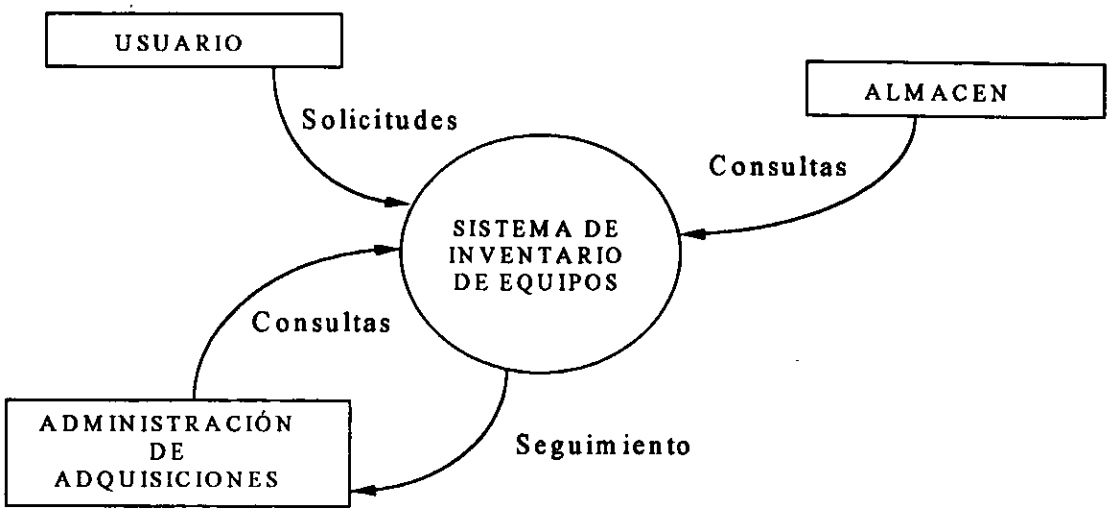


FIGURA 1: DIAGRAMA DE CONTEXTO DEL SISTEMA DE INVENTARIO DE EQUIPOS.

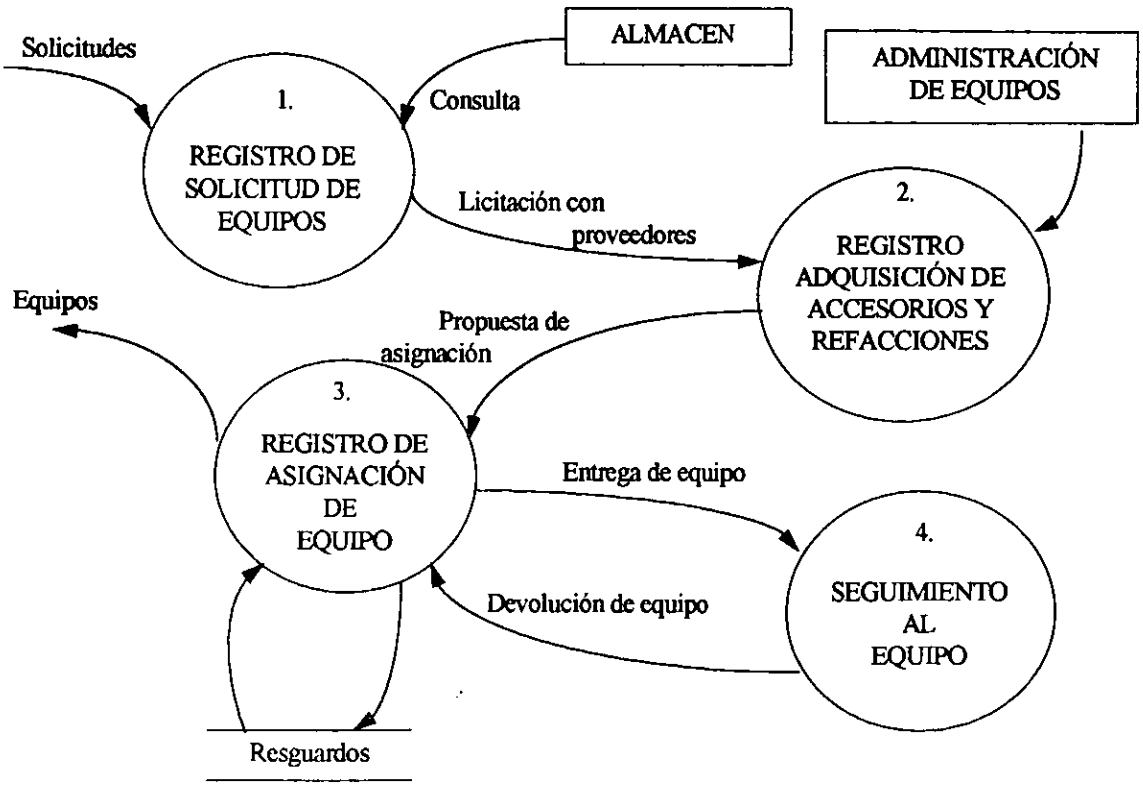


FIGURA 2: DIAGRAMA DE FLUJO DE DATOS DEL SISTEMA DE INVENTARIO DE EQUIPOS

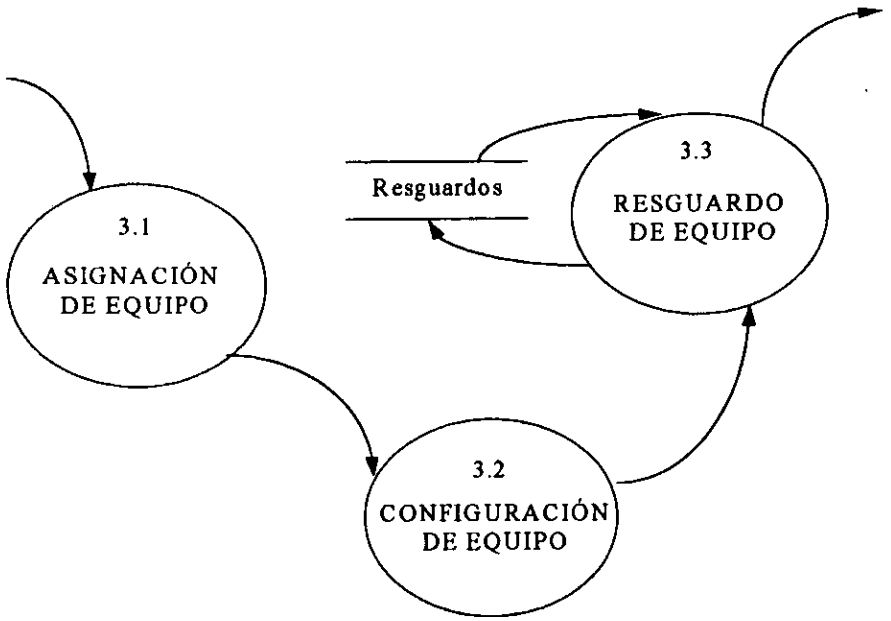


FIGURA 3: DFD DE REGISTRO DE ASIGNACIÓN DE EQUIPO

4.1.2 Diagramas de 'entidad-relación' (ERD -*Entity Relationship Diagrams*-)

A través de ellos se definen las **'entidades'** y **'relaciones'** que manipulará el sistema propuesto, mediante **'tablas'** definidas durante el diseño e 'implementación' de la(s) **'base(s) de datos'** del mismo[†]. Cabe aclarar que en en los DFD las 'entidades' corresponden típicamente a *data stores*, pero que estos diagramas no dicen nada acerca de las 'relaciones' entre las 'entidades', por ejemplo sobre su **'cardinalidad'** -si la 'relación' es **'uno a uno'**, **'uno a muchos'**, etc.-; por otra parte, la definición y **'atributos'** de las 'entidades' y 'relaciones' deben incluirse en el **'diccionario de datos'** -explicado adelante en 4.1.4-. En los ERD la representación gráfica de las 'entidades' corresponde a rectángulos y la de las 'relaciones' a rombos, como se muestra en la figura 4 (página siguiente):

[†] Debo mencionar genéricamente, que por entidad se entiende al conjunto de atributos que la definen propiamente (*v.g.* nombre está compuesto por nombre de pila, apellido materno y apellido paterno) y que relación es una forma de asociación entre varias entidades (*e.g.* nombre-domicilio). Tabla corresponde a un archivo de cómputo que tiene los datos de las entidades o relaciones en cuestión, siendo representados a través de renglones y columnas. Base(s) de datos es un conjunto de tablas relacionadas de alguna manera entre si.

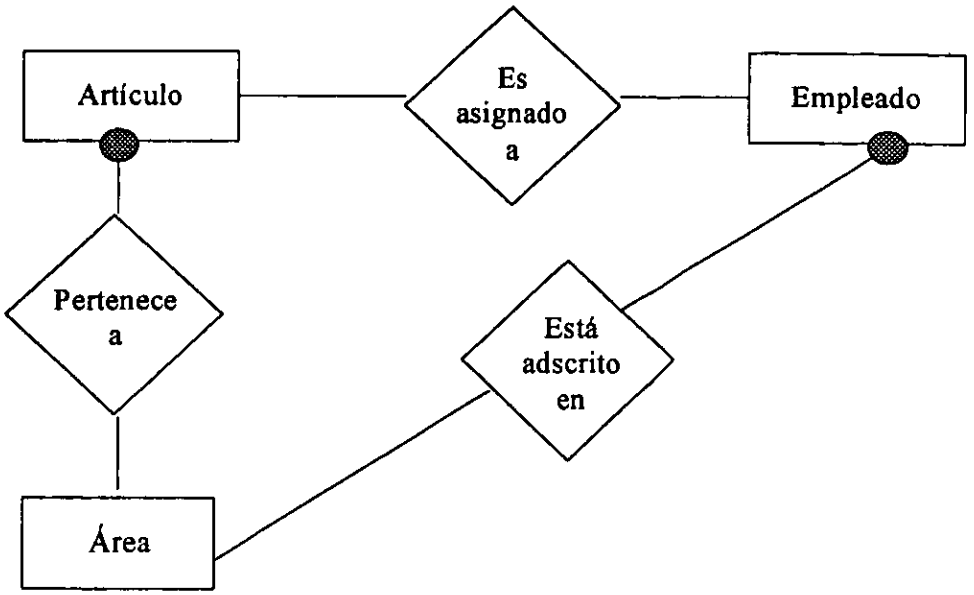


FIGURA 4: ERD DE ARTÍCULOS, EMPLEADOS Y ÁREAS

A guis de ejemplo, nótese que la "bola negra" de Áreas pretende indicar que la relación entre Áreas y Empleados es 'uno a muchos' (*i.e.*, diversos empleados están adscritos a una Área o, por cada Área hay diversos empleados). Caso análogo para la relación Área-Artículo.

4.1.3 Diagramas de transición de estados (STD -*State Transition Diagramas*-)

En lo general corresponden a **'diagramas de bloques'** que representan la secuencia de la 'funcionalidad' del sistema en el tiempo, esto es, su comportamiento en términos de estados, situaciones o escenarios reconocibles, en los que puede estar el sistema o aplicación de cómputo, dependiendo del tiempo y de los eventos que en él transcurren; donde cada estado describe un lapso en el que el sistema mantiene un cierto comportamiento identificable. La representación gráfica de esos estados corresponde a rectángulos y se conectan por flechas que muestran los cambios o transiciones entre ellos; asociado a cada cambio de estado habrá una o más condiciones -justamente los eventos o circunstancias que los originan-, así como cero o más acciones de respuesta del sistema o, también, acciones que se ejecutan pero que son **'transparentes'** para el usuario, como se ejemplifica en la figura 5:

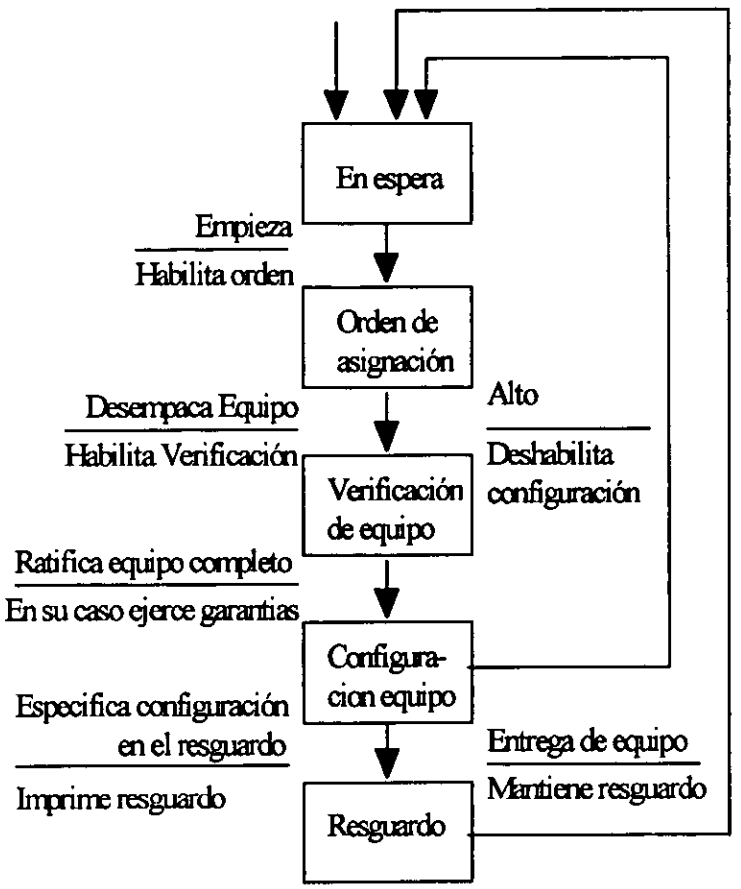


FIGURA 5: STD DE ASIGNACIÓN DE EQUIPO[‡]

Cabe mencionar que en el caso de sistemas **'batch'**, generalmente de un solo usuario, los estados se reflejan directamente en la secuencia de ejecución del **'código'**, característica que es muy diferente para sistemas **'multiusuario en línea'**, que generalmente tienen un comportamiento complejo y diversos 'procesos' OLTP (*Online Transaction Processing*), dependiente del tiempo y de los sucesos que proporcionen los usuarios, la propia aplicación y/o la 'base de datos' (*e.g.* ***'triggers'***), por lo que deben ser diseñados y modelados cuidadosamente en términos de su 'funcionalidad' y relaciones de afectación entre los datos.

[‡] Lo importante en este tipo de diagramas es clarificar las condiciones que deben ser verdaderas antes ("Pre") y después ("Post") de que se ejecute un 'proceso'.

4.1.4 'Diccionario de datos' (DD -*Data Dictionary*-)

Conjunto organizado de información con definiciones y características de todos los datos en el sistema propuesto, esto es, contiene especificaciones de los mismos. Generalmente se contempla la siguiente notación:

x = a + b	x consiste de a y de b
x = [a | b]	x consiste de a ó de b
x = a + (b)	x consiste de a y opcionalmente de b
x = {a}	x consiste de 0 o más ocurrencias de a

Un ejemplo en la materia es este:

Catálogo de Artículos = Nombre | Descripción Genérica + (Descripción Particular) + Marca + Modelo + (Número de parte)

Por lo general se adicionan, entre otras, características como **'tipos'** y **'rangos'** de validez a los datos que manipulará el sistema, su ubicación en las 'tablas' de la(s) 'base(s) de datos', etc.; al respecto es necesario convenir en **'estándares'** para: nombres de 'tablas', **'catálogos'**, **'campos'** o **'atributos'**, así como para la creación de **'librerías'** y declaración de **'funciones'**, **'procedimientos'**, etc. Con relación a los 'estándares' se plantea una propuesta en el Anexo 2 de este documento. Continuando con el ejemplo, se muestra un fragmento de su DD:

TABLA	CAMPO	DESCRIPCIÓN	LONG	RANGO
CARTIC	*CveArt	**Clave** del **Artículo** (llave)	10	1..99,999,999
CARTIC	ANomArt	**Nombre** del **Artículo**	25	Acepta: /,-,:,;
CARTIC	ADesGenArt	**Descripción Genérica** del **Artículo**	25	Acepta: /,-,:,;
CARTIC	ADesParArt	**Descripción Particular** del **Artículo**	40	Acepta: /,-,:,;
CARTIC	AMar	**Marca**	20	'A'..'Z','0'..'9'
CARTIC	AMod	**Modelo**	20	'A'..'Z','0'..'9'
CARTIC	ANumPar	**Número de Parte**	10	'A'..'Z','0'..'9'
CARTIC	AStaArt	**Status** del **Artículo**	01	'C','A'

De igual forma es conveniente, adicional al DD, definir un glosario que coadyuve en su consulta. Por ejemplo: *"El número de parte es manejado por algunos proveedores para determinar con toda precisión piezas o consumibles que componen o funcionan para algunos de sus artículos (e.g. cada toner tiene asociado un número de parte según el tipo de impresora lasser)"*. La estándarización de aspectos como los comentados y la descripción detallada de la información propia del sistema persigue, entre otros rubros, evitar construirlos sobre "torres de Babel", mantener consistencia en los desarrollos y que posteriormente, personas ajenas a los proyectos puedan darles mantenimiento comprendiendo que es cada término (*e.g.* no resulta fácil saber qué significa SIEKTAR2).

4.1.5 Especificaciones de 'procesos' (*Process specifications*)

Entre otros, contienen la descripción de las políticas del usuario o **'reglas de negocio'** que debe ejecutar el sistema. Generalmente corresponden a **'textos estructurados'** que pueden contener condiciones que deben ser verdaderas antes y/o después de que opere un **'procesamiento'** o bien, a 'diagramas de flujo', **'tablas de decisión'**, etc. La idea central es que cada 'proceso' identificado sea sustentado por una especificación que comprenda y describa con precisión una sola pieza del sistema en su conjunto. Un ejemplo de una **'miniespecificación de proceso'** o simplemente **'miniespecificación'** es la siguiente:

1. *SI* la cotización del dólar llega a $8.75 *ENTONCES*:
    a. Estimar costos de compra
    b. Determinar existencias en inventario
    c. Calcular déficit presupuestal

2. *EN OTRO CASO*:
    a. Determinar liquidez
    b. Comprar insumos, según peticiones previas y liquidez

Típicamente las 'miniespecificaciones' se componen de verbos en imperativo (*v.g.* calcula, valida), instrucciones de control de **'programación'** (*e.g.* Si... Entonces... En otro caso... ;

Mientras... Ejecuta...), bases de cálculo o fórmulas matemáticas y, conceptos que pueden incluso definirse **'localmente'** dentro de la propia 'miniespecificación', pero que en su momento y en su caso, se deben reflejar en el DD y/o en el glosario. Cabe mencionar que en estas especificaciones, no es conveniente establecer el **'algoritmo del proceso'** mediante el cuál se producirá el comportamiento planteado para el sistema, a fin de que el diseñador o el programador proponga el más apropiado, con y desde un punto de vista de diseño e 'implementación'. A las mencionadas 'miniespecificaciones' se les denomina también como **'pseudocódigo'**.

4.1.6 'Interfaz' o 'interface' de usuario (GUI -*Graphics User Interface*-)

Para efectos de diseño e 'implementación' de los sistemas, particularmente en lo relativo a su 'interfaz' de usuario en un contexto del ambiente de ***Windows***, es conveniente definir, convencionalmente, 'estándares' que facilitarán el mantenimiento del sistema en materia de: **'ventanas'**, **'diálogos'**, **'mensajes'**, **'errores'**, **'menús'**, uso de ***'mouse'***, etc.; por ejemplo para el uso de **'cursores'**, retomar algunos de los predefinidos para el ambiente mencionado, como se muestra enseguida:

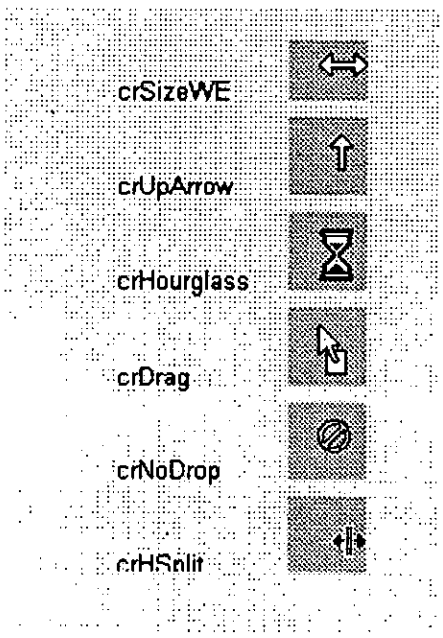


FIGURA 6: MUESTRA DE CURSORES

4.2. METODOLOGÍA DE YOURDON

La metodología expuesta a continuación es orientada al **'desarrollo estructurado'** de sistemas y entre los conceptos que emplea hay tres indispensables para su comprensión: las denominadas fases de análisis, diseño e 'implementación'. Para dichos términos, es posible manejar dos características que permitirían denominarlos como "clásicos" (*v.g.* "análisis clásico"), por su uso y aplicación en el contexto de otras metodologías:

* una fuerte tendencia a 'implementar' los sistemas en forma **'ascendente'** (*bottom-up*)

* una insistencia sobre una progresión lineal y secuencial de una fase a otra.

Por 'implementación ascendente' se entiende, clásica y genéricamente, una construcción de subsistemas en forma independiente, con una visión también independiente e incluso, inconsciente del resto del propio sistema, que posteriormente a su 'implementación' particular, serán "pegados" en un sistema único. A su vez, por progresión lineal y secuencial de fases se entiende que cualquiera de ellas no inicia sino hasta que se ha dado por concluida la precedente (*i.e.* no se comienza el diseño hasta concluir el análisis).

En el caso de la metodología de Yourdon, si bien hay otras 'actividades' bien especificadas además de las ya mencionadas, propone la construcción de sistemas por una parte, en forma *top-down*, con un enfoque integrador que permita suspender temporalmente 'actividades' hasta un cierto nivel de profundidad, dejando la necesaria especificación detallada de sus **'tareas'** (*v.g.* secuencia de 'procesos' definidos por un 'código') para consideraciones subsecuentes; por otra parte, también propone la ejecución de las mismas no necesariamente en forma lineal secuencial, en razón de que esto no permite retroalimentación entre una 'actividad' y otra. Esto respondería a la típica pregunta sobre si es posible para un proyecto tener un 20 % de avance en el análisis y un 25% en el diseño. No es muy ortodoxo, pero definitivamente es posible.

Justamente en la metodología de Yourdon, se sugiere la ejecución de diversas 'actividades' y de sus 'tareas' en paralelo, con un sinfin de variantes de grado en la forma de aplicación del paralelismo en cuestión, desde una forma "ultraconservadora", donde se contempla la ejecución estricta de 'actividades' lineal y secuencialmente, hasta en una forma "radical", donde prácticamente se inicia la generación de 'código' desde el comienzo del proyecto y el análisis concluye, incluso, hacia la parte final del mismo: una alternativa a esa forma radical de trabajar, consiste en el modelado de sistemas por 'prototipos' o **'desarrollo iterativo'** (ver 4.3). Como un ejemplo sobre las variantes de grado mencionadas, es posible considerar iniciar la 'actividad' de 'implementación' cuando se tenga un 75% de avance en el análisis y un 50% en el diseño, por citar solamente dos cifras en el amplio rango de variabilidad de las mismas. Considerando lo antes expuesto, las 'actividades' del método de Yourdon se representan mediante el siguiente DFD:

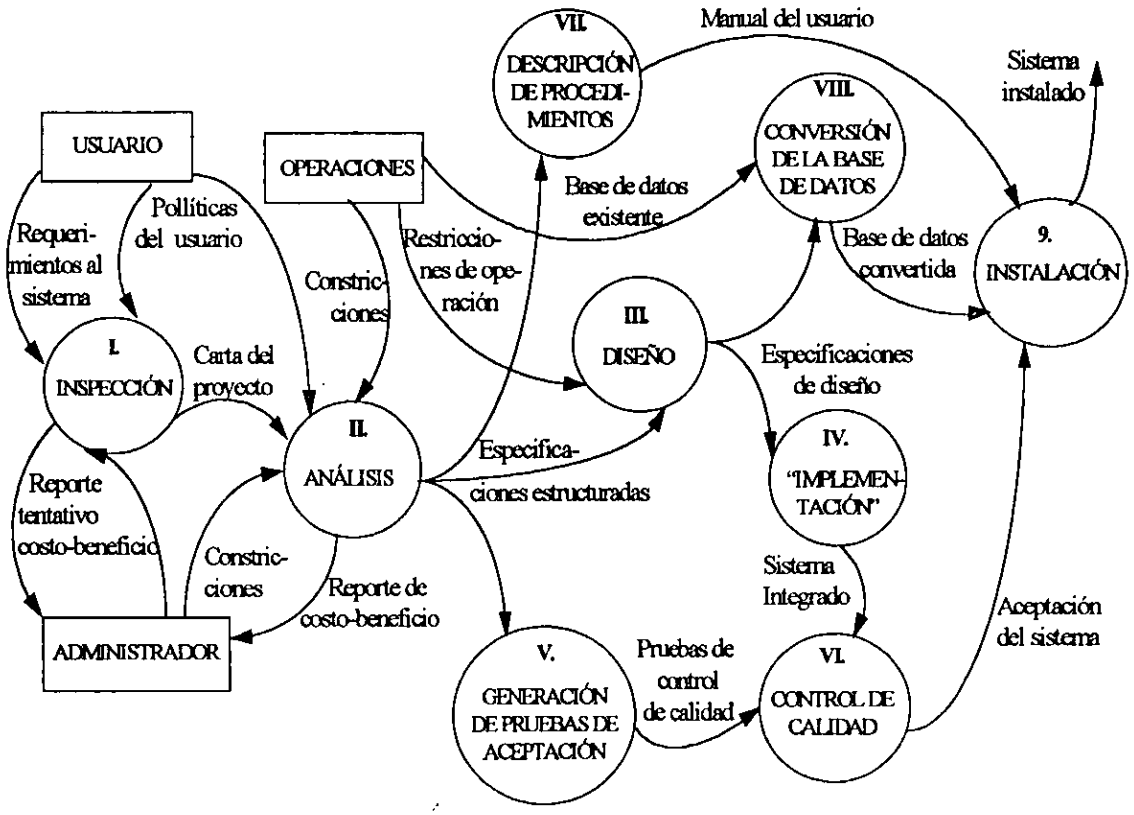


FIGURA 7: CICLO DE VIDA DEL SISTEMA

4.2.1. Explicación genérica de la metodología ('terminadores' y 'burbujas')

Como puede observarse, el DFD anterior contempla nueve 'actividades' y tres 'terminadores': **usuarios, operaciones y administrador**; estas tres instancias son, de una forma o de otra, los destinatarios últimos del sistema y proveen entradas de información al grupo de personas que desarrollan el proyecto:

* El usuario tiene una característica que lo hace distinguible de cualquier otro involucrado en el desarrollo de un sistema: es el único que puede aceptarlo cuando esté terminado, es decir, el único que puede decidir si es conveniente integrarlo en su operación cotidiana.

* El administrador es una autoridad de la organización donde se pretende desarrollar el proyecto, que detecta las constricciones de la organización para el sistema y define sus restricciones[§]; puede ser que el usuario y el administrador, en ciertas organizaciones, sean la misma instancia.

* El concepto de operaciones corresponde a la instancia responsable de la operación cotidiana de *hardware* y *software* en la organización y que derivado de lo mismo, conoce las constricciones "ambientales" en la materia para el nuevo sistema (*e.g.* limitaciones de *hardware*, complejidad de las **'facilidades'** de comunicación).

Las nueve 'burbujas' del DFD pueden explotarse para un mayor entendimiento en términos de otras, pero esto se hará más adelante (en el punto 4.2.2); por otro lado, cabe mencionar que en el método de Yourdon se emplea el término 'actividad' en vez de fase, pues este último, de suyo, da la idea de ejecución lineal secuencial. En este contexto, enseguida se comentan brevemente las 'actividades' del método en cuestión.

[§] En este contexto lo que diferencia a una constricción de una restricción es que mientras las primeras, consideradas como estrecheces se pueden modificar en el transcurso del proyecto, las segundas son inamovibles, incluso después de concluido el proyecto como tal.

I. Inspección

A esta 'actividad' también se le conoce como **estudio de factibilidad** y generalmente tiene su origen en peticiones de personas para automatizar o modernizar parte de sus funciones operativas; consiste de cuatro **'subactividades'**: identificar deficiencias del entorno actual, establecer nuevas metas, sugerir **escenarios de aceptación** y determinar si es factible modernizar y/o automatizar procesos organizacionales. Como producto de estas 'subactividades' se debe preparar la **carta del proyecto** (ver punto I.4).

II. Análisis

Pretende la transformación de las políticas del usuario y de la carta del proyecto en un conjunto de **'especificaciones estructuradas'** para modelar el sistema, mediante el uso de herramientas gráficas. Esta 'actividad' contempla siete 'subactividades': desarrollar **modelo ambiental**, desarrollar **modelo de comportamiento**, establecer **frontera hombre/máquina**, efectuar análisis de costo/beneficio, seleccionar la opción, generar el **sistema restringido** y empacar las especificaciones.

III. Diseño

Concierne a la determinación y ubicación de porciones de las especificaciones del análisis, para realizar una asignación apropiada de 'tareas' a los diversos elementos del sistema (*e.g.:* personas, **'procesadores'**, **'programas'** de cómputo, **'servidores'**); particularmente, para cada 'tarea' detectada, se pretende establecer en forma *top-down*, una jerarquía de **'módulos'**** de 'programación' e 'interfaces' entre esos 'módulos' a fin de poder 'implementar' apropiadamente las especificaciones del análisis. Al efecto se contemplan siete 'subactividades': ubicar especificaciones en 'procesadores', ubicar especificaciones en 'tareas', derivar **'cartas estructuradas'**, evaluar 'cartas estructuradas', diseñar 'módulos', diseñar 'base de datos' y empacar el diseño.

** Brevemente, por módulo se entiende un grupo de líneas de programas de cómputo que ejecutan 'tareas' simples de forma bien definida, que están bien acotados y, de alguna forma, son "contiguos" y tienen un nombre simple por el cual pueden ser referenciados como una unidad.

IV. 'Implementación'

Contempla tanto la **'codificación'** como la integración de los 'módulos' en un **'esqueleto'** del sistema esperado, siendo detallado progresivamente hacia niveles de mayor complejidad. En esta 'actividad', característicamente se emplea 'programación estructurada' e 'implementación' *top-down*. Se compone de tres 'subactividades': seleccionar 'módulo' siguiente, 'codificar' el 'módulo' y probar el 'esqueleto' del sistema.

V. Generación de pruebas de aceptación

Se deben efectuar con base en el conjunto de 'especificaciones estructuradas' del sistema, mismas que son la base para su 'implementación'. Se conforma por cinco 'subactividades': generar el plan de pruebas, preparar pruebas de **desempeño**, preparar pruebas de **trayectoria normal**, preparar pruebas de **trayectoria errónea** y empacar pruebas.

VI. Control de calidad

Requiere para su aplicación, de la generación de las pruebas de aceptación antes mencionadas y del sistema integrado en la 'actividad' de 'implementación'. Debe considerase como el seguro más completo para poder "aterrizar" el sistema lo mejor posible.

VII. Descripción de procedimientos

Consiste en la generación de una descripción formal para el uso del nuevo sistema (para su entrega al usuario), contemplando tanto de sus partes automatizadas como de aquéllas de ejecución manual. Un resultado particular de esta actividad es el **'manual del usuario'**.

VIII. Conversión de 'bases de datos'

Puede ser que exista una 'base de datos' en operación o, en su defecto, que deba generarse a efectos de **'implantación'** del sistema; por otro lado, hay casos en que la sola conversión de datos implica recursos tanto o más grandes que los necesarios para el desarrollo e 'implantación' del propio sistema; en todo caso, se requieren al efecto las especificaciones correspondientes derivadas del diseño.

IX. Instalación

Esta 'actividad' requiere como entradas, algunas de las salidas de las 'actividades' de descripción de procedimientos -*v.g.* 'manual del usuario'- y del control de calidad, que implica la entrega del sistema toda vez aplicado el control mencionado y, de la conversión de 'base de datos'. A saber hay dos extremos en la forma de 'implantar' sistemas: ***'big-bang'*** o gradual; entre dichos extremos hay una gama de posibilidades para la **administración del cambio**.

4.2.2. Explicación detallada de la metodología (explosión de 'burbujas')

A continuación se detallan las 'actividades' de la metodología de Yourdon mediante la 'explosión' de cada una de sus nueve 'burbujas', estableciendo que el método en cuestión se explica precisamente en función de diagramas de flujo de datos, diferentes de los 'diagramas de flujo', en razón de que en los primeros no hay implicación alguna de que la 'actividad' *n-ésima* deba concluir antes de iniciar la *n+1-ésima* (a diferencia de los otros); por el contrario, de la red del flujo de datos representada en los DFD, se deriva que diversas 'actividades' pueden ejecutarse en paralelo; en ese sentido, se debe hacer notar que los diagramas de flujo de datos sólo muestran las entradas o "insumos" requeridos por cada 'actividad' y las salidas ahí producidas; asimismo, los mencionados DFD no muestran explícitamente retroalimentación (*feedback*) ni control sobre las 'burbujas', sin embargo y de hecho, información producida en diferentes 'actividades' puede impactar y retroalimentar a otras. Como ejemplo: por cuestiones de diseño pueden cambiar las especificaciones del equipamiento -lo que impactará la estimación de costos-, cambiando incluso decisiones tempranas del proyecto -*v.g.* su alcance-. Por otro lado, cabe aclarar que aunque en la metodología sólo se indica explícitamente la participación del administrador en la 'actividad' de análisis, ciertamente ejerce control sobre todas las que componen el proyecto.

I. Inspección

Objetivo: Dotar a los tomadores de decisiones con un documento de **factibilidad del proyecto**, para dimensionar su alcance *y* decidir su continuidad. El detalle de las 'subactividades' se presentan en el siguiente DFD (figura 8):

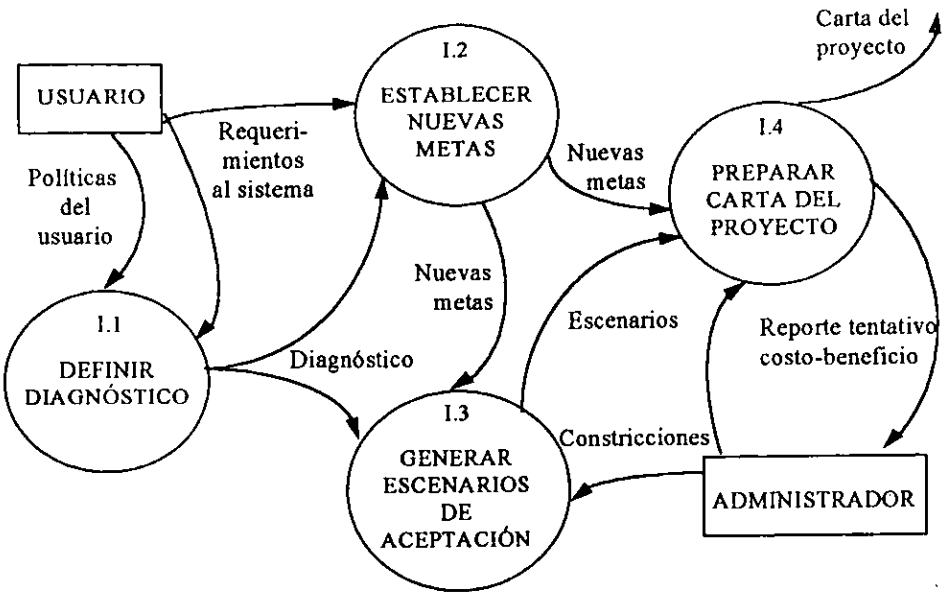


FIGURA 8: DFD DE INSPECCIÓN

Tareas:

I.1 Definir **diagnóstico** o problemática y deficiencias actuales del entorno

* Insumo(s):

** **Requerimientos al sistema:** Lista inestructurada y exhaustiva con las características de operación y funcionalidad requerida al sistema.

** **Políticas del usuario:** Establecidas previamente o al efecto en documentos -por escrito-.

* Salida(s):

** **Diagnóstico:** Lista con deficiencias en el entorno del usuario, exhaustiva y priorizada, clasificada por orden de importancia (*e.g.:* costos excesivos, funciones no cubiertas, problemas de control y operación, información poco veraz).

I.2 Establecer **nuevas metas** (para el sistema propuesto)

* Insumo(s):

** Requerimientos al sistema.

** Diagnóstico.

* Salida(s):

** **Nuevas metas:** Frases con objetivos concisos de 'funcionalidad' y desempeño, basados en: Requerimientos al sistema + Respuestas esperadas del sistema + Limitaciones de presupuesto + Aseguramiento de la confiabilidad de requerimientos.

I.3 Generar escenarios de aceptación

* Insumo(s):

** Diagnóstico.

** Nuevas metas.

** **Constricciones:** Conjunto de limitaciones proporcionadas por el administrador en materia de: Tiempo + Presupuesto + Operacionales + Recursos humanos asignados al proyecto. Las constricciones se pueden modificar en el transcurso del tiempo.

* Salida(s):

** **Escenarios de aceptación:** Resúmenes preliminares con la propuesta de funcionalidad de sistemas posibles (preferentemente deben presentarse en formato de DFD) + Costos/beneficios asociados.

I.4 Preparar la carta del proyecto, con base en la cual se genera el documento formal de factibilidad del mismo, contemplando su objetivo definitivo

* Insumo(s):

** Constricciones.

** Nuevas metas.

** Escenarios de aceptación.

* Salida(s):

** **Reporte tentativo de costo/beneficio:** Costos asociados con un escenario de aceptación + Beneficios asociados (*v.g.* financieros).

** **Carta del proyecto:** Documento integrado con la información de: **Planteamiento del(los) objetivo(s)** + **Extracto del proyecto** + **"Ajuste a la medida"** del ciclo de vida del proyecto + Constricciones de: tiempo, técnicas y operacionales + Escenarios de aceptación.

*** **Planteamiento del(los) objetivo(s):** Recopilación de frases concisas con: la 'funcionalidad' por ser 'implementada' en el sistema + Deficiencias por ser remediadas + Características por ser modificadas.

*** **Extracto del proyecto:** Nombre del proyecto + Redacción breve con su misión o propósito genérico + Delimitación del (las) área(s) organizacional(es) a ser comprendida(s) + Fecha de inicio + Fecha de entrega + Asignación de presupuesto original + Nombre del responsable por parte de los usuarios + Nombre del líder de proyecto.

*** **"Ajuste a la medida" del ciclo de vida del proyecto *(customize):*** DFD con el cíclo de vida ajustado a la medida por seguir en el proyecto, con los **'entregables'** del mismo por 'actividad' (*v.g.* DFD derivados del análisis, DD, diseño del 'esqueleto' del sistema). Por "ajustar a la medida" se entiende el hecho de fusionar y/o obviar o incluso, traslapar 'actividades' o 'burbujas' a los DFD originalmente propuestos por Yourdon (*e.g.* incluir en la 'actividad' del análisis la de inspección). Al integrar fechas de entrega con las limitaciones del punto anterior en los DFD ajustados a la medida -y al efecto-, a través de una serie de anotaciones al mismo, es posible generar un plan de trabajo detallado en formato Gant o Pert.

*** Constricciones impuestas por el administrador en materia de: Tiempo + Técnicas + Operación.

Puntos finos de la inspección:

* Definición precisa del propósito del sistema.
* Acotamiento del alcance del estudio de factibilidad (hasta donde "se cubre").
* Detección de causas en vez de síntomas.
* Interacción entre administradores y usuarios, así como negociación y acuerdo de tiempos y presupuestos.
* Entrevistas con usuarios apropiados.
* Revisión de escenarios y costos/beneficios; usualmente genera retroalimentación en el sentido de volver a revisar y, en su caso, modificar estrecheces del proyecto.

* La carta del proyecto es su referencia primordial, desde su generación hasta su conclusión, independientemente del tiempo y de los cambios de su entorno.

II. Análisis

Objetivo: Producir las 'especificaciones estructuradas' o "modelo en papel" del sistema, a través de: un conjunto de DFD nivelados apropiadamente; el DD; un conjunto de ERD; 'miniespecificacione de proceso' para todas las funciones requeridas y la descripción concreta de las restricciones impuestas al sistema.

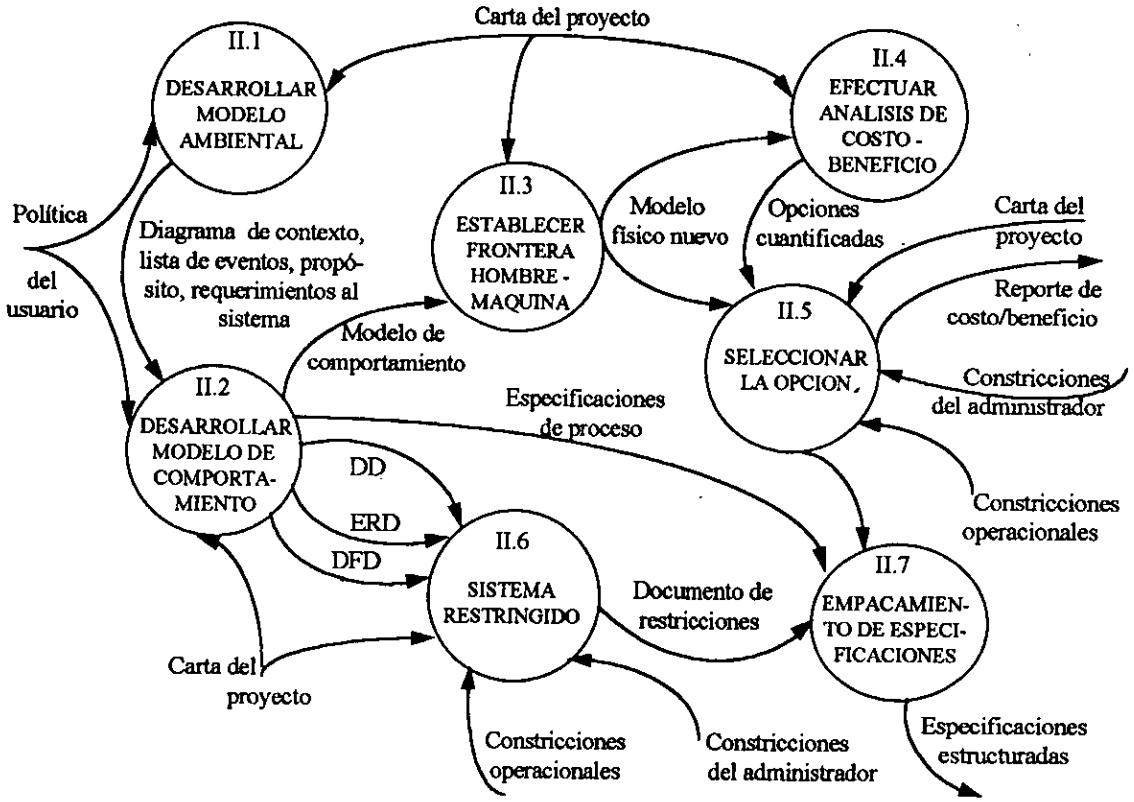


FIGURA 9: DFD DE ANÁLISIS

Tareas:

II.1 Desarrollar modelo ambiental

* Insumo(s):

** Carta del proyecto.

** Políticas del usuario.

* Salida(s):

** **Modelo ambiental:** Documento que muestra cómo debe interactuar el sistema con su medio ambiente, a través de: Propósito del sistema + **Diagrama de contexto del sistema** + **Lista de eventos**.

*** **Diagrama de contexto del sistema:** DFD correspondiente al nivel 0, donde se delimitan las fronteras entre el sistema y su entorno (*e.g.*: 'terminadores', otros sistemas), así como los datos que deben ser **'procesados'** o producidos por el mismo y, aquéllos que deben ser generados o consumidos por 'entidades' externas.

*** **Lista de eventos:** Relación exhaustiva de estímulos del medio ambiente -externos al sistema-, ante los cuales debe responder, donde se debe indicar la persona, instancia o proceso que dan origen a esos eventos.

II.2 Desarrollar **modelo de comportamiento**

* Insumo(s):

** Carta del proyecto.

** Política del usuario.

** Diagrama de contexto.

** Lista de eventos.

** Requerimientos al sistema.

* Salida(s):

** **Modelo de comportamiento:** Documento integrado y sancionado por el administrador: muestra el qué debe hacer el sistema y el cómo debe interactuar adecuadamente con su medio ambiente, con base en: DD + Conjunto de ERD + Conjunto de DFD + 'Miniespecificaciones de proceso' -entre otros insumos-.

II.3 Establecer frontera hombre-máquina (y máquina-máquina)

* Insumo(s):

** Carta del proyecto: Con base en los escenarios ahí propuestos, el analista debe ratificar o rectificar, los **modelos físicos nuevos.** En su caso, debe proponer opcionalmente otros modelos.

** Establecer -por parte del administrador-, las tareas manuales del sistema que debe realizar el área organizacional involucrada y las funciones que debe ejecutar el sistema en forma semi o totalmente automatizada.

* Salida(s):

** **Modelos físicos nuevos:** especifican las 'tareas' automatizadas y aquéllas semiautomatizadas en el(los) sistema(s) propuesto(s), con base en: DD + Conjunto de DFD nivelados.

II.4 Efectuar análisis de costo/beneficio

* Insumo(s):

** Constricciones operacionales y aquéllas impuestas por el administrador -plasmadas en la carta del proyecto-.

** Modelos físicos nuevos.

* Salida(s):

** **Opciones cuantificadas:** Modelos físicos nuevos + Costos asociados estimados de: desarrollo, operación, riesgos, etc. + Beneficios asociados.

II.5 Seleccionar la opción

* Insumo(s):

** Carta del proyecto, incluyendo las constricciones operacionales y las impuestas por el administrador.

** Modelos físicos nuevos.

** Opciones cuantificadas.

* Salida(s):

** Reporte de costo/beneficio: Costos y beneficios asociados al modelo físico seleccionado.

** **Modelo físico seleccionado:** Conjunto de DFD nivelados + DD (separando las 'tareas' del sistema de las ejecutadas por personas).

II.6 Sistema restringido

* Insumo(s):

** Carta del proyecto -que debe incluir constricciones en general-.

** Modelos físicos nuevos.

* Salida(s):

** **Sistema restringido:** Documento con las restricciones impuestas al sistema definitivamente por el administrador.

II.7 Empacamiento de especificaciones

* Insumo(s):

** Especificaciones y 'miniespecificaciones de proceso' derivadas del modelo de comportamiento.

** Sistema restringido.

** Modelo físico seleccionado.

* Salida(s):

** **Especificaciones empacadas**: Documento integrado coherentemente (índice, introducción, narrativa breve, etc.) con las salidas de las 'actividades' precedentes consistente en: Conjunto de DFD nivelados + DD + Especificaciones y 'miniespecificaciones de proceso' + Sistema restringido

Puntos finos del análisis:

* La conjunción de las tareas 2.1 y 2.2 da origen al denominado **'modelo esencial del sistema'**: qué y cómo debe proceder el mismo. En otros lugares a este modelo se le

denomina **'modelo lógico'** -evitado aquí en razón de que, para el usuario, "no hay modelos ilógicos"-.

* El *quid* del asunto del análisis es resolver el manejo del desarrollo del sistema, a través de un empleo apropiado de sus "curvas de nivel", *i.e.*: de la nivelación propuesta y modificada -por el administrador- de sus DFD y, de como poder navegar entre éstos -a través de los compromisos adquiridos- y, de las limitaciones del medio ambiente; en otras palabras: la manipulación de los riesgos para el sistema y del presupuesto, así como del personal involucrado (¿hasta dónde se llega?) entre otros:

** parálisis: por ejemplo, una vez establecido el 'diagrama de contexto', ¿cómo poder disgregarlo en términos de otras 'burbujas' -y cuántas más deben ser-?.

** sesgo físico: el hecho de concebir la 'explosión' de las 'burbujas' del sistema, en razón de la estructura organizacional actual subyacente al mismo o, de las aparentes 'tareas' en términos de las cuáles debería descomponerse el sistema; por ejemplo, opiniones del tipo: *"la función operativa de cálculo y seguimiento del área organizacional involucrada, la debe ejecutar un 'programa' de cómputo, en sustitución del área que la ejecuta -hoy-"*.

** el síndrome que denominaría del *"octavo pasajero"*; muy simple de comprender con esto: si la disponibilidad de analistas para el caso es de siete, pero se presume que existe un octavo pasajero, *v.g.*: un recurso humano adicional, ¡las 'burbujas' del sistema deben ser ocho y no siete -cuando en realidad..., es suficiente con cinco!-. Al efecto, se sugiere partir de la lista de eventos -tal cual se obtiene-, para que el administrador se encargue de realizar el primer corte de las mismas; como resultado de la interacción entre él y "sus" analistas, se genera a su vez un interesante proceso de retroalimentación entre el usuario y el citado administrador, que generalmente produce un plan de trabajo más realista.

* Se debe hacer hincapie en que el modelo físico nuevo no implica, todavía hasta este punto, ninguna decisión acerca del *hardware* indispensable para el proyecto, empero, la carta del proyecto debe asentar de alguna manera y en lo posible, las premisas acerca del mismo -*v.g.* *"el sistema corre en equipos PC con un mínimo de 16 MB en RAM y reloj de 100 Mhz"*-.

* Las opciones cuantificadas deben tener el detalle de información suficiente para que el administrador pueda discernir entre una opción y otra, a fin de seleccionar la idónea.
* La selección de la opción implica un conjunto de aspectos que trascienden a un simple cálculo de costos, tales como: consideraciones más complejas de manejo de impuestos, descuentos, tiempos de entrega, políticas impositiva generales, etc.
* Ejemplos típicos de restricciones físicas impuestas definitivamente al sistema restringido pueden ser:

** *"No hay nuevo hardware para el proyecto".*
** *"El hardware se comprará al vendedor X y el software al Y".*
** *"De ninguna forma se comprará equipo al proveedor Z".*
** *"El tiempo de respuesta por consulta debe ser menor a tres segundos".*
** *"El tiempo medio de falla del sistema debe ser de tres meses".*
** *"El tiempo de recuperación del sistema en caso de caída debe ser de 15 minutos máximo".*
** *"El hardware no debe requerir equipo ambiental adicional, tal como aire acondicionado".*

* La integración coherente de las especificaciones empacadas requieren de un esfuerzo adicional en términos del cuidado de la calidad del documento respectivo a fin de evitar DFD con 'burbujas' que tienen un sinnúmero de entradas -pero ninguna salida-; 'bases de datos' que son leidas -pero nunca actualizadas-; DFD derivados de otros con almacenes de datos -que no existen en el DFD precedente-; datos no definidos en ninguna parte; etc.

III. Diseño

Objetivo: Generar las especificaciones de diseño del sistema para su 'implementación' -ver en la página siguiente la figura 10-.

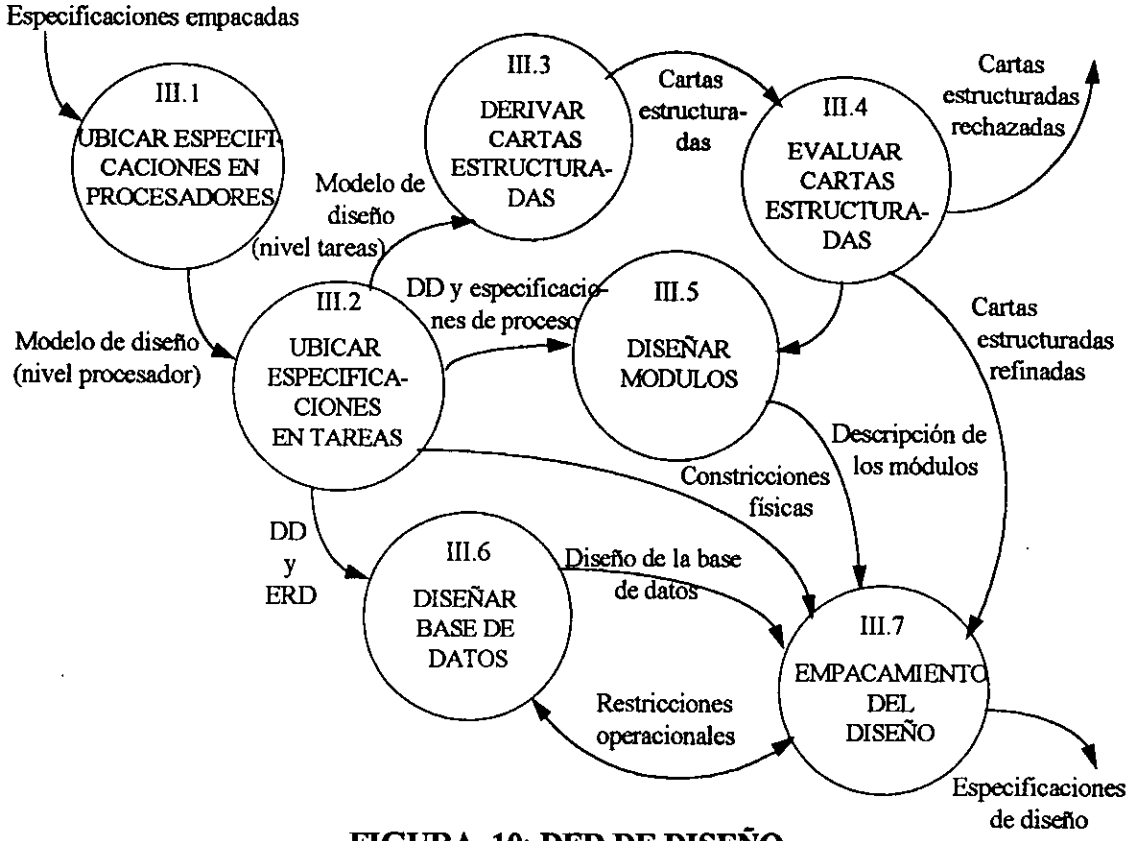


FIGURA 10: DFD DE DISEÑO

Tareas:

III.1 Ubicar especificaciones en 'procesadores'

* Insumo(s):

** Especificaciones empacadas.

* Salida(s):

** **Modelo de diseño (nivel 'procesador'):** Definición de si el sistema será **'local'** o ***'standalone'***, **'multiusuario'**, etc. Consiste básicamente en decidir la ubicación de trozos de los diagramas generados en el análisis para el modelo de comportamiento (DFD y *data stores*), en él o los 'procesadores' seleccionados a efectos de 'implementación' del sistema, como se ejemplifica en la figura 11; note que ahí se supone que los 'procesadores' 1 y 2 tienen asignada su propia **'facilidad'** de almacenamiento de datos -sea una cinta o un ***'drive'*** de disco-, aunque puede ser que

el manejo de los *data stores* se asignen a otro 'procesador', tal como el de un **'servidor de disco de una red'**.

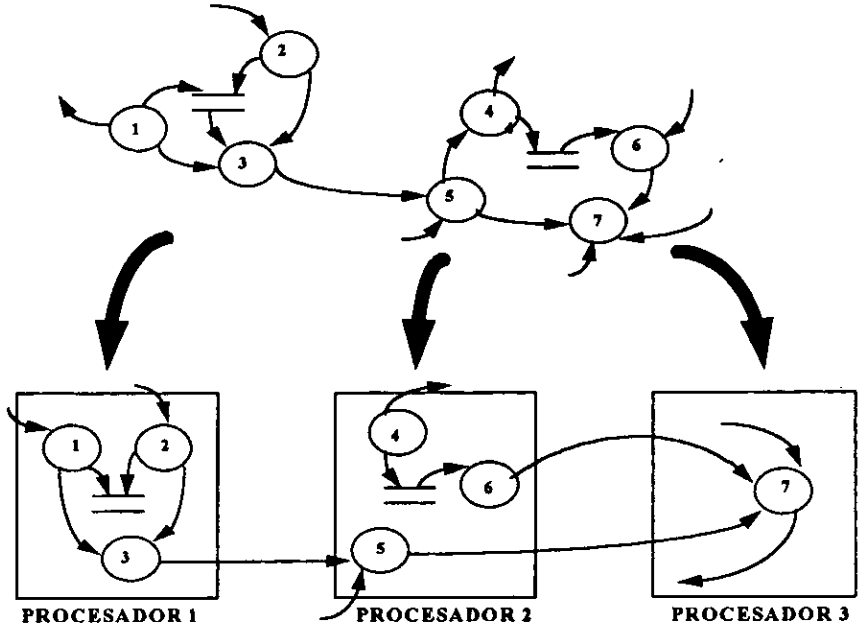


FIGURA 11: UBICAR ESPECIFICACIONES EN PROCESADORES

III.2 Ubicar especificaciones en 'tareas'

* Insumo(s):

** Modelo de diseño (nivel 'procesador').

* Salida(s):

** **Modelo de diseño (nivel 'tareas'):** Consiste básicamente en decidir la ubicación de trozos de los diagramas generados en el análisis para el modelo de comportamiento, tanto DFD como *data stores*, en una o más 'tareas' asignadas a los 'procesadores' seleccionados en 3.1, para la 'implementación' del sistema.

III.3 Derivar 'cartas estructuradas'

* Insumo(s):

** Especificaciones empacadas.

** Modelo de diseño (nivel 'procesador' y 'tareas').

* Salida(s):

** **'Cartas estructuradas':** Representación gráfica de 'módulos' jerarquizados correspondientes a los DFD nivelados, que indican entre otros aspectos cuál precede a cuál, cuál depende de cuál, incluyendo documentación de interfaces entre los 'módulos', con referencia al modelo de diseño (nivel 'procesador' y 'tareas'), donde la jerarquización en principio estará dada por la nivelación de los propios DFD.

III.4 Evaluar 'cartas estructuradas'

* Insumo(s):

** 'Cartas estructuradas'.

* Salida(s):

** 'Cartas estructuradas' rechazadas.

** **'Carta estructurada' refinada:** Se refiere al ajuste en por lo menos una vez a la original, con relación a cuántos 'módulos' más controla o, al propio "tamaño" de cada 'módulo'.

III.5 Diseñar 'módulos'

* Insumo(s):

** Especificaciones empacadas.

** 'Carta estructurada' refinada

* Salida(s):

** **Descripción de los 'módulos'**: Descripción detallada con la lógica de los procedimientos de cómputo de cada 'módulo', expresados en términos de DFD, 'miniespecificaciones de proceso' o incluso en 'diagramas de flujo', cuya transcripción en lógica de algoritmos a menudo resulta no trivial y que contendrá información y notación adicional que no está en la especificación original (*v.g.* **EOF**)

III.6 Diseñar la 'base de datos'

* Insumo(s):

** Especificaciones empacadas (incluye DD, ERD y DFD).

** Restricciones operacionales.

* Salida(s):

** **Diseño o 'esquema de la base de datos':** Diagrama con la estructura física de las 'tablas' de la 'base de datos', sus 'relaciones', etc. En su caso, es conveniente añadir borradores de otros 'programas' adicionales que tal vez sean requeridos por la aplicación en documentación anexa al 'esquema' (*v.g.* 'programas' del 'servidor').

III.7 Empacamiento del diseño

* Insumo(s):

** 'Carta estructurada' refinada.

** Descripción de los 'módulos'.

** Diseño o 'esquema' de la 'base de datos'.

** Constricciones físicas.

** Restricciones operacionales.

* Salida(s):

** **Especificaciones de diseño:** Documento integrado coherentemente con las salidas de las 'actividades' y ' 'tareas' precedentes.+ Actualización de consideraciones de *hardware*, **'configuración'**, etc.

Puntos finos del diseño:

* En algunos casos la comunicación entre 'procesadores' suele ser lenta y cara, así como también suele serlo la intercomunicación de 'procesos' -pero de forma relativa-; por lo anterior, el diseñador debe balancear la interacción entre 'procesadores' y 'tareas', buscando un justo balance en el desempeño o ***'performance'*** del sistema, después del mayor número posible de pruebas en la materia.

