UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

METODOLOGIA PARA EL DESARROLLO DE SOFTWARE: DICCIONARIO DE TERMINOS ADMINISTRATIVOS

Seminario de Investigación en Informática

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN INFORMATICA

P R E S E N T A:

MARISELA BARRIOS VAZQUEZ

ASESOR:

M. C. LUIS EDUARDO LOPEZ CASTRO

MEXICO, D. F. 1997

A mis padres por su confianza y fé, pero sobre todo
haberme dado la oportunidad de elegir
mi profesión....

Gracias

A todos mis hermanos pero en especial a
Toña y Ale que con su ejemplo
forjaron a la persona que
soy ahora...

Gracias

Recuerda que nadie logra el éxito
sólo. Ten un
corazón agradecido y se rápido en
reconocer a
quienes te ayudaron.

Al Ing. Jesús Díaz Barriga Arceo por su ayuda y apoyo
que siempre me brindó para realizar esta tesis.

A mi asesor M.C. Luis Eduardo López Castro por
permitirme trabajar con él.

A mis amigas Liliana, Elva, Laura, Gloria L. y todas
aquellas personas de la DGSCA que de una
u otra forma me alentaron y ayudaron
para terminar esta tesis.

Gracias

INDICE

INTRODUCCIÓN

Hoy en día los continuos cambios de la nueva empresa constituyen un nuevo paradigma organizacional. Así como las estructuras organizacionales y el ambiente de los negocios están alterándose continuamente mediante progresivos combos globales, la primera era de la tecnología de la información también experimenta un destino similar.

Se están presentando cambios en la utilización de la tecnología y en la tecnología en si misma, en donde la tecnología se encuentra estrechamente ligada a la evolución dentro de las organizaciones y a la transformación de éstas con respecto al mundo; es decir, al mercado competitivo.

En la actualidad, la tecnología de la información se ha desplazado a la linea frontal en la mayor parte de las organizaciones; se ha hecho estratégica en cuanto a que es considerado un componente necesario para la ejecución de una estrategia de negocios.

También ha ocurrido un cambio en cuanto a quién utiliza las computadoras. Anteriormente los principales usuarios, eran los especialistas técnicos, profesionales y gerentes, quienes diseñaban, implementaban, administraban, controlaban y a menudo eran los dueños de la infraestructura tecnológica de la empresa; con la transición a la nueva era de la tecnología de la información, los usuarios de negocios de la tecnología se han colocado a la vanguardia y han crecido de manera descomunal, contándose por decenas de millones y cada vez más complejos y exigentes. Ya no se satisfacen con depender de los departamentos de sistemas de información de la administración para alcanzar los beneficios que pueda traer la tecnología. Los usuarios desean modelar la tecnologías que se implementa en sus organizaciones, desean controlar su uso y determinar el efecto que tendrá en su propio trabajo, rápidamente comprenden que el uso efectivo de la tecnología acoplada a un cambio en la manera como ellos desempeñan sus negocios determinará su éxito personal y organizacional.

En estos momentos se generan tres cambios fundamentales en la aplicación de las computadoras en los negocios; cada uno afecta un nivel diferente de oportunidad de negocios:

1. De la computación personal al trabajo de la computación en red.
2. De los sistemas aislados a los sistemas integrados.
3. De la computación interna a la computación interempresarial.

La tecnología de la información hace posible que las empresas tengan una estructura de equipo de alto desempeño, para funcionar como negocios

integrados independientemente de la gran autonomía de cada negocio, y alcanzar y desarrollar nuevas relaciones con organizaciones externas, con el objetivo de convertirse en una empresa más grande.

Dada esta situación en la empresas es necesario recalcar la importancia que tiene el mundo de la informática en los negocios y particularmente en las empresas o cualquier tipo de organización; sin embargo, también es importante tener presente que el desarrollo de sistemas como tal juega un papel transcendental en el desarrollo tecnológico, debido básicamente a que cualquier tipo de empresa cuenta con algún sistema de información (o aplicación) que le ayuda a disminuir su carga de trabajo administrativo y que de alguna forma le ayudan a la toma de decisiones. Por ende hay que considerar todos aquellos aspectos técnicos y herramientas que ayudan al profesional de sistemas a construirlos.

Por todas estas razones surge la inquietud de desarrollar un documento en el cual se marquen los puntos más importantes y que deberían tomarse en cuenta para el desarrollo de software (generalmente llamados en las organizaciones como sistemas o aplicaciones).

Para algunas de las empresas de consultoría que se dedican a la prestación de servicios para el desarrollo de sistemas así como las que cuentan con su propia área o departamento de sistemas trabajan tan rápido de acuerdo a la necesidades del usuario, que ocasionalmente olvidan las fases a seguir para el desarrollo de sistemas, pero que instintivamente siguen aun cuando no hay un detalle muy riguroso.

Aunado a todo esto, hoy en día existe una gran diversidad de textos en administración, lo que hace que sea tedioso encontrar una breve y clara explicación de un término muy especifico que nos ayude en el proceso para la toma de decisiones, ya que esto involucra buscar entre los diferentes textos y llevar a cabo un análisis para llegar a una conclusión; de ello, surge la propuesta por parte del Lic. Benjamin Franklin F. de diseñar una aplicación a través de la computadora que permita dilucidar o precisar la definición de términos de manera fácil y rápida.

Como respuesta a esta necesidad se propone la creación de un Diccionario Electrónico de Términos Administrativos que permita la búsqueda de información de manera eficiente, haciendo uso de los recursos informáticos actuales como lo es el ambiente Windows (y bajo el ambiente de programación de Visual Basic 3.0), el cual proporciona una interfaz gráfica accesible para los usuarios, permitiéndoles ejecutar otras aplicaciones simultáneamente mientras hacen uso del diccionario, su fácil manejo permite su acceso a cualquier tipo de usuario con interés en los términos administrativos.

La idea de presentar el diseño de una aplicación aunado al texto de la metodología de desarrollo de software es con el mero interés de demostrar que por muy pequeño que sea un sistema es necesario llevar a cabo un análisis para su desarrollo.

En los siguientes capítulos se explica la interrelación de las organizaciones con la informática (capítulo 1)as etapas básicas para el desarrollo de sistemas (capítulo 2), la importancia del modelado de bases de datos (capítulo 3) y la elección de la herramienta en la cual se va a desarrollar el sistema (capítulo 4).

A partir del capítulo 5 se expone el desarrollo del un Diccionario de Términos Administrativos y al cual se aplicó la metodología explicada a lo largo de esta tesis, tanto la metodología de análisis y diseño estructurado de E. Yourdon, así como el modelo relacional, para bases de datos.

CAPÍTULO 1

ASPECTOS GENERALES

1.1 Introducción

Actualmente la administración a evolucionado de forma considerable, que ha sido necesario integrar otras áreas más o menos afines a la administración, tal es el caso de la informática y las telecomunicaciones entre otras. Estas áreas proporcionan a la administración nuevas expectativas de desarrollo, ya que ambas se conjugan para ofrecer las mejores resultados a los usuarios como a las personas que llevan a cabo los procesos.

Hay que mencionar que la informática funciona como una herramienta que nos permite, en primer lugar analizar procesos que las personas llevan a cabo y que en ocasiones, estos procesos son tan repetitivos que las personas ya no son tan capaces de distinguir en que momento el proceso ya no es tan optimo o si realmente es la mejor forma de llevarlo a cabo, en este aspecto, a través del análisis de sistemas se pueden detectar inconsistencias en los procesos y proponer mejoras al proceso.

En segundo lugar, el objetivo de implantar sistemas, es el de reducir los tiempos de procesamiento, disminuir el margen de error y aprovechar al máximo los recursos disponibles.

Aunado a todo esto, desde mi punto de vista hay que tener muy presente todo el ámbito administrativo y sobre todo conocerlo, para poder ofrecer soluciones acertadas en el momento de proponer cambios en cualquiera de las áreas de una organización. Por todo esto, para mi es importante identificar el marco de referencia en el cual se desenvuelven los profesionales en informática, así como la nueva perspectiva de la informática en la administración.

1.2 Importancia y necesidad de la administración

Actualmente el desarrollo organizacional es uno de los factores más importantes en toda empresa que pretende sobrevivir ante los inevitables cambios de hoy en día; sin embargo, para desempeñar con efectividad las actividades administrativas que se realizan se debe contar con un conjunto de herramientas en forma de conceptos y principios organizacionales que formen parte del lenguaje administrativo. El conocimiento de éstos proporcionará los medios más

eficaces para comunicarse entre los miembros de la organización. Además, estos conceptos definen ciertas acciones sobre "como lograr que se hagan las cosas" en las organizaciones de la mejor manera.

El objetivo de éste capítulo es presentar la importancia que existe entre la administración y el uso de la tecnología (en el área de la informática) para el mejor desarrollo de las organizaciones, pudiéndose lograr esto con el uso de sistemas de información. A continuación se exponen las ideas básicas que explican el mecanismo administrativo en las organizaciones, así como algunas de las definiciones de lo que es la administración.

A la administración se le ha llamado "el arte de lograr que se hagan ciertas cosas a través de las personas".[1] Esta definición se centra en el hecho de que los gerentes cumplen las metas organizacionales haciendo que otros desempeñen las tareas que se requieran, no desempeñando ellos mismos esas tareas.

La administración es eso y mucho más. Por ello es difícil establecer una sola definición; sin embargo, podríamos decir que la administración es el proceso de planear, organizar, dirigir y controlar los esfuerzos de los miembros de la organización, y de aplicar los recursos de ella para alcanzar las metas establecidas.[2] Este término es definido como un proceso porque todos los administradores, prescindiendo de sus habilidades o aptitudes, realizan ciertas actividades interrelacionadas con el fin de lograr sus metas deseadas, entre los pasos que sigue este proceso se encuentran los siguientes: planeación, organización, dirección y control.[3]

La anterior definición de administración indica asimismo que los administradores utilizan todos los recursos de la organización (sus finanzas, equipo, información y personal) para alcanzar sus metas. Dado que, las personas son el recurso más importante de cualquier organización, los administradores limitarían sus logros si no recurrieran a técnicas que les ayuden a motivar a las personas.

Por último dicha definición afirma que la administración consiste en cumplir con las "metas formuladas" de la organización. Ello significa que los gerentes de cualquier organización, tratarán de conseguir finalidades específicas; obviamente, estos fines son particulares de cada organización. Cualesquiera que sean las metas de una organización particular, la administración es el proceso en virtud del cual se alcanzan.

La administración posee ciertas características inherentes que la diferencian de otras disciplinas, y son las siguientes:

---

[1] Stoner, James. *Administración para producir*, pág 4.
[2] *Idem.*
[3] Ver proceso administrativo, pág 8.

a) **Universalidad.** Existe en cualquier grupo social y es susceptible de aplicarse lo mismo en una empresa industrial, que en una agencia de viajes o en un hospital.

b) **Valor instrumental.** Dado que su finalidad es eminentemente práctica, la administración resulta ser un medio para lograr un fin y no un fin en sí misma: mediante ésta se busca obtener determinados resultados.

c) **Amplitud de ejercicio.** Se aplica en todos los niveles o subsistemas de una organización formal.

d) **Especialidad.** Aunque la administración se auxilie de otras ciencias y técnicas, tiene características propias que le proporcionan su carácter específico.

e) **Flexibilidad.** Los principios administrativos se adaptan a las necesidades propias de cada grupo social en donde se aplican.

Después de analizar sus características, resulta innegable la gran trascendencia que tiene la administración en la vida del hombre. Sin embargo, es necesario enunciar algunos de los argumentos más relevantes que fundamentan la importancia de esta disciplina:

1. Con la universalidad de la administración se demuestra que ésta es imprescindible para el adecuado funcionamiento de cualquier organismo social.

2. Simplifica el trabajo al establecer principios, métodos y procedimientos, para lograr mayor rapidez y efectividad.

3. La productividad y eficiencia de cualquier empresa están en relación directa con la aplicación de una buena administración.

4. A través de sus principios la administración contribuye al bienestar de la comunidad, ya que proporciona lineamientos para optimizar el aprovechamiento de los recursos, para mejorar las relaciones humanas y generar empleos.

Sin embargo, todo lo anterior no se logra sólo, como se mencionó en páginas anteriores, se requiere de personas, por esto, es importante conocer cómo se administran las organizaciones y sobre todo quienes (gerentes) realizan las tareas para hacer que las organizaciones funciones sin problemas y de modo eficaz. También es necesario conocer cómo el gerente lleva a cabo esas tareas, qué necesita saber a fin de dirigir en forma eficaz y cómo aplica sus destrezas y conocimientos para alcanzar las metas de la organización.

Usualmente, cuando alguien se convierte en empleado, su primer paso consiste en aprender a ser un buen subordinado. Antes de llegar a ser gerentes, casi todos tienen antes que saber desempeñar bien el papel de subordinado, ya que, prácticamente todos los miembros de una organización están subordinados a alguien.

Una de las mejores maneras de ser un buen subordinado es entender el trabajo del jefe. Ello significa comprender las exigencias que le imponen las necesidades de la organización. Un eficaz subalterno será además capaz de ver su propio papel en relación con otros integrantes del departamento o subdivisión, la función de la subdivisión en relación con la organización, las responsabilidades del gerente de subdivisión y las metas de la organización en su conjunto.

Características esenciales para entender la administración

1. La administración tiene un propósito. La administración trata del logro de algo específico, expresado como objetivo o meta. El éxito administrativo generalmente se mide por el grado en que se alcanzan los objetivos. La administración existe debido a que es un medio efectivo de hacer que se haga el trabajo necesario.

2. La administración hace que sucedan cosas. Los gerentes centran su atención y sus esfuerzos para producir una acción exitosa, saben dónde principiar, qué hacer para mantener las cosas en movimiento y cómo seguir.

3. La administración es una actividad, no una persona o grupo de personas. La palabra manejar es más precisa y descriptiva que administración; sin embargo, el uso popular ha hecho de *administrar* un término ampliamente aceptado; además de ser una actividad que puede estudiarse para obtener conocimientos y adquirir habilidad en su aplicación

4. La administración se logra por, con y mediante los esfuerzos de otros. para participar en la administración se necesita eliminar la tendencia normal de hacer uno mismo las cosas y hacer que las tareas se ejecuten por, con y mediante los esfuerzos de los miembros del grupo. Por lo general una persona adquiere destreza en un tipo especializado de trabajo y logra promociones mediante los conocimientos y habilidad incrementados en este campo de especialización; sin embargo, llega el momento en que una promoción adicional requiere cambiar del papel de especialista al de miembro de la administración. La medida primordial del éxito se convierte en fijar o asegurar un acuerdo sobre los objetivos adecuados y hacer que los demás cumplan estos objetivos. El éxito con que se haga este cambio deliberado determina el potencial del nuevo gerente.

5. Por lo general la administración está asociada con los esfuerzos de un grupo. Es común asociar la administración con un grupo; sin embargo, también es aplicable a los esfuerzos de un individuo. Una empresa cobra vida para alcanzar sus objetivos y éstos son alcanzados con más facilidad por un grupo que por una sola persona. los individuos se convierten en miembros de una empresa para satisfacer sus necesidades y porque creen que los beneficios serán mayores que sus pérdidas o cargas como miembros de grupo.

6. La administración es intangible. Se le ha llamado la *fuerza invisible*, debido a que su presencia está evidenciada por el resultado de sus esfuerzos (orden, empleados satisfechos, espíritu entusiasta y una alta productividad).

7. La administración se ayuda, no se reemplaza, por la computadora. La computadora es una excelente y poderosa herramienta de administración, ya que puede ampliar la visión de un gerente y agudizar su percepción proporcionando más información en forma más rápida para tomar decisiones.

Que hacen los administradores

Una definición funcional describe a los gerentes como planificadores, organizadores, líderes y controles de la organización. En realidad, todo gerente asume una gama mucho más amplia de papeles o funciones para conducir la organización a sus objetivos establecidos. En esta exposición se especificará lo que son y que hacen los gerentes:

Los gerentes trabajan con y por medio de otras personas. El término personas comprende no sólo a los subordinados y supervisores, sino también a otros gerentes en la organización, incluso individuos que no pertenecen a la organización (clientes, proveedores, representantes sindicales y otros). Esas personas y otras proporcionan bienes y servicios o bien utilizan el producto o servicio de la organización. Así pues, los gerentes trabajan en cualquier nivel con todo aquel que les ayude a obtener las metas de la organización.

Los gerentes son responsables y además deben asumir la responsabilidad de los resultados. A ellos les compete verificar que las tareas específicas sean efectuadas debidamente. Se les evalúa atendiendo a la eficacia con que coordinan la realización de dichas tareas. También son responsables por las acciones de sus subordinados. El éxito o fracaso de estos últimos es un reflejo directo del éxito o fracaso de los gerentes.

Los gerentes equilibran metas que rivalizan y establecen prioridades. En todo momento un gerente se halla frente a varios problemas, metas y necesidades de la organización, lo cual requiere su tiempo y recursos (tanto de orden humano como material). Debido a que generalmente dichos recursos son limitados, el gerente debe encontrar un equilibrio entre las diversas metas y necesidades ordenando las tareas diarias por orden de prioridad. Los gerentes han de decidir

asimismo quién llevará a cabo una tarea particular y deben asignar el trabajo al subordinado idóneo.

El gerente debe pensar en forma analítica y conceptual. Para ser un pensador analítico, necesita saber dividir un problema en sus componentes, analizarlos y luego llegar a una solución factible. Pero lo más importante es que debe ser un pensador conceptual, capaz de ver la tarea entera en forma abstracta y relacionarla con otras.

Los gerentes son mediadores. Dado que las organizaciones están constituidas por personas y éstas a veces están en desacuerdo o riñen. Las disputas dentro de una organización pueden disminuir la moral y la productividad, pudiendo volverse tan desagradables o negativas que los empleados competentes algunas veces optan por abandonar la organización. Tales hechos obstaculizan la obtención de las metas de la organización; en consecuencia, a veces los gerentes asumen un papel de mediadores y resuelven las disputas antes que escapen de su control.

Los gerentes toman decisiones difíciles. Toda organización afronta dificultades de cuando en cuando y por lo tanto los gerentes son las personas que tienen la obligación de hallar la solución a problemas difíciles y no dar marcha atrás en su decisión.

1.3 Proceso administrativo

Es más fácil la administración, si se describe como una serie de partes o funciones individuales que integran un proceso total. Un proceso es el conjunto de fases o etapas necesarias para llevar a cabo una actividad. Al definir el término administración se menciona que ésta comprende varias fases, etapas o funciones, cuyo conocimiento exhaustivo es indispensable a fin de aplicar el método, los principios y las técnicas de esta disciplina, correctamente.

De la manera más sencilla se puede definir el proceso administrativo como la administración en acción, o también como: *el conjunto de fases o etapas sucesivas a través de las cuales se efectúa la administración, mismas que se interrelacionan y forman un proceso integral.*[4]

Cuando se administra cualquier empresa, existen dos fases: una estructural, en la que a partir de uno o más fines se determina la mejor forma de obtenerlos, y otra operativa, en la que se ejecutan todas las actividades necesarias para lograr lo establecido durante el periodo de estructuración. A estas dos fases, Lyndall F. Urwick[5] les llama: mecánica y dinámica de la administración. La *mecánica* administrativa es la parte teórica de la administración en la que se establece lo

[4] Munch Galindo & García Martínez. *Fundamentos de administración*, pág. 20.
[5] *Idem*, pág 33.

que *debe hacerse*, es decir, se dirige siempre hacia el futuro. Mientras que la dinámica se refiere a *cómo manejar* el organismo social.

Es importante mencionar que existen diversas opiniones en cuanto al número de etapas que constituyen el proceso administrativo aunque, de hecho, para la mayoría de los autores los elementos esenciales sean los mismos. A continuación se mencionan las características básicas de las cuatro actividades de la administración.

Planeación

La planeación consiste en fijar el curso concreto de acción que ha de seguirse, estableciendo los principios que habrán de orientarlo, la secuencia de operaciones para realizarlo y las determinaciones en tiempos y de números para su realización.[6] Los planes dan a la organización sus objetivos y fijan el mejor procedimiento para obtenerlos. Además, permiten:

1. Que la organización consiga y dedique los recursos que se requieren para alcanzar sus objetivos

2. Que los miembros realicen las actividades acordes a los objetivos y procedimientos escogidos.

3. Que el progreso en la obtención de los objetivos sea vigilado y medido para imponer acciones correctivas en caso de ser insatisfactorio.

El primer paso en la planeación es la selección de las metas de la organización. Después se fijan los objetivos de las subunidades (sus divisiones, departamentos, etc.). Una vez escogidos los objetivos, se fijan los programas para alcanzarlos en una forma sistemática, sin olvidar que al seleccionar los objetivos y elaborar los programas, se debe considerar su factibilidad y aceptación.

La planeación es esencial para el adecuado funcionamiento de cualquier grupo social, ya que a través de ella se prevén las contingencias y cambios que puede deparar el futuro, y se establecen las medidas necesarias para afrontarlas. Los fundamentos básicos que muestran la importancia de la planeación son:

1. Propicia el desarrollo de la empresa al establecer métodos de utilización racional de los recursos.

2. Reduce los niveles de incertidumbre sobre lo que puede presentarse en el futuro, más no los elimina.

---

[6] Reyes Ponce, Agustín. *Administración de empresas*, pág. 165.

3. Prepara a la empresa para hacer frente a las contingencias que se presenten, con las mayores garantías de éxito.

4. Mantiene una mentalidad futurista teniendo más visión del porvenir y un afán de lograr y mejorar las cosas.

5. Condiciona a la empresa al ambiente que le rodea.

6. Reduce al mínimo los riesgos y aprovecha al máximo las oportunidades.

7. Las decisiones se basan en hechos y no en emociones.

8. Al establecer un esquema o plan de trabajo, suministra las bases a través de las cuales operará la empresa.

9. Promueve la eficiencia al eliminar la improvisación.

10. Maximiza el aprovechamiento del tiempo y los recursos, en todos los niveles de la empresa.

Organización

Una vez que se han establecido los objetivos y preparados los planes o programas, el gerente deberá diseñar y desarrollar una organización que ayude a llevarlos a su término. La organización se puede definir como el establecimiento de la estructura necesaria para la sistematización racional de los recursos, mediante la determinación en jerarquías, disposición, correlación y agrupación de actividades, con el fin de poder realizar y simplificar las funciones del grupo social.[7] Los fundamentos básicos que demuestran la importancia de la organización son:

1. Es de carácter continuo; jamás se puede decir que ha terminado, dado que la empresa y sus recursos están sujetos a cambios constantes (expansión, contratación, nuevos productos, etc.), lo que obviamente redunda en la necesidad de efectuar cambios en la organización.

2. Es un medio a través del cual se establece la mejor manera de lograr los objetivos del grupo social.

3. Proporciona los métodos para que se puedan desempeñar las actividades eficientemente, con un mínimo de esfuerzos.

4. Evita la lentitud e ineficiencia en las actividades, reduciendo los costos e incrementando la productividad

[7] Munch Galindo & Garcia Martínez. *Fundamentos de administración*, pág. 103.

5. Reduce la duplicidad de esfuerzos, al delimitar funciones y responsabilidades.

Dirección

Esta etapa del proceso administrativo, llamada también ejecución,, es una función de tal trascendencia, que algunos autores consideran que la administración y la dirección son una misma cosa. Esto es, en gran parte, debido a que al dirigir es cuando se ejercen más representativamente las funciones administrativas, de manera que todos los dirigentes pueden considerarse administradores.

Una vez trazados los planes, hay que decidir la estructura de la organización, el reclutamiento y adiestramiento del personal; el siguiente paso es hacer que se avance en la obtención de las metas definidas, esta función consiste en hacer que los miembros de la organización actúen de modo que contribuyan al logro de los objetivos.

Los componentes comunes a esta etapa son:

- Ejecución de los planes de acuerdo con la estructura organizacional.
- Motivación.
- Guía o conducción de los esfuerzos de los subordinados.
- Comunicación.
- Supervisión.
- Alcanzar las metas de la organización.

En base a los elementos anteriores podemos decir que la dirección es la ejecución de los planes de acuerdo con la estructura organizacional, mediante la guía de los esfuerzos del grupo social a través de la motivación, la comunicación y la supervisión.[8] Dado esto, su importancia radica en que:

1. Pone en marcha todos los lineamientos establecidos durante la planeación y la organización.

2. A través de ella se logran las formas de conducta más deseables en los miembros de la estructura organizacional.

3. La dirección eficiente es determinante en la productividad.

4. Su calidad se refleja en el logro de los objetivos, la implementación de métodos de organización y en la eficiencia de los sistemas de control.

---

[8] *Idem*, pág. 161.

Control

Por último se debe asegurar que las acciones de los miembros de la organización lleven a la obtención de las metas. El control es la medición de los resultados actuales y pasados, en relación con los esperados, ya sea total o parcialmente, con el fin de corregir, mejorar y formular nuevos planes;[8] es primordial en la administración, pues, aunque una empresa cuente con magníficos planes, una estructura organizacional adecuada y una dirección eficiente, no se podrá verificar cuál es la situación real de la organización si no existe un mecanismo que se cerciore e informe si los hechos van de acuerdo con los objetivos. La función de control consta de tres elementos primordiales:

1. Establecer las normas del desempeño.

2. Medir el desempeño actual y compararlo con las normas establecidas.
3. Tomar medidas para corregir el desempeño que no cumpla con esas normas.

En base a estos elementos, su importancia radica en que:

1. Establece medidas para corregir las actividades, de tal forma que se alcancen los planes exitosamente.
2. Se aplica a todo: a las personas, a las cosas y a los actos.

3. Determina y analiza rápidamente las causas que pueden originar desviaciones, para que no se vuelvan a presentar en el futuro.

4. Localiza a los sectores responsables de la administración, desde el momento en que se establecen medidas correctivas.

5. Proporciona información acerca de la situación de la ejecución de los planes, sirviendo como fundamento al reiniciarse el proceso de planeación.

6. Reduce los costos y ahorra tiempo al evitar errores.

7. Su aplicación incide directamente en la racionalización de la administración y consecuentemente, en el logro de la productividad de todos los recursos de la empresa.

En la práctica, el proceso administrativo no incluye cuatro conjuntos aislados o poco conexos de actividades, sino un grupo de funciones interrelacionadas.

[8] Stoner, James. *Administración*, pág. 15.

1.4 Reingeniería de procesos

Actualmente pocas son las empresas cuya administración no afirme que necesite una organización lo bastante flexible para poder ajustarse a las cambiantes condiciones del mercado, ágil para para poder superar el precio de cualquier competidor, tan innovadora que sea capaz de mantener sus productos y servicios tecnológicamente frescos, y tan dedicada a su misión que rinda el máximo de calidad y servicio al cliente.

Lo que ahora se busca son empresas competitivas, diligentes, eficientes, enfocadas al cliente y rentables, ¿por qué tantas son pesadas, torpes, rígidas, perezosas, lentas, no competitivas, no creativas, ineficientes, desdeñosas con respecto a las necesidades del cliente y además pierden dinero?. La respuesta está en cómo hacen su trabajo estas empresas y por qué lo hacen así; ya que, en ocasiones sus resultados son muy distintos de los que buscaban sus administradores.

Las corporaciones no funcionan mal porque los trabajadores sean perezosos o los administradores ineptos, se debe a que el mundo en que operan ha cambiado y rebasa los límites de su capacidad de adaptarse o evolucionar. Los principios sobre los cuales están organizadas se adaptaban a las condiciones de una era anterior, pero ya no dan más. Con el uso de la tecnología y las nuevas expectativas de los clientes, la situación de las organizaciones ha cambiado para dejar obsoletos los objetivos, métodos y principios organizacionales de las corporaciones clásicas.

Renovar su capacidad competitiva no es cuestión de hacer que la gente trabaje más duro, sino de aprender a trabajar de otra manera. Esto significa que las compañías y sus empleados tienen que olvidar muchos de los principios y técnicas que les aseguraron el éxito durante mucho tiempo pero ahora no.