* Con relación a los 'módulos', hay dos conceptos que deben evitar su aplicación extremosa sobre ellos -en lo posible-:

** **'acoplamiento'**: "fuerza" de interconexión entre un 'módulo' y otro, donde, a mayor 'acoplamiento', mayor dependencia entre ellos, con lo que el impacto de hacer modifcacione a un 'módulo', impactará definitivamente en el otro.

** **'cohesión'**: "fuerza" de interconexión entre los elementos propios de un 'módulo'; un caso concreto son aquéllas **'rutinas'** que al ser entremezcladas con otras, pierden la claridez de las 'tareas' que cumpen originalmente de forma simple y bien definida.

De esta forma, 'módulos' con baja 'cohesión' tenderán a un fuerte 'acoplamiento' y aquéllos que tengan fuerte 'cohesión' funcional en cada nivel de la jerarquía, tenderán a que su 'acoplamiento' sea menor.

* La especificación detallada de los 'módulos' debe ser suficiente para su 'implementación', pero sin que llegue a representar costos excesivos (*v.g.* tiempos); su extensión y detalle es una decisión del administrador. Es indispensable verificar la consistencia entre este documento con las 'especificaciones estructuradas' del análisis y la lista de requerimientos original, a fin de estar completamente seguro de que el sistema por 'implementar' sea el que espera el usuario.

IV. 'Implementación'

Objetivo: Definir el 'esqueleto' del sistema y generar e integrar el 'código' de los 'módulos' al mismo.

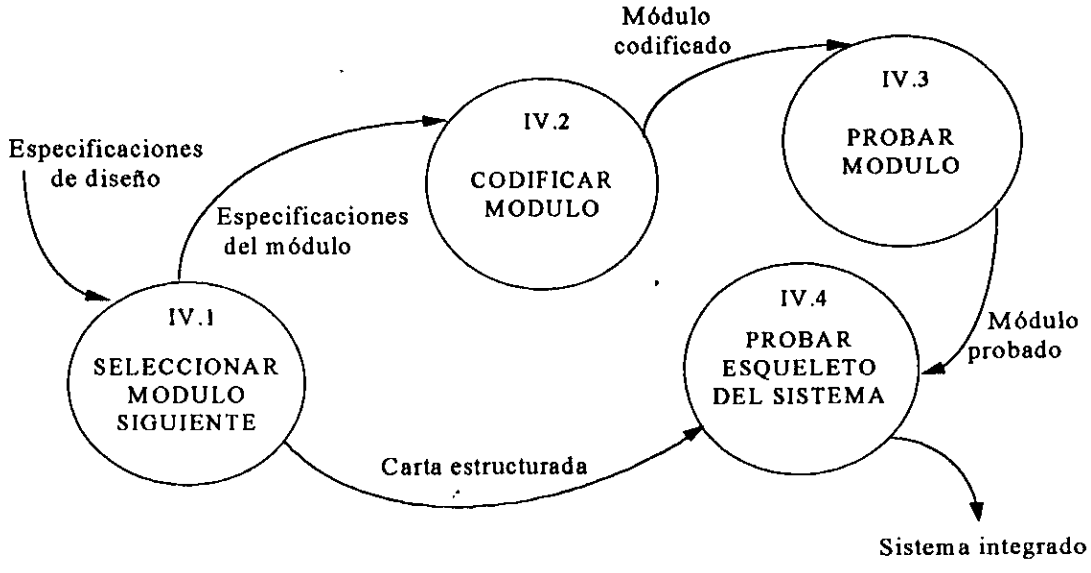


FIGURA 12: DFD DE "IMPLEMENTACIÓN"

Tareas:

IV.1 Seleccionar 'módulo' siguiente

* Insumo(s):

** Especificaciones de diseño.

* Salida(s):

** Especificaciones de diseño del 'módulo' seleccionado.

IV.2 'Codificar' 'módulo'

* Insumo(s):

** Especificaciones de diseño del 'módulo' seleccionado.

* Salida(s):

** 'Módulo' 'codificado'.

IV.3 Probar 'módulo'

* Insumo(s):

** 'Módulo' 'codificado'.

* Salida(s):

** 'Módulo' probado.

IV.4 Probar el 'esqueleto' del sistema

* Insumo(s):

** 'Carta estructurada' refinada.

** Especificaciones de diseño del 'módulo' seleccionado.

* Salida(s):

** Sistema integrado.

Puntos finos de 'implementación':

* El aspecto principal de estas 'subactividades' está en la decisión sobre la secuenciación en que los 'módulos' serán 'implementados', integrados -al 'esqueleto' del sistema- y probados, donde 'módulos' de mayor jerarquía serán desarrollados antes que otros de jerarquía menor.

Justamente en este punto es donde es posible evaluar las bondades y restricciones de alcance de las herramientas de desarrollo que se aplican en el proyecto, así como la capacidad del *hardware* empleado en el mismo.

* Las pruebas a que se hace referencia en el punto anterior se pueden denominar como "internas y locales" al grupo de desarrollo. Paralelamente a estas pruebas, debe revisarse el apego del grupo de trabajo a las especificaciones de diseño.

V. Generación de pruebas de aceptación

Objetivo: Generar las pruebas al sistema para su aceptación y aplicación posterior del control de calidad.

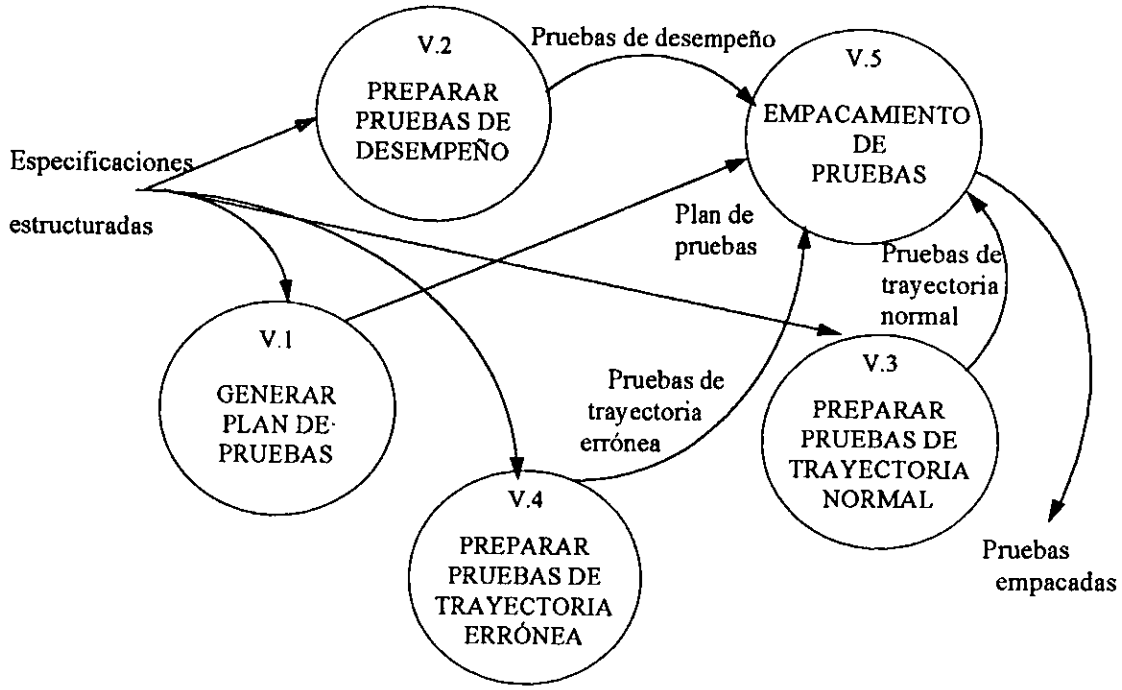


FIGURA 13: DFD DE PRUEBAS DE ACEPTACIÓN

Tareas:

V.1 Generar plan de pruebas

* Insumo(s):

** 'Especificaciones estructuradas'.

* Salida(s):

** Plan de pruebas.

V.2 Preparar pruebas de desempeño

* Insumo(s):

** 'Especificaciones estructuradas'.

* Salida(s):

** **Pruebas de desempeño:** Conjunto de procedimientos descriptivos de cada prueba de rendimiento del sistema, incluyendo datos prueba y resultados esperados.

V.3 Generar pruebas de trayectoria normal

* Insumo(s):

** 'Especificaciones estructuradas'.

* Salida(s):

** **Pruebas de trayectoria normal:** Conjunto de procedimientos descriptivos de las pruebas para el 'procesamiento' correcto de datos, con: Datos correctos + Especificación de prueba de los datos + Resultados esperados.

V.4 Generar pruebas de trayectoria errónea

* Insumo(s):

** 'Especificaciones estructuradas'.

* Salida(s):

** **Pruebas de trayectoria errónea:** Conjunto de procedimientos descriptivos de prueba para funcionamiento correcto del sistema, con: Datos incorrectos + Especificación de prueba de los datos + Mensajes de error esperados o comportamientos alternativos del sistema.

V.5 Empacamiento de pruebas

* Insumo(s):

** Plan de pruebas.

** Pruebas de desempeño.

** Pruebas de trayectoria normal.

** Pruebas de trayectoria errónea.

* Salida(s):

** **Pruebas empacadas:** Documento integrado para efectos de entrega al área responsable del control de calidad.

Puntos finos de la generación de pruebas de aceptación:

* Se debe nombrar de antemano una persona o grupo de personas como responsables de la generación de las pruebas, sin relación alguna con el grupo de 'implementación', con la intención expresa de encontrar errores en el sistema.

* Todas las pruebas de aceptación deben ser derivadas de las especificaciones o, en otras palabras: *"The specification is the acceptance test " (sic)* -como escribió Tom De Marco-; así mismo, las pruebas deben tener un resultado binario: "pasa o no pasa". Si a pesar de aplicar con todo cuidado el control de calidad se llega a fallar con el proyecto, entonces se infiere que el análisis fue lo que efectivamente falló, pues de esta forma las especificaciones resultaron mal elaboradas desde un inicio.

* Se deben preparar datos prueba, conforme a 'estándares' de control establecidos al efecto, tales como cotejo de resultados contra sistemas preexistentes -de ser posible-, intervalos de confianza, cifras de control esperadas, etc.

* Muchas de las pruebas de desempeño deben orientarse a la capacidad del *software* de la 'base de datos' y de las comunicaciones, simulando **'eventos'** críticos en materia de cómputo como exceso de tráfico en la red (***'overhead'***) o crecimiento explosivo de **'registros'** en la 'base de datos'. Una alternativa es restringir el ambiente del sistema, en el sentido de ejecutarlo con *hardware* de menor capacidad al requerido.

VI. Control de calidad

Objetivo: Lograr la aceptación del sistema, previa verificación al efecto de que el sistema hace lo que se especificó -ni más, ni menos-.

Tareas: Ejecutar las pruebas planteadas en la 'actividad' V (ver figura14, página 39).

* Insumo(s):

** Pruebas empacadas.

** Sistema integrado.

* Salida(s):

** Sistema aceptado

Puntos finos del control de calidad:

* Recordando que no es justo "ser juez y parte", esta 'actividad' debe efectuarse con un punto de vista optimista por parte de un grupo creado al efecto, para la entrega de los resultados de las pruebas. Preferentemente los resultados deben ser entregados y evaluados por una instancia colegiada superior o a un comité con poder de decisión en la materia.

* Es posible adicionar al control de calidad, revisiones al sistema en materia de:

** su documentación y 'entregables',

** su **'código fuente'**, para verificar entre otros aspectos cumplimiento de 'estándares',

** su seguridad y auditabilidad,

etc., aunque, en todo caso, como se ha citado: *"La especificación es la prueba de aceptación"*, y no otro(s) aspecto(s).

* Sugerencias para alcanzar pruebas completas a un sistema son:

** Cada 'módulo' debe ser invocado y revisado al menos una vez, aunque no todas las combinaciones viables de los 'módulos' hayan sido ejecutadas.

** Cada decisión (*If.. Then.. Else*) debe verificarse, aunque no todas las combinaciones de las decisiones se ejecuten y verifiquen.

** Aplicar métodos exhaustivos de depuración (***'debug'***) a aquéllas partes obscuras del sistema a fin de esclarecerlas en su documentación y en su 'código'.

** Diferencias entre resultados esperados y obtenidos, deben revisarse exhaustivamente para proceder en consecuencia (correción de 'código', diseño o, en el peor de los casos, de análisis).

** Probar, una por una, la validación de datos de entrada contra el DD (*v.g.* 'rangos' de validez, "tamaño" o longitud de los datos), así como la especificación de salida de los mismos (*e.g.* impresión de cantidades numéricas en documentos tales como cheques). Un ejemplo trivial pero que ilustra es el siguiente:

Supóngase como rango válido de datos 1..20:

*** Por prueba de trayectoria normal se entiende ingresar al sistema números como 1, 5, 10 o 20.

*** Por prueba de trayectoria errónea se comprende ingresar al sistema números como -1, 0, 100 o 200 y..., caracteres como #%&$?!@!!.

VII. Descripción de procedimientos

Objetivo: Traducir las especificaciones de los analistas y diseñadores en un 'manual del usuario' (ver figura 14, página siguiente).

Tareas: Desarrollar el manual del usuario

* Insumo(s):

** 'Especificaciones estructuradas' de análisis y diseño.

* Salida(s):

** Manual del usuario

Puntos finos de la descripción de procedimientos:

* Una forma conveniente para desarrollar el manual del caso, es hacerlo en sincronía con la 'implementación' del sistema, persiguiendo retroalimentación continua con los usuarios. En los mejores casos es posible involucrar tanto al usuario con su sistema que, inluso, llega a participar en su elaboración elevando invariablemente en estos casos la calidad original del documento respectivo.

* Debe considerarse que en México para cuestiones de derechos de autor, uno de los elementos indispensables para el registro del *software* -actualmente-, lo constituye el documento en cuestión.

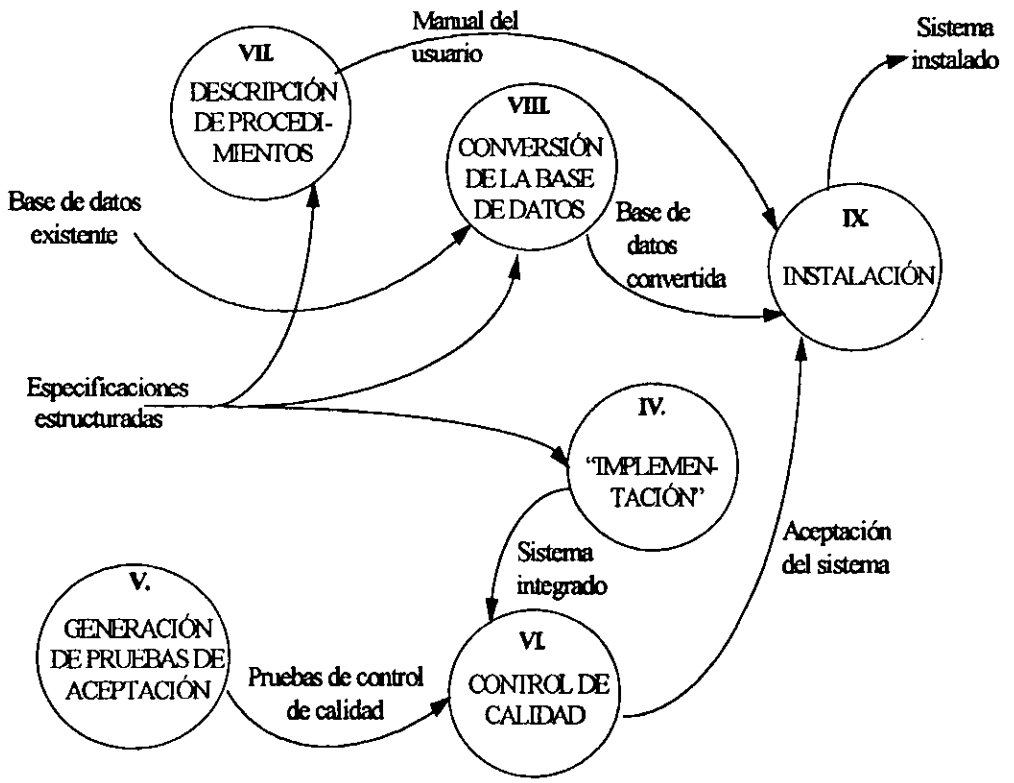


FIGURA 14: ACTIVIDADES IV A IX DEL CICLO DE VIDA DEL SISTEMA

VIII. Conversión de bases de datos

Objetivo: Contar con la 'base de datos' convertida para operar el sistema.

Tareas: Convertir la 'base de datos'.

* Insumo(s):

    ** 'Base de datos' existente.

* Salida(s):

    ** 'Base de datos' convertida.

Puntos finos de la conversión de bases de datos:

* Es posible que no haya datos por convertir, como en el caso de un nuevo sistema -que no sustituye a otro- o, a la inversa, que haya un sistema preexistente con 10'000,000 de 'registros' por convertir; en casos como el último debe darse la más alta prioridad al asunto e incluso, dimensionarlo tal vez, como otro proyecto paralelo al desarrollo del propio sistema, sobretodo cuando los datos son demasiados y poco confibles o, no están disponibles en medios magnéticos sino solamente en papel (*e.g.* archivos de tarjetas fisicas) ante lo cual, una posibilidad es generar los datos a través del propio sistema, una vez liberado.

IX. Instalación

Objetivo: Lograr que el usuario cuente con su sistema instalado (ver figura 14 -página 39-).

Tareas: Instalar el sistema.

* Insumo(s):
    ** 'Base de datos' convertida.
    ** Manual del usuario.
    ** Sistema aceptado.
* Salida(s):
    ** Sistema instalado.

Puntos finos de la instalación:

* Esta 'actividad' tiene dos extremos en la forma de realizarse: 'big-bang' o gradual; entre dichos extremos hay una gama de posibilidades para la administración del cambio, dependiendo invariablemente, del entorno y de la envergadura propia del sistema. A veces todo se reduce a instalar el sistema en un equipo PC y explicar brevemente las bondades del mismo a un par de usuarios; en otras debe realizarse todo un plan e incluso, una estrategia completa de 'implantación', con sus propios requerimientos y calendarios, conformando en si todo un proyecto completo de tamaño y costos equivalentes a los del desarrollo del propio sistema, al contemplar cuestiones de capacitación o infraestructura.

4.3 'Desarrollo rápido de aplicaciones'

4.3.1 Introducción

RAD (Rapid Application Development) significa en síntesis que una aplicación de cómputo se elabore y se logre ejecutar lo más rápido posible. Suelen usarse en este contexto las frases: *"diseñar un poco"*, *"'codificar' un poco"*, *"probar un poco"* (como pretende mostrarse en la figura 15); es un paradigma reciente que se popularizó posteriormente al uso de las herramientas de desarrollo denominadas **'Turbo'**, con la llegada comercial de las GUI, sobretodo con ciertos **'desarrolladores de aplicaciones'** como el empleado para 'programar' o 'codificar' el GAR, que conducen los esfuerzos correspondientes de 'implementación' a través del 'desarrollo iterativo': en una primera iteración, se crea un diseño de alto nivel que correspondería, al 'esqueleto' de sistema propuesto por Yourdon -en su metodología-; enseguida se elabora un 'prototipo' y se prueba. En la segunda iteración, se afina el diseño, se hace el 'prototipo' de las partes nuevas y se vuelve a probar. Estos ciclos de 'desarrollo iterativo' continúan hasta que la aplicación queda terminada y lista para distribuirse:

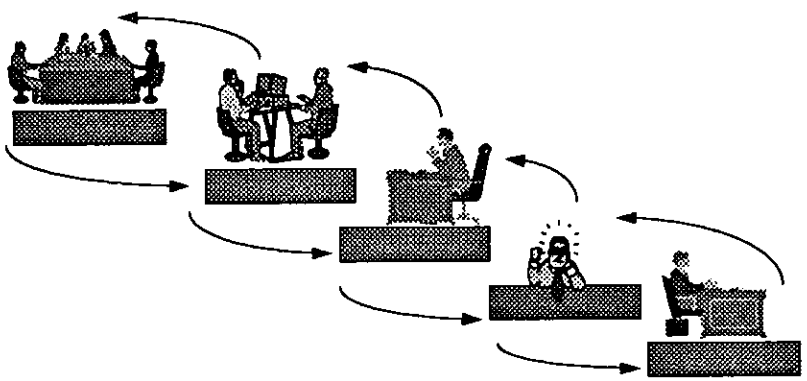


FIGURA 15: PROCESO DE DESARROLLO ITERATIVO O ESPIRAL

Las premisas del RAD recomiendan que los programadores trabajen directamente con el usuario a través de 'prototipos', planteando como idea principal demorar los detalles de la aplicación tanto como sea posible. Al efecto se debe considerar: diseñar la GUI, establecer las propiedades iniciales de los 'objetos' o **'componentes'** manipulados por un 'desarrollador de aplicaciones', crear y 'programar' los **'eventos'** necesarios de esos 'objetos' y luego, agregar características más avanzadas y 'módulos' de 'código'; así, al utilizar los principios RAD es posible afinar la 'interfaz' hasta que quede exactamente como la desea el usuario final; empero, entre otros de los retos por afrontar en estas condiciones, se debe convencer al usuario (sobretodo a aquéllos desesperados), de que el desarrollo no se acaba una vez que ha visto el 'prototipo' de la GUI de la aplicación y que, invariablemente, con su participación se logrará una mayor riqueza en la 'funcionalidad' de la misma.

Después, se debe comenzar a agregar 'código', preferentemente un 'módulo' a la vez, depurar y probar hasta que sea sólido como una roca. Cada ciclo de 'desarrollo iterativo' incorpora un 'módulo', y lo hace a un 'código' seguro, ya probado.

Aqui debe considerarse que la construcción del diseño de aplicaciones vía 'prototipos', se ha convertido en uno de los pasos más importantes para su logro, hasta el punto de que los componentes obtenidos no se consideran como "desechables" al final y que, con herramientas anteriores a los 'desarrolladores de aplicaciones', los generadores de 'prototipos' sólo permitían crear las partes visuales de los sistemas y, cuando quedaba completa la 'interfaz' de usuario, se tenía que comenzar de nuevo la 'codificación' en un lenguaje "real" (*e.g. C++, Pascal, Visual Basic*); sin embargo, en el caso de la presente tesis en lo relativo a la aplicación de cómputo del GAR, el 'código' por 'implementar' al diseño de la 'interfaz' de usuario se generó simultáneamente al elaborar el 'prototipo'. En iteraciones subsecuentes de desarrollo, se trabajó sobre ese 'código' inicial para pasar a través de los 'prototipos' hasta lograr el *software* finalmente logrado.

En otras palabras y aunque suene redundante -pues no lo es-: las herramientas 'desarrolladoras de aplicaciones' facilitan el desarrollo de las aplicaciones y deben usarse las

mismas tanto para la elaboración de los 'prototipos' como para la 'implementación' propia de la aplicación. A guisa de ejemplo: "nada de que el diseño de las 'ventanas' se hace en una herramienta, su 'prototipo' en otra y el 'programa' respectivo en otra".

Uno de los principales beneficios del RAD consiste en la satisfacción inmediata que se obtiene al ver como surge rápidamente la aplicación de cómputo a partir del diseño; otro beneficio es que al completar cada parte de su desarrollo, puede depurarse y probarse inmediatamente. Cada ronda de diseño, 'prototipo' y prueba, principalmente con los usuarios, asegura que la aplicación se está construyendo sobre bases sólidas, elaborando así *software* más seguro. Otra ventaja más es que cada 'módulo' depurado y probado, se convierte en una pieza reutilizable que puede ayudar a reducir el tiempo de desarrollo en proyectos futuros.

Genéricamente, las tres 'actividades' básicas del mencionado 'desarrollo iterativo' consisten de:

* Requerimientos y diseño
* 'Prototipos' y depuración
* Prueba de análisis y resultados

4.3.2 Requerimientos y diseño

Para reunir los requerimientos de la aplicación, primero se debe preguntar qué acciones necesitará llevar a cabo el usuario. Este paso, obvio pero básico, conforma el marco de referencia para el diseño. Una vez decididas las 'tareas' que deben efectuarse, se necesitará saber que entradas y salidas se relacionan con -y entre- ellas, como se establece en la metodología de Warnier (ver 4.4). Invertir tiempo en esta etapa, a la larga, redituará en beneficios. Un principio reconocido del desarrollo de *software* indica que la detección de un error en la fase de diseño es de 10 a 100 veces menos costoso que la detección del mismo error en la etapa de pruebas..., así que se debe disponer del tiempo indispensable al efecto y razonar cómo utilizarán la aplicación los usuarios.

4.3.3 'Prototipos' y depuración

Muchos desarrolladores experimentados dicen que se debe hacer un 'prototipo' en papel antes de intentar escibir cualquier 'código'. Con un 'desarrollador de aplicaciones' como el empleado en el desarrollo del GAR (*Delphi*), se pueden hacer 'prototipos' con la misma facilidad sobre la pantalla y obtener resultados inmediatos, pues el 'código' de *Object Pascal* se **'compila'** de manera muy rápida, siendo posible experimentar y examinar visualmente diferentes elementos de diseño de la 'interfaz' de usuario. A pesar de esto, siempre es bueno tener un diseño -al menos a manera de estructura-, de lo que se hará con la aplicación. Para tal fin, se deben colocar los elementos de la 'interface' de usuario de acuerdo con los requerimientos de diseño, ordenarlos de manera que sean atractivos visualmente, y etiquetarlos de tal modo que su uso sea claro. Si en esta primera etapa se desea, por ejemplo, que las **'opciones del menú'** realicen alguna acción, se puede 'programar' en el 'evento' del 'objeto' respectivo; por ejemplo, en el 'evento' *OnClick* del 'componente' *TMainMenu* se puede simplemente exhibir el mensaje: "Aquí se ejecutará la captura", para posteriormente proseguir con la 'codificación' del caso.

4.3.4 Prueba de análisis y resultados

Se pueden efectuar diversos tipos de pruebas, a saber: de **'utilización'**, de 'funcionalidad' y de **'despliegue'**.

4.3.4.1 *'Utilización'*

Durante el diseño de los elementos de la 'interfaz' de usuario de la aplicación, la prueba de 'utilización' puede brindar información sobre si el flujo de 'tareas' de la aplicación es o no intuitivo al usuario, *e.g.:* ¿Se puede encontrar con facilidad el camino a través de los 'menúes', 'diálogos' y 'ventanas' de la aplicación?, ¿Están las cosas donde el usuario lo espera?. Éste es un buen momento para ver cuando acude el usuario a la tecla <F1> para intentar activar la **'ayuda'**, de modo que se puede captar en donde se debe agregar aquélla sensible al contexto.

4.3.4.2 *'Funcionalidad'*

Estas pruebas también denominadas de **'caja negra'** se debe realizar en cada etapa del 'desarrollo iterativo'. Comprende la captura de valores de entrada apócrifos y la verificación de las salidas adecuadas. Esto puede hacerse mediante la ejecución de la aplicación y la prueba de diferentes casos de entrada/salida. Para ayuda en esta 'actividad', hoy existen en el mercado varias soluciones de pruebas automatizadas; pero si el presupuesto es restringido, pueden hacerce las pruebas manualmente y en esta situación, se debe asegurar de llevar un buen registro de como se realizaron, de modo que puedan repetirse, en su caso, de manera confiable. Aqui también debe verificarse y mejorarse el rendimiento del 'prototipo'.

4.3.4.3 *'Despliegue'*

Ciertamente no menos importante es la prueba del 'código' en lo relativo a su instalación y 'despliegue'. Si la primera impresión que tiene el usuario es que la aplicación es defectuosa o difícil de usar, se habrá arruinado el intenso trabajo dedicado en el desarrollo iterativo. En este sentido, previamente se debe probar en varios tipos de *hardware*: discos compactados, sin espacio suficiente, etc.; para intentar crear cada tipo de situación de error que se ocurra durante la etapa de prueba. Luego, se debe revisar y verificar de nuevo que todos los **'archivos'** queden en el lugar adecuado y tengan los atributos correctos.

Una vez que se tengan los resultados de las pruebas, debe dedicarse tiempo para su análisis, *v.g.*: ¿Algunos errores provocaron confusión al usuario sobre el uso de la aplicación?, ¿otros los causaron omisiones comunes en la 'programación'? ... Tal vez hubo pruebas que omitieron lo que no debían; en otras palabras, la aplicación no detecta errores que debería detectar. Al terminar el análisis se debe usar para corregir el diseño, y asegurar la documentación de esos cambios en el mismo.

Con el RAD debe tenerse muy presente y hacer notar al usuario, que el 80% del trabajo se realiza en el 20% del tiempo disponible y que el 20% del trabajo restante requiere del otro 80% del tiempo.

* 4.4 RESUMEN DE LA METODOLOGÍA DE WARNIER

4.4.1 INTRODUCCIÓN

Esta metodología fue presentada por Warnier J. D. en 1974 y Orr extendió sus trabajos para abarcar una visión más amplia del dominio de la información. A pesar del tiempo trnascurrido desde ese entonces, constituye una herramienta lógica muy útil para el diseño de sistemas o 'programas' de cómputo y persigue que los mismos sean fáciles de entender para simplificar su mantenimiento. Su filosofía consiste en: *"Estructurar jerárquicamente un problema dividiéndolo en 'módulos' y en cada 'módulo' limitarse a usar tres estructuras de control: flujo secuencial, repetición y selección"* y su idea principal es: *"Las estructuras del 'programa' deben de basarse en las estructuras de los datos"*.

Partiendo de esta idea Warnier diseñó una notación con la cual se pueden describir datos al igual que 'procesos', acciones o 'programas', la cuál se representa mediante llaves que enmarcan información. Por ejemplo, al definir una función en matemáticas se emplea:

$$|x| = \begin{cases} x & x \text{ mayor o igual a cero} \\ -x & x \text{ menor que cero} \end{cases}$$

o al ejemplificar el siguiente 'proceso':

Usar un extinguidor
- Quite el seguro jalándolo
- Acérquese lo mas posible al fuego
- Oprima la palanca superior
- Dirija la nube de polvo a la base del

Las reglas para el uso de los diagramas en cuestión son:

* En esta parte del documento los 'Textos' indicados así, corresponden a títulos del reporte del estado de cuenta de cheques aquí mencionado. Nótese que 'Texto' es diferente de 'Texto' empleado en general para términos de cómputo.

A). Por medio de llaves se pueden describir conjuntos de datos o 'procesos'. Del lado izquierdo de la llave se pone el nombre del conjunto o del 'proceso' y del lado derecho, los elementos del conjunto o la descripción del 'proceso' o 'tarea', respectivamente.

B). La selección o alternativa se indica con el símbolo $\oplus$ que corresponde a ó (exclusivo), como en el siguiente caso:

Escribir valor absoluto
- $x < 0$ ⊕ $x < 0$
  - $x < 0$ → escribir -x
  - $\overline{x < 0}$ → escribir x

Nótese que aquí se empleó el símbolo $\overline{}$, que indica la negación de algo, particularmente:

$\overline{X < 0} = X \geq 0$.

Otros símbolos utilizados son: ϕ (letra griega fi), para indicar ausencia de acción.

Ejemplo
- $x < 0$ ⊕ $x < 0$
  - $x < 0$ → ϕ
  - $\overline{x < 0}$ → △3

△3 Un tríangulo con número adentro se usa para referirse a algo descrito en otro lugar.

C). La repetición se indica con una variable o número entre paréntesis bajo el nombre del conjunto o del 'proceso'.

Dirección → Subdirección (2) → Departamento (D)

En este ejemplo, el conjunto de Dirección. tiene **2** Subdirecciones y cada Subdirección se compone de varios **D**epartamentos. Generalmente el nombre de la variable es la inicial del conjunto o 'proceso'. Cuando se quiere hacer énfasis en que la repetición se hace un número determinado de veces por lo menos, se utiliza la siguiente notación: (n, N) donde N es la inicial del conjunto o 'proceso' y n es el número Ej. (0, N).

Debe considerarse que con base en llaves se establecen jerarquías, es decir, se describen los subconjuntos de un conjunto al abrirse una llave. Las llaves describen una instancia de los subconjuntos a los cuales están asociadas. En el presente caso se aprecia que una instancia cualquiera del subconjunto Subdirección se compone de Departamentos.

D). El flujo secuencial queda implícito en estos diagramas al establecer el orden de lectura, el cual es de arrriba hacia abajo.

La metodologían de Warnier descrita en términos de sus propios diagramas, es la siguiente:

Diseño de un programa utilizando la metodología de Warnier {
- Definir los productos o salidas
- Definir la estructura de los datos
- Definir la estructura lógica de las salidas
- Definir la base de datos lógica
- Análisis de eventos
- Diseño físico de la base de datos
- Diseño lógico del proceso

4.4.2 DESCRIPCIÓN BREVE

Aquí se pretende explicar resumidamente en que consisten los pasos mencionados en el diagrama anterior, con base en notas de clase que datan de 1981, elaboradas por el Act. Javier García. Al efecto se muestra su aplicación al diseño de un 'programa' que emite un reporte del estado de cuenta cheques.

4.4.2.1 Definir los productos o salidas

En este primer paso se describe la apariencia física de los resultados o el producto final. Al efecto se debe elaborar un dibujo descriptivo del mismo, como se muestra enseguida:

Ejemplo:

Banco

Sucursal 01 Merced

Estado de Cuenta

Cuenta Número 3091300 Saldo Inicial $17,500

Fecha	Cargo Diario	Abono Diario	Saldo
28-Jul	3,500		14,000
3-Sep	2,000		12,000
5-Sep		5,000	17,000
Total Cuenta 3091300	5,500	5,000	

Estado de Cuenta

Cuenta Número 3093200 Saldo Inicial $6,000

Fecha	Cargo Diario	Abono Diario	Saldo
10-Jul		5,000	11,000
10-Sep	2,000		9,000
10-Sep	1,000		8,000
Total Cuenta 3093200	3,000	5,000	

.
.
.

Sucursal 05 Observatorio

Cuenta Número 209000

•
•
•

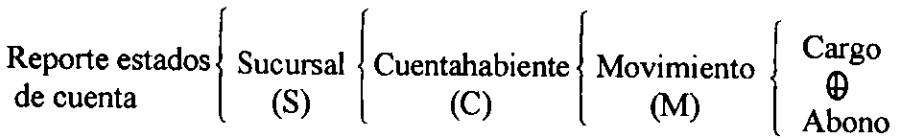
4.4.2.2 Definir la estructura de los datos

Una vez definida la salida, en este caso el reporte, se debe elaborar la estructura de los datos; para esto, se debe hacer lo siguiente:

1.- Buscar repeticiones

2.- Buscar selecciones o alternativas.

Al observar la salida del ejemplo se observa que para el conjunto banco se repiten las sucursales, dentro de cada sucursal se repiten los cuentahabientes, asi mismo, dentro de cada cuentahabiente se repiten los movimientos. Se observa también que estos últimos pueden ser o un cargo o un abono. Lo anterior en diagramas de Warnier se representa así:

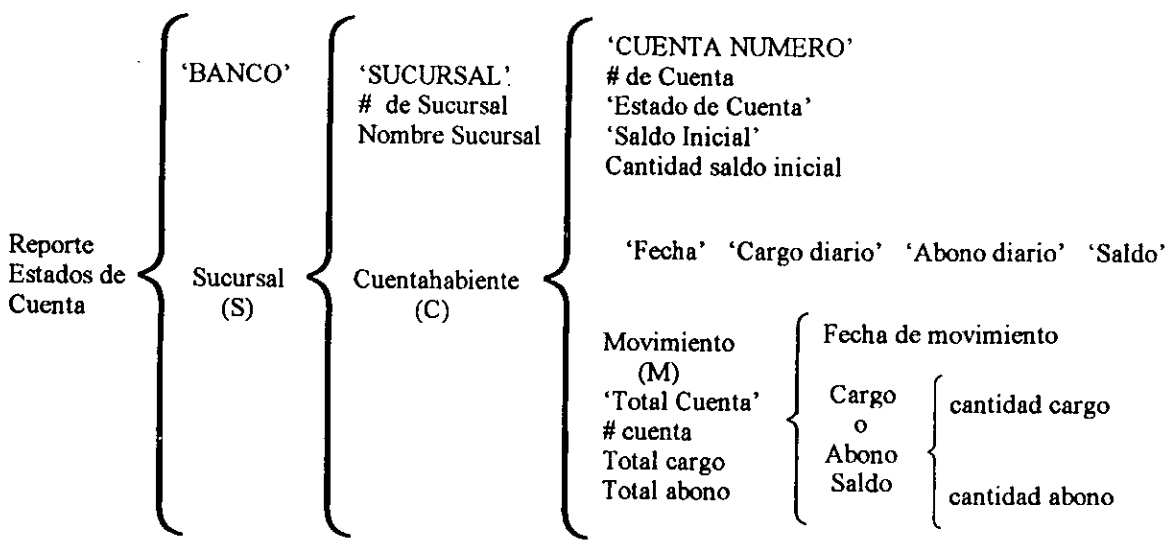


Alrededor de este diagrama girará el resto del análisis.

4.4.2.3 Definir la estructura lógica de las salidas

La estructura lógica de las salidas la constituyen todos los datos que aparecen en la salida, quedando incluidos encabezados, titulos y 'campos' de datos. Tomando como base la estructura de datos se elabora un diagrama que contenga todos los datos de la salida considerando su ubicación.

El diagrama del caso que se está analizando queda así:



Para hacer este tipo de diagramas, mucho ayuda elaborar una lista de todos los elementos que aparecen en la salida y para cada uno preguntarse:

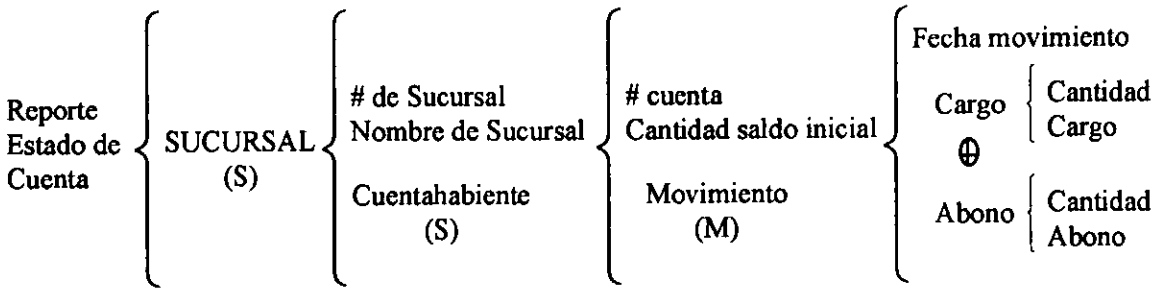
1). ¿Cuántas veces aparece?

2). ¿Cuándo?

y de esta manera se ubica cada dato en su lugar correspondiente

4.4.2.4 Definir la 'base de datos lógica'

La 'base de datos lógica' la constituye un diagrama que representa la mínima información necesaria para producir la salida; para obtenerlo, basta hacer lo siguiente: tomando como base el diagrama anterior se eliminan constantes, datos calculados y datos redundentes, dejando la primera ocurrencia. En el ejemplo la 'base de datos lógica' es:



Las literales 'BANCO', 'SUCURSAL', etc., se eliminan por ser constantes.

Los datos Saldo, Total Cargo y Total Abono se eliminan por ser datos que se pueden calcular. Obsérve que el dato # Cuenta se repite dos veces, en consecuencia, se deja la primera ocurrencia y se elimina la siguiente.

La 'base de datos lógica' representa también los datos que el 'programa' necesita para obtener el resultado deseado.

4.4.2.5 Análisis de 'eventos'

Con este análisis se identifican los 'eventos' que afectan la 'base de datos lógica' y su naturaleza. Es en esta fase del diseño donde se definen las limitaciones y suposiciones del 'programa', cuales 'eventos' van a considerarse válidos y cuales no.

En la 'base de datos lógica' se puede ver que existen dos clases de datos; unos llamados 'entidades', se encuentran encabezando las llaves a su lado izquierdo y otros denominados 'atributos' los cuales no encabezan ninguna llave y pertenecen a la 'entidad' que encabeza la llave donde se encuentran.

En el ejemplo las 'entidades' son: Sucursal, Nombre Sucursal, Cuentahabiente, Movimiento, Cargo, Abono y los 'atributos' son: No. de Sucursal, No. de Cuenta, Cantidad Saldo Inicial, Fecha Movimiento, Cantidad Cargo, Cantidad Abono. Una vez identificadas las 'entidades' y sus 'atributos' se necesita hacer lo siguiente para cada 'entidad':

1.- Encontrar como se crea una nueva instancia u ocurrencia.
2.- Como desaparece una instancia.
3.- Identificar cuáles y cómo pueden transformarse sus 'atributos'.

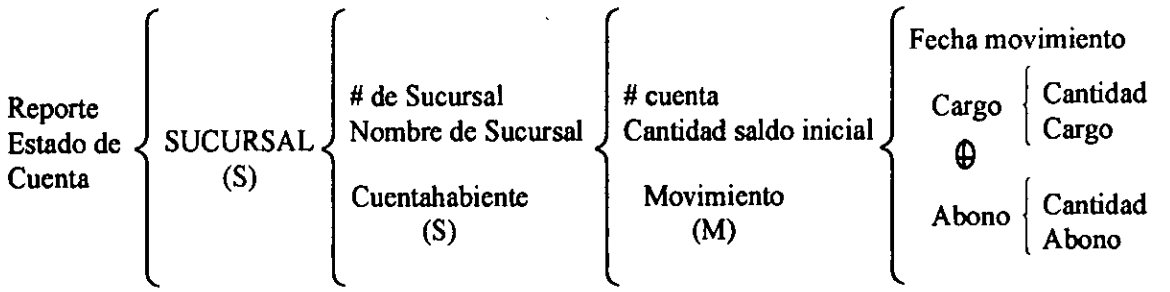
Como puede verse, se pretende identificar los 'eventos' que pueden afectar la 'base de datos lógica' y, en este paso, se determinan las medidas que se tomarán con respecto a cada uno de ellos. Estas medidas pueden ser por ejemplo: 'rutinas' de validación de información y 'rutinas' de actualización. Estas medidas se ubicarán posteriormente en el 'proceso' si se decide que el 'programa' debe contemplarlas. Para simplificar el ejemplo en cuestión, supóngase que los datos que se reciben están prevalidados y se requiere unicamente listar el reporte.

4.4.2.6 Diseño físico de la 'base de datos'

Se pospone hasta este punto el diseño físico de la 'base de datos'. La información necesaria para obtener los resultados se tiene en la 'base de datos lógica', ahora el problema es cómo distribuir esta información en 'archivos'.

Cada 'entidad' con sus 'atributos' formará un 'archivo' distinto. El o los 'atributos' que identifican inequivocamente una instancia de la 'entidad' conformará la **'llave del archivo'** o 'tabla'.

Esto se hace con el fin de **'normalizar'** los datos que "entran" al 'programa'. Esta distribución puede cambiar dependiendo de las 'facilidades' que se tengan y de otros criterios; por ejemplo: si se tienen 'llaves de archivo' alternas, si se desea tener redundancia, si se cuenta con 'archivos' previamente definidos, etc. Para el ejemplo se tiene:



Los 'archivos' propuestos podrían ser, entre otros:

1).- (Sucursal) No.Sucursal, Nombre Sucursal.

2).- (Cuenta) No. Sucursal, No. Cuenta, Cantidad Saldo Inicial.

3).- (Movimientos) No. Sucursal, No. Cuenta, Fecha Movimiento Cargo(-)
o Abono (+)

Donde se han subrayado las 'llaves de los archivos'.

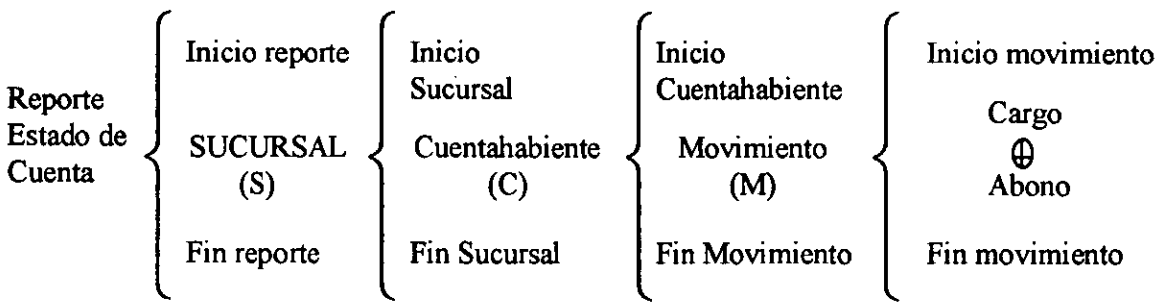
4.4.2.6 Diseño lógico del 'proceso'

En este paso se cuenta prácticamente con la estructura y el flujo de control del 'programa'. Para cada dato que aparece en la salida debe haber en el 'proceso' 'código' que lo genere y que debe ser ejecutado tantas veces como se repita el dato, es por esto que el 'proceso' tiene la misma estructura que los datos.

Basándose en la estructura de los datos, se debe empezar a llenar el diagrama correspondiente con acciones elementales sobre los datos:

Leérlos
Escribirlos
Calcularlos
Prepararlos

Para esto, primeramente se divide cada 'proceso' en tres partes: inicio, mitad y fin. En el ejemplo se tiene:



En la parte inicio del reporte deben estar las 'tareas' que se realizarán antes de hacer el 'proceso' sucursal y en la de fin del reporte, las que se deberán hacer después del 'proceso' sucursal y así sucesivamente.

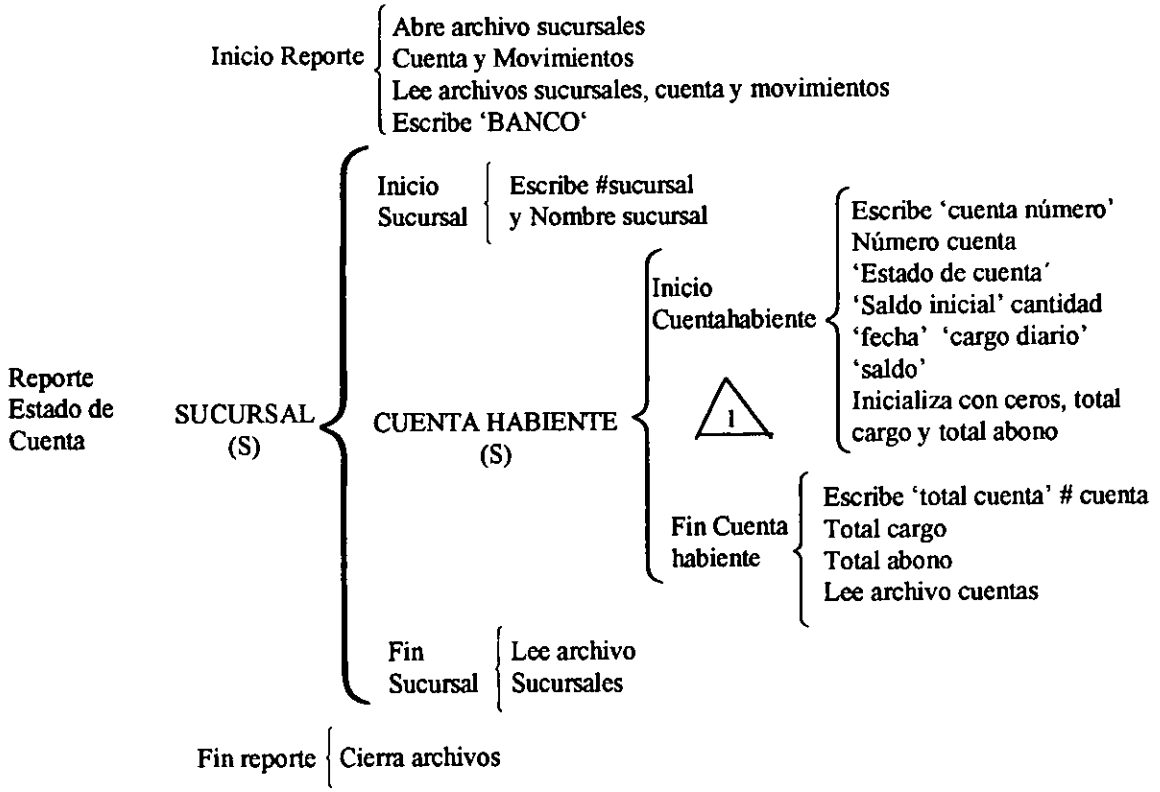
Para saber que acciones elementales se deben realizar, se cuenta con lo hecho en los pasos anteriores, lo cual indica con qué datos se cuenta, qué datos deben aparecer, en qué momento y lo que se debe hacer con respecto a los 'eventos' considerados previamente.

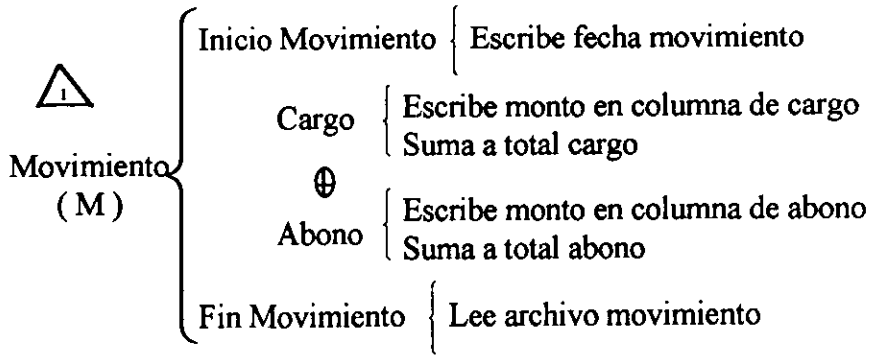
La idea es ir llenando el diagrama tomando siempre en cuenta la salida o resultados del 'proceso'. Al poner una 'tarea' en el diagrama se puede verificar que esté bien ubicada si se pregunta lo mismo que cuando se construyó la estructura lógica de las salidas, pero ahora:

¿Cuántas veces se debe ejecutar esto?

¿Cuándo se ejecuta?

Prosiguiendo con el ejemplo y para concluirlo, el diagrama del 'proceso' quedaría como se muestra enseguida:





5. VARIANTE DE LA METODOLOGÍA DE YOURDON

La presente variante de la metodología que se trata está orientada a 'implementación', vía 'prototipos', de sistemas y aplicaciones de cómputo que manipulan **'bases de datos relacionales'**. Se conformó contemplando como base técnica fundamental la metodología de Yourdon, intercalando aspectos de técnicas de RAD y diagramación de Warnier -estas últimas en una interpretación libre-. De una forma u otra obliga a generar documentación 'entregable' de los proyectos, misma que debe formar parte del acervo de organizaciones amplias, sobretodo ante cambios en sus administraciones a través del tiempo, constituyendo la base técnica de informática para efectos de entrega, recepción e incluso auditorías de sus áreas organizacionales de sistemas, además de considerar lo indispensable que resulta la documentación para efectos de mantenimiento. En divergencia con los puntos de vista más extremistas del RAD, la propuesta mantiene tres conceptos sobre las 'actividades' por realizar ampliamente conocidos en materia de desarrollo de sistemas: análisis, diseño e 'implementación' y pretende cubrir el hueco o "dimensión desconocida" entre el análisis y el diseño, a través de la elaboración en una forma particular de diagramas de Warnier; particularmente plantea para sus 'actividades' dos características sustantivas:

* La construcción y documentación de sistemas en forma 'descendente' o *top-down*, pero con un enfoque integrador que permita suspender temporalmente partes de las 'actividades' antes mencionadas, hasta un cierto nivel de profundidad, dejando la necesaria especificación detallada de las 'tareas' o 'procesos' para consideraciones subsecuentes, coincidiendo en esta parte con las técnicas del RAD.

* El desempeño de diversas 'actividades' y 'tareas' en forma paralela, a fin de permitir retroalimentación entre ellas, ligándolas a través de los 'prototipos', que a su vez, son la 'interface' entre los usuarios y el grupo de desarrolladores del sistema o aplicación. En este caso el mencionado paralelismo con la retroalimentación generada por el mismo, no es una forma de trabajar, sino que es la forma de trabajar.

Por otro lado, si bien la variante propuesta como cualquier otra, no tiene en principio nada que enseñar a personas experimentadas en el ámbito del diseño y la construcción de sistemas, es útil en la medida que facilita organizar las ideas, permite anticipar decisiones y trabajar de una forma más ordenada y consistente, así como determinar con precisión las 'actividades' y 'tareas' por desarrollar y uniformar los trabajos en la materia, sobretodo en organizaciones donde resulta muy desconcertante administrar simultáneamente varios proyectos, donde cada uno se trabaja de forma diferente. De igual forma, no resuelve la problemática derivada de los seres humanos: irresponsabilidad y/o falta de motivación del personal, carencias de presupuesto, etc.; en todo caso, resolver situaciones como éstas es un "privilegio" del líder de proyecto o administrador de sistemas, recordando que *"el hombres es y su circunstancia"*.

Aquí es muy pertinente parafrasear a Yourdon: la propuesta no enseña como organizar ni a organizarse, no indica como planificar o resolver problemas de "política", ni explica técnicas de 'prototipos' o a manejar 'desarrolladores de aplicaciones' o, a diseñar 'bases de datos', ni mucho menos a 'programar'. Tampoco pretende ser lo que en otros lugares se le conoce como *"la Biblia del desarrollo de sistemas"*.

De igual forma no enseña a ser un líder de proyecto. Para intentar llegar a serlo se requiere de un sentido básico de delegación, organización y administración .... y muchos años de esfuerzo. Esto se considera por la gran cantidad de personas, desde aficionados hasta licenciados que no se prepararon profesionalemente en el área, lo cual no es un impedimento para participar en ella -siempre que se fogueen y preparen adecuadamente-, haciendo referencia en particular a ese tipo de personas que que incluso, se engañan y llegan a sentir que son administradores de sistemas y hasta desarrolladores, cuando solamente tienen los nombramientos o cargos por el simple hecho de estar..., donde no les corresponde.

¿Cuántos "licenciados" que ni siquiera han ejercido su carrera de origen no son conscientes de que viven en el subempleo al pretender realizar las funciones de cierto nivel de responsabilidad en el área de sistemas cuando no están preparados ni tienen por lo menos la

experiencia a nivel de usuario, ni mucho menos la formación académica mínima indispensable?. Por regla general en casos como estos, tal tipo de "profesionales" se llegan a sostener sea por su falta de dignidad, su físico, sus compromisos y silencios, su corporativismo a ultranza -caiga quién caiga menos ellos-, entre otras de sus "virtudes", estando siempre a la expectativa de que alguién los "rescate"; empero, Newton sigue teniendo toda la razón: invariablemente, "las cosas por su propio peso caen".

Sin perder de vista lo anterior, se presenta la red de 'actividades' con la propuesta sobre una forma de trabajar aplicaciones o sistemas de cómputo, mediante el siguiente DFD:

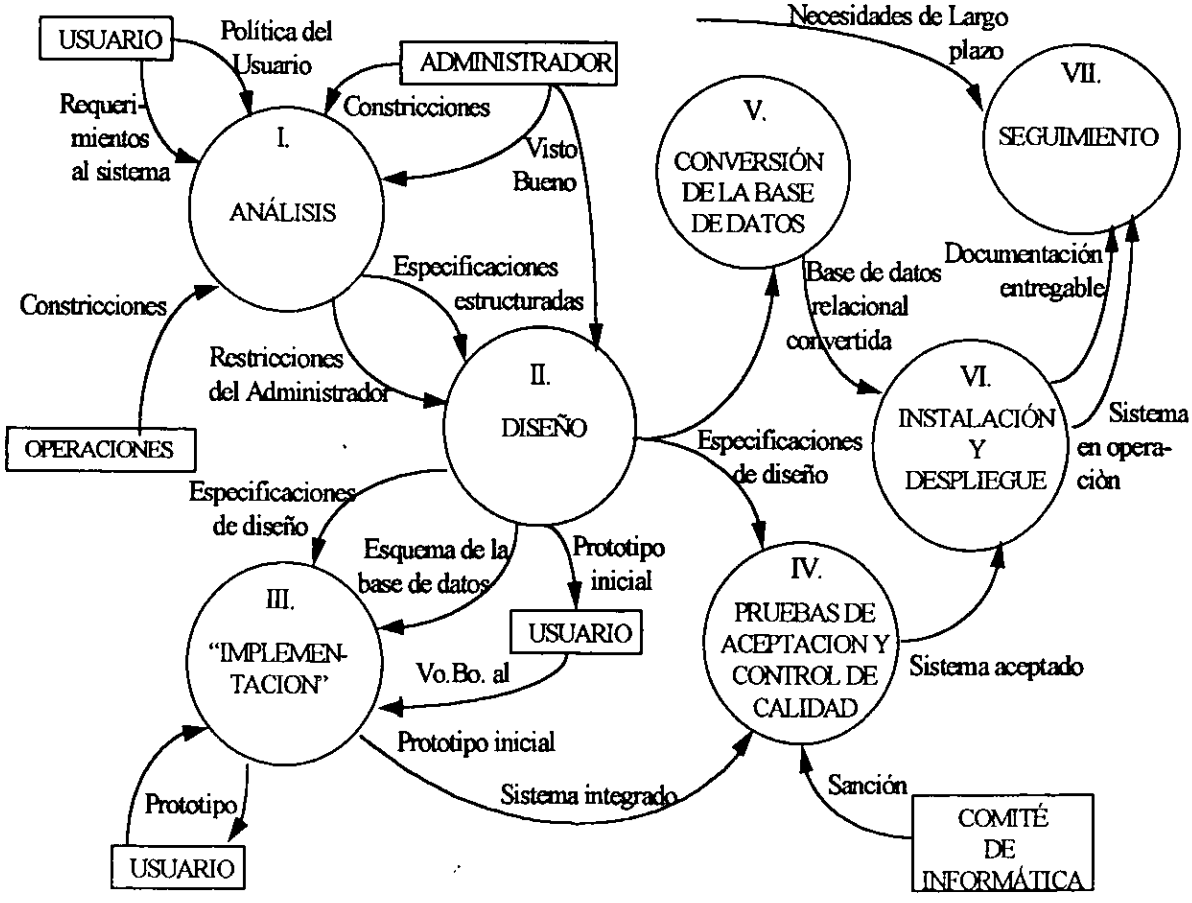


FIGURA 16: VARIANTE DE LA METODOLOGÍA DE YOURDON

5.1. EXPLICACIÓN GENÉRICA DE LA VARIANTE

Como puede observarse, el DFD anterior contempla siete 'actividades' y cuatro 'terminadores': **usuarios, administrador, operaciones y comité de informática**; estas instancias son, de una forma o de otra, los destinatarios últimos de los sistemas y proveen de información sustantiva al grupo de personas que desarrollen el proyecto:

* El usuario o área organizacional usuaria, es la única instancia que puede aceptar la aplicación de cómputo o sistema cuando esté terminado, es decir, es la única instancia que puede decidir si es conveniente integrarlo en su operación cotidiana y asumirlo como algo propio.
* El segundo término corresponde al administrador y/o líder de proyecto, que define entre otros aspectos las restricciones inamovibles para el mismo, asimismo, verifica el cumplimiento de acuerdos previos con los usuarios; puede ser que el usuario y el administrador, en ciertos casos, sean la misma instancia.
* El tercer concepto en cuestión corresponde a las instancias o áreas responsables de la operación cotidiana de *hardware* y *software* en la organización, que conocen las constricciones ambientales en la materia para el nuevo sistema (*e.g.* limitaciones de *software*, acceso a 'facilidades' de comunicación).
* Por comité de informática se entiende la instancia colegiada de la organización que, entre otras de sus funciones, debe otorgar aprobación a sus sistemas o aplicaciones de cómputo para su conjunto de áreas, a efectos de su 'implantación' y operación, así como para posibles auditorías en materia de informática.