Lo más importante es que si los mismos administradores no aciertan con el origen de sus dificultades, mucho menos saben qué hacer al respecto; muchas personas creen que estas empresas reaccionarían vigorosamente si sólo contaran con el producto o servicio adecuado para los tiempos actuales; sin embargo, no son los productos sino los procesos que los crean los que llevan a las empresas al éxito.

Algunas personas creen que las compañías podrían curarse de todos sus males con nuevas estrategias corporativas, como vender una división y comprar otra, cambiar de mercados, pasarse a otro negocio, manipular los activos o reestructurarse con una compra de acciones a crédito. Pero este modo de pensar distrae a las empresas de efectuar cambios básicos en el trabajo real que hacen.

Incluso, hay quienes atribuyen los problemas corporativos a deficiencias de la administración y piensan que si las compañías fueran manejadas de otra manera prosperarían, pero ninguna de las modas administrativas de los últimos veinte años han detenido el deterioro del desempeño competitivo de las corporaciones, sólo han servido para distraer a los administradores de la tarea realmente necesaria. Otras opinan que la automatización es el remedio para los problemas de los negocios; ciertamente la computadora acelera el trabajo al automatizar tareas que antes se hacían manualmente, pero ello, no significa que existan mejoras fundamentales en el rendimiento.

Actualmente, la nueva tendencia es la ***reingeniería***, que postula abandonar procedimientos establecidos hace mucho tiempo y examinar otra vez el trabajo que se requiere para crear el producto o servicio de una empresa y entregarle algo de valor al cliente. Para Hammer[10] la reingeniería es "la revisión fundamental y el diseño radical de procesos para alcanzar mejoras espectaculares en medidas críticas y contemporáneas de rendimiento, tales como costos, calidad, servicio y rapidez".

Todo ellos significa rechazar las creencias y supuestos recibido, creando enfoques de la estructura del proceso con poca o ninguna semejanza con los anteriores. La definición de reingeniería maneja cuatro términos importantes que son:

a) Fundamental. En la reingeniería de procesos se debe hacer las preguntas más básicas acerca de la organización y sobre cómo funciona. Se debe cuestionar lo siguiente:

¿Por que hacemos lo que estamos haciendo?

¿Y por qué lo hacemos de esa forma?

Hacer estas preguntas obliga a examinar las reglas tácitas y los supuestos en que se basa el manejo de la organización. Con frecuencia esas reglas resultan anticuadas, equivocadas, sin sentido o inapropiadas. La reingeniería de proceso inicia determinado qué debe hacer una organización, luego cómo debe hacerlo. Se olvida por completo de lo que es y se concentra en lo que debe ser.

b) Radical. Rediseñar radicalmente significa llegar hasta la raíz de las cosas, en la reingeniería de procesos no se trata de efectuar cambios superficiales, ni se trata de arreglar lo que ya está instalado, sino abandonar lo arcaico. Se requiere descartar todas las estructuras y procedimientos existentes e inventar maneras realmente nuevas de realizar el trabajo.

---

[10] Hammer, M. y Campy, J., *Reingeniería*, págs. 34-37

c) **Espectacular.** En la reingeniería de procesos no se trata de hacer mejoras marginales o incrementales sino dar saltos gigantescos en rendimiento. La mejora marginar requiere afinación cuidadosa; la mejora espectacular exige eliminar lo viejo y cambiarlo por algo nuevo.

d) **Procesos.** En la actualidad muchas personas que trabajan en las organizaciones no están "orientadas a los procesos", están enfocadas en tareas, oficios, en personas, en estructuras, en división del trabajo, pero no en procesos. Las tareas individuales dentro de un proceso son importantes, pero, ninguna de ellas tiene importancia para el cliente si el proceso global no funciona; es decir, que no entregue los productos y servicios ofrecidos.

Lo que no es la reingeniería de procesos

No obstante que la informática juega un papel importante en la reingeniería, debe quedar claro que la reingeniería de procesos no es lo mismo que automatización. De acuerdo con Hammer y Champy[11] la automatización simplemente ofrece maneras más eficientes de hacer los que no se debe hacer. Tampoco es:

a) Reingeniería de software. Lo cual implica reconstruir sistemas obsoletos de información con tecnología de información más moderna.

b) Reestructurar ni reducir. Reducirse y reestructurarse sólo significa hacer menos con menos, mientras que en la reingeniería de procesos significa hacer más con menos.

c) Reorganizar una organización. Reducir el número de niveles o hacerla más plana. Aún cuando la reingeniería de procesos puede producir una organización más plana, el problema que enfrentan las organizaciones no proviene de su estructura organizacional sino de la estructura de sus procesos.

d) Mejora de la calidad, ni gestión de calidad. Empero, los problemas de calidad y reingeniería de procesos comparten ciertos temas comunes; ambos, reconocen la importancia de los procesos y ambos empiezan con las necesidades del cliente del proceso y trabajan de ahí hacia atrás.

Dado éste nuevo pensamiento administrativo, una empresa que no pueda cambiar su modo de pensar acerca de la informática no puede ser rediseñada. En todo esto, la informática desempeña un papel crucial, debido a que cuenta con herramientas de la tecnología de vanguardia, pero también es muy fácil utilizarla mal; ya que cabe señalar, que el solo hecho de destinar más computadoras a un problema no significa que se haya rediseñado.

---

[11] *Idem.* pág. 40.

Para reconocer el poder inherente de la informática moderna y visualizar su aplicación se requiere que las compañías utilicen una manera de pensar que las personas de negocios no suelen aprender y que tal vez no saben manejar. la mayoría de los gerentes saben pensar en *forma deductiva*. Es decir, saben definir muy bien un problema y luego buscar y evaluar sus diversas soluciones, pero para aplicar la informática a la reingeniería de negocios es necesario pensar en *forma inductiva*: la capacidad de reconocer primero una solución poderosa y en seguida buscar los problemas que ella podría resolver.

Pensar deductivamente sobre la tecnología no sólo hace que la gente desconozca lo que es realmente importante en ella, sino que también la hace entusiasmarse con tecnologías y aplicaciones que son, en realidad, triviales o carecen de importancia. El poder real de la tecnología no está en que puede hacer funcionar mejor los viejos procesos, sino que permite romper con las reglas y crear nuevas maneras de trabajar.

1.5 Tecnología de información

En estos días las sociedades modernas están orientadas a la tecnología y empiezan a modificar sus características a ritmo acelerado. Algunos cambios en el sector de la información han repercutido de manera profunda en la sociedad y en las empresas, debido a que:

- Existe una mayor cultura informática entre los gerentes y la población en general.
- Los progresos en las telecomunicaciones como la fibra óptica, satélites, redes y bases de datos a nivel internacional han reducido virtualmente tiempos y distancias.
- La aparición, transformación y proliferación de las computadoras a reducido los costos asociados al uso de la información.
- Los delitos por computadora y las medidas tendientes a combatirlos son ahora dos aspectos que deben tenerse muy en cuenta.

A medida que aumenta la cantidad y la velocidad con que la información se pone al servicio de los gerentes, el flujo de ella debe ser cada vez más selectiva, siendo la tecnología uno de los aspectos más importantes; sin embargo, hay quienes desconocen la manera en que puede apoyar la tecnología de información a un proceso dentro de una organización; aún cuando las formas son diversas, algunas de ellas se presentan a continuación:

a) Aumentar la velocidad. La tecnología de información puede utilizarse para realizar una actividad en forma más rápida que una persona; además, puede disminuir el tiempo empleado en la parte crítica de un proceso.

b) Archivo y recuperación. La tecnología de información permite archivar información y recuperarla posteriormente con la rapidez, organización y capacidad de búsqueda necesarias, pero con costos crecientes para volúmenes cada vez mayores.

c) Comunicaciones. La tecnología de información puede mover datos e información en un proceso, de un punto a otro, casi de manera instantánea y de diversas formas.

d) Controlar las tareas del proceso y mejorar la calidad. La tecnología de información puede controlar en forma directa las tareas en un proceso de una organización, aumentando en general la calidad del resultado, ya que elimina el error humano y el equipo automatizado puede proveer mediciones y controles de fabricación más exactos que los suministrados por un individuo. Así mismo puede aplicarse para mejorar los procesos de oficina y de conocimientos del trabajador que está vinculado a transacciones y decisiones complejas.

e) Monitoreo. La tecnología de información puede comparar la actividad que se está realizando con un conjunto de estándares, mientras se ejecuta el proceso o después que ha concluido. Así, pueden corregirse los problema tan pronto como se detecten y la función de monitoreo puede someterlo a prueba nuevamente.

f) Apoyo en la toma de decisiones. La información necesaria para la toma de decisiones de la organización puede recopilarse y ampliarse en una parte del proceso para ayudar al personal a tomar mejores decisiones o en algunos casos, para hacerlo de manera automática. Los datos pueden presentarse en formas apropiadas, como gráficas, para que el proceso de decisiones sea más fácil.

g) Apoyo a las funciones de trabajo del proceso. La tecnología de información puede asesorar en diferentes formas a los trabajadores para aumentar la velocidad y mejorar la calidad. Con frecuencia, la automatización permite reducir el costo de un esfuerzo, ya que resulta menos costosa que la mano de obra.

Pese a todo esto, muchos hombres de negocios confunden las masas de datos con información; sin embargo, es importante distinguir ambas términos para desarrollar un sistema de información; que permita examinar y recuperar los datos provenientes del ambiente, y que además los filtre, organice, y seleccione

para presentarlos como información proporcionando de esta forma el medio necesario para tomar decisiones adecuadas.

Con el advenimiento de las computadoras de gran velocidad y capacidad de almacenamiento ha cobrado gran auge la aplicación de la computadora en las actividades administrativas. Realizándose tres cambios básicos en las empresas progresistas:

1. La administración se ha ido orientando a los sistemas y se ha vuelto más refinada en las técnicas gerenciales.

2. La información se planea y se pone al servicio de los gerentes según la vayan necesitando.

3. Un sistema de información vincula la planeación y control, ejercido por los gerentes con los sistemas operacionales de implantación.

El resultado combinado de los conceptos anteriores es el surgimiento de los sistema de información, cuya finalidad es hacer que el proceso de administración deje de manejar información fragmentada, conjeturas inspiradas en la intuición y solución de problemas aislados para alcanzar el nivel de conocimientos basados en los sistemas, un refinado procesamientos de datos que ayuden en la solucion de problemas y a la toma de decisiones.

A continuación se ilustran algunas situaciones en las cuales la tecnología de información ha causado un gran impacto para mejorar los procesos dentro de las organizaciones, algunas de ellas ya familiares y otras relativamente nuevas.

☑ Bases de datos compartidas

Cuando la información se capta en papel y se almacena en una carpeta, solamente una persona la puede usar a la vez. Sacar copias y distribuirlas no es siempre factible, y en todo caso, lleva a la producción de múltiples versiones que tal vez no coinciden entre sí. En consecuencia, el trabajo que necesita esta información tiende a estructurarse secuencialmente. la tecnología de base de datos cambia esta regla, les permite a muchas personas usar la información simultáneamente. Haciendo que un documentos exista en diversos lugares al mismo tiempo, esta tecnología libera a un proceso de las limitaciones artificiales de la secuencia.

☑ Sistemas expertos

A principios de los años 80, cuando la tecnología de sistemas expertos apareció en las pantallas de radar de algunas de las compañías en telecomunicaciones, muchos vieron su utilidad de manera directa y simplista, la explotarían para automatizar el trabajo de expertos muy especializados, capturando la habilidad

de éstos en el software de una computadora. Ésta era una idea sumamente necia por varias razones: la tecnología realmente no daba para tanto, se tenían que conservar a los expertos, para que continuarán aprendiendo y avanzando en su campo; y no se veía muy claro cómo gente tan inteligente iba a participar compartiendo todo su conocimiento con una computadora destinada a reemplazarla.

Sin embargo, con el tiempo, organizaciones más sofisticadas han aprendido que el valor real de la tecnología de sistemas expertos está en que les permite a individuos relativamente no calificados operar casi al nivel de expertos altamente calificados, ya que pueden realizar el trabajo de muchos especialistas, y este hecho tiene profundas consecuencias en cuanto a la forma en que se puede estructurar el trabajo.

☑ Redes de telecomunicaciones

Generalmente, hay que tratar a las fábricas, a las instalaciones de servicios y a las oficinas de ventas situadas lejos de la oficina central como organizaciones separadas, descentralizadas y autónomas para que funcionen eficientemente. ¿Por qué?, Porque si toda cuestión que surge en las sucursales tuviera que consultarse con la oficina central para obtener la solución, sería poco lo que se podría realizar, y aún ese poco sería tardío.

Cuando las empresas se valen de viejas tecnologías (el servicio de correo, el teléfono o incluso el correo expreso) para mandar la información de un punto a otro, tienen que sacrificar el control de la administración central a fin de obtener flexibilidad en las operaciones de las sucursales.

En cambio, las nuevas tecnologías liberan a las empresas de esta necesidad de tolerancia. Redes de comunicación de alta amplitud de banda permiten a la oficina central disponer de la misma información que tienen las sucursales y ver los datos que éstas usan (y viceversa) en tiempo real. Con esta capacidad compartida, toda sucursal puede verdaderamente ser parte de la oficina central y ésta a su vez puede formar parte de toda sucursal. La informática, usada con buena imaginación, elimina la necesidad de sucursales separadas y completamente informadas, con sus propias áreas de soporte.

☑ Instrumentos de apoyo a decisiones (bases de datos, software de modelos)

Actualmente las compañías dicen que se dan cuenta de que los trabajadores de primera línea tienen que estar facultados para tomar sus propias decisiones, pero esto no puede lograrse simplemente dándoles a los empleados autorización para tomarlas. Necesitan también disponer de los instrumentos adecuados.

La moderna tecnologia de bases de datos permite hacer ampliamente accesible la información que anteriormente sólo estaba a la disposición de la administración. Cuando la información accesible se combina con análisis y herramientas de simulación fáciles de utilizar, los trabajadores de primera linea, debidamente capacitados, tienen en su mano instrumentos refinados para la toma de decisiones.

☑ Radiocomunicación y computadoras portátiles

Con comunicación de información por medio de banda ancha y computadoras portátiles, el personal cuyas actividades se desarrollan fuera de la empresa, de cualquier ocupación que sea, puede solicitar, ver, manipular, usar y transmitir datos casi a cualquier parte sin tener que acudir a la oficina.

La radiocomunicación de datos se vale de una tecnologia parecida a la que se usa en los teléfonos celulares, con la importante diferencia de que le permite al usuario enviar datos además de la voz. Con terminales y computadoras, los empleados pueden comunicarse con fuentes de información donde quiera que estén.

☑ Videodisco interactivo

Algunas empresas han empezado a utilizar videodiscos interactivos que permiten ver un segmento de video en una pantalla de computadora y luego hacer preguntas o contestarlas en la pantalla. Esta tecnologia se aplicó inicialmente en la capacitación, pero su poder potencial va más allá.

Por ejemplo, varios minoristas están experimentando con video interactivo para reforzar su personal de ventas. En estas tiendas los clientes escogen un producto mediante un catálogo, ven una presentación por video sobre él, hacen preguntas y luego lo compran con una tarjeta de crédito (sin intervención humana). Los bancos han empezado a usar video interactivo para explicarles a los clientes sus servicios, que se hacen cada vez más complejos; y los clientes pueden pedirle a la máquina aclaraciones sobre puntos que no entienden.

☑ Identificación automática y tecnologia de rastreo

En combinación con la radiocomunicación de datos, la tecnologia de identificación automática "permite" que las cosas (por ejemplo, los camiones) le digan a uno dónde están. No hay necesidad de buscarlos, y si uno quiere que vayan a otra parte, reciben la orden instantáneamente. Ya no hay que esperar a que el conductor llegue a la siguiente parada de camiones y pueda telefonear al despachador. Una empresa que sabe en tiempo real dónde están sus camiones (o sus vagones de ferrocarril o sus técnicos de servicio) no necesita tantos de ellos. No necesita tanta duplicidad de personal, de equipos y de

materiales para compensar las demoras inherentes a localizar y volver a encaminar cosas y gente en tránsito.

Con estos ejemplos se observa claramente que los nuevos avances de la tecnología rompen con las reglas de cómo se realizan o realizaban los negocios. Por consiguiente, explotar el potencial de la tecnología de información para cambiar los procesos que lleven a tener éxito en una época de cambio constante. Las compañías que mejor reconozcan y apliquen el potencial de la nueva tecnología gozarán de una ventaja continua y creciente sobre sus competidores.

1.6 Sistemas de información

Estamos en una época cambiante, en la cual la información juega un papel muy importante para el desarrollo de las actividades de una organización y sobre todo de su crecimiento. Por ello, actualmente todas las organizaciones cuentan con alguna clase de sistemas de información, que satisface diversas necesidades, según sea el caso y según lo requieran los usuarios. Sin embargo, tanto para las organizaciones como para la gente, el verdadero reto, radica no en adquirir la tecnología de sistemas de información, sino en saber administrarla y desarrollarla para mejorar su nivel productivo. El interés debe estar no en los sistemas de información para el trabajo administrativo, sino en el uso eficaz de los sistemas de información en la propia administración de los procesos.

Un sistema de información es un conjunto de personas, datos y procedimientos que funcionan en conjunto, que buscan un objetivo común para apoyar las actividades de la organización. Estas incluyen las operaciones diarias de la organización, la comunicación de datos e informes, la administración de las actividades y la toma de decisiones.[12] Un sistema de información ejecuta tres actividades generales:

1. Recibe datos de fuentes internas y externas como datos de entrada.

2. Actúa sobre los datos para producir información.

3. El sistema produce la información para el usuario.

De acuerdo con James A.Senn[13] las organizaciones necesitan sistemas de información por siete razones, que son las siguientes:

- La explosión de la información. La humanidad se halla en medio de una explosión de la información, entender el origen de la explosión o expansión

[12] Senn, J., *Sistemas de información para la administración*, págs. 6-10
[13] *Ídem.*

informativa ayudará a situar en perspectiva la trascendencia de la información administrativa para analizar los sistemas de información. La mayoría de los trabajadores en la actualidad son empleados del conocimiento, -es decir, que pasan el tiempo creando, distribuyendo o utilizando información-. Los administradores, como la mayoría de las personas, reciben detalles y hechos en forma continua. Lo que más se necesita es información que esté a la altura de las tareas que se realizan o las decisiones que se toman.

- El ritmo rápido del cambio. Los administradores descubren diariamente que el cambio es lo único constante. La mayoría de la gente está consciente del cambio. Nuevos métodos de producción se vuelven obsoletos a las líneas de montaje, fuentes de energía renovables, cambios sociales radicales, derribo de burocracias ancestrales, políticas económicas mundiales y concentraciones cambiantes de riqueza y poder.

- La creciente complejidad de la administración. Debido en parte al ritmo de vida de una organización, y en parte al alcance y dimensión de las tareas administrativas, el trabajo de la gerencia está creciendo en complejidad. Las organizaciones mismas crean nuevas tensiones a medida que su tamaño aumenta y surgen nuevas formas de estructurar la organización.

- La interdependencia de las unidades de la organización. Los directores administrativos utilizan la información para comunicarse entre sí y entre los empleados. Deben transmitir información a los demás y cerciorarse que los departamentos y las unidades de las que dependen estén progresando de acuerdo con los objetivos planeados. La información es el ingrediente que mantiene unidos a los componentes del sistema organizacional.

- El mejoramiento de la productividad. La productividad es la aptitud para incrementar el rendimiento de un proceso. En las organizaciones esto se relaciona con los procesos, en la fabricación de artículos o en el manejo de ventas a los clientes, y con la habilidad de los administradores para dirigir un mayor número de actividades. Los sistemas de información computarizados, desarrollados y utilizados adecuadamente pueden mejorar la productividad con la cual se ejecutan las transacciones. Igualmente trascendente es la posibilidad de reducir errores y aumentar la precisión.

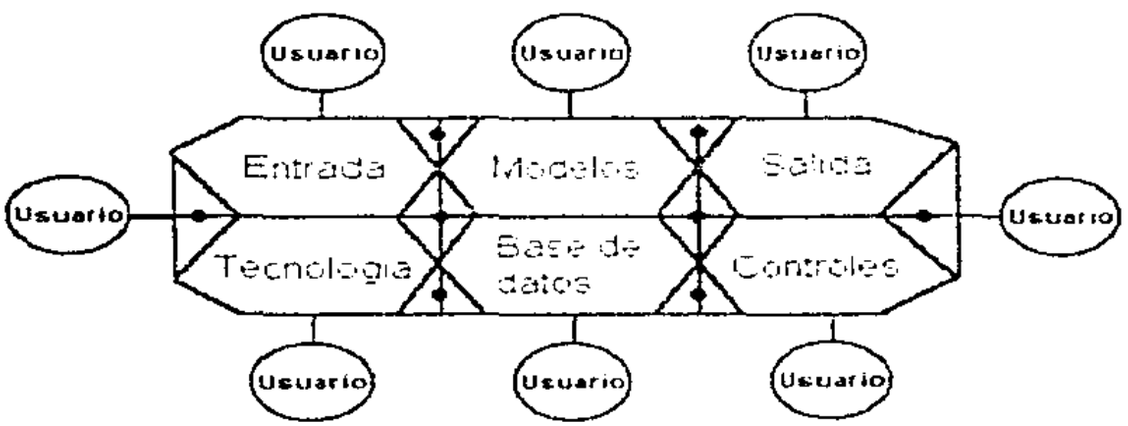
- La disponibilidad de las computadoras para los usuarios finales. Los sistemas de información computarizados son accesibles a una gran variedad de usuarios. finales, personas que utilizan las computadoras pero que no son analistas de sistemas, programadores u otros profesionales de informática. Con una computadora personal se puede manejar información contable y administrativa para probar el impacto de estrategias alternas así como evaluar el motivo de los resultados actuales de la organización; resumir grandes volúmenes de datos en una presentación gráficas; transmitir y recibir registros

de información que atraviesan toda una región en segundos; elaborar informes, propuestas y correspondencia.

- El reconocimiento de la información como un recurso. La información es reconocida como un recurso. Tiene valor porque influye en la manera como opera la organización. Carecer de información vital puede ocasionar que los administradores cometan errores, pierdan oportunidades y se enfrenten a grandes problemas de rendimiento. Los sistemas de información incrementan la capacidad de los administradores, trabajadores, y hacen posible lograr nuevos niveles de eficacia y eficiencia

A todo esto, la computadora proporciona una analogía abstracta (un modelo) del mundo físico, que permite la sustitución de recursos físicos para el tratamiento de información. En la medida en que el costo del tratamiento de información continúe disminuyendo con respecto a otros recursos, continuarán apareciendo nuevas oportunidades atractivas para tal sustitución.

Dado que un sistema de información es un conjunto sistemático y formal de componentes, capaz de realizar operaciones de procesamiento de datos, sin importar las organizaciones a las que sirven o la forma en que se desarrollan y diseñan, todos los sistemas de información están compuestos de los siguientes componentes estructurales: entradas, modelos, salidas, tecnología, base de datos y controles, cuya interacción se muestra en la siguiente figura.



Como se mostró en la figura anterior, se puede apreciar que cada uno de éstos componentes interactúan con el usuario, que es quién influye de manera directa en el desarrollo del sistema (como se explicará en el siguiente capítulo).

El qué tan bien se combinen estos componentes y el tipo de sistema de información que resulte depende del diseñador, que es quien desarrolla los sistemas. La comprensión de estos componentes estructurales, sus relaciones y acoplamientos y su contenido lógico y físico, proporcionan los conocimientos básicos para describir, desarrollar y diseñar sistemas de información. La descripción de éstos componentes se presenta a continuación.

Bloque de entrada

La entrada representa a todos los datos, texto, voz e imágenes que entran al sistema de información y los métodos y los medios por los cuales se capturan e introducen; generalmente compuesta de transacciones, solicitudes, consultas, instrucciones y mensajes. La entrada sigue un protocolo y un formato para que el contenido, la identificación, la autorización, el arreglo y el procesamiento sean adecuados. La introducción puede hacerse mediante escritura manual, formas en papel, teclado, ratón, voz, sensores táctiles, códigos ópticos y magnéticos.

En la actualidad, los medios más comunes para la introducción de transacciones y texto son las lectoras de código de barras y láser y el teclado. Una de las principales tendencias en la siguiente década será hacia los sistemas de reconocimiento de voz y escritura manual, tanto fijos como portátiles. También una diversidad de dispositivos con entrada eficiente, como las pantallas sensibles al tacto, que responden directamente a la presión de los dedos.

Bloque de modelos

Consta de modelos lógico-matemáticos que manipulan de diversas formas la entrada y los datos almacenados, para producir los resultados deseados o salida. Un modelo lógico-matemático puede combinar ciertos elementos de datos para proporcionar una respuesta adecuada a una consulta o puede reducir o agregar volúmenes de datos para obtener un reporte conciso. El componente de modelos también contiene una descripción de algunas de las técnicas de modelado más populares empleadas por los analistas de sistemas para diseñar y documentar las especificaciones de los sistemas. Estas técnicas incluyen tablas y árboles de decisión, diagramas de flujo tradicionales, diagramas Nassi-Shneiderman, HIPO, diagramas de estructura, diagramas Warnier-Orr y el uso de prototipos.

Bloque de salida

El resultado del sistema de información es la salida de información de calidad y documentos para todos los niveles de la gerencia y para todos los usuarios dentro y fuera de la organización. La salida es, en gran medida el componente que guía e influye en los otros componentes. Si el diseño de este componente no satisface las necesidades del usuario, entonces los otros componentes tienen poca importancia.

La salida está compuesta de elementos tales como reportes, respuestas a consultas, mensajes, órdenes, resultados de una toma de decisiones programadas, escenarios y simulaciones y reglas de decisión. La calidad de esta salida se basa en su exactitud, oportunidad y relevancia. Además, esta salida debe tratarse en función de su destino, uso, frecuencia de uso y seguridad. En cuanto al medio de transmisión, la salida se puede producir en pantallas, impresoras, dispositivos de audio o microfilm.

En la actualidad, las personas están eligiendo otras formas de salida que vayan más de acuerdo a sus gustos, como gráficas, video y audio. Debido a que la mayoría de los trabajadores pasan entre 50 y 90% de su tiempo manejando información y comunicándose, están buscando una gama más amplia de medios que se aproximen a los métodos humanos naturales de emitir información y de comunicarse con el sistema y entre sí.

Bloques de tecnología

La tecnología es la herramienta de trabajo en el desarrollo de sistemas de información. Captura la entrada, activa los modelos, almacena y accesa datos, produce y transmite salida, y ayuda a controlar todo el sistema. La tecnología consta de tres componentes principales: la computadora, las telecomunicaciones y el software. Las telecomunicaciones comprenden el empleo de medios electrónicos y de transmisión de luz para la comunicación entre nodos a lo largo de una cierta distancia. El software corresponde a los programas que hacen que funcione el hardware de la computadora y le dan instrucciones sobre la forma de procesar los modelos. El hardware está compuesto de una variedad de dispositivos que proporcionan el soporte físico para los componentes estructurales.

Bloque de base de datos

La base de datos es el lugar en donde se almacenan todos los datos necesarios para atender las necesidades de todos los usuarios. De igual manera los datos pueden ser una combinación de voz, imágenes, texto y números. La base de datos es considerada desde dos puntos de vista, el físico y el lógico. La base de datos física está compuesta de los medios de almacenamiento, como las cintas, discos, disquetes, tarjetas magnéticas, chips y microfilms, ésta es la forma en que los datos se almacenan realmente; sin embargo, otro problema, quizá el más importante, es cómo buscar, asociar y recuperar los datos almacenados para satisfacer necesidades específicas de información. Esto, obviamente es el lado lógico de la base de datos y si está estructurada correctamente, asegura la recuperación oportuna, relevante y exacta de la información. También tiene que ver con el componente de software del sistema e incluye técnicas lógicas y asociativas de datos como índices, directorios, llaves, y asociaciones.