Se debe aclarar que del DFD anterior no se deriva implicación alguna acerca de que debe haber estrictamente un grupo de trabajo por cada 'burbuja', cuando en México es de lo más común -"por exceso de los dineros"-, que las mismas personas en diferentes momentos, participen prácticamente en todas las 'actividades'; tampoco se implica que la 'actividad' *n-ésima* deba concluir del todo antes de iniciar la *n+1-ésima* sino que, justamente, la red de flujos de datos mostrada conecta "fuertemente" las 'actividades' ahí representadas, sin excluir la posibilidad de realizar algunas de elllas en paralelo lo cuál se muestra en detalle en el

punto siguiente (5.2) -al 'explotar' las 'burbujas' de este diagrama-; en todo caso solamente muestra las entradas o "insumos" requeridos por cada 'actividad' y las salidas o "productos" ahí realizados; por otro lado, cabe aclarar que aunque en el DFD sólo se indica explícitamente la participación del administrador en la 'actividad' del análisis, ciertamente ejerce control y tiene responsabilidad sobre todas y cada una de las demás, -hasta "aterrizarlo" en el mejor de los casos-. Cabe comentar que, aunque se aplique la mejor metodología del mundo a un proyecto, si el líder del mismo es incompetente.... resulta en un engaño el esperar algún resultado positivo, sobre todo si de por medio existe algún plan de trabajo...; al caso se puede aplicar al responsable del proyecto la frase que se emplea con los toreros: *"además de serlo, hay que parecerlo"*.

Por otro lado, debe notarse que el usuario está involucrado en el proyecto no sólo en sus inicios, sino que también lo está desde la 'implementación' inicial del 'prototipo', logrando de esta forma que desde esos momentos él vea al proyecto como "su sistema" y a él tambien se le vea -desde esos momentos-, como "el usuario", siendo esto, relativamente, muchísimo antes de la liberación del sistema o aplicación de cómputo.

Antes de profundizar más es conveniente indicar, parafraseando a Yourdon, que el empleo de cualquier metodología, y particularmente la variante propuesta, no garantiza la supresión de los grandes males en el desarrollo de sistemas: pobreza de análisis, control inexistente o mínimo sobre diseño y 'codificación', así como 'implementación' 'ascendente'. En todo caso, la aplicación de la metodología propuesta disminuye la probabilidad de incurrir en esos males, sin relevar al líder del proyecto de administrarlo y "aterrizarlo", en los sentidos más amplios de las palabras, dotándole justamente, de una herramienta que le permitirá identificar problemas acertadamente, en el tiempo pertinente para su solución; en este contexto es pertinente mencionar que, en todo caso, es una responsabilidad absoluta del líder del proyecto definir como 'programar', graduar y calendarizar las 'actividades' del mismo. Contemplando lo anterior, a continuación se plantea la descripción de cada una de las 'burbujas' plasmadas en el DFD correspondiente (figura 16).

I. ANÁLISIS

Con base en el **"cuestionario de inspección"** (Anexo 1), se persigue iniciar la generación del conjunto de 'especificaciones estructuradas' del modelo de comportamiento del sistema, a partir de una adecuada transformación de su carta del proyecto (ver I.1, página 65) y de las políticas del usuario, así como mediante el uso de herramientas gráficas, que deben conformar parte de la documentación 'entregable' del proyecto (Anexo 3). También aquí se deberá generar una narrativa o descripción formal estructurada para el uso del nuevo sistema, que será parte de su 'ayuda' para su entrega al grupo o área responsable de la 'implementación' y para efectos del desarrollo e incluso de su seguimiento o mantenimiento, así como para su refinamiento y entrega, en su momento, al usuario final.

II. DISEÑO

Consiste en la determinación y ubicación de porciones de las especificaciones del análisis (ver figura 11 -página 29-), para discernir una apropiada asignación de 'tareas' a los diversos elementos del sistema, donde para cada una de ellas se debe establecer, en forma 'descendente' o *top-down*, una jerarquía de 'módulos' de 'programación' y las 'interfaces' entre ellos; por ejemplo, entre los **'programas cliente'** con el o los **'servidores de base de datos'**, (*v.g.* ***'stored procedures'***), a fin de poder 'implementar' apropiadamente las especificaciones derivadas del análisis y el diseño en el **'prototipo inicial'** del sistema de cómputo o aplicación. Esto es muy relevante en virtud de las actuales bondades de los sistemas operativos y ambientes de trabajo (**'multitareas'**), que permiten la ejecución de diversas 'burbujas' de nivel bajo -correspondientes a una 'burbuja' origen-, de forma **'concurrente'** en un mismo 'procesador'; en el caso concreto del **'sistema operativo'** (OS por sus siglas en inglés) denominado UNIX, a través de conexiones de tubería y otras herramientas, es posible instrumentar e intercomunicar en su ***'background'*** dichos 'procesos', de forma eficiente y totalmente 'transparente' a los usuarios y a otras 'aplicaciones'.

III. 'IMPLEMENTACIÓN'

Contempla tanto la 'codificación' como la integración de los 'módulos' en el 'prototipo' que dará origen a la aplicación de cómputo o sistema, vía 'desarrollador de aplicaciones' o 'programación estructurada' y, en su caso, con la programación del 'servidor de base de datos', mediante la

ejecución invariable de cíclos de 'desarrollo iterativo', donde con cada ronda de diseño, 'prototipo' y pruebas, se puede verificar que la aplicación se está construyendo sobre bases firmes, hasta comprender la complejidad requerida al sistema, aplicando en este sentido el siguiente planteamiento del RAD: por cada cíclo de desarrollo concluido se incorpora un 'módulo', y lo hace a un 'código' ya probado y seguro, *i.e.* la aplicación se debe construir de forma **'incremental'**.

IV. GENERACIÓN DE PRUEBAS DE ACEPTACIÓN Y CONTROL DE CALIDAD

Se deben elaborar con base en el conjunto de 'especificaciones estructuradas' de la aplicación, mismas que son el basamento para su 'implementación'. En esta 'actividad' es posible efectuar diversos tipos de pruebas, entre otras de 'utilización', 'funcionalidad' y 'despliegue'. Para la aplicación del control de calidad se requiere de la generación de las propias pruebas de aceptación y del sistema integrado. Debe concebirse como un cíclo del tipo: prueba --> ¿aceptación del usuario? --> ajustes o aceptación.

V. CONVERSIÓN DE 'BASES DE DATOS'

Es posible que existan datos en operación o, en su defecto, que deban generarse para la 'implantación' de la aplicación; en todo caso para su migración, se requiere de las especificaciones correspondientes derivadas del diseño.

VI. INSTALACIÓN Y DESPLIEGUE

Esta 'actividad' requiere como entradas la 'base de datos' convertida y el sistema aceptado, que no es otra cosa que el todo integrado que ha aprobado el control de calidad y recibido la sanción del comité de informática.

VII. SEGUIMIENTO

Consiste de las tareas básicas de seguimiento y mantenimiento de las aplicaciones de cómputo en producción. Muchos logros importantes en la materia en organizaciones amplias "se llegan a caer" por ausencia o descuido de este rubro.

5.2 EXPLICACIÓN DETALLADA DE LA VARIANTE

I. ANÁLISIS

Objetivo:

Producir las 'especificaciones estructuradas' o **"modelo en papel"** del sistema, donde se establece qué y cómo debe proceder el mismo, a través de: su modelo ambiental, un conjunto de DFD nivelados apropiadamente, el DD, por lo menos un diagrama del ERD general con las 'entidades' detectadas hasta ese momento o si se tiene capacidad suficiente, un borrador inicial del 'esquema de la base de datos' incluyendo una descriptiva con su 'funcionalidad' tentativa; también se debe generar la descripción concreta de las constricciones impuestas a la aplicación de cómputo, así como una narrativa inicial de su 'ayuda'.

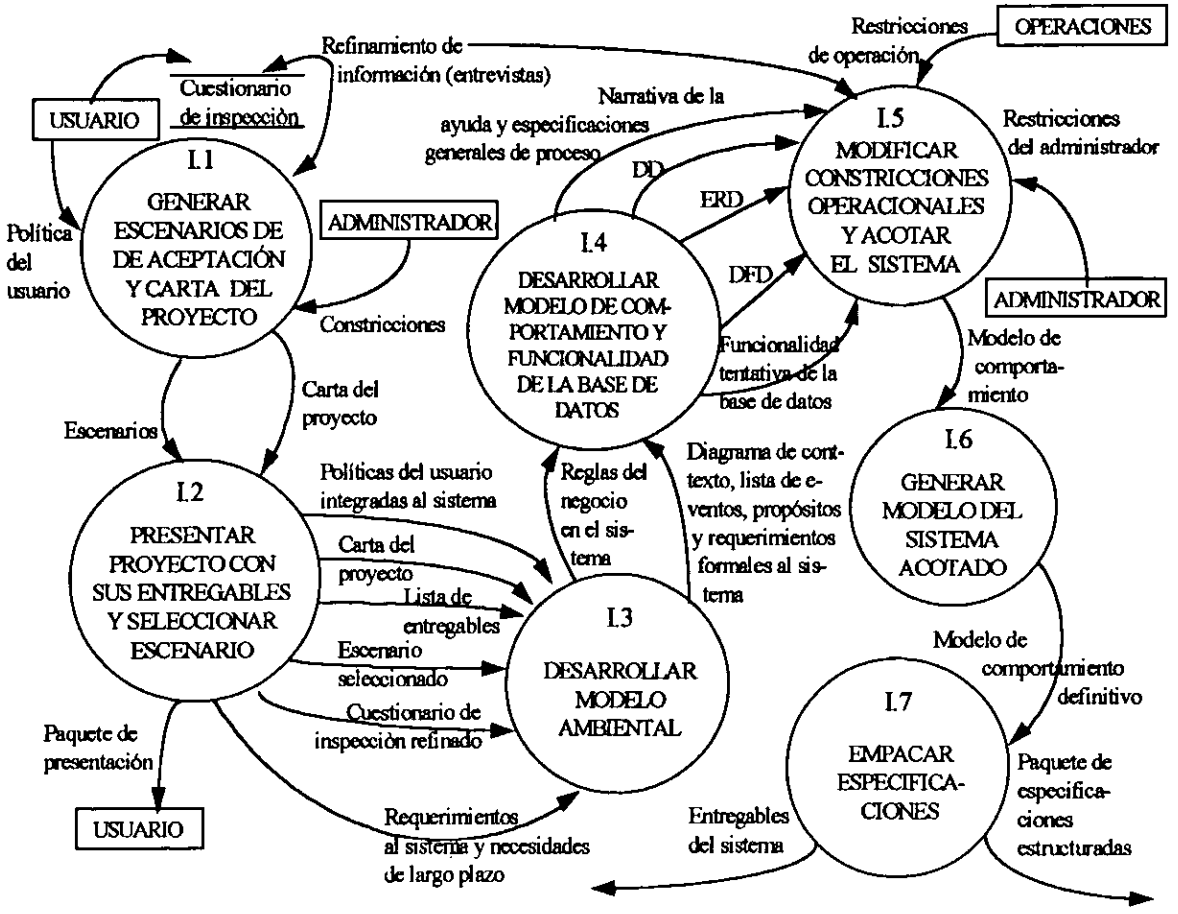


FIGURA 17: DFD DE ANÁLISIS

Tareas

I.1 Generar escenarios de aceptación y la carta del proyecto

*** Sugerencia(s):**

** A mayor precisión en el propósito del sistema es posible acotar con mejor exactitud su alcance.

** El cuestionario de inspección (ver Anexo 1), es una herramienta que capta información sustantiva para el proyecto y además permite dar entrada a diferentes interacciones con los usuarios en diversos niveles de la organización donde se pretenda realizar su 'implantación', perseguiendo detectar causas de problemas en vez de síntomas. Debe tenerse cuidado con lo anterior pues suele suceder que se solicite desarrollar sistemas para áreas organizacionales en donde no se ubica la fuente de la problemática que se pretende resolver.

** Como resultado de todas las interacciones con los usuarios, es recomendable generar minutas de los acuerdos a que se llegue, a fin de preservarlos a través del desarrollo del proyecto para cualquier aclaración al respecto.

** Una cuidadosa revisión de escenarios y beneficios asociados generará dudas que deberán clarificarse, vía retroalimentación con los usuarios y sus directivos, para obtener más información y, en su caso, modificar constricciones para dar viabilidad técnica al proyecto. En este contexto, cabe volver a aclarar que lo que diferencia a una constricción de una restricción es que mientras las primeras se pueden modificar en el transcurso del desarrollo, las segundas son inamovibles, incluso después de concluido el proyecto como tal.

*** Insumo(s):**

** **Cuestionario de inspección:** Requerimientos al sistema o aplicación de cómputo + Respuestas esperadas de la aplicación + Lista verificable de infraestructura de cómputo + Objetivos tentativos del sistema o aplicación de cómputo + Metas y necesidades de largo plazo del proyecto + Recursos humanos disponibles para la 'implantación'.

*** **Requerimientos al sistema o aplicación de cómputo:** Lista inestructurada y exhaustiva con las características de operación y 'funcionalidad' requerida al sistema o aplicación de cómputo.

*** **Respuestas esperadas del sistema:** Lista inestructurada y exhaustiva con las características de comportamiento planteadas al sistema o aplicación de cómputo, en diferentes situaciones y "tiempos".

*** **Lista verificable de infraestructura de cómputo:** Relación de requerimientos mínimos para instrumentar la aplicación o sistema de cómputo.

*** **Recursos humanos disponibles para la 'implantación':** Lista del personal involucrado para estos efectos y características de sus conocimientos en informática para detectar, en su caso, necesidades de capacitación.

** **Políticas del usuario:** Establecidas previamente o al efecto por escrito.

** **Refinamiento de información:** Entrevistas diversas con usuarios a fin de detallar y/o precisar información del cuestionario de inspección.

** **Constricciones:** Conjunto de limitaciones proporcionadas por el administrador de la organización al administrador o líder del proyecto en materia de: Tiempo + Presupuesto + Operacionales + Recursos humanos asignados al proyecto. Pueden variar en el tiempo.

* **Salida(s):**

** **Escenarios de aceptación:** Resúmenes preliminares con la propuesta de 'funcionalidad' de sistemas o aplicaciones posibles que preferentemente deben presentarse en formato de DFD + Beneficios asociados a los escenarios (*v.g.* reducción de tiempos de trámite) + Costos asociados a los escenarios + Planes de trabajo asociados a los escenarios + Políticas del usuario contempladas en el sistema o aplicación de cómputo.

** **Carta del proyecto:** Documento integrado con la información de: Planteamiento del(los) objetivo(s) del escenario seleccionado + Extracto del proyecto correspondiente al escenario seleccionado + Constricciones de: tiempo, costos, técnicas y operacionales + Políticas del usuario contempladas para su integración al

sistema + Escenarios de aceptación desechados (para tener historia del porqué se seleccionó el escenario elegido).

*** **Planteamiento del (los) objetivo(s):** Recopilación de frases concisas con: 'Funcionalidad' por ser 'implementada' en el sistema + Deficiencias por ser remediadas + Características por ser modificadas.

*** **Extracto del proyecto:** Nombre del proyecto + Redacción breve con su misión o propósito genérico + Delimitación del (las) área(s) organizacional(es) a ser comprendida(s) + Fecha de inicio + Fecha de entrega + Asignación de presupuesto y/o costo original (*hardware*, *software*, personal) + Nombre del responsable por parte de los usuarios + Nombre del líder de proyecto. Al integrar fechas de entrega con las limitaciones del punto siguiente, es posible generar un plan de trabajo más o menos detallado, según se solicite.

*** **Constricciones** impuestas por el administrador del proyecto en materia de: Tiempo + Técnicas + Operación. Se debe recordar que estas estrecheces iniciales pueden variar a favor o en contra del proyecto en el transcurso del mismo.

I.2 Presentar proyecto con sus 'entregables' y seleccionar escenario

* **Sugerencia(s):**

** Si bien la presentación del proyecto no durará tal vez mas allá de dos horas, debe considerarse como un evento de la mayor relevancia y en razón de esto, se debe invertir el mayor tiempo y la mayor calidad de trabajo posible a la misma. También, de ser posible, la presentación debe autoexponerse un par de veces antes de la fecha compromiso a fin de evitar el menor error en su exposición.

** Posterior a la presentación, debe procurarse que el escenario seleccionado se derive de una decisión consensada entre el administrador y el usuario y sus directivos.

** Es una decisión del administrador incluir en la presentación, por iniciativa propia, una lista de 'entregables' del proyecto, sobretodo en materia de su contenido y alcance: para unos, dependiendo de la organización de que se trate, puede bastar con 'instalar' los **'programas ejecutables'** y para otros, adicional a lo anterior, entregar todo el cúmulo de documentación hasta este momento citada, adémas de la que se llegue a desarrollar en el transcurso del proyecto (*e.g.* 'diccionario de datos', **'programas fuente'**, DFD). En todo caso y sobretodo con empresas particulares ("despachos"), como usuario debe estar uno muy atento para que no "se lo lleven al baile" y como administrador muy cauto para no comprometerse a entregar en tiempos inapropiados lo que no es posible, lo que en su caso conduce a ambas partes a asesorarse legalmente y generalmente, a la firma de uno o varios contratos. Normalmente los despachos llevan la ventaja sobre los usuarios en estas situaciones -aunque hay usuarios "muy bien asesorados", que ignoran o que pretenden hacer creer ignoran este pequeño detalle-.

Preferentemente, la definición de la lista de 'entregables' del proyecto debería ser una decisión consensada entre el administrador y los directivos de los usuarios. En el óptimo de los casos, se debe definir que los 'entregables' forman, de suyo, parte del acervo de la organización.

.* **Insumo(s):**

** Escenarios de aceptación.

** Carta del proyecto.

* **Salida(s):**

** **Paquete de presentación:** Síntesis de la información pertinente y relevante de los escenarios de aceptación y de la carta del proyecto, esta última preparada para fines de exposición formal a los usuarios del sistema y sus directivos; adicionalmente persigue definir la infraestructura definitiva disponible para el proyecto, así como "congelar" las constricciones ambientales para definir sus restricciones definitivas. Debería mostrarse con algún paquete de cómputo específico para presentaciones.

** **Escenario seleccionado:** Debe ser elegido del conjunto de escenarios de aceptación para, a partir de él, modelar el sistema y proceder a su diseño e 'implementación'. Con base en el se concreta la carta definitiva del proyecto.

** **Lista de 'entregables':** Lista anexa al paquete de presentación que contemplará según el caso, entre otros, los siguientes rubros: Cuestionario de inspección refinado + Carta del proyecto + Un conjunto de DFD nivelados + El 'diccionario de datos' + 'Esquema(s) de la(s) base(s) de datos', etc., para los cuáles se establecerá el compromiso de la entrega de avances (*v.g.* al final de cada 'actividad') y la entrega de la documentación formal al término del proyecto. Mucho de la extensión y contenido de la lista dependerá de los puntos de vista y necesidades del usuario, así como del escenario seleccionado. Previo acuerdo con los usuarios y sus directivos se debería obtener la definición de la lista. Cabe comentar que por cada proyecto debería existir en la organización un respaldo en medios magnéticos correspondiente a la documentación 'entregable' mencionada, misma que de esta manera, debe formar parte del acervo de la organización.

** **Cuestionario de inspección refinado:** Es un compendio de todos los cuestionarios de inspección aplicados a través de la estructura organizacional, en sus diferentes niveles, a los diversos usuarios y fuentes de datos involucrados en el proyecto, con anexos de toda aquélla información adicional necesaria para el mismo; es obtenido con los usuarios a través de diversas entrevistas directas, donde se ratifica o rectifica la información recabada originalmente.

** **Requerimientos al sistema y necesidades de largo plazo:** Lista estructurada y exhaustiva con las características de operación y 'funcionalidad' requerida al sistema, clasificada por orden de importancia, donde se señalan aquéllas necesidades de largo plazo que a pesar de no estar contempladas inicialmente en el alcance del sistema, deben considerarse para incluir las previsiones necesarias en el diseño a fin de poder realizar en un futuro inmediato, las ampliaciones pertinentes -en una siguiente oportunidad-.

I.3 Desarrollar modelo ambiental

*** Sugerencia(s):**

** Entre otros elementos, pero fundamentalmente con base en las políticas del usuario, deben identificarse y/o construirse las 'reglas del negocio' (*bussines rulers*) para su integración al sistema o aplicación de cómputo; dichas 'reglas' no son otra cosa que mandatos del más alto nivel y aplicación organizacional relativos a las funciones sustantivas o de servicio de la organización, que deben ser establecidas en la 'funcionalidad' del sistema y que por ningún motivo deben ser soslayadas en el mismo (*v.g. "el número de inventario de los equipos debe ser único."*).

** Resulta muy conveniente integrar la 'funcionalidad' y 'facilidades' proporcionadas por herramientas ***CASE*** (*Computer-Aided Software Engineering*) al desarrollo de proyectos de cómputo, particularmente en la 'actividad' del análisis, como se muestra en la figura siguiente:

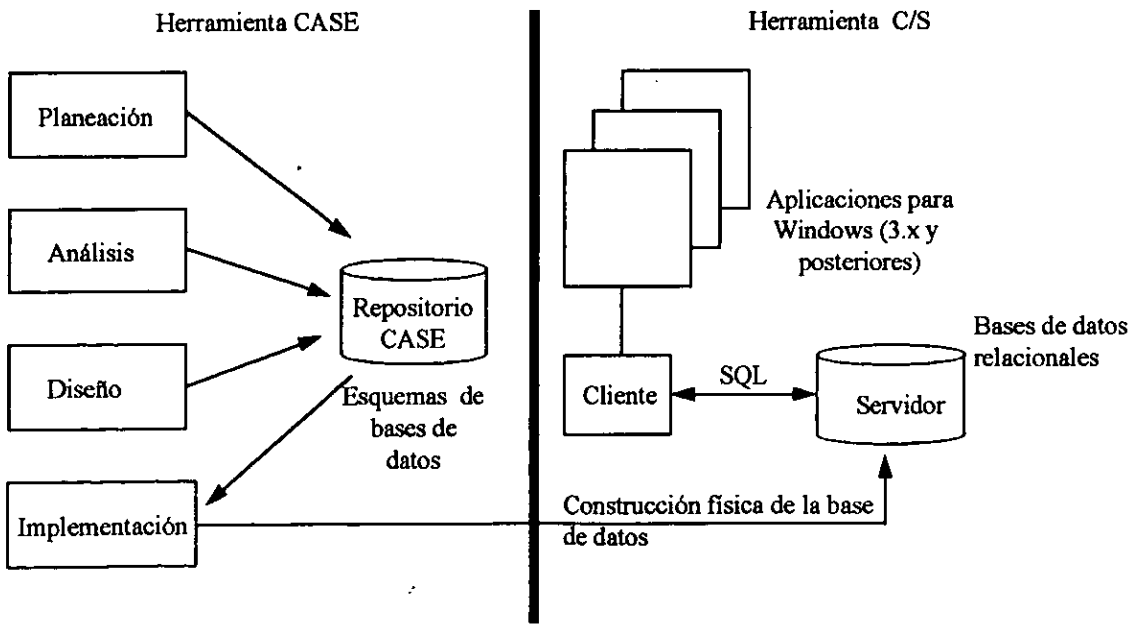


FIGURA 18: INTEGRACIÓN DE HERRAMIENTAS CASE CON C/S

*** Insumo(s):**

** Escenario seleccionado.

** Carta del proyecto.

** Lista de 'entregables'.

** Cuestionario de inspección refinado.

** Requerimientos al sistema y necesidades de largo plazo.

*** Salida(s):**

** **Modelo ambiental:** Documento que muestra cómo debe interactuar el sistema o aplicación de cómputo con su medio ambiente, a través de: Lista de eventos + 'Diagrama de contexto' del sistema + Propósitos y requerimientos formales al sistema.

*** **Lista de eventos:** Relación exhaustiva de estímulos del medio ambiente externos al sistema, ante los cuales debe responder, donde se debe indicar la persona, 'entidad' o 'proceso' que dan origen a dichos eventos, preferentemente en formato: sujeto, verbo y complemento.

*** **'Diagrama de contexto' del sistema o aplicación:** DFD correspondiente al nivel 0, donde se delimitan las fronteras entre el sistema y su entorno (*e.g.* otros sistemas y fuentes de datos con los que tiene que ver), así como los datos que deben ser 'procesados' o producidos y aquéllos que deben ser generados o consumidos por 'entidades' externas.

*** **Propósitos y requerimientos formales al sistema:** Refinamiento de la información inicial obtenida para estos rubros, que han sido transcritos pero en una forma y lenguaje más acorde para el trabajo de los análistas.

** **'Reglas del negocio' en el sistema:** Mandatos del más alto nivel y aplicación organizacional relativos a las funciones sustantivas de la organización que deben ser vigiladas por el sistema al ser establecidas en el mismo -ni una 'regla' más ni una menos-.

I.4 Desarrollar modelo de comportamiento y 'funcionalidad' tentativa de la 'base de datos'

*** Sugerencia(s):**

** El planteamiento del sistema derivado del análisis no debe implicar todavía ninguna decisión acerca del *hardware* para el proyecto; de ahí el concepto "modelo en papel".

** En esta 'subactividad', en las entrevistas con los usuarios, el "olfato" del analista debe estar lo más sensible posible para captar y comprender flashes de información de los usuarios a fin de poder vertirlos en la 'funcionalidad' tentativa de la 'base de datos'. Ejemplos de esto son frases como: *"mi reporte de ventas lo hago puntualmente cada semana el viernes por la mañana, para su entrega al jefe, religiosamente, a las 17:00 Hrs.":* aquí huele a que hay que 'programar' algo que se debe "disparar" sistemáticamente, tal vez a las 14:00 Hrs. de cada viernes -mientras no cambie ese jefe-. *"El problema es que los de la mañana luego duplican en la libreta el registro de las entregas...":* aquí huele a un ***'constraint'*** para evitar la duplicación del registro de ciertos 'atributos'.

*** Insumo(s):**

** Modelo ambiental.

** 'Reglas del negocio' en el sistema.

*** Salida(s):**

** **Modelo de comportamiento:** Documento integrado y sancionado por el administrador; muestra el qué debe hacer el sistema y el cómo debe interactuar adecuadamente en sí, como con su medio ambiente, con base en: DD + Conjunto de ERD y/o borrador inicial de 'esquema de la base de datos' + Conjunto de DFD nivelados apropiadamente + Narrativa de 'ayuda' + Especificaciones generales de 'procesos' + 'Funcionalidad' tentativa de la 'base de datos'.

** **Narrativa de la 'ayuda' y especificaciones generales de 'proceso':** Es muy importante desarrollar el trabajo correspondiente a ambos conceptos en forma paralela, sea con un solo equipo de trabajo o con dos diferentes, en cuyo caso se requerirá un esfuerzo adicional de coordinación.

*** **Narrativa de la 'ayuda':** Textos que describen exactamente como trabajará el 'programa', incluyendo los rangos de validación y otros detalles. Esto se logra en mucho a partir de entrevistas e investigación con los

usuarios, que a su vez genera un conocimiento completo de lo que está por aplicarse. Con la narrativa en cuestión se tiene una especificación más a partir de la cual trabajar, una guía de los valores a probar y el borrador de los manuales para el usuario final que serán integrados al sistema mediante su propia 'ayuda': con éste y los otros elementos considerados en el método propuesto, es posible que el administrador con una perspectiva amplia, en su interior, pueda comenzar a diseñar incluso de forma muy anticipada, algo que denominaría un "preprototipo" de la aplicación (abstracción pura). Cabe mencionar que sistemas de uso exclusivo para pocas o una sola área organizacional, de uso frecuente y repetitivo, demandarán menor trabajo al respecto con relación a sistemas con difusión amplia, de uso en múltiples y variadas áreas organizacionales.

.*** **Especificaciones generales de 'proceso':** Textos semiestructurados que pueden contener condiciones que deben ser verdaderas antes y/o después de que opere un 'proceso', o bien pueden corresponder a 'diagramas de flujo', 'tablas de decisión', etc. La idea central es que cada 'proceso' identificado sea sustentado por una especificación de si mismo y descrito por una narrativa de 'ayuda' -*v.g.*: *"al oprimir el botón x aparecerá en la 'ventana'... y, en ese evento se suma la serie de datos..."*-, que comprendan con precisión el qué y el cómo debe funcionar el sistema. Cabe mencionar que en estas especificaciones, no es conveniente establecer el 'algoritmo del proceso' mediante el cual se producirá el comportamiento planteado para el sistema, a fin de que el diseñador o el programador proponga el más apropiado -con y desde un punto de vista de diseño e 'implementación'-.

** **'Funcionalidad' tentativa de la 'base de datos':** Se construye contemplando, entre otros aspectos, los requerimientos al sistema o aplicación y la lista de eventos asociados; *grosso modo*, debe contener una descripción genérica de los *data store* empleados en los DFD, donde debe detallarse de ser posible, qué características tendrán los datos de entrada, así como describir su 'funcionalidad' y cuáles deberá producir y vigilar la propia 'base de datos'; en otras palabras, es la explicación de las

flechas de conexión entre las 'burbujas' de los DFD y los 'almacenes de datos' ahí contenidos. Esta descriptiva debería realizarse con base en la construcción -preferentemente en paralelo-, de los DD, ERD y/o borrador inicial de 'esquema de la base de datos', y DFD iniciales. Al respecto debe considerarse y aplicarse, cuando es posible, lo que escribieron Yourdon y Coad :

> *"Events are those ocurrences in the outside world that planned-response system must respond to. Each event corresponds to a bubble; for a system with 150 events, draw 150 bubbles."*

I.5 Modificar constricciones operacionales y acotar el sistema

*** Sugerencia(s):**

** Un aspecto de la mayor relevancia es poder modificar en caso necesario -por parte del administrador-, las estrecheces iniciales de operación para dar viabilidad técnica al proyecto (*e.g.* negociar, conseguir ampliar el presupuesto).

** Sin mayor comentario, otra vez Yourdon y Coad: *"System context is an indication of how much of the problem domain will be embraced by the automated system..., all within the 'quadruple constraint' (inspired by Rosenau, 1981) that affects all systems:*

Quadruple Constraint = Capability + Schedule + Budget + People

System context is set by quadruple constraint negotiations. To control a project, a manager must be accountable in all four areas."

*** Insumo(s):**

** DFD.

** DD.

** ERD o borrador inicial del 'esquema de la base de datos'.

** 'Funcionalidad' tentativa de la 'base de datos'.

** Narrativa de la 'ayuda'.

** **Refinamiento de información (entrevistas):** Serie de intercambios de información entre el grupo de analistas y los usuarios, que persiguien precisarla, para refinar toda aquélla pertinente en beneficio del trabajo. En el mejor de los casos, desde los inicios el usuario hace suyo el proyecto como tal, así como al nuevo sistema en su momento.

** **Restricciones de operación:** Son establecidas por las instancias o áreas responsables de la operación cotidiana de *hardware* y *software* en la organización y surgen a partir de su conocimiento de las constricciones ambientales en la materia para el nuevo sistema; el caso concreto puede ser alguna limitación que, posiblemente, no sea modificable hasta el punto de volverse una restricción de operación inamovible (*e.g.* no hay presupuesto para la nueva versión del 'desarrollador de aplicaciones' ni para el **'cableado estructurado'** hasta el año próximo).

* **Salida(s):**

** Modelo de comportamiento: El administrador debe revisarlo con mucho cuidado y detectar si es que se están haciendo supuestos erróneos, para volver a verificar con el área de operaciones de la organización y definir las restricciones del administrador.

** **Restricciones del administrador:** Conjunto de limitaciones (*v.g.* técnicas) inamovibles impuestas al sistema, por ejemplo: *"No hay equipo nuevo o adicional para 'implantar', en su momento, el proyecto. Con lo que se tiene "in situ" se debe poder echar a andar".*

I.6 Generar modelo del sistema acotado.

* **Sugerencia(s):**

** Con toda la información recabada de la 'actividad' uno a la cinco, es conveniente hacer un punto de corte con los usuarios, para traducir y explicar con toda paciencia y de ser el caso, en diversas reuniones, el modelo de comportamiento generado en definitiva, persiguiendo lograr el cabal y total entendimiento del usuario de qué es lo que recibirá efectivamente como producto final, así como realizar los últimos ajustes al modelo. Preferible esto a proseguir trabajando el diseño y la 'implementación' sobre

bases erróneas o a estar generando falsas expectativas; en otras palabras el modelo de comportamiento debe ser fruto del consenso. A este punto de corte se le conoce en la jerga de las áreas de computo como **'congelamiento'** de requerimientos que persigue como consecuencia, 'congelar' también las especificaciones del análisis.

*** Insumo(s):**

**** Modelo de comportamiento definitivo:** Es posible que derivado del establecimiento de las restricciones del administrador se generen una serie de ajustes al modelo en cuestión que, tal vez, impliquen el tener que mover las fechas de compromiso iniciales o incluso, que se tenga que modificar el alcance del sistema o aplicación de cómputo. Corresponde al 'congelamiento' de requerimientos y especificaciones antes mencionados.

*** Salida(s):**

****Modelo de sistema acotado**: Corresponde al modelo que prácticamente se 'implementará' en definitiva, salvo cambios derivados de acotaciones del líder de proyecto. A estas alturas del proyecto, considerando que los documentos del modelo de comportamiento (*v.g.* DD, DFD) han sido revisados suficientemente y contemplando friamente las restricciones al sistema, es posible tomar las últimas decisiones fuertes en la materia, por citar un ejemplo "dramático": *"..., siempre no se desarrollará el sistema para trabajar como paquete standalone, mejor se diseñará para ambiente multiusuario, aunque para esto hay que modificar todos los DFD derivados de la cuarta y de la quinta 'burbujas' del sistema"*.

I.7 Empacar especificaciones

*** Sugerencia(s):**

** La integración coherente de las especificaciones del análisis en un solo paquete requiere de un esfuerzo adicional en términos del cuidado de la calidad del documento respectivo -que debería ser uno de los 'entregables' del sistema-, pues en mucho, la que corresponde a las 'actividades' subsecuentes de esta variante de la metodología de Yourdon y la del producto terminado depende de ella.

*** Insumo(s):**

** Modelo de comportamiento.

** Restricciones del administrador.

*** Salida(s):**

**Modelo de comportamiento definitivo: Corresponde al modelo que se 'implementará'; sin embargo y a pesar del adjetivo "definitivo", el administrador y también el líder del proyecto deben estar preparados -incluso psicológicamente-, para "cambios de último momento... en cualquier momento".

II. DISEÑO

Objetivo

Generar las especificaciones de diseño de la aplicación o sistema de cómputo para la 'implementación' del 'prototipo inicial' y de la aplicación en su conjunto.

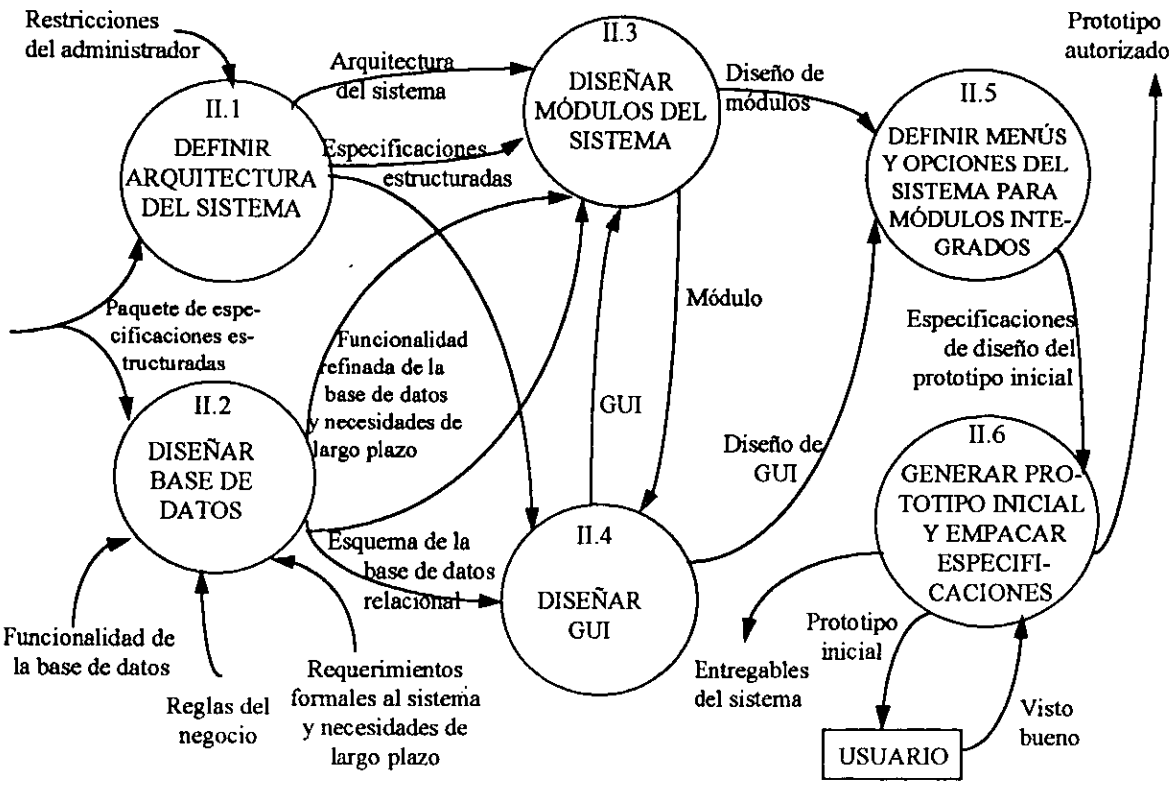


FIGURA 19: DFD DE DISEÑO

Tareas

II,1 Definir la **'arquitectura'** del sistema

*** Sugerencia(s):**

** Para las decisiones correspondientes, lo más conveniente es anticiparse y tener a la mano las mayor cantidad posible de opiniones: de técnicos de otras áreas u organizaciones, de varios proveedores y/o distribuidores, 'configuraciones' diversas de *hardware* y *software* -incluso antes de que el total del análisis esté completo- y, nunca perder de vista las constricciones de operación que estén vigentes a través del tiempo en la organización (*e.g.* presupuesto), en virtud de que puede ser que la recepción física del equipo (*v.g.* el 'servidor'), dependa de consideraciones externas e incontrolables, tales como: política, licitaciones y "grilla" de vendedores voraces, de tiempo, autorización y liberación efectiva del presupuesto, etc., "pegando" todo esto en la definición de la 'configuración' formal del equipo, sin la cual prácticamente es imposible trabajar.

También debe obtenerse información de las innovaciones tecnológicas que proporcionan las diferentes marcas de *hardware* y *software*, pues muchas veces problemas aparentemente irresolubles, se solucionan, por ejemplo, solamente con una actualización del ***'release'*** del *software* -muy a menudo sin costo-.

** En esta 'subactividad' debe especificarse también el **'protocolo de comunicación'** o 'interface' que permitirá conectar a las 'burbujas' de un 'procesador' con otro (*e.g.* vía TCP/IP, vía *'stored procedures'* -respectivamente-).

** Nótese que el punto es la 'arquitectura', no la **'plataforma'** de *hardware* o *software* o la **'topología de la red'**; sin embargo esos elementos, entre otros, entran en juego como componentes de una 'arquitectura', adicional a la propuesta de infraestructura del sistema, que en suma, definen la propia.

A guisa de ejemplo en la materia, se ponen a consideración del lector cinco 'configuraciones' básicas para un ambiente de tipo **'distribuido'**, ilustradas en la figura 20 (página siguiente), que inclusive, ¡es posible entremezclar entre sí!:

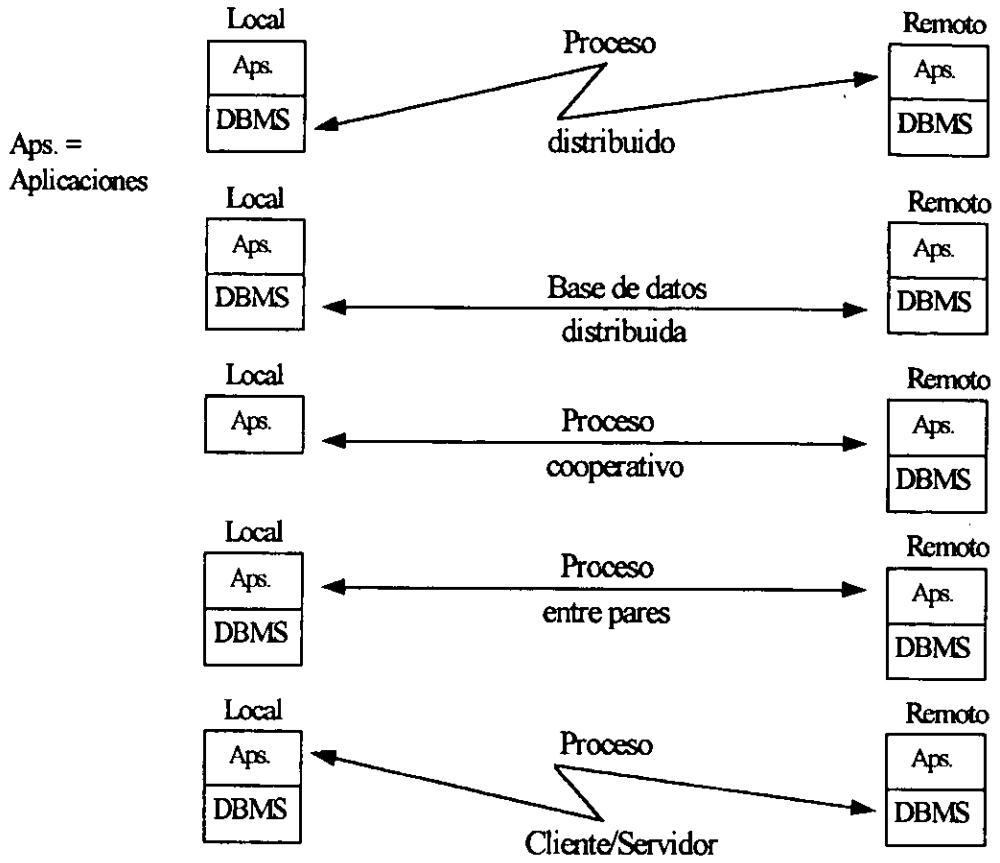


FIGURA 20: ARQUITECTURAS EN UN AMBIENTE DE TIPO DISTRIBUIDO

'Proceso distribuido'.- Cualquier aplicación en cualquier localidad puede interactuar y acceder a datos de la organización, los cuáles son típicamente un repositorio en un sistema integrado 'localmente'.

'Base de datos distribuida'.- La información de la organización está dispersa sobre la red en cualquier tipo de esquema que sea ventajoso para la misma. Algunos datos pueden ser **'replicados'** o copiados en varios ***'sites'***, otros pueden estar en modo de **'copia simple'** o **'primaria'**, todo dependiendo de la organización y de las necesidades de cómputo en las diferentes localidades.

'Proceso cooperativo'.- Conjunto de aplicaciones que comparten recursos y que se ejecutan en concurrencia dependiendo de la disponibilidad de 'facilidades' de cómputo y de su ubicación. El punto es maximizar el uso de los recursos compartidos para, de esta forma, abatir costos.

'Proceso entre pares o *peer to peer*'.- En un contexto de *Client/Server* (*C/S*), permite a sistemas de ***'desktop'*** funcionar como **'clientes'**, tanto como 'servidores' o ambos; un dispositivo puede ser un 'cliente' en una aplicación y un 'servidor' en otra.

'Proceso Cliente/Servidor'.- Diversos aplicaciones o sistemas de *desktop* acceden a 'archivos' y 'bases de datos' en **'servidores remotos'**. El acceso y almacenamiento del *Data Base Managment System* (*DBMS*) se da para soporte de la aplicación cuando así se requiere, por ejemplo para la ejecución de 'rutinas' de **SQL** (Structured Query Language) .

Existen diversas modalidades de arquitectura C/S, entre otras: Múltiples 'clientes'/Múltiples 'servidores', 'Cliente'/'Servidor'/'Servidor', 'Servidores' que en algunas aplicaciones funcionan como 'clientes' de otro 'servidor' destinado para replicación y/o copia de datos -*v.g.* 'bases de datos distribuidas', **'servidores de alta disponibilidad'**, **'servidores en espejo'**-, etc.

Por cuestión de costos y como todos los 'servidores' son importados -y pagados en dólares-, por lo general, se selecciona una 'configuración' de Múltiples 'clientes'/ 'Servidor' único para 'implementar' sistemas -abatiendo de esta forma presupuesto-; esta modalidad de 'proceso' se ilustra en la figura 21 (página siguiente):

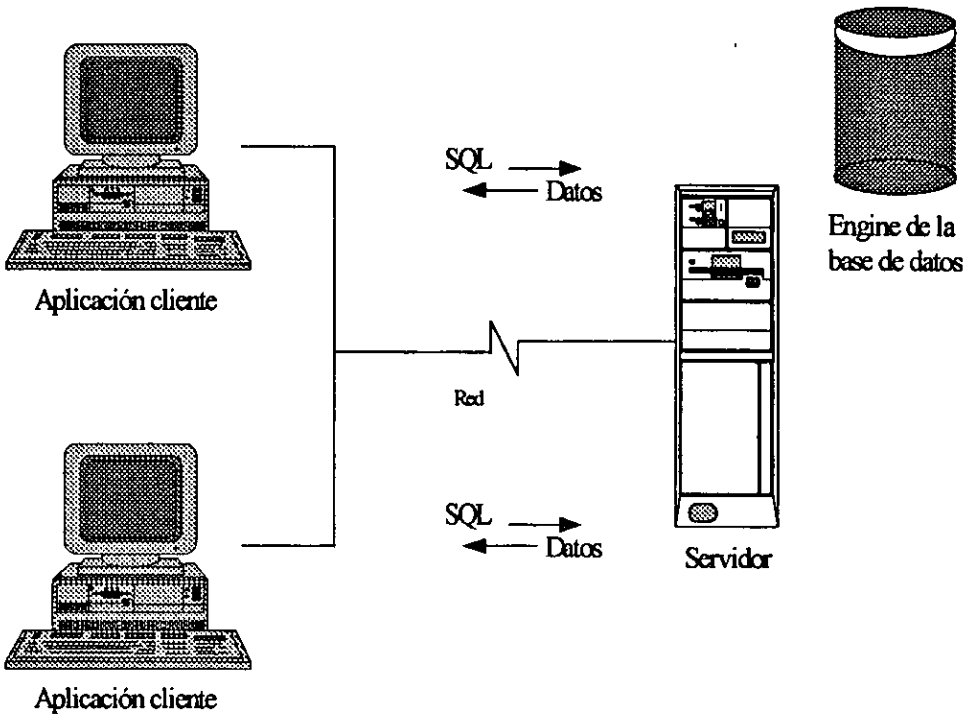


FIGURA 21: MODELO DE MÚLTIPLES CLIENTES / SERVIDOR ÚNICO

** Como antes se comentó, es pertinente considerar y evaluar cuidadosamente las actuales bondades de los OS -'multiprocesos'- y ambientes de trabajo, que permiten la ejecución de diversas 'burbujas' de nivel bajo, de forma 'concurrente' en un mismo 'procesador'. Asimismo debe considerarse de lo más relevante la definición de qué partes del sistema deben ejecutar los 'programas cliente' y qué partes el 'servidor', aunque siempre será posible refinar esto en razón de su desempeño para, posiblemente, cambiar las especificaciones iniciales del "reparto de 'tareas'" a fin de intercambiar cargas de trabajo entre los 'programas cliente' y el 'servidor'.

** Resulta muy conveniente integrar la 'funcionalidad' y 'facilidades' proporcionadas por herramientas CASE al desarrollo de proyectos de sistemas o aplicaciones de cómputo, particularmente en la 'actividad' de diseño, como antes se ilustró en la figura 18 (página 69).

*** Insumo(s):**

** Paquete de 'especificaciones estructuradas'.

** Restricciones del administrador.

*** Salida(s):**

** 'Arquitectura' del sistema.

II.2 Diseñar 'base de datos relacional'

*** Sugerencia(s):**

** Existen en el mercado herramientas que permiten generar los diagramas de los **'esquemas de base de datos relacionales'**, así como las estructuras físicas de las mismas en los dispositivos magnéticos establecidos al efecto.

** Para diseñar en forma una 'base de datos relacional', se requiere experiencia y conocimientos sólidos en la materia, en el lenguaje SQL y sobre el producto de *software* que se seleccione para instrumentar el **'servidor de base de datos relacional'** Cabe comentar que un error aquí puede tener repercusiones de niveles insospechados, por lo que esta parte del diseño debe realizarse con la mayor calidad posible, sin escatimar el tiempo y los recursos necesarios al respecto.

*** Insumo(s):**

** Paquete de 'especificaciones estructuradas'.

** 'Funcionalidad' de la 'base de datos relacional'.

** 'Reglas del negocio'.

** Requerimientos formales al sistema y necesidades de largo plazo.

*** Salida(s):**

** Necesidades de largo plazo en y para la 'base de datos'.

** **Funcionalidad refinada de la 'base de datos relacional':** Contiene genéricamente la misma información que el documento de 'funcionalidad' de la 'base de datos', pero ya corregida, aumentada y ajustada por el grupo de diseñadores expertos en la materia, adicionándole sus puntos de vista técnicos, las 'miniespecificaciones de proceso' del caso para efectos de 'programación' del 'servidor de base de datos relacional', debiendo discernir en esta 'actividad' entre lo que es viable

instrumentar de inmediato y lo que debe postergarse en el proyecto; incluso, al integrar esta información con las necesidades de largo plazo, es posible establecer que 'tablas' y/o 'atributos' se contemplan en el 'esquema', pero no serán objeto de ningún desarrollo, 'implementación' o actualización sino hasta, tal vez, una versión posterior del sistema.

** **'Esquema de la base de datos relacional':** Diagrama que interrelaciona sus 'tablas' y describe sus **'llaves'**, 'atributos', **'índices'** y *'constraints'*. Adicionalmente se debe considerar e indicar que 'programación' se debe desarrollar, en su caso, en el 'servidor de base de datos relacional' (*v.g. 'stored procedures'*, manejo de **'integridad referencial'**, **'vistas'**, *'triggers'*). En 7.1 de este documento se pueden visualizar varios 'esquemas'.

II.3 Diseñar 'módulos' del sistema

* **Sugerencia(s):**

** Para no caer en discusiones bizantinas, se considera que de una forma o de otra los 'módulos' están conformados por un grupo de 'líneas de programación', sitas en uno o varios 'archivos' y/o en varios equipos (*v.g.* PC, 'servidores'), que ejecutan 'tareas' de forma bien definida, que están bien acotados, que de alguna forma son "contiguos" y, que tienen un nombre simple por el cual puede ser referenciados como un 'módulo'.

** Es conveniente establecer en forma *top-down*, una jerarquía de 'módulos' de 'programación' e 'interfaces' entre los mismos a fin de poder 'implementar' apropiadamente las especificaciones del análisis, posiblemente a través de 'cartas estructuradas' de 'programación' -representación gráfica de 'módulos' jerarquizados correspondientes a los DFD nivelados- o a través de las mismas 'ventanas' de la GUI en enlace con diversos diagramas de flujo de datos, que indican entre otros aspectos cuál precede a cuál, cuál depende de cuál; la jerarquización mencionada debe estar dada, en principio, por la nivelación de los propios DFD.

** El diseño en cuestión debe realizarse en paralelo y conjuntamente al de la GUI, persiguiendo un efecto de retroalimentación continua, sea desarrollado por el mismo grupo de personas o por diferentes grupos de trabajo, caso en el que se requerirá de un esfuerzo adicional de coordinación.

** Una herramienta útil para el diseño es el uso de diagramas de Warnier, que sirven de 'interface' entre los analistas y los diseñadores, así como posteriormente entre los diseñadores y los 'implementadores', siendo posible de esta forma reflejar en y a través de ellos, en forma ordenada, los ajustes y retroalimentaciones necesarias entre las 'actividades' mencionadas. Debe concebirse como un cíclo del tipo: cambios en los 'módulos' --> cambios en los diagramas de Warnier--> ajustes a la GUI del 'prototipo'.

*** Insumo(s):**

** 'Arquitectura' del sistema.

** Paquete de 'especificaciones estructuradas'.

** 'Esquema de base de datos relacional'.

** Diseño de la GUI.

*** Salida(s):**

** Diseño de los 'módulos'.

II.4 Diseñar GUI

*** Sugerencia(s):**

** El diseño en cuestión debe realizarse en paralelo y conjuntamente al de los 'módulos', persiguiendo un efecto de retroalimentación continua, sea desarrollado por el mismo grupo de personas o por diferentes equipos de trabajo, caso en el que se requerirá de un esfuerzo adicional de coordinación.

** Una herramienta útil para el diseño es el uso de diagramas de Warnier, que sirven de 'interface' entre los analistas y los diseñadores, así como posteriormente entre estos últimos y los 'implementadores', siendo posible de esta forma reflejar en y a través de ellos, en forma ordenada, los ajustes y retroalimentaciones necesarias entre las 'actividades' mencionadas. Debe concebirse como un cíclo del tipo: cambios en la GUI --> cambios en los diagramas de Warnier--> ajustes a los 'módulos' del 'prototipo'.

** Una buena guía de referencia para diseñar la GUI es contemplar el diseño de las correspondientes a los diversos productos comerciales conocidos para ambiente Windows.

*** Insumo(s):**

** Paquete de 'especificaciones estructuradas'.

** Esquema de la 'base de datos relacional'.

** Diseño de los 'módulos'.

*** Salida(s):**

** Diseño de la GUI.

II.5 Definir 'menú' y opciones del sistema para los **'módulos integrados' ('módulos' y GUI).**

*** Sugerencia(s):**

** Debe tenerse mucha atención y cuidado en el flujo de retroalimentación entre el diseño de la GUI y los 'módulos', pues es el momento justo todavía, para valorar y decidir cambios "al vuelo" en el esquema de la 'base de datos relacional', sin golpear gravemente a los tiempos del proyecto. Este tipo de cambios para el líder del proyecto deben ser considerados como "focos rojos", pues denotan fallas en el análisis y deberían implicar una revisión *ipso facto* de la mayoría -si no es que todo- del trabajo ahí realizado.

** Una referencia para diseñar los 'menús' y opciones del sistema es contemplar el diseño de los correspondientes a los diversos productos comerciales, considerando la complejidad que ya se maneja, derivado de la integración de la documentación de los 'módulos' (*e.g.* 'programas cliente', 'programas' del 'servidor de la base de datos relacional', GUI, 'interfaces') y que todo lo anterior, debe ser 'transparente' al usuario y tan simple de usar, como al dar un "click" en un botón de la GUI.

*** Insumo(s):**

** Diseño de la GUI.

** Diseño de los 'módulos'.

*** Salida(s):**

** **Especificaciones de diseño del 'prototipo inicial':** Conjunto de información contenida en la documentación de los 'módulos' integrados, suficiente e indispensable, para iniciar la 'codificación' del 'prototipo inicial'.

II.6 Generar 'prototipo inicial' y empacar especificaciones.

*** Sugerencia(s):**

** Coloquial y tradicionalmente lo que se muestra hasta aquí, es a lo que se le conoce como "presentación de avance" al usuario y, generalmente, consiste de un espectáculo de "pantallazos" del sistema (que en algunas organizaciones justamente pretenden impresionar al usuario). En el caso de la variante, se pretende que el 'prototipo' tenga integrada incluso alguna 'funcionalidad' mínima de 'base de datos relacional' y parte de su 'ayuda' como aspectos relevantes, sin excluir aquéllos otros que el administrador considere deban ser contemplados.

** Dependiendo de la complejidad comprendida, acuerdos previos entre los usuarios y sus directivos con el administrador y, en virtud de la posibilidad de diseñar e 'implementar' 'módulos' de forma 'incremental', es posible presentar 'prototipos' de cada 'módulo' de vez en vez o bien, del 'prototipo' "completo" en su conjunto.

** La integración coherente del paquete de especificaciones de diseño requiere de un alto cuidado de la calidad en el documento respectivo, que debería ser uno de los 'entregables' del sistema. Es muy conveniente verificar de forma intermitente y continua a lo largo del proyecto, la consistencia entre el documento en cuestión con las 'especificaciones estructuradas' derivadas del análisis (*e.g.* lista de eventos, requerimientos), para estar lo más seguro de que el sistema por desarrollar sea el que espera el usuario. A mayor calidad del diseño, menor tiempo y esfuerzo para su 'implementación'.

** De no obtener el visto bueno del usuario para el 'prototipo', se debe "echar toda la carne al asador" para ajustarlo -tanto como la documentación del caso-, a fin de conseguir la aprobación en el menor tiempo posible y poder proseguir con la 'implementación' sobre bases firmes.

*** Insumo(s):**

** Especificaciones de diseño del 'prototipo inicial' ('módulos integrados').

*** Salida(s):**

** 'Prototipo inicial' autorizado por el usuario.

III. 'IMPLEMENTACIÓN'

Objetivo

Generar el sistema integrado a través de la generación e integración del 'código' de los 'módulos' y diversas pruebas al 'prototipo'.

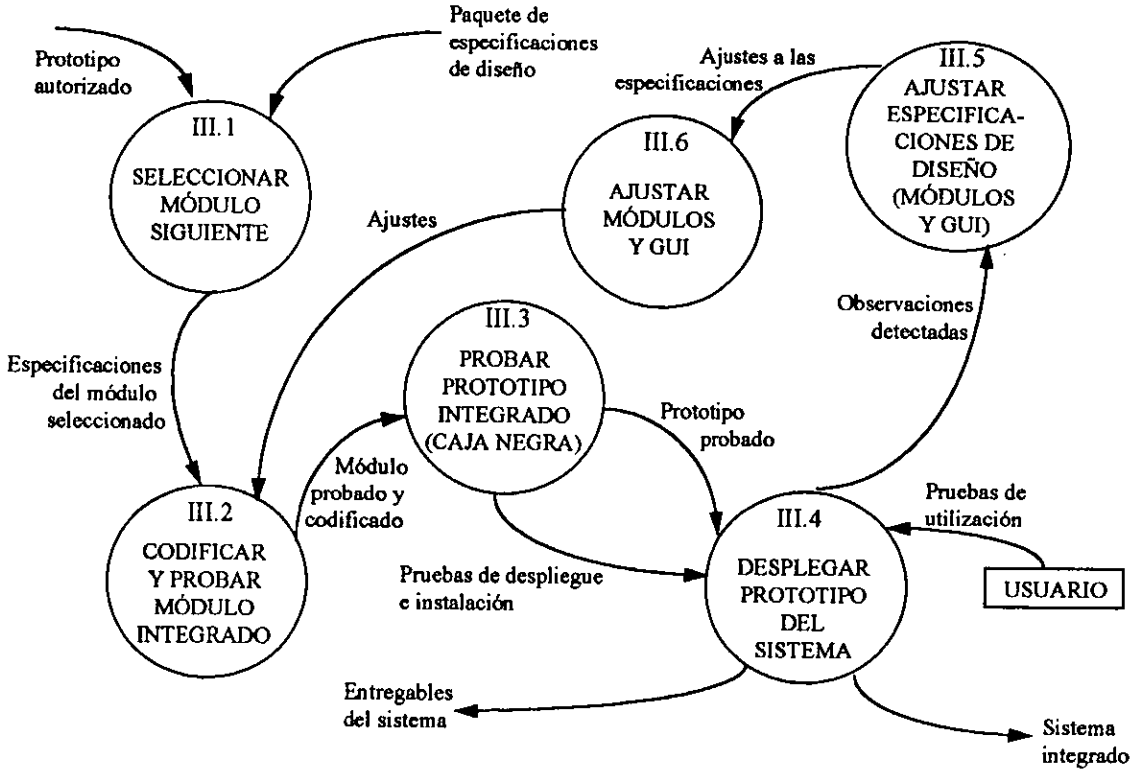


FIGURA 22: DFD DE "IMPLEMENTACIÓN"

Tareas

III.1 Seleccionar 'módulo' siguiente.