Bloque de controles

Todos los sistemas de información están sujetos a una diversidad de peligros y amenazas, como desastres naturales, fallas de los sistemas, errores y omisiones, deficiencias y mutilaciones maliciosas. En muchos casos, los peores abusos del sistema provienen de procedimientos operacionales inadecuados, empleados incompetentes y una pobre administración. Algunos de los controles que necesitan diseñarse en el sistema para asegurar su protección, integridad y operación uniforme son la instalación de un sistema de administración de registros, la creación de un plan de contingencias, la preparación de una documentación completa y actualizada, la aplicación de monitores de hardware y software, el establecimiento de sistemas de respaldo y almacenamiento fuera de las instalaciones, la instalación de sistemas ininterrumpidos de energía y sistemas contra incendio, el empleo de adecuados procedimientos de programación y controles y la aplicación de una diversidad de procedimientos de seguridad, dispositivos y controles de acceso.

Las personas que tienen la responsabilidad del desarrollo y diseño de sistemas de información están en gran medida influenciados y limitados por un número de fuerzas del diseño. Deben determinar cuáles son las fuerzas del diseño, cómo afectan a sus proyectos de diseño y guiarse por ellas, incorporando al mismo tiempo creatividad e innovación en su trabajo, ya que, además deben tomar en cuenta que entre mejor sea la interfaz entre el usuario y el sistema, sin obstrucción, interferencia externa o dependencia de intermediarios, mejor será el flujo de información. En los siguientes capítulos se hablará de las herramientas que el analista emplea para el desarrollo de sistemas.

De acuerdo a la anterior descripción de un sistema de información, se puede observar que éstos son susceptibles de ser aplicados en otros ámbitos de las actividades humanas como lo es en el sector educativo y/o en el aspecto cultural. El desarrollo de sistemas de información va más allá de programas que son de uso específico para los negocios, existen otro tipo de programas, como son los programas de aplicación, desarrollados para un uso específico y que son hechos para controlar el procesamiento de una tarea determinada (como un sistema de nómina, de contabilidad, sistema de alarmas, sistemas que controlan el alumbrado eléctrico, etc.).

Aunque muchos de los programas de aplicación son creados especialmente para tareas específicas, hoy en dia pueden realizarse un sin número de actividades de proceso de datos en el hogar, escuela y lugar de trabajo, usando programas generalizados, de los cuales los usuarios pueden elegir de entre una gran diversidad de paquetes de aplicación, que les ayuden a trabajar, cómodamente con las computadoras para producir los resultados necesitados.

CAPÍTULO 2

ANÁLISIS Y DISEÑO DE SISTEMAS

2.1 Introducción

Los sistemas de información se desarrollan con diferentes propósitos dependiendo de las necesidades de la organización o del usuario final (en el caso de aplicaciones comerciales), éstos sistemas de información computarizados generalmente son analizados y diseñados mediante la aplicación de los conceptos y técnicas del análisis y diseño de sistemas.

El análisis y el diseño de sistemas, tal como lo realizan los analistas de sistemas, pretende estudiar sistemáticamente la operación de ingreso de datos, el flujo de los mismos y la salida de la información, para que sean elaborados de la mejor manera y puedan satisfacer las necesidades para las cuales fueron creados; ya que, si se implantan sin una planeación adecuada, es muy probable que no sea satisfactorio y pronto quede en el olvido el sistema.

El análisis y diseño de sistemas permite estructurar el desarrollo e implantación de los sistemas, a través de una serie de procesos, que al ejecutarse sistemáticamente van dando vida al nuevo sistema.

2.2 Quienes desarrollan los sistemas

Una buena parte del análisis y el diseño de sistemas involucra el trabajo en colaboración con los futuros usuarios del sistema. A continuación se enlistan las diferentes personas y entidades involucradas en el desarrollo de sistemas:

- Usuarios
- Administración
- Analistas de sistemas
- Diseñadores de sistemas
- Programadores

Usuarios

El participante más importante en el desarrollo de los sistemas es alguien que el analista conoce como usuario. El usuario es aquél (o aquéllos) para quien se construye el sistema. Es la persona a la que tendrá que entrevistar, a menudo con

gran detalle, a fin de conocer las características que deberá tener el nuevo sistema para poder tener éxito.

En la mayoría de los casos, es fácil identificar al usuario: de manera característica, es aquel que formalmente solicita un sistema. En una organización pequeña, esto suele ser un procedimiento muy informal, pudiendo consistir simplemente en que el usuario llame por teléfono al analista y lo solicite. En una organización grande, el inicio de un proyecto de desarrollo de sistemas suele ser mucho más formal, por lo común, la "solicitud de consideración y estudio de sistemas", como se le suele conocer, pasa por diversos niveles de aprobación antes de que se involucre al analista de sistemas.

Sin embargo, hay un gran número de situaciones en las que no se conoce la identidad del verdadero usuario o bien en las que hay poca oportunidad de que éste interactúe con el analista. Un ejemplo muy común de esto es el de un sistema en proceso de ser desarrollado por un negocio de consultoría o por una compañía productora de software: la interacción que exista entre el cliente y la compañía pudiera llevarse a cabo a través de administradores de contratos u otras agencias administrativas, a veces con cláusulas explícitas de que el analista no puede tener comunicación directa con el usuario.

En situaciones de este tipo, hay una gran posibilidad de malos entendidos: lo que el usuario quiere que el sistema haga pudiera no serle comunicado de manera correcta al analista, y lo que éste crea que está construyendo para el usuario pudiera no serle comunicado tampoco de manera correcta, hasta que ya estuviera todo terminado, cuando ya sería demasiado tarde.

Uno de los errores más frecuentes que se cometen en el momento del análisis es suponer que todos los usuarios son iguales, la palabra "usuario", da a entender que el analista sólo tendrá que interactuar con una persona. Aun cuando es obvio que deberá intervenir más de un usuario, y que cada uno de ellos opina diferente y ve de diferente manera al sistema, así como la forma en que éste operará.

En un proyecto de análisis de sistemas se pasará una considerable cantidad de tiempo entrevistando a los usuarios para determinar los requerimientos del sistema; pero, ¿quiénes son los usuarios y a qué nivel se encuentran?, obviamente, esto dependerá del proyecto y de las políticas de la organización. Sin embargo, habitualmente se tendrá que interactuar con individuos de tres categorías de trabajo: usuarios operacionales, usuarios supervisores y usuarios ejecutivos.

Los usuarios operacionales son oficinistas, administradores y operadores que son los que más tendrán contacto diario con el nuevo sistema. Generalmente se deben tener tres aspectos en mente cuando se trabaja con este tipo de usuarios:

1. Los usuarios de este nivel se preocupan mucho por las funciones que tendrá el sistema, pero es más probable que se preocupen por los detalles de la interfaz humana. Por ejemplo ¿el teclado es igual al de la máquina de escribir?, ¿cómo aparecerán las cosas en la pantalla?, ¿cómo indicará el sistema si se ha cometido algún error?, ¿se tendrá que volver a teclear todo?, etc.

2. Los usuarios operacionales tienden a poseer un panorama específico del sistema y a menudo no están familiarizados con el panorama general; es decir, puede ser que tengan dificultad para describir cómo es que su propia actividad encaja dentro de la organización. Una consecuencia de esta situación es que el analista debe poder desarrollar modelos de sistemas descripciones globales que permitan el entendimiento de ambos panoramas.

   Las discusiones abstractas acerca de "funciones" y "tipos de datos" pueden resultar difíciles, por lo que el analista debe hablar con el usuario únicamente en términos familiares y luego traducir esta descripción física a un modelo de lo que el sistema debe hacer, independientemente de la metodología a usar para realizarlo.

3. Los usuarios supervisores son empleados supervisores que usualmente administran a un grupo de usuarios operacionales, y son con quienes se tendrá el contacto cotidiano primario, ya que son quienes definen los requerimientos y las políticas de la empresa que el sistema deberá realizar.

Los usuarios de nivel ejecutivo en general no se involucran directamente con el proyecto de desarrollo del sistema, a menos que el proyecto sea tan amplio y tan importante que tenga un impacto de primer orden en la organización

Administración

El analista puede interactuar con diversos tipos de administradores, como son:

- Administradores usuarios. Son administradores que están a cargo de varias personas en el área operacional donde se va a implantar el nuevo sistema. Por lo general son administradores de nivel medio que desean sistemas que produzcan una variedad de informes internos y de análisis.

- Administradores de informática. Son las personas encargadas del proyecto en sí, y los administradores de nivel superior encargados de la administración global y distribución de los recursos de todo el personal técnico de la organización de creación o desarrollo de sistemas.

- Administración general. Son los administradores de nivel superior que no están directamente involucrados con la organización de informática ni son de la organización usuaria.

La principal interacción entre el analista de sistemas y todos estos administradores tiene que ver con los recursos que se asignarán al proyecto. Es tarea del analista identificar y documentar los requerimientos del usuario y las limitaciones dentro de las cuales se tendrá que implantar el sistema. Generalmente, estas limitaciones son los recursos: personas, tiempo y dinero; por lo que, el analista deberá hacer un documento que especifiqué los recursos a emplear (estudio de factibilidad).

Analista de sistemas

El analista de sistemas es el personaje central en cualquier proyecto de desarrollo de sistemas desempeñando varios papeles:

- Como analista, una de las principales labores es descubrir detalles y documentar la política de una organización que pudiera existir sólo como "tradiciones" transmitidas por los usuarios.

- Como consultor, el analista debe distinguir entre síntomas, problemas del usuarios y causas, con sus conocimientos debe ayudar al usuario a explotar aplicaciones novedosas y más útiles de las computadoras así como formas nuevas de hacer sus actividades.

- Como mediador, ya que, a menudo se encuentra en medio de usuarios, administradores, programadores y otros participantes, los cuales frecuentemente están en desacuerdo entre sí y su labor es obtener un consenso de todos ellos.

- Como responsable, dado que el analista suele tener más experiencia que los programadores que laboran en el proyecto y que usualmente se le asigna el proyecto antes que a los programadores hay una tendencia a asignarle también las responsabilidades del proyecto.

Diseñadores de sistemas

El diseñador de sistemas es quien recibe los resultados del trabajo de análisis y cuya labor es transformar la solicitud de los requerimientos del usuario, en un diseño de alto nivel que servirá de base para el trabajo de los programadores.

En muchos casos, el analista y el diseñador son la misma persona o el mismo grupo unificado de personas. En caso contrario, debe existir una retroalimentación continua entre diseñador y analista, ya que el analista tiene que ofrecer suficiente información detallada al diseñador para que éste pueda desarrollar un buen diseño; el diseñador debe proveer suficiente información para que el analista pueda darse cuenta de si los requerimientos que el usuario esta solicitando son realmente posibles.

Programadores

Son quienes reciben una descripción del hardware y software que se usará para poner en práctica el sistema y llevarlo a cabo. A menudo, son quienes descubren los errores y ambigüedades en la "propuesta de requerimientos" entregada por el analista, pues durante la programación donde una reseña superficial de los requerimientos del sistema se traduce en un juego de instrucciones muy especificas y detalladas; si algo falta o esta mal o confuso, el programador tiene dos opciones: pedirle una aclaración al analista o preguntarle al usuario.

2.3 Ciclo de vida de desarrollo de sistemas

Para ser un buen analista de sistemas se requiere de métodos para llevar a cabo el sistema, uno de estos métodos es el ciclo de vida de desarrollo de sistemas (o ciclo de vida del proyecto), el cual es un conjunto de actividades de los analistas, diseñadores y usuarios, que necesita llevarse a cabo para desarrollar y poner en marcha un sistema de información. los períodos en los que se llevan a cabo estas actividades se denominan fases; donde una *fase* es un período particular del ciclo de desarrollo.

Recientemente, se ha empezado a cambiar el enfoque que se le da al desarrollo de sistemas. Cada vez son más las organizaciones, tanto grandes como pequeñas que están adoptando un ciclo de vida uniforme y único para sus proyectos. Esto a veces se conoce como el plan del proyecto, la metodología del desarrollo de sistemas o simplemente "la forma en que hacemos las cosas aquí". El manual del ciclo de vida del proyecto suele ofrecer un procedimiento común a seguir para desarrollar un sistema que puede orientar a cualquier miembro de la organización de desarrollo de sistemas. Los objetivos sobre los cuales se fundamenta la existencia del ciclo de desarrollo de sistemas, se presentan a continuación:

1. Definir las actividades a llevarse a cabo en un proyecto de desarrollo de sistemas.

2. Lograr congruencia entre la multitud de proyectos de desarrollo de sistemas en una misma organización.

3. Proporcionar puntos de control y revisión administrativos de las decisiones sobre continuar o no con el proyecto.

Las fases que integran el ciclo de desarrollo de sistemas son las siguientes:

a) Estudio preliminar
b) Fase de análisis

c) Fase de diseño
a) Fase de implementación
e) Fase de mantenimiento

Estudio preliminar

Una etapa preliminar a la fase de análisis es el estudio de factibilidad, el cual inicia cuando un usuario solicita que una o más partes de su sistema se automaticen. El estudio de factibilidad consiste en una visión general de alto nivel de todo el proyecto, destinado a contestar a un número determinado de preguntas: ¿cuál es el problema?, ¿existe una solución factible para el problema?, etc; éste tipo de estudio debe ser relativamente corto, pues su objetivo no consiste en resolver el problema, sino en obtener una idea de su complejidad y tamaño. Los principales objetivos del estudio son los siguientes:

- Identificar a los usuarios responsables y crear un campo de actividad inicial del sistema.

- Identificar las posibles deficiencias actuales en el ambiente del usuario y en el sistema actual.

- Establecer metas y objetivos para el sistema nuevo, pudiendo ser una simple lista narrativa que contenga las funciones existentes que deben reimplantarse, las nuevas que necesitan añadirse y los criterios de desempeño del nuevo sistema.

- Determinar si es factible automatizar el sistema y de ser así, sugerir escenarios aceptables, lo cual implicará hacer algunas estimaciones bastante rudimentarias y aproximadas del costo y el tiempo necesarios para construir un sistema nuevo y los beneficios que se derivaran de ello; así como el equipo de cómputo y el tipo de procesamiento a emplearse.

- Preparar el esquema que se usará para guiar el resto del proyecto, el cual incluirá toda la información que se lista anteriormente, además de identificar al administrador responsable del proyecto. También se podrían incluir los detalles del ciclo de desarrollo que se emplearán en el resto del proyecto.

Fase de análisis

El análisis es un proceso lógico, cuyo objetivo es determinar exactamente qué se debe realizar para construir el sistema, para ello se debe tener conocimiento de lo que realiza el sistema actual (ya sea manual o automatizado), lo que implica involucrarse profundamente con el funcionamiento del sistema.

Durante esta fase, el analista de sistemas trabaja con el usuario para desarrollar un modelo lógico del sistema, en donde el usuario necesita de las habilidades y conocimientos técnicos del analista y el analista necesita los conocimientos que el usuarios tiene del sistema.

La tentación en la que caen muchas personas es la de ir directamente a la realización del diseño del programa, es decir, en convertirlo prematuramente en algo físico. Se debe tener presente que el objetivo de esta fase es resolver el problema del usuario y tomar en cuenta que es la persona que mejor conoce el problema, si el analista comienza a hablar de detalles de programación, es posible que el usuario pierda el interés de la conversación y sea incapaz de contribuir al análisis; consecuentemente, se podría desarrollar un sistema que no resuelva el problema.

La utilización de una metodología estructurada podrá ayudar a evitar la mención de los detalles físicos. El uso de una metodología estructurada involucra el desarrollo de un modelo lógico del sistema, uso de herramientas como diagramas de flujo de datos, diccionarios de datos y descripciones someras de los algoritmos más importantes. Para los fines de este trabajo se utilizó el análisis estructurado de Edward Yourdon, siendo las herramientas involucradas en está metodología serán explicadas a detalle más adelante.

Fase de diseño

Una vez terminada la fase de análisis, el analista tendrá aclarado lo que ha de hacer. La fase de diseño se dedica a asignar partes de la especificación (modelo lógico) a procesadores adecuados y a labores apropiadas (tareas o procesos). Dentro de cada labor, la actividad de diseño se dedica a la creación de una jerarquía apropiada de módulos de programas y de interfaces entre ellos para implantar la especificación creada en la fase de análisis.

Fase de implementación

Durante la fase de implementación, el sistema se llega a crear de forma física, los programas se codifican, depuran y documentan. El hardware nuevo se selecciona, ordena e instala. En esta etapa se deberán desarrollar los procedimientos que utilizará el sistema, así como los procedimientos de seguridad y auditoria necesarios. Posiblemente también sea necesario establecer un plan para realizar la validación y verificación del sistema.

La fase de implementación termina con la realización de una prueba formal del sistema, comprobando todos los componentes y procedimientos.

Fase de mantenimiento

Después de la implementación, comienza la fase de mantenimiento. El objetivo de esta fase es procurar que el sistema mantenga un nivel aceptable de operatividad. Muchas veces no se detectan todos los errores en las tareas de validación y verificación, por lo tanto, deben corregirse dentro de la fase de mantenimiento. Muchas veces, los algoritmos y parámetros del sistema deben sufrir modificaciones y esto significa que los programas deben actualizarse.

2.4 Metodología de desarrollo de sistemas

Es la implantación de un ciclo de vida de desarrollo de sistemas que sirve como marco de referencia para el desarrollo de software en donde se indique qué actividades hay que realizar y cuándo se deben llevar a cabo, todo ello, bajo un enfoque filosófico.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software, que van indicando paso a paso todas las actividades a realizar para lograr el producto deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben hacer en la mismas. Además, detallan la información necesaria para comenzar cada actividad.

Inicialmente, las metodologías estaban enfocadas a ciertas fases del ciclo de vida: fase de análisis, diseño, implantación, etc.; como por ejemplo, la metodología de análisis estructurado, diseño estructurado, metodología de programación estructurada, entre otras.

Actualmente, las metodologías de desarrollo de software van hacia el enfoque completo del ciclo de vida, mejorando la integración de las distintas fases del ciclo de vida de desarrollo.

Importancia de la metodología

Aunado a todo lo anterior, existen un gran número de factores que repercuten en la persona que trabaja dentro de un entorno de desarrollo de software. Los cambios en el sistema operativo disponible, el lenguaje de programación, la organización del proyecto, el formato de las especificaciones, el tipo de terminal, o los estándares establecidos para los diferentes aspectos del ciclo de vida de un proyecto pueden repercutir tanto en el trabajador como en la cantidad de trabajo que puede realizar.

El objetivo en el desarrollo de software es establecer un entorno que no sólo mejore la productividad del que desarrolla el software, sino que también genere la creación de mejores productos. El corazón de este entorno es una metodología de desarrollo de software que establezca los pasos a dar, con sus procedimientos y técnicas, para la producción de software. La metodología, generalmente, consta de una secuencia de pasos combinados, que son:

- procedimientos de gestión,
- métodos técnicos y
- soporte automatizado

En donde los procedimientos de gestión determinan la naturaleza del soporte automatizado que se va a proveer, en términos de equipo de cómputo que se va a emplear y de los lenguajes y herramientas que soportarán el desarrollo de software. Los procedimientos de gestión también coordinan y guían las técnicas que los informáticos emplean.

La gestión se realiza con más efectividad si se usan técnicas que aporten productos intermedios, tales como documentos de especificación y de diseño. El soporte automatizado mejora la productividad del equipo de desarrollo, al mismo tiempo que valida subproductos que se van obteniendo de forma automática, evitando así el error humano y consecuentemente incrementando la calidad final.

La organización de desarrollo de software debe seleccionar entre un vasto número de posibilidades y combinaciones de métodos de gestión, técnicas de desarrollo y soporte automatizado para crear y desarrollar la metodología de desarrollo de software. Existe asimismo la posibilidad de analizar y evaluar las distintas metodologías ya desarrolladas para adoptar aquéllas que se adapten mejor a las necesidades planteadas.

La metodología a usar también esta influenciada por muchas otras consideraciones entres ellas:

- el tamaño y estructura de la organización que desarrolla software,
- el tipo de aplicaciones que se van a desarrollar,
- el número de veces que se van a usar,
- la importancia de la aplicación a desarrollar y
- las necesidades planificadas para el mantenimiento.

Debido al gran número de posibilidades y grandes variaciones en uso y aplicaciones del software, es imposible pensar que distintas organizaciones usan la misma metodología de desarrollo de software. Incluso grupos que sigan una metodología estándar para el desarrollo de software podrían modificarla para adoptarla a su organización y o sus proyectos. Por otra parte, las metodologías

cambian constantemente para reflejar nuevos desarrollos en hardware y software, así como nuevas estructuras de dirección y de personal.

Antonio de Amescua[1] señala que los mejores informáticos necesitan un entorno disciplinado y estructurado en el cual puedan realizar un trabajo en equipo, para obtener productos de con alta calidad, y que el uso de una metodología no va a coartar la creatividad de los buenos profesionales, sino todo lo contrario, al liberarlos de los posible errores que otros producen.

2.5 Cambios en el análisis de sistemas

Hasta fines de los años 70, la gran mayoría de los programas de desarrollo de sistemas empezaban con la descripción de los requerimientos del usuario, es decir, el analista escribía lo que entendía de los requerimientos del usuario en un enorme documento que consistía inicialmente en una narración del sistema actual. Los primeros autores de textos de análisis estructurado, sobre todo Tom de Marco, Gane & Sarson y Weinberg señalaron que estas especificaciones se veían afectadas por diversos problemas, entre los cuales destaca el hecho de que:

- Eran hechos en un solo bloque, es decir, había que leer completamente la especificación de principio a fin para poder entenderla. Esta es una falla importante pues existen muchas situaciones en las que el analista quisiera leer y comprender una parte de la especificación sin tener necesariamente que leer las demás.

- Eran redundantes, en ocasiones se repetía la misma información en diversas partes del documento. El problema con esto es que si se cambia cualquier aspecto de los requerimientos del usuario durante la fase de análisis el cambio debe reflejarse en diversas partes del documento.

- Existía ambigüedad, el reporte detallado de los requerimientos podía ser interpretado de diferente manera por el usuario, el analista, el diseñador y el programador.

- En ocasiones resultaban imposibles de mantener, debido a que la especificación era casi obsoleta para cuando llegaba el final del proceso de desarrollo del sistema.

Mientras se debatían todos estos problemas, ya se estaba adoptando un conjunto complementario de ideas en el área de programación y diseño. Estas ideas, conocidas como diseño y programación estructurados, prometían grandes

[1] De Amescua, S. Antonio. *Ingeniería del software de gestión*, pág. 27.

mejoras en la organización, codificación, prueba y mantenimiento de los programas de computadora.

Como resultado, ha habido un movimiento gradual tendiente a hacer especificaciones funcionales que sean de tipo:

- Gráfico: compuestas de una variedad de diagramas, apoyadas con un poco de texto, que sirva de material de referencia.
- Particionando: de manera que permita leer independientemente porciones individuales de la especificación.
- Poco redundante: es decir, que permita que los cambios en los requerimientos del usuario puedan incorporarse normalmente en sólo una parte de la especificación.

Este enfoque, al que por lo general se conoce como ***análisis estructurado***, se utiliza ahora en la mayoría de las organizaciones de desarrollo de sistemas orientadas a los negocios, al igual que en un gran número de las orientadas hacia la ingeniería y en el desarrollo de aplicaciones comerciales.

2.6 Análisis y diseño estructurado

Al igual que muchas de las contribuciones importantes al diseño de software, el análisis estructurado no fue introducido en un sólo artículo o libro que incluyera un tratamiento completo del tema. Los primeros trabajos sobre modelos de análisis aparecieron a finales de los 60 y principios de los 70, pero la primera aparición del enfoque de análisis estructurado fue como complemento del diseño estructurado, debido a que los investigadores necesitaban una notación gráfica para representar los datos y los procesos que los transforman, los cuales quedaron finalmente establecidos en una arquitectura de diseño.

El término "análisis estructurado" fue popularizado por DeMarco (1979), quien presentó una serie de símbolos gráficos que permitían a un analista crear modelos de flujo de información; sugirió modelos heurísticos para utilizar esos símbolos; sugirió también el uso de un diccionario de datos y descripción de procesamientos como complemento a los modelos de flujo de información.

En los años siguientes Page-Jones (1980) y Gane & Sarson (1982) entre otros propusieron variaciones del enfoque de análisis estructurado. En todos los casos, el método se centraba en aplicaciones de sistemas de información y no proporcionaban una notación adecuada para los aspectos de control y de comportamiento de los problemas de ingeniería de tiempo real. Finalmente, es Edward Yourdon quien da los últimos retoques a esta metodología y compra los

derechos de autor a Tom DeMarco para continuar desarrollando el análisis estructurado, siendo él quien a la fecha a popularizado entre los analistas de sistemas el uso de ésta metodología.

La metodología de Yourdon esta compuesta por dos fases:

- ☑ **Modelo ambiental.** Es la fase en la cual se describen las relaciones con el medio ambiente, incluyendo el origen y destino de la información usada por el sistema, así como la definición de los eventos que requieren una respuesta.

- ☑ **Modelo de comportamiento.** Describe los procesos que son parte del sistema, incluyendo información usada y/o producida en cada proceso así como la función del mismo.

Modelo ambiental

Para el analista, la labor más difícil en la especificación de un sistema es determinar ***qué es parte del sistema*** y qué ***no***, de esta manera el primer modelo importante que se debe desarrollar como analista es uno que defina las interfaces entre el sistema y el resto del ambiente; este modelo se conoce como ***modelo ambiental***, cuya función es determinar qué está en el interior del sistema y que en el exterior, así como definir las interfaces entre el sistema y el ambiente, debido a que es necesario saber qué información entra al sistema desde el ambiente exterior y que información produce como salida al ambiente externo.

La característica esencial de esta metodología es que su propósito específico es producir salidas como respuesta a algún ***acontecimiento*** o ***estímulo*** en el ambiente, así como de identificar los acontecimientos que ocurren en el ambiente a los cuales debe responder el sistema.

Generalmente el usuario tiene idea de la frontera general entre el sistema y el ambiente y esta frontera debe ser también conocida por el analista para poder delimitar el alcance del sistema; sin embargo, existe un "área gris" que está abierta a negociaciones, es decir, un área sobre la cual el usuario:

a) no está seguro,
b) no había pensado,
c) tiene algunas ideas preconcebidas y
   esta dispuesto a reflexionarlas,
d) todas las anteriores.

En la siguiente figura se ejemplifica la manera gráfica para representar esta situación, la cual debe quedar aclarada con los usuarios, ya que, de lo contrario, se estará trabajando en un sistema en el cual se desconoce su alcance y al mismo tiempo el punto en el cual éste deja de interactuar con el ambiente que

le rodea, teniendo posiblemente como consecuencia un sistema incompleto; de aquí la importancia de dedicar bastante tiempo y tener suficiente participación del usuario en la elección de los limites apropiados del sistema.

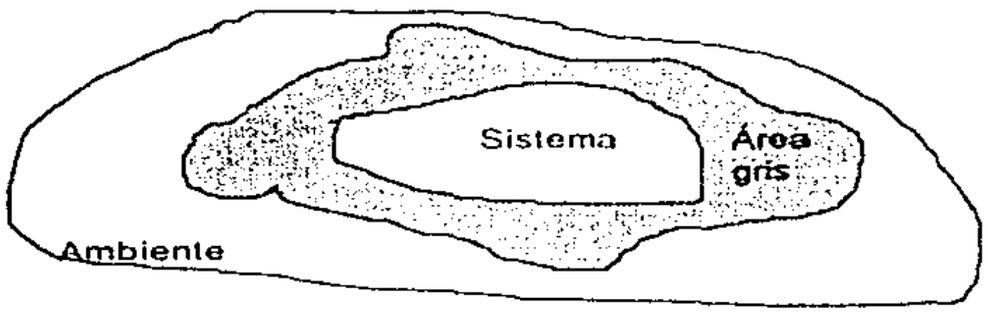


Figura 1. Frontera entre el sistema y el ambiente

El modelo ambiental consta de tres componentes para definir el ambiente, los cuales son.

- Propósito del sistema
- Diagrama de contexto
- Lista de eventos

Propósito del sistema

El primer componente del modelo ambiental es una declaración textual breve y concisa del objetivo del sistema, dirigida a quienes estén directamente involucrados con el desarrollo del sistema.

La declaración del propósito puede constar de una, dos o más frases; pero sin ir más allá de un párrafo, ya que la intención no es proporcionar una descripción completa y detallada del sistema, aun cuando la declaración del propósito sea deliberadamente vaga en cuanto a muchos detalles.

Diagrama de contexto

La siguiente fase del modelo ambiental empieza a contestar algunas de las preguntas que surgen a raíz de la estructuración del propósito. En el diagrama de

contexto el funcionamiento general del sistema se representa a través de una única transformación de información, que aparece en la siguiente figura como una ***burbuja***, mientras que las entidades externas son representadas por ***rectángulos***, los cuales originan una o más entradas, que aparecen como flechas.

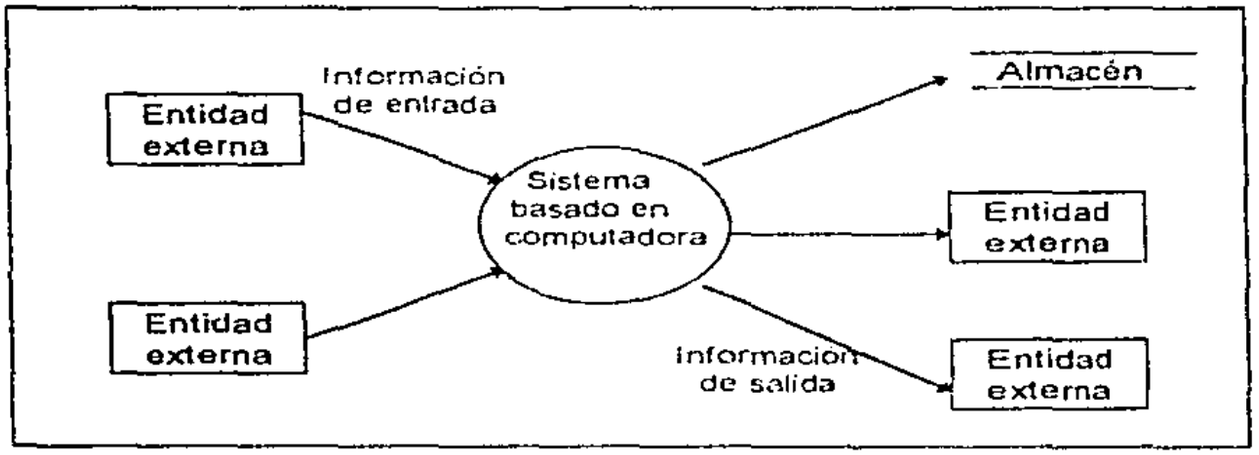


Figura 2. Diagrama de contexto

A continuación se explican cada uno de los componentes empleados por el diagrama de contexto:

- Las personas, organización y sistemas con los que se comunica el sistema se conocen como ***terminadores***, (rectángulos).

- Los datos que el sistema recibe del mundo exterior y que son procesados de alguna forma; así como, los datos que produce y que se envían al exterior, son los flujos de datos, representados por flechas e identificado por un nombre en singular.

- Existen los almacenes de datos que el sistema comparte con los terminadores. Estos almacenes de datos se crean fuera del sistema para su uso o bien son creados en él y usados afuera.

En el diagrama de contexto, el proceso es representado por una sola burbuja que contiene el nombre completo del sistema o un acrónimo convenido. Los terminadores son representados con rectángulos y se comunican directamente con el sistema a través de flujos de datos o a través de almacenes externos, como se muestra en la siguiente figura:

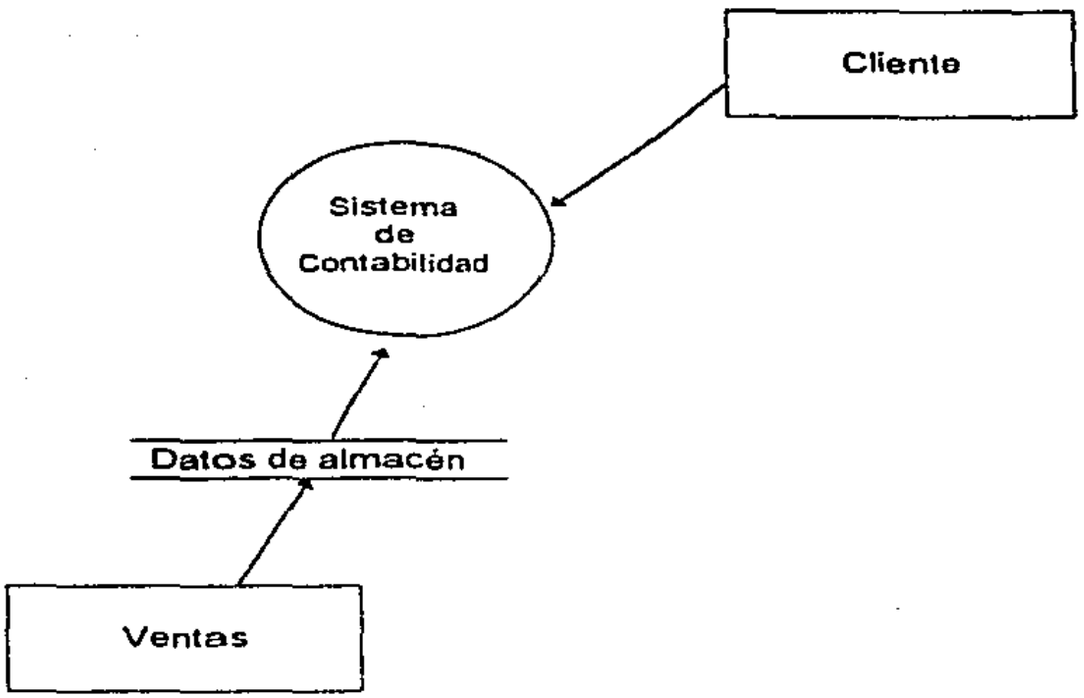


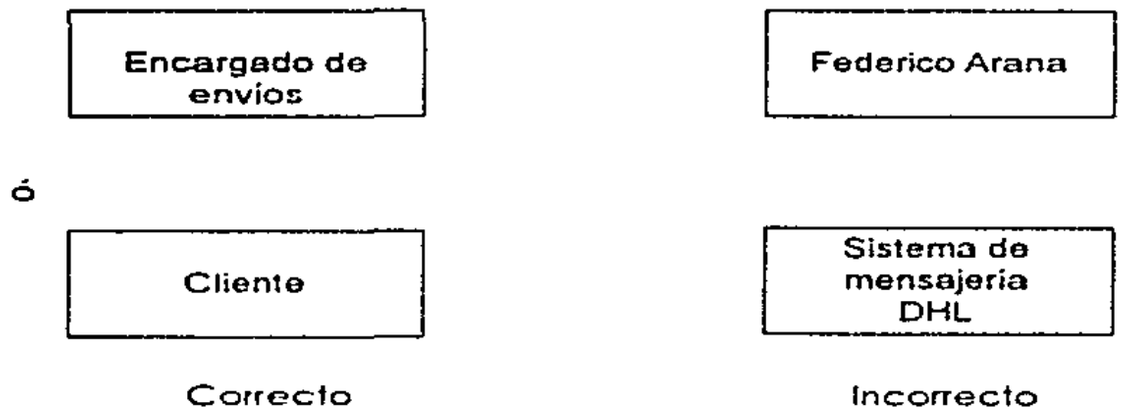
Figura 3 Comunicación entre terminador y sistema

No es correcto establecer la comunicación entre terminadores, aún cuando en realidad éstos se comuniquen entre sí, esto se debe a que por definición los terminadores son *externos* al sistema, por lo que la naturaleza y contenido de las interacciones terminador-terminador son irrelevantes para el sistema. Si durante la discusión con los usuarios se encuentra que es esencial saber cuándo, por qué o cómo se comunican entre sí, entonces los terminadores son parte del sistema y deben incluirse dentro de la burbuja de proceso del diagrama de contexto.

Caracteristicas de los terminadores:

- Algunos terminadores tienen un buen número de entradas y salidas, para evitar un diagrama innecesariamente lleno conviene dibujar el terminador más de una vez; representando los terminadores repetidos con una diagonal, como se muestra a continuación:

- Cuando el terminador es una persona o una organización es preferible indicar el rol que desempeña en vez de su identidad.



Existen reglas mínimas que debe de cumplir un diagrama de contexto, siendo estas:

- Debe haber un solo proceso.
- Al menos debe existir un flujo de salida.
- Debe existir un flujo de datos entre el terminador y los procesos.
- Los almacenamientos deben ser accesados tanto por el sistema como por los terminadores.
- Los flujos de datos nombrados deben especificar el mismo nivel de detalle.

Lista de eventos

Los eventos son sucesos que ocurren en el medio ambiente para las cuales el sistema tiene que responder en un tiempo determinado. Identificar los eventos permite conocer los procesos del sistema tomando como base los efectos que los eventos tienen en los datos, así como su organización.

Existen dos tipos de eventos, los de **flujo**, aquellos que proporcionan datos al sistema y al ser captados hacen que se ejecute una acción y los **temporales** que son requeridos en el sistema, pero tienen ocurrencia con una periodicidad preestablecida por las políticas del negocio. Los eventos temporales no están asociados con una entrada, generalmente causan una salida en el diagrama de contexto.

Las reglas de consistencia entre la lista de eventos y el diagrama de contexto son:

- Cada estímulo es una entrada del proceso de contexto.
- Cada flujo de salida del proceso de contexto debe ser definido como una salida del sistema de al menos un evento.
- Cada flujo de entrada del proceso de contexto debe ser definido como un estímulo para al menos un evento de flujo.

Diccionario de datos (DD)

El diccionario de datos es un conjunto de definiciones de todos los datos que aparecen tanto en el diagrama de contexto como en los DFD (Diagramas de Flujo de Datos[2]), pudiendo ser éstos, datos almacenados o flujos de datos. En términos generales un DD es un conjunto de datos sobre otros datos, cuya idea básica es proporcionar toda la información necesaria sobre la definición, estructura y utilización de los datos de un programa.

La regla básica para el diccionario de datos es que los datos definidos deben ir en orden alfabético y los campos o datos que lo definen deben ir en el orden en que aparecen.

Notación para el diccionario de datos

Simbolo	Descripción	Ejemplo
=	Esta compuesto de...	NOMBRE = ap+am+nomb
+	Y	SUELDO_NETO = sueldo+ispt
{...}	Iteraciones	FOLIO = 1{dígito} 10
[...|...]	Seleccionar una de las opciones	SEXO = [m | f]
(...)	Opcional	NOMBRE = ap+am+nomb+(seg_nomb)
"..."	Comentarios	ISPT="deducción"
@	Identificador de almacenamiento	ALUMNO = @no_cuenta+nombre+carrera

[2] Ver diagrama de comportamiento, pág. 41.

Al final de la definición de todos los datos empleados en el diagrama de contexto y los DFD's se deben incluir los datos de los almacenes (archivos) utilizados por el sistema, a continuación se presenta el formato a utilizar la esta descripción:

NOMBRE (del archivo):
ALIAS:
COMPOSICIÓN:
ACCESO: (llave primaria del archivo)

Figura 4 Descripción de un archivo en el DD

Modelo de comportamiento

En el tema anterior se describió como desarrollar el modelo ambiental para un sistema, modelando el diagrama de contexto, la lista de eventos y el propósito del sistema. Además, de tener todos éstos elementos se debe comenzar a construir el diccionario de datos, con por lo menos la definición de los datos que representan interfaces entre los terminadores externos y el sistema.

Los productos derivados de esta fase son:

- ☑ Diagrama de Flujo de Datos
- ☑ Miniespecificaciones

Diagrama de Flujo de Datos (DFD)

Su objetivo es mostrar la transformación de una salida a través de un proceso, el flujo de datos entre dos procesos o entre un proceso y un terminador, los almacenamientos que contienen los datos que comparten dos o más procesos en diferentes tiempo o etapas. Indicando los flujos por medio de los cuales se almacenan datos a través de un proceso.

Componentes:

Proceso. Representa una función de la organización que es hecha por el sistema, el proceso es nombrado con un verbo o un sustantivo que describe la función.

Flujo de datos. Representan las entradas y salidas de información, contienen los datos mínimos necesarios para el proceso. Su nombre representa el significado de la colección de datos.

Almacenamiento. Es un repositorio de datos disponibles para los diferentes tipos de accesos, representa la colección de información sobre entidades, relaciones o entidades asociativas.

Para iniciar el diseño del DFD 0 (cero) deberán tomarse en cuenta las siguientes consideraciones:

1. Se dibuja una burbuja para cada acontecimiento de la lista de eventos.

2. La burbuja se nombra describiendo la respuesta que el sistema debe dar al acontecimiento asociado.

3. Se dibujan las entradas y salidas apropiadas de tal forma que la burbuja pueda dar la respuesta requerida y se dibujan los almacenes, como sea apropiado, para la comunicación entre burbujas.

4. El DFD resultante se compara con el diagrama de contexto y la lista de eventos para asegurar que esté completo y sea consistente.

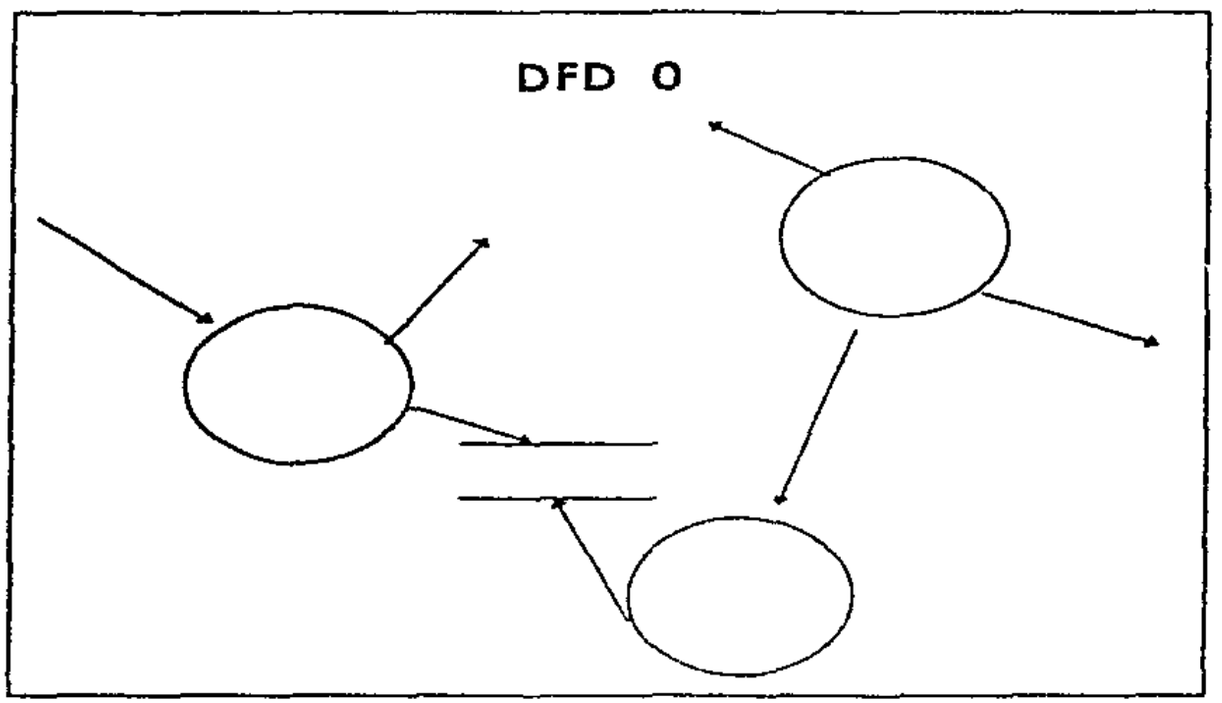


Figura 5 Representación de un DFD 0

También es bueno tener en cuenta que un DFD no debe ocupar más de una hoja por motivos de claridad. Si se quiere representar un sistema que tenga una complejidad tal que no se posible realizar esto, se requiere aplicar el principio de descomposición por niveles, que consiste en definir subsistemas de menor complejidad. Si los subsistemas resultantes son todavía demasiado grandes, se continuará subdividiendo.

El DFD 0 (nivel 1) establece el entorno del sistema incluyéndose fuentes y destinos de datos de naturaleza externa, al igual que las principales entradas y salidas del sistema, su funcionalidad y su frontera con entidades externas.

Los procesos del primer nivel reflejan la visión que tiene el usuario del funcionamiento de las principales áreas del sistema. Cada uno de ellos podrá descomponerse en un DFD de nivel inferior. Los procesos pueden verse como ventanas a un DFD de nivel inferior en la jerarquía, de tal forma que la caja de un proceso de nivel superior representa la frontera del proceso hijo. Según se baja de nivel, los diagramas contendrán más detalles sobre los flujos, datos y almacenes.

En el segundo nivel, los procesos también pueden descomponerse en un tercer nivel y se podría continua, aunque no resulta muy recomendable. Los procesos que no explotan en otros se denominan procesos de bajo nivel o primitivos, y pueden aparecer en cualquier nivel.

Reglas en la descomposición por niveles

1. Numeración de DFD y procesos

   - El DFD del primer nivel se numera como 0 (cero) y sus procesos son numerados como 1,2,3,...
   - La numeración de los procesos en un DFD no indican secuencialidad.
   - Cada DFD hereda el número y nombre del proceso del que se deriva.
   - Los números de los procesos están formados por la concatenación del número del DFD, un punto y un número local.

2. Diagramación

   - Los flujos de datos que se comunican con el proceso padre aparecen cruzando los límites del DFD hijo, entrando o saliendo de algún subproceso.
   - Los almacenes, si sólo se usan dentro de los límites del DFD hijo, aparecerán en él, pero si van a acceder a ellos otros procesos externos deberán permanecer fuera.
   - Las entidades externas también tienen que estar en el exterior.
   - No se aconseja llegar a un nivel de descomposición inferior al tercero, pues se incluiría información demasiado exhaustiva.

3. Descomposición de procesos

La descomposición de procesos significa profundizar en sus detalles, sin por ello perder la relación que tengan con otros proceso, almacenes o entidades externas. El analista tendrá que revisar los DFD de niveles inferiores para

cumprobar su consistencia y completitud con respecto a los superiores. La consistencia en la descomposición significa que los procesos del DFD del nivel 1 se descompondrán en otros. El DFD que muestra la descomposición se denomina "DFD hijo" y el proceso a que define "proceso padre". Para mantener la consistencia en la jerarquia de los DFD se debe aplicar la siguiente regla:

"Las entradas y salidas del proceso padre tienen que ser las mismas que las del DFD hijo".

A esta equivalencia se le llama balanceo. Si el conjunto de DFD está balanceado se asegura la consistencia en todos los niveles, es decir, que lo que está descrito en el DFD de nivel 1, es lo mismos que lo descrito en el DFD de nivel inferior y viceversa. Si no es asi, el DFD superior no refleja todos los DFD que se han derivado de él, entonces tendrá que ser modificado.

4. Descomposición de flujo de datos, almacenes y entidades externas

Cuando un proceso de descompone, puede que los flujos de datos, ya sean de entrada o salida y las fuentes y destinos de datos sean los mismos en los dos niveles. Sin embargo, en el DFD hijo puede mostrarse más detalle explotando los elementos compuestos en otros más sencillos. De esta forma, tanto los flujos de datos como las entidades externas y almacenes pueden explotar en otros, permitiendo con ello que el DFD del nivel superior sea más simple y el inferior más detallado.

Miniespecificaciones

Su objetivo es mostrar una descripción detallada de un proceso primitivo. Cada miniespecificación debe ir en una página separada y sólo llevará el título de "Miniespecificación" la primera de ellas. Las miniespecificaciones o especificación de proceso consta de los siguientes componentes:

1. El nombre del proceso correspondiente en el DFD.

2. Nombre de los flujos de entrada y salida.

3. Los datos de entrada que serán utilizados para validaciones o lecturas.

4. Los datos de salida que serán creados, modificados y borrados por el proceso.

5. Reglas locales, que son las condiciones en las cuales se basan las funciones.

6. Funciones, que es la lógica empleada por las miniespecificaciones.

A continuación se presenta un ejemplo de una miniespecificación.

Miniespecificación

1. Elabora tarjeta de profesores

Para cada periodo escolar hacer:

Obtener de cada profesor los siguientes datos:
- Nombre y apellidos
- RFC
- Dirección
- Grupos y horario

Imprimir cada tarjeta de profesores.

Diseño estructurado

El diseño estructurado es una metodología que transforma el diagrama de flujo de datos (DFD) en un estructura de programas (mapa de estructuras) usando como técnicas: el análisis de transformaciones y el análisis de transacciones.

En la actividad de diseño se identifican los módulos y las interfaces existentes entre ellos para implementar las especificaciones estructuradas. Esta fase consta de:

- *Elaboración del diagrama de estructura (o mapa de estructura).* Consiste en obtener una representación gráfica del sistema mediante una jerarquía de módulos. Para ello, se suelen aplicar dos tipos de estrategias de diseño: el análisis de transformaciones y/o el análisis de transacciones.

- *Evaluación del diagrama de estructuras.* Consiste en refinar el mapa de estructuras, en base a conceptos generales de diseño, tales como: bajo acoplamiento entre los módulos, alta cohesión en la estructura interna de cada módulo, anchura de control (número de módulos dependientes de uno lado) entre 2 y 7 módulos; ámbito de efecto comprendido en el aspecto de control (es decir, cuando en un módulo se produce un suceso, los módulos afectados por ese suceso tienen que estar subordinado a él).

En esta etapa de refinamiento se puede llegar incluso a rechazar el mapa de estructuras existente y diseñar uno nuevo.

- *Diseño de los módulos*. Consiste en obtener el diseño detallado de los procedimientos de cada módulo del sistema. Utilizando el mapa de estructuras refinado, los DFD y DD del sistema lógico, junto con las miniespecificaciones del sistema.

- *Diseño de las bases de datos*. Consiste en diseñar las bases de datos físicas, de acuerdo a la información lógica de los DFD y DD de las especificaciones estructuradas, teniendo, también en cuenta las restricciones físicas del sistema. En esta etapa deberás elaborarse un documento que contenga las diferentes entidades, atributos y valores de los componentes de las bases de datos.

- *Empaquetamiento del diseño*. Consiste en dividir el mapa de estructuras en módulos ejecutables, de acuerdo con el entorno hardware que se va a utilizar. El empaquetamiento se refiere a la memoria disponible, lenguajes de programación utilizados, tipo de hardware, etc. Para hacer este empaquetamiento se usa el mapa de estructura, el diseño de los módulos, el diseño de las bases de datos y las restricciones físicas del sistema.

Como se ha comentado, la transformación de los DFD's en el diagrama de estructuras (mapa de estructuras), que es la etapa fundamental de diseño, se realiza en la metodología estructurada mediante los análisis de transformaciones y transacciones que se describen a continuación.

Análisis de transformaciones

El análisis de transformaciones o diseño centrado en transformaciones es una estrategia de "arriba abajo", modular y centrada en el modelo de transformaciones de Constantine, la cual consiste en definir estructuras equilibradas entre las ramas ***aferentes***[3] que obtienen información y la pasan al nivel superior y las ***eferentes***, que toman información de un nivel superior y la pasan al inferior. Produciendo como resultado, estructuras factorizadas con módulos bien dimensionados.

La estrategia consiste en identificar las entradas de alto nivel, identificar las funciones centrales del sistema e identificar las salidas de alto nivel. Dicha estrategia consta de las siguientes etapas:

[3] Ver Etapa 3 del análisis de transformaciones.

Etapa 1. Revisión del modelo de comportamiento del sistema. El modelo fundamental del sistema engloba el DFD de nivel 0 y la información complementaria. En realidad, el paso de diseño comienza con una evaluación de la especificación del sistema y de la especificación de requisitos del software. Ambos documentos describen el flujo y la estructura de la información de la interfaz del software.

Etapa 2. Revisión y refinamiento de los DFD. Con el fin de conseguir un mayor detalle, se redefine la información contenida en el DFD 0. Para explicar la elaboración del diseño se ejemplificará con el siguiente DFD:

Examinando los DFD's de niveles 1 y 2 de monitorización de sensores (Figuras 6 y 7)

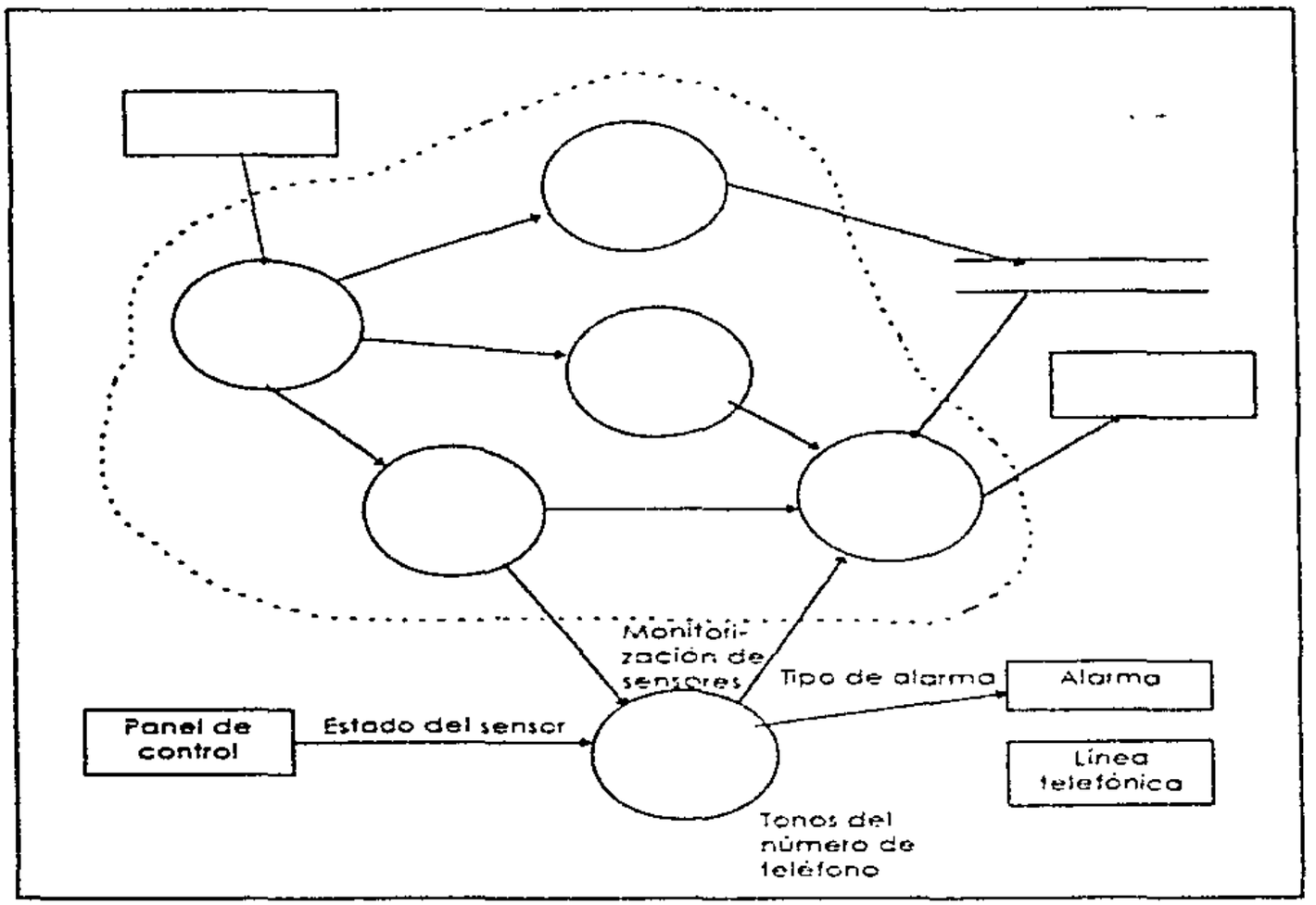


Figura 6 Ejemplo. DFD 0 (nivel 1).