*** Sugerencia(s):**

** Esta es una 'actividad' obvia pero no por eso irrelevante: Consiste en decidir el orden en que los 'módulos' del 'prototipo' serán 'implementados', integrados y probados, de forma 'incremental', donde 'módulos' de mayor jerarquía deberían ser elaborados antes que otros de jerarquía menor; empero la decisión recae enteramente en el administrador, con base en su su buen juicio y experiencia, compromisos

establecidos, **'código reutilizable'**, etc. No es imposible desarrollar dos o tres 'módulos' en forma paralela -en función de los recursos disponibles y de la complejidad del asunto-, pero la sugerencia es desarrollar un 'módulo' a la vez, sobretodo cuando se hace frente a equipamiento y/o *software* no empleado o probado anteriormente o, a un nuevo grupo de colaboradores.

*** Insumo(s):**

** Paquete de especificaciones del diseño.

** 'Prototipo' autorizado.

*** Salida(s):**

** **Especificaciones de diseño del 'módulo' seleccionado**: Documento con la información mínima indispensable para 'codificar' el 'módulo' seleccionado en su totalidad.

III.2 'Codificar' y probar 'módulo integrado'.

*** Sugerencia(s):**

** Dependiendo de la complejidad del 'módulo' y de la calidad del trabajo de los recursos humanos, será necesario darle un seguimiento continuo, recurrente o periódico a esta 'subactividad', así como graduar el rigor de las pruebas -"internas" del grupo de desarrollo- por aplicar: *"Al ojo del amo engorda el caballo"*.

** Siempre resulta conveniente **'comentar'** el 'código' de los 'programas fuente' (al efecto se presenta una propuesta en el Anexo 2). Lo anterior no excluye la posibilidad de 'comentar' y aclarar el funcionamiento y la lógica de las **'rutinas'** de los 'programas', de los propios 'programas' y sus **'bibliotecas'** (más conocidas como 'librerías'), etc, generados para su ejecución en los **'procesos cliente'**, según el lenguaje de 'programación' o 'desarrollador de aplicaciones' utilizado o, en el 'servidor' de datos -según sea necesario-, dependiendo de su legibilidad.

*** Insumo(s):**

** Especificaciones de diseño del 'módulo' seleccionado.

*** Salida(s):**

** 'Módulo codificado'.

III.3 Probar 'prototipo' integrado (**'caja negra'**)

*** Sugerencia(s):**

** Este tipo de pruebas son independientes de aquéllas que se mencionan en la 'actividad' relativa a la generación de pruebas de aceptación y control de calidad.

** Las pruebas que se refieren aquí son conducidas típicamente por el líder del proyecto y tienen la característica de ser de **'caja negra'**, "internas y locales" al grupo de desarrollo; pueden hacerse manualmente; sin embargo se debe asegurar llevar un buen registro de como se realizaron, de modo que puedan repetirse, en caso necesario, de manera confiable en iguales condiciones. Nada de que: *"ayer si funcionó y hoy que se agregó el módulo X, ya no"*.

** La connotación de 'caja negra' hace referencia a pruebas de 'funcionalidad' del 'prototipo' del tipo: entrada--> 'caja negra'--> salida y, se deben realizar continuamente en cada paso del 'desarrollo iterativo' y cada vez que se integre un 'módulo' al 'prototipo'. Comprende entre otros rubros la captura de valores de entrada apócrifos y la verificación de las salidas adecuadas, persiguiendo la localización de ***'bugs'***. De forma equivalente debe revisarse, conjuntamente a la integración de los 'módulos' del 'programa cliente', la integración de la 'programación' del 'servidor de base de datos relacional' montado tal vez, en un 'servidor' de UNIX -cuando esto aplique-, definiendo para el caso las pruebas de tipo **ACID** por aplicar a las **'transacciones'**, entendidas como unidades lógicas de 'programación' o 'tareas' que comprenden las instrucciones *insert, update, delete*. Es pertinente aclarar que ACID es un acrónimo proveniente de *Atomicity, Consistency, Isolation, Durability*:

*** ***Atomicity*** (atomicidad o integridad): Si una 'transacción' hace un cambio a un recurso compartido, debe darse en términos de todo o nada, es decir, si una 'transacción' se aborta el estado de la 'base de datos relacional' debe quedar como estaba antes de la 'transacción'.

*** ***Consistency*** (consistencia): Si una 'transacción' se aborta, debe regresar los datos a un estado válido previo; en otras palabras, una 'transacción' debe transformar y pasar a una 'base de datos relacional' solamente de un estado consistente a otro estado consistente.

*** ***Isolation*** (aislamiento): Una 'transacción' debe ser 'transparente' a otras 'transacciones' -y viceversa- hasta que sea completada.

*** ***Durability*** (durabilidad): Toda vez que una 'transacción' es concretada (*"committed"*), sus efectos sobre el recurso de datos -la 'base de..'-, debe permanecer inalterado, descartando así posibles fallas posteriores del cíclo del 'proceso' completo.

** Paralelamente a las pruebas de 'caja negra', debe revisarse por parte del líder del proyecto el apego del grupo de trabajo a las especificaciones de diseño en el 'código' generado.

*** Insumo(s):**

** 'Módulo codificado'.

*** Salida(s):**

** 'Prototipo' probado.

** Pruebas de 'instalación' y 'despliegue': Se debe asegurar, previamente a presentaciones con los usuarios, que funciona la 'instalación' y 'despliegue' del 'prototipo' en varios tipos de *hardware* y 'configuraciones' posibles, intentando crear situaciones de error (*v.g.* discos llenos, saturación del **'log'** de 'transacciones'). También se debe verificar que todos los 'archivos' queden en el lugar adecuado con sus atributos correctos, así como que los equipos (PC, 'servidores') y *software* estén **'puestos a punto'** y 'configurados' correctamente. Ni que decir de las comunicaciones.

III.4 'Desplegar' 'prototipo' del sistema

*** Sugerencia(s):**

** Se debe hacer énfasis en el proceso de retroalimentación con los usuarios, tanto para perfeccionar al 'prototipo' como para detectar carencias y en su caso, derivar los ajustes necesarios al diseño integrado de los 'módulos' y la GUI, para su 'implementación' en el 'prototipo' completando así el proceso de 'desarrollo iterativo'.

*** Insumo(s):**

** 'Prototipo' probado.

** Pruebas de instalación y 'despliegue'.

*** Salida(s):**

** Pruebas de 'utilización': Los usuarios pueden y deben proporcionar información sobre si el flujo de 'tareas' de la aplicación es o no intuitivo *e.g.*: ¿Puede encontrar con facilidad el camino a través de los 'menús', 'diálogos' y 'ventanas' de la aplicación?. ¿Están las cosas donde el usuario lo espera?: este es un buen momento para discernir en donde se debe agregar 'ayuda' sensible al contexto.

III.5 Ajustar especificaciones del diseño (Módulos y GUI)

*** Sugerencia(s):**

** Dependiendo de la calidad del 'prototipo inicial', serán necesarias más o menos ajustes al diseño, 'prototipo' y pruebas, para lo cual es necesario establecer un control y una cronología sobre los cambios efectuados a la aplicación, que deben ser anotados en la documentación correspondiente.

*** Insumo(s):**

** **Observaciones detectadas:** Conjunto de notas con sugerencias y peticiones de los usuarios sobre cambios al 'prototipo'.

*** Salida(s):**

** **Ajustes a las especificaciones:** Conjunto de notas al calce en las especificaciones del 'módulo' integrado, realizadas por el líder de proyecto, a fin de proceder a la 'implementación' de los mismos en el 'prototipo'. Aquí debe ser muy eficiente su control al efecto y muy clara la comunicación entre él y los demás miembros de su equipo para ajustar estrictamente lo que se indique -ni más ni menos-. Posterior e invariablemente, se debe actualizar formalmente el documento 'entregable' de diseño.

III.6 Ajustar 'módulos' y GUI

*** Sugerencia(s):**

** Dependiendo de la calidad del 'prototipo inicial', serán necesarias más o menos rondas de rediseño, re'codificación' del 'prototipo' y pruebas al 'implementar' los ajustes y cambios documentados en el 'prototipo' a través de su evolución.

*** Insumo(s):**

** Ajustes a las especificaciones de los 'módulos' y de las GUI correspondientes.

*** Salida(s):**

** **Especificaciones del 'módulo' integrado:** Integración y actualización de las especificaciones procedentes realizadas por el líder de proyecto en el documento 'entregable' de diseño, así como para actualizar al grupo de 'implementación'.

** **Sistema integrado**: 'Prototipo' prácticamente listo para pasar a las 'actividades' de aplicación de pruebas y control de calidad.

IV. GENERACIÓN DE PRUEBAS DE ACEPTACIÓN Y CONTROL DE CALIDAD

Objetivo

Plantear las pruebas al sistema o aplicación de cómputo para, con base en ellas, efectuarle el control de calidad y obtener su aceptación.

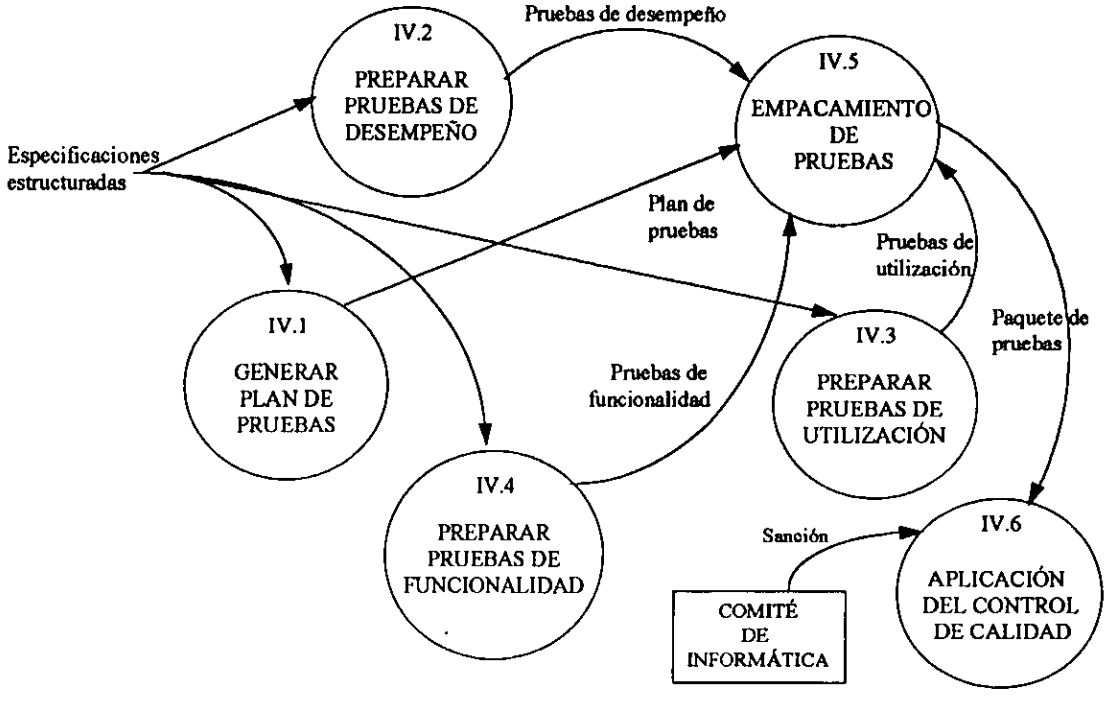


FIGURA 23: DFD DE PRUEBAS DE ACEPTACIÓN Y CONTROL DE CALIDAD

Tareas

IV.1 Generar plan de pruebas

*** Sugerencia(s):**

** Esta 'actividad', generalmente y según la secuenciación de la metodología, se realiza hacia el final del proyecto cuando también, generalmente, el plazo para su entrega está ya muy consumido. No por esta circunstancia deja de ser importante la generación de las pruebas de aceptación y el ejercicio del control de calidad, recordando que todos los errores que no surjan durante este tipo de 'actividades', invariablemente, surgirán ante el usuario.

** Nombrar a una persona o grupo de personas como responsable(s) de la generación del plan de pruebas, sin relación alguna con el grupo de 'implementación', con la intención expresa de encontrar errores en el sistema o aplicación de cómputo.

** Considerar criterios de evaluación precisos, así como los recursos materiales, de personal y "tiempos" necesarios al efecto.

** Una posibilidad de trabajar el plan de pruebas establecido previamente, consiste en intercambiar los roles de desarrollador y aplicador del control de calidad entre diferentes áreas organizacionales de sistemas (*e.g.* la Subdirección "Y" aplica el control a un proyecto 'implementado' por la Subdirección "X" -y viceversa-)

*** Insumo(s):**

** 'Especificaciones estructuradas'.

*** Salida(s):**

** **Plan de pruebas:** Conjunto de especificaciones que contienen el tipo de pruebas por realizar, el orden en que se probarán los 'módulos' y la 'funcionalidad' de cada uno de ellos, así como la aplicación en su conjunto -*e.g.* según la lista de eventos, las fechas y condiciones ambientales en que se deben realizar-.

IV.2 Preparar pruebas de desempeño

*** Sugerencia(s):**

** Requerimientos adicionales de lógistica indispensables a considerar para las pruebas son: claridad en los lineamientos generales para su ejecución a todos los recursos humanos involucrados; por ejemplo: *"no se vale comentar nada con personas ajenas al proyecto y mucho menos con las del área del otro departamento"*; contar con criterios de evaluación precisos, disponibilidad suficiente de recursos materiales, así como personal y "tiempos" involucrados (*e.g.* **'tiempos de procesador'**).

** Deben armarse muchas pruebas de desempeño que deben orientarse sobretodo a la capacidad del *software* de la 'base de datos relacional' y de las comunicaciones, simulando eventos críticos como exceso de tráfico en la red o crecimiento explosivo de 'registros'. Una opción es restringir el ambiente del sistema, en el sentido de ejecutarlo con *hardware* de menor capacidad al requerido; por ejemplo, en máquinas PC con 8 MB de memoria, en lugar de equipos con 16 MB.

** El sistema debe probarse también durante esta 'actividad' en *hardware* de capacidad equivalente y 'configuración' igual a aquél con el que cuenta el área donde se pretenda realizar la 'implantación'.

*** Insumo(s):**

** 'Especificaciones estructuradas'.

*** Salida(s):**

** **Pruebas de desempeño**: Conjunto de procedimientos descriptivos de cada prueba de desempeño del sistema, incluyendo datos prueba (*v.g.* un 'archivo' con 180,000 'registros') y resultados esperados (*e.g.* recuperación de datos en menos de 5 segundos).

IV.3 Generar pruebas de utilización

*** Sugerencia(s):**

** Se deben preparar datos prueba y ejemplos, conforme a los 'estándares' establecidos al efecto (ver Anexo 3), tales como orden de presentación de 'ventanas', 'diálogos', 'mensajes', etc.

** Las pruebas deben prepararse con base en la experiencia recabada con los usuarios cuando por su parte, durante la 'implementación', fueron evaluando los 'prototipos', sin perder de vista que en esta 'actividad' la comprobación sobre qué tan simple resulta usar la aplicación de cómputo la realizarán personas ajenas a los usuarios y a los desarrolladores del *software*.

*** Insumo(s):**

** 'Especificaciones estructuradas'.

*** Salida(s):**

** **Pruebas de utilización**: Conjunto de procedimientos descriptivos de las pruebas para el 'procesamiento' correcto de datos + Especificación de prueba de los datos + Resultados esperados.

IV.4 Generar pruebas de 'funcionalidad'

*** Sugerencia(s):**

** Se deben preparar datos prueba, conforme a estándares de control establecidos al efecto, tales como intervalos de confianza, cifras de control, etc. Asimismo deben contemplarse datos prueba fuera de esos estándares de control.

*** Insumo(s):**

** 'Especificaciones estructuradas'.

*** Salida(s):**

** **Pruebas de 'funcionalidad':** Conjunto de procedimientos descriptivos de prueba para 'procesamiento' correcto de datos, contemplando: Datos incorrectos + Especificación de prueba de los datos + Mensajes de error esperados o comportamientos opcionales del sistema.

IV.5 Empacamiento de pruebas

*** Sugerencia(s):**

** Se debe ser lo más explícito y detallado posible en la descripción de las pruebas en virtud de que, en principio, personal lo más ajeno al grupo de 'implementación' las debe aplicar en su momento en la 'actividad' de control de calidad.

*** Insumo(s):**

** Plan de pruebas.

** Pruebas de 'desempeño'.

** Pruebas de 'utilización'.

** Pruebas de 'funcionalidad'.

*** Salida(s):**

** **Paquete de pruebas**: Documento integrado para efectos de entrega a la instancia responsable de la aplicación del control de calidad, que invariablemente debe reportar los resultados de las mismas, entre otras instancias, al comité de informática de la organización.

IV.6 Aplicación del control de calidad

*** Sugerencia(s):**

** Para aplicar las pruebas planteadas para el sistema en las 'actividades' precedentes, se recomienda su ejecución por parte de un grupo creado al efecto, preferentemente de parte del comité de informática de la organización.

** El control de calidad sobre la aplicación de cómputo o sistema puede comprender, entre otros rubros y previo acuerdo al efecto, aspectos adicionales sobre la calidad de:

*** su documentación 'entregable',

*** su 'arquitectura' y diseño,

*** su 'código fuente' -para verificar cumplimiento de 'estándares'-,

*** su seguridad y posibilidades de auditabilidad (*v.g. passwords*).

empero, en todo caso, como se ha citado en el marco teórico de est documento: *"La especificación es la prueba de aceptación"* y no otros aspectos.

*** Insumo(s):**

** Paquete de pruebas.

** Sistema integrado.

*** Salida(s):**

** Sistema aceptado.

V. CONVERSIÓN DE 'BASES DE DATOS'

Objetivo

Contar con la 'base de datos relacional' convertida para operar el sistema.

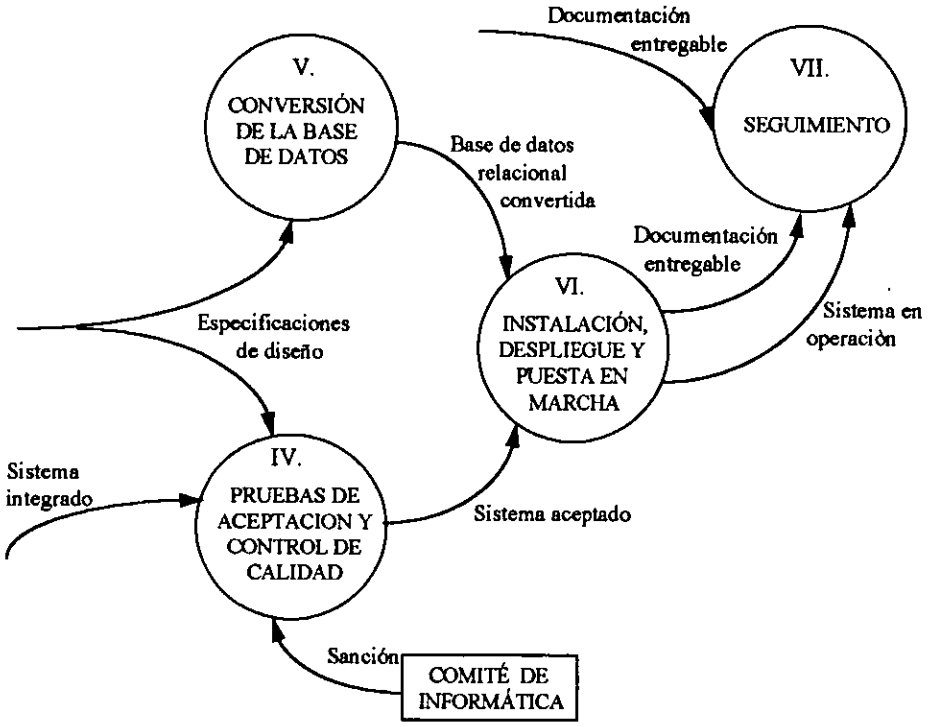


FIGURA 24: ACTIVIDADES IV A VII DE LA VARIANTE DE LA METODOLOGÍA DE YOURDON

Tareas

Convertir la 'base de datos'.

*** Sugerencia(s):**

** Cuando los datos están disponibles en medios magnéticos, pero en otra 'plataforma' diferente a aquélla en que se 'implantará' el sistema, debe haberse presupuestado el costo respectivo para asignar un grupo de personas exprofeso a la 'actividad' en cuestión, para que trabaje en coordinación con los usuarios y poder estar seguros de la validez de la transformación de los datos. Cuando simplemente no hay datos, una posibilidad es generarlos a través del propio sistema ya 'implantado'.

*** Insumo(s):**

** 'Base de datos' existente.

*** Salida(s):**

** 'Base de datos relacional' convertida.

VI. INSTALACIÓN, 'DESPLIEGUE' Y PUESTA EN MARCHA

Objetivo

Poner en marcha el sistema o aplicación de cómputo (ver figura 24).

Tareas

Instalar y desplegar el sistema, así como hacer que "arranque".

*** Sugerencia(s):**

** El *quid* del asunto es lograr la administración del cambio, dependiendo invariablemente, del entorno y de la envergadura propia del sistema, de forma tal que el mismo sea integrado en la operación de los usuarios y que realmente sea aceptado por los beneficios que les proporcione. Esto solamente se logra cuando el nuevo sistema o aplicación de cómputo es benéfico por si mismo.

** Muchas de las 'actividades' para la 'implantación' deben estar previstas desde las pruebas de 'despliegue' e instalación de la 'actividad' de 'implementación', lo que facilitará en mucho los trabajos en cuestión. Un ejemplo es prever con precisión la 'puesta a punto' de los equipos (*v.g.* 'servidores') y el **'tuning'** del *software* (*e.g.* 'servidor de base de datos'), sobretodo si depende de recursos externos -generalmente incontrolables e impredecibles- tales como proveedores o despachos de consultoría.

*** Insumo(s):**

** 'Base de datos' convertida.

** Sistema aceptado.

*** Salida(s):**

** Sistema en operación o producción.

VII. SEGUIMIENTO

Objetivo: Mantener en operación el sistema (ver figura 24 -página 96-).

Tareas

Realizar el seguimiento y mantenimiento del sistema con base en su documentación 'entregable'.

*** Sugerencia(s):**

** En la medida que se cuente con la documentación 'entregable' del sistema y dependiendo de su calidad, será necesario más o menos esfuerzo para su sostenimiento en producción.

** Cada vez que se realicen modificaciones al 'código' deben asentarse de ser el caso, los cambios en la documentación correspondiente, sobretodo en su diseño; asimismo, deben escribirse invariablemente, los comentarios pertinentes en los 'programas fuente' sobre los cambios realizados.

** Debe considerarse no perder contacto con los usuarios e investigar y confirmar, periódicamente (*v.g.* cada tres meses, cada quincena), el buen funcionamiento y operación del sistema.

** En esta 'actividad' es posible definir con toda precisión los insumos necesarios para realizar, en su caso, la capacitación al personal para el sostenimiento del sistema.

** Previa evaluación de problemas no contemplados o no resueltos por la aplicación de cómputo y contrastando con las necesidades de largo plazo, es posible definir la forma de dar mantenimiento al mismo o bien, decidir la elaboración de una nueva versión del mismo o tal vez, hasta un sistema nuevo.

*** Insumo(s):**

** Documentación 'entregable' del sistema.

** Sistema en operación (incluye datos en producción).

** Necesidades de largo plazo.

5.3. RESUMEN

Entre tantas otras variantes y/o "ajustes" de la metodología de Yourdon, la aqui expuesta aporta como consecuencia de su aplicación una constante que debe lograrse y mantenerse en el más alto nivel de organizaciones amplias: que la documentación 'entregable' de los sistemas forme parte de su acervo; el simple planteamiento anterior establece la pauta para la construcción de un esquema normativo formal al efecto en aquéllas donde no exista y en un reforzamiento de las políticas en la materia para aquéllas otras donde su aplicación es "letra muerta".

Para lograr lo anterior, forzosamente y para no trabajar de más ("ley del mínimo esfuerzo"), se tiene que homogeneizar la elaboración de los sistemas a través de la 'estandarización' de sus componentes (*e.g.* 'módulos' completos), documentación, 'funcionalidad', forma y presentación, lo cual facilita su control, coordinación, enlace y, a su vez, genera -necesariamente-, consistencia informática institucional y, en el óptimo de los casos, facilita el flujo de información entre las áreas organizacionales, mediante el establecimiento de uniformidad en la misma.

En lo relativo a aspectos estrictamente de cómputo, aporta el intercalamiento de técnicas de RAD y diagramación de Warnier al desarrollo de aplicaciones de cómputo que manipulan 'bases de datos relacionales' y, en este sentido, es que simplemente se le puede considerar como una actualización de la misma, pero al adicionarle en forma muy pragmática herramientas -hoy tan en boga- tales como un 'desarrollador de aplicaciones' o un CASE para el diseño de la 'base de datos relacional', es cuando realmente pueden apreciarse sus bondades en términos de la facilitación de la coordinación de trabajos en la materia y en una verdadera disminución de tiempos de desarrollo.

Empero, todo lo anterior implica un mínimo de *know how* para que verdaderamente funcione, *i.e.*, según se aplica la frase en diversos deportes: "hay que tener cancha y oficio" suficientes. Esto solamente se adquiere trabajando en la materia... intensamente.

6. AMBIENTE PARA LA "IMPLEMENTACIÓN" DEL GAR

6.1 ANTECEDENTES

El Generador Automático de Reactivos, en adelante **GAR**, tiene su historia: Cuando terminé los créditos de Actuaría en la Facultad de Ciencias de la UNAM en febrero de 1982, decidí obtener mi título profesional en seis meses para tenerlo justo al cabo de 8 semestres de haber iniciado la carerra. No lo hice. Volví a retomar el asunto en diciembre de 1985 cuando fuí becado para desarrollar un sistema por la en ese entonces, Dir. Gral. de Servicios de Cómputo para la Administración de dicha Institución, pues a instancia de la Unidad de Cómputo Instrumentación e Informática de la Facultad de Psicología (UCII), también de la UNAM, surgió el planteamiento sobre la necesidad de contar con *"un sistema de cómputo que permitiese generar exámenes de manera automatizada y mantener actualizada la información que los sustenta"*. Por tal motivo tuve el honor de conocer al Lic. Daniel Zarabozo Enriquez de Rivera, quién dirigió inicialmente los trabajos en la materia y quién, de hecho, era mi director de tesis, pero como él tuvo la oportunidad de radicar fuera de la ciudad y yo dedique más tiempo al trabajo que a este asunto no concluí, formalmente, los trabajos bajo su dirección. Sin embargo con este documento y la aplicación lista para instalarse -aunque sea a destiempo-, espero concretar el compromiso de terminar este proyecto que con mi familia, con la Institución y con él adquirí, pues considero que la necesidad planteada aún no se satisface.

Aquí debo hacer mención de una frase trillada pero no por ello carente de substancia en lo relativo a la informática: *"la única constante es el cambio"* y todo profesional que se dedique a ella debe estar preparado -a diferencia de otras áreas profesionales-, a asimilar y aprovechar la rapidez de esos cambios, enfrentándose cotidianamente a este reto, considerando a aquél que no estudia y no se actualiza al mismo ritmo está irremediablemente fuera del negocio.... "a la de ya" o en el corto plazo. Por esto y como el asunto del desarrollo del GAR pretendo describirlo en forma anecdótica, cada vez que sea posible, mencionaré como estaba el entorno técnico inicialmente y como repercutieron los cambios

globales de tecnología a través de diez años, en cambios particulares para la aplicación aquí expuesta. Así también, describiré los ajustes a los planteamientos y a las concepciones iniciales del generador a través del tiempo, hasta llegar a redefinir la aplicación de cómputo actual; *las anecdotas técnicas las señalaré con letra de este tamaño y tipo.*

Así pues, durante 1986 desarrolle una primer versión de un generador automático de reactivos, magra, pero que *funcionaba en modo 'multiusuario' para terminales TTY y se ejecutaba sobre una computadora ONYX 8002-A, con 512 KB en memoria RAM, Disco Duro de 20 MB y 'sistema operativo' UNIX ver. 3.0, a través del 'compilador' de lenguaje Pascal USCD, donde dicho software era propio de esa máquina.* En ese entonces la computadora citada era administrada por mi amigo el Lic. Javier Katsuma Arao Toyohara.

Por compatibilidad con los equipos PC con que contaría la UCIA -y la UNAM en su conjunto- a partir de mediados de 1986 y como la ONIX iba de salida en el mercado como tantos otros equipos y marcas -¿se acuerdan de las Burrough's?-, el 'código' desarrollado se transportó a equipo PC con 256 Kb en memoria RAM, Disco Duro de 10 MB, reloj de 10 Mhz y 'sistema operativo' MS-DOS ver. 3.2, donde el 'compilador' utilizado fue Turbo Pascal v 3.0, pasando en el tiempo por sus versiones 4.0 y 5.5. Esto se realizó con la valiosa ayuda de los Ingenieros en computación Jesús Flamenco, Miguel Angel Guevara y Ricardo Mostalac (en orden de aparición en mi vida). *El lector notará las capacidades que los equipos de cómputo mencionados tenían en ese entonces, comparados con sus equivalentes actuales, por citar un ejemplo, equipos PC con 16 Mb en*

memoria RAM, Disco Duro de 1 GB, reloj de 120 Mhz y ambiente de trabajo Windows 95 -por cierto, ya está por salir el 98-. A este tipo de cambios rápidos y drásticos es para lo que un profesional de sistemas de cómputo debe estar preparado.

Por otro lado y según aumentaban mis responsabilidades (pues en esas epocas tuve mi primer nombramiento como funcionario -entre otros compromisos-), la proporción de mi tiempo disponible para el proyecto y para la escritura de su 'código' cambiaba en forma inversamente proporcional, hasta que lo abandoné por ahí de mediados de 1987. Debo decir que en el caso de los primeros generadores para la ONIX y para PC, siento que "le metí" muchísimo tiempo y esfuerzo, estimo que derivado de la curva de aprendizaje al entrar de lleno al área de sistemas y, como he dicho, al incremento en las responsabilidades de su servidor, pero sobretodo, ¡debido a que trabajaba sin ninguna metodología de por medio!; tal cual lo expresa Yourdon en los siguientes términos, hablando de proyectos simples: *"A computer program or system that involves only a few hundred lines of code can usually be implemented by one person in a period of a few days to a few weeks .... And even though structured analysis, structured design, and structured programming, can be of great benefit even in such small projects, the programmer -by brute force or just common sense- can get the job done..."(sic).*

Si trabajar sin método puede resultar en algo caótico y tortuoso para un solo individuo, imagínense su repercusión en áreas organizacionales completas de sistemas -aún cuando se concluyan los proyectos-..., ¡y vaya si lo he vivido!; mi frase al respecto es: "metodizarse y reorganizarse o morir".

Con la aplicación de la variante de la metodología de Yourdon a la aplicación de cómputo que aquí expongo, siento haber invertido -estimativamente-, la cuarta parte del tiempo y esfuerzo empleados hace diez años, haciendo a un lado el "colmillo" que ya traigo -aunque "sea de leche"-, y considerando que los esfuerzos y "tiempos" actualmente empleados,

incluyen la elaboración de este documento. Aquí un comentario personal: la vez anterior que inicie los trabajos para titularme, donde me atoré no fue "en la programada" sino en la escritura de la tesis; ahora cambie la estrategia, primero escribí este texto, luego elaboré la documentación técnica de la aplicación y finalmente 'codifiqué'. Ciertamente por ahí de 1987, desarrollé algunos 'programas' que permitían al alumno, incluso, autoaplicarse reactivos: quién sabe donde quedaron.

Prosiguiendo con la historia del GAR, resulta que para mi fortuna y gracias a mi amigo el Mat. Esteban Alatriste Peredo, pude disponer de más tiempo a partir de mediados de 1997 y al presentar la idea del proyecto a mi director de tesis y luego de cabilar sobre los varios años de trabajar en el área de sistemas, pude aceptar en mi interior lo que racionalmente ya sabía: *que la versión de mi proyecto para MS-DOS había pasado a mejor vida.*

Por lo mismo me di a la tarea de meditar y medir que hacer: *iniciar un nuevo proyecto para titularme, 'codificar' desde cero otra vez la aplicación en algún "Visual" como C++ para Windows o... qué*; ahí fue cuando descubrí un producto que se denomina *'desarrollador de aplicaciones'* (*Delphi*), al cual logré acceder sin costo y que permite distribuir e instalar el *software* generado con dicho producto, también sin costo -vaya que si hay que "buscársela" en el negocio del cómputo-; con esto el GAR puede instalarse gratuitamente en mi Facultad. El producto en cuestión de alguna forma maneja de forma 'transparente' 'programación' orientada a objetos -OOP por sus siglas en inglés-, y fue elaborado por Borland, empresa propietaria del **'compilador'** de TurboPascal en el que escribí 'código' hace aproximadamente 10 años, lo que me permitió, por compatibilidad entre productos de la misma compañía, reutilizar partes y 'rutinas' de los 'programas' que 'implementé' y que había dejado en el tintero en 1987. *De hecho en el inter el lenguaje Pascal de la Cía. Borland, ahora que lo*

redescubrí bajo el producto Delphi cambió su conceptualización y denominación a Object-Pascal (i.e. propiamente para OOP).

Vale la pena comentar que, inicialmente, la idea del Lic. Jarabozo respecto a la 'funcionalidad' del generador era mantener un banco de reactivos que fuese administrado de forma centralizada por la ya citada UCSI y accedido a través del generador 'multiusuario' -vía terminales tontas-, por diversos profesores de carrera de la Fac. de Psicología. Con la transportación del 'programa' -por "cable serial"- de la UNIX a equipo PC la idea sobre su uso se matizó a un paquete de cómputo 'standalone' para profesores, incluso como los que laboran en el sistema abierto; posteriormente, en épocas del Congreso Universitario, entró en juego la idea de la generación de exámenes departamentales, etc.

A lo anterior podría agregar múltiples anécdotas y pláticas con muchos profesores de diversas Facultades de la UNAM, sobre cuántas veces no ha ocurrido la situación de plantearse: ¿esta tarea ya la puse?, ¿en qué semestre y año?, ¿creo que esta pregunta del examen nadie la respondió bien hace n años?. Esto es parte de lo que esperaría se resolviera con el GAR.

Por lo expuesto hasta aquí y como en mi Alma mater, salvo contadas excepciones, funcionan verdaderamente los asuntos de forma colegiada -¿qué fue de ti H. Congreso Universitario?-, la aplicación actualmente desarrollada fue pensada como una herramienta de cómputo que permite generar sistemáticamente y de forma automatizada, conjuntos de reactivos que permitirán, a su vez, generar los exámenes o tareas por aplicar al alumnado, así como mantener un registro de los mismos a través del tiempo con base en la actualización pertinente de la información relativa a los reactivos, siendo posible para cada profesor con

acceso a una PC, trabajar con el GAR en lo individual de forma *'standalone'* o bien, posibilitando a grupos de profesores y ayudantes de asignatura con acceso a un equipo PC, una LAN y un 'servidor', trabajar en modo 'multiusuario' con la misma aplicación de cómputo, es decir, sin requerirse una sola 'línea' más de 'código' para pasar de modo 'standalone' a 'multiusuario'.

En este contexto, debo escribir que el GAR no excluye la posibilidad técnica de que los reactivos se conjunten, uniformen, integren y compartan en la Facultad e incluso en la UNAM -digamos por Consejo Académico de Área- o incluso en otros ámbitos, creando así Bancos de reactivos únicos por asignatura, según la carrera y el plan de estudios al que correspondan. Al efecto, estimo, se tendría que **'escalar'** la aplicación para transformar su actual 'arquitectura' -en caso óptimo-, en Cliente/Servidor; esto se lograría sin modificar sustancialmente el 'código fuente' de la aplicación y 'programando' lo necesario para el ***'engine'*** del 'servidor' de datos con que se contara (*e.g. 'stored procedures'*), aprovechando las bondades del 'desarrollador de aplicaciones' empleado para la 'implementación' de la aplicación en cuestión -según se indica en su documentación-, pero requiriendo al efecto y entre otros rubros de una LAN (o en su caso WAN), un 'servidor' preferentemente de UNIX y un 'servidor de bases de datos', los administradores de cada caso, la respectiva capacitación y... por lo menos, un par de impresoras!, pues resulta lamentable comentarlo, pero hoy el servicio de impresión para los profesores de mi Facultad es muy limitado cuando, por lo menos, en el actual proceso enseñanza-aprendizaje que ahí se vive, la impresión de documentos es algo básico e indispensable -hasta hoy así es, más adelante ¿quién lo sabe?-.

Reitero sobre mi preocupación de que los reactivos se conjunten, integren, uniformen y compartan en nuestra Facultad y por que no, con otras de la UNAM u otras instituciones inclusive: la posibilidad técnica existe mientras que, por parte del elemento humano, quién sabe... al respecto me permito plantear una propuesta de organización a fin de instrumentar el generador de forma colegiada en la UNAM en el Anexo 4; asímismo, en el punto 7.1

(Muestra de documentación técnica), se encontrará una propuesta del 'esquema de la base de datos relacional' para una arquitectura Cliente/Servidor y aquél que hoy contempla la aplicación de cómputo.

Otro escenario de empleo del GAR o alguna herramienta equivalente, lo constituye su uso en la educación básica de mi país, aunque sea a través de papeles, al posibilitar la uniformidad en la generación de exámenes para el alumnado; empero, ahí el problema sería de voluntad política ¿o ustedes creen que las autoridades y/o el sindicato de profesores lo aceptarían?.... por lo menos yo no lo sé. Lo que si puedo afirmar es que una forma de elevar el nivel educativo consiste en elevar de manera homogénea el nivel de los instrumentos de evaluación, particularmente de los exámenes, lo que implicaría -en el mejor de los casos-, una verdadera cobertura de los planes de estudio y una mejor calidad en la enseñanza, esto si, quién sabe con que otras consecunecias sociales. ¿Verdad que actualmente no es lo mismo un 7 en una escuela privada que un 9 en una pública....? y, análogamente en casos ciertos ¿verdad que actualmente no es lo mismo un 7 en una escuela pública que un 9 en una privada....?, ¡Cuidado!, con este comentario no pretendo demeritar a la enseñanza pública sino todo lo contrario: contribuir con el presente trabajo aunque sea con una idea, un granito de arena en su dignificación y en el mejoramiento de su nivel.

La aplicación de cómputo en sí, más mi corta experiencia en el área y las ideas que traía en la cabeza sobre formas de trabajar proyectos de aplicaciones y sistemas de cómputo, es lo que intento "aterrizar" en la presente tesis.

6.2 MARCO CONCEPTUAL

Como inicialmente en este asunto fui a caer "en tierra de Psicólogos" y con la valiosa colaboración de algunos de ellos, entre quiénes destacan Rocio Bibriesca, Tonantzin Gómez, Jorge Ameth y Martha Fuentes, me di a la tarea de investigar sobre como se deben hacer las preguntas, digamos, para un examen. A partir de esto encontré en esa Facultad libros interesantes acerca del tema: de Bloom, Gronlund y Pérez Rivero. Particularmente, en el libro de Gronlund se pueden encontrar varios ejemplos de los asuntos tratados con detalle en

la descriptiva del generador (ver punto 6.4) y, en el libro de Bloom, se pueden encontrar ejemplos ilustrativos de reactivos para cada uno de los niveles de la taxonomía de dominio cogniscitivo por él construida.

6.2.1 TAXONOMÍA DE DOMINIO COGNOSCITIVO DE BLOOM

Esta taxonomía fue elaborada por Benjamín S. Bloom en 1913 y sirve como una lista de comprobación de los resultados del aprendizaje, que pretende impedir la omisión de aspectos importantes y se divide en los seis niveles siguientes:

1.- Conocimiento

Se refiere básicamente al uso de la memoria. Implica la capacidad de almacenar información y recuperarla mediante el recuerdo. El estudiante repite la información sin cambios.

2.- Comprensión

En este nivel se requiere que el estudiante haya entendido el significado del material, sin pedirle que lo aplique, lo analice, o lo relacione con otros materiales. Hay tres subniveles de tipos de comprensión, que son:

* Traducción: el estudiante solamente puede hacer una sustitución, término a término, de la información recibida.

* Interpretación: el alumno es capaz de reorganizar el material recibido. independientemente del orden en que se presente.

* Extrapolación: el educando puede derivar del material recibido algo más de lo que se le informó explícitamente; puede inferir y aún concluir.

3.- Aplicación

El estudiante generaliza y usa lo aprendido en otros contextos. Implica el manejo de abstracciones en situaciones concretas.

4.- Análisis

La persona descompone la información en sus elementos constitutivos y determina la naturaleza de sus relaciones.

5.- Síntesis

El alumno integra los elementos que recibió por separado, para producir una entidad nueva y diferente a los elementos iniciales.

6.- Evaluación

El estudiante elabora u organiza criterios para juzgar materiales o métodos ya definidos. Implica una toma de decisiones.

6.2.2 PLANES DE ESTUDIO Y FORMAS DE EVALUACIÓN

Aquí debo mencionar que según recomienda una corriente de la Pedagogía, los planes de estudio de las asignaturas, es decir, los contenidos de las mismas, deberían estar organizados como se muestra esquemáticamente a continuación, donde por lo menos debe existir el planteamiento de un objetivo de aprendizaje por cada uno de los temas o por cada uno de los temas y subtemas ahí establecidos como se muestra en la figura 25.

Dos ejemplos de objetivos de aprendizaje son:

* "Al término del tema el alumno será capaz de obtener el gradiente de cualquier función real de variable real en que sea posible obtenerse y podrá indicar el porqué no es posible en casos de excepción".

* "Al cubrir el subtema sobre el uso de la z del tema de ortografía, el educando podrá escribir sin errores ortográficos cualquier palabra que contenga la consonante mencionada".

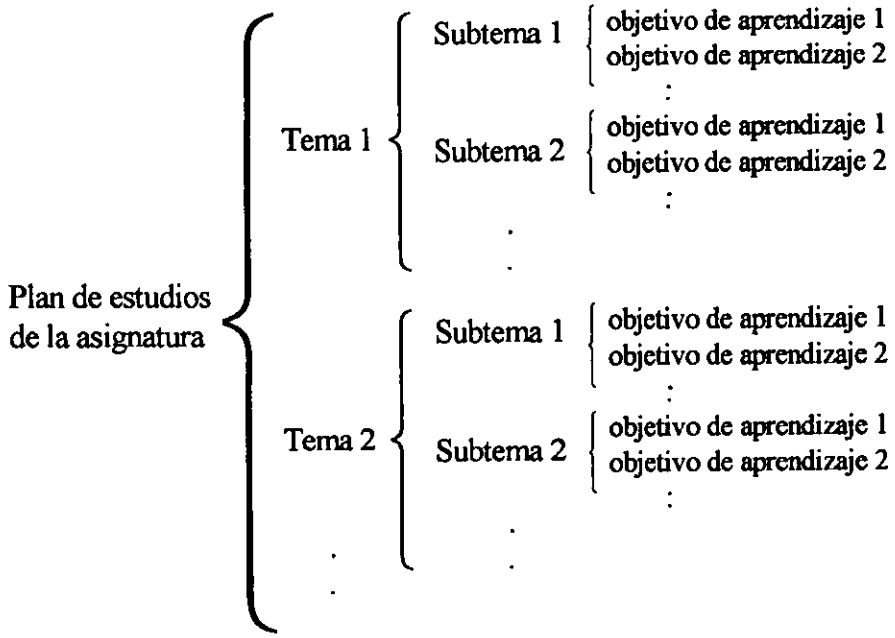


FIGURA 25: ORGANIZACIÓN DE CONTENIDOS DE ASIGNATURAS

En este contexto y con base en la anterior estructuración de las asignaturas, debo mencionar dos conceptos pedagógicos referentes a dos formas posibles de realizar la *evaluación* del aprovechamiento de los alumnos:

* Según considera la *evaluación formativa*, el (los) tema(s) y subtema(s) específicos para un examen o tarea, deben contener al menos un reactivo de cada objetivo de aprendizaje, de manera tal que se evalúe el logro de todos y cada uno de los objetivos del curso.

* Alternativamente por parte de la *evaluación sumaria*, se considera que con evaluar por medio de una selección al azar los objetivos de aprendizaje, es suficiente, pues presupone que el alumno se prepara para cubrir todos los objetivos planteados para el curso. En este sentido, dado(s) el (los) tema(s) y subtemas, específicos para un

examen o tarea, basta que los reactivos del mismo, correspondan a algunos, no necesariamente todos, los objetivos considerados en el plan de estudios de la materia para dicho(s) tema(s) y subtemas.

El GAR fue construido con base en la evaluación sumaria, lo que en su momento me llevó a investigar como poder seleccionar en alguna forma aleatoria reactivos de un banco, digamos, de un tema en específico. Los aspectos técnicos del caso en materia de la 'implementación' se detallan en 7.1 del presente documento.

Por otro lado, las razones del porqué el Lic. Zarabozo prefirió que la aplicación se realizara sobre la base conceptual de la evaluación sumaria en lugar de la formativa, están más allá del alcance de este documento; empero, debo comentar que estos dos conceptos o formas de evaluación del aprovechamiento de los alumnos, dan lugar a fuertes debates y polémicas entre especialistas y expertos en la materia -en la cuál me confieso como un humilde aficionado-.

Sin embargo, debo decirr que en mi experiencia como alumno y docente, típicamente, salvo muy contadas excepciones, todo mundo aplica las evaluaciones a los estudiantes a través de exámenes que contemplan solamente a un subconjunto propio de los temas y subtemas planteados para el curso -de los objetivos de aprendizaje ni hablar-, aunque intentando abarcarlos en su totalidad (*v.g.* exámenes finales); en resumidas cuentas: ¿quién no ha resuelto al menos un examen final con las características mencionadas en el transcurso de su carrera o formación profesional?.

Adicional a lo anterior y después de un largo rato de pensar en esto -como aficionado-, estimo que hay dos aspectos a mencionar sobre la necesaria retroalimentacion para el proceso enseñanza-aprendizaje, que lamentablemente no siempre sucede y si se da, no necesariamente ocurre en forma sistemática y en cambio muy a menudo sucede a nivel personal, pero de forma inconsciente.

Por parte de las instituciónes o entidades académicas donde llega a ocurrir, por lo general sucede en forma totalmente burocrática y sin tener nada que ver con aspectos estrictamente educativos (*v.g.* se evalua la cobertura de los cursos ¡a través de las asistencias del profesor!).

Los aspectos mencionados que desde mi punto de vista deben integrarse de alguna manera en forma sistemática al proceso enseñanza-aprendizaje son:

* La experiencia de los docentes; resultado del trabajo de exposición y/o guía y evaluación a los alumnos. Con el GAR, esa experiencia debe revertirse en los bancos de reactivos para su mejoramiento al contemplar en detalle la cobertura de objetivos de cada curso -sin pretender atentar para nada contra la la libertad de cátedra- y la evaluación de resultados derivados de la aplicación al alumnado, a través del tiempo, de los reactivos del banco, en búsqueda de su perfeccionamiento. A mi entender de una manera u otra, los mecanismos de evaluación de los estudiantes son compendiados en reactivos en los cuales, por lo menos hasta ahora, se sustenta la evaluación de los educandos.

* El trabajo y desempeño de los alumnos; al ser evaluados mediante reactivos homogéneos y mantener una historia sobre los resultados de su aplicación a través del tiempo, es posible ubicar a los reactivos destinados para fines de evaluación en ese contexto.

Profesionales versados en el asunto dirán tal vez que estoy descubriendo el "agua tibia" o que mis comentarios están totalmente fuera de lugar, pero en el contexto de esta tesis, la aportación para la mejora del proceso en cuestión es el propio GAR, que facilita condiciones para sistematizar de alguna manera, los dos aspectos antes citados y posibilita el establecimiento de un proceso de retroalimentación como el que se plasma en la página siguiente (figura 26), sin excluir otros que debiesen darse sistemáticamente, como la evaluación de los profesores -por citar un ejemplo-.

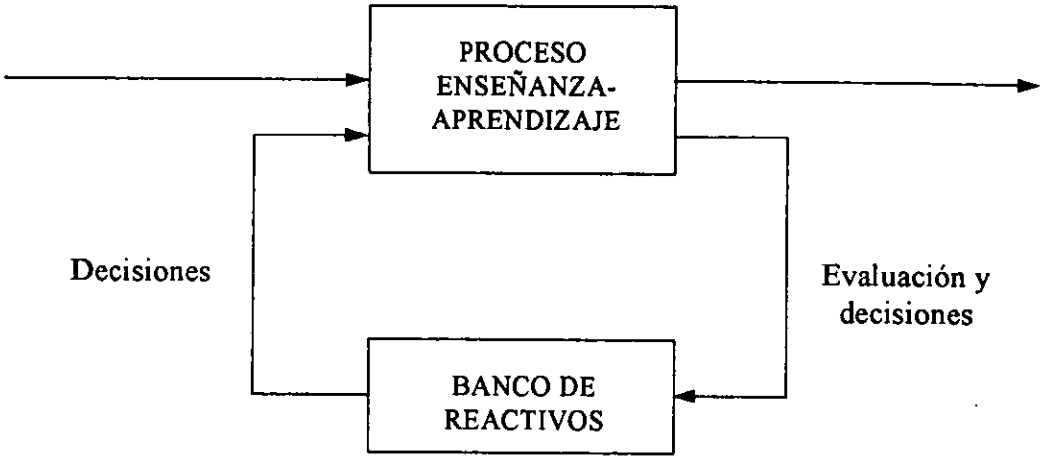


FIGURA 26: PROCESO DE RETROALIMENTACIÓN

6.3 DISEÑO CONCEPTUAL

En el contexto del marco conceptual previo y del propio GAR, enseguida planteó la conceptualizacion de los elementos que permite manipular directamente la aplicación de cómputo: reactivos y bancos de reactivos; estas definiciones conceptuales fueron generadas por su servidor para lograr su instrumentación técnica en la aplicación de cómputo es decir, con base en ellas se crearon los conceptos técnicos y parte de las 'tablas' de la 'base de datos' del GAR. En otras palabras: son la interpretación técnica de los conceptos en cuestión originalmente investigados en la bibliografía citada al final de este documento.

6.3.1 REACTIVOS

Entendemos en adelante por reactivo una entidad que satisfaga el esquema presentado en el siguiente diagrama:

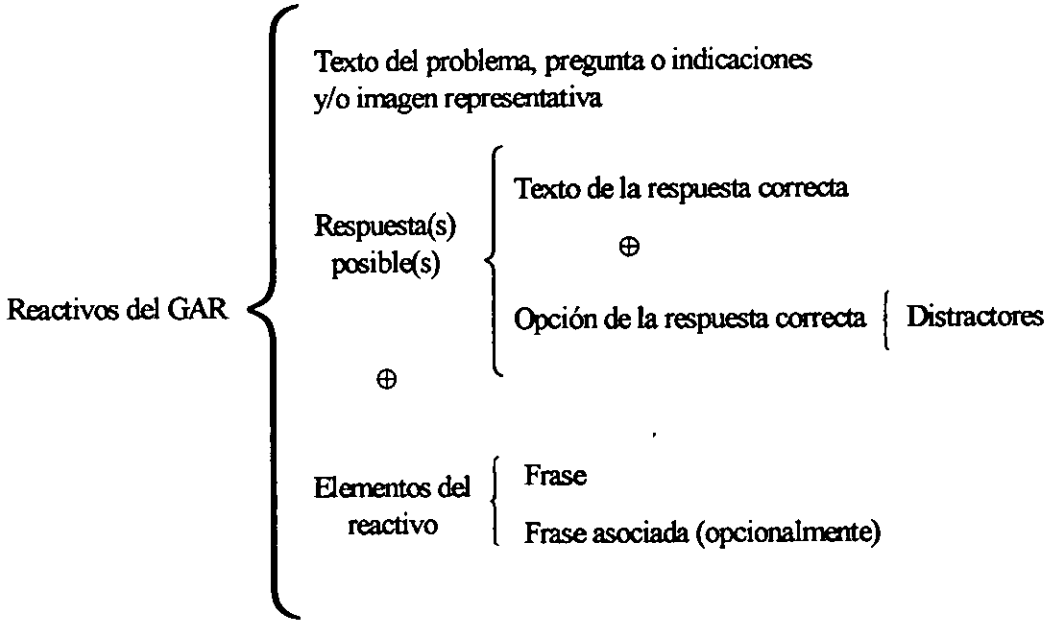


FIGURA 27: ESQUEMA DE REACTIVOS DEL GAR

donde es indispensable para constituir un reactivo, el texto de la pregunta o problema o indicaciones y el texto o la opción de la respuesta correcta.

Consecuentemente con el esquema anterior, el GAR permite la captura o copia, desde otra aplicación para Windows, de:

* los textos de la pregunta o problema o indicaciones y, en su caso, de su imagen representativa,

* el texto o la opción de la respuesta correcta,

* los textos de los distractores del caso o,

* los textos de los **elementos del reactivo**, cuando así se requiera.

El concepto **elemento del reactivo** está constituido por a lo más dos partes: un texto que contiene una frase y, opcionalmente, otro texto que contiene una frase asociada a la primera, como se representa en el esquema anterior.

6.3.2 TIPOLOGÍA DE REACTIVOS DEL GAR

Los tipos de reactivos del GAR se definieron de acuerdo a los atributos que los componen y en términos de su funcionalidad propia para la aplicación de cómputo.

En este contexto y retomando clasificaciones preexistentes y ejemplos de reactivos de los autores mencionados en el marco conceptual, me permití armar una tipología técnica que me permitió a su vez, la instrumentación técnica de los reactivos mencionados en el GAR, considerando así, en forma implícita y desde un inicio, los requerimientos básicos para el almacenamiento de los reactivos en 'tablas' de una 'base de datos' y la 'funcionalidad' básica de los 'programas' de computo correspondientes.

Aquí un paréntesis, aunque esté totalmente fuera de lugar -perdón, pero aquí se me ocurrió y creo que aquí queda bien-: todos estos aspectos que estoy abordando constituyen parte de la 'actividad' del análisis de la variante de la metodología y pretenden mostrar como traté de comprender la información que tenía disponible sobre el tema de los reactivos. En mi experiencia, sobretodo en un rol de administrador o líder de proyecto, adicional a su pericia en el área de sistemas -en un sentido amplio- y al dominio del aspecto técnico y del personal a su cargo, así como para lograr hacer y "aterrizar" una aplicación o sistema de cómputo sobre "lo que sea", de entrada no hay mejor recomendación que conocer lo más amplia y anticipadamente posible, qué es y cómo se comporta y funciona "lo que sea". Justamente lo mismo recomiendan Yourdon y Coad en su libro de OOA y es a lo que se denomina, en forma "tropicalizada" -dirían los gringos en español-: el dominio sobre el *"dominio del problema"*. ¡Cierro paréntesis!.

Prosiguiendo con la tipología de reactivos del caso, se construyó acorde y consecuentemente con las posibilidades de manejo de textos e imágenes en una computadora y, como he mencionado, contemplando clasificaciones preexistentes de reactivos, concluyendo la del caso según el tipo de estructuración de las respuestas a las preguntas de los reactivos, de tal suerte que llegué a lo siguiente (figura 28):

FIGURA 28: TIPOLOGÍA DE REACTIVOS DEL GAR

Llanamente: el GAR maneja reactivos de respuesta Abierta, Breve, de Complementación, de Falso/Verdadero, de Opción múltiple, Jerarquización y Correspondencia.

Técnicamente: el GAR maneja reactivos almacenados en las 'tablas': Treactiv, TCodRes, TRespAb, TImagen, TOpcion, TOpcAsoc. Para efectos de comparación, veáse el 'esquema' de su 'base de datos' en el punto 7.1 (Muestra de documentación técnica).

Enseguida describo y ejemplifico qué es cada tipo de reactivo, mencionando que la tipología planteada para los mismos depende de la estructuración de su(s) posible(s) respuesta(s).

6.3.2.1 Reactivos que maneja el GAR cuya respuesta es texto elaborado por el alumno

1.- Respuesta abierta

Consiste de un enunciado que presenta una situación problema o indicaciones sobre lo que el alumno debe considerar o redactar y puede ser en forma de pregunta o en modo imperativo y/o mezclas de las formas anteriores. En este sentido la respuesta es un texto, que denominaría amplio, correspondiente a lo que el alumno considere es lo más adecuado y congruente con el texto del planteamiento. Se aplicaría para el caso de la Facultad sobre todo para textos o imágenes con demostraciones plausibles. Ejemplos al efecto son:

¿Cuántas y cuáles estadísticas de tendencia central existen?. Describa en detalle cada una de ellas y en qué casos se recomienda su aplicación.

Demuestre que la suma de los ángulos de cualquier triángulo es π.

Opine sobre las votaciones de 1997. No se extienda más de una cuartilla.

"Algún texto de Trabajo y Energía....."
Escribe en las siguientes líneas, todo lo que hayas aprendido acerca del concepto de energía, partir de la lectura del texto de Trabajo y Energía que acabas de leer, en forma sintética.

Las respuestas que deberían plasmarse para los planteamientos respectivos deberían entonces, corresponder respectivamente a los enunciados anteriores -más o menos-, con:

La aseveración de que las estadísticas de tendencia central son tres: media, mediana y moda y la explicación de en qué consiste cada una de ellas, como se calculan (v.g. fórmula) y en cuáles casos se recomienda su aplicación, incluso con ejemplos ilustrativos.

Una o varias formas de demostración de que la suma de los angulos de cualquier triángulo es π.

Un breve ensayo de las votaciones de 1997 -elaborado por el profesor-.

El resumen del concepto de energía, acorde al texto de "Trabajo y Energía".

2.- Respuesta Breve

Se forma de un enunciado que establece parte de una frase o una pregunta o la transcripción por parte del alumno de algún tipo de conocimiento relacionado con el planteamiento. La respuesta la definiría como un texto que denominaría corto o de una frase, de a lo más una línea, correspondiente a la respuesta correcta, como se presenta enseguida:

La energía no se crea ni se destruye, solamente.. ¿qué sigue?

¿Cuáles son las medidas de tendencia central?

¿Cuáles son las siglas de los partidos políticos registrados ante el IFE en 1997 -en orden alfabético-?

Escribe el nombre del principio al que se hace referencia en el siguiente parrafo: "Si cierta cantidad de determinado tipo de energía desaparece, siempre es posible verificar la aparición de otro tipo de energía, en cantidades equivalentes a la energía disipada".

Las respuestas asociadas a los textos anteriores que corresponden a las preguntas o indicaciones respectivas son:

se transforma.

media, mediana y moda.

PAN, PC, PFCRN, PPS, PRD, PRI, PT, PVEM.

Lavoisier.

3.- Respuesta de complementación

Los elementos del reactivo son un frase en forma de pregunta o de enunciado, conciso pero incompleto; su respuesta es un texto que denominaría muy corto, preferentemente de una palabra importante o "clave", que complementa al texto del planteamiento del reactivo:

La media, mediana y moda coinciden en la densidad de la distribución _____.

La velocidad de la luz es de _____.

El Principio de _____ establece que "Si cierta cantidad de determinado tipo de energía desaparece, siempre es posible verificar la aparición de otro tipo de energía, en cantidades equivalentes a la energía disipada".

Las respuestas de los anteriores planteamientos son respectivamente:

Normal.

300,000 Km/Seg.

Lavoisier.

6.3.2.2 Reactivos que maneja el GAR cuya respuesta es seleccionada por el alumno de un conjunto de opciones

1.- Falso/Verdadero

El planteamiento del reactivo es un enunciado muy conciso que es, exclusivamente, verdadero o falso y que es complementado por las dos opciones de respuesta correspondientes:

La velocidad de la luz es de 360,000 Km/Seg.
( ) Falso ( ) Verdadero

El primer Psicólogo que se dedicó al estudio de la memoria fue Galton.
( ) Verdadero ( ) Falso

2. -Opción múltiple

Este tipo de reactivos son ampliamente conocidos e incluso en la jerga estudiantil se les denomina de "confusión múltiple". Los elementos de este tipo de reactivo son :

* Un enunciado que presenta una situación problema y puede ser en forma de pregunta o estar incompleto.
* Una opción correcta de respuesta.
* Varias opciones incorrectas que se denominan distractores.

Ejemplos de reactivos de opción múltiple son:

Es un trastorno del habla que incluye la repetición mecánica de la ultima palabra escuchada.
(a) Dislexia (b) Dislalia (c) Ecolalia (d) Ecolexia

¿La velocidad de la luz es de?
1) 350,000 Km/Hra. 2) 360,000 Km/Hra. 3) 370,000 Km/Seg. 4) 300,000 Km/Seg. 5) 350,000 Km/Seg.

3.- Jerarquización

Los elementos de este tipo de reactivo son:

* Una enunciado o instrucción general que indique el establecimiento de un ordenamiento o jerarquización de los elementos presentados en una columna.
* Una columna que contenga enunciados con información que pueda ser ordenada, preferentemente de la misma naturaleza.

En este contexto, entendemos que un enunciado y el número de orden que le corresponde, forman una pareja llamada elemento del reactivo. Dos ejemplos en la materia son:

Ordene en términos de su rapidez las siguientes velocidades y anote en los paréntesis, respectivamente, números del 1 al 4, donde el 1 corresponde a la menos rápida y el 4 a la más rápida:

( )	Velocidad del sonido en el agua.
( )	Velocidad del sonido en el aire.
( )	Velocidad de la luz.
( )	Velocidad del sonido en sólidos.

Jerarquice de menor a mayor, usando las letras del alfabeto donde a < b, b < c, etc., los siguientes ordenamientos legales:

( )	Ley Federal del Trabajo.
( )	Constitución Política de los Estados Unidos Mexicanos.
( )	Ley Orgánica de la Administración Pública Federal.
( )	Ley Federal de Responsabilidades de los Servidores Públicos.

4.- Correspondencia

Los elementos de este tipo de reactivo son :

* Una enunciado o instrucción general que indique el establecimiento de una relación entre dos columnas.
* Una columna que contenga frases de aseveración o premisas.
* Otra columna que contenga frases asociadas.

En este terreno, comprendemos que una frase de aseveración y una frase asociada forman una pareja llamada elemento del reactivo. Los elementos de las columnas, preferentemente deben ser del mismo tema.

Dos ejemplos del caso son:

La columna B es una lista de eventos históricos. Sobre la línea que está a la izquierda de cada enunciado coloque la letra que está a la izquierda de la columna A que contiene los personajes principales que intervinieron en los eventos descritos en la columna B.

	Columna A (Frases de aseveración)	**Columna B** (Frases asociadas)
A	Lázaro Cárdenas	___ Participó en el Movimiento Revolucionario de 1910
B	Francisco Villa	___ A quién se atribuye la frase: Sufragio Efectivo, NoReelección
C	Benito Juárez	___ Fue el primer presidente del México Independiente
D	Francisco I. Madero	___ Quién instituyó la Expropiación petrolera
E	Guadalupe Victoria	_ Proclamó las Leyes de Reforma

Relacione las unidades de longitud de la izquierda con los paréntesis de la derecha.			
1)	1 kilómetro	( )	0.10 m.
2)	1 decámetro	( )	0.01 m.
3)	1 centímetro	( )	1,000 m.
4)	1 decímetro	( )	10 m.

Todos y cada uno de los tipos de reactivos aqui establecidos, considerados como requerimientos, dan origen a parte de la 'funcionalidad' del GAR para fines de su registro en la 'base de datos'.

6.3.3 BANCO DE REACTIVOS

A un conjunto de reactivos que cumplen con poder ser conceptualizados dentro de la tipología antes propuesta y que sea posible sean captados a través del GAR, es a lo que identificamos como banco de reactivos; en la aplicación de cómputo correspondiente, la captura o copia desde otra aplicación para Windows de los textos o imagen de los reactivos, genera un conjunto de 'archivos' o 'tablas' en el disco de una computadora o 'servidor'; es ahí donde se preserva la información que es la base para la generación automática de los reactivos.

La información posible de ser almacenada en el banco es pues, básicamente el contenido, los textos y/o imágenes representativas de los reactivos así como su respuesta, siendo estos prácticamente de cualquier tema.

Los reactivos pueden y deben ser almacenados, según se organicen por tema, subtema y objetivo de aprendizaje a los que correspondan, según se clasifiquen debiendo contemplar al efecto, invariablemente, los planes y programas de estudio a los que correspondan, según la asignatura de estudio respectiva.

Como puede anticiparse, además de características comúnes (*v.g.* tema), los reactivos tienen y "aterrizan" en la 'base de datos', características que los distinguen propiamente tales como:

número único de identificación, clave del nivel taxonómico -según Bloom-, clave del tipo de reactivo conforme a la tipología descrita, fecha de creación -el año se registra en 4 digitos por el cambio de milenio-, número de veces que se ha aplicado el reactivo, número de veces que se ha respondido correctamente por los estudiantes, código(s) de respuesta(s) correcta(s), así como los siguientes indicadores: grado de dificultad e índice de discriminación.

6.3.4 INDICADORES

Con esta parte pretendo que se comprenda el significado y la obtención de los indicadores de un reactivo.

6.3.4.1 Grado de dificultad

Al elaborar reactivos para evaluar el aprovechamiento del estudiante es importante determinar y evaluar que tan fácil o dificil es responder a cada reactivo.

El grado de dificultad se refiere al porcentaje de veces en que un reactivo es respondido correctamente.

Es recomendable comenzar con reactivos fáciles y aumentar gradualmente la dificultad a lo largo del examen; esto es irrelevante para el caso de las tareas.

Para determinar el grado de dificultad es necesario seguir un procedimiento empírico, es decir, aplicar un examen típico y calcular el porcentaje de estudiantes que respondieron correctamente a cada reactivo. A continuación se presenta un ejemplo de pasos necesarios para determinar dicho grado para un reactivo de opción múltiple. Al efecto supondrémos el uso de 15 exámenes.

1) Ordenar los 15 exámenes desde la puntuación más alta hasta la más baja.

2) Seleccionar la tercera parte de los exámenes correspondientes a las calificaciones más altas; estas formarán el "grupo superior" (5 exámenes).

3) Seleccionar el mismo de exámenes con las calificaciones más bajas; estos formarán el "grupo inferior" (5 exámenes).

4) Separar el "grupo intermedio" (5 exámenes).

5) Para cada reactivo, contar el número de estudiantes del "grupo superior" que eligió cada opción. Hacer lo mismo para el "grupo inferior".

6) Registrar los resultados correspondientes como se muestra a continuación:

	OPCIONES				
GRUPO	A	B	C	D	E
SUPERIOR	0	4	1	0	0
INFERIOR	0	1	1	3	0

donde la opción B representa la respuesta correcta

7) Estimar la dificultad del el porcentaje de estudiantes que lo respondieron correctamente. Un procedimiento sencillo es:

* Obtener el total de los casos del grupo superior e inferior (5 + 5 = 10)
* Sumar los casos de elección de respuesta correcta (4 + 1 = 5)
* Dividir la segunda suma entre la primera (5/10)

Como ya se mencionó, la dificultad se refiere al porcentaje de veces que se responde correctamente al reactivo. En consecuencia, un porcentaje menor indica mayor dificultad del reactivo y viceversa.

6.3.4.2 Índice de discriminación

Al elaborar reactivos para evaluar el aprovechamiento del estudiante es importante determinar si estos realmente discriminan entre los estudiantes que obtienen altas puntuaciones y los estudiantes que obtienen bajas. De esta forma, el índice de discriminación indicará que: el estudiante "aplicado" deberá tener una puntuación muy alta, el estudiante "medio" obtendrá un 50% de respuestas correctas y, el estudiante "inferior" una puntuación ligeramente superior a la que se puede obtener por azar. Es recomendable que los reactivos tengan un índice de discriminación relativamente alto, por ejemplo, entre 0.50 y 1.00

Para determinar el índice en cuestión, es necesario seguir un procedimiento empírico, es decir, aplicar un examen que contenga reactivos provenientes del banco de reactivos.

Para ilustrar un ejemplo de la aplicación de un procedimiento al efecto, tomemos como base las puntuaciones obtenidas para el reactivo de opción múltiple de la página anterior. Los pasos a seguir son:

1.- Obtener el número de estudiantes que contestaron correctamente el reactivo a analizar y que pertenezcan al grupo "superior" (4).

2.- Obtener el número de estudiantes que contestaron correctamente el reactivo a analizar del grupo "inferior" (1).

3.- Restar el número obtenido en el paso 2 del número obtenido en el paso 1

$$4 - 1 = 3$$

4.- Dividir el número obtenido entre el número de estudiantes que forman cada grupo.

$$3 / 5 = 0.6$$

El máximo poder discriminativo positivo indica un índice de 1. Este se obtiene solamente cuando todos los estudiantes del "grupo superior" eligen la respuesta correcta sin que lo haga ninguno del "grupo inferior", ya que obtendríamos lo siguiente (5 -0) / 5 = 1.

6.4 DESCRIPTIVA DEL GAR

Este rubro pretende explicar genéricamente la 'funcionalidad' de la GUI correspondiente al concepto que comprende a todas las ideas, términos y requerimientos que hasta aquí he mencionado, a lo cual he denominado *G*enerador *A*utomático de *R*eactivos. Consiste de una explicación de la aplicación GAR, en razón de los rubros conceptuales básicos alrededor de los cuales se desarrolló: generación automática de reactivos y manejador de bancos de reactivos, así como aquéllos necesarios para completarlo: seguridad, estadísticas y utilerías.

El detalle de la 'funcionalidad' mencionada está redactado e incluido en la GUI propia de la aplicación a través de su 'ayuda', sin embargo debo mencionar que la manera en que se proporcionan datos e instrucciones al GAR es a través de su digitación y del uso de 'objetos' o 'componentes' del 'desarrollador de aplicaciones' utilizado, que permiten por ejemplo la selección de 'objetos' y/o datos mediante herramientas y 'eventos' bien conocidos del usuario del ambiente Windows, como el dar "click" a un botón del *'mouse'* en un dato o al **'arrastrarlo y soltarlo'** con el *'mouse'* (**'*drag & drop*'**), según lo vaya facilitando la aplicación.

Como parte del anecdotario, aquí cabe citar que en textos de la versión del generador de diez años, el correspondiente a esta parte decía: "La manera en que se proporcionan los datos para su almacenamiento en el banco, es capturándolos como en una máquina de

escribir y registrándolos con la tecla de <Enter>" (sic) Aquí perfectamente se puede apreciar un cambio técnico sustantivo subyacente en el cambio en la 'implementación' de la 'interface' con el usuario -inicialmente en modo de caracteres bajo MS-DOS-, pasando hacia el uso de la GUI de Windows. *Esto es de lo que se puede apreciar "desde fuera" de la aplicación, pero sobretodo en términos de 'implementación' y diseño, el cambio mencionado -por lo menos para mi-, y el paso de 'programación estructurada' a 'programación' según los 'eventos' y mensajes que "le llegan" al 'objeto' -¡que no propiamente OOP!-, es tanto o más drástico que el cambio de la 'interface' del usuario. De hecho conozco gente del área de sistemas que en varios años no ha podido asimilar del todo cambios como estos y a otra, que por lo menos hasta ahora, ¡el cambio les resulta ¡imposible!.*

Para continuar con este texto genérico y sólo para clarificar ideas así como para "abrir boca", presento el 'menú' principal del GAR que persigue controlar su 'funcionalidad', así como los **'permisos de acceso'** de los usuarios a las opciones de dicho 'menú' y a los bancos de reactivos, en conjunción y a través del 'módulo' de seguridad:

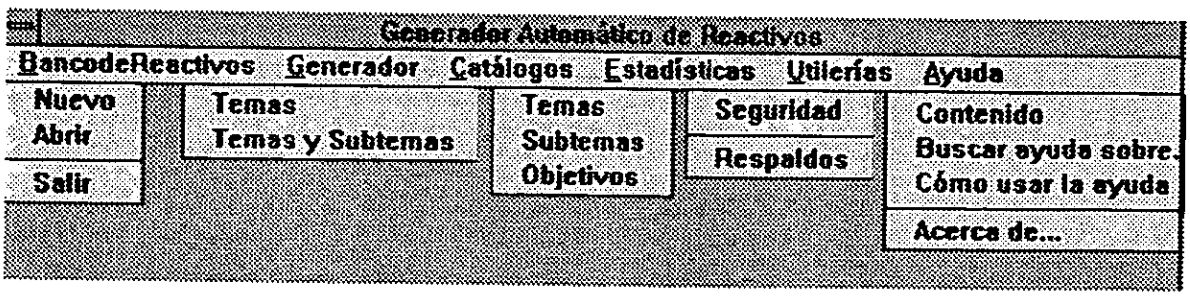


FIGURA 29: MENÚ PRINCIPAL DEL GAR

Como podrá observarse la 'implementación' la hice con 'componentes' ***TForm*** de Delphi que corresponden a 'ventanas' de Windows versión 3.x y por lo tanto, las 'ventanas' del GAR incluyen por *default* la 'funcionalidad' estándar correspondiente, es decir:

Cuadro de menú de control (esquina superior izquierda).

Botones de maximizado y minimizado (esquina superior derecha).

Barra de títulos (parte superior).

Bordes ajustables (en tamaño).

Barra de menú (que contiene los **'comandos'** de la 'ventana').

Para Windows 95 se tiene la 'funcionalidad' correspondiente con los cambios del caso -*v.g.* botones para maximizar, minimizar o cerrrar-. Por simplificación del documento, en adelante explicaré el funcionamiento general del GAR y sus 'ventanas' solamente para ambiente de trabajo Windows versión 3.x. En este contexto es pertinente mencionar que el mismo 'programa' elaborado por su servidor con la versión 1 de Delphi, se puede utilizar indistintamente en la versiones 3.x y 95 del ambiente de trabajo aquí mencionado.

6.4.1 SEGURIDAD

A continuación se explica la 'funcionalidad' de este 'módulo', cuyos objetivos son:

A) Permitir a los usuarios en general el acceso al 'menú' principal de la aplicación mediante la captura de su **'clave de usuario'** y la digitación de su ***'password'*** o contraseña o Número de Identificación Personal (NIP), validando su registro previo en la 'base de datos' y leyendo entre otra información, sus datos personales y sus 'permisos de acceso' a las 'opciones del menú' mencionado y a los bancos de reactivos.

B) Permitir a los usuarios con atributos suficientes de administración, el registro de otros usuarios a través de la actualización de su 'clave', *'password'* y datos personales, así como de los 'permisos de acceso' correspondientes citados en el inciso anterior.

Con relación al objetivo A) -previa instalación de la aplicación de cómputo-, al ejecutar el GAR aparece la 'ventana' mostrada en la siguiente página (figura 30):

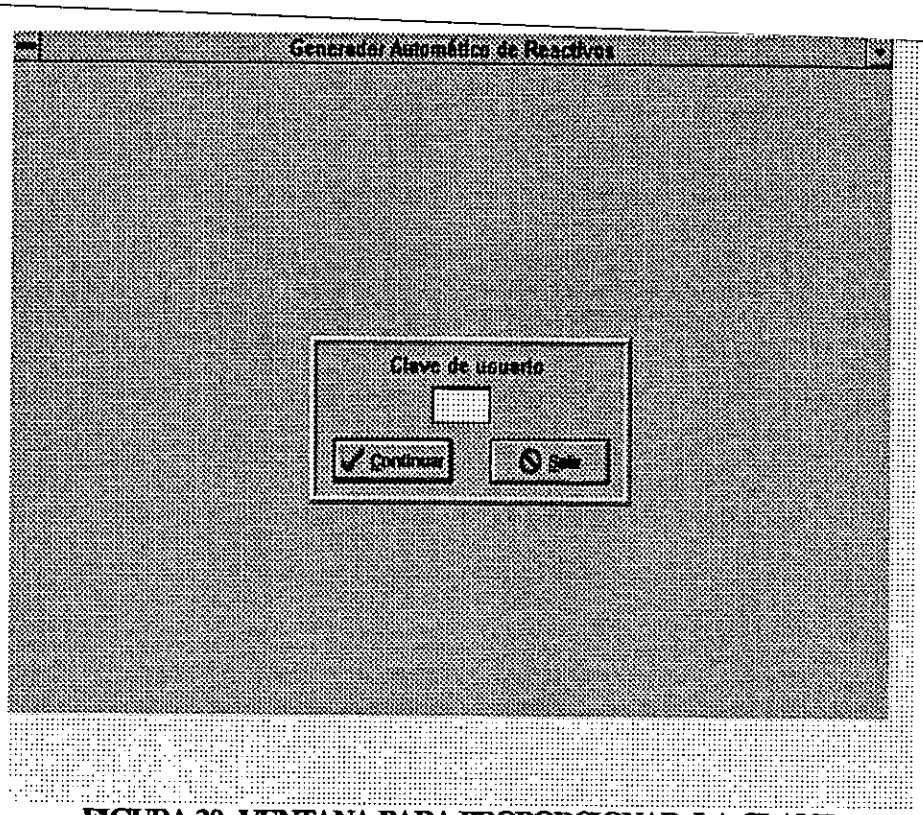


FIGURA 30: VENTANA PARA PROPORCIONAR LA CLAVE DEL USUARIO

en la cual el usuario digitará su 'clave' -que debe ser un número entero-; dando "click" con el '*mouse*' al botón "Continuar" se podrá proseguir con la captura del '*password*' siempre que exista registro del dato proporcionada en la 'base de datos'. De no existir la 'clave de usuario', previo mensaje al efecto, aparecerá nuevamente la misma 'ventana', sin poder acceder a otra parte de la aplicación y para salir de ella, basta dar "click" con el '*mouse*' al botón respectivo o dar doble "click" con el '*mouse*' en el cuadro de menú de control de la 'ventana'. En lo subsecuente la frase dar "click" significará lo mismo que dar "click" con el '*mouse*'.

Debo aclarar que para lograr la 'funcionalidad' anterior, el administrador del GAR debió haber registrado previamente al usuario en cuestión a través de la propia aplicación de cómputo, mediante su 'módulo de seguridad'.

Para digitar el '*password*' del usuario se presenta la 'ventana' siguiente (figura 31):

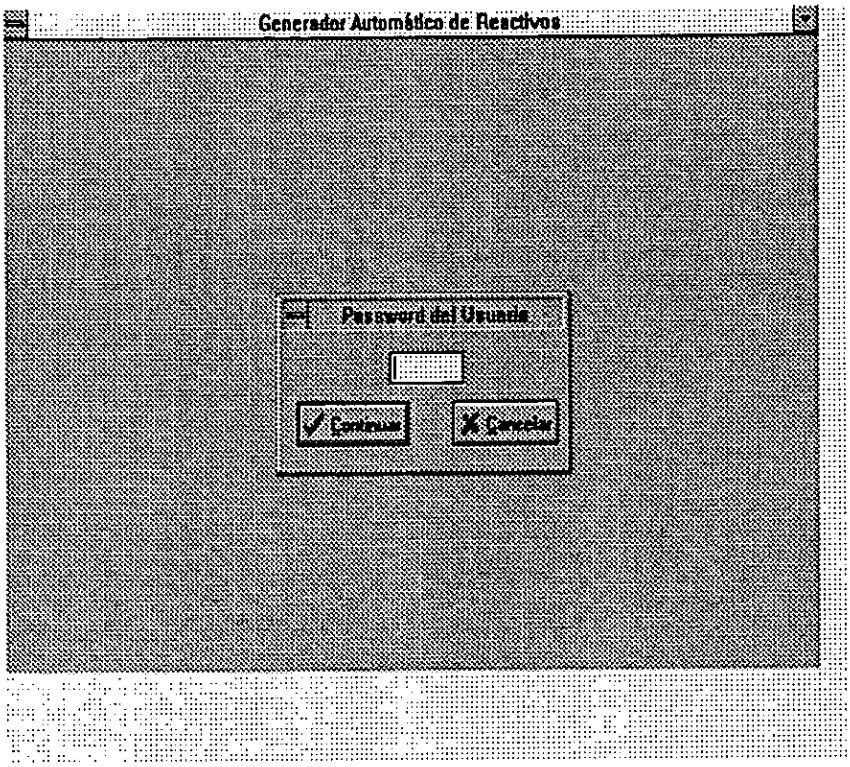


FIGURA 31: VENTANA PARA PROPORCIONAR EL PASSWORD DEL USUARIO

Aquí el usuario digitará su '*password*' -de 4 dígitos-, dando "click" al botón "Continuar" para proseguir con el acceso al 'menú' principal (ver figura 29 -página 128-).

Alternativamente, al dar "click" al botón "Cancelar" o en caso de no coincidir el dato proporcionado con el '*password*' correspondiente a la 'clave' digitada en la 'ventana' precedente (ambos datos registrados en la 'base de datos' del GAR) y, previo mensaje al efecto, aparecerá nuevamente la 'ventana' de la figura 30.

Por otra parte, con referencia al objetivo B) acerca del registro y mantenimiento de datos de cada usuario, así como a sus 'permisos de acceso' al 'menú' principal y a bancos de reactivos, cuando se tienen aquéllos suficientes para acceder a la opción Utilerías|Seguridad del 'menú' principal, es decir, cuando la 'opción' mencionada aparece "habilitada" al usuario y es seleccionada, se presenta la 'ventana' mostrada en la siguiente página (figura 32):

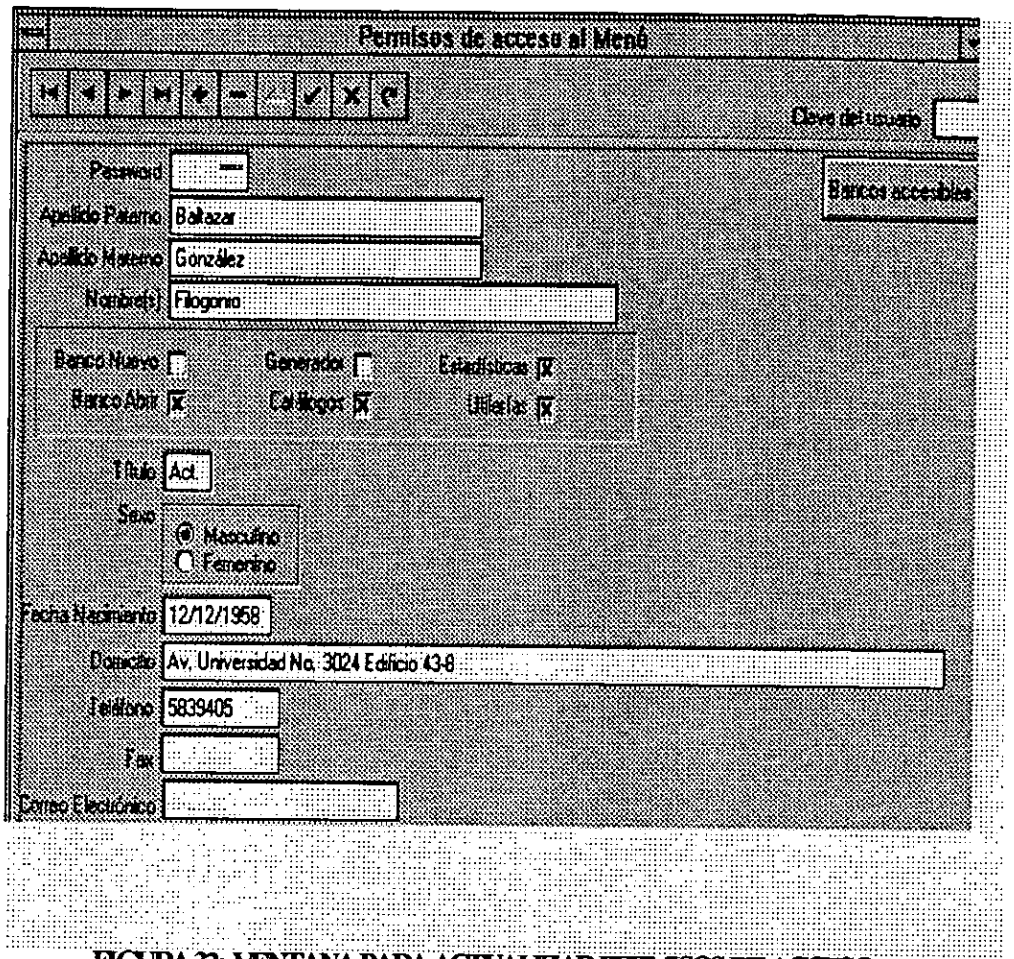


FIGURA 32: VENTANA PARA ACTUALIZAR PERMISOS DE ACCESO AL MENÚ Y DATOS PERSONALES DEL USUARIO

Según se de "click" a los botones del 'componente' ***'TDBNavigator'*** sito en la parte casi superior izquierda de la figura, es posible consultar, registrar y/o actualizar la 'base de datos' del GAR, en lo relativo a su 'tabla' de usuarios, accediendo a datos tales como '*password*', 'permisos de acceso' al 'menú' principal y datos personales. El uso de dicho 'componente', entre otros, se detalla en la 'ayuda' de la aplicación. Adicionalmente, al dar "click" al botón "Bancos accesibles" sito en la parte superior derecha de la figura en cuestión, es posible actualizar para el usuario cuyos datos estén presentes en la 'ventana' actual, sus 'permisos de acceso' a bancos de reactivos, dando paso de esta forma a lo siguiente (figura 33):

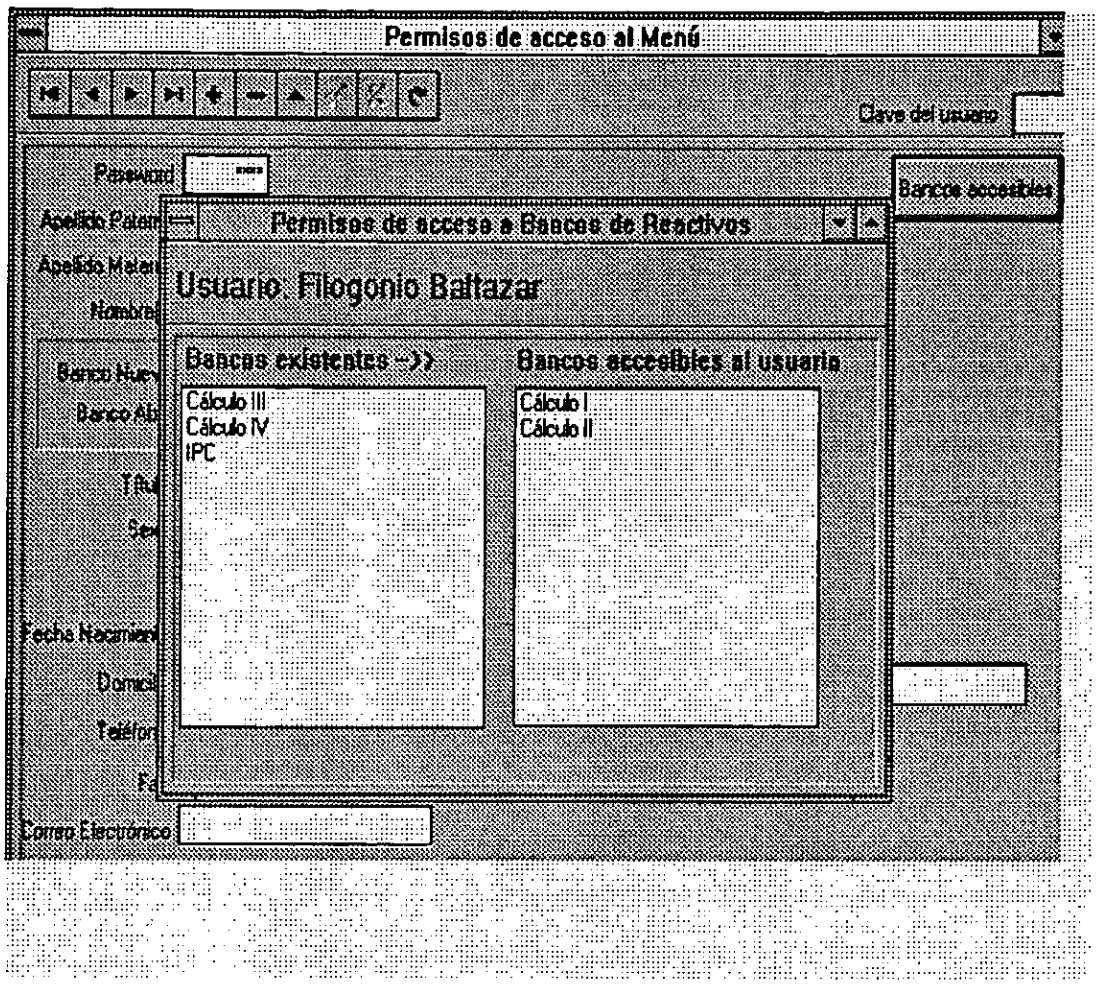


FIGURA 33: VENTANA PARA ACTUALIZAR PERMISOS DE ACCESO A BANCOS DE REACTIVOS

En esta 'ventana' basta seleccionar del área "Bancos existentes" el nombre de aquél que se desee pueda ser accedido por el usuario en cuestión, darle "click", 'arrastrarlo y soltarlo' (*'drag & drop'*) hacia el área "Bancos accesibles al usuario". En sentido inverso, o sea para revocar al usuario el permiso de acceso a un banco de reactivos, se debe seleccionar del área de "Bancos accesibles al usuario" aquél que no deba ser accedido y, efectuar doble "click" en el mismo para revocar el permiso respectivo, enviando la aplicación en este caso un 'diálogo' de confirmación, pudiendo continuar con el otorgamiento o revocación de permisos para el usuario en cuestión o, regresar a la 'ventana' anterior (figura 32), dando doble "click" en el cuadro de menú de control de la 'ventana' actual.

6.4.2 MANEJADOR DE BANCOS DE REACTIVOS

El objetivo de este 'módulo' es permitir la actualización de los bancos contemplando la posibilidad de su creación y/o actualización, así como registrar y/o actualizar a los propios reactivos, en enlace con el 'módulo' de seguridad del GAR y en función de los permisos de cada usuario en materia de acceso al 'menú' principal, así como a los propios bancos; en resumen: diversos usuarios del GAR pueden tener acceso a distintos bancos de reactivos, así como a diferentes 'opciones' del 'menú'.

6.4.2.1 Creación del banco de reactivos

Para trabajar con un banco es necesario que exista y por esta razón, obvia pero no trivial en materia de 'implementación' de la aplicación de cómputo, la misma permite crear un banco a la vez y cuando ha sido creado -siempre que el usuario tenga los permisos suficientes-, no es posible "borrarlo" desde el GAR. En todo caso es un responsabilidad del administrador de la aplicación el realizar este tipo de acciones, delicadas de suyo, operativamente hablando.

Por lo anterior, se puede trabajar sin temor a que la información "se pierda", además de que la aplicación en modo 'standalone' contempla una opción para **'respaldar'** los bancos y en su modalidad 'multiusuario', contempla a esta función como una responsabilidad del administrador del 'servidor' empleado al efecto. De hecho, el 'programa' de cómputo asigna al banco un conjunto de 'archivos' o 'tablas' en un disco de la PC o en algún dipositivo de almacenamiento del 'servidor' -según el caso-; es ahí donde se preserva la información y al conjunto de tales 'archivos' es a lo que se denomina banco de reactivos.

Desde un punto de vista técnico a ese conjunto de 'tablas' interrelacionadas es a lo que se denomina genéricamente la **'base de datos de la aplicación'** y al efecto, posterior a la instalación del GAR -momento en que se crea el **'directorio'** o **'carpeta'** \BancUNAM.Dat-, se graba la 'base de datos' correspondiente al banco de reactivos en un **'subdirectorio'** o 'carpeta' \BancUNAM.Dat\Clave de la asignatura, donde dicha clave debe ser de cuatro caracteres y acorde al Catálogo de planes de estudios de la UNAM.

Para crear un banco de reactivos -en caso de tener el permiso respectivo-, se toma del 'menú' la opción BancoDeReactivos|Nuevo, apareciendo la siguiente 'ventana':

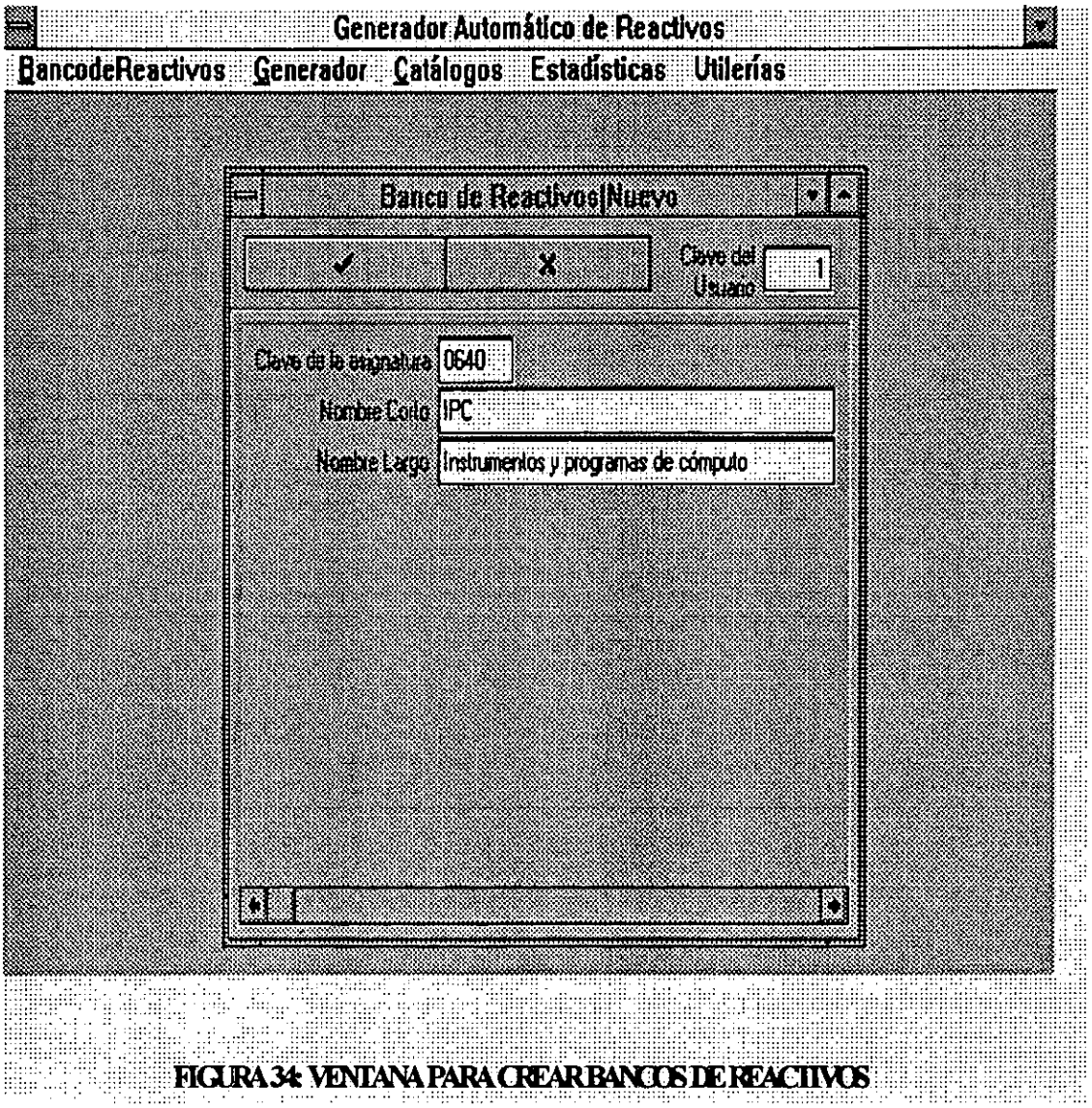


FIGURA 34: VENTANA PARA CREAR BANCOS DE REACTIVOS

Aquí procede teclear los 4 dígitos de la clave de la asignatura acorde a los planes de estudio de la UNAM en el primer espacio destinado al efecto, apareciendo entonces -a su derecha-, un segundo espacio para verificar que se digitó correctamente, prosiguiendo la captura del nombre corto o coloquial de la asignatura, como: Cálculo III, Analítica I, así como su nombre oficial en la institución, correspondiendo respectivamente: Cálculo Diferencial e Integral III y Geometría Analítica I. La captura de los nombres coloquial y oficial es obligatoria.

Para finalizar la creación y el registro del banco de reactivos hay que oprimir el botón izquierdo -"Confirmar"- del 'objeto' *'TDBNavigator'* o llanamente **"navegador"**, sito en la parte casi superior izquierda de la 'ventana' y para anular dicha acción, basta oprimir el otro botón -"Cancelar"-, regresando así al 'menú' principal. De oprimir el botón para confirmar el registro, aparecerá un 'diálogo' que cuestiona si se desea actualizar los 'catálogos' del banco (temas, subtemas y objetivos) o en caso contrario, se regresa al 'menú' referido.

Como antes he mencionado, la explicación detallada de la 'funcionalidad' del "navegador" está contenida en la 'ayuda' de la aplicación, pero aquí debo hacer notar que solamente aparecen dos de sus botones a diferencia de las otras 'ventanas' del GAR hasta aquí presentadas y que, según lo requiere su GUI, se activan o no diversos botones del objeto en cuestión.

Por otra parte y como también antes expliqué, por la forma como diseñé y construí la aplicación, por cada banco de reactivos se crea un 'subdirectorio' o 'carpeta', "bajo" o en la 'carpeta' propia de la aplicación: \BancUNAM.Dat. Justamente el nombre del 'subdirectorio' en cuestión es la clave de la asignatura previamente mencionada: de ahí su importancia. Aqui estoy suponiendo, espero no erróneamente, que los usuarios del GAR trabajarán estableciendo una relación biunívoca -uno a uno-, entre la asignatura con su banco de reactivos. Esta idea no es vana -desde mi punto de vista-, pues de proceder así, técnicamente es posible integrar con posterioridad y de forma muy sencilla los bancos de diferentes profesores o grupos de trabajo, en uno solo, para la asignatura de que se trate a fin de poder enriquecerlos y compartirlos.

6.4.2.2 Apertura del banco de reactivos

Para trabajar con los reactivos de un banco es necesario abrirlo desde la GUI. Como he mencionado, cada uno de ellos debe tener asignada una clave para poder ser manipulado desde el GAR y en el mejor de los casos, dicha clave debe corresponder a la registrada en el catálogo de planes de estudio de la UNAM. Contemplando lo anterior y siempre que el usuario tenga permisos suficientes, al seleccionar del 'menú' principal la opción BancoDeReactivos|Abrir, aparece lo siguiente (figura 35):

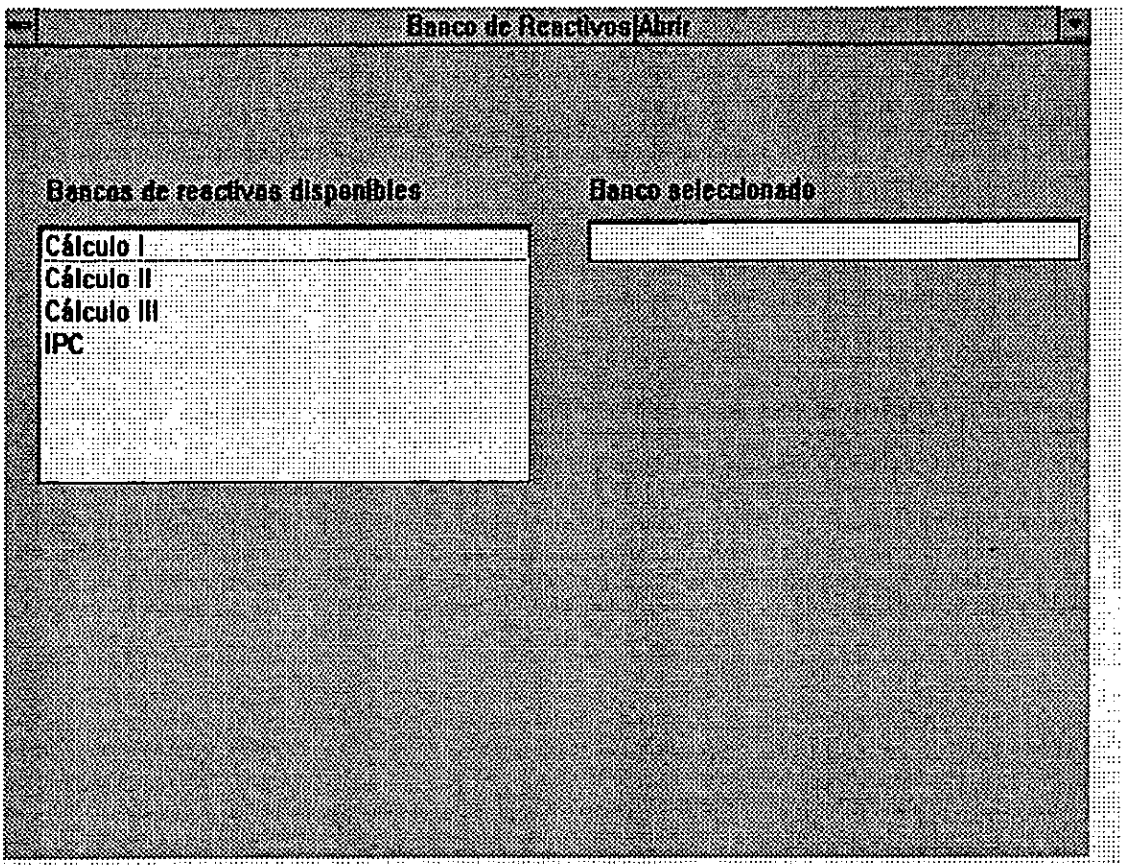


FIGURA 35: VENTANA PARA ABRIR EL BANCO DE REACTIVOS POR ACTUALIZAR

En esta 'ventana' basta seleccionar del área "Bancos de reactivos disponibles" el nombre de aquél que se desea actualizar al darle "click", 'arrastrarlo y soltarlo' en el área "Banco seleccionado", enviando la aplicación un 'diálogo' de confirmación, pudiendo continuar en caso de respuesta afirmativa con la actualización de reactivos o, en caso contrario, dejando la 'ventana' en su estado inicial. Desde aquí siempre es posible regresar al 'menú' principal al dar doble "click" en el cuadro de menú de control.

Cabe aclarar que en la lista de "Bancos de reactivos disponibles" solamente aparecerán aquéllos a los que tiene 'permiso de acceso' el usuario en cuestión, *i.e.*, es posible que haya más bancos registrados en el GAR que los que aparecen en la lista mencionada. Usuarios con permisos suficientes, como el de administración, pueden acceder a todos.

6.4.2.3 Actualización del banco de reactivos

El 'proceso' de actualización comprende las acciones básicas que en general operan sobre 'archivos' de datos: altas (*insert*), bajas (*delete*) y cambios (*update*); en nuestro caso, de reactivos.

Contemplando lo antes expuesto y en función de la evaluación de los alumnos, será posible retroalimentar y evaluar al banco (ver figura 25, página 112). ¿Cómo? pues valorando a su vez los reactivos, hecho que permitirá decidir con respecto a su situación en el banco de la asignatura correspondiente. En este contexto, las decisiones posibles sobre un reactivo que forma parte del banco, son: eliminarlo, modificar su contenido y/o sus datos propios (*e.g.* nivel taxonómico). En cuanto a un reactivo que no forme parte del banco, la única opción posible es incluirlo en este dándolo de alta.

Considerando lo anterior y retomando la 'funcionalidad' del GAR, enseguida de la selección y apertura del banco aparece la 'ventana' para actualizar reactivos.

Por lo anterior y solamente para aclarar ideas, enseguida se muestra la figura 36 que contiene un fragmento de la 'ventana' correspondiente, donde por ejemplo, no aparecen los botones con los que cuenta (*v.g.* "Buscar reactivo"), ni los datos detallados del registro, pues la 'funcionalidad' de esta parte de la GUI varía sustancialmente, y mucho, en razón del tipo de reactivo que se esté accediendo o se desee registrar o modificar.

Describir aquí lo que antes comenté, sería larguísimo*. Por lo mismo todo esto se explica, y muy detalladamente, en la 'ayuda' del GAR.

---

*De hecho en los borradores del intento de tesis de hace diez años, la mayoría de sus textos correspondían a la explicación del funcionamiento del GAR y había contemplado muy poco de aspectos teóricos y técnicos en la materia. No es justificación, pero que bueno que no presente aquélla tesis sino esta, la cual considero comparativamente muchas, pero muchas veces mejor -aunque no de lo mejor-.... creo que valió la pena la espera. "Metaconclusión": ¡Cómo nos cambia el paso del tiempo!.

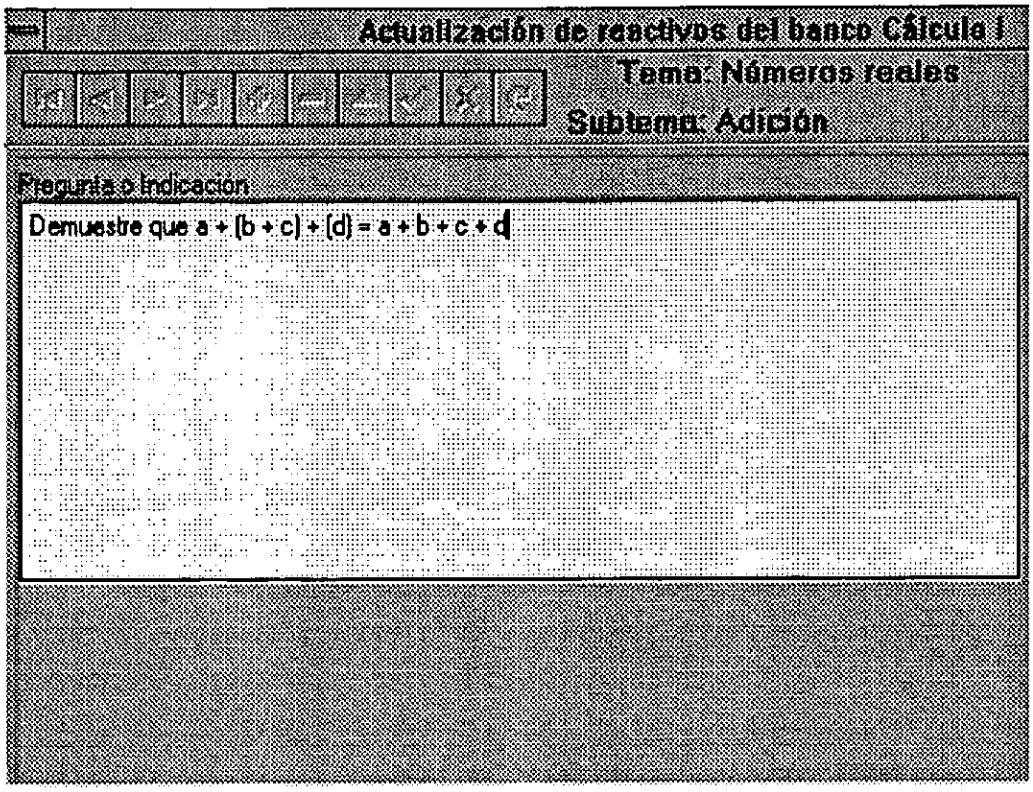


FIGURA 36: VENTANA PARA ACTUALIZAR REACTIVOS

Esta 'ventana' también basa su 'funcionalidad' básica en el 'componente' "navegador" sito en su parte casi superior izquierda y, según se "opriman" sus botones -se de "click" en ellos-, es posible consultar, registrar (o insertar o dar de alta), modificar (editar) o borrar reactivos.

La secuencia lógica en que el GAR permite ingresar al banco los atributos o características de cada reactivo, cuando se da de alta, inserta o registra , es:

1° Tema
2° Subtema
3° Objetivo
4° Nivel Taxonómico
5° Tipo de reactivo

La selección de los datos mencionados se efectuan al dar doble "click" en el renglón correspondiente del 'objeto' ***'TDBGrid'*** de Delphi, que es mostrado enseguida y cuyo uso es también explicado al usuario, con todo detalle, en la 'ayuda':

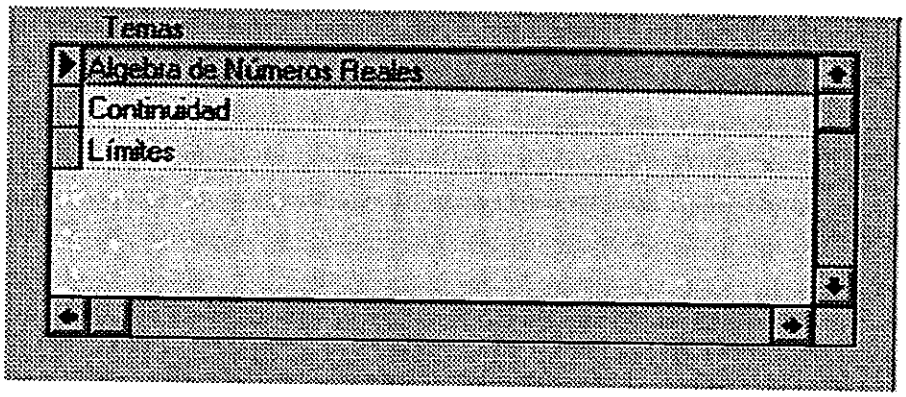


FIGURA 37: OBJETO TDBGRID

Previamente al registro de reactivos debe haberse realizado lo propio para los temas, subtemas y objetivos de la asignatura del caso, a través del 'menú' principal en su opción de "Catálogos" (ver figura 29 -página 128-). Los correspondientes a nivel taxonómico y tipo de reactivo están integrados al GAR y son solamente para lectura -*i.e.* no se pueden modificar-; ambos corresponden a los expuestos en el diseño conceptual de este documento, en sus páginas 107 y 115, respectivamente.

Justo aquí debo mencionar que el GAR considera el registro del objetivo como opcional, *i.e.*, no es obligatorio -para cualquier aclaración sobre la razón por la cual no es indispensable el objetivo, favor de remitirse a la página 144-, pero resulta mucho mejor tener un banco organizado como se ha citado en la definición conceptual de banco de reactivos, prácticamente desde cualquier punto de vista (figura 25, página 109).

Prosiguiendo con la descripción que nos ocupa, las demás características del reactivo las va proporcionando automáticamente el GAR. Por ejemplo, su número en el banco -estrictamente de control interno para la aplicación-, se asigna conforme al registro que se esté digitando; de este modo, si se graba el cuarto reactivo, el número que tendrá asignado

en el banco, es el 4. Lo mismo sucede en el caso de que la aplicación esté funcionando en modo 'multiusuario', de forma 'transparente' para aquélllos que así lo empleen.

Después de proporcionar los datos anteriores, procede capturar o copiar el contenido o texto de la pregunta, problema o indicación del reactivo, así como en su caso, la imagen representativa del mismo y/o sus opciones de respuesta o sus elementos, lo cual se efectúa a través de la 'ventana' de la figura 36, según lo requiera la GUI.

De vuelta al anecdotario; en el texto viejo tenía escrito "...por una pantalla, se entiende un sector de la pantalla total del monitor de la computadora, el cual está contenido en un marco que define los límites de la misma."(sic). El 'desarrollador' de Delphi define las 'ventanas' como 'objetos' que pueden ser 'programados' a través de sus 'eventos' (e.g. OnActivate), así como en 'servidores' de datos, de forma correspondiente pero diferente, se 'programan' 'triggers'.

Por otra parte, la secuencia lógica en que generalmente se ingresa el texto de la respuesta o la opción de la respuesta correcta y las opciones que he denominado "distractores" para los reactivos es:

1o. Respuesta correcta
2o. Distractores (en su caso)

Para los tipos de reactivos de jerarquización o correspondencia, explicados en las páginas 120 a la 123, el orden lógico para la captación de sus "elementos" es:

A) Jerarquización.- Conforme la secuencia lógica en en que se plantea el reactivo: de menor a mayor, de mayor a menor, etc.

B) Correspondencia.- Según la correspondencia de y entre los "elementos" del reactivo -¡valga la redundancia!-, es decir, se capta la primera frase de aseveración y su asociada, prosiguiendo con la segunda y su asociada y así, sucesivamente.

En el caso de reactivos donde el alumno selecciona la respuesta de un conjunto posible (ver figura 28 -página 115-), el GAR asigna el orden en que aparecerán para el estudiante las posibles opciones de respuesta y generalmente, éste será diferente al orden en que se registraron; empero la aplicación hará que la 'base de datos' tenga almacenado el código correspondiente a la respuesta u opción correcta del reactivo.

Al concluir el registro o actualización de cada reactivo es posible:

* Continuar registrando reactivos del mismo tema, subtema y objetivo y cambiar o mantener el tipo de reactivo y/o nivel taxonómico.

* Cambiar de tema y/o subtema y/o objetivo, para registrar otros reactivos.

* Suspender el registro al cerrar la 'ventana', en cuyo caso se regresa al 'menú'.

La fecha de actualización, es registrada automáticamente tanto para el banco, como para el reactivo que se haya afectado.

Toda la programación que había realizado para el registro de reactivos en el generador versión MS-DOS "se murió" en términos de que no tiene sentido para el ambiente Windows; mismo comentario para las 'rutinas' del "minieditor"de textos' que había 'programado'. Con relación al manejo de imágenes ni siquiera -en sueños- estaba contemplado en mi trabajo precedente. En lugar de lo anterior, actualmente 'implementé'

alguna 'codificación' para los 'eventos' de los 'componentes' TDBNavigator y TDBMemo de Delphi: estos 'objetos' se encargan prácticamente "solitos" de hacer todo lo que acabo de mencionar, en lo relativo a los 'procesos' de registro y/o actualización del banco de reactivos y en el 'proceso' de captura o copia de textos –e imágenes–, respectivamente.

Aquí no debe ser simplista la reflexión, pues la ganancia en el uso de los 'componentes' mencionados -entre otros-, se traduce en el tiempo invertido en conocer y comprender la filosofía de 'programación' del 'desarrollador de aplicaciones', en aprender a usar los atributos o **'propiedades'**, 'eventos' y **'métodos'** de sus 'objetos' o 'componentes', solamente por citar algunos aspectos relevantes en la materia. En otras palabras: "No cualquier burro toca la flauta; para eso necesita la flauta y cubrir el requisito animal de soplar -no cualquier animal sopla-"; otra frase para la reflexión que le escuché a un merolico que anunciaba las bondades de ciertas hierbas medicinales en la Alameda Central y que entraña una gran verdad: "El tonto cree todo lo que el sabio investiga....". Estos comentarios los pongo sin ningún afán peyorativo, al pensar que si no trabajamos e investigamos lo suficiente -y aún más-, en áreas verdaderamente técnicas y científicas, nuestro país, irremediablemente, continuará en lo que se ha denominado subdesarrollo.

6.4.3 GENERADOR DE REACTIVOS

A continuación se expone la narrativa de la 'funcionalidad' genérica del 'módulo' de la aplicación al cual denominé "Generador" (automático de reactivos). Sus objetivos son:

> A) Posibilitar al usuario la elección -entre otras características-, de los temas o, de los temas y subtemas para que el GAR "seleccione al azar" los reactivos correspondientes al banco de la asignatura de que se trate, donde la selección de objetivos la realiza el sistema aleatoriamente, derivado de la aplicación del concepto pedagógico relativo a la evaluación sumaria (ver página 109).

B) Generar reactivos en forma automática almacenándolos en un medio magnético al efecto, sean para exámenes o tareas, posibilitando la impresión de los mismos.

6.4.3.1 Selección de características de los reactivos y generación de estos

Con relación a la cobertura del objetivo A), para generar reactivos es necesario seleccionarlos del banco con que se cuente y tener los 'permisos de acceso' necesarios al efecto. Cuando se accede a la opción "Generador" del 'menú' principal -ver figura 29, página 128-, se presentan las opciones básicas para generar los reactivos, sea por "Temas" o, por "Temas y Subtemas".

Un psicólogo o un pedagógo astuto preguntaría: ¿dónde queda la cuestión de los objetivos de aprendizaje?.

Debo volver a comentar -como mencioné en el marco conceptual-, que el GAR lo diseñe y construí con base en la evaluación sumaria y por tanto en el momento que el usuario selecciona un tema para la generación de reactivos, aquéllos que tienen opción a formar parte del conjunto seleccionado, corresponderán efectivamente y exclusivamente a dicho tema, siendo posteriormente seleccionados aleatoriamente por el GAR. Como consecuencia, los objetivos de aprendizaje del reactivo también son seleccionados de igual forma en función de las posibilidades de un equipo de cómputo de generar números aleatorios o en forma pseudoaleatoria, sobre lo cuál hay mucho que discutir en términos de su validez; empero aquí si no hay de otra, pues si no se toma el generador de ese tipo de números del *hardware* y *software* disponible ¿de que otra forma práctica, en el contexto de la aplicación de cómputo, sería posible instrumentar el GAR?.

Prosiguiendo con la narrativa, al dar "click" en la 'opción del menú' Generador|Temas aparece la siguiente 'ventana':

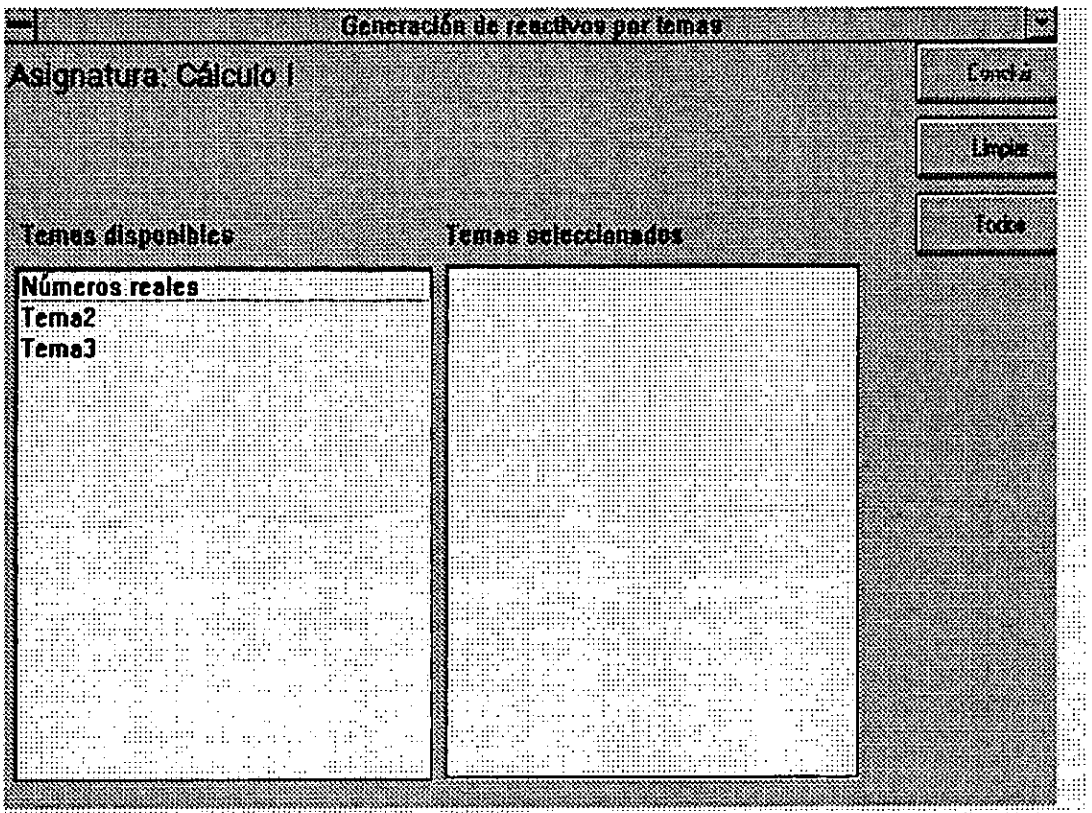


FIGURA 38: VENTANA PARA SELECCIONAR TEMAS

donde inicialmente aparece un 'diálogo' similar al 'implementado' para la digitación del *'password'* (ver figura 31 -página 131-), que solicita al usuario el número de reactivos que desea sean generados. De proporcionar ese dato y oprimir el botón "Continuar" se presenta la 'ventana' como se muestra en la figura y con un procedimiento de seleccionar, 'arrastrar y soltar', totalmente análogo al explicado en la 'ventana' para abrir bancos de reactivos (figura 35 -página 137-), se seleccionan los temas que se desean para la generación automática de los reactivos y, al dar "click" al botón "Concluir" sito en la parte superior derecha de la 'ventana', procederá la generación automática. De oprimir el botón "Todos" es innecesario realizar el proceso de 'drag & drop', pues automáticamente se selecciona el total de temas registrados y se efectua la automáticamente la generación. De oprimir el botón "Limpiar", la 'ventana' se vuelve a presentar como en la figura, cancelando el 'proceso' de generación y limpiando cualquier selección de temas previa. Al salir de la 'ventana' se regresa al 'menú'

principal. De forma totalmente análoga se procede para la selección de temas y subtemas. Así pues, en función del número de reactivos solicitado la aplicación "busca'' en el banco y "selecciona al azar" aquéllos que cumplen con las especificaciones dadas por el usuario en lo referente al (los) tema(s) y/o subtema(s), sin considerar el tipo de reactivo. En resumen: el GAR selecciona pseudoaleatoriamente los objetivos de aprendizaje por evaluar y los reactivos para la propuesta, a partir del contenido del banco y de los temas y subtemas especificados por el usuario; todo esto concordando con la evaluación sumaria.