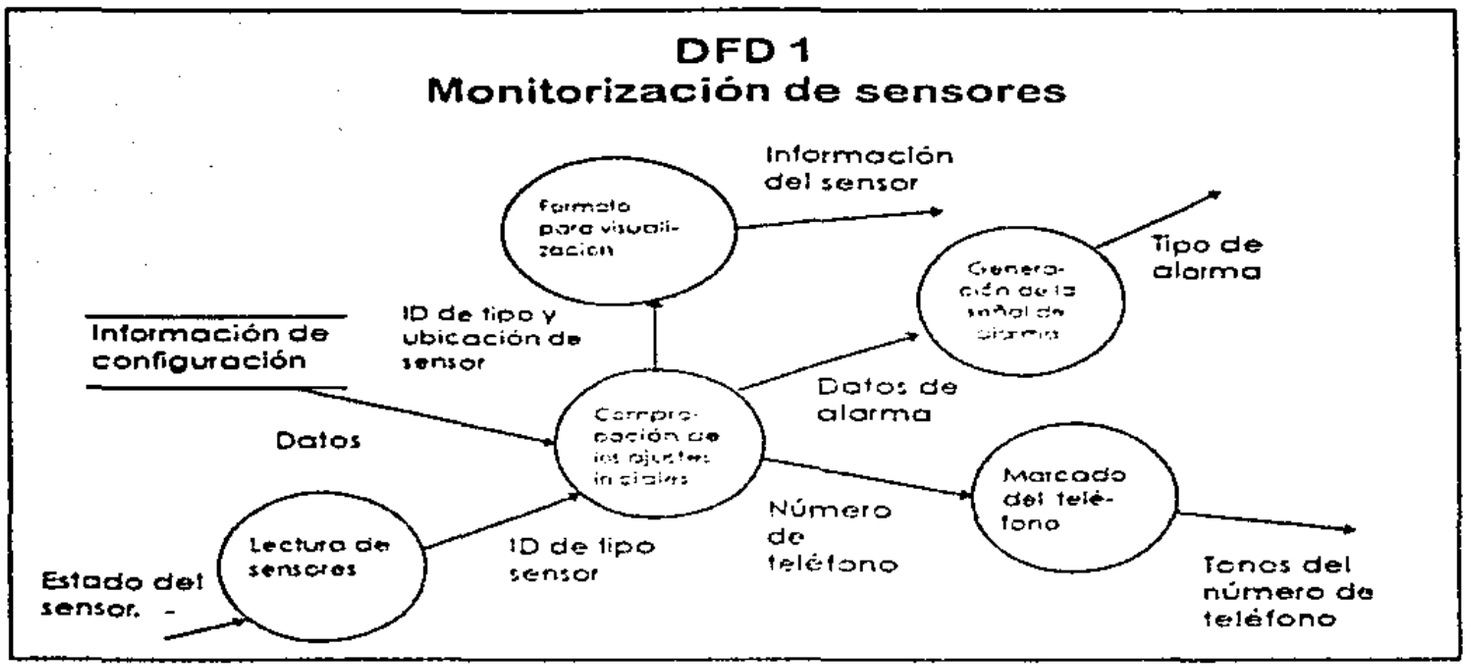


Figura 7 DFD de nivel 2

Se obtiene un diagrama de flujo de datos de nivel 3 como el de la Figura 8, en donde , cada transformación del diagrama de flujo de datos exhibe una cohesión relativamente alta: es decir, cada proceso implicado en una transformación ejecuta una función sencilla y diferenciada, que se puede implementar como un módulo. Por lo tanto, el DFD de la Figura 8 contiene ya el suficiente detalle como para establecer un diseño inicial para la estructura del programa del subsistema monitorización de sensores y poder proceder sin más refinamiento.

Etapa 3. Identificar los elementos de datos aferentes y eferentes. Son elementos de datos "aferentes" aquellos elementos de alto nivel que constituyen las entradas al sistema, las más alejadas de las entradas físicas. Son datos limpios, convertidos, formateados, validos y listos para procesar. Se identifican partiendo de las entradas y moviéndose a lo largo de DFD hasta encontrar un flujo de información que no pueda ser considerando como dato de entrada. Este proceso se efectúa para cada flujo de entrada.

Son elementos de datos "eferentes" aquellos elementos de salida de alto nivel, los más alejados de las salidas físicas. Son datos de salida lógica que produce el

sistema y que se transforman en salidas físicas. Este sistema de división normalmente deja en medio a las "transformaciones centrales" y evita estructuras dirigidas por la entrada o por la salida.

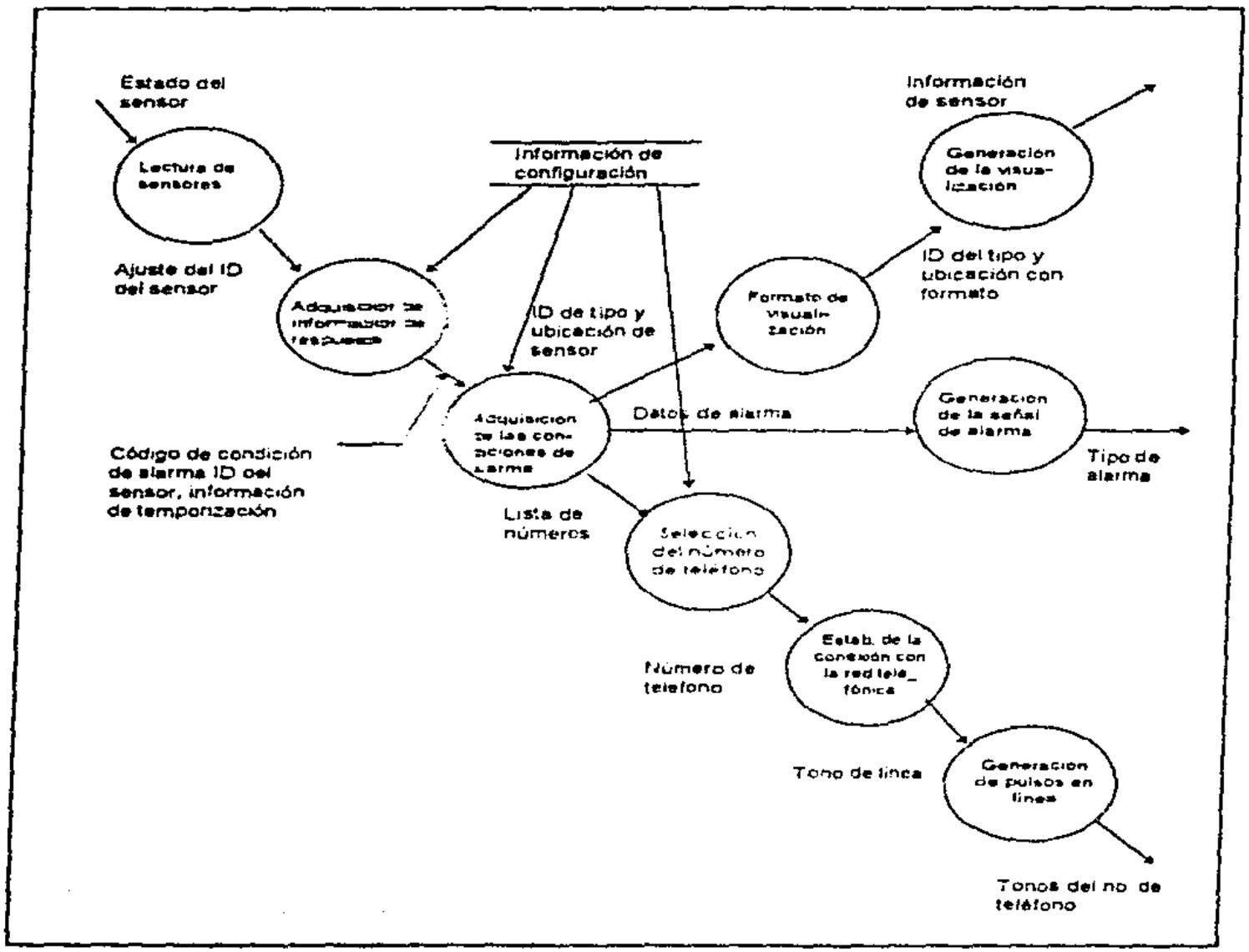


Figura 8 DFD nivel 3

Evaluando el DFD de la Figura 8, se puede observar que los datos entran por un camino de llegada y salen por tres caminos. No existe un centro de transacción claramente definido, aunque la *transformación adquisición de las condiciones de alarma* podría percibirse como tal.

La interpretación de los límites de los flujos aferentes y eferentes es algo subjetivo, debido a que se pueden establecer lugares diferentes para la situación de los límites de flujo. De hecho, pueden obtenerse distintas soluciones de diseño alternativas variando la ubicación de los límites de flujo. Aunque debe tenerse cuidado al establecer los límites, una variación de una burbuja en un camino de flujo, normalmente tendrá poco impacto en la estructura final del programa.

La siguiente figura muestra la situación de los límites de flujo para el ejemplo; en donde las transformaciones (burbujas) que conforman el centro de transformación se encuentran entre los dos límites punteados que van de arriba a abajo en la figura.

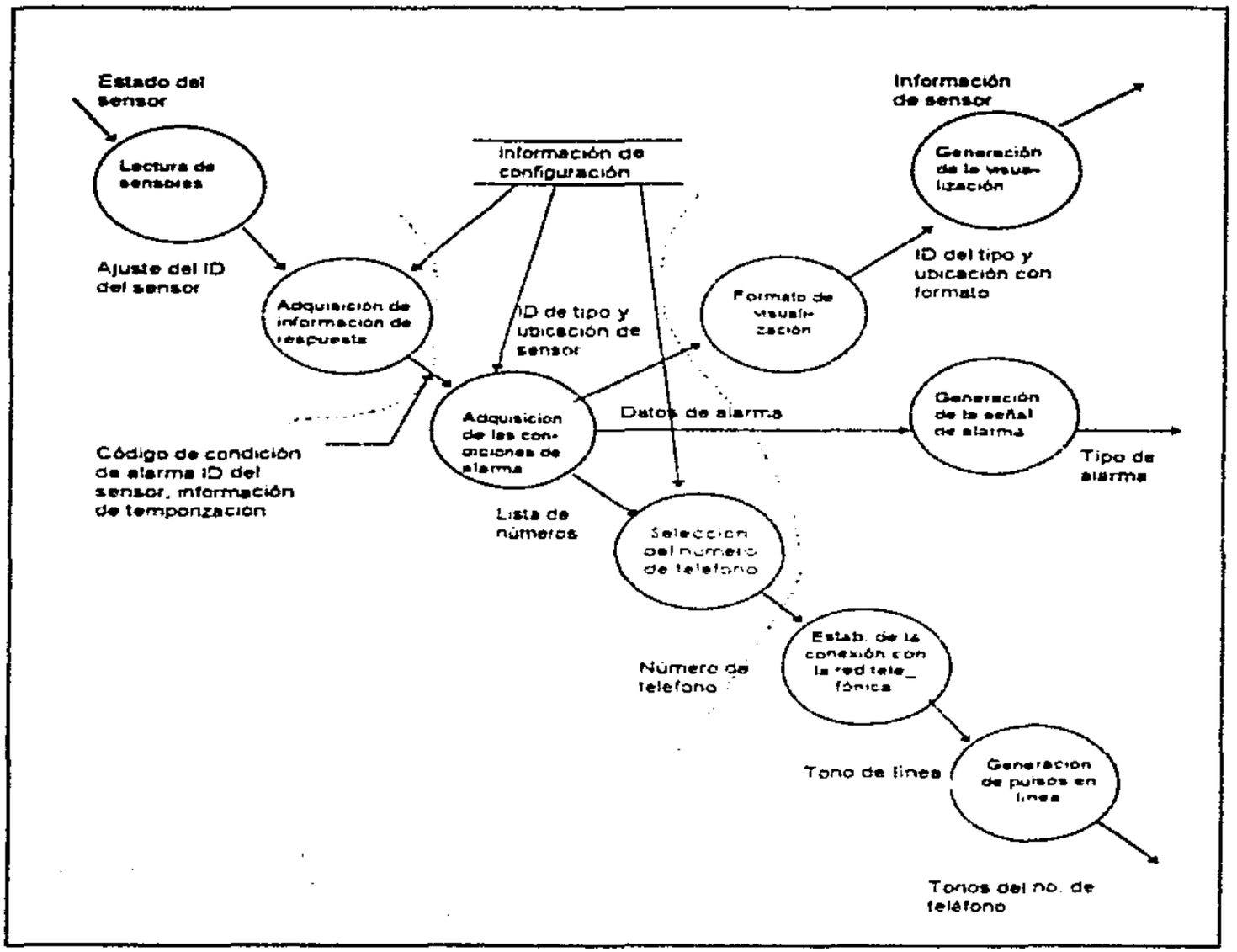


Figura 9 Especificación de los límites de flujo

Etapa 4. Primer nivel de factorización. La estructura del programa representa una distribución descendente del control. La factorización da como resultado una estructura de programa en la que los módulos de nivel superior toman las decisiones de ejecución y los módulos de nivel inferior ejecutan la mayoría del trabajo de entrada, proceso y salida. Los módulos de nivel intermedio ejecutan algún control y realizan cantidades moderadas de trabajo.

Cuando se encuentra un flujo de transformación un DFD se organiza en una estructura especifica que proporciona el control para el procesamiento de la información de entrada, de transformación y de salida. En al siguiente figura se muestra este primer nivel de factorización. En la parte superior de la estructura de programa se encuentra un módulo de control, que sirve para coordinar las funciones del control subordinadas, que son las siguientes:

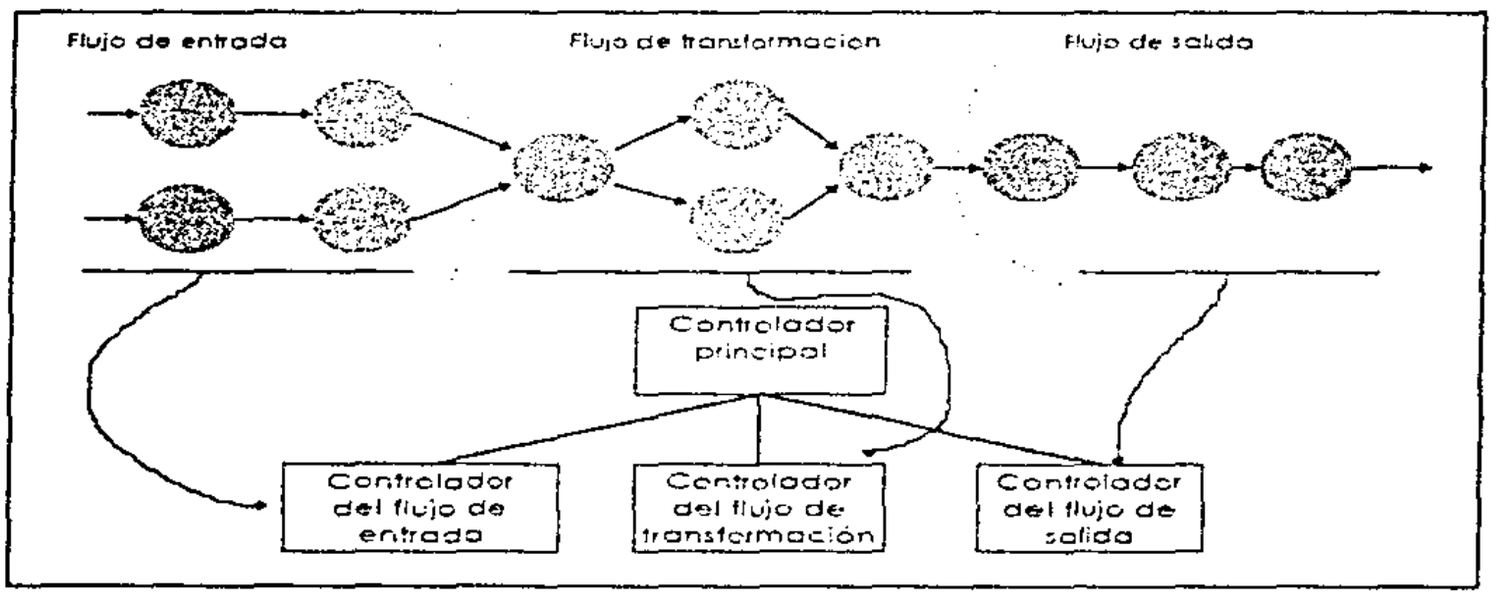


Figura 10 Primer nivel de factorización

- Un controlador del procesamiento de la información de entrada, que coordina la recepción de todos los datos que llegan.
- Un controlador del centro de transformación, que supervisa todas las operaciones sobre los datos en su forma interna.
- Un controlador del procesamiento de la información de salida, que coordina la producción de la información que sale.

Aunque la Figura 10 implica una estructura de tres ramas, en grandes sistemas la complejidad del flujo puede hacer que existan dos o más módulos de control, uno para cada una de las funciones de control genéricas descritas anteriormente. El número de módulos del primer nivel debe limitarse al mínimo necesario, para que puedan realizarse las funciones de control y mantener al mismo tiempo buenas características de acoplamiento y cohesión.

La figura 11 muestra la factorización de primer nivel como una estructura, donde cada módulo de control tiene un nombre que indica la función de los módulos subordinados.

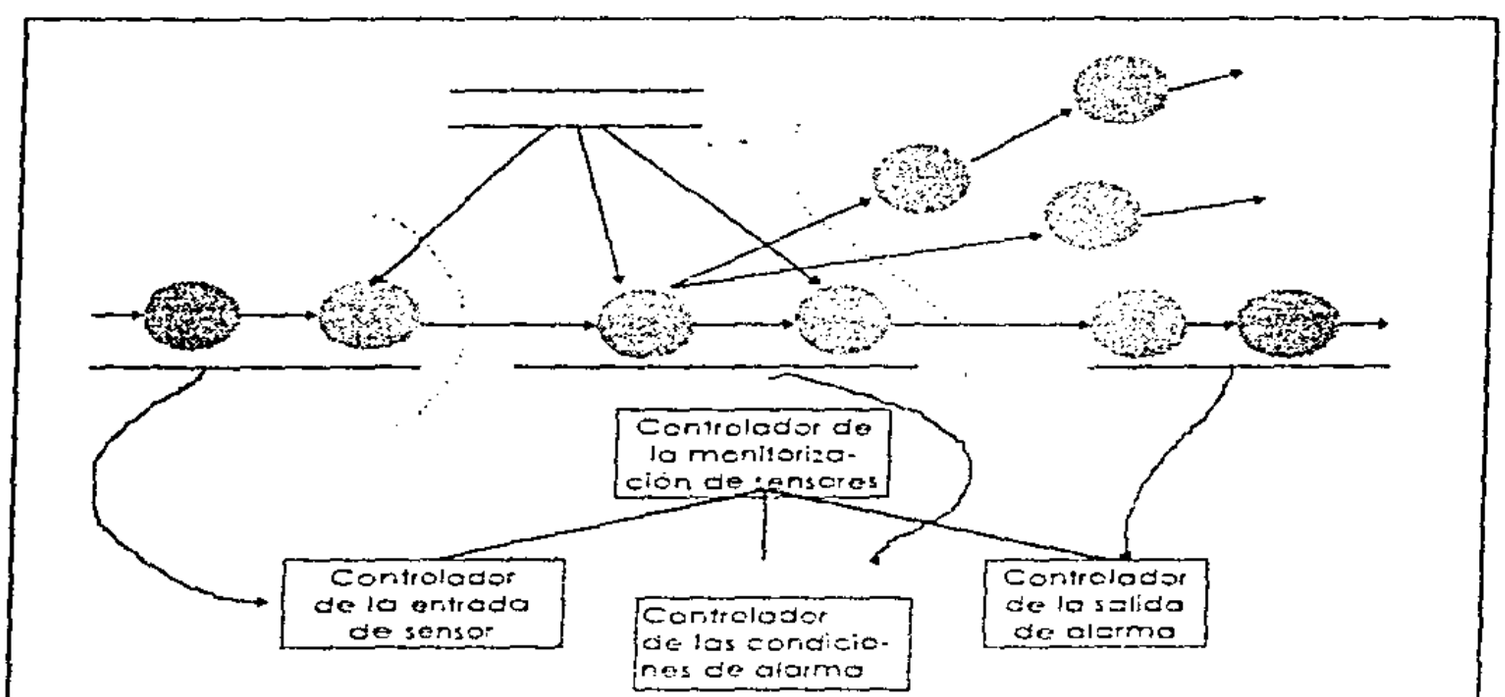


Figura 11 Primer nivel de factorizacion para **monitorización de sensores**

Etapa 5. Segundo nivel de factorización. El segundo nivel de factorización se realiza mediante la conversión de las transformaciones individuales (burbujas) de un DFD, en los módulos correspondientes de la estructura del programa. Comenzando dentro de los límites del centro de transformación y yendo hacia fuera a través de los caminos de entrada y luego de salida, las transformaciones se convierten en niveles subordinados de la estructura del software.

En la siguiente figura se muestra la factorización de segundo nivel:

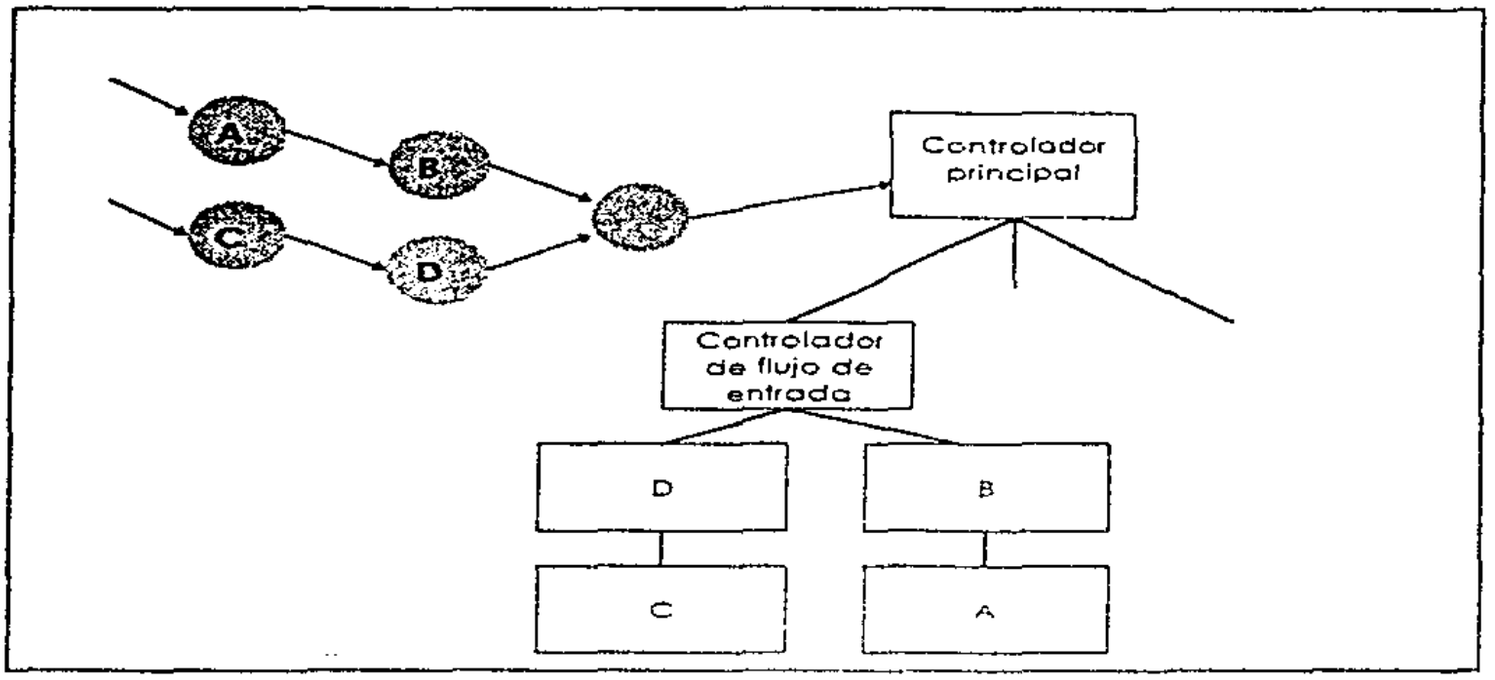


Figura 12 Segundo nivel de factorización

Aunque la figura anterior muestra una correspondencia uno a uno entre las transformaciones del DFD y los módulos del sistema, frecuentemente pueden presentarse diferentes conversiones. Se pueden combinar dos o incluso tres burbujas, representándolas como un sólo módulo (teniendo presentes los problemas de cohesión) o puede dividirse una burbuja en dos o más módulos.

La Figura 13 muestra la estructura de programa derivada para los caminos del flujo de entrada del DFD (ver figura 9). Puede observarse que existe una correspondencia sencilla de uno a uno entre las burbujas y los módulos, siguiendo hacia atrás el flujo a partir del límite del centro de transformación.

El segundo nivel de factorización para el centro de transformación se realiza de la misma manera. De nuevo, se lleva a cabo la factorización yendo hacia fuera desde el límite del centro de transformación correspondiente al flujo de entrada. Cada una de las transformaciones de cálculo o de conversión de datos de la parte de transformación del DFD se convierten en un módulo subordinado al controlador de transformación.