En este contexto, para que el GAR pueda concluir su 'tarea', la única limitante es que el banco cuente con un número suficiente de reactivos para poder realizar su selección; esto es, no es posible generar diez reactivos, si el banco cuenta sólo con ocho. En caso de no haber reactivos suficientes, esto se notifica al usuario quedando la 'ventana' en su estado inicial.

Prosiguiendo con la descriptiva, en el momento en que la aplicación cuenta con un conjunto de reactivos definido -armado por la generación de reactivos 'procesada'-, los envía al **'portapapeles'** de Windows para efectos de su posterior impresión. Adicionalmente, al generar los reactivos, se crea un 'archivo' de control que contiene la clave de la materia, fecha y hora del 'proceso' y los códigos de respuesta correspondientes a dichos reactivos. Esto posibilita calificar manualmente los exámenes o tareas correspondientes de una manera ágil y para el caso de reactivos donde el alumno selecciona la respuesta correcta, posibilita calificarlos automáticamente, pues bastaría captar a través de algun medio, las respuestas de los educandos al examen o tarea y compararlas con el 'archivo' antes mencionado.

6.4.3.2 Impresión de exámenes o tareas

En virtud de la variabilidad de impresoras y configuraciones posibles no 'programé' esta parte de la aplicación, a pesar de que el requerimiento de impresión para completar la 'funcionalidad' de la aplicación es indispensable. En su lugar 'codifiqué' una 'interface' que permite copiar cada conjunto de reactivos generados al 'portapapeles' y de ahí a un 'procesador de texto' que se pueda ejecutar "simultáneamente" al GAR. En este sentido, se envían al 'portapapeles' los reactivos en cuestión con el siguiente formato general:

Asignatura y Fecha de generación
Tema(s) y subtemas(s) de los reactivos
Nombre del usuario que generó los reactivos

Número del reactivo	Texto del planteamiento
para impresión	Imagen representativa (en su caso)

Imagen representativa (en su caso)

Para reactivos donde el alumno elabora la respuesta, un conjunto de líneas al efecto:

_____
_____
_____

se envían 5 líneas por *default* para reactivos de respuesta abierta y una sola línea para reactivos de respuesa breve o de complementación

Para reactivos de falso/verdadero:

Falso ( )	Verdadero ( )

Para reactivos de opción múltiple:

1)	Texto de la primera opción

2)	Texto de la segunda opción
•	•
•	•
n)	Texto de la opción n-ésima

Para reactivos de jerarquización:

1)	Texto del primer elemento
2)	Texto del segundo elemento
•	•
•	•
n)	Texto del elemento n-ésimo

Para reactivos de correspondencia:

1)	Texto del primer elemento	Texto asociado
2)	Texto del segundo elemento	Texto asociado
•	•	
•	•	
n)	Texto del elemento n-ésimo	Texto asociado

Donde el valor máximo de n es 5.

Al realizar la 'interface' de Windows entre el GAR y el 'procesador de texto' que se prefiera, basta formatearlos al gusto en la segunda aplicación mencionada e imprimir.

Una posiblidad, no se si plausible, es que se deseen **'guardar'** esos reactivos como un 'archivo' del 'procesador de texto', empero, esta acción queda totalmente a criterio de los usuarios y al efecto se debería considerar el espacio que se ocuparía en disco por cada conjunto de reactivos generado y la seguridad de acceso a ese tipo de documentos. Como decía a nuestros alumnos uno de mis amigos y compañero de banca de la Facultad, cuando él daba clases ahí y yo fungía como su ayudante: "Al cliente, lo que pida".

6.4.4 CATÁLOGOS, ESTADÍSTICAS Y UTILERÍAS

En virtud de que la 'funcionalidad' de estas opciones del 'menú' principal (figura 29, página128) están explicadas en detalle en la 'ayuda' del GAR, aquí solamente hago mención de que el 'módulo' de "Catálogos" permite registrar y/o actualizar los temas, subtemas y objetivos de aprendizaje de cada asignatura a través de 'ventanas' como la mostrada enseguida y mediante el uso de "navegadores", espacios para edición y uno que otro botón auxiliar:

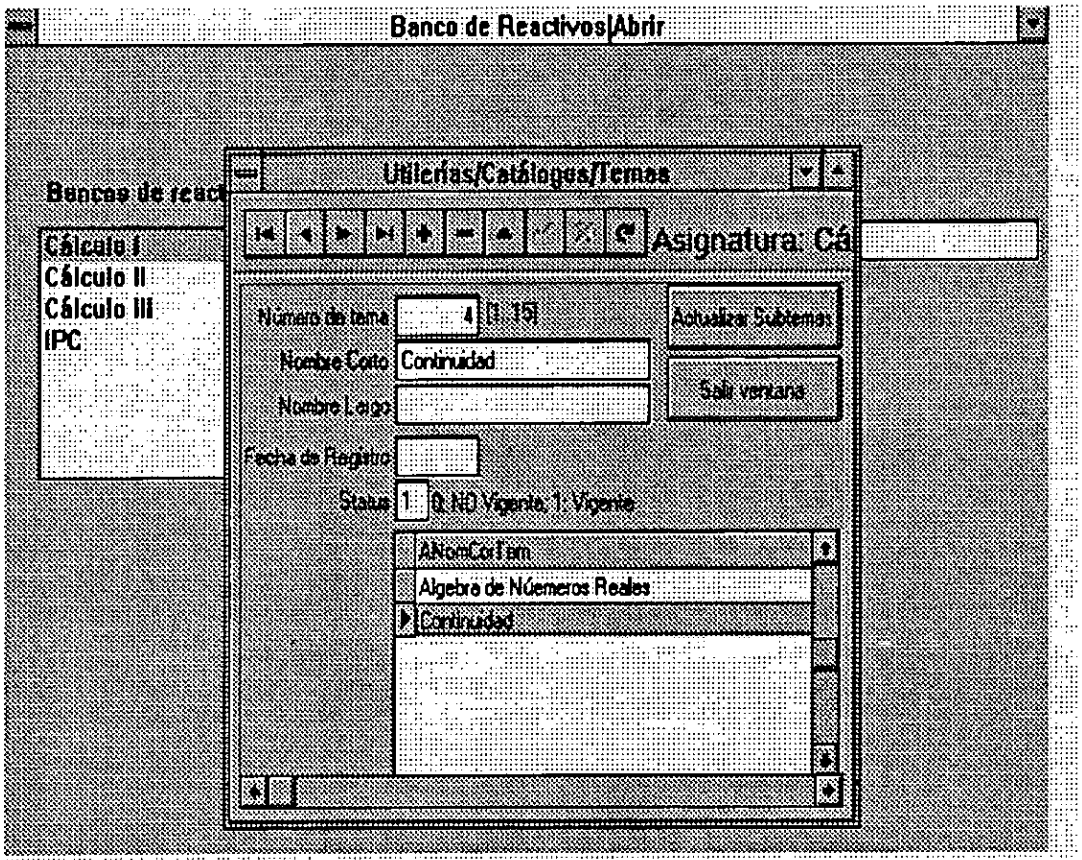


FIGURA 39: VENTANA PARA REGISTRAR Y/O ACTUALIZAR TEMAS

Por otro lado, la opción de "Estadísticas" del 'menú' genera un documento que se envía al 'portapapeles' o a un 'archivo', que contiene el desglose del número de reactivos del banco como se muestra en la siguiente página:

Nombre del Banco: *Total de reactivos*			Fecha
Tema 1	*Total del Tema 1*		
	Subtema 1	*Total del Subtema 1*	
		Objetivo 1	*Total del Objetivo 1*
		Objetivo 2	*Total del Objetivo 2*
		•	•
		•	•
		Objetivo n	*Total del Objetivo n*
	Subtema 2	*Total del Subtema 2*	
		Objetivo 1	*Total del Objetivo 1*
		Objetivo 2	*Total del Objetivo 2*
		•	•
		•	•
		Objetivo m	*Total del Objetivo m*
	Subtema 3	*Total del Subtema 3*	
	•	•	•
		•	•
Tema 2	*Total del Tema 2*		
•	Subtema 1	*Total del Subtema 1*	
•	•	Objetivo 1	*Total del Objetivo 1*
•	•	Objetivo 2	*Total del Objetivo 2*
•	•	•	•
•	•	•	•

La opción de "Utilerías" en su opción de seguridad, da la posibilidad de acceder a las 'ventanas' de las figuras 32 y 33. La otra opción permite 'respaldar' en ***'diskette'*** los bancos en cuestión sólo a aquéllos usuarios con permisos suficientes.

Finalmente, la 'ayuda' del generador funciona exactamente como la de Windows, por lo que al usuario acostumbrado a este ambiente de trabajo le resultará muy sencillo su uso y el del propio GAR.

6.4.5 INSTALACIÓN DE LA APLICACIÓN Y EJECUCIÓN

Para instalar el GAR solamente hay que introducir el *'diskette'* correspondiente en un *'drive'* de la PC, cambiarse desde el **'explorador'** o **'administrador de archivos'** de Windows al *'drive'* respectivo y seleccionar el 'archivo' InstGAR.Exe: aparecerá un mensaje de notificación de la instalación y solamente resta esperar a que concluya la acción en curso.

De no haber espacio suficiente en disco se avisa al usuario y se aborta el proceso. De existir una 'carpeta' \BancUNAM.Dat en el disco del equipo se notifica para que el usuario decida entre continuar o abortar la 'tarea'; al estar en esta situación y en caso de continuar, los 'programas' y las estructuras de la 'base de datos' se graban bajo el 'directorio' mencionado. En caso de reinstalación procede lo mismo, sin alterar el contenido de otros 'subdirectorios' o 'carpetas' que existan bajo \BancUNAM.Dat y que corresponderán muy posiblemente a bancos de reactivos preexistentes.

Una vez concluida la instalación y para iniciar la ejecución del 'programa', basta proceder como con cualquier otra aplicación para Windows con el icono enseguida mostrado:

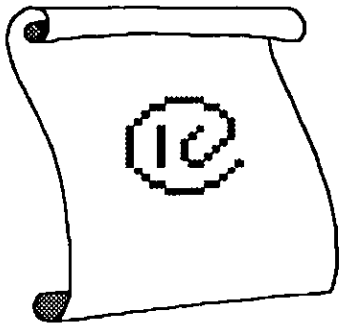


FIGURA 40: ICONO DEL GAR

Como he explicado en la página 130 (ver figura 30), al ejecutar el GAR aparece la 'ventana' para digitar la 'clave del usuario' y enseguida de la instalación, solamente es posible continuar

proporcionando aquéllla registrada o que existe hasta ese momento y que denominé "Clave de usuario de instalación".

Al acceder por el 'menú' principal vía Utilerías|Seguridad a la 'ventana' para actualizar permisos de acceso al 'menú' y datos personales del usuario (ver figura 32 -página 132-), la contraseña y los datos mencionados son modificables, a fin de que la persona que se desee tenga todos los permisos posibles, es decir, se pueden editar los datos del "usuario de instalación" con los datos de que se deseen, considerando que por *default* la clave 1 del usuario en cuestión tiene registrados 'permisos de acceso' a todos los 'comandos' u 'opciones del 'menú' principal. Sin embargo, existe la opción de borrar la clave y el registro del usuario del caso y darse de alta uno mismo como tal, pero en ese caso la clave asignada es la número 2. Al proseguir con el registro de usuarios, se asigna su clave por número consecutivo.

En cualquiera de las dos opciones planteadas y de no haber registro de otros usuarios con permisos suficientes, cuidado con quitarse los correspondientes para acceder al 'módulo' de seguridad por que entonces, para dar de alta otros usuarios o modificar permisos, ¡se debe reinstalar el GAR completo!.

Hecho lo anterior basta, definir usuarios, crear bancos de reactivos, ponerse a trabajar y, ... ¡suerte!

7 APLICACIÓN DE LA VARIANTE DE LA METODOLOGÍA DE YOURDON AL DESARROLLO DEL GAR

Como planteé en el capítulo 5, la variante de la metodología de Yourdon, en adelante denominada simplemente como la variante, mantiene los tres conceptos ampliamente conocidos en materia de desarrollo de sistemas: análisis, diseño e 'implementación'. Por lo anterior y derivado del "ajuste" o *customize* de dicha variante para el GAR, en esta parte del documento haré énfasis en los mismos; el correspondiente al proyecto es mostrado a través del siguiente DFD:

proporcionando aquéllla registrada o que existe hasta ese momento y que denominé "Clave de usuario de instalación".

Al acceder por el 'menú' principal vía Utilerías|Seguridad a la 'ventana' para actualizar permisos de acceso al 'menú' y datos personales del usuario (ver figura 32 -página 132-), la contraseña y los datos mencionados son modificables, a fin de que la persona que se desee tenga todos los permisos posibles, es decir, se pueden editar los datos del "usuario de instalación" con los datos de que se deseen, considerando que por *default* la clave 1 del usuario en cuestión tiene registrados 'permisos de acceso' a todos los 'comandos' u 'opciones del 'menú' principal. Sin embargo, existe la opción de borrar la clave y el registro del usuario del caso y darse de alta uno mismo como tal, pero en ese caso la clave asignada es la número 2. Al proseguir con el registro de usuarios, se asigna su clave por número consecutivo.

En cualquiera de las dos opciones planteadas y de no haber registro de otros usuarios con permisos suficientes, cuidado con quitarse los correspondientes para acceder al 'módulo' de seguridad por que entonces, para dar de alta otros usuarios o modificar permisos, ¡se debe reinstalar el GAR completo!.

Hecho lo anterior basta, definir usuarios, crear bancos de reactivos, ponerse a trabajar y, ... ¡suerte!

7 APLICACIÓN DE LA VARIANTE DE LA METODOLOGÍA DE YOURDON AL DESARROLLO DEL GAR

Como planteé en el capítulo 5, la variante de la metodología de Yourdon, en adelante denominada simplemente como la variante, mantiene los tres conceptos ampliamente conocidos en materia de desarrollo de sistemas: análisis, diseño e 'implementación'. Por lo anterior y derivado del "ajuste" o *customize* de dicha variante para el GAR, en esta parte del documento haré énfasis en los mismos; el correspondiente al proyecto es mostrado a través del siguiente DFD:

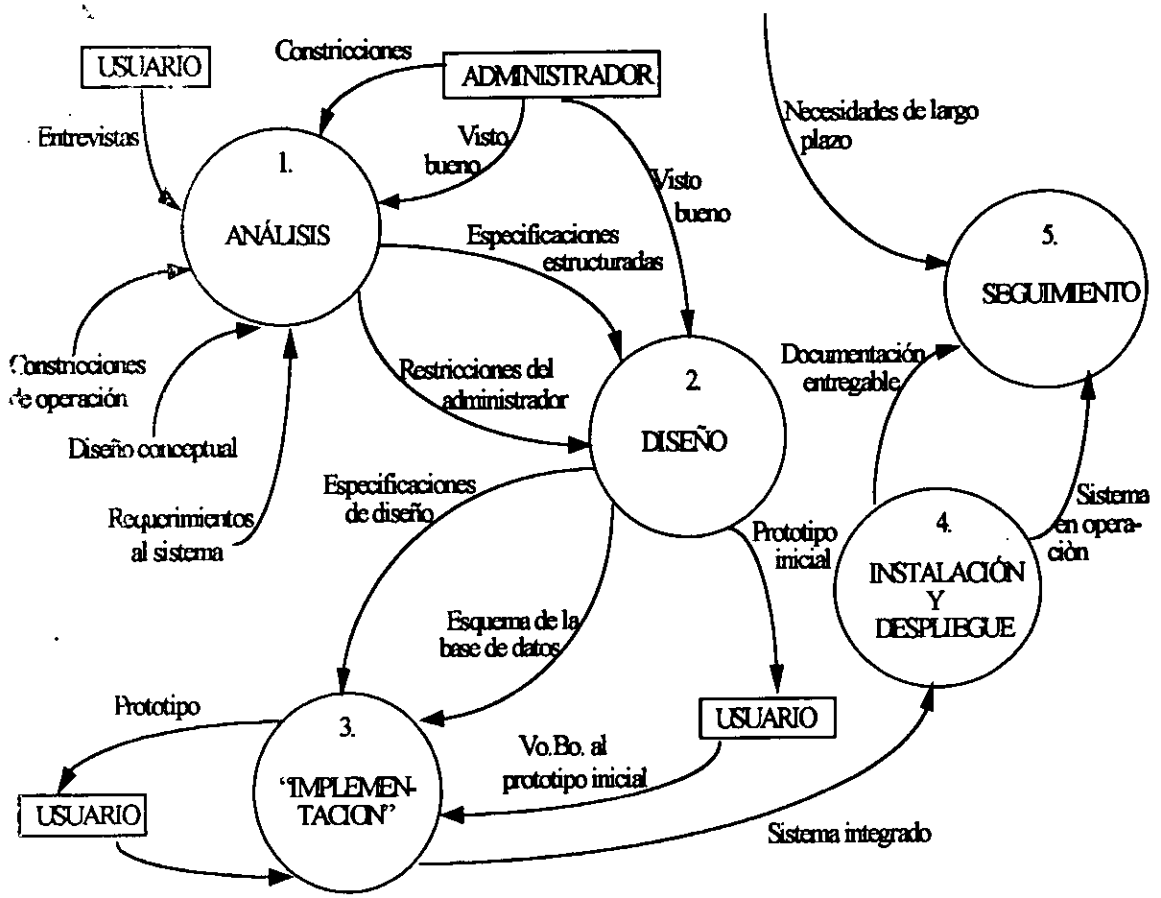


FIGURA 41: AJUSTE DE LA VARIANTE DE LA METODOLOGÍA DE YOURDON APLICADO AL GAR

Si el lector compara este DFD con el de la página 58 (figura 16), observará que hay dos 'burbujas' que aquí han desaparecido y son las que corresponden a conversión de la 'base de datos' y a pruebas de aceptación y control de calidad.

Lo anterior por que hasta donde yo conozco, no hay una fuente de datos estructurada que a la fecha permita realizar la primer 'actividad' mencionada -salvo quizá en la Facultad de Medicina-, en todo caso tal vez existan 'archivos' aislados de diversos profesores, generados en 'procesadores de texto' o alguna herramienta equivalente -para tal efecto la aplicación permite la copia de textos desde ese tipo de fuentes-. Asimismo y con relación a la segunda 'burbuja' referida, no se de un comité u órgano equivalente facultado en la UNAM para sancionar u otorgar vistos buenos a *software* desarrollado para la institución ... ¡pero cómo

hace falta!. Se de un intento por instrumentar para un proyecto, un cuerpo colegiado al efecto, pero la verdad es que no vale la pena mencionar por que falló y como intento usarse.

Debo comentar que el DFD expuesto es resultado de la aplicación de la variante en cuestión al GAR y que según el proyecto de que se trate, resultarán diferentes y diversos DFD ajustados al caso de que se trate, en otras palabras, "según el sapo es la pedrada".

También debo comentar que en el desarrollo de este pequeño proyecto, salvo el rol de usuario y comité de informática, todos los otros posibles mencionados explícitamente como el del administrador o el de líder de proyecto y aquéllos otros no mencionados explícitamente, tales como analista, diseñador, programador, secretario, dibujante, "chalan", etc., los desempeñó su servidor.

Al observar el DFD de referencia, el lector notará la existencia de la 'burbuja' 5 que implica los usuarios consideran a la aplicación lo suficientemente funcional como para usarse y "entrar en producción" -aunque para mi no deja de ser un 'prototipo' que puede perfeccionarse-; pues bien, a pesar de estar "diagramada" la 'actividad' de seguimiento, al respecto comento que el mantenimiento no pienso hacerlo yo; en su caso haría entrega de los 'programas fuente', previo *copy rigth* a mi Alma mater, en espera de que alguien o algún grupo de trabajo, preferentemente de mi Facultad, lo retomase en beneficio de la institución. ¿Qué tal suena la idea de completarlo, perfeccionarlo y pasarlo a C/S sobre Internet -lo cual técnicamente es posible y "aparentemente" no muy complicado-?. De hecho hay un libro de "Delphi para Internet" o "Delphi con Internet" o algo así, pero cuando lo busqué ya se había agotado. Quiero suponer que además de requerir conocimientos básicos de HTML, en la versión 3 de dicho producto deben venir incluidos 'objetos' que facilitan el funcionamiento de los 'programas' sobre la "supercarretera" -de aquí lo de "aparentemente"-.

Sobre el tema del "ajuste" y uso de la variante, así como para explicar como realicé las 'actividades' y 'tareas' correspondientes a las 'burbujas' 1 a 4 del DFD de la figura 41, los siguientes puntos:

7.1 ANÁLISIS

Mediante este DFD me permito presentar el ajuste de la 'actividad' correspondiente para el GAR:

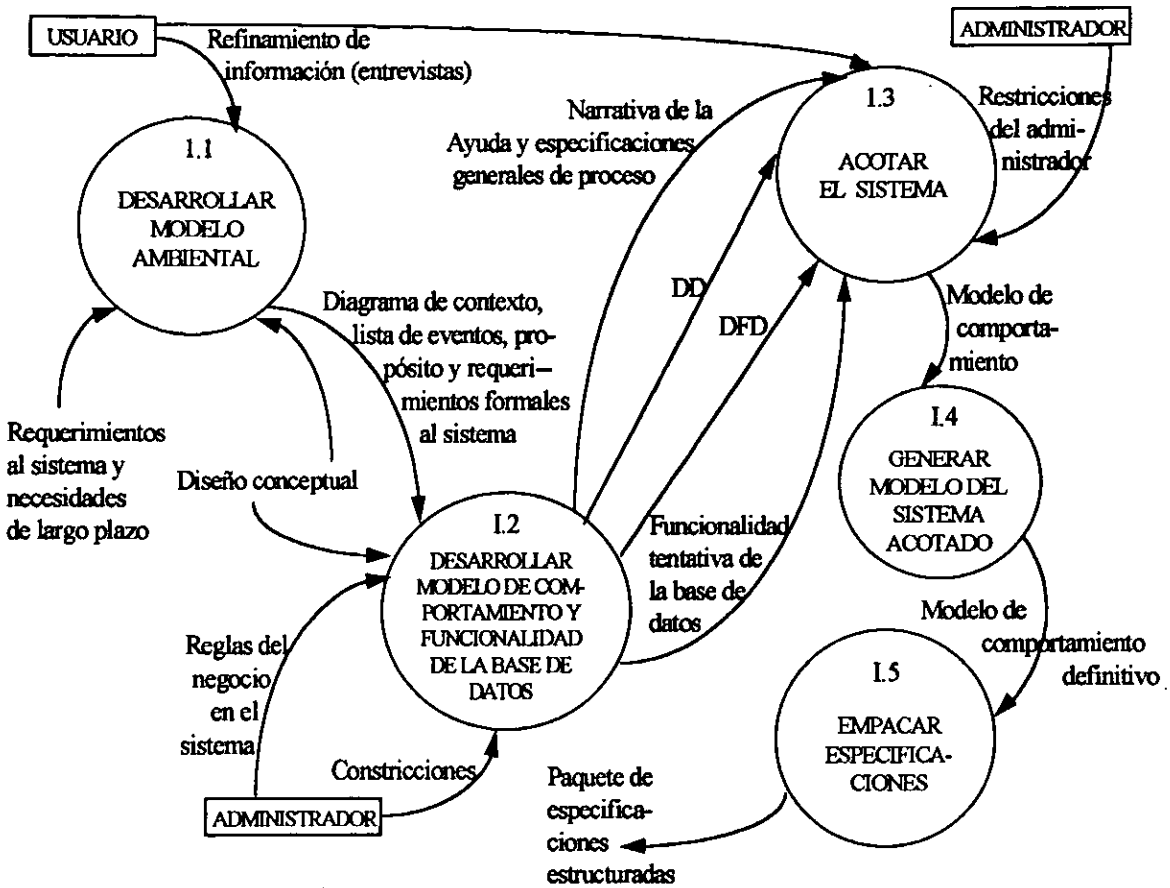


FIGURA 42: AJUSTE DE LA VARIANTE PARA EL DFD DE ANÁLISIS

La cuestión de realizar un análisis en forma según la variante (ver figura 17, página 63), a través de un cuestionario de inspección -entre otros elementos ahí mencionados- para generar escenarios de aceptación, una carta de proyecto y las presentaciones a directivos del caso, no era posible para mi en función de la "no tan alta" disponibilidad de mi tiempo, además de que, curiosamente y esto es de llamar la atención, no conozco -"ni de oidas"- si existe alguna instancia o autoridad a nivel Escuela o Facultad y mucho menos a nivel UNAM, con la atribución para definir, coordinar y evaluar qué criterios y/o elementos hay que considerar para elaborar reactivos a fin de evaluar el aprovechamiento de los alumnos,

i.e., a la fecha cada quién evalua como mejor puede; en todo caso ¿a quién debería haberme dirigido?. Como consecuencia de lo anterior, mucho menos existen -hasta donde sé-, políticas en la materia (¿percibe el lector las repercusiones de este simple hecho a nivel institucional y nacional?). Por lo mismo y sólo para clarificar ideas, enseguida planteó algunas de las 'reglas del negocio', que elaboré y consideré debían ser incluidas en la funcionalidad del GAR (aquí puede observarse cómo algunas de elllas pueden ser consideradas, de entrada, como requerimientos al sistema):

* El acceso a los comandos de la aplicación de cómputo, será a través de un menú y cada usuario deberá tener definido a que opciones puede acceder.
* El usuario de administración tendrá, de inicio, 'permiso de acceso' a todas las opciones del menú.
* Ningún reactivo será identificable por su número de registro en el Banco.
* Al registrar enunciados de reactivos cuya respuesta es seleccionada por el alumno, no se debe conocer el código de la respuesta correcta, por lo que la GUI funcionará dependiendo del tipo de reactivo de que se trate.
* Etc.

Para definir un modelo ambiental y con relación al planteamiento del propósito formal de la aplicación de cómputo, inicie contemplando la frase ya citada -del Lic. Daniel Zarabozo- relativa a la necesidad de contar con *"un sistema de cómputo que permitiese generar exámenes de manera automatizada y mantener actualizada la información que los sustenta"*. Con base en ella y despues de varios intentos llegue a concretar los objetivos genéricos básicos del GAR, para efectos del análisis, en términos de:

I) Generar en forma automática reactivos de la asignatura que se especifique, sean para exámenes o tareas, almacenándolos en un medio magnético al efecto, posibilitando su impresión, previa elección de los temas o, de los temas y subtemas, para que la aplicación de cómputo seleccione en forma aleatoria o pseudoaleatoria los reactivos correspondientes al banco de que se trate, acorde al concepto

pedagógico relativo a la evaluación sumaria (comparar con los textos respectivos de las páginas 143 y 144).

II) Permitir la actualización de los bancos contemplando la funcionalidad para su creación y/o actualización, así como para el registro y/o actualización de los propios reactivos, acorde a la tiplogía de reactivos establecida en el marco conceptual del GAR (comparar con el texto respectivo de la página 134, que integrado a aspectos de seguridad, define el objetivo del módulo manejador de bancos de reactivos -elaborado en la 'actividad' de diseño-).

Con base en lo anterior, fue posible definir un diagrama de contexto -presentado en 7.1.- y una lista inicial de eventos, que fué puliendose en forma similar a la descrita para los objetivos del GAR.

Por otro lado, para obtener información sobre la elaboración de reactivos y poder precisar requerimientos al sistema y necesidades de largo plazo -sin perder de vista la ausencia de políticas en la materia-, me base en múltiples pláticas y entrevistas con muy diversos profesores de la institución (*i.e.* empleé la misma mecánica con la que obtuve el diagnóstico -punto 2 de esta tesis-), sobretodo con aquélla gente muy dedicada, a la que no le queda de otra que investigar en bibliografia diversa o elaborar sus propios reactivos para fines de evaluación del alumnado, además de hacer la investigación mencionada en 6.2 (Marco conceptual), tratando de seguir lo que recomiendan Yourdon y Coad al respecto para análisis orientado a 'objetos':

"Investigate the problem domain, observe first-hand; listen actively; check previous OOA results; check other systems; read. read, read; and prototype." (sic).

En otras palabras, "le di la vuelta" al asunto del acopio de información para el análisis, armando un marco y un diseño conceptual para el GAR con elementos como los antes mencionados.

Otro paréntesis fuera de lugar (¡perdón por el desvío!): Con el marco y diseño conceptual mencionados estimo -salvo mejor opinión-, es posible incluso aplicar una metodología totalmente orientada a 'objetos' para obtener el producto "final"; empero, confieso mi inexperiencia en el diseño y la 'implementación' correspondiente vía OOP y francamente, ya no quise invertir más tiempo en este asunto; tal vez para la tesis de matemáticas -algún día-. Cierro paréntesis.

Aí pues, como mencioné en la página 114, de esta forma consideré *"en forma implícita y desde un inicio los requerimientos básicos para el almacenamiento de los reactivos en la base de datos"* y como está escrito en la página 123: *"Todos y cada uno de los tipos de reactivos aquí establecidos, considerados como requerimientos dan origen a parte de la funcionalidad del GAR"*.

De este modo, a la información conceptual anterior lo único que le hice -¿cuántas veces he hecho esto?-, fue efectuarle una traducción técnica para convertirla en el conjunto de **'requerimientos funcionales'** impuestos a la de la aplicación de cómputo y, a partir de la construcción de la lista de eventos (ver 7.1), derivada del refinamiento de la lista de eventos inicial, fue posible definir un conjunto de 'especificaciones estructuradas' iniciales y otro de especificaciones generales de 'proceso' -más refinadas que las primeras-, para el modelo de comportamiento de la aplicación. Para la gente que no tenga esta experiencia de "traductor" y colabore de alguna forma en una organización amplia, le recomiendo "de todas, todas" seguir los pasos del análisis de la variante planteada, si le es posible y "si le acomoda"; para hacer mayor énfasis en esta 'actividad', debo abundar más sobre lo qué es la conversión de una lista (estructurada) de eventos y requerimientos, en 'requerimientos funcionales' a través de una traducción técnica -tratando de explicar que también lo qué entiendo por esto último-. Al efecto me permito intentar clarificar el asunto con el siguiente ejemplo:

Considérese la regla del negocio relativa al planteamiento: *".... la GUI funcionará dependiendo del tipo de reactivo de que se trate"* y contémplese también el evento: *"El usuario capta reactivos de tipo Falso/Verdadero"*.

Según la tipología de reactivos del GAR (figura 28, página 115), se considera al tipo de reactivo Falso/Verdadero con la característica de que su respuesta es seleccionada por el alumno del conjunto de opciones respectivas y según ejemplos del caso (ver página 119), es pertinente considerar como un requerimiento inicial llano, que a través del GAR, se puedan captar para este tipo de reactivos:

* el texto correspondiente al enunciado y
* la opción de la respuesta correcta

Analizando lo anterior, es plausible establecer como una 'especificación estructurada' inicial (que bien puede incluirse, con otra redacción al efecto, como parte de la redacción de la narrativa inicial de la 'ayuda'):

"... posterior a la definición del tema, subtema y/o objetivo, así como del tipo de reactivo, procede para el caso de reactivos de tipo Falso/Verdadero:

** Preparar la pantalla para la captación del texto correspondiente al texto del enunciado y, al terminar el registro respectivo,*

** Preparar la pantalla para la captación de la opción de la respuesta correcta, sindo posible grabar solamente una de las dos siguientes opciones: Falso o Verdadero.*

** Terminar el registro del reactivo."*

Como un 'requerimiento funcional' (ya con mayor grado de precisión y de alguna manera, con un contexto), puede considerarse a:

"La 'ventana' para captar reactivos permitirá seleccionar el tema, subtema y/o objetivo, y tipo de reactivo: Cuando el tipo de reactivo es:

** De respuesta abierta: • •*

** De respuesta breve: • •*

•

•

** Falso/Verdadero:*

* *Activar en la 'ventana' un 'componente' (del 'desarrollador de aplicaciones') para la captación del texto correspondiente al texto del enunciado y, al terminar su registro,*

* *Activar en la 'ventana' un 'componente' para la captación de la opción de la respuesta correcta, sindo posible registrar solamente uno de los dos siguientes caracteres: 'F' para Falso o 'V' para verdadero, terminando con esto el registro del reactivo.*

* • • "

Con lo anterior y según recomienda Yourdon: "a plasmar ideas en 'burbujas'" o bien: "¡a hacer bolas! -pero sin hacese *idem*-"; de esta forma, se obtiene a algo como[*]:

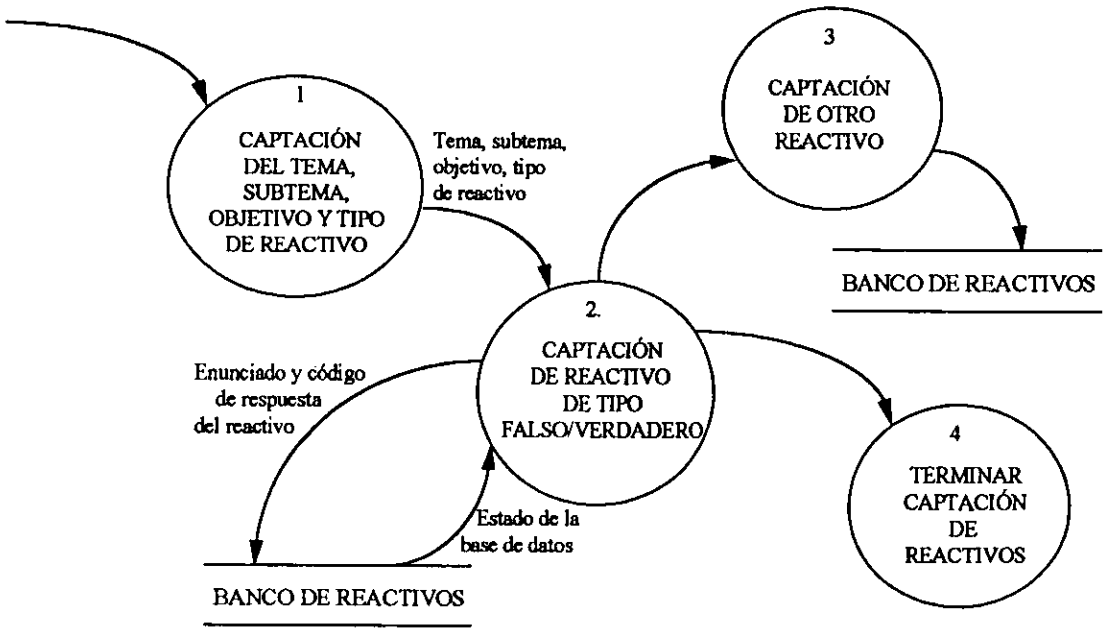


FIGURA 43: EJEMPLO DE CONTEXTO DE "EL USUARIO CAPTA REACTIVO DE TIPO FALSO/VERDADERO"

[*] En estos menesteres, es una sana costumbre elaborar para cada evento identificado, el DFD de su contexto y "explotar" ese diagrama con otros, cuidando el mayor "nivelamiento" posible entre ellos.

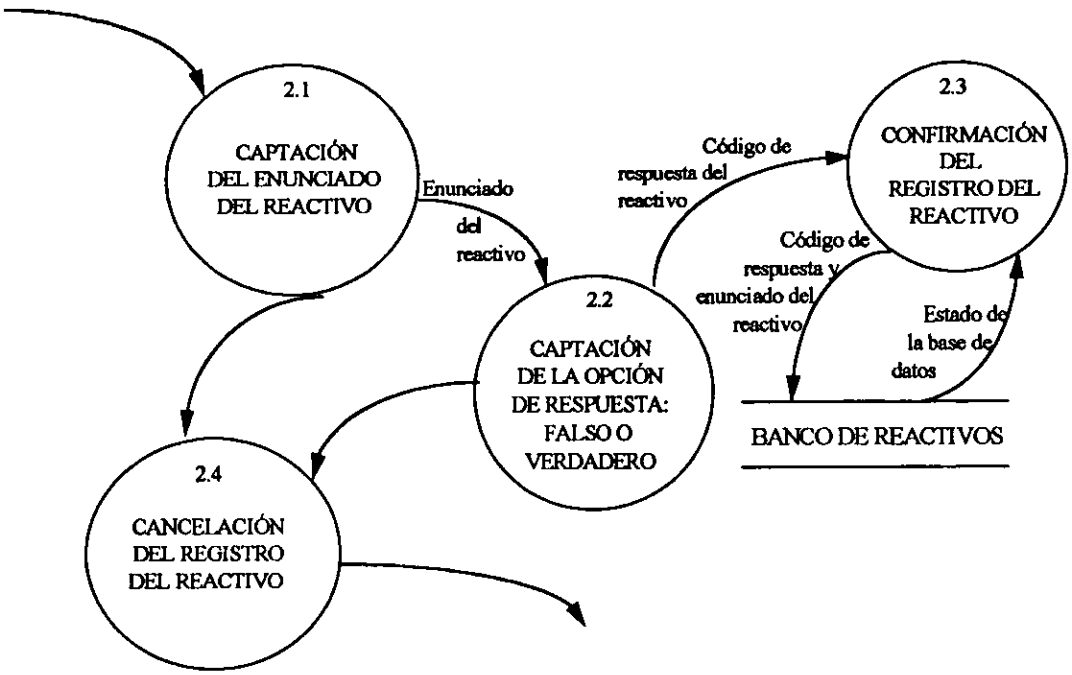


FIGURA 44: EJEMPLO DE EXPLOSIÓN DE "CAPTACIÓN DE REACTIVO DE TIPO FALSO/VERDADERO"

Retomando el DFD del "ajuste" del análisis y acerca de a la narrativa o descripción formal estructurada para el uso del GAR, el lector observará que una parte de ella está contenida "ya en bonito" y arreglada para su integración en este documento en 6.4. A partir de la redacción inicial sobre el asunto y después de irla puliendo, construí los textos para la 'ayuda' de la aplicación. Cabe mencionar que la narrativa de la 'ayuda' de la GUI muy amenudo clarifica y "GUÍa" los trabajos y que, verdaderamente, me resultó un insumo sustancioso para el diseño e 'implementación' del GAR.

Acerca de la funcionalidad tentativa de la base de datos, y retomando el ejemplo del reactivo de tipo Falso/Verdadero, resulta conveniente comentar que después de "echarle un vistazo" al 'requerimiento funcional' y al DFD correspondiente, es posible anticipar mínimamente que

la 'base de datos' debe contemplar para la 'entidad' correspondiente a los reactivos, 'atributos' que correspondan (adicionalmente a tema, subtema, objetivo) a:

* tipo de reactivo
* texto correspondiente al enunciado y
* código de respuesta correcta ['F' | 'V']

Con relación a la definición de las constricciones para acotar el sistema y adelantándome un poquito a los comentarios sobre el diseño en materia de restricciones, debo comentar que estimo conocer lo suficiente a la UNAM en la materia, pues me tocó en muchas y muy diversas ocasiones, instalar y mantener *software* en bastantes Escuelas, Facultades, Institutos y Direcciones Generales de la misma (como recuerdo las "vueltas y vueltas" a todas y cada una de las FES y de las ENEP -entre otras entidades académicas y otras no tan académicas-). Como consecuencia de lo anterior, evalue la necesidad de acotar el modelo de comportamiento de la aplicación de cómputo, pero la verdad con la infraestructura en la materia que en general cuenta la Institución (*e.g.* equipos PC, comunicaciones), no había nada que el modelo derivado del análisis del GAR no pudiese contemplar (*i.e.* no había nada que eliminar al "modelo en papel").

Sobre el modelo de comportamiento definitivo de la aplicación de cómputo, parte de él, concretamente el correspondiente al módulo de seguridad, se muestra en 7.1,

7.2 Diseño

Acerca de la 'arquitectura' y la asignación de 'tareas' a los diversos elementos de la aplicación de cómputo, concretamente, sobre la distribución de cargas en 'procesadores' el diseño "estuvo muerto", pues la aplicación de principio es *'standalone'* y a pesar de poder funcionar en modo 'multiusuario', realmente ocupa solamente el procesador de la PC y, en este caso, con un 'protocolo' de comunicación adecuado, diversas copias del 'programa' del GAR pueden acceder a un *'drive'* o **'disco virtual'** de un 'servidor' sobre una LAN, compartiendo dicho

recurso y la 'base de datos', pero 'procesando' 'localmente' sus 'tareas' o sea, ejecutandose en cada PC con los recursos propios de cada equipo (entre otras cosas, con su propio 'procesador'). El control de las actualizaciones a la 'base de datos' la efectua otro 'programa' de 'interface' denominado IDAPI -llamado comercialmente *Borland DataBase Engine* (BDE)-, incluso al ser compartida por varios usuarios. Su aplicación se muestra en el diagrama de 'arquitectura' del GAR en el punto 7.1 de esta tesis.

Por otro lado, al considerar los objetivos básicos del GAR -establecidos en el análisis- y con base en la el modelo de comportamiento (*i.e.* en sus 'especificaciones estructuradas') y en la narrativa de de la aplicación (6.4), me resultó muy sencillo definir sus 'módulos' básicos:

* Bancos de reactivos
* Generador
* Seguridad
* Catálogos
* Estadísticas

Para mi fortuna tales 'módulos' resultaron no imbricados: si acaso el más "tardadito" en términos de "talacha" -que no complejidad- fue el manejador de los bancos, pues para crearlos y permitir la captura de los reactivos el 'programa' manipula 6 'tablas' y 7 'catálogos'; a su vez, el más "enredadito" a nivel diseño resultó el de seguridad, pues aunque maneja solamente un 'catálogo' y una 'tabla', tiene 4 'ventanas' de alguna forma "contiguas" y que "interfacean" o se interrelacionan a nivel diseño y funcionalmente entre sí. Para los casos anteriores y sobretodo para el generador, lo que me facilitó la 'codificación' fue "cualquier cantidad" de 'rutinas' o ***'procedures'*** que tenía desde la versión del 'programa' para MS-DOS, pues este 'módulo' se comporta muy "batcheramente" y los restantes son a nivel diseño "talacha pura" -¡perdóname Miguel de Cervantes!-.

La construcción del DD y del 'esquema de la base de datos' -para el que realicé tres ejercicios-, los hice con la ayuda de una herramienta CASE y el diseño de la GUI con el

propio 'desarrollador de aplicaciones': estas tareas traté de realizarlas lo más en paralelo posible, es decir, tratando de pensar simultáneamente en ellas, sin perder de vista en ningún momento los requerimientos derivados de la traducción técnica del marco y diseño conceptual a través del análisis, así como la descriptiva de la funcionalidad del GAR, logrando de esta forma concretar el 'prototipo inicial' en un lapso corto con relación al tiempo en que empece a diseñar. De hecho el DFD de diseño "ajustado" para esta parte es:

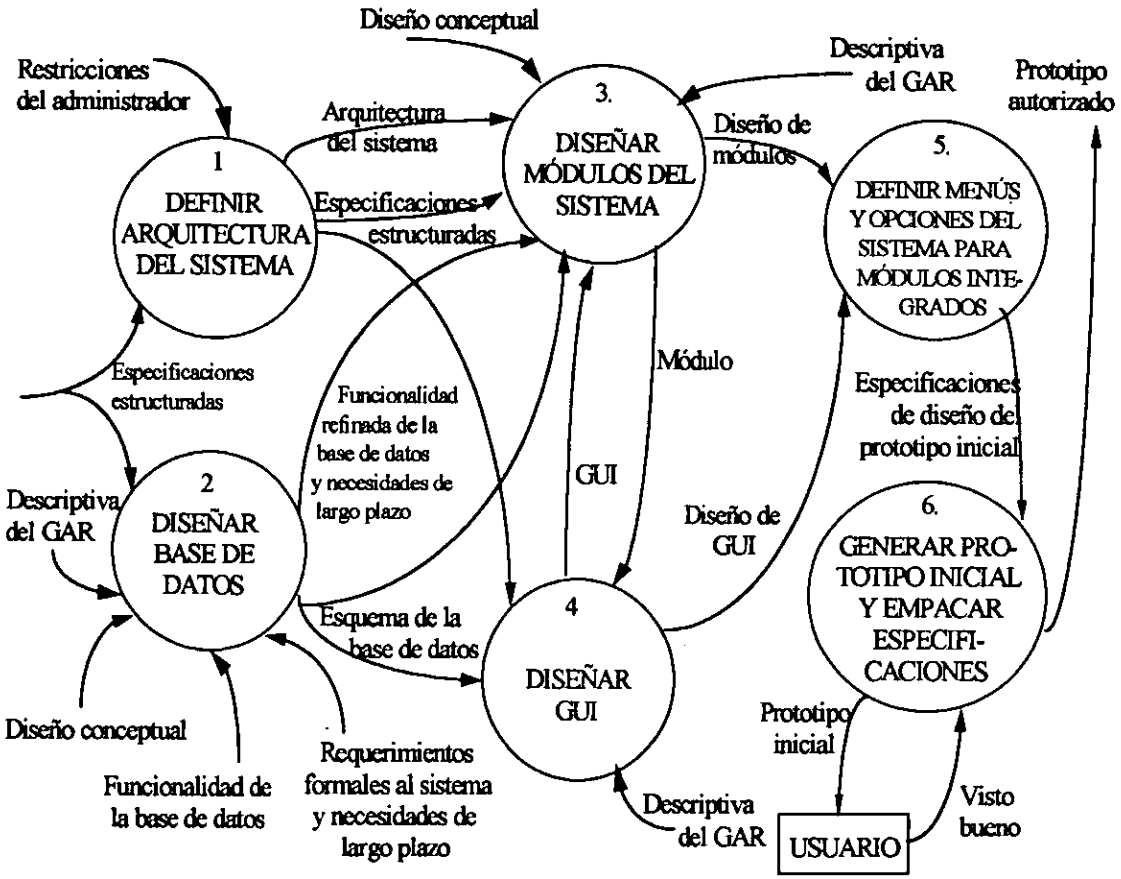


FIGURA 45: AJUSTE DE LA VARIANTE PARA EL DFD DE DISEÑO

Debo reiterar sobre el valor agregado que obtuve derivado del análisis sobre la investigación realizada (*i.e.* el marco y el diseño conceptual de este documento), pues con base en él fue simple definir el 'menú' principal y sus opciones, las cuales prácticamente engloban a los 'módulos' en cuestión, mismos que estimo, pueden crecer 'incrementalmente': ejemplos de lo anterior se tienen al considerar el 'esquema de la base de datos' -hacia el final de este

documento-, que contiene la propuesta para construir un GAR más completo pues, salvo mejor opinión: bastaría añadir las 'ventanas' correspondientes a los 'catálogos' adicionales ahí expuestos, con lo que "crecería" el 'módulo' correspondiente, retomando en su caso parte de la 'codificación' existente.

Por lo que toca a seguridad, prácticamente -estimo- quedaría igual, a reserva de instrumentar su contraparte en un 'servidor' de datos, previa revisión y redefinición -en su caso-, de sus políticas de acceso y contemplando las 'facilidades' que al respecto contemple el *software* mencionado.

Donde realmente podría haber un impacto relevante es al tratar de definir en el GAR otro tipo de reactivos que "no soporte" el esquema actual; por ello traté de 'implementar' los que detecté se llegan a usar más frecuentemente.

Los ejemplos que cito pretenden exhibir la idea de lo que significa 'código reutilizable' y de lo eficiente y tal vez, barato, que puede resultar su uso para efectos de mantenimiento de cualquier sistema, destacando su aplicación en organizaciones amplias cuando así se plantea desde un inicio y que, en el peor de los casos, permite por lo menos tener sobre los costos involucrados, una estimación más precisa y mínimamente un control en la materia, pues si el precio estimado "en billetes" para desarrollar cualquier *software* es alto, el que realmente se paga finalmente, es más alto y, mantenerlo, por regla general, es "lumbre". Por cierto: una "ley" genérica en la materia, práctica y muy pragmática que le aprendí a un "gringo" con experiencia bastante en estos negocios, que se basa en tres adjetivos o conceptos -según se les aplique-: rápido, barato y calidad. En México como en otros lugares, casi siempre se puede optar, exclusivamente, por dos de tres.

Regresando al asunto con relación al DFD anterior y aunque, como dice un amigo mío: *"aquí en México no vale el know how sino el know who"*, para quién le aproveche aquí va un poco de *expertise*, para lo cual voy a explotar ese diagrama un poco más en la página siguiente -aunque sea en forma un poco desnivelada-:

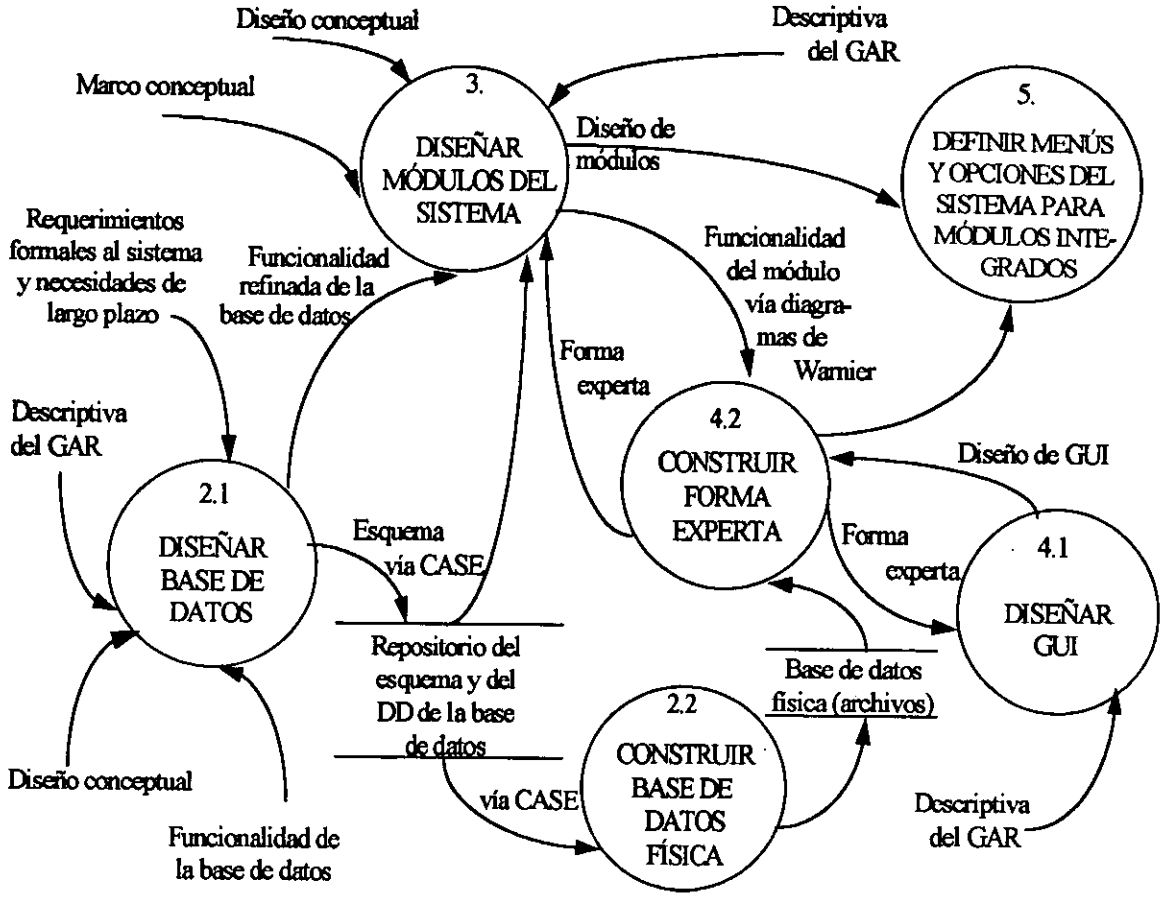


FIGURA 46: EXPLOSIÓN DEL DFD DE DISEÑO DEL AJUSTE DE LA VARIANTE PARA EL GAR

Aquí vale la pena volver a destacar lo valioso que resulta aplicar herramientas CASE al desarrollo de proyectos de sistemas y comento particularmente sobre la que empleé para manejar el diseño de la 'base de datos', el gran ahorro en tiempo que me brindó para las 'actividades' 2.1 y 2.2; con esto no quiero dejar la idea de que herramientas como éstas o los 'desarrolladores de aplicaciones' sean la panacea en el área de sistemas: estoy convencido de que aúnque se tengan las mejores, si no se saben usar y se emplean inadecuadamente resultan en estorbos a la larga.... he visto un caso excepcional en que su mal uso resultó en la siguiente analogía: pretender desatornillar con un martillo, clavar con un taladro y atornillar con una pinza, con lo que usualmente y "llendo para atrás", general y lamentablemente en casos como ese, las áreas de sistemas involucradas vuelven a trabajar como se hacían las cosas hasta antes de pretender 'implementar' CASE o cualquier otra nueva tecnología similar.

¿Se acuerdan del inicio de la novela de Arthur C. Clark adaptada después para la película *"Odisea espacial 2001"*?, dónde el prototipo del ser humano a partir del uso de un hueso como arma y fundamentalmente en razón de su "protointeligencia" -por mínima que fuese-, logra predominar -quién sabe si para bien o para mal-, sobre las demás especies del planeta -¿cuántas se extinguen cada año por causa del ser humano?-...., lo mismo creo de las herramientas de sistemas: por más simples o "viejas" que sean, si se saben aprovechar redundan en muy altos beneficios de productividad.

Regresando a los comentarios del DFD "semiexplotado", para realizar las tareas 4.1 y 4.2 el 'desarrollador de aplicaciones' me resultó muy eficaz y, de hecho, con su utilería denominada ***'Database Form Expert'***, es posible definir y armar 'prototipos' de 'ventanas' que funcionan en un minuto a lo más, sin perder de vista que lo que realmente esta atrás de esta ganancia de tiempo es aquél invertido, entre otros rubros, en el análisis y en el diseño de la 'base de datos', lo cual definitivamente lleva más, pero mucho más tiempo y trabajo de lo que se puede hacer en un minuto.

Para concluir los comentarios de esta parte y referente a las tareas 3 y 4.2, por lo menos para el programador que usa el 'desarrollador de aplicaciones' ya mencionado, los diagramas de Warnier -¡sin inicio, mitad ni fin!-, resultan una herramienta de la mayor utilidad a fines de concretar la 'codificación' y poder "darle la vida" rápidamente al 'prototipo'; esto lo digo porque los he usado no sólo conmigo mismo en este proyecto -en mi rol de "todólogo"-, sino recientemente en grupos de trabajo donde, por cierto, no hay estrellas del cómputo: su uso permite anticipar en general problemas en el diseño, clarificar efectivamente lo que se pretende sea 'codificado' y generar documentación entregable de los proyectos. Una muestra de los elaborados para el módulo de seguridad del presente trabajo se encuentra en 7.1.

Debo aclarar que el rol de "todólogo" que me asigné en el presente trabajo no pretende nada más que mostrar la viabilidad de la aplicación de la variante y generar el 'prototipo' de una herramienta de cómputo que coadyuve al profesorado de mi escuela en su trabajo -además

de que no se me hace honesto que otras personas me ayudaran en mi tesis-; quedaría pendiente la aplicación de la variante de forma integra en organizaciones amplias, pero como en mi contexto y al efecto sólo dispongo de mi fuerza de trabajo y carezco de los medios de producción pertinentes (¿dónde he oido esto?), no me es posible probarla de momento como la he pensado -aunque hé de buscar la forma-. Al respecto debo abundar que tengo clarísimo, por ejemplo, el hecho de que una sola persona no puede elaborar un C/S complejo ni con varios lustros de trabajo; o como dice otro amigo: *"el último todólogo que hubo de a deveras fue Leonardo Da Vinci"*.... y de eso ya hace mucho; por mi parte agregaría que los tiempos en que las edificaciones llevaban incluso siglos ha quedado muy atrás y que ahora en el trabajo de sistemas un detalle de lo que más cuenta es el menor tiempo posible para la construcción de los mismos y que justamente la variante persigue -de suyo-, la disminución de tiempos de desarrollo y su aplicación en grupos de trabajo de organizaciones amplias.

7.3 'IMPLEMENTACIÓN'

Si lo que mata la productividad de un poeta, ensayista o escritor de oficio es la falta de tiempo para escribir, en mi opinión, igual o mucho peor es la falta de tiempo para escribir 'programas' de cómputo: mientras que en los primeros casos, con capacidad, creatividad y algún tipo de respeto a la lengua -¿dónde andas "Gabo"?- es posible producir obras de altísima calidad, nosotros, además de la capacidad y la inspiración, para producir aplicaciones medianamente aceptables, debemos un indispensable e invariable respeto a las restricciones tecnólogicas a que nos enfrentemos y al lenguaje de cómputo en que 'codifiquemos'; por un simple símbolo de ';' fuera de lugar o un *'end'* de más o de menos... no funciona nada. Ni que decir del uso de la lógica que requerimos para la construcción de aplicaciones o sistemas.... compararía, toda proporción guardada, la construcción de los mismos -de sus 'programas' de cómputo-, con la solución de integrales: con conocimientos sólidos, trabajo y un poquito de suerte e inspiración se puede llegar a concretar; sin embargo hay algunos y algunas que no salen..... como tantas otras cosas en la vida.

Por lo anterior, uno debe de buscar aprovechar al máximo su tiempo disponible para escribir y no hay otra forma de hacerlo que volviéndose eficiente, primordialmente 'codificando' lo necesario y no de más. En este sentido, *"diseñar un poco"*, *"'codificar' un poco"*, *"probar un poco"*, se vuelve una regla de oro a seguir para no 'programar', como he dicho, de más. El intercambio con el usuario en este sentido es primordial, pues en cada ronda de 'desarrollo iterativo' es posible detectar lo antes expuesto, sin que quede la menor duda para nadie. Nótese que todo el tiempo estoy suponiendo que para escribir se tiene la creatividad e inspiración suficiente para hacerlo y si no se cuenta con ella, mejor dedicarse a otra cosa.

Con relación al GAR en esta parte de 'implementación', comento que el orden para 'codificar' los 'módulos' fue: bancos de reactivos, generador, seguridad -adicionando respaldos para definir utilerías-, catálogos y estadísticas. Un reflejo de esto se puede observar en los tres ejercicios que hice para el 'esquema de la base de datos' y espero, también "reflejen" de alguna manera que significa el desarrollar una aplicación de forma 'incremental'; el "pegar" las 'ventanas' en un sólo proyecto de cómputo, es decir, en la propia aplicación vía 'desarrollador de aplicaciones' resulta, predefinido el 'esquema de la base de datos', y teniendo a la mano una definición del diseño de cada 'módulo' y de la GUI, en algo casi trivial.

Debo mencionar que aunque una modalidad para la interacción con los usuarios consiste en irles mostrando 'prototipos' de cada 'módulo', como la aplicación aquí expuesta no es compleja, preferí mostrar el 'prototipo' del GAR completo a mi director de tesis -principalmente para darle trámite lo más rápido posible a este asunto-.