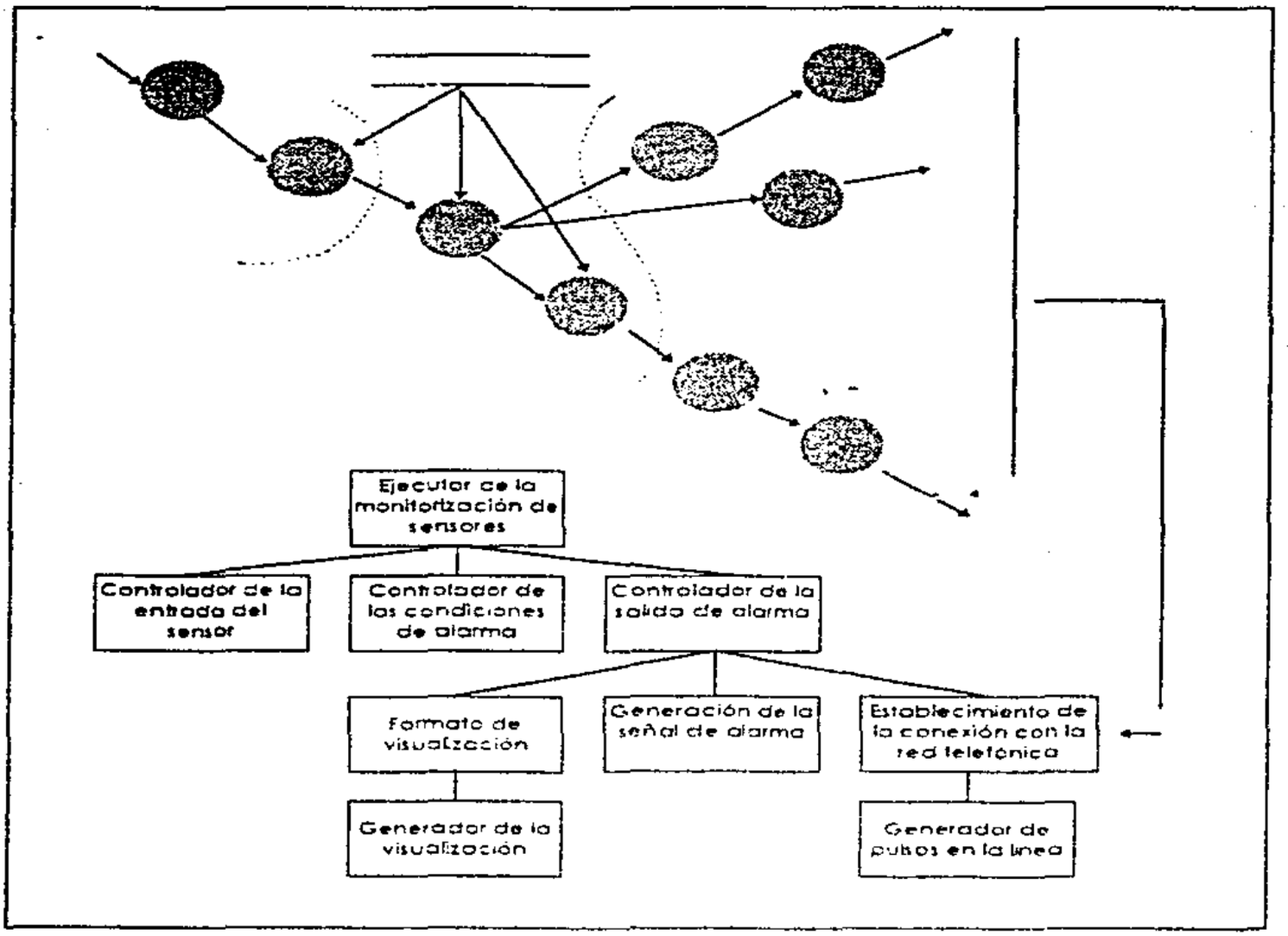


Figura 13 Factorizacion final para **monitorización de sensores**

Finalmente, la estructura definitiva del programa quedaría representada de la siguiente manera:

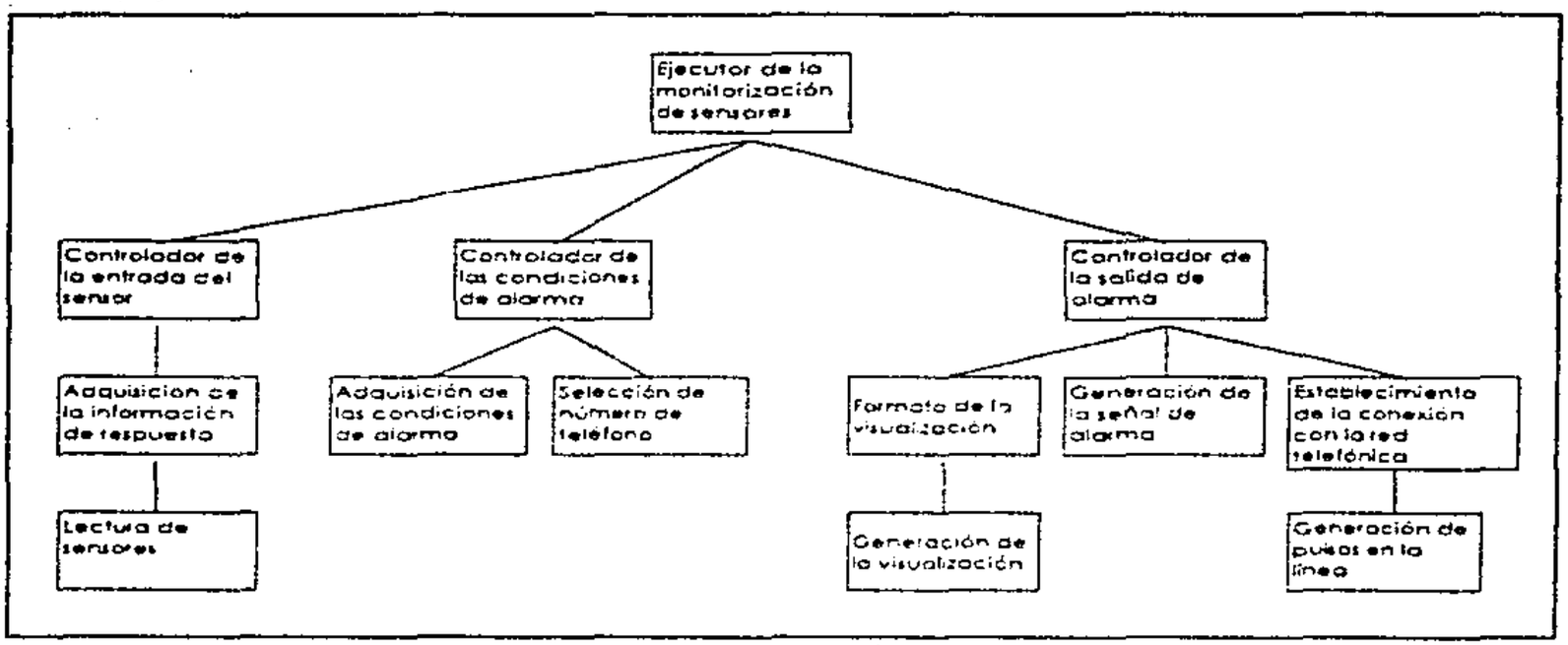


Figura 14 Factorización final

Análisis de transacciones

Este diseño se aplica cuando una "burbuja" del DFD divide el flujo de entrada en varios flujos de salida, construyéndose un sistema alrededor de esa transacción.

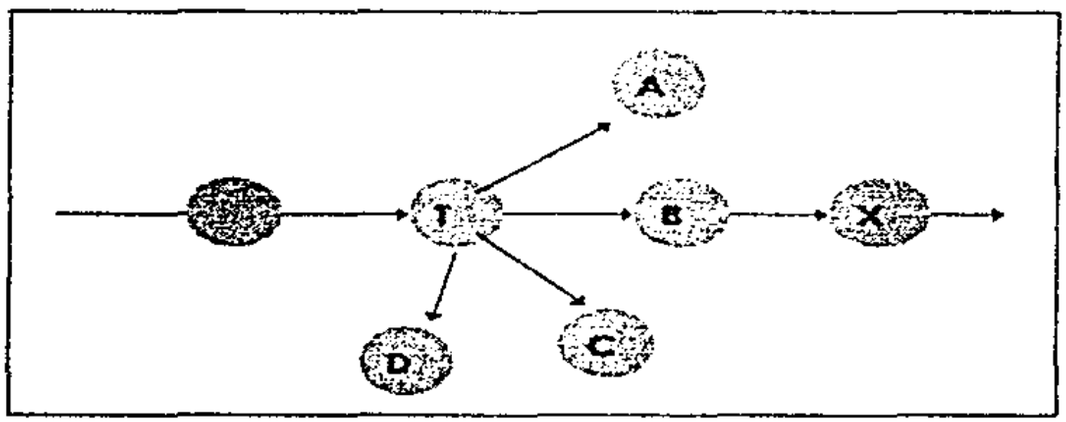


Figura 15 Centro de transacciones

Un centro de transacciones de un sistema debe ser capaz de: obtener transacciones de forma general, analizar cada transacción para determinar su tipo, llevar a cabo la transacción y completar su proceso.

Etapas de la estrategia del análisis de transacciones.

- Identificar las fuentes de las transacciones. Esto se consigue mejor después de la primera etapa de factorización del diseño centrado de transformaciones.
- Especificar la organización apropiada centrada en transacciones, como se muestra en la Figura 15.
- Identificar las transacciones y sus acciones definidas.
- Definir los diferentes módulos estructurados que abarquen a todas las transacciones. Es decir, señalar las diferentes situaciones en que los módulos se pueden combinar.
- Por cada transacción especificar los niveles.
- Por cada acción de una transacción especificar su módulo subordinado.
- Por cada etapa detallada de un módulo de acción, especificar del nivel de detalle.

CAPÍTULO 3

MODELADO DE BASE DE DATOS

3.1 Introducción

Antes de comenzar a diseñar es bueno haber hecho previamente un análisis del sistema, lo cual se hace durante la fase de análisis descrito en el capítulo anterior; sin embargo no es suficiente con identificar las entidades que intervienen en el sistema, sino también conocer de que manera van a interactuar ellas si en el sistema.

Otra parte importante durante el desarrollo de sistemas es el modelado de la base de datos que va administrar los datos del sistema; la primera fase de análisis nos ayuda a identificar las entidades involucradas en el sistema; sin embargo, falta realizar el análisis formal de los datos en sí mismos, buscar la mejor manera de optimizar los medios disponibles de almacenamiento, y la forma en que estos datos serán manipulados para la obtención de información.

De todo esto se encarga el análisis de base de datos, así como de su diseño, el cual es muy útil en la fase final de desarrollo que es la programación. A lo largo del siguiente capítulo se expone un procedimiento a seguir para diseñar una base de datos, el cual se complementa con el análisis y diseño previo.

3.2 Antecedentes

Anterior a la existencia del concepto de base de datos, el procesamiento de información se realizaba mediante archivos de datos para aplicaciones específicas; el centro de atención eran las funciones del procesamiento, mientras que la estructura de los datos jugaba un papel menos importante. Cuando un sistema de procesamiento de datos era diseñado, la función del sistema debía especificarse primero y después los requerimientos de los datos.

Un resultado de este enfoque fue que cada aplicación tuvo sus propios archivos para resolver sus propias necesidades lo que implicaba alta redundancia de datos, inconsistencia, poca seguridad y era difícil compartir la información. Con el objeto de resolver estos inconvenientes, surgió el "enfoque de bases de datos", cuyo objetivo era administrar la información de manera más fácil y segura. Una base de datos centraliza el control sobre todos los datos, reduce redundancia, previene inconsistencias, la información puede compartirse y pueden implantarse medidas de seguridad para su acceso.

El enfoque de base de datos dio como resultado la necesidad de contar con un software como vinculo entre los datos y los usuarios: lo que se conoce como Sistema Manejador de Bases de Datos, mejor conocido bajo las siglas DBMS (en inglés Data Base Management System). Este sistema realiza principalmente las funciones de agregar, recuperar, borrar y modificar información en las bases de datos.

El primer DBMS fue Integrated Data Store (IDS), desarrollado en una computadora General Electric, por el Dr. Charles Bachman en 1962. En 1968 IBM anunció su primer DBMS bajo el nombre de Information Management System (IMS). Para ese tiempo todos los DBMS existentes adoptaban los modelos de datos jerárquico y de red. En 1970 Edgar F. Codd, investigador de IBM, propuso el "modelo relacional para bases de datos".

A partir de las investigaciones de Codd, científicos de esa misma compañía crearon el primer prototipo de DBMS relacional llamado System/R Durante la década de los setenta, en el ámbito comercial predominaron las bases de datos con estructuras jerárquicas y de red, mientras que el nuevo modelo relacional sólo tuvo aceptación en instituciones académicas norteamericanas, con el propósito de seguir la línea de investigación y es hasta 1983 cuando IBM lanza al mercado DB2, su primer DBMS relacional, para "mainframes".

El enfoque de base de datos en general reduce o elimina las desventajas de un sistema convencional de archivos; sin embargo, cuando las estructuras que contienen los datos no se diseñan correctamente, continúan presentes los problemas de redundancia y se corre el riesgo de modificar o borrar datos produciéndose de esta forma información inconsistente. Una base de datos mal diseñada arroja información incorrecta que crea desconfianza entre los usuarios.

Debido a que el éxito o fracaso de una base de datos depende de su diseño y no exclusivamente del DBMS utilizado, es importante, que las personas que diseñan bases de datos tengan a su alcance una metodología que les facilite esta labor.

3.3 Requerimientos para el diseño de base de datos

a) Análisis de datos

El análisis de datos se concentra en los requerimientos del usuario y en el desarrollo de un modelo preciso de los datos del sistema, los cuales deben almacenarse en la base de datos, mientras en el diseño técnico se seleccionan las mejores técnicas de implementación a fin de almacenar los datos definidos

por el modelo del sistema. El análisis de datos se debe basar en un modelo formal, que satisfaga estos objetivos:

1. Que pueda usarse para identificar los requisitos del usuario y presentarlos en una forma tal que los entiendan fácilmente el usuario y todas las personas involucradas en el proyecto.

2. Que se pueda convertir con facilidad en una implementación completamente técnica.

3. Que proporcione reglas y criterios para estructurar la base de datos.

El modelo relacional fue uno de los primeros en enfocar estos tres objetivos y que reúne las características esperadas para un buen diseño, razón por la cual es tomado este modelo para ser explicado en los siguientes temas.

b) Requerimientos

El diseño se puede dividir en varias etapas en donde el objetivo de la primera etapa es analizar los datos en el sistema del usuario. Este paso, que por lo regular se llama *análisis de datos*, es parte importante del análisis del sistema total y produce un modelo de datos del usuario que se emplea en la etapa de diseño del sistema para producir una especificación de una base de datos. Esta especificación se forma con tres componentes, los cuales incluyen:

1. El modelo de la empresa:
   a) El tipo de datos en la empresa
   b) Las relaciones de datos en la empresa

2. El acceso a los requerimientos:
   a) La manera en la cual se van a almacenar o accesar los datos
   b) Que el acceso sea en línea o por lotes

3. Los datos cuantitativos:
   a) La frecuencia con la que se accesan los datos
   b) Tiempo y lugar en que los datos deben estar disponibles
   c) Volumen de los datos
   d) Lugar en el que se capturan los datos

Estos tres componentes se necesitan para implementar una base de datos y por tanto es esencial estar familiarizado con los tipos de datos y sus relaciones con el propósito de almacenar los datos requeridos en la base de datos. El acceso a los requerimientos se usa para estructurar los datos en una forma que los haga más accesibles para todos los usuarios interesados. Los datos cuantitativos son necesarios para asegurar que los sistemas seleccionados tengan suficiente capacidad para almacenar todos los datos.

c) Diseño técnico

Una vez hecho el análisis de los datos y la especificación de la base de datos, el siguiente paso es elaborar un modelo de datos en el cual se definen las estructuras de los datos para construir la base de datos, al seleccionar:

- Los métodos para almacenar todos los datos requeridos.
- El programa que se usa para mantener y accesar los datos.
- La estructura de la base de datos que permitirá a los usuarios accesar los datos en la forma requerida.
- Los métodos para distribuir los datos en varios lugares.

En el diseño técnico la implantación física de la base de datos debe satisfacer los siguientes criterios:

1. Disponibilidad de los datos. Los datos requeridos y sus relaciones se almacenan en la base de datos.
2. Confiabilidad de los datos. Los datos no se perderán ni se dañarán.
3. Actualización de los datos. El valor de los datos debe ser el valor más reciente.
4. Consistencia en los datos. Los mismos valores de datos se obtendrán para el mismo dato, en las diferentes consultas y en igual tiempo.
5. Flexibilidad en los datos. Se pueden hacer extensiones con facilidad para satisfacer nuevos requerimientos.
6. Eficiencia en los datos. Los datos se almacenan y se recuperan a un costo mínimo.

El diseño técnico también especifica los programas que se usarán para accesar la base de datos. Estos programas pueden construirse con programas de bases de datos, lenguajes procedurales y herramientas de desarrollo para el usuario final; es decir, una evaluación de la herramienta con la cual se optimicen los recursos de explotación de la información.

3.4 Sistema Manejador de Bases de Datos
(Database Management System: DBMS)

En 1970 el doctor Edgar F. Codd de IBM publicó un escrito titulado A *Relational Model of Data for Large Shared Banks*. Esta publicación señaló el inicio de la base de datos relacional. Durante los años siguientes, el enfoque relacional a las bases de datos ha recibido mucha publicidad; sin embargo, sólo ha sido durante las últimas décadas que han aparecido en el mercado sistemas de manejo de base de datos que son viables desde el punto de vista comercial.

Un DBMS es un intermediario, ya que en el medio ambiente de procesamiento de datos sirve como interfaz entre los archivos de datos y las personas que buscan los datos de esos archivos; ya sea que el acceso a los archivos se realice en la forma de una consulta o por medio de un informe impreso, la solicitud y la respuesta deben pasar a través del DBMS.

De manera formal un DBMS[1] "es un elemento complejo de software que especifica la forma en que los datos pueden estructurarse, controla todos los accesos a éstos y proporciona algunos otros servicios esenciales de datos". La estructura de datos, en algunos casos de acuerdo con la forma en que esa información se manipula, debe proporcionarse para obtener integración en los datos, reducir su redundancia y para expresar sus relaciones.

El acceso a los datos se proporciona a través de un consulta, por medio de llamadas incluidas en un lenguaje de nivel más alto, o ambos. Otros servicios de datos suministrados por un DBMS son: la seguridad, respaldo y recuperación y control concurrente de las actualizaciones. Un DBMS constituye un "medio de conocimiento" acerca de la estructura de los datos almacenados y de los procedimientos para:

1. agregar nuevos archivos a la base de datos.
2. eliminar archivos de la base de datos.
3. restringir el acceso a los archivos de la base de datos.
4. insertar nuevos datos a los archivos existentes
5. actualizar datos a los archivos existentes.
6. borrar datos de los archivos existentes.
7. recuperar información de los archivos existentes.

Objetivos de un DBMS

- Un medio de separación entre la descripción de los datos y su manipulación.
- Independencia lógica de datos.

---

[1] Gillenson, Mark. *Introducción a las Bases de Datos*, pág. 96.

- Independencia física de datos.
- Interfaz procedural.
- Procesamiento eficiente de las operaciones de la base de datos.
- Fácil administración y control de la administración de datos.
- Minimizar la redundancia y el espacio de almacenamiento.
- Controlar la integridad de los datos.
- Mantener la seguridad de los datos.

Los DBMS pueden ser clasificados de acuerdo al método que usan para estructurar sus datos internamente en tres categorías, que son las más conocidas:

a) DBMS jerárquicos.
b) DBMS de red.
c) DBMS relacional.

Estas estructuras son llamadas **modelos de datos**, porque representan un modelo de la estructura de los datos; ya que, independientemente de la estructura que se utilice debe existir un modelo que indique cómo deben estar organizados los datos y puestos a disposición para todas las aplicaciones posibles.
Un DBMS esta compuesto de dos partes básicas: definición de datos (Data Definition) y manipulación de datos (Data Manipulation).

La definición de datos se refiere a como la estructura de la base de datos se comunica con el DBMS. La estructura completa de la base de datos es conocida como esquema. El esquema describe todos los registros, todos los elementos de los datos y todos los archivos usados en la base de datos. Todos estos elementos son usualmente especificados a través de algún tipo especial de lenguaje llamado ***lenguaje de definición de datos*** (Data Definition Language: DDL). Generalmente cada DBMS tiene su propio DDL.

No todos los usuarios finales o administradores de la base de datos (ABD) necesitan accesar a cada registro o a cada dato de un registro, existen mecanismos que son utilizados para restringir el acceso a los registros y/o a parte de ellos. Este mecanismo empleado es llamado subesquema.

El esquema es la vista lógica global de la base de datos completa y el subesquema representa la vista de un programa de la base de datos; aún cuando es verdad que un subesquema podría contener todos los registros y elementos de datos para cada uno de los registros definidos en el esquema, usualmente esto no ocurre. El usuario final tiene acceso a la base de datos únicamente a través de subesquemas.

La manipulación de datos (Data Manipulation Language: DML) se refiere al método usado para recuperar, agregar, modificar y borrar datos en la base de datos.

cada método es usado a través de operaciones integradas del DBMS. Existen cuatro métodos básicos de manipulación de base de datos[2]:

a) ***Programming language interfaces***, son usados para modificar los valores de la base de datos y producir reportes que requieren cálculos u operaciones sofisticadas, usando algún tipo de lenguaje de programación, ya sea, COBOL, Pascal, RPG, etc.

b) ***Query language*** son lenguajes usados para preguntar a la base de datos, ya que permiten un rápido acceso a ella y son considerados como lenguajes de cuarta generación (4GL). Estos lenguajes no tienen que ser compilados o ligados antes de ser usados, ya que su objetivo es la recuperación de datos. Para este método existen dos categorías: ***command-oriented query language y screen-oriented query language;*** el primero es usado por la mayoría de los lenguajes de consulta, en el cual se usan comandos en inglés, un ejemplo es el Standar Query Language (SQL); en el segundo, el usuario introduce en la pantalla un ejemplo de los datos que él espera recuperar, este método es conocido como Query-By-Example (QBE).

c) ***Report writers*** son diseñados para producir infinidad de reportes grandes, que normalmente son dirigidos a una impresora en línea. Estos programas proporcionan al usuario la posibilidad de generar reportes con características tales como: salto de página, sumas totales, promedios, máximos, mínimos, y contadores.

d) ***System utilities*** son procesos o procedimientos que permiten a los administradores de sistemas mantener el control de cualquier elemento de la base de datos, como los respaldos, combinar bases de datos, restaurar bases de datos, etc.

3.5 Modelo Relacional

El modelo relacional esta basado en el conceplo matemático de relación de la teoría de conjuntos; los matemáticos definen una *relación* como un subconjunto de un producto cartesiano de una lista de dominios, la tabla surge bajo la filosofía de una relación. La diferencia es que en bases de datos se le asigna un nombre a los atributos, mientras que los matemáticos se basan en nombres numéricos, usando el entero 1 para indicar el atributo cuyo dominio aparece primero en la lista de dominios; 2 para indicar el atributo cuyo dominio aparece segundo y así sucesivamente. Puesto que las tablas son esencialmente relaciones, se usan indistintamente los términos matemáticos ***relación*** y ***tupla*** ó los términos correspondientes ***tabla*** y ***fila***.

---

[2] Bisland, Ralph. *Database Management*, pág. 28.

Una entidad es una persona, un lugar, una cosa, un evento o un concepto acerca del cual se desea registrar información y que esta entidad se representa a través de tablas en el modelo relacional.

El DBMS relacional es concebido como una serie de *tablas* bidemensionales en donde a cada una de las tablas se le asigna un nombre único y representa una entidad del mundo real. A su vez cada tabla esta representada por *filas* y *columnas* Una fila de una tabla representa una *relación* entre un conjunto de valores. Las columnas, campos o atributos de una tabla pueden arreglarse en cualquier orden sin afectar el significado de los datos.

Una tabla siempre tiene un campo o grupo de campos que sirven como llave única; debido a que dos registros no pueden ser idénticos. Aquel campo o grupo de campos que se elija(n) como llave única de identificación se denomina *llave primaria*. Si en un conjunto de tablas que constituyen una base de datos relacional un atributo sirve como llave primaria de una tabla y aparece como campo de otra, entonces se denomina *llave foránea*.

Considerando la tabla **depósito** con cuatro atributos *Nombre_Sucursal, No_Cuenta, Nombre_Cliente* y *Saldo*; donde, para cada atributo hay un conjunto de valores permitidos, llamados **dominio**, del atributo. Para el atributo *Nombre_Sucursal* el dominio es el conjunto de todos los nombres de las sucursales, existiendo cuatro tuplas para esa tabla.

TABLA DEPOSITO

Nombre_Sucursal	No_Cuenta	Nombre_Cliente	Saldo
EDS	101	Santiago	500
Sanborn's	215	Fernandez	700
Sandak	102	Romo	400
Audiotec	305	Diaz	350

Como se puede notar la tabla **depósito** contiene datos que de alguna manera son significativos para un conjunto de usuarios y que pueden ser utilizados para posteriores aplicaciones. Otros aspectos importantes en el manejo de bases de datos son las llaves o claves a través de las cuales el programador tiene acceso a tuplas y las asociaciones entre tablas que ayudan a optimizar el manejo de bases de datos; sin embargo, estos temas que también son de interés e importancia serán tratados más adelante.

Básicamente esta es la filosofía en que se fundamenta el modelo relacional y la manera en que son representados los datos en una base de datos. Bajo este modelo se desarrollo el Diccionario -electrónico de términos administrativos, se crearon los archivos siguiendo las normas que cumplen con una base de datos relacional, aunque son pocas tablas las que utiliza el Diccionario, se procuró eliminar redundancia y mantener la integridad de los datos; por esta razón, en el

contenido de éste capítulo no se utiliza para ejemplificar las tablas del Diccionario, sino ejemplos que de alguna manera están asociados al tema que se esta explicando, en el capitulo 5 que es el desarrollo del sistema, se explica la forma en que se aplicó el modelo relacional a la base de datos del Diccionario.

En el enfoque relacional es posible analizar y revisar tres aspectos básicos para su diseño e implementación, los cuales son: la integridad de los datos, el diseño de las bases de datos y el desempeño de las bases de datos.

1. **Integridad de datos.** La integridad de los datos durante la inserción, modificación y borrado de datos es muy significativa, ya que, al realizar cualquiera de las operaciones mencionadas, las tablas donde se encuentren tuplas iguales deberán ser actualizadas automáticamente para mantener la consistencia de los datos.

2. **Diseño y modificación.** Cuando una base de datos relacional se diseña, se hace con base en la forma en que opera la entidad en el medio ambiente, lo cual ayuda para la adecuada distribución de los campos en las aplicaciones, asegurándose de esta manera que cualquier consulta solicitada sea efectuada en su momento para encontrar los campos correctos que son requeridos, aun cuando, se soliciten consultas muy rebuscadas que requieran el rastreo en una o varias tablas haciéndose el proceso más lento de lo común. Las inserciones posteriores a las bases de datos, son más fáciles, ya que, únicamente se expande el número de tuplas de la tabla, sin afectar las aplicaciones o consultas a las tuplas existentes.

3. **Desempeño.** Los sistemas modernos y complicados de base de datos relacional dependen de dispositivos especiales de hardware para mejorar el desempeño o de ingeniosas técnicas de software para acelerar las búsquedas en las aplicaciones de consulta requeridas por el usuario. la eficiencia en los DBMS relacionales por lo común depende del acceso directo a un determinado registro de una tabla, generalmente los RDBMS permiten la creación de indices a través de varios atributos de una tabla.

Reglas de integridad

Las reglas de integridad proporcionan un medio para asegurar que los cambios que se hacen en la base de datos por usuarios autorizados no resulten una pérdida de consistencia de los datos, de esta manera las reglas de integridad protegen la base de datos contra daños accidentales.

a) **Valores nulos.** En la inserción de tuplas incompletas pueden introducirse valores vacios en la base de datos, pero para determinados atributos los valores nulos pueden ser inapropiados; por ejemplo, en la tabla **cliente** si el atributo ***nombre_cliente*** es un valor vacio, posteriormente esta tupla proporcionara una calle y una ciudad de un cliente anónimo, siendo datos que no contiene

información útil. En casos como éste, se deben prohibir los valores nulos, restringiendo el dominio de *nombre_cliente* para que excluya los valores nulos introducidos por error o intencionalmente.

b) **Integridad referencial.** A menudo se quiere asegurar que un valor que aparece en una tabla para un conjunto de atributos dado también aparezca para un cierto conjunto de atributos de otra tabla. Esto se llama *integridad referencial*; es decir, que las actualizaciones (modificación, eliminación, inserción) efectuadas en una tabla que este asociada con otra tabla, deben ser también actualizadas sin que se produzca inconsistencia.

c) **Afirmaciones.** Una afirmación es un predicado que expresa una condición que se desea que siempre satisfaga la base de datos. Por ejemplo que cada cliente de un banco debe mantener una cuenta con un saldo mínimo de $100.00. Cuando se hace una afirmación, el sistema prueba su validez. Si la afirmación es válida, entonces cualquier modificación futura de la base de datos está permitida solo si no provoca que se viole la afirmación.

d) **Disparadores.** Un disparador es una sentencia que el sistema ejecuta automáticamente como un efecto secundario de una modificación de la base de datos. Para diseñar un mecanismo disparador se deben especificar las condiciones bajo las cuales se va a ejecutar el disparador y las acciones que se van a tomar cuando se ejecute el disparador. Para ilustrarlo, supóngase que en lugar de permitir saldos negativos de cuentas, el banco trata los saldos deudores, poniendo el saldo de cuenta a cero y creando un préstamo en la cantidad del saldo deudor. A este préstamo se le da un número de préstamo igual al número de cuenta de la cuenta del saldo deudor; la condición para ejecutar el disparador es una actualización de la tabla *depósito* que resulta en un valor de *saldo* negativo.