7.4 INSTALACIÓN Y DESPLIEGUE

Como antes he expuesto, la aplicación está lista para instalarse "derecha" y sin costos para la institución y como de hecho funciona *'standalone'*, no requiere mayor parafernalia que la de cualquier *software* comercial para equipo PC, caso muy diferente a su despliegue en caso de querer que funcione en modo 'multiusuario'.

7.5 MUESTRA DE DOCUMENTACIÓN TÉCNICA

Como se establece en el marco teórico de este documento y como el lector podrá haber notado, entre otras pretensiones este documento persigue por sí mismo hacer énfasis en el uso de herramientas gráficas tales como las que fueron empleadas en la variante para su aplicación al GAR.

Espero que dichas herramientas hayan facilitado la lectura de la presente tesis y su comprensión.

El lector juzgará sí con la documentación en cuestión y la que se muestra en adelante es posible cubrir los atributos que deben caracterizarla: gráfica -más que transcrita-, modular -en vez de constituir un ente completo prácticamente indivisible-, independiente de su implementación -con lo que es viable lograr un modelo abstracto del sistema o aplicación de cómputo-, *top-down* o 'descendente' -lo que permite presentar una descriptiva del sistema en niveles progresivos de detalle-.

Al efecto me permito presentar en formato horizontal o *landscape*: el DD y los esquemas trabajados para la 'base de datos de la aplicación', así como parte de los DFD y de las especificaciones de diseño del 'módulo' de seguridad, vía diagramas de Warnier[†] en una interpretación libre y, un 'programa' de cómputo del 'módulo' citado correspondiente a la digitación de la clave del usuario, así como la arquitectura del GAR.

Toda la demás documentación está en borradores y huelga presentarla en este documento, ya de por sí un poco extenso.

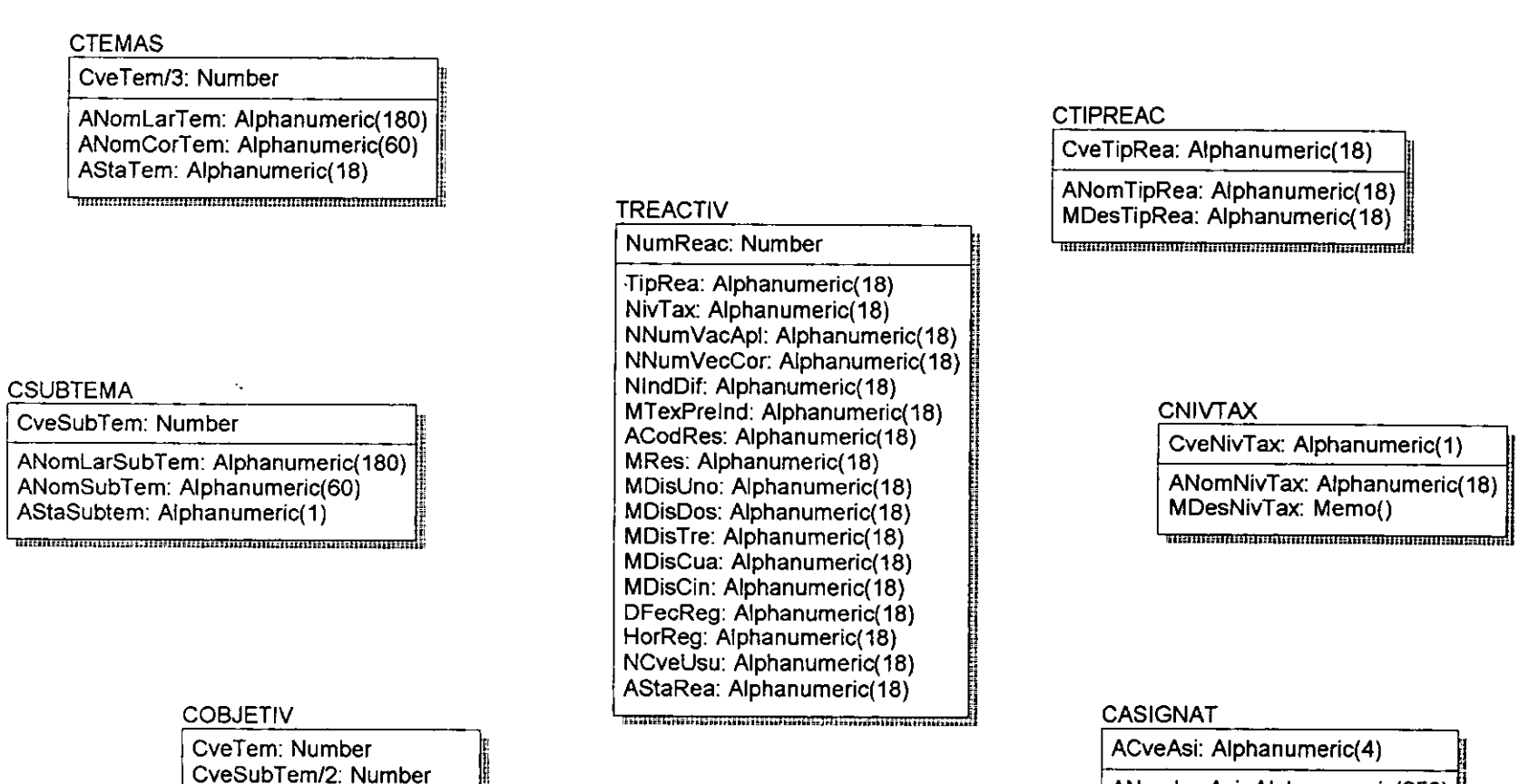
[†] El lector debe considerar que en estos diagramas, 'EVENTO' tiene el significado de una acción que tiene que realizar un 'objeto' o 'componente' del 'desarrollador de aplicaciones'; por ejemplo: en el 'evento' *OnClick* del 'componente' *TButton* se puede simplemente exhibir el mensaje: "Aquí se presenta este mensaje". Por otra parte, en los DFD, EVENTO corresponde a un estímulo del medio ambiente al que debe responder el GAR.

TABLA	ATRIBUTO	DESCRIPCIÓN	LON	RANGO/OBSERVACIONES
CUSUARIO	*CveUsu	Clave del Usuario del GAR		Tipo autoincremento (asignación automática)
CUSUARIO	NPas	Password	4	[1..9999]
CUSUARIO	LBanNue	Permiso de acceso a Banco de Reactivos|Nuevo		Tipo lógico (Falso, Verdadero)
CUSUARIO	LBanAbr	Permiso de acceso a Banco de Reactivos|Abrir		Tipo lógico (Falso, Verdadero)
CUSUARIO	LGenRea	Permiso de acceso a Generador de Reactivos		Tipo lógico (Falso, Verdadero)
CUSUARIO	LCat	Permiso de acceso a Catálogos		Tipo lógico (Falso, Verdadero)
CUSUARIO	LEst	Permiso de acceso a Estadísticas		Tipo lógico (Falso, Verdadero)
CUSUARIO	LUti	Permiso de acceso a Utilerías		Tipo lógico (Falso, Verdadero)
CUSUARIO	AApePat	Apellido Paterno	40	['A'..'Z'] + ['a'..'z']
CUSUARIO	AApeMat	Apellido Materno	40	['A'..'Z'] + ['a'..'z']
CUSUARIO	ANom	Nombre(s)	60	['A'..'Z'] + ['a'..'z']
CUSUARIO	AGraAca	Grado Académico	6	['A'..'Z'] + ['a'..'z']
CUSUARIO	ASex	Sexo	1	M':Masculino; 'F':Femenino
CUSUARIO	DFecNac	Fecha de Nacimiento		Dar formato de dd/mm/aaaa
CUSUARIO	ADom	Domicilio	120	['A'..'Z'] + ['a'..'z'] + ['0'..'9'] + '-' + '#'
CUSUARIO	ATel	Teléfono	15	['0'..'9']
CUSUARIO	AFax	Fax	15	['0'..'9']
CUSUARIO	ACorEle	Correo Electrónico	35	['A'..'Z'] + ['a'..'z'] + ['0'..'9'] + '-' + '#'
CUSUARIO	AStaUsu	Status del Usuario	1	0':No Vigente; '1':Vigente
CPERACBA	*NCveUsu	Clave del Usuario del GAR	4	[1..9999]
CPERACBA	*ACveBan	Clave del Banco	4	['0'..'9']
CPERACBA	AStaPerUsuBan	Status del Permiso de acceso del Usuario al Banco	1	0':No Vigente; '1':Vigente
CASIGNAT	*ACveAsi	Clave de la Asignatura (según Cátalogo UNAM)	4	['0'..'9']
CASIGNAT	ANomCorAsi	Nombre Corto de la Asignatura	60	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
CASIGNAT	ANomLarAsi	Nombre Largo de la Asignatura	180	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
CASIGNAT	NCveUsuAsi	Clave del Usuario que actualizó Asignaturas	3	[1..999]
CASIGNAT	DFecReg	Fecha de Registro		Dar formato de dd/mm/aaaa
CASIGNAT	AStaAsi	Status de la Asignatura	1	0':No Vigente; '1':Vigente
CTEMAS	*SCveTem	Clave del Tema	2	[1..15]
CTEMAS	ANomCorTem	Nombre Corto del Tema	60	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
CTEMAS	ANomLarTem	Nombre Largo del Tema	180	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
CTEMAS	NCveUsuTem	Clave del Usuario que actualizó Temas	3	[1..999]

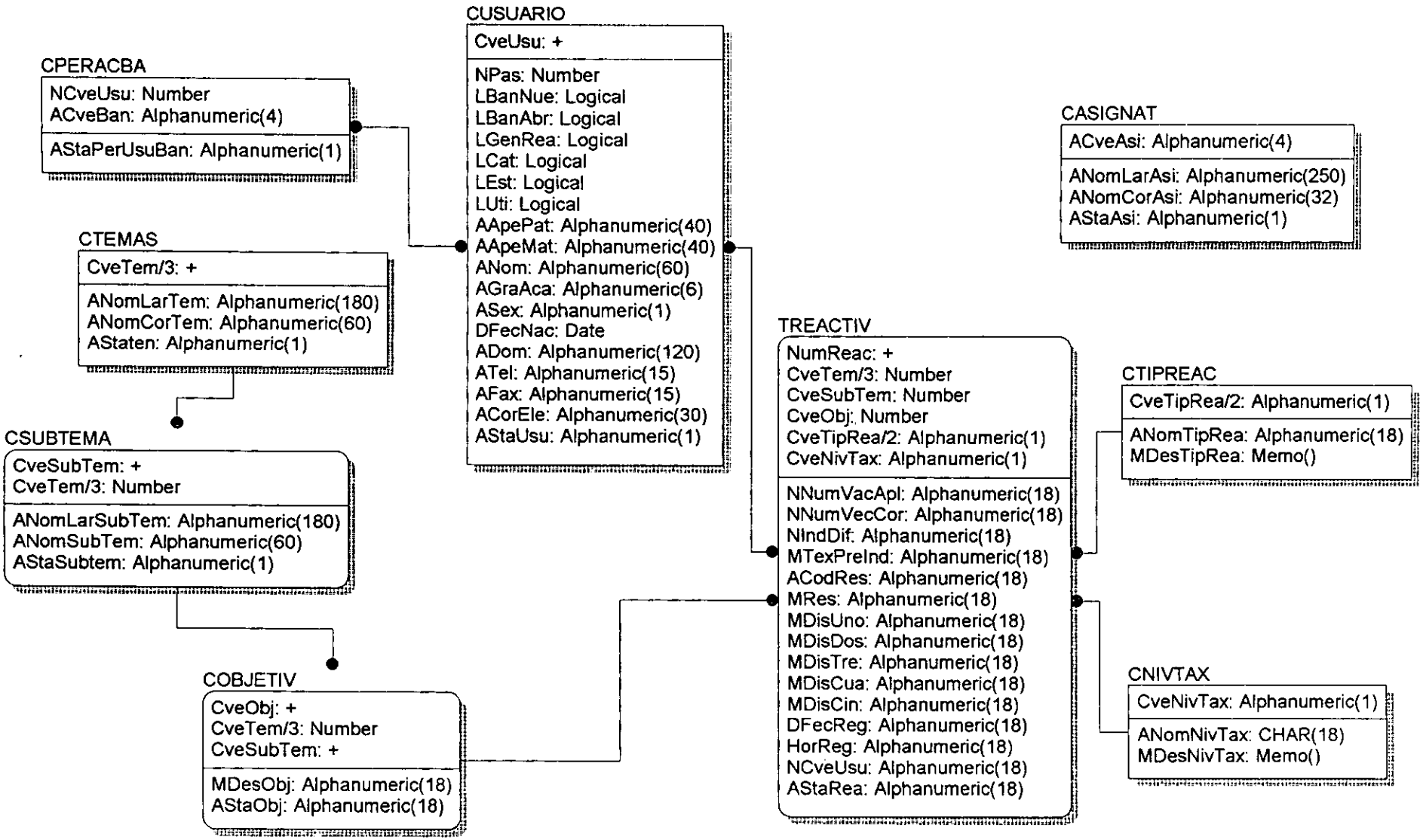
DICCIONARIO DE DATOS DEL GAR

TABLA	ATRIBUTO	DESCRIPCIÓN	LON	RANGO/OBSERVACIONES
CTEMAS	DFecReg	**Fecha de Registro**		Dar formato de dd/mm/aaaa
CTEMAS	AStaTem	**Status del Tema**	1	0':No Vigente; '1':Vigente
CSUBTEMA	*SCveTem	**Clave del Tema**	2	[1..15]
CSUBTEMA	*SCveSubTem	**Clave del SubTema**	2	[1..15]
CSUBTEMA	ANomCorSubTem	**Nombre Corto del SubTema**	60	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
CSUBTEMA	ANomLarSubTem	**Nombre Largo del SubTema**	180	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
CSUBTEMA	NCveUsuSubTem	**Clave del Usuario que actualizó SubTemas**	3	[1..999]
CSUBTEMA	DFecReg	**Fecha de Registro**		Dar formato de dd/mm/aaaa
CSUBTEMA	AStaSubTem	**Status del SubTema**	1	0':No Vigente; '1':Vigente
COBJETIV	*SCveTem	**Clave del Tema**	2	[1..15]
COBJETIV	*SCveSubTem	**Clave del SubTema**	2	[1..15]
COBJETIV	*SCveObj	**Clave del Objetivo**	2	[1..15]
COBJETIV	ADesCorObj	**Descripción Corta del Objetivo**	60	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
COBJETIV	MDesLarObj	**Descripción Larga del Objetivo**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
COBJETIV	NCveUsuObj	**Clave del Usuario que actualizó Objetivos**	3	[1..999]
COBJETIV	DFecReg	**Fecha de Registro**		Dar formato de dd/mm/aaaa
COBJETIV	AStaObj	**Status del SubTema**	1	0':No Vigente; '1':Vigente
CTIPREAC	*ACveTipRea	**Clave del Tipo de Reactivo**	1	['1'..'7']
CTIPREAC	ANomTipRea	**Nombre del Tipo de Reactivo**	15	['A'..'Z'] + ['a'..'z']
CTIPREAC	MDesTipRea	**Descripción del Tipo de Reactivo**	240	['A'..'Z'] + ['a'..'z']
CNIVTAX	*ACveNivTax	**Clave del Nivel Taxonómico**	1	['1'..'7']
CNIVTAX	ANomNivTax	**Nombre del Nivel Taxonómico**	12	['A'..'Z'] + ['a'..'z']
CNIVTAX	MDesNivTax	**Descripción del Nivel Taxonómico**	240	['A'..'Z'] + ['a'..'z']
TREACTIV	*NumRea	**Número de Reactivo**		Tipo autoincremento (asignación automática)
TREACTIV	SCveTem	**Clave del Tema**	2	[1..15]
TREACTIV	SCveSubTem	**Clave del SubTema**	2	[1..15]
TREACTIV	SCveObj	**Clave del Objetivo**	2	[1..15]
TREACTIV	ACveTipRea	**Clave del Tipo de Reactivo**	1	['1'..'7']
TREACTIV	ACveNivTax	**Clave del Nivel Taxonómico**	1	['1'..'6']
TREACTIV	NNumVecApl	**Número de Veces Aplicado**	5	[1..99999]

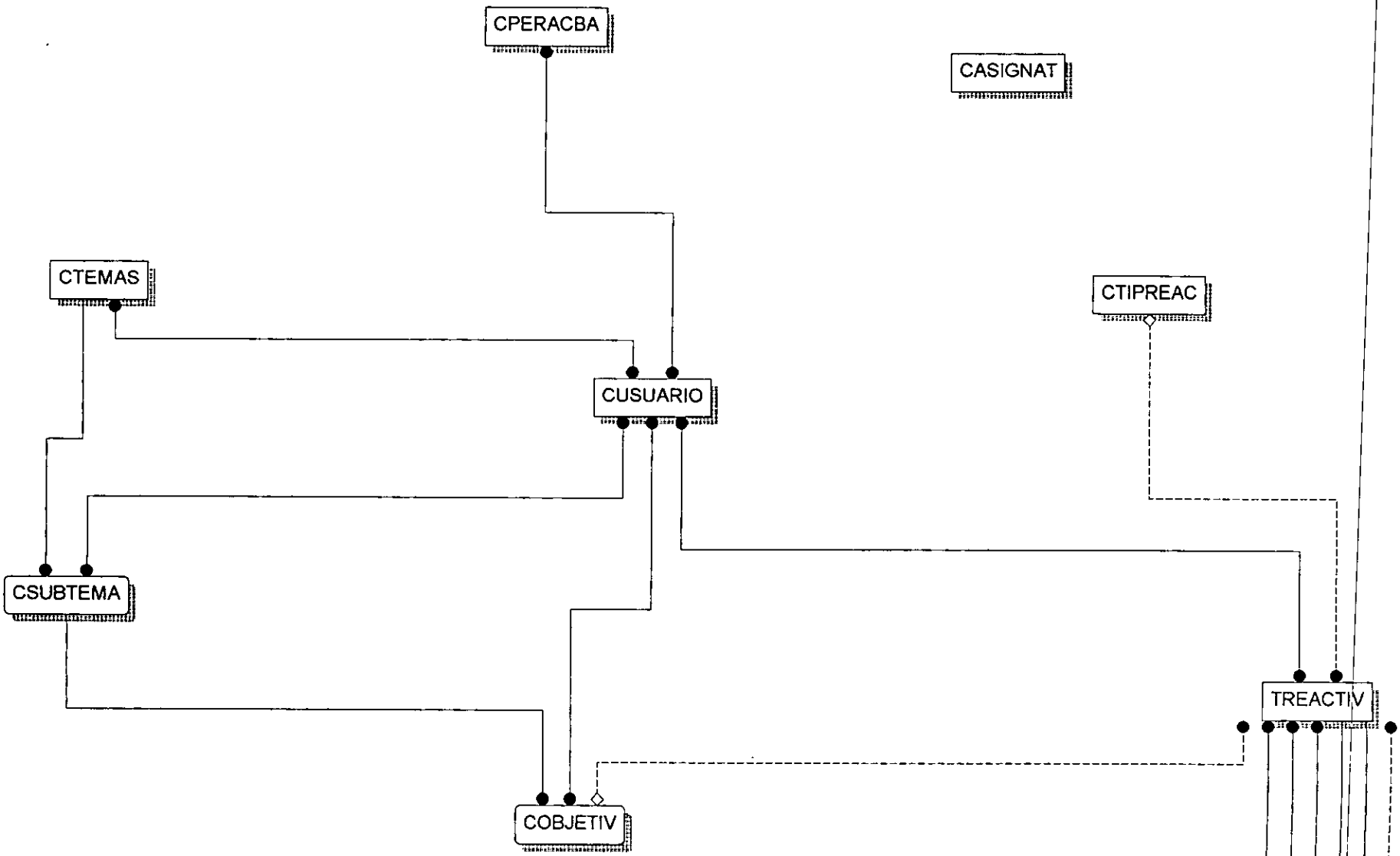
TABLA	ATRIBUTO	DESCRIPCIÓN	LON	RANGO/OBSERVACIONES
TREACTIV	NNumVecResCor	**Núm**ero de **Veces Respondido Correctamente**	5	[1..99999]
TREACTIV	NIndDis	**Índice de Discriminación**		[0..1]
TREACTIV	NGraDif	**Grado de Dificultad**		[0..1]
TREACTIV	MPreInd	Texto de la **Pregunta o Indicación**	240	
TREACTIV	NCveUsuRea	**Clave del Usuario que actualizó Reactivos**	3	[1..999]
TREACTIV	DFecReg	**Fecha de Registro**		Dar formato de dd/mm/aaaa
TREACTIV	THorReg	**Hora de Registro**		Dar formato de hh:mm
TREACTIV	AStaRea	**Status del Reactivo**	1	0':No Vigente; '1':Vigente
TRESPAB	*NResAbiReaNum	**Respuesta Abierta del Reactivo Número**		Valoración del número de reactivo como entero
TRESPAB	MResAbi	Texto de la **Respuesta Abierta**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TIMAGEN	*NImaReaNum	**Imágen del Reactivo Número**		Valoración del número de reactivo como entero
TIMAGEN	GIma	**Imágen asociada al reactivo**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCION	*NOpcReaNum	**Opciones del Reactivo Número**		Valoración del número de reactivo como entero
TOPCION	MOpcUno	Texto de la **Opción de Respuesta Uno**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCION	MOpcDos	Texto de la **Opción de Respuesta Dos**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCION	MOpcTre	Texto de la **Opción de Respuesta Tres**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCION	MOpcCua	Texto de la **Opción de Respuesta Cuatro**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCION	MOpcCin	Texto de la **Opción de Respuesta Cinco**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TCODRESP	*NCodResReaNu	**Código de Respuesta del Reactivo Número**		Valoración del número de reactivo como entero
TCODRESP	ACodRes	**Código de Respuestas**	40	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCASOC	*NOpcAsoReaNu	**Opciones Asociadas al Reactivo Número**		Valoración del número de reactivo como entero
TOPCASOC	MAsoOpcUno	Texto **Asociado a la Opción de Respuesta Uno**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCASOC	MAsoOpcDos	Texto **Asociado a la Opción de Respuesta Dos**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCASOC	MAsoOpcTre	Texto **Asociado a la Opción de Respuesta Tres**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCASOC	MAsoOpcCua	Texto **Asociado a la Opción de Respuesta Cuatro**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']
TOPCASOC	MAsoOpcCin	Texto **Asociado a la Opción de Respuesta Cinco**	240	['A'..'Z'] + ['a'..'z'] + ['0'..'9']



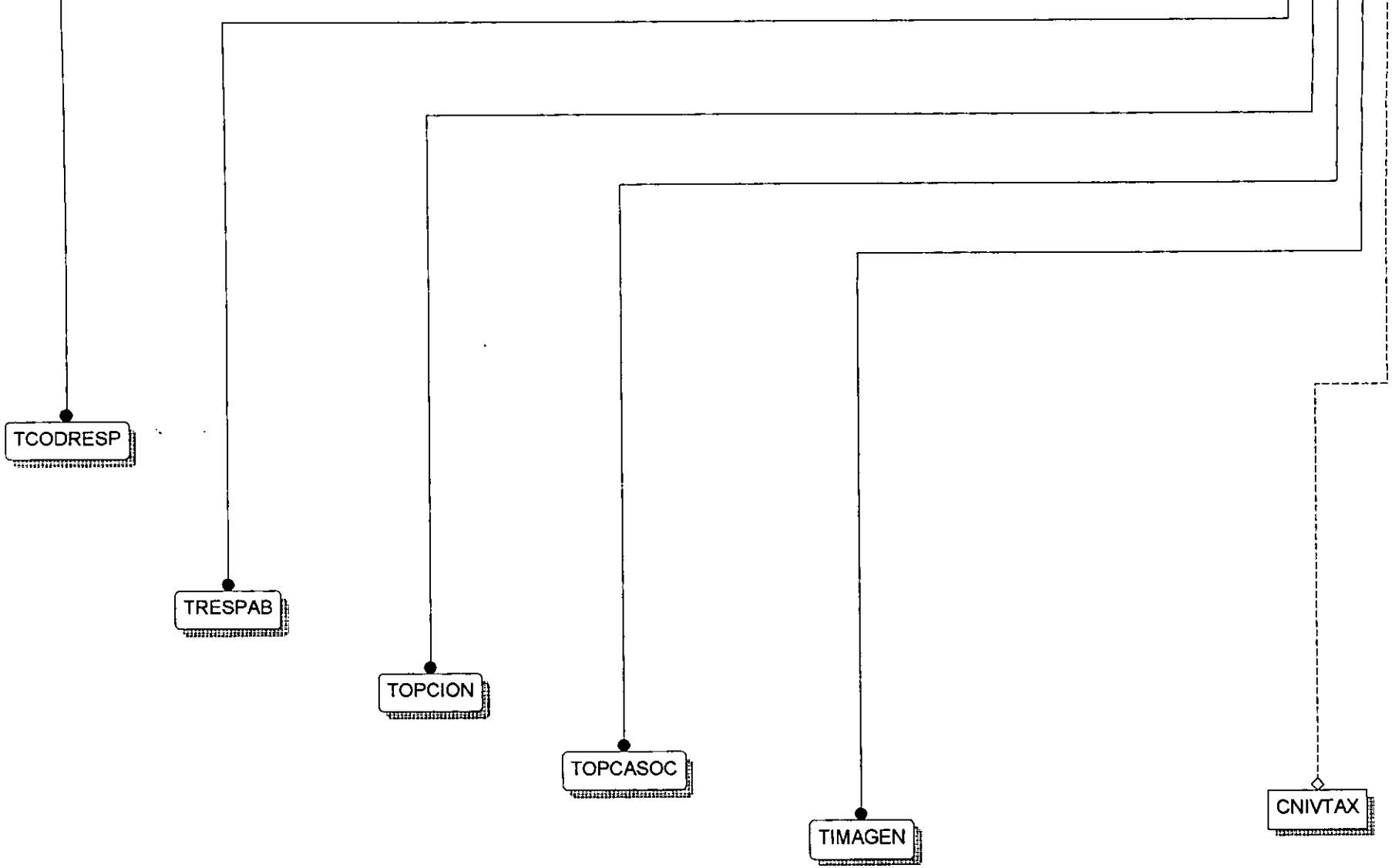
PRIMER ESQUEMA DE LA BASE DE DATOS DEL GAR



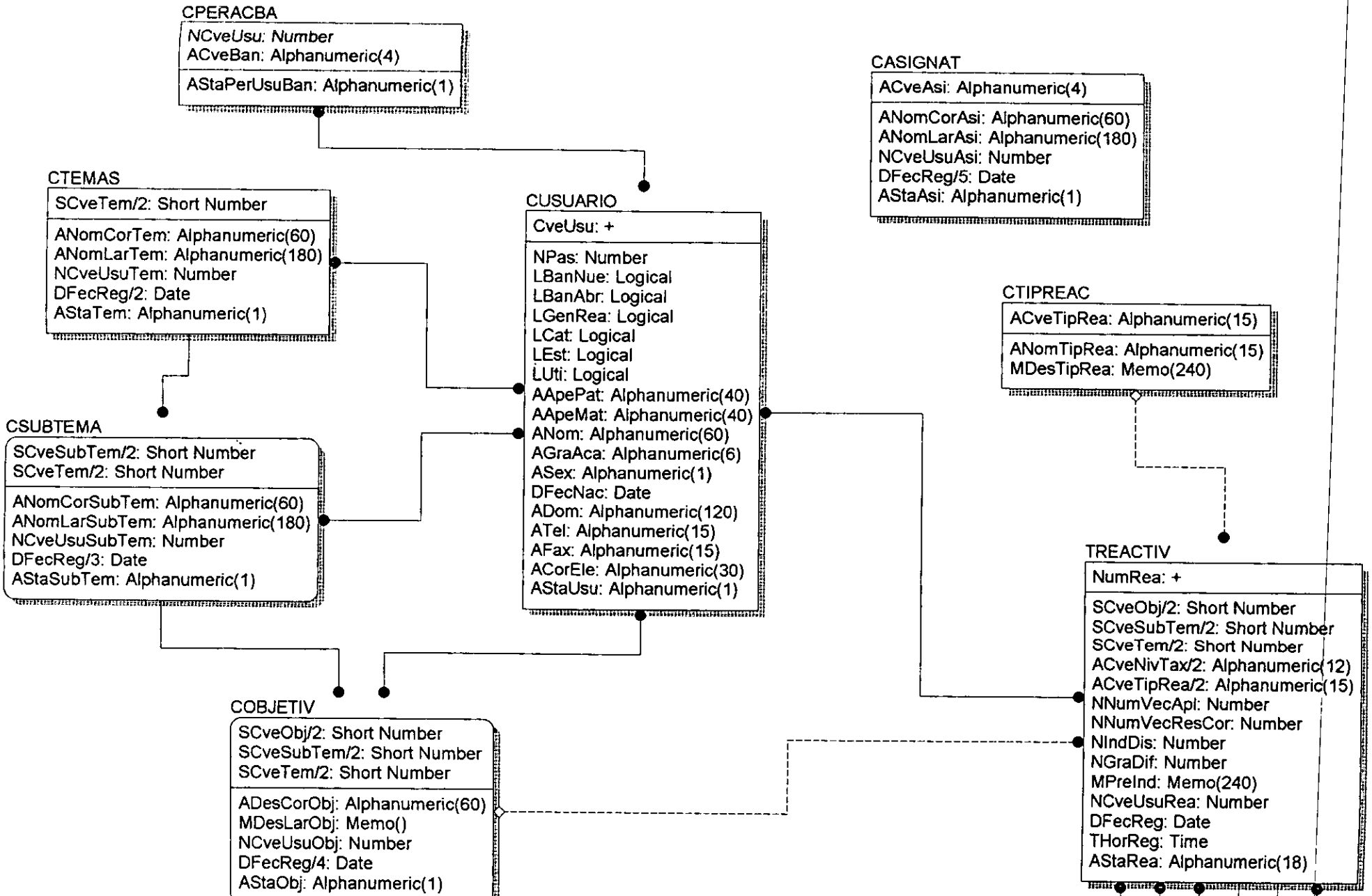
SEGUNDO ESQUEMA DE LA BASE DE DATOS DEL GAR



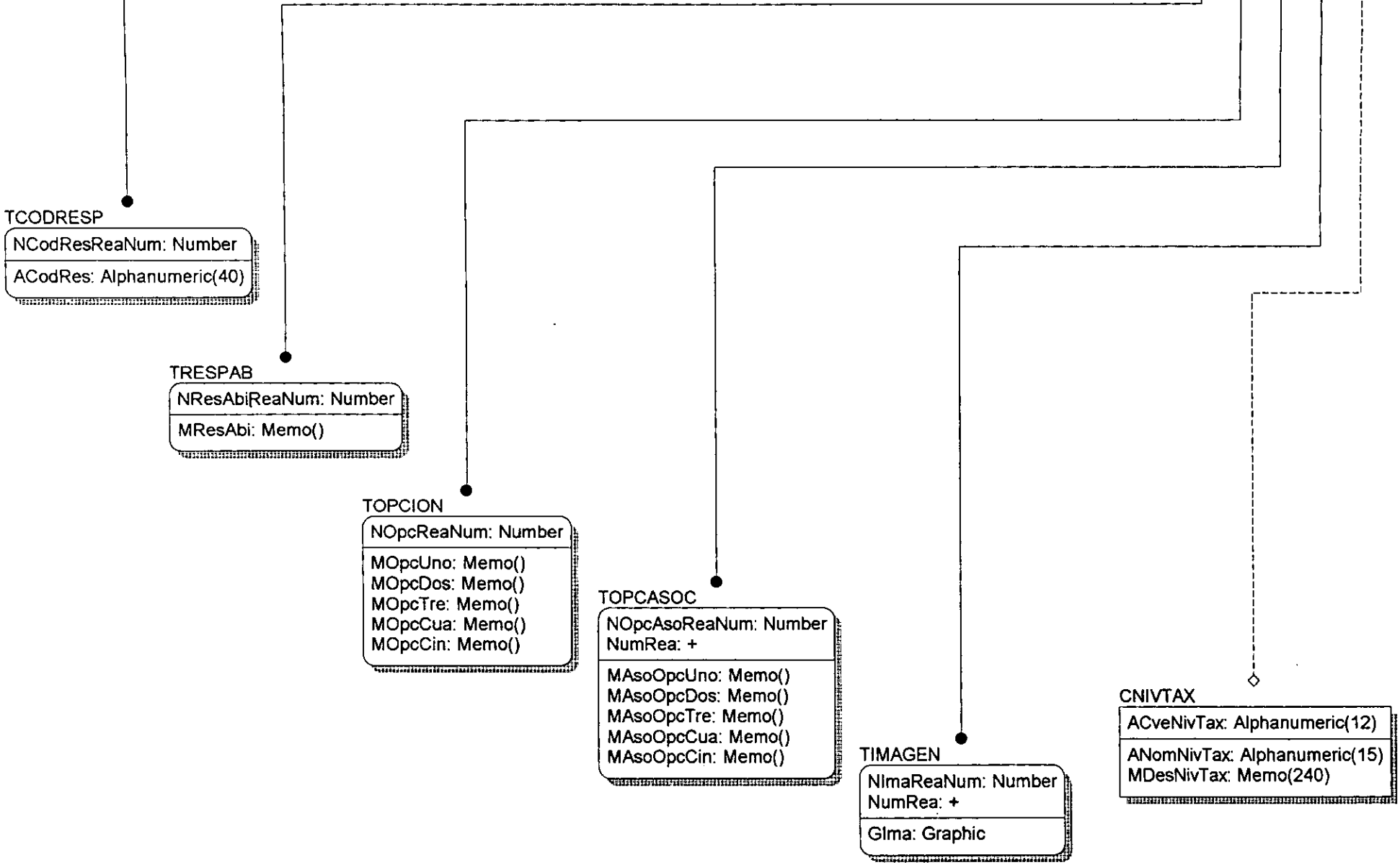
VISTA GLOBAL DEL ESQUEMA ACTUAL DE LA BASE DE DATOS DEL GAR (1/2)



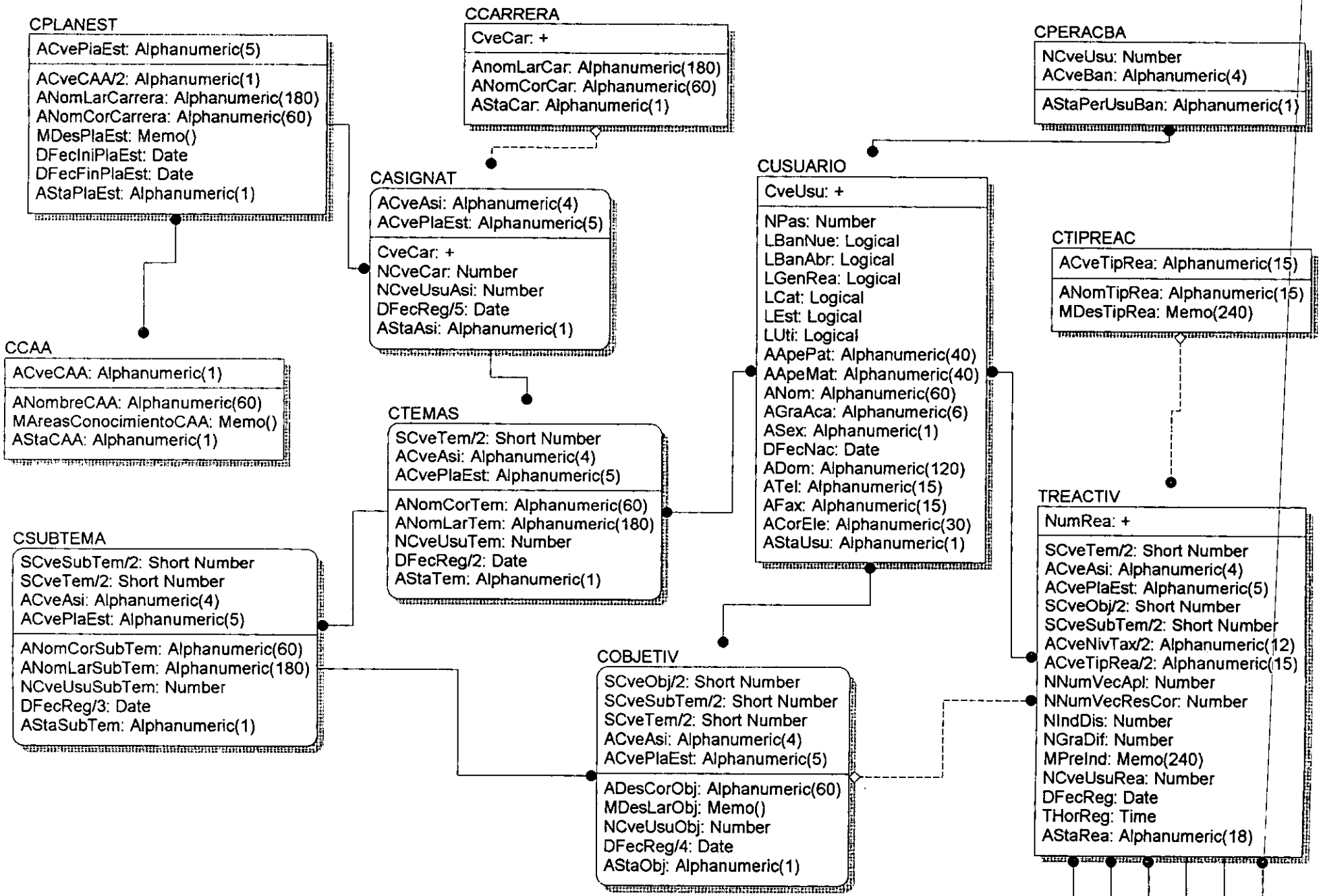
VISTA GLOBAL DEL ESQUEMA ACTUAL DE LA BASE DE DATOS DEL GAR (2/2)



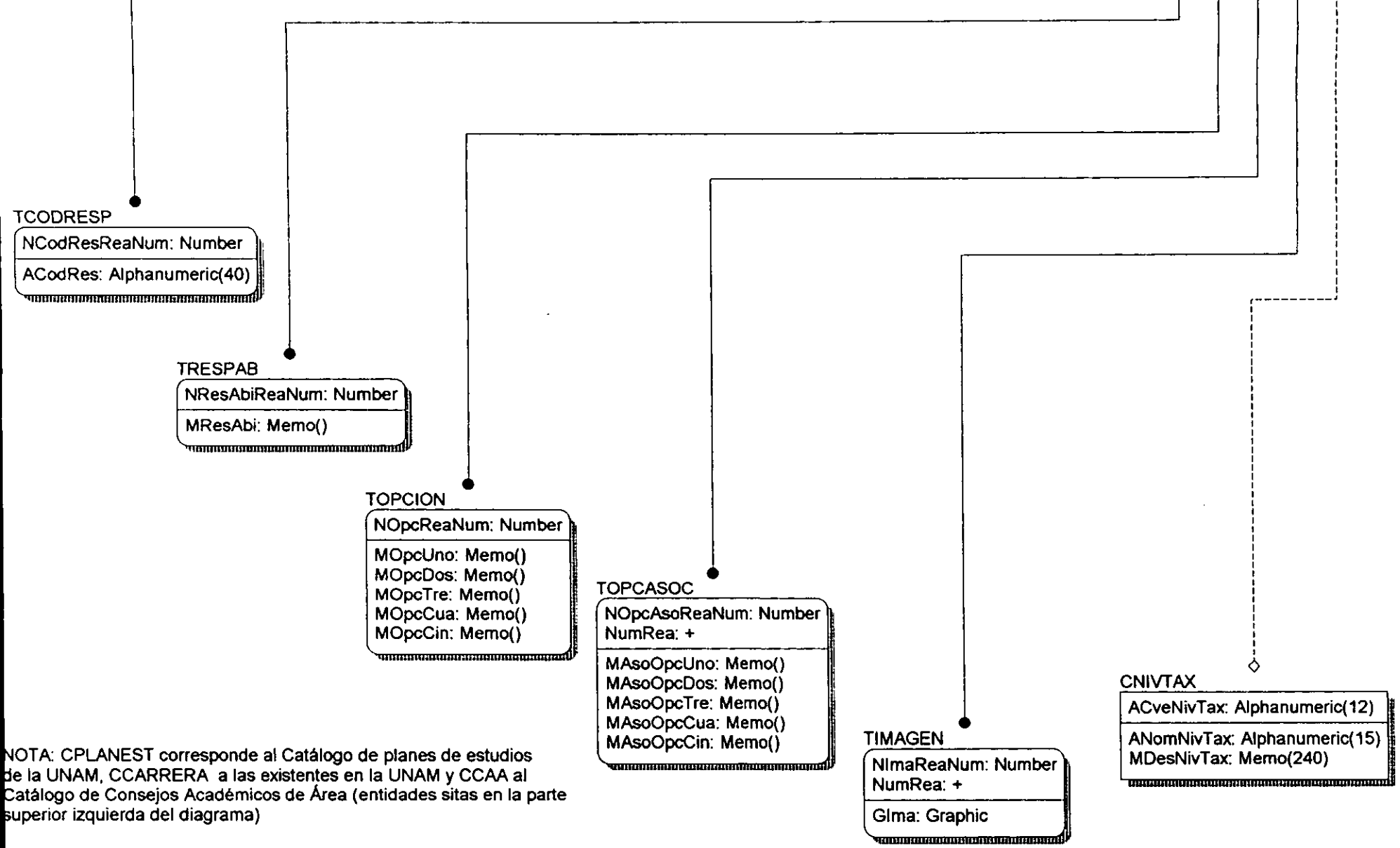
ESQUEMA ACTUAL DE LA BASE DE DATOS DEL CAP (1/2)



ESQUEMA ACTUAL DE LA BASE DE DATOS DEL GAR (2/2)



ESQUEMA DE LA BASE DE DATOS DEL GAR PARA UNA



NOTA: CPLANEST corresponde al Catálogo de planes de estudios de la UNAM, CCARRERA a las existentes en la UNAM y CCAA al Catálogo de Consejos Académicos de Área (entidades sitas en la parte superior izquierda del diagrama)

ESQUEMA DE LA BASE DE DATOS DEL GAR PARA UNA ARQUITECTURA CLIENTE SERVIDOR (2/2)

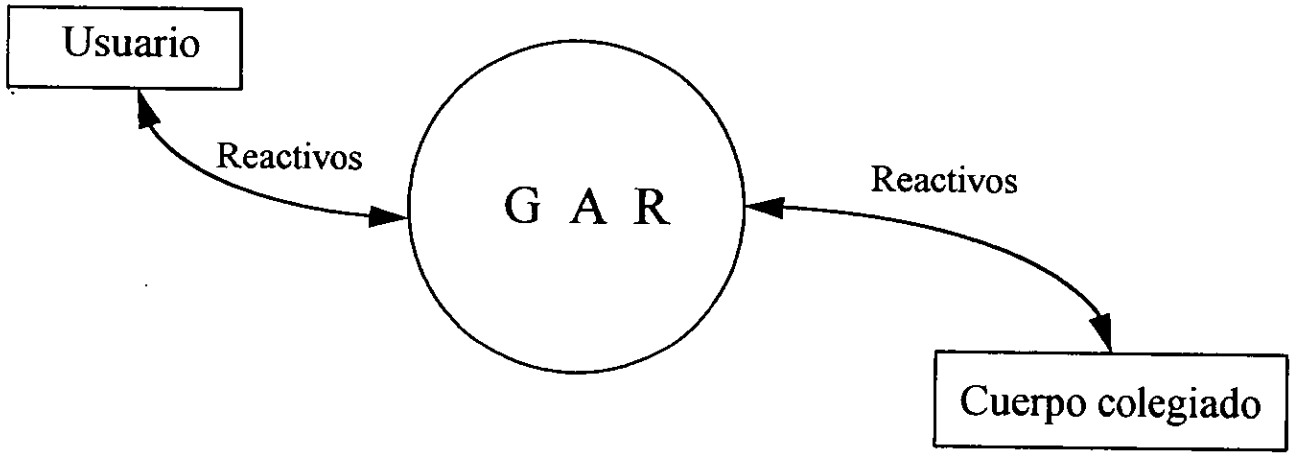
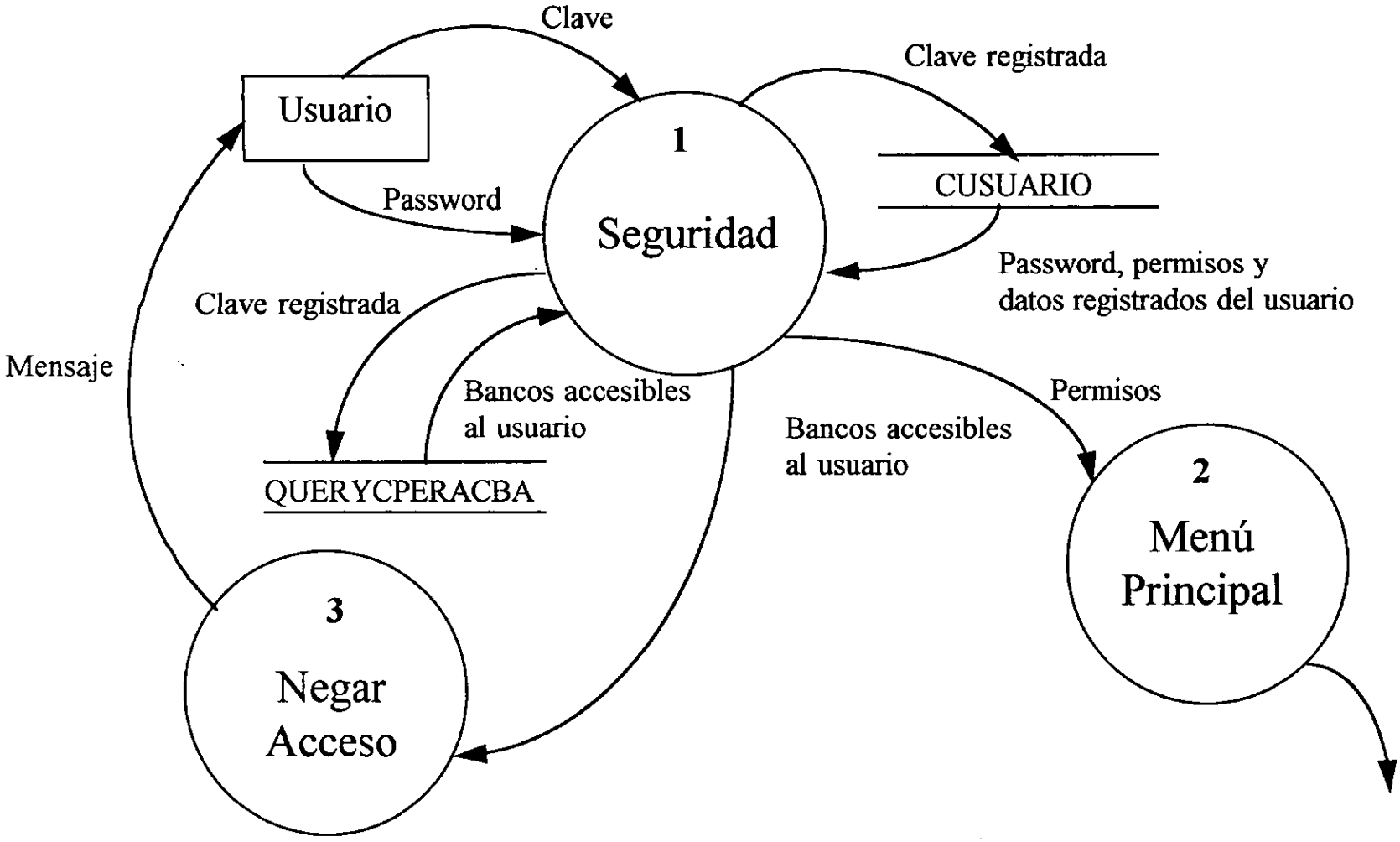
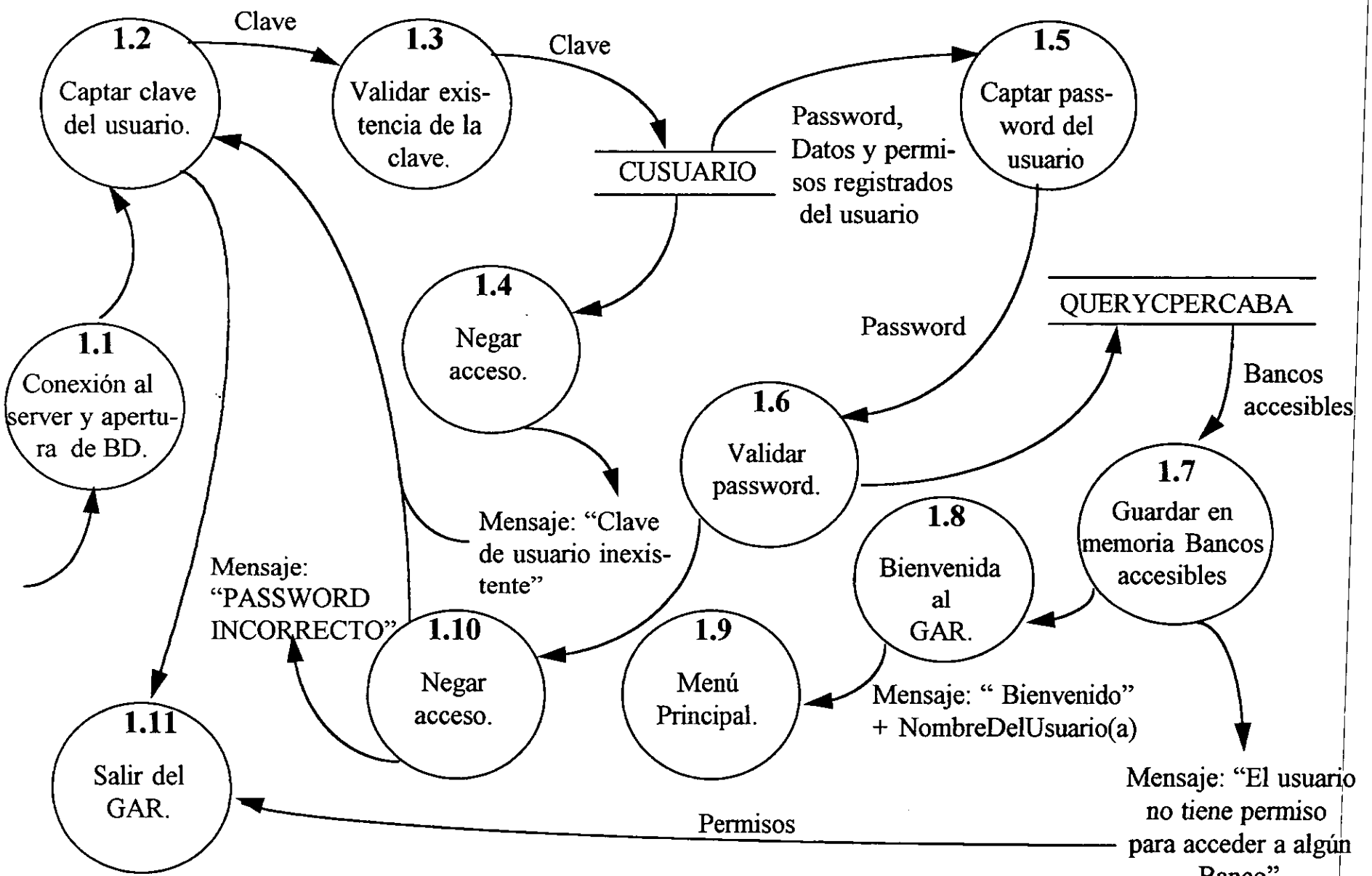


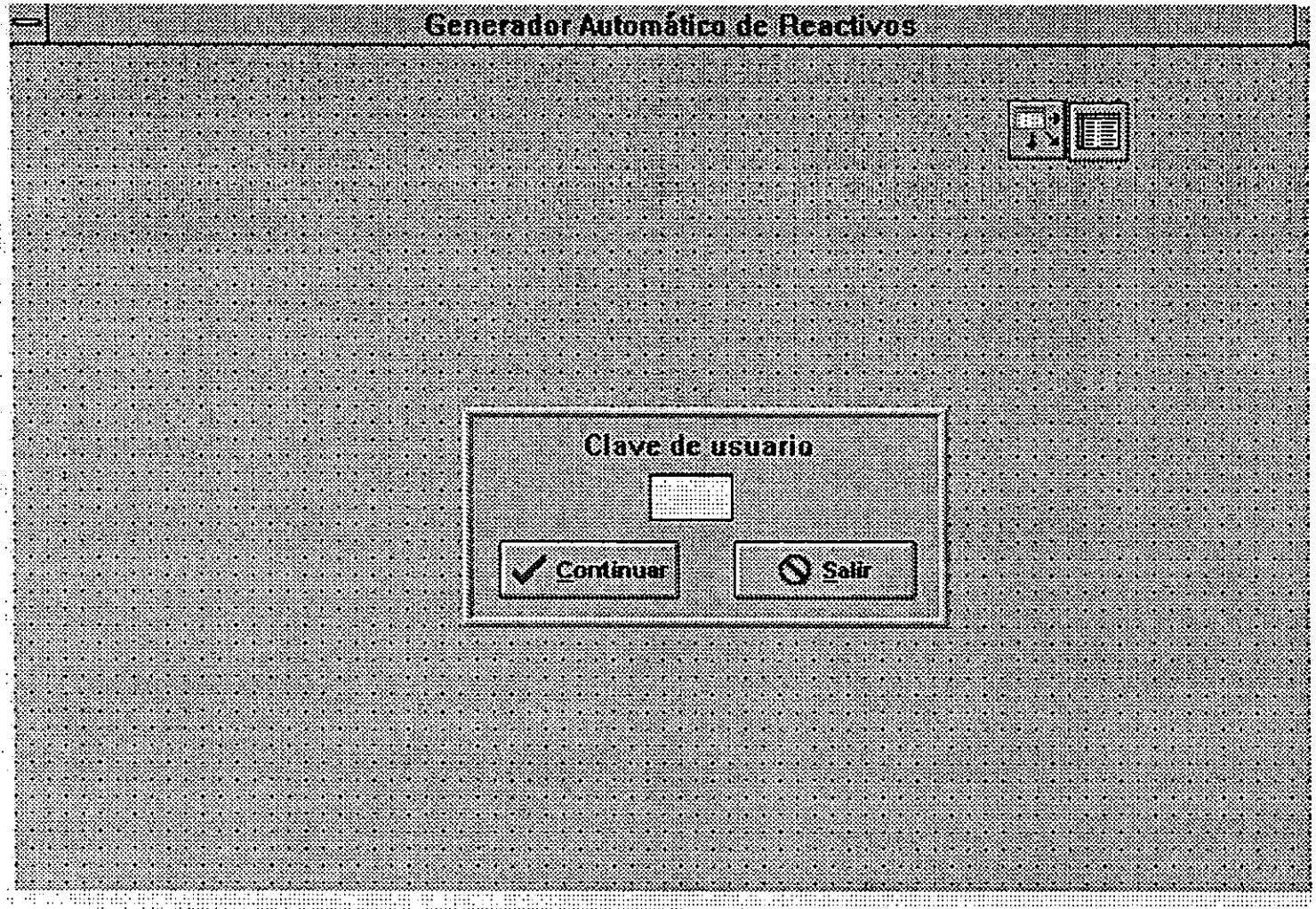
DIAGRAMA DE CONTEXTO



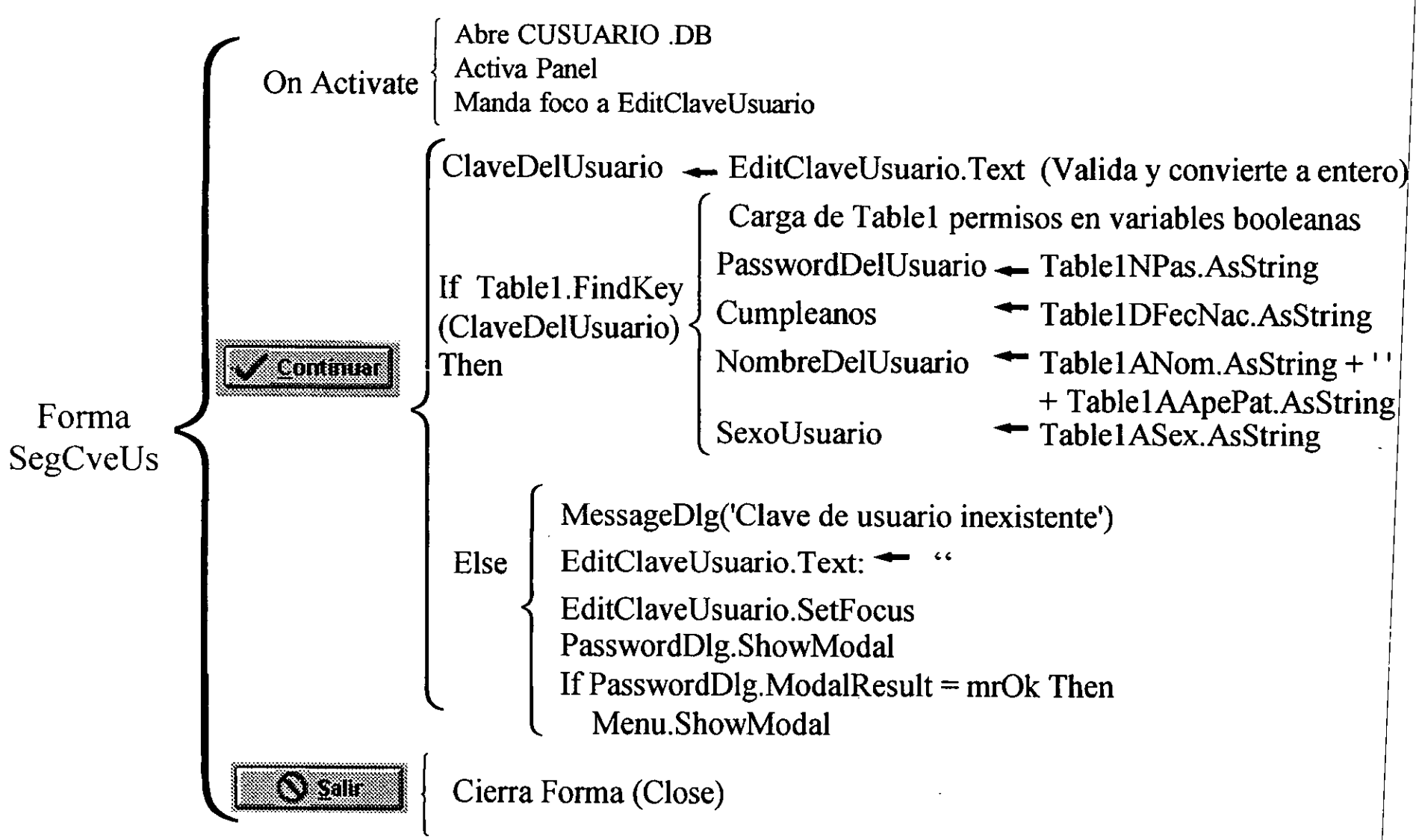
GAR \ MÓDULO DE SEGURIDAD: CONTEXTO DEL EVENTO "EL USUARIO PROPORCIONA CLAVE Y PASSWORD PARA ACCEDER AL MENÚ PRINCIPAL"



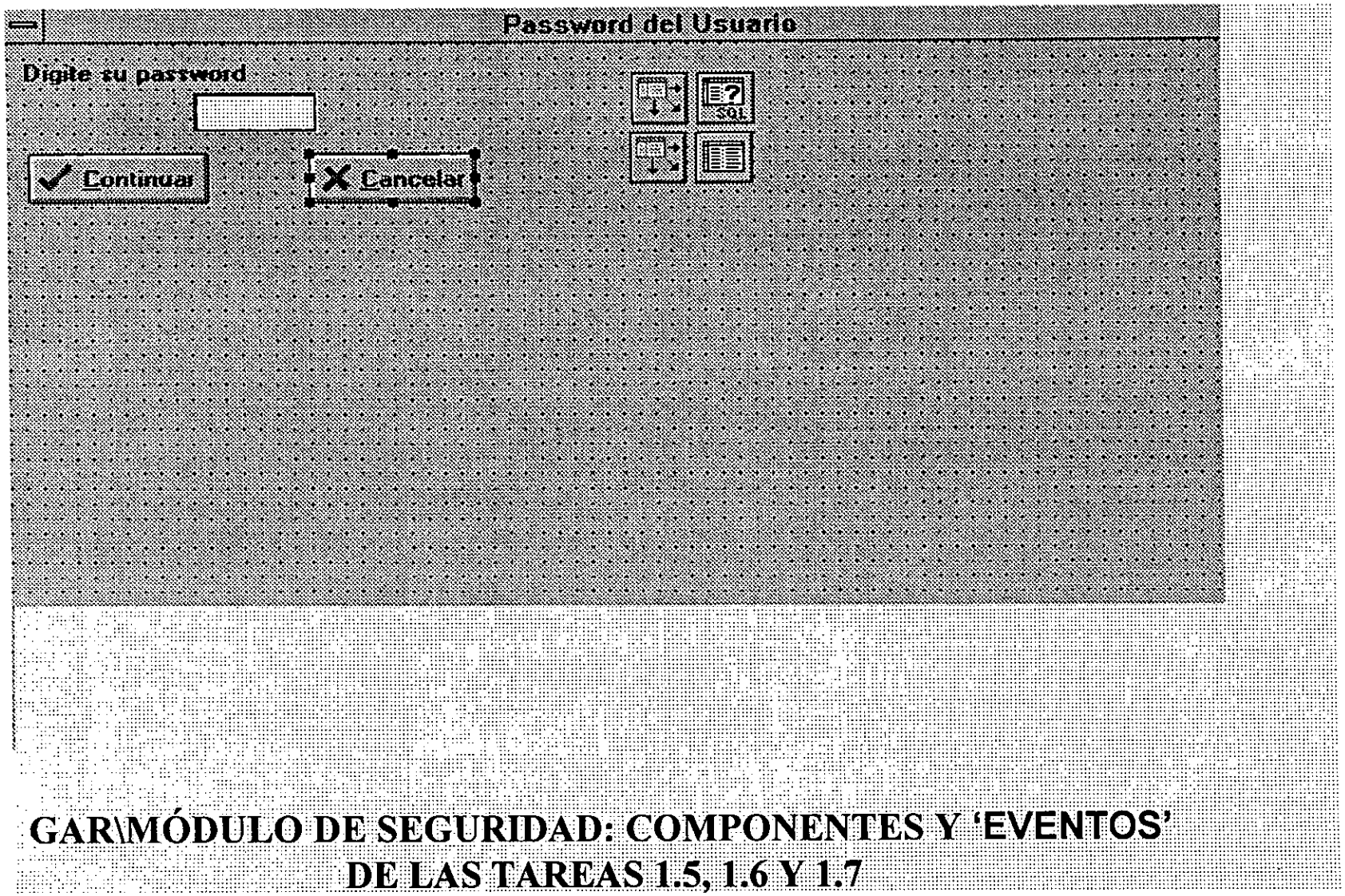
GAR\MÓDULO DE SEGURIDAD: EXPLOSIÓN DE "EL USUARIO PROPORCIONA CLAVE Y PASSWORD PARA ACCEDER AL MENÚ PRINCIPAL"



GAR\MÓDULO DE SEGURIDAD: COMPONENTES Y 'EVENTOS' DE LAS TAREAS 1.2, 1.3 Y 1.4 (FORMA SEGCVEUS)

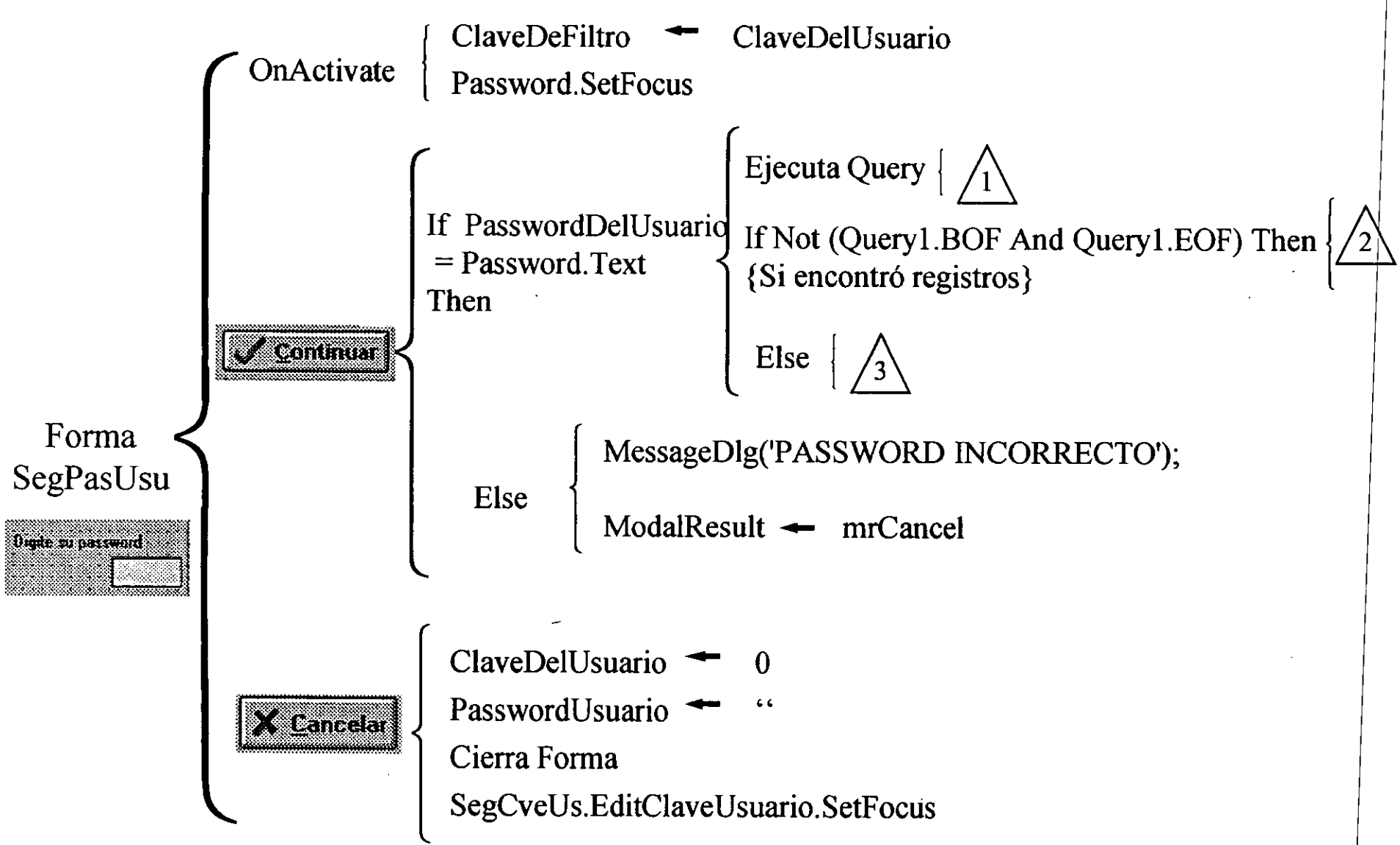


GAR\MÓDULO DE SEGURIDAD: COMPONENTES Y 'EVENTOS' DE LAS TAREAS 1.2, 1.3 Y 1.4



GAR\MÓDULO DE SEGURIDAD: COMPONENTES Y 'EVENTOS' DE LAS TAREAS 1.5, 1.6 Y 1.7

(FORMA SEGPASUS)



GAR\MÓDULO DE SEGURIDAD: EVENTOS Y COMPONENTES DE LAS TAREAS 1.5, 1.6 Y 1.7

1

```
With Query1 Do
   Begin
      Close;
      Prepare;
       Params[0].AsInteger:= ClaveDeFiltro;
      Open
   End;
```

```
Query1.SQL
   Select *
   From CPerAcBa
   Where NCveUsu =:ClaveDeFiltro
```

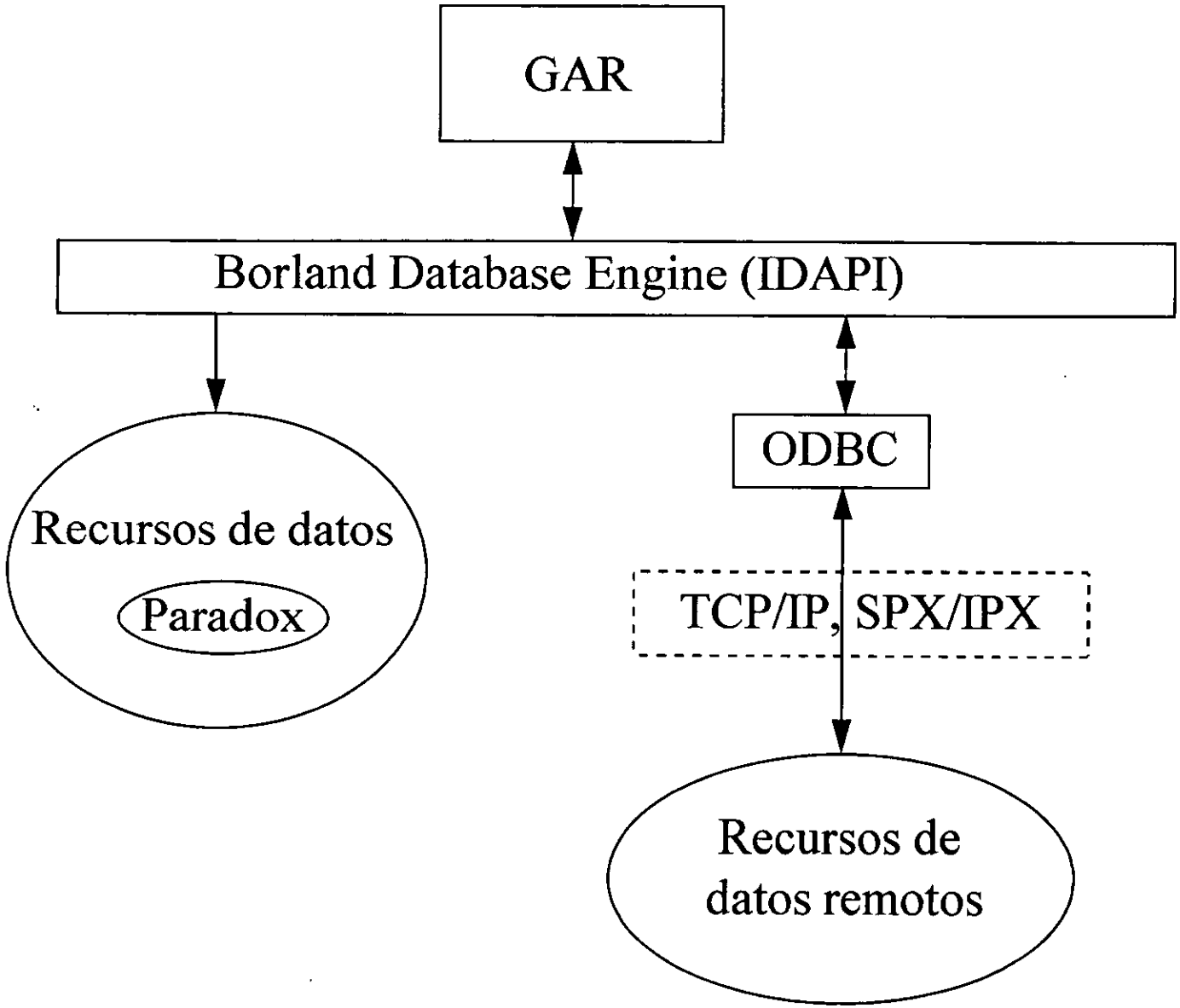
2

```
ArregloDeClaves:=  TStringList.Create;
Query1.First;
While Not Query1.EOF Do {en el Query... And AStaPerUsuBan = '1' -activos-}
    With ArregloDeClaves Do
        Add(Query1ACveBan.AsString);
Query1.MoveBy(1);
ModalResult::= mrOk;
```

3

```
ShowMessage  (NombreDelUsuario + ' no tiene' + Chr(13) +
                  'permiso para acceder a ningún Banco' + Chr(13) + Chr(13)+
                  'CONSULTE CON EL ADMINISTRADOR');
Application.Terminate
```

GAR\MÓDULO DE SEGURIDAD: EVENTOS Y COMPONENTES DE LAS TAREAS 1.5, 1.6 Y 1.7



ARQUITECTURA DEL GAR

```
Unit SegCveUs;
{
+ GENERADOR AUTOMÁTICO DE REACTIVOS (GAR).
+ Objetivos:
  ++ Captar la clave del usuario y validar su registro previo en la tabla  CUsuario.DB, asi como leer entre  otros datos el password
     ahi registrado,  a fin de pasarlos a la Forma PasswordDlg.
+ Formas Madre: No tiene.
+ Formas Hijas:
  ++ FPasswordDlg.
  ++ FMenu que es el Menú principal.
+ Autor: Lucio Gerardo Chávez Heredia.
}

Interface

Uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,  Forms, Dialogs, DB, DBTables, StdCtrls, Buttons, ExtCtrls,
          SegPasUs,
          Menu,
          Reactivo;

Type
  EEnteroErroneo = Class(Exception);
  TFSegPasUsu = Class(TForm)
    PanelUsu: TPanel;
    Label2: TLabel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    EditClaveUsuario: TEdit;
    DataSource1: TDataSource;
    Table1: TTable;
```

```
  Table1CveUsu: TIntegerField;
  Table1NPas: TFloatField;
  Table1LBanNue: TBooleanField;
  Table1LBanAbr: TBooleanField;
  Table1LGenRea: TBooleanField;
  Table1LCat: TBooleanField;
  Table1LEst: TBooleanField;
  Table1LUti: TBooleanField;
  Table1AApePat: TStringField;
  Table1AApeMat: TStringField;
  Table1ANom: TStringField;
  Table1AGraAca: TStringField;
  Table1ASex: TStringField;
  Table1DFecNac: TDateField;
  Table1ADom: TStringField;
  Table1ATel: TStringField;
  Table1AFax: TStringField;
  Table1ACorEle: TStringField;
  Table1AStaUsu: TStringField;

  Procedure FormCreate    (Sender: TObject);
  Procedure FormActivate  (Sender: TObject);
  Procedure BitBtn1Click  (Sender: Tobject); {Botón Salir}
  Procedure BitBtn2Click  (Sender: Tobject); {Botón Continuar}

Private
  { Private declarations }
Public
  { Public declarations }
End;
```

```
Var
 FSegPasUsu                                                    : TFSegPasUsu;
 BienvenidaAlGar, NombreDelUsuario, PasswordDelUsuario, Cumpleanos  :  String;
 ClaveDelUsuario                                               :  Integer;
 PasswordNoValido, OpcBancoNuevo,  OpcBancoAbrir, OpcGenera,
 OpcCatalogo, OpcEstadistica, OpcUtileria                      :  Boolean;



Implementation
{$R *.DFM}

Procedure TFSegPasUsu.FormCreate(Sender: TObject);
Begin
  Table1.Open
End;

Procedure TFSegPasUsu.FormActivate(Sender: TObject);
Begin
  OpcBancoNuevo := False;
  OpcBancoAbrir  := False;
  OpcGenera      := False;
  OpcCatalogo    := False;
  OpcEstadistica := False;
  OpcUtileria    := False;
  PanelUsu.Visible:=  True;
  EditClaveUsuario.SetFocus
End;
```

```
Procedure TFSegPasUsu.BitBtn1Click(Sender: TObject);
Var
 ExisteUsuario: Boolean;

Begin
 Try
  ClaveDelUsuario:= StrToInt(EditClaveUsuario.Text);
 Except
  On EConvertError Do
    Begin
     EditClaveUsuario.Text:= '';
     EditClaveUsuario.SetFocus;
     Raise EEnteroErroneo.Create('La clave debe ser un número entero');
    End;
 End; {Try}
 If Table1.FindKey([ClaveDelUsuario]) Then
   Begin
    PanelUsu.Visible:= False;
    Cumpleanos:= Table1DFecNac.AsString;
    PasswordDelUsuario:= Table1NPas.AsString;
    NombreDelUsuario:= Table1ANom.AsString + ' ' + Table1AApePat.AsString ;
    If Copy(Table1ASex.AsString,1,1) = 'F' Then
      BienvenidaAlGar:= 'Bienvenida '
    Else BienvenidaAlGar:= 'Bienvenido ';
    OpcBancoNuevo := Table1LBanNue.AsBoolean;
    OpcBancoAbrir  := Table1LBanAbr.AsBoolean;
    OpcGenera      := Table1LGenRea.AsBoolean;
    OpcCatalogo    := Table1LCat.AsBoolean;
    OpcEstadistica := Table1LEst.AsBoolean;
    OpcUtileria    := Table1LUti.AsBoolean;
    ExisteUsuario  := True;
```

```
      EditClaveUsuario.Text:= '';
      PasswordDlg.ShowModal;
      If PasswordDlg.modalresult = mrOk Then
         Form1.ShowModal
    End
  Else
    Begin
      MessageDlg('Clave de usuario inexistente',mtInformation,[mbok], 0);
      EditClaveUsuario.Text:= '';
      EditClaveUsuario.SetFocus;
      ExisteUsuario:= False
    End
End;

Procedure TFSegPasUsu.BitBtn2Click(Sender: TObject);
Begin
  Close
End;

End.
```

8. CONCLUSIONES

8.1 SOBRE LA VARIANTE DE LA METODOLOGIA DE YOURDON

Hoy mismo, trabajar en el desarrollo de *software* sin una metodología de por medio es heroico ..... pero muy costoso e inadecuado, sobretodo para paises como el nuestro. Como cite en alguna parte del documento: *"sistema complejo aterrizado, implica alguna metodología se aplicó en su desarrollo"*.

No es el caso de aquéllos "sistemas" enormes -por su número de líneas- que son denominados así, porque sue 'programas' se han ido construyendo *ex profeso*, en y para diversas empresas e instituciones, con muy diversos 'lenguajes' y técnicas de 'programación', así como en muy diferentes 'plataformas' (de lo más heterogéneo posible) a lo largo de 10 o 20 años, de forma inconsciente y que se sostienen -por que no queda de otra- con 'compiladores' y equipos obsoletos o, a través del intercambio de 'archivos planos', por citar algunas situaciones derivadas de inconsistencia informática institucional. De suyo, se sobreentiende que cualquier metodología, y particularmente la variante expuesta en este documento, persigue facilitar y en el mejor de los casos, hacer más eficiente la administración del desarrollo de sistemas y/o aplicaciones de cómputo; por lo mismo la variante inspirada en Yourdon, trata de definir recursos, 'actividades' e instancias organizacionales involucradas en la materia, así como homogeneizar los propios sistemas y/o aplicaciones a través de la 'estandarización' de su documentación, 'funcionalidad', forma y presentación, entre otros elementos, de manera tal que se facilite su control, coordinación y enlace, generando en consecuencia flujos de información uniformes y consistencia informática organizacional.

Derivado del actual proceso de globalización -que desde mi punto de vista inicia apenas su verdadera expansión-, aquéllas organizaciones que cuenten con alguna(s) metodología(s)

que posibiliten de alguna forma consistencia informática organizacional, verán el tercer milenio, bien entrado en años. Las otras no.

"Metodizarse y reorganizarse o morir".

8.2 SOBRE EL 'PROTOTIPO' DEL GAR

Dificilmente se puede dar por terminada la tarea para el desarrollo de una aplicación de cómputo como la expuesta en este documento, que pretende sea aplicado a prácticamente cualquier area de conocimiento para almacenar reactivos y facilitar su acceso y generación de conjuntos de los mismos. Por ejemplo: ¿qué decir sobre la posibilidad de que diversos alumnos de la UNAM y de otras instancias, sitas en distintos lugares del país puedan acceder a un 'servidor' institucional, tal vez vía Internet, para autoevaluar su aprovechamiento -idea, que entre otras mucho más plausibles, aporta el reforzamiento del sentido Instuticional en materia de nacionalidad y liderazgo académico-?. Soslayando el ejemplo, considero por una parte que mínimamente se cumple el objetivo relativo a proporcionar el 'prototipo' de una herramienta que coadyuve al profesorado en el mejor desempeño de sus funciones (siempre y cuando tenga acceso a un equipo PC), al posibilitarles incrementar la calidad de los reactivos usados en los exámenes o tareas, necesarios para la evaluación del aprendizaje de sus alumnos y, por otra parte, sienta las bases para la solución de cuestionamientos como el establecido -entre otros posibles sobre el tema-, aunque a la larga este 'prototipo', como cualquier otro *software* conocido a la fecha, termine por acabar su *ciclo de vida,* en su destino final: el bote de la basura; si acaso, en el recuerdo de las personas que lo conocieron.

Por otro lado, con un solo profesor que aplique el 'prototipo' y de alguna manera pueda intentar, con su ayuda, sistematizar la generación de los instrumentos de evaluación para sus alumnos y homologar los procesos para el diseño y elaboración de los exámenes o tareas del caso, me doy por bien servido, pues demostraría prácticamente que esto es posible -aunque a la fecha no lo parezca-.

8.3 SOBRE ESTA TESIS

Si bien la aplicación adecuada y correcta de cualquier metodología facilita el desarrollo de cualquier *software*, mientras no exista una nueva 'generación' de OS, 'lenguajes' y herramientas de 'programación', equipos o robots, no va ha quedar de otra que cada pieza de *software* -que integradas en su conjunto llegan a constituir verdaderos sistemas de cómputo-, se siga haciendo de forma prácticamente artesanal.

En la novela de Traven sobre las *"canastitas en serie"*, queda claro porque los vecinos del norte pueden seguir produciendo *software* de forma artesanal, pero en serie, con grandes éxitos de venta y exportación, mientras que nosotros no -¡aunque si podamos hacer las canastitas y ellos muchas veces no!-.

Este aspecto de reflexión, me permite estimar que los mexicanos profesionales (que no "profesionistas") de este negocio, tienen la competencia suficiente para construir *software* con calidad de exportación. ¿Valdrá la pena intentarlo...? ¿Cómo, cuándo y dónde?: No lo se; pero con la globalización, cierto estoy de que esto ocurrirá -porque no hay, y no queda de otra ante la incesante y creciente demanda a nivel mundial de productos en la materia-.

Creo que esto podrá iniciarse, cuándo entre otras circunstancias, se haga mayor conciencia de la responsabilidad social que se adquiere con el bagaje que se proporciona en la Facultad.

Mi contribución en esta arena es principalmente, el denotar que deben existir muchas ideas con relación a sistemas innovadores por construir, a diferencia del caso del GAR; así mismo existen muchas áreas de oportunidad en la materia y la cosa es estar atento y ponerse listo.

Sobre el GAR, o mejor dicho, sobre cualquier herramienta de *software* que posibilite el logro de los objetivos planteados en esta tesis, mi comentario final es: "Los reactivos esperan por ser explotados".

ANEXO 1

CUESTIONARIO DE INSPECCIÓN PARA DESARROLLO DE SISTEMAS O APLICACIONES DE CÓMPUTO

xx/yy/199z

OBJETIVO Y CONFIDENCIALIDAD DE LA INFORMACIÓN

El presente cuestionario pretende recabar información mínima necesaria para iniciar el estudio de viabilidad del desarrollo del proyecto de sistematización solicitado por xxxxx xx xxxxxxxxxx. La información que de él se obtenga es de uso confidencial entre ixxxxx xxxxxxxxxx y zzzzzzzzzzzzzzzz zzzzzzzzzzzzzzzz zzzzzzzzzzzzzzz, por lo que no tendrá efecto para cualquier otro fin diferente al mencionado previamente.

Exploración sobre requerimientos al sistema

1.- ¿Cuál es la situación actual del área a su cargo en materia de informática?

_____

_____

_____

_____

_____

_____

2.- ¿Cuáles son las principales necesidades del área a su cargo en orden de importancia?

NECESIDAD 1: _____

_____

NECESIDAD 2: _____

_____

NECESIDAD 3: _____

_____

3.- En materia de lo que se pretende sistematizar ¿El área a su cargo cuenta con alguna herramienta o sistema de cómputo para el tratamiento y explotación de la información en cuestión?

_____

_____

_____

_____

_____

_____

4.- ¿Cuáles son los principales problemas del área a su cargo en orden de importancia?

DE ORDEN ( ): Exceso de controles

DE ORDEN ( ): Falta de controles

DE ORDEN ( ): Comunicación entre las áreas a su cargo

DE ORDEN ( ): Comunicación con otras áreas

DE ORDEN ( ): Falta de información o poca confiabilidad de la misma

DE ORDEN ( ): Falta de oportunidad de la información

DE ORDEN ( ): Exceso de cargas de trabajo

DE ORDEN ( ): Otro (especifique): _____

_____

_____

DE ORDEN ( ): Otro (especifique): _____

_____

_____

DE ORDEN ( ): Otro (especifique): _____

_____

_____

5.- ¿Por qué cree que la situación actual del área a su cargo puede mejorarse con la sistematización?

_____

_____

_____

_____

_____

_____

_____

6.- ¿Cuáles son las principales oportunidades de mejoramiento que tiene el área a su cargo y que es posible abordar o cubrir con la sistematización solicitada, en orden de importancia?

OPORTUNIDAD 1: _____

¿POR QUÉ? _____

_____

_____

OPORTUNIDAD 2: _____

¿POR QUÉ? _____

_____

_____

OPORTUNIDAD 3: _____

¿POR QUÉ? _____

_____

_____

OPORTUNIDAD 4: _____

¿POR QUÉ? _____

_____

_____

7- ¿Cuáles son las principales respuestas hacia los usuarios que esperaría de la sistematización?

_____

_____

_____

_____

_____

8- ¿Cuáles son las principales políticas que debe vigilar el área a su cargo y debe contemplar la sistematización?

POLÍTICA 1: _____

_____

_____

POLÍTICA 2: _____

_____

_____

POLÍTICA 3: _____

_____

_____

POLÍTICA 4: _____

_____

_____

POLÍTICA 5: _____

_____

_____

9.- ¿Se cuenta con el apoyo o el conocimiento de la superioridad para la sistematización?

SI ____ ——> ¿Qué expectativa tiene la superoridad de la sistematización? _____

_____

_____

_____

NO ____ ——> ¿Por qué? _____

_____

_____

_____

10.- Indique las *Áreas involucradas* y/o *Fuentes de información* que debe contemplar la sistematización:

Áy/oF 1: _____

¿RECIBE INFORMACIÓN DEL **Áy/oF 1**? **NO** ____ **SI** ____ ——> ¿De qué tipo, con qué frecuencia y volumen?

_____

_____

¿ENVÍA INFORMACIÓN AL **Áy/oF 1**? **NO** ____ **SI** ____ ——> ¿De qué tipo, con qué frecuencia y volumen?

_____

_____

Áy/oF 2: _____

¿RECIBE INFORMACIÓN DEL **Áy/oF 2**? **NO** ____ **SI** ____ ——> ¿De qué tipo, con qué frecuencia y volumen?

_____

_____

¿ENVÍA INFORMACIÓN AL **Áy/oF 2**? **NO** ____ **SI** ____ ——> ¿De qué tipo, con qué frecuencia y volumen?

_____

Áy/oÉ 3: _____

¿RECIBE INFORMACIÓN DEL **Áy/oÉ 3**? **NO** ____ **SI** ____ ----> ¿De qué tipo, con qué frecuencia y volumen?

_____

_____

¿ENVÍA INFORMACIÓN AL **Áy/oÉ 3**? **NO** ____ **SI** ____ ----> ¿De qué tipo, con qué frecuencia y volumen?

_____

_____

Áy/oÉ 4: _____

¿RECIBE INFORMACIÓN DEL **Áy/oÉ 4**? **NO** ____ **SI** ____ ----> ¿De qué tipo, con qué frecuencia y volumen?

_____

_____

¿ENVÍA INFORMACIÓN AL **Áy/oÉ 4**? **NO** ____ **SI** ____ ----> ¿De qué tipo, con qué frecuencia y volumen?

_____

_____

Áy/oÉ 5:_____

¿RECIBE INFORMACIÓN DEL **Áy/oÉ 5**? **NO** ____ **SI** ____ ----> ¿De qué tipo, con qué frecuencia y volumen?

_____

_____

¿ENVÍA INFORMACIÓN AL **Áy/oÉ 5**? **NO** ____ **SI** ____ ----> ¿De qué tipo, con qué frecuencia y volumen?

_____

_____

11.- ¿Qué documentación tiene disponible el área a su cargo en materia de su organigrama, funciones, manual de procedimientos, proyectos, etc. ?

_____

_____

_____

_____

_____

_____

_____

_____

12.- ¿Cuáles son los principales riesgos y/o obstáculos que percibe para la sistematización?

_____

_____

_____

INFRAESTRUCTURA DE CÓMPUTO Y DE PERSONAL

13- ¿Cuál es el equipamiento con el que cuenta el área a su cargo en materia de:

Hardware?

Servidor **NO** ____ **SI** ____ —> ¿Cuántos y de qué características? _____

_____

Equipos PC **NO** ____ **SI** ____ —> ¿Cuántos y de qué características? _____

_____

Cableado de Red **NO** ____ **SI** ____ —> ¿Cuántos nodos y de qué características?

_____

NO-Break **NO** ____ **SI** ____ ----> ¿Cuántos y de qué características? _____

_____

Impresoras **NO** ____ **SI** ____ —> ¿Cuántos y de qué características? _____

_____

Software?

Base de datos **NO** ____ **SI** ____ —> ¿Cuáles y de qué características? _____

_____

De oficina **NO** ____ **SI** ____ —> ¿Cuáles y de qué características? _____

_____

De la institución **NO** ____ **SI** ____ —> ¿Cuáles y de qué características? _____

_____

14.- ¿Cuál es el personal con el que cuenta para instrumentar la sistematización?

NOMBRE PERSONA 1:_____

HERRAMIENTAS DE CÓMPUTO QUE DOMINA: _____

_____

NOMBRE PERSONA 2: _____

HERRAMIENTAS DE CÓMPUTO QUE DOMINA: _____

_____

NOMBRE PERSONA 3:_____

HERRAMIENTAS DE CÓMPUTO QUE DOMINA: _____

_____

NOMBRE PERSONA 4: _____

HERRAMIENTAS DE CÓMPUTO QUE DOMINA: _____

_____

OBJETIVOS Y METAS TENTATIVOS DEL SISTEMA

15.- Redacte en sus propias palabras los objetivos tentativos de la sistematización:

_____

_____

_____

_____

16.- En una etapa posterior a que la sistematización esté implantada ¿Cuáles son las metas adicionales que con base en esa sistematización es posible alcanzar?

_____

_____

_____

_____

DATOS DEL ENTREVISTADO Y DEL ENCUESTADOR

Entrevistado:

Nombre: _____

Cargo: _____

Adscripción: _____

Teléfonos: _____

Encuestador:

Nombre: _____

Cargo: _____

Adscripción: _____

Teléfonos: _____

OBSERVACIONES DEL ENCUESTADOR

_____

_____

_____

_____

_____

_____

_____

_____

ANEXO 2

ESTÁNDARES PARA DISEÑO E IMPLEMENTACIÓN DE SISTEMAS O APLICACIONES DE CÓMPUTO

La siguiente propuesta está orientada para efectos de 'implementación' en el desarrollador de aplicaciones Delphi y en el caso de combinación de productos deben definirse los estándares correspondientes para cada uno de ellos.

1. ESTÁNDARES DE DISEÑO

1.1 Nombre de tablas, catálogos, campos o atributos

* En el contexto de esquemas de bases de datos, los nombres de las tablas y catálogos deben dar idea clara de la información que contienen; para identificar a cada una, el nombre de la tabla será antecedido por una 'T' y el de los catálogos por una 'C' (mayúsculas). Además, la segunda letra de los catálogos y tablas deberá comenzar con mayúscula como en los siguientes ejemplos:

'CArtículos'	Catálogo de artículos
'TProveedorArtículo'	Tabla de proveedores y artículos

* Cada atributo o campo de una tabla deberá indicar al principio de su nombre, con un prefijo, de qué tipo de dato se trata, a fin de que con el sólo nombre del campo el lector del esquema de la base de datos no tenga que buscar la definición en el programa de cómputo o en la base de datos para saber que información está contenida en el campo. Para definir un atributo se utilizarán tres letras de cada palabra de tal forma que permita saber de que se trata o que se quiere decir, como se ejemplifica enseguida:

iNumUsu	Número de usuario (*integer*)
siNumPagDes	Número de pagos descontados (*smallint*)

En caso de existir alguna duda en cuanto a lo que se quiere representar con un nombre, siempre existe la posibilidad de poder consultar el diccionario de datos, el cual debe permitir en todo momento identificar la completa descripción del concepto o atributo.

* Los tipos de datos básicos acorde al estándar de SQL son:

Prefijo	Tipo de dato
b	binary
bt	bit
c	char
d	date
f	float
g	image
m	memo
n	numeric
o	money
r	real
si	smallinteger
sd	smalldate
so	smallmoney
ti	tinyt
vc	varchar ()
vb	varbinary
x	text
y	integer

1.2 Nombre de componentes en Delphi

Enseguida se enlistan los nombres, prefijos y páginas (en Delphi) de los componentes disponibles para desarrollo de sistemas:

Nombre del componente	Prefijo	Página
BatchMove	bMov	Data Access
Bevel	bevel	Additional
BiGauge	bGau	VBX
BiPict	bPic	VBX

Nombre del componente	Prefijo	Página
BiSwitch	bSwi	VBX
BitBtn	bBtn	Additional
Button	btn	Standard
Calendar	cal	Samples
ColorDialog	cDia	Dialogs
ColorGrid	cGrid	Samples
ComboBox	cBox	Standard
ChartFX	chFX	VBX
CheckBox	chBox	Standard
DataBase	dBas	Data Access
DataSource	dSou	Data Access
DBComboBox	dBCBox	Data Controls
DBCheckBox	dBCHBox	Data Controls
DBEdit	dBEd	Data Controls
DBGrid	dBGrid	Data Controls
DBImage	dBIm	Data Controls
DBListBox	dBLBox	Data Controls
DBLookUpCombo	dBLCom	Data Controls
DBLookUpList	dBLList	Data Controls
DBMemo	dBMem	Data Controls
DBNavigator	dBNav	Data Controls
DBRadioGroup	dBRGpo	Data Controls
DBText	dBTex	Data Controls
DdeClienConv	dCCon	System
DdeClientItem	dcIte	System
DdeServerConv	dSCon	System
DdeServerItem	dSIte	System
DirectoryListBox	dLBox	System
DirectoryOutLine	dOutl	Samples
DrawGrid	dGrid	Additional
DriveComboBox	dCBox	System
Edit	edt	Standard
FileListBox	fLBox	System
FilterComboBox	fCBox	System
FindDialog	fiDia	Dialogs
FontDialog	foDia	Dialogs
Form	frm	
Gauge	gau	Samples
GroupBox	gBox	Standard
Header	head	Additional
Image	image	Additional

Nombre del componente	Prefijo	Página
Label	lab	Standard
ListBox	lBox	Standard
MainMenu	menu	Standard
MaskEdit	mED	Additional
MediaPlayer	mPlay	System
Memo	mem	Standard
NoteBook	nBok	Additional
OleContainer	ole	System
OpenDialog	oDia	Dialogs
OutLine	outL	Additional
PaintBox	pBox	System
Panel	pan	Standard
PopUpMenu	popM	Standard
PrintDialog	pDia	Dialogs
PrinterSetupDialog	pSDia	Dialogs
Query	sql	Data Access
RadioButton	rBtn	Standard
RadioGroup	rGpo	Standard
ReplaceDialog	rDia	Dialogs
Report	rep	Data Access
SaveDialog	sDia	Dialogs
ScrollBar	sBar	Standard
ScrollBox	sBox	Adittional
Shape	shape	Additional
SpeedButton	sBtn	Additional
SpinButton	spinB	Samples
SpinEdit	spinE	Samples
StoredProc	sProc	Data Access
StringGrid	sGrid	Additional
TabbedNoteBook	tBok	Additional
Table	Tbl	Data Access
TabSet	tSet	Additional
Timer	time	System

Las descripciones aquí anotadas corresponden a la vesión 1 del producto y en su caso, deben actualizarze con las correspondientes a versiones subsecuentes del mismo.

2. ESTÁNDARES PARA INTERFAZ GRÁFICA Y FUNCIONALIDAD

2.1 Ventanas

* Las ventanas (mensajes, mensajes transitorios, diálogos, etc.) que sean necesarias para la ejecución deberán utilizar preferentemente los colores y fonts estándar de la versión de Windows en dónde se este ejecutando la aplicación. Las ventanas deberán ser del tipo Windows MDI (Multiple Document Interface), mismas que son perfectamente manipulables desde Delphi a través del objeto TForm.

* Todos los sistemas deben llevar la siguiente figura o splash de presentación:

Aquí se debe incluir la figura y logotípos del correspondiente

2.2 Diálogos

Un dialógo es un tipo especial de ventana, através de la cual el usuario puede proporcionar información a la computadora, ya sea por captura (teclado), por seleción (mouse) o por ejecución directa (teclas aceleradoras).

* Todos deberán de ser del tipo proporcionado por Delphi, teniendo así, al menos el botón para CANCELAR la operación.

* Todos deberán tener un botón con foco por *default*.

* Solamente deberán ser representados con iconos aquellos botones que interactúen con dispositivos distintos de la misma computadora (*v.g.* impresoras, *scaner*, *plotter*).

* El orden de los botones que se lleguen a 'implementar' para manejo de bases de datos, deberán ser rigurosamente el de la forma maestra correspondiente de Delphi.

2.3 Mensajes

Un mensaje puede tener dos tipos particulares de ventanas:

A) No interacción con el usuario: solamente informa al usuario, de procesos o cálculos que se están llevando a cabo en la computadora; tienen una duración de unos pocos segundos con el tiempo suficiente para que el usuario pueda leer el mensaje (dependiendo de la duración del cálculo o del poco o mucho contenido del mensaje).

B) De interacción con el usuario: generalmente son de confirmación con el usuario, así que dichos mensajes deberán contener los botones ACEPTAR, CANCELAR, TERMINAR, SI, NO, etc.

En este contexto:

* Todos deberán aparecer en el centro de la pantalla.

* Todos deberán ser "hijos" de la ventana que los generó.

* Los mensajes de "no interacción" mostrarán la información y el icono "ASTERISK" o "INFORMATION!"

* Los mensajes de interacción con el usuario deberán mostrar el icono "QUESTION" predefinido de Windows y al menos uno de los botones listados en la definición arriba citada, en B).

2.4 Errores

* Un error es cualquier acción o cálculo que la computadora no pudo realizar o completar satisfactoria y totalmente. Un error deberá manifestarse ante un usuario con sonido conocido como BEEP.

* La ventana de error deberá mostrar al usuario el mensaje correspondiente, el icono de "EXCLAMATION" o el icono "STOP" y el botón de "ACEPTAR".

* Durante el mantenimiento de sistemas, debe cuidarse para estos casos el uso adecuado de las sentencias *Try..Except* así como *Try..Finally*.

2.5 Menús

* El menú deberá comenzar con la opción ARCHIVO o equivalente, seguir con SESIÓN (o CONECTARSE) para el caso de acceso a servidores de bases de datos, continuar con las opciones propias del sistema y deberá terminar con la opción AYUDA.

* Los menús deberán ser de tipo dropdown y cascading (facilitados por Delphi).

* El último elemento de la lista de la primera opción antes mencionada (ARCHIVO o equivalente), debe ser SALIR.

* El último elemento de la lista de la opción AYUDA deberá ser ACERCA DE...

* Se deberá subrayar una de las letras (de preferencia la inicial) de las opciónes del menú para acceso rápido desde el teclado.

* Cuando las opciónes del menú no sean aplicables para ciertos usuarios o en casos predeterminados, deberán desactivarse y presentarse con letras desvanecidas.

2.6 Mouse y cursores

Los cursores que se utilizarán dentro de una sesión podrán ser los siguientes:

* Para el modo de edición el cursor "IBEAM"
* Para la navegación el cursor "ARROW"
* Para la navegación dentro de la ayuda "HAND"
* Para el arrastre o selección de elementos "CROSS"

Cualquier interacción con un dispositivo distinto de la misma computadora (acceso al disco duro, una impresora, otra computadora, una base de datos, etc.), asociado a un botón el cual pudiera provocar un lapso considerable de espera, deberá ser representada por el cursor "HOURGLASS"

2.7 Ayuda

Todos los sistemas deben incluir, por lo menos, los siguientes niveles de Ayuda:

Contenido: Donde se explicitarán los diferentes niveles de ayuda por 'implementar'.

Las ayudas que se utilicen se deberán referir a:

* Instruccionés para utilizar la ayuda
* Información acerca de la aplicación
* Tópicos
* Teclados, mouse y sus funciones

Acerca de...: Donde se indicarán los créditos del área desarrolladora y el objetivo del sistema.

3. ESTÁNDARES PARA IMPLEMENTACIÓN

3.1 Guía de aplicación y uso de componentes

* Por cada forma empleada para la 'implementación' del sistema, vía Delphi, se debe elaborar la guía de aplicación y uso de componentes, acorde al siguiente formato:

* Nombres oficial y coloquial, así como versión del sistema. * Nombre del alias del sistema. * Objetivo del sistema.	* Nombre de la unidad y de la trayectoria de los archivos asociados en disco (*.PAS, *.DFM). * Objetivo de la forma en el contexto del sistema.		
Propiedades 'Name' y 'Caption' del componente ('Tag' opcionalmente)	Prefijo del componente	Eventos programados	Observaciones
•	•	•	•
•	•	•	•
•	•	•	•

3.2 Guía de aplicación de componentes para bases de datos

* Por cada componente *DataSource* y sus derivados empleado para la 'implementación' del sistema, vía Delphi, se debe elaborar la guía de aplicación y uso de componentes para bases de datos, acorde al siguiente formato:

* Nombres oficial y coloquial, así como versión del sistema. * Nombre del alias del sistema. * Objetivo del sistema.	* Nombre de la unidad y de la trayectoria de los archivos asociados en disco (*.PAS, *.DFM). * Objetivo de la forma en el contexto del sistema.				
Propiedades 'Name' y 'Caption' del componente ('Tag' opcionalmente)	Prefijo del compo-nente	Propiedad 'Name' de otros com-	Nombre de la tabla y de la trayec-	Nombre de los índices asociados	Observa-ciones

* Nombres oficial y coloquial, así como versión del sistema. * Nombre del alias del sistema. * Objetivo del sistema.	* Nombre de la unidad y de la trayectoria de los archivos asociados en disco (*.PAS, *.DFM). * Objetivo de la forma en el contexto del sistema.				
		ponentes asociados	toria del archivo asociado en disco (*.DB, *.DBF, etc).	a la tabla de la trayecto-ria del archivo asociado en disco (*.MDX, *.PX).	
•	•	•	•	•	•
•	•	•	•	•	•
•	•	•	•	•	•

3.3 Comentarios en el código

Al tope de cada unidad (*.PAS) o biblioteca (*.DLL) empleada por el sistema bajo Delphi, deben aparecer comentadas las siguientes líneas:

* Nombres oficial y coloquial, así como versión del sistema.
* Objetivo de la unidad o biblioteca.
* Ascendiente de la forma (Padre), en su caso.
* Descendientes de la forma (Hijas), en su caso.
* Nombres correspondientes a las instancias organizacionales pertinentes.
* Nombres de los analistas y/o diseñadores y programadores.

Siempre que sea posible debe comentarse el código de la aplicación, componente por componente, sus propiedades iniciales o de default definidas manualmente vía inspector de objetos y *procedure* por *procedure*.

ANEXO 3

ENTREGABLES POR SISTEMA DE CÓMPUTO O APLICACIÓN DE CÓMPUTO

1. Todos los documentos de cada sistema o aplicación de cómputo deben tener la siguiente carátula de presentación:

Aquí se debe incluir la figura y/o logotipo(s) correspondientes

con los nombres correspondientes a las instancias organizacionales pertinentes, así como el nombre del proyecto, informe o rubro contenido en el documento, fecha de actualización y, en su caso, versión del mismo. Todos los documentos deben formarse en forma vertical y en tamaño carta, a excepción del correspondiente a la presentación del sistema, el cual debe formarse en forma horizontal, pero también en tamaño carta.

2. La documentación <u>entregable</u> de los sistemas debe contener:

1 Cuestionario de inspección.
2 Carta del proyecto.
3 Presentación del proyecto.
4 Requerimientos al sistema.
5 Lista de eventos.
6 Un conjunto de DFD nivelados adecuadamente.
7 El diccionario de datos.
8 Esquema(s) de la(s) base(s) de datos y/o diagramas de entidad-relación.
9 Diagrama con la arquitectura del sistema.
10 Reglas de la organización contenidas en el sistema.
11 Soporte normativo de las reglas de la organización contenidas en el sistema.

12 Programas fuente y compilados de los procesos *standalone* o cliente y de los procesos del servidor, en su caso.

13 Formas, iconos y demás componentes gráficos empleados por el sistema.

14 Narrativa de la ayuda del sistema y/o manual de Operación.

15 Manual y/o programas de instalación

16 Guía de aplicación y uso de componentes.

17 Guía de aplicación y uso de componentes para bases de datos.

18 Sanción del cumplimiento de estándares de diseño, diagramación y documentación de sistemas.

19 Definición del flujo e interacciones para el intercambio de información entre las diferentes áreas organizacionales, atendiendo a estrictos niveles de confidencialilidad en la materia, en su caso.

3. La documentación en medios magnéticos de los sistemas deben almacenarse en un servidor a cargo del Área yyy:

* Por cada sistema se definirá un subdirectorio cuyo nombre será compuesto por el acrónimo del mismo (*v.g.* Sistema Integral para la Administración -SIA-) y la versión a que coresponda (*e.g.* SIA**01**, SIA**02**, SIA**03**).

* Por cada subdirectorio de sistema, existirá un subdirectorio para cada rubro de la documentación entregable antes mencionada.

ANEXO 4

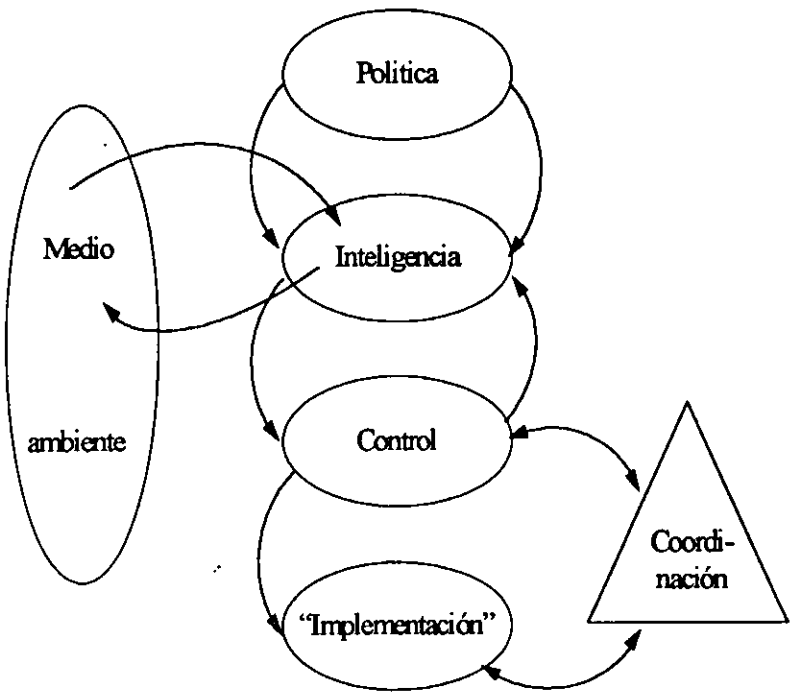
PROPUESTA DE ORGANIZACIÓN PARA IMPLANTAR EL GAR

Uno de los problemas fundamentales a resolver al implantar cualquier sistema es el referente a la organización de los recursos humanos que formarán parte del mismo. En este contexto, se plantéa una propuesta en la materia considerando el modelo definido por Staford Beer entre 1972 y 1979, relativo a la cibernética en las organizaciones, que plantea manejar un esquema de sistemas a partir de las funciones que desarrollan sus elementos.

EL MODELO

En este contexto se define **complejidad** como el número posible de estados de la organización, que le permite responder a los requerimientos del medio ambiente y **variedad**, se define como una "medida" de la complejidad.

La conceptualización del modelo se presenta esquemáticamente como sigue:



CARACTERÍSTICAS DEL MODELO

La descripción de los componentes o "funciones" antes establecidas es la siguiente:

"Implementación".- Se constituye por las actividades que lleva a cabo una organización; es precisamente lo que define está función: todo aquéllo que deja de hacer y que repercuta directamente en su subsistencia, queda comprendido en este concepto. Por ejemplo: una fábrica de tornillos de precisión no subsistiría en el mercado como tal, si deja de hacer tornillos. Se puede caracterizar por las siguientes frases:

Las actividades sustantivas de la organización se realizan aquí, así como la "implementación" de políticas preestablecidas en la materia.

Presenta una cascada de niveles jerárquicos -con cierta autonomía a cada nivel-.

Se conforma por varios elementos, donde cada elemento es responsable de su propio desempeño.

Plantea a sus directivos el problema de mantener a la organización integrada y con identidad.

Control.- Es la integración de las partes en un todo a través de la administración efectiva de la función instrumentación y trata de integrar las partes en un todo, considerando que:

Si las divisiones de la organización presentan mayor variedad que la de los directivos, de algún modo se deben emparejar las variedades entre ambos.

Si se atenúa la complejidad de las divisiones, debe considerar que también se atenúa la complejidad de la organización en su conjunto.

Debe existir un canal de información jerárquico, tal que permita máxima autonomía a las divisiones y máximo logro de objetivos.

Coordinación.- Es la coordinación de las interacciones entre las funciones instrumentación y control. Si las diversas funciones de instrumentación de la organización se autorregulan adecuadamente, la necesidad de la función control es menor, de donde, en caso contrario, la coordinación resulta un medio para atenuar la complejidad de las organizaciones y poder ejercer la función control correctamente. Se caracteriza también por los siguientes rubros:

Puede considerarse un mecanismo o filtro que atenúa la complejidad de la función "implementación", para obtener un buen desempeño e incrementar el desempeño de la organización.

Si la función control es muy estricta, se tendrá una menor necesidad de coordinación.

Inteligencia.- Esta función la definen la detección y "monitoreo" de cambios en el medio ambiente que afecten la operación de la organización y que determinan las condiciones de adaptación de ésta a su medio ambiente. En otras palabras: es la encargada de captar y anticipar oportunidades y riesgos para la organización. Al respecto, Beer comenta que:

Siempre debe plantearse la cuestión: ¿está haciendo la organización lo correcto de acuerdo a los objetivos corporativos?.

Debe tener gran capacidad de adaptación al continuo desarrollo tecnológico.

Tiene la capacidad de "mirar" hacia afuera del sistema, detectar oportunidades y riesgos del medio ambiente y analizarlos mediante la función control; no obstante hay que considerar que esa función sólo se aplica hacia adentro del sistema.

Política.- Es el nivel clásico de toma de decisiones. El factor fundamental de su calidad casi siempre está dado por la capacidad del decisor y su conocimiento y valoración sobre hasta donde es posible que la organización responda ante una situación problemática, considerando que:

Se tendrá capacidad de decisión, siempre que exista capacidad suficiente para empatar la variedad que presenten las situaciones problemáticas y para adecuarse al medio ambiente

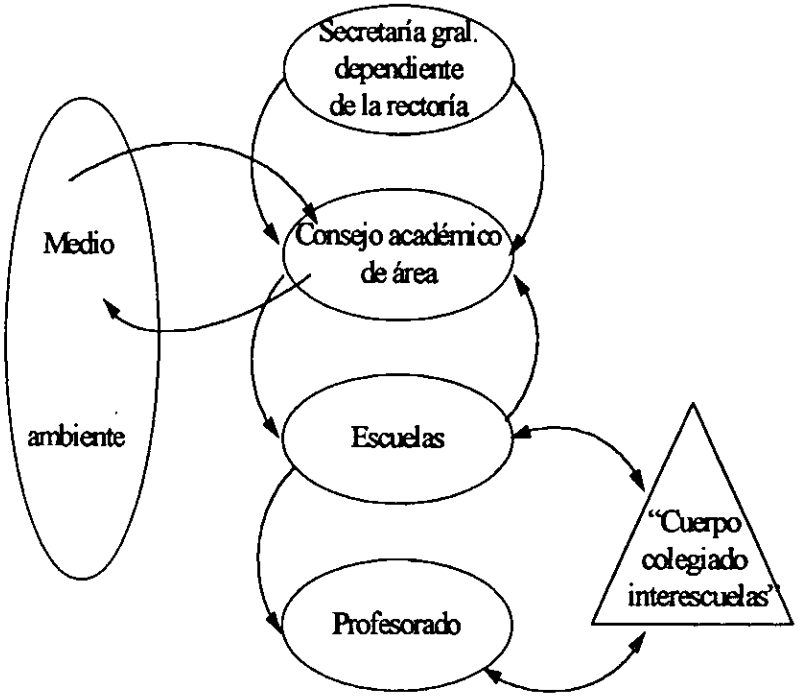
La calidad de la toma de decisiones dependerá de la capacidad de los directivos para "monitorear" todo lo que suceda; de ésta forma se tendrá un equilibrio, en cuanto a complejidades, particularmente entre las funciones control e inteligencia y, en lo general en la organización.

A pesar de que no es explícito, invariablemente existe un flujo de comunicación entre las funciones de control e "implementación" -en ambas direcciones- y, en este sentido es que la flecha que relaciona la primera con la segunda debe entenderse como un "monitoreo".

Una característica importante del modelo es la de ser recursivo, esto es, se puede aplicar tal cual a diferentes niveles jerárquicos de la organización donde se aplique.

PROPUESTA A NIVEL UNAM

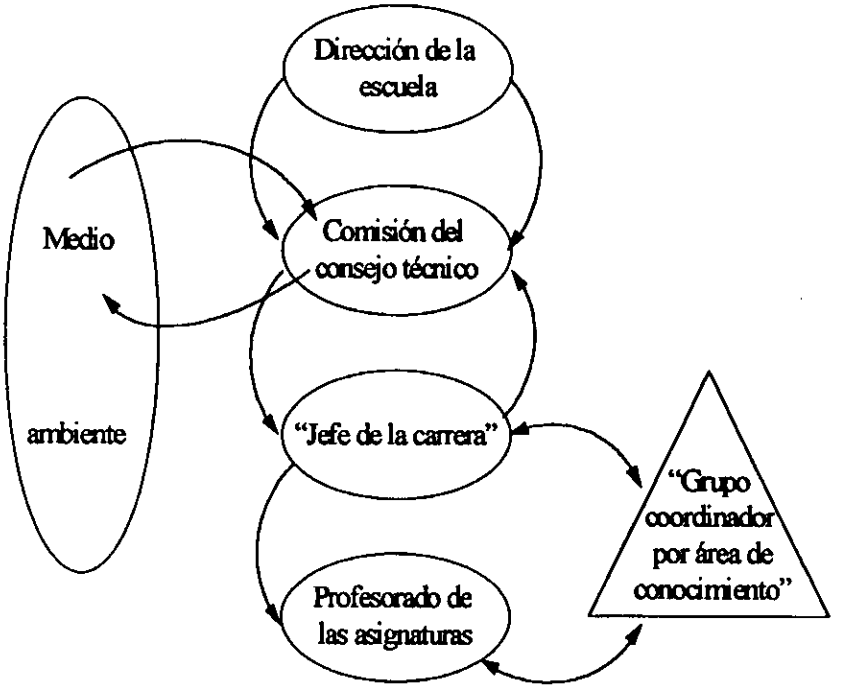
Considerando lo antes expuesto, la propuesta concreta de organización de nivel general para la implantación de un generado de reactivos es la siguiente:



El lector notará que se está suponiendo que los consejos académicos de área tienen una "fuerza" y funcionalidad que a la fecha no tienen y que se plantea la instrumentación de un órgano colegiado interescolar al interior de la UNAM -que a la fecha no opera-, que entre otras de sus funciones podria contribuir en la valoración de los bancos de reactivos.

PROPUESTA A NIVEL FACULTAD, ESCUELA, UMD Y BACHILLERATO

En virtud de la recursividad del modelo, se presenta una propuesta particular para la organización interna de las funciones en el ámbito en cuestión:



El lector observará la propuesta para que una comisión del consejo técnico pueda interactuar en materia de bancos de reactivos con su medio ambiente (otras universidades y/o empresas), lo cuál no existe a la fecha. De forma análoga, la función denominada "jefe de la carrera", que bien puede ser otro órgano colegiado, se plantea para "controlar" al profesorado en aspectos académicos sobre actualizaciones a los bancos de reactivos, en enlace con un "grupo coordinador por área de conocimiento": ese grupo bien podría estar formado por grupos de trabajo y/o profesores eméritos de las escuelas. Como establece el modelo de Beer, en la medida que las funciones del profesorado sobre los bancos de reactivos se autorregulen adecuadamente, la necesidad del "control" tenderá a ser menor con lo que las funciones "jefe de la carrera" y grupo coordinador por área de conocimiento" tenderían a fusionarse a la larga. El el caso de esta propuesta, el flujo de información implicito entre la el "control" y la "implementación" lo constituye el GAR.

ÍNDICE

LISTA DE ACRÓNIMOS

BDE	Borland Database Engine
CASE	Computer-Aided Software Engineering
C/S	Client/Server
DBMS	Data Base Managment System
DD	Data Dictionary
DFD	Data Flow Diagram
EOF	End Of File
ERD	Entity Relationship Diagram
GAR	Generador Automático de Reactivos
GUI	Graphics User Interface
HTML	HyperText Markup Text
IDAPI	Integrated Database Applied Program Interface
LAN	Local Area Network
MDI	Multiple Document Interface
MS-DOS	MicroSoft Disk Operating System
NIP	Número de Identificación Personal
OS	Operating System
OLTP	Online Transaction Processing
OOA	Object Oriented Analysis
OOP	Object Oriented Programming
RAD	Rapid Application Development
SQL	Structured Query Language
STD	State Transition Diagram
TCP/IP	Transmission Control Protocol/Internet Protocol
UCII	Unidad de Cómputo Instrumentación e Informática de la Facultad de Psicología de la UNAM
UNAM	Universidad Nacional Autónoma de México
WAN	Wide Area Network

ÍNDICE DE FIGURAS

BIBLIOGRAFÍA

Benjamin Samuel Bloom: Taxonomía de los objetivos de la educación, Ed. El Ateneo, Buenos Aires, 1913.

Cashin Jerry: Client/Server technology. The new direction in computing networking, Ed. Computer technology research, 1993.

Centro Europeo de Soluciones Cliente/Servidor de IBM: Fundamentos Cliente/Servidor, Colección: Documentos ComputerWorld, 1995.

DeMarco Tom: Structured Analysis and System Specification, Yourdon Press, 1978.

Gronlund, N. E.: Elaboración de test de aprovechamiento. Trillas México, 1980.

Korth, Henry F. y Silbersc Abraham: Fundamentos de bases de datos. Mc Graw-Hill, 1988

Martin James: Organización de las bases de datos. Prentice-Hall Hispanoamericana, 1977.

Microsoft: The Windows Interface Guideliness For Software Design

Orr K. T.: Structures Systems Development, Yourdon Press, N.Y. 1977.

Peter Coad & Edward Yourdon: Object Oriented Analysis, Yourdon Press, Prentice-Hall. Inc., 1991.

Warnier J:D: Logical construction of Program. Van Nostrand Reinhold, Co. 1976.

Yourdon Edward: Managing the system life cicle, Yourdon Press, Prentice-Hall. Inc., 1988.