Las doce reglas de Codd

Conforme el concepto relacional crecía en popularidad, muchas bases de datos que se llamaban a sí mismas *relacionales*, formalmente no lo eran. En respuesta a la corrupción del término relacional el Dr. Codd escribió un artículo en 1985 estableciendo doce reglas que una base de datos debe obedecer para que sea considerada verdaderamente relacional. Las doce reglas de Codd han sido aceptadas desde entonces como la definición de un DBMS verdaderamente relacional. A continuación se transcriben las doce reglas de Codd:

1. **La regla de información.**
   Toda la información de una base de datos relacional está representada explícitamente a nivel lógico y exactamente de un modo mediante valores en tablas.

2. **Regla de acceso garantizado.**
   Todos y cada uno de los datos (valor atómico) de una base de datos relacional se garantiza que sean lógicamente accesibles recurriendo a una combinación de nombre de tabla, valor de clave y nombre de columna.

3. **Tratamiento sistemático de valores nulos.**
   Los valores nulos (distintos de la cadena de caracteres vacía o una de cadena de caracteres en blanco y distinta del cero o de cualquier otro número) se soportan en los DBMS completamente relacionales para representar la falta de información inaplicable de un modo sistemático e independiente del tipo de datos.

4. **Catálogo en línea dinámico basado en el modelo relacional.**
   La descripción de la base de datos se representa a nivel lógico del mismo modo que los datos ordinarios, de modo que los usuarios autorizados puedan aplicar a su interrogación el mismo lenguaje relacional que aplican a los datos regulares.

5. **Regla de sublenguaje completo de datos.**
   Un sistema relacional puede soportar varios lenguajes y varios modos de uso de terminal (por ejemplo, el modo de rellenar con blancos). Sin embargo, debe haber al menos un lenguaje cuyas sentencias sean expresables, mediante alguna sintaxis bien definida, como cadenas de caracteres y que sea completa en cuanto al soporte de todos los puntos siguientes: Definición de datos, Manipulación de datos, Restricciones de integridad y Seguridad. Esta regla se refiere a que se debe emplear un lenguaje de bases de datos relacional y que dicho lenguaje debe ser capaz de soportar todas las funciones básicas de un DBMS (creación de una base de datos, recuperación y entrada de datos, implementación de la seguridad de la base de datos, etc.).

6. **Reglas de actualización de vista.**
   Todas las vistas que sean teóricamente actualizables son también actualizables por el sistema.

7. **Inserción, actualización y supresión de alto nivel.**
   La capacidad de manejar una relación de base de datos o una relación derivada como único operando se aplica no solamente a la recuperación de datos, sino también a la inserción, actualización y supresión de los datos.

8. **Independencia física de los datos.**
   Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cualquiera que sean los cambios efectuados ya sea a las representaciones de almacenamiento o a los métodos de acceso.

9. **Independencia lógica de los datos.**
   Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cuando se efectúen sobre las tablas base, cambios preservadores de la información de cualquier tipo que teóricamente permitan alteraciones.

10. **Independencia de integridad.**
    Las restricciones de integridad especificas para una base de datos relacional particular deben ser definibles en el sublenguaje de datos relacional y almacenables en el catálogo no en programas de aplicación.

11. **Independencia de distribución.**
    Un DBMS relacional tiene independencia de distribución: es decir, que el lenguaje de base de datos debe ser capaz de manipular datos distribuidos localizados fisicamente en otros sistemas informáticos

12. **Regla de no subversión.**
    Si un sistema relacional tiene un lenguaje de bajo nivel (un sólo registro cada vez), ese bajo nivel puede ser utilizado para suprimir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de nivel superior (múltiples registros a la vez). Impedir "otros caminos posibles" en la base de datos que pudieran modificar la estructura relacional y la integridad de los datos.

3.6 Modelo lógico de datos

Un Sistema Manejador de Base de Datos (DBMS) utiliza un modelo de datos para definir la estructura fundamental de los mismos. Un modelo de datos expresa las entidades y sus relaciones: es la herramienta usada para representar la organización conceptual de los datos. El modelo a utilizar es el modelo relacional que consta de los siguientes elementos básicos:

1. Entidad. Una entidad es una persona, un lugar, un evento o un objeto que se puede identificar en forma única del cuál se registra información y que además cae dentro del alcance del sistema, acerca de la cual el sistema debe mantener, correlacionar y desplegar información. Las entidades son sustantivos y son representadas por medio de tablas. Las entidades pueden ser:

   - Tangibles. Empleados, alumnos, piezas, articulos o lugares.
   - Intangibles. Un suceso, un nombre de actividad, la cuenta de un cliente o un concepto abstracto.

2. **Tabla.** Es un arreglo de dos dimensiones compuesta por renglones y columnas. Una columna contiene un mismo tipo de datos y un renglón contiene un mismo tipo de información; ambos, contienen información perteneciente a una entidad. Consideremos la entidad **Requisiciones** y elaboremos su correspondiente tabla.

REQUISICIONES

NoReq			
8721210-3	Torres	Laureles #21	554-80-44
8834129-1	Quintana	Safiro #45	621-45-67
8723121-4	Bastida	Triunfo #890	841-41-57

Cada renglón de la tabla contiene información perteneciente a una sola persona y una columna contiene datos del mismo tipo; por ejemplo, la columna *NoCuenta* contiene únicamente el número de cuenta de todos los alumnos de la tabla y no más datos de diferente tipo, además:

a) Cada tabla debe tener un nombre único de tabla.
b) Cada columna debe tener un nombre de columna.
c) El nombre de la columna debe ser único dentro de la tabla.
d) El orden de columnas y renglones no es significativo.
e) Las columnas deben ser atómicas (no divisibles).

2. **Atributos.** Son las características propias de la entidad, toda entidad tiene atributos básicos que la caracterizan. Los atributos se modelan como columnas de la entidad. La forma de diferenciar entidades es por medio de atributos, dichas entidades deben tener por lo menos un atributo diferente.

3. **Tupla.** Colección ordenada de uno o más elementos de datos que forman un registro. Es el renglón *n* de la tabla.

4. **Grado.** Son los diferentes valores que puede tomar la tabla con respecto al dominio, es decir, la amplitud de la tabla en cuanto al número de atributos.

5. **Dominio.** Es la colección de valores de los cuales uno o más atributos obtienen sus valores reales.

De acuerdo a la tabla anterior la terminología expuesta se ubica de la siguiente manera:

ALUMNOS

NoCuenta	Nombre	Dirección	Teléfono
8721910-3	Torres	Laureles #21	554-80-44
8834129-1	Quintana	Safiro #45	621-45-67
8723121-4	Bastida	Triunfo #890	841-41-57

ATRIBUTOS

GRADO (4)

Llaves

Puesto que todas las tuplas de la tabla que representan una relación son distintas, tiene que haber uno o más atributos que puedan servir de identificador único de cada tupla. Se denomina llave (o clave) al atributo que permite identificar de manera única a una entidad. Una llave es, en otras palabras, el campo a partir del cual se pueden inferir otros campos de una tabla; por lo que, cada tupla debe estar asociado con una llave que permita su identificación. En la tabla de **alumnos** la llave que identifica unívocamente a un elemento de la tabla es el atributo *NoCuenta*. Es posible que el campo *Nombre* sea también la llave que identifique a un estudiante, pero puede darse el caso que dos estudiantes se llamen igual y ya no habria forma de identificar a los dos estudiantes.

Llave Primaria

La llave primaria (Primary Key, PK) de una tabla identifica en forma única a cada renglón de la tabla.

ALUMNOS

NoCuenta	Nombre	Dirección	Telefono	CodFacultad
8721910-3	Torres	Laureles #21	554-80-44	001
8834129-1	Quintana	Safiro #45	621-45-67	003
8723121-4	Bastida	Triunfo #890	841-41-57	002

PK FK

FACULTADES

CodFacultad	Facultad
001	Ingenieria
002	Quimica
003	Contaduria y administración

PK

En la tabla **alumnos** la PK es el atributo *NoCuenta*, con el cual se identifica a un alumno en particular; y en la tabla **facultades** su PK es *CodFacultad*.

En una tabla la llave primaria no puede contener valores nulos (faltantes o espacios en blanco) ni valores duplicados y no se permiten cambios sobre los valores de la llave primaria; además, una llave puede constar de más de una columna.

Una llave primaria que consta de más de una columna se le llama llave *primaria compuesta*. Una tabla con una llave primaria no compuesta se llama *tabla prima*; y la tabla con una llave primaria compuesta es una *tabla no prima*.

Llave Foránea

Una llave foránea (Foreign Key: FK) es la llave primaria de una tabla y al mismo tiempo forma parte de otra tabla únicamente como atributo. Retomando las tablas **alumnos** y **facultades**, el atributo *CodFacultad* se encuentra en ambas tablas; en la tabla **alumnos** forma parte de ella como un atributo más y en la tabla **facultades** además de ser un atributo de la tabla es la PK de la misma.

ALUMNOS

NoCuenta	Nombre	Direccion	Teléfono	CodFacultad
8721910-3	Torres	Laureles #21	554-80-44	001
8834129-1	Quintana	Safiro #45	621-45-67	003
8723121-4	Bastiaa	Triunfo #890	841-41-57	002

PK FK

FACULTADES

CodFacultad	Facultad
001	Ingeniería
002	Química
003	Contaduría

FK

El atributo *CodFacultad* aparece en más de una tabla (ALUMNOS y FACULTADES) y ambos atributos son del mismo tipo. Generalmente las llaves foráneas (FK) indican asociaciones entre tablas, por lo que sus valores pueden ser nulos o repetidos dentro de la tabla donde son FK.

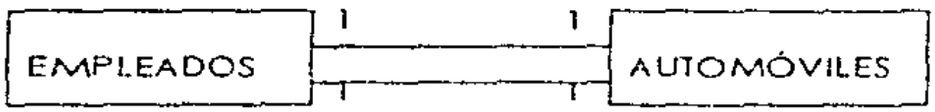
Asociaciones

Una asociación es la unión o enlace entre dos o más entidades (u otras asociaciones), que caen dentro del alcance del sistema, y acerca del cual el

sistema debe mantener, correlacionar y desplegar información. Generalmente las asociaciones requieren de al menos dos entidades (es decir, tablas). Existen tres tipos de asociaciones:

Asociación UNO a UNO (1:1)

Las ocurrencias de una entidad se pueden relacionar sólo a una ocurrencia de la otra entidad. Supóngase que una de sus prestaciones de la empresa 'X', es proporcionar un automóvil a cada uno de sus empleados. En la ***asociación uno a uno*** teniendo las entidades **empleados y automóviles** la asociación se establece de la siguiente manera:



En donde, un empleado puede tener asignado sólo un automóvil y un automóvil sólo puede ser asignado a un empleado; por lo que, su asociación es uno a uno (a un empleado un automóvil y a un automóvil un empleado 1:1).Los atributos correspondientes a las tablas **empleados** y **automóviles** son las siguientes.

EMPLEADOS

Num_Emp	Nombre	Direccion	Num_Auto
001	Alcantara	Olmos #24	102
002	Suarez	Pino #239	101
003	Hernández	Ocoba #453	103

PK

FK (ND)

AUTOMÓVILES

Num_Auto	Tipo	Modelo
101	Shadow	92
102	Spirit	94
103	Jetta	94

PK

Las siglas ND indican que no se admiten duplicados, debido a que dos empleados no pueden tener el mismo auto asignado para su uso personal. Una característica importante es que las asociaciones 1:1 son simétricas; es decir, el campo *Num_Auto* es FK en la tabla **empleados** y PK en la tabla **automóviles** pero es posible que las tablas tengan el siguiente aspecto:

EMPLEADOS

Num_Emp	Nombre	Dirección
001	Alcantara	Olmos #24
002	Suárez	Pino #239
003	Hernández	Ocaba #453

PK

AUTOMÓVILES

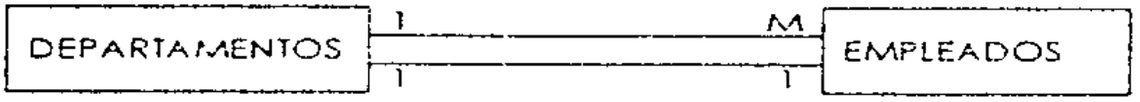
Num_Auto	Tipo	Modelo	Num_Emp
102	Shadow	92	001
101	Spirit	94	002
103	Jetta	94	003

PK FK

Sin embargo; al modelar este tipo de asociaciones hay que hacerlo cuidadosamente de manera que los valores nulos se minimicen o se eviten totalmente.

Asociación UNO a MUCHOS (1:M)

Se dice que una relación entre entidades es uno a muchos, si las ocurrencias de una entidad están relacionadas con diversas ocurrencias de otra entidad. Continuando con el mismo ejemplo la empresa 'X' cuenta con *n* departamentos y de igual manera *m* empleados, considerando, ahora las entidades **departamentos** y **empleados**, su asociación uno a muchos es la siguiente:



En este caso un departamento de la empresa puede tener a su servicio *muchos* empleados y de forma inversa, cada empleado sólo puede prestar sus servicios dentro de la empresa 'X' en *un* departamento a la vez; por lo que, aquí la asociación que se crea es de uno a muchos (1:M).

DEPARTAMENTOS

Cod_Dep ↑	Departamento
NO	Nómina
SI	Sistemas
FI	Finanzas

PK

EMPLEADOS

Num_Emp	Nombre	Dirección	Cod_Dept
001	Alcantara	Olmos #24	NO
002	Suárez	Pino #239	SI
003	Hernández	Ocaba #453	FI

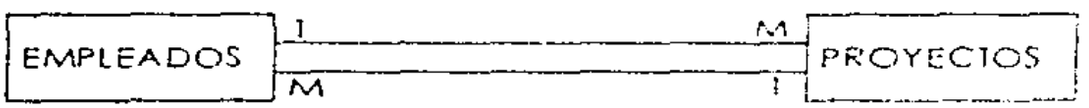
PK FK

Es importante señalar que las asociaciones 1:M no son simétricas, si en el ejemplo anterior se colocara *Num_Emp* como FK en la tabla **departamentos**, sólo se podría asignar un valor a este atributo (en una tupla) pero hay que recordar que un departamento puede tener más de un empleado.

Asociación MUCHOS a MUCHOS

Ocurre cuando se asocian una ocurrencia de una entidad con muchas ocurrencias en la otra entidad y viceversa.

Continuando con la empresa 'X' que es una firma de consultores que elaboran proyectos para otras empresas, en donde cada empleado tiene asignado a su cargo varios proyectos. Teniendo la entidad **empleados** consideremos una nueva entidad llamada **proyectos** para tener un control de las personas que tienen asignado un proyecto. La asociación que se establece es la siguiente:



La asociación que aquí se presenta es de MUCHOS a MUCHOS donde un empleado puede tener a su cargo más de un proyecto y un proyecto estar a cargo de más de un empleado. Para relacionar dos tablas de una asociación M:M, se emplea una tercer tabla que comúnmente se llama tabla asociativa o tabla prima, que no representa una entidad sino una asociación entre entidades.

Esta tabla asociativa forma su llave primaria de la llave primaria de las dos tablas principales de que proviene. Una tabla asociativa es una tabla prima en la cual todos los componentes de su llave primaria son llaves foráneas; además, esta tabla puede contener más atributos que la caracterizan.

EMPLEADOS

Num_Emp	Nombre	Dirección
002	Suárez	Pino #239
001	Alcantara	Olmos #24
003	Hernández	Ocaba #453

PK

PROYECTOS

Num_Proy	Nom_Proy	Fecha_Inicio	Duración	Cliente
201	Sistema de Nómina	Mayo 15, 92	2 año	Sanborn's
202	Inventarios	Enero 01,90	1 año	Audiotec
203	Control de profesores	Dic 18,94	14 meses	La Salle

PK

Como se menciono anteriormente, para relacionar EMPLEADOS y PROYECTOS se utiliza una tercer tabla que es la asociativa.

EMPLEADOS/PROYECTOS

Num_Emp	Num_Proy	Actividades
001	201	Control del equipo. Análisis.
002	201	Diseño
001	202	Proyecto completo.
003	203	Análisis
002	203	Diseño y programación

FK FK

PK

3.7 Normalización

En términos generales, el objetivo del diseño de una base de datos relacional es generar un conjunto de esquemas (diseño global de la base de datos) de relaciones que nos permitan almacenar información sin redundancia innecesaria, pero que a la vez nos permitan recuperar información fácilmente. Una técnica consiste en diseñar esquemas que tengan una *forma normal* adecuada. Una forma normal representa un buen diseño de bases de datos Para determinar si un esquema de relaciones tiene una de las formas normales, se necesita información adicional sobre la entidad del mundo real que se modelara con la base de datos

La normalización es una técnica para el diseño de bases de datos, representativa del tipo de métodos que toman de entrada una lista de campos y

las asociaciones entre ellos. La normalización de datos es una metodología para arreglar campos en tablas de manera que se elimine la redundancia entre los campos no llave y donde cada una de las tablas resultantes se ocupa de una sóla área de conocimiento. La entrada requerida por el proceso de normalización es una lista del conjunto de campos de datos y las asociaciones entre ellos.

Dependencia funcional

Al intentar la definición de las relaciones entre datos primero se debe tratar de descubrir cuáles campos (atributos) dependen de cuáles otros. La dependencia funcional se expresa como: *el atributo B de la tabla Tipos es funcionalmente dependiente del atributo A de la tabla Tipos si, en cada instante, cada valor de A está asociado con no más de un valor de B dentro de la tabla Tipos.*

Decir que B es funcionalmente dependiente de A es equivalente a decir que A identifica a B o de igual manera, si en cualquier instante es conocido el valor de A, el valor de B queda determinado. Por ejemplo, considerando la tabla:

EMPLEADO (NroEmpleado, NombreEmpleado, Salario, NroProyecto, FechaTerminación)

Las dependencias funcionales en esta relación son las siguientes:

NroEmpleado	es dependiente de	NombreEmpleado
NombreEmpleado	es dependiente de	NroEmpleado
Salario	es dependiente de	NombreEmpleado o NroEmpleado
NroProyecto	es dependiente de	NombreEmpleado o NroEmpleado
FechaTerminación	es dependiente de	NombreEmpleado, NroEmpleado, o NroProyecto

NroEmpleado no es funcionalmente dependiente de Salario, porque varios empleados pueden tener el mismo salario; de igual modo, *NroEmpleado* no depende funcionalmente de *NroProyecto*, pero si de *FechaTerminación*. Nótese que ningún otro atributo de esta relación es totalmente dependiente de *NroProyecto*.

Datos no normalizados

En la siguiente tabla se muestran los datos de ejemplo en una tabla no normalizada, caracterizada por el exceso de redundancia.

Mecánicos

NumMec	NumCap	CatCap	NomMec	EdadMec	NumTall	CdTaller	Superv.	Calif
21	113	Carroceria	Pérez	55	52	Monterrey	Robles	3
35	113 179 204	Carroceria Motor Transmis	Castro	32	44	D.F.	Rios	5 1 6
50	179	Motor	López	40	44	D.F.	Rios	2
77	148 361	Llantas Motor	Garcia	47	52	Monterrey	Robles	6 6

Datos no normalizados

Primera Forma Normal (1NF)

Los datos en la primera forma normal tienen la propiedad de que cada anotación de datos o valor de campo, debe ser indivisible (atómico). La siguiente figura es la representación de la primera forma normal de los datos de la tabla anterior. Cada anotación de campo de cada registro está constituida por un solo elemento de datos no subdivisible. En esencia, los registros no normalizados con campos multivaluados se desglosaron para producir varios registros con algunos datos repetidos, según sea necesario; la representación de los datos en la primera forma normal no es, por sí misma, útil para el control de la redundancia sino sólo un punto de arranque para continuar trabajando.

Mecánicos

NumMec	NumCap	CatCap	NomMec	EdadMec	NumTall	CdTaller	Superv	Calif
21	113	Carroceria	Pérez	55	52	Monterrey	Robles	3
35	113	Carroceria	Castro	32	44	D.F.	Rios	5
35	179	Motor	Castro	32	44	D.F.	Rios	1
35	204	Transmis	Castro	32	44	D.F.	Rios	6
50	179	Motor	López	40	44	D.F.	Rios	2
77	148	Llantas	García	47	52	Monterrey	Robles	6
77	361	Motor	Garcia	47	52	Monterrey	Robles	6

PK

Primera Forma Normal

Segunda Forma Normal (2NF)

Tal cual se puede apreciar, los datos de la tabla anterior (1NF) son muy redundantes. En este punto la metodologia se orienta al problema de *qué* buscar en la estructura de datos y *qué* modificar para disminuir la redundancia.

La combinación de campos *NumMec* y *NumCap* es una llave (PK) válida para este archivo(tabla), lo cual significa "que cada campo no llave del archivo

depende de la llave primaria"; aunque ambas partes de la llave primaria compuesta son necesarias para definir el campo *Calif* sólo una de ellas se necesita para definir cada uno de los otros campos no llave (*CatCap, NoMec, EdadMec, NumTall, CdTaller, Superv*).

En las tablas siguientes se muestran los mismos datos en la segunda forma normal; los cuales, se han dividido en tres tablas, cada una de las cuales tiene la propiedad de que su llave primaria define a cada uno de sus campos no llave. Dentro de una tabla, ningún campo no llave está definido por una parte de la llave solamente.

Varios campos se duplicaron en este proceso, NumMec que sólo aparecia una vez como campo en la primera forma normal, ahora aparece tanto en la tabla de **mecánicos** como en la tabla de **calificación**; este tipo de duplicación de campo es necesario en esta etapa entre aquellos campos que participan como campos llave (PK). En realidad, el número total de ocurrencias de valor de campo (total de datos) ha disminuido de 63 de la tabla anterior (1NF) a 55 en las tablas siguientes (2NF), lo que indica un decremento en la redundancia. Obsérvese también que todas las entidades identificables de manera individual están representadas en cada una de las tablas.

Mecánicos

NumMec	NomMec	EdadMec	NumTall	CdTaller	Superv
21	Pérez	55	52	Monterrey	Robles
35	Castro	32	44	D.F.	Rios
50	López	40	44	D.F.	Rios
77	Garcia	47	52	Monterrey	Robles

PK

Capacidad

NumCap	CatCap
113	Carroceria
148	Llantas
179	Motor
204	Transmis
361	Motor

PK

Calificación

NumMec	NumCap	Calif
21	113	3
35	113	5
35	179	1
35	204	6
50	179	2
77	148	6
77	361	6

PK

Segunda Forma Normal

Tercera Forma Normal (3NF)

La tabla de **capacidad** y la de **calificación** en las tablas de 2NF están libres de redundancia por que todos los campos no llave dependen de su llave primaria. Así, ambas tablas ya están en tercera forma normal. Regresando a la tabla **mecánicos** se observa cierta redundancia residual; por ejemplo, los registros 1 y 4 indican el taller número 52 en la ciudad de Monterrey y lo supervisa el Sr. Robles y algo similar sucede con los registros 2 y 3, y el taller número 44; sin embargo, de acuerdo con la llave primaria de esa tabla *NumMec*, identifica cada uno de sus otros campos, pero dado, que la llave está constituida por sólo un campo, la regla de la segunda forma normal de dependencia de los campos no llave con respecto a la llave primaria no se cumple.

El problema está relacionado con las asociaciones entre, *NumTall* y *CdTaller*, y entre *NumTall* y *Superv*. *NumTall* define *CdTaller* y *Superv*, pero claramente no es una llave primaria para toda la tabla (no define a *NumMec*, *NomMec* o *EdadMec*). De esta manera, la situación que se crea es que un campo no llave define a otros del mismo tipo. Esta forma de relación inesperada, que es otra indicación de que se están mezclando tipos fundamentalmente distintos de información en la misma tabla, es la que provoca la redundancia que queda.

Mecánicos

NumMec	NomMec	EdadMec	NumTall
21	Pérez	55	52
35	Castro	32	44
50	López	40	44
77	García	47	52

PK

Talleres

NumTall	CdTaller	Superv
44	D.F.	Ríos
52	Monterrey	Robles

PK

Capacidad

NumCap	CatCap
113	Carroceria
148	Llantas
179	Motor
204	Transmis
361	Motor

PK

Calificación

NumMec	NumCap	Calif
21	113	3
35	113	5
35	179	1
35	204	6
50	179	2
77	148	6
77	361	6

PK

Tercera Forma Normal

En las tablas anterior se muestra la representación de los datos en la tercera forma norma: obsérvese que se ha creado una nueva tabla, la de **taller**, para separar los datos relacionados con los talleres y que al dividir la tabla taller se dejó una copia de la columna *NumTall* en la tabla de mecánicos, esto de hace debido a que fue la única forma de continuar indicando con qué taller estaba asociado un mecánico.

En la tercera forma normal no existe situación alguna en la que un campo no llave defina a otro del mismo tipo en una tabla.

CAPÍTULO 4

SELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN

4.1 Introducción

Una vez que se ha llevado a cabo análisis y diseño de sistema, así con de la base de datos, otra etapa importante es realizar una evaluación o cuando menos hacer una consideración de las herramientas que se tienen disponibles para ver cual de ellas se adecua a las necesidades del sistema.

Esta evaluación incluye revisar, las características de los lenguajes que se encuentran en el mercado para ver que tan viable es el uso de uno de ellos. La evaluación no sólo incluye la facilidad del lenguaje, sino la filosofía bajo la cual opera el lenguaje en sí.

A continuación se presentan algunos de los paradigmas de programación bajo los cuales se fundamentan algunos lenguajes que existen hoy en día en el mercado, el conocer estos paradigmas, nos puede ayudar en la elección del lenguaje adecuado para desarrollar la aplicación, aunque siempre hay que tener presentes los volúmenes de información a manejar y sobre todos la complejidad de los procesos ya que es frecuente que el usuario siempre espera obtener el menor tiempo de respuesta posible.

4.2 Paradigmas de programación

Un *paradigma de programación* es una colección de modelos conceptuales que juntos conforman el proceso de diseño y determinan, al final, la estructura de un programa, de acuerdo a directrices específicas, tales como: estructura modular, fuerte cohesión[1], alta reusabilidad, etc.

Esa estructura conceptual de modelos está pensada de forma que esos modelos determinen la forma correcta de los programas y controlen el modo de pensar y formular soluciones. Para que este proceso sea efectivo, las características del lenguaje deben reflejar adecuadamente los modelos conceptuales de ese paradigma.

---

[1] *"Mide la fuerza funcional relativa de un módulo y se da cuando un módulo ejecuta una tarea sencilla y requiere poca interacción con procedimientos que se ejecutan en otras partes o módulos".*

Existen tres categorías de paradigmas de programación:

a) Los que soportan técnicas de programación de bajo nivel.

b) Los que soportan métodos de diseño de algoritmos.

c) Los que soportan soluciones de programación de alto nivel.

Dentro de los paradigmas que soportan la programación de alto nivel se encuentran tres categorías de acuerdo con la solución que aportan para resolver el problema.

a) Solución procedural u operacional. Describe etapa a etapa el modo de construir la solución, señala ***cómo*** obtener la solución.

b) Solución demostrativa. Es una variante de la procedural. Especifica la solución describiendo ejemplos y permitiendo que el sistema generalice la solución de esos ejemplos para otros casos.

c) Solución declarativa. Señala las características que debe tener la solución, sin describir cómo procesarla; es decir, señala ***qué*** se desea obtener pero no ***cómo*** obtenerlo.

a) Paradigmas procedurales u operacionales

La característica fundamental de estos paradigmas es la secuencia realizada etapa a etapa para resolver un problema. Su mayor dificultad reside en determinar si el valor obtenido es la solución correcta del problema, por lo que se han desarrollado diferentes técnicas de depuración y verificación para probar la veracidad de los problemas implementados con este tipo de paradigmas.

Los paradigmas procedurales son de dos tipos básicos: los que actúan modificando repetidamente la representación de los datos (efecto de lado), y los que actúan creando continuamente nuevos datos (sin efecto de lado).

Los paradigmas con efecto de lado utilizan un modelo en el que las variables están estrechamente relacionadas con direcciones de memoria. Cuando se realiza una ejecución del programa, el contenido de estas direcciones es actualizado repetidamente (las variables reciben múltiples asignaciones), y cuando el proceso termina, los valores finales de las variables representan el resultado.

Existen dos tipos de paradigmas con efecto de lado:

- el imperativo y
- el orientado a objetos.

Los paradigmas sin efecto de lado incluyen a los denominados paradigmas funcionales. Los paradigmas procedimentales definen la secuencia explícitamente, pero esta secuencia se puede procesar en serie o en paralelo. Es este segundo caso, el procesamiento paralelo puede ser asincrono (cooperación de procesos paralelos) o síncrono (procesos simples aplicados simultáneamente a muchos objetos).

b) Paradigmas declarativos

En este tipo de paradigmas, un programa se construye señalando hechos, reglas, restricciones, ecuaciones, transformaciones y otras propiedades derivadas del conjunto de valores que configuran la solución.

A partir de esta información, el sistema debe proporcionar un esquema que incluya el orden de evaluación que procese la información.. Aquí no existe la descripción de las diferentes etapas a seguir para alcanzar una solución como en el caso procedural. Estos paradigmas permiten utilizar variables para almacenar valores intermedios, pero no para actualizar estados de información.

Dado que estos paradigmas especifican la solución sin indicar como construirla, en principio eliminan la necesidad de probar que el valor obtenido sea el valor solución. En la práctica, mientras que muchos de los paradigmas no tienen secuencia de control y efecto de lado que requiera la noción de estado, las soluciones son producidas más como construcciones que cómo especificaciones. Por lo que, los paradigmas resultantes y los lenguajes que los soportan no son verdaderamente declarativos, son ***seudodeclarativos***. Se encuentran en este grupo: el funcional, el lógico y el de transformación.

c) Paradigmas demostrativos

Cuando se programa bajo un paradigma demostrativo, el programador no especifica de forma procedural cómo construir una solución, en su lugar, se presentan soluciones de problemas similares y permite al sistema que generalice una solución procedural a partir de estas demostraciones. Los esquemas individuales para generalizar tales soluciones van desde simular una secuencia procedural hasta inferir intenciones.

Los sistemas que infieren, intentan generalizar usando razonamientos basados en el conocimiento. Una solución basada en la inferencia intenta determinar en qué son similares un grupo de datos u objetos, y, a partir de ello, generalizar estas similitudes.

Otra solución es la programación asistida: el sistema observa acciones que el programador ejecuta, y si son similares o acciones pasadas, intentará inferir cuál es la próxima acción que hará el programador.

Los dos principales inconvenientes a los sistemas de inferencia:son:

- Si no se comprueban exhaustivamente pueden producir programas erróneos que trabajan correctamente con los ejemplos de prueba, pero que fallen posteriormente en otros ejemplos.

- La capacidad de inferencia es tan limitada que el usuario debe guiar el proceso en la mayoría de los casos.

Los resultados más satisfactorios de los sistemas de inferencia han sido en áreas limitadas donde el sistema tiene un conocimiento semántico importante de la aplicación. El mayor problema que se presenta con los sistemas demostrativos es conocer cuándo un programa es correcto. En el caso de los sistemas procedurales, esto se consigue estudiando el algoritmo y el resultado de juegos de ensayo apropiados.

En el caso de los sistemas demostrativos, el algoritmo se mantiene en una representación interna y su estudio se sale del ámbito de estos sistemas. Por lo que la veracidad de la decisión se debe hacer exclusivamente en base a la eficiencia del algoritmo sobre los casos específicos de prueba.

Cabe señalar que la programación demostrativa es una solución "bottom-up", la cual se adapta a nuestra capacidad natural de pensar; sin embargo, esto no es así en la mayor parte de los paradigmas en que la resolución del problema se efectúa aplicando modelos abstractos "top-down".

A continuación, se presentan los paradigmas que actualmente tienen una mayor relevancia en el campo de la programación:

- *El paradigma imperativo*, basado en el tipo procedural con efecto de lado.

- *El paradigma funcional*, basado en el tipo procedural sin efecto de lado, o en el tipo pseudodeclarativo, según que la secuencia de control tenga o no mayor relevancia.

- *El paradigma lógico*, basado en el tipo pseudodeclarativo.

- *El paradigma orientado a objetos*, basado en el tipo procedural con efecto de lado y que incorpora tipos de datos abstractos.

- *El paradigma heurístico*, basado en el tratamiento de "reglas lógicas".

Hay que tener en cuenta que este último tipo de paradigma de programación no se puede clasificar en base al grado en que la secuencia de control es más o menos importante, ya que la programación heurística puede encajar en ambos tipos de programación procedural o declarativa.

El hecho por el que se incorpora la programación heurística como un tipo nuevo de paradigma se debe, fundamentalmente, a que el tipo de problemas que trata suele exigir técnicas específicas de representación, de búsqueda y resolución, en las que el componente heurístico es esencial.

Paradigma imperativo

La programación en el paradigma imperativo consiste en determinar qué datos son requeridos para el cálculo, asociar a esos datos unas direcciones de memoria y efectuar, paso a paso, una secuencia de transformaciones en los datos almacenados, de forma que el estado final represente el resultado correcto.

En su forma pura, el paradigma imperativo únicamente soporta sentencias simples que modifican la memoria y efectúan bifurcaciones condicionales e incondicionales. Incluso cuando se añade una forma simple de abstracción procedural el modelo permanece básicamente sin cambiar. Los parámetros de los procedimientos son "alias" de las zonas de memoria, por lo que pueden alterar su valor y no retornar ningún tipo de cálculo. La memoria también se puede actualizar directamente mediante referencias globales.

El paradigma imperativo debe su nombre al papel dominante que desempeñan las sentencias imperativas. Su esencia es el cálculo iterativo, paso a paso, de valores de nivel inferior y su asignación a posiciones de memoria. Entre las características de éste paradigma se encuentran:

- *Concepto de celda de memoria ("variable") para almacenar valores.* El principal componente de la arquitectura es la memoria, compuesta por un gran número de celdas donde se almacenan los datos. Las celdas tienen nombres que las referencian, y sobre las que se producen efectos de lado y definiciones de alias.

- *Operaciones de asignación.* Estrechamente ligado a la arquitectura de la memoria se encuentra la idea de que cada valor calculado debe ser almacenado, es decir, asignado a una celda. Esta es la importancia de la sentencia de asignación en el paradigma imperativo. Las nociones de celda de memoria y asignación en bajo nivel, se extienden a todos los lenguajes de programación.

- *Repetición*. Un programa imperativo, normalmente realiza su tarea ejecutando repetidamente una secuencia de pasos elementales, ya que en este modelo la única forma de ejecutar algo complejo es repitiendo una secuencia de instrucciones.

A este tipo de paradigma de programación se le denomina también paradigma algorítmico debido a que es similar al método, técnica, procedimiento o rutina y se define como "un conjunto finito de reglas diseñadas para crear una secuencia de operaciones para resolver un tipo específico de problemas".[2]

No obstante, se entiende que aunque el concepto de algoritmo encaja en otros tipos de paradigmas, es privativo del tipo de programación procedural en el que su característica fundamental es la secuencia de procedimientos. Atendiendo a los lenguajes imperativos, cabe clasificarlos en "orientados a expresiones" y "orientados a sentencias", según jueguen las expresiones o las sentencias un papel más predominante en el lenguaje. Ambos términos son relativos, y ningún lenguaje cuenta únicamente con expresiones y sentencias. Se puede decir que FORTRAN, ALGOL, PASCAL, C, entre otros son lenguajes orientados a expresiones, mientras que COBOL y PL/1 están orientados a sentencias.

Las expresiones se han encontrado útiles principalmente porque son simples y jerárquicas y pueden combinarse uniformemente para construir expresiones más complejas. En resumen, se puede señalar que el paradigma imperativo propicia un entorno de programación procedural, en el que la secuencia de control, el concepto de variable y la operación de asignación son sus componentes fundamentales. Por otra parte el algoritmo es la estructura de información utilizada para definir las acciones de un programa, y las estructuras de datos del tipo variable, array, registro, archivos, etc. son elementos pasivos sobre los que actúa el algoritmo.

Paradigma funcional

El paradigma funcional está basado en el modelo matemático de composición funcional. En este modelo, el resultado de un cálculo es la entrada del siguiente y así sucesivamente hasta que una composición produce el resultado deseado.

No existe el concepto de celda de memoria que es asignada o modificada; sino que, existen valores intermedios que son el resultado de cálculos anteriores y las entradas a subsiguientes cálculos. Tampoco existen sentencias imperativas y todas las funciones tienen transparencia referencial.

La programación funcional incorpora el concepto de función como objeto de primera clase, lo que significa que las funciones pueden ser tratadas como datos

[2] Alonso Amo & Segovia Perez. *Entornos y metodologías de programación*. pág 9

(pueden pasar como parámetros, calculadas y devueltas como valores normales y mezcladas en el cálculo con otras formas de datos).

En este paradigma, se concibe la solución como una composición de funciones. Por ejemplo, para ordenar una lista, se puede diseñar la solución como una concatenación de listas más pequeñas, cada una de las cuales ya está clasificada. Esto reduce el problema a seleccionar las listas más pequeñas. La forma en que se especifican las funciones puede variar. En particular, se pueden especificar procedimentalmente, o matemáticamente mediante su definición sin secuencia de control.

En la solución procedimental se controla explícitamente el orden en que se efectúa el cálculo. Así, al ordenar una lista, primero debemos determinar si la lista está vacía y realizar el tratamiento en el caso de que no lo esté. Este orden explícito de cálculo origina una sobrespecificación que es lo que caracteriza a las soluciones procedimentales.

En la práctica, los lenguajes que soportan esta forma procedimental del paradigma funcional a menudo incluyen construcciones imperativas, tal como la asignación, que destruye el no-efecto de lado de este paradigma y obligan a considerar la secuencia del programa en su construcción.

Un programa funcional se encuentra definido por la ecuación.

Funciones + Estructura de datos = Programas

siendo sus elementos fundamentales:

a) Las funciones, que son reglas de correspondencia o asociación entre miembros de un conjunto (el dominio) y miembros de otro conjunto (el rango). La definición de una función especifica el dominio, el rango, y la regla de correspondencia para la función. por ejemplo: cuadrado (x) = x^2, es una función donde:

   - x es el dominio,
   - x * x es la regla de correspondencia (la multiplicación por sí mismo de cada elemento del dominio),
   - cuadrado(x) es el rango, que corresponde de aplicar los números enteros sobre los números enteros.

b) La recursividad, que es la iteración con asignación no destructiva. El carácter recursivo de esta programación exige disponer de este mecanismo, especialmente utilizado en la construcción de funciones.

c) Las listas, que son las estructuras básicas de almacenamiento de la información. Aunque la programación funcional utiliza también otros tipos de

estructuras, se fundamenta básicamente en el manejo de información simbólica con estructura en lista. Este tipo de programación, la expresión algebraica carece de la relevancia que tiene en la programación imperativa.

Dado que el mecanismo de operación de un programa funcional consiste en la ejecución sucesiva de funciones, unas ya definidas en el software del sistema y otras que se van incorporando al programa, este tipo de programación utiliza un proceso interpretativo.

Por otra parte, es bueno mencionar que mientras que la programación imperativa está fundamentada, normalmente, en lenguajes que utilizan procesos de compilación proporcionando módulos objetos, la programación funcional utiliza lenguajes de tipo intérprete. Los lenguajes funcionales pseudodeclarativos, tipo LISP, HOPE, DAPLEX, etc, constan de:

- Funciones primitivas predefinidas en el lenguaje: de control, de estructuras, aritméticas, etc.
- Formas funcionales para crear y combinar funciones.
- Mecanismos para aplicar los argumentos y producir valores de funciones.
- Objetos de datos, del tipo átomos, listas, etc.

El LISP es un buen representante del paradigma funcional; sin embargo, las numerosas versiones mejoradas de LISP (incluyendo el COMMONLISP), soportan igualmente la programación imperativa, al incluir conceptos tales como: efecto de lado con modificación destructiva de su contenido, tipos de datos con almacenamiento fijo (arrays y otros objetos manipulados básicamente con efecto de lado) y constructores del flujo de control (por ejemplo, iteraciones y manejo de excepciones).

Paradigma lógico

El paradigma lógico asume la definición de un conjunto de hechos tal como "Andrés es padre" y un conjunto de reglas que permiten la deducción de otros hechos. Por ejemplo, dada la regla " V **y**, si **y** es padre, **y** es varón", con el aserto anterior se puede deducir que Andrés es varón. De esta manera, la programación lógica, desde la perspectiva del programador, es una técnica que consiste en expresar apropiadamente todos los hechos y reglas necesarios que definen un problema.

El paradigma lógica fue creado por Robert Kowalski en el Imperial College de Londres, y lo implementó en base a las cláusulas de Horn, que son un subconjunto de la lógica de predicados de primer orden. La notación clausal de

la lógica de predicados combina variables, constantes y expresiones para expresar proposiciones condicionales tales como (en el caso de PROLOG):

casado(x,y):-marido(x,y);esposa(x,y)

que señala que "x está casado con y, si y es el marido de x ó x es la esposa de y". Las cláusulas de Horn son una forma restrictiva de lógica de predicados con una sola conclusión en cada cláusula. La programación lógica es una instrucción de componentes lógicos de un algoritmo, proporcionando el sistema la secuencia de control.

Al separar el control y la lógica, el programa se transforma en un conjunto de declaraciones formales de especificaciones que deben ser correctas por definición por lo que la corrección del programa debería estar probada automáticamente. En este tipo de programación la evaluación empieza definiéndose una meta e intentando probar que dicha meta se ajusta a un hecho o se deduce de alguna regla.

Una meta se deduce de una regla si todos los antecedentes de la regla se verifican con la información existente; para lo cual, estos antecedentes se convierten en nuevas metas que deben equiparse con hechos o resolverse vía otras reglas. El proceso termina cuando todas las submetas han sido probadas. Y la solución final viene determinada al aplicar los resultados intermedios obtenidos a las variables de la meta inicial.

La evaluación así descrita es puramente "declarativa", y asume que si se selecciona una regla: o bien sólo existe esa posibilidad o la regla que se necesita para alcanzar la solución es la que se algún modo ha sido seleccionada. Se alcanza la solución si existe un conjunto apropiado de reglas y sustituciones, tales que aplicando las sustituciones a las reglas se obtiene un conjunto de reglas base (sin variables libres) que permiten deducir la meta desde los hechos conocidos.

El principal problema radica en definir el mecanismo de búsqueda apropiado para seleccionar las reglas, siendo la forma más popular la búsqueda "primero en profundidad" con un proceso de "backtracking". Cuando se precisa seleccionar una regla, este algoritmo selecciona la primera que encuentra, si conduce a un punto muerto, selecciona la segunda, y así hasta que todas las posibilidades han sido probadas.

Mediante un cuidadoso orden de los hechos y reglas, se puede aumentar la eficiencia de la selección, así como su terminación, por lo que esta acción transformaría el paradigma en uno pseudodeclarativo.

La programación lógica se ha convertido hoy en día en una programación de propósito general. La lógica pseudodeclarativa se utiliza para producir programas en la industria informática donde la descripción de hechos y reglas es

apropiada para ciertos tipos de problemas, tales como los de búsqueda con "backtracking" con múltiples soluciones; problemas que se describen de forma natural mediante reglas de producción, como la traducción a lenguaje natural y para la definición e implementación de especificaciones ejecutables mediante prototipos rápidos.

Un programa lógico se estructura como un conjunto de hechos (aserciones o proposiciones) y de reglas lógicas previamente establecidas, que obtienen conclusiones en base a una serie de preguntas o cuestiones lógicas, definidas por la ecuación:

Lógica + Control + Estructura de datos = Programas

Donde:

- La lógica la constituyen las aserciones y las reglas lógicas.
- El control, inherente al sistema, son las estrategias a seguir para investigar las cuestiones lógicas.
- Las estructuras de datos, son los elementos que soportan la base de conocimiento y las cuestiones lógicas.

Los lenguajes lógicos constan fundamentalmente de :

- Predicados para la declaración de aserciones.
- Cláusulas para definir reglas lógicas.
- Mecanismos de búsqueda y reconocimiento para resolver las ecuaciones lógicas.
- Estructuras de datos, tales como variables, listas, etc.

En el caso del lenguaje PROLOG, estos elementos se implementas básicamente de la siguiente manera.

a) Predicados, son instrucciones que son ciertas o falsas. Se pueden formar con un argumento o con varios. Su sintaxis:

predicado (argumento)
predicado (arg 1,arg2,...,arg n)

b) Cláusulas. Definen reglas lógicas, que permiten inferir otros conceptos al aplicarlas a su base de conocimiento que incorpora aserciones y reglas lógicas. Su sintaxis:

<parte-izquierda-regla>:-<parte-derecha>
hombre(x):-varón(x),adulto(x).

c) Mecanismo de búsqueda. En PROLOG, la búsqueda en la base de conocimiento se efectúa de arriba a abajo, utilizando como mecanismo de

búsqueda el algoritmo de "primera búsqueda en profundidad" con aplicación del "backtracking. Las reglas son resueltas de izquierda a derecha.

d) Estructura de datos. Los tipos de datos simples utilizados son
- Constantes. Normalmente incorporan átomos, que son nombres simbólicos, literales y valores enteros.
- Variables, que empiezan en algunas versiones por mayúscula o por subrayado.
- Listas que se representan con la notación entre corchetes, por ejemplo: [a,b,c,d].

Paradigma orientado a objetos

La orientación a objetos está constituyendo una metodología de diseño y desarrollo de software de gran trascendencia para la producción de software eficiente y barato. Esta metodología de análisis, diseño y programación que configura las fases fundamentales del ciclo de vida de un sistema informático, se está asentando como la estructura metodológica de los años 90.

La orientación a objetos se puede definir como "una disciplina de ingeniería de desarrollo y modelado de software que permite construir más fácilmente sistemas complejos a partir de componentes individuales. Además, permite una representación más directa del modelo del mundo real, reduciendo fuertemente la transformación radical normal desde los requerimientos del sistema, definidos en términos del usuario, a las especificaciones del sistema.

Actualmente, la tendencia es la de producir componentes reutilizables para ensamblarlos a otros y obtener así, el producto completo. Estos elementos reutilizables son denominados "Componentes Integrados de Software" (CIS) por su teórica similitud con los "componentes integrados de hardware" (chips), innovación que revolucionó la industria de la computación hace 20 años.
La programación orientada a objetos (POO) es una filosofía de desarrollo y empaquetamiento de software que permite crear unidades funcionales extensibles y genéricas, de forma que el usuario las pueda aplicar según sus necesidades y de acuerdo con las especificaciones del sistema a desarrollar. La orientación a objetos proporciona mejores herramientas para.

- Modelar el mundo real de un modo más cercano a la perspectiva del usuario.
- Interactuar fácilmente con un entorno computacional, usando metáforas familiares.
- Construir componentes reutilizables de software y librerías específicas de estos componentes fácilmente extensibles.
- Modificar y ampliar con facilidad la implementación de estos componentes sin afectar al resto de su estructura.

En cuanto a los elementos fundamentales que configuran el paradigma orientado a objetos, algunos autores, los centran en tres: los tipos abstractos de datos, la herencia y la identidad de los objetos. Mientras que otros señalan siete aspectos básicos a considerar para una verdadera orientación al objeto:

- Estructura modular basada en objetos, dado que los sistemas en esta metodología son modularizados sobre la base de sus estructuras de datos.
- Abstracción de datos, porque los objetos son descritos como implementaciones de tipos de datos abstractos.
- Administración automática de memoria, de forma que los objetos no utilizados sean designados por el propio sistema sin intervención del programador.
- Clases, en las que cada tipo no simple sea un módulo y cada módulo de alto nivel sea un tipo.
- Herencia, que permita que una clase sea definida como una extensión o restricción de otra.
- Polimorfismo y enlace dinámico, que las entidades del programa puedan referenciar en tiempo de ejecución a objetos de diferentes clases.
- Herencia múltiple y repetida para que se pueda declarar una clase como heredera de varias e incluso de ella misma.

Por otra parte, el paradigma orientación a objetos presenta diferente terminología según la etapa del ciclo de vida que se contempla. Así, en las fases de análisis y de diseño, se utilizan los conceptos: identificación del objeto, identificación de las estructuras y materias, definición de elementos, definición de atributos, conexión de instancias, definición de servicios y conexión de mensajes. Mientras que la etapa de programación, los términos a utilizar son: tipo de dato abstracto, objeto, mensaje, clase, sobrecarga, enlace dinámico, polimorfismo, herencia, variables de clase o de instancia, métodos, constructores, destructores, aserción, invariante, etc.

El paradigma orientado a objetos se describe a menudo usando el concepto objeto/mensaje, en el que cada objeto (elemento autónomo de información creado en tiempo de ejecución) es solicitado para realizar un determinado servicio mediante el envío a ese objeto del mensaje apropiado. El solicitante no precisa conocer cómo el objeto proporciona el servicio pedido; la implementación es interna al objeto y la gestiona el suministrador del objeto. El énfasis se produce en qué se puede obtener más bien que en cómo se obtiene. Un programa orientado a objetos se define como:

Objetos + Mensajes = Programa

Donde el objeto es una instancia de una clase, la cual implementa un tipo abstracto de dato (TAD); y el mensaje es la información especifica que se envía al objeto para que ejecute una determinada tarea.

Un TAD define conjuntos encapsulados de objetos similares, con una colección asociada de operaciones; y especifica la estructura y el comportamiento de los objetos. Las especificaciones estructuradas del TAD describen las características de los objetos pertenecientes a ese TAD y las especificaciones de comportamiento describen qué mensajes son aplicables a cada objeto.

Normalmente se implementan en un lenguaje de programación, mediante una clase (class) que es el marco para implementar TAD's y poder crear objetos de la misma estructura. Un objeto del TAD *pila* consta de los siguientes componentes:

- Variables de clase y de instancia (atributos, entidades, ...), que definen el estado interno del objeto. En el caso de la pila es usual utilizar una lista encadenada, aunque también se puede implementar mediante un array.

- Métodos (rutinas, miembros, función, ...) que definen el comportamiento de las instancias y son invocados mediante el envío de mensajes a los objetos. Un mensaje indica el objeto destino, el selector (nombre del método) y los argumentos del operador. En el caso de la pila los métodos descritos son: *crear, insertar, eliminar, cabeza_de_la_pila y es_vacía*.

Por último, cabe señalar que el entorno de programación orientado a objetos es:

a) Procedimental, pues incorpora el concepto de encapsulación de datos.
b) Extensible, ya que permite ampliar los componentes software sin necesidad de reprogramar todo el componente.
c) Reutilizable, por permitir crear un programa en base a una serie de componentes definidos en librerías de componentes.
d) Interactivo, a través de una interfaz gráfica de usuario que proporciona la funcionalidad necesaria en materia de edición puesta a punto de programas y archivo de información.

Paradigma heurístico

La programación heurística se basa en el uso del conocimiento específico del dominio para cubrir una amplia gama de posibilidades, guiando la búsqueda por las direcciones más prometedoras; y se puede definir como "aquel tipo de programación computacional que aplica reglas lógicas para la resolución de problemas, denominadas heurísticas, las cuales proporcionan entre varios cursos de acción uno que presenta avisos de ser el más prometedor, pero no garantiza necesariamente el curso de acción más efectivo".[3]

La programación heurística implica una forma de modelar el problema en lo que respecta a la representación de su estructura, estrategias de búsqueda y métodos de resolución, que configuran el paradigma heurístico.

[3] *Idem*, pág. 27.

Este tipo de programación se aplica con mayor intensidad en el campo de la inteligencia artificial (IA) y en especial, en el de la ingeniería del conocimiento, dado que el ser humano opera la mayor parte de las veces utilizando heurística. Es cierto que la heurística es la conclusión del razonamiento humano en un dominio específico, por lo que es normal que este tipo de programación quede encuadrado en el área de la IA ya que implementa el conocimiento humano, validada por la experiencia, utilizando reglas de buena lógica.
El paradigma heurístico define un modelo de resolución de problemas en el que se incorpora algún componente heurístico en base a:

- Una representación apropiada de la estructura del problema para su resolución con técnicas heurísticas.

- La utilización de métodos de resolución de problemas aplicando funciones de evaluación con procedimientos específicos de búsqueda heurística para la consecución de las metas.

Por otra parte la programación heurística se presenta y utiliza desde diferentes puntos de vista:

- Como técnica de búsqueda para la obtención de metas en problemas no algorítmicos o con algoritmos que generan explosión combinatoria.

- Como un método aproximado de resolución de problemas utilizando funciones de evaluación de tipo heurístico.

- Como método de poda para estrategias de programas que juegan, aunque estos métodos no son realmente heurísticos.

Zanakis[4] aconseja utilizar un modelo heurístico cuando:

- Los datos, limitados e inexactos, utilizados para estimar los parámetros del modelo pueden contener errores inherentes muy superiores a los proporcionados con la solución de una buena heurística.

- Se utiliza un modelo simplificado, que por sí es una representación imprecisa de un problema real, por lo que la solución "óptima" es puramente académica.

- No se dispone de un método exacto que sea fiable para ser aplicado en el modelo del problema; o si existe, es intratable.

[4] *Idem*, pág. 28.

- *Se desea mejorar la eficiencia de un algoritmo optimizador aplicado al modelo; por ejemplo, proporcionando buenas soluciones de inicio, guiando la búsqueda y reduciendo el número de soluciones posibles.*

- *Se tiene la necesidad de resolver el mismo problema frecuentemente, o sobre una base de tiempo real, y el tratamiento heurístico significa un ahorro.*

La programación heurística no ha producido un lenguaje específico de programación, debido a que la heurística al "ser un conjunto de reglas de sentido común", se pueden implementar con cualquiera de los lenguajes descritos en los diferentes paradigmas de programación. Por otra parte, al aplicarse este tipo de programación fundamentalmente, en el campo de la IA la heurística está siendo implementada enormemente en herramientas de esta área.

4.3 Selección del lenguaje

Los lenguajes de programación son un medio de comunicación entre las personas y las computadoras, es por ello que las características del lenguaje afectan directamente a la calidad del software. Las características técnicas de un lenguaje pueden influenciar la calidad del diseño y afectar en el aspecto humano.

Una visión psicológica

Ben Shneiderman[5], en su libro Software Psychology, observó que el papel del psicólogo del software es centrarse en los aspectos humanos tales como la facilidad de uso, la simplicidad de aprendizaje, la mejora en fiabilidad, la reducción de la frecuencia de error y el aumento de satisfacción del usuario, mientras se mantiene una garantía de eficiencia de la máquina, de la capacidad del software y de las restricciones del hardware.

Otro psicólogo del software, Gerard Weinberg, opina que los diseñadores de lenguajes de programación a menudo hacen que tengamos que ajustar nuestras necesidades de un problema de manera que se tengan en cuenta las limitaciones impuestas por ese determinado lenguaje de programación. Debido a que los factores humanos son de importancia crítica en el diseño de lenguajes de programación, las características psicológicas de un lenguaje tienen una fuerte influencia en el éxito del diseño, la traducción de código y la implementación.

Varias características psicológicas aparecen como resultado del diseño de un lenguaje de programación. Aunque esas características no se pueden medir de

[5] Pressman, Roger. *Ingeniería del software*, pág. 448.

forma cuantitativa, se reconoce su manifestación en todos los lenguajes de programación. A continuación se mencionan algunas de esas características.

La uniformidad indica el grado en que un lenguaje usa una notación consistente, aplica restricciones aparentemente arbitrarias o incluye excepciones a reglas sintácticas o semánticas.

La ambigüedad de un lenguaje de programación es percibida por el programador. Un compilador siempre interpreta una sentencia de una única forma, pero el lector puede interpretar la sentencia de formas diferentes. Aquí es donde reside la ambigüedad psicológica. Por ejemplo, la ambigüedad psicológica aparece cuando la precedencia aritmética no es obvia:

$$X = X_1/X_2{}^*X_3$$

Al leer el código fuente se puede interpretar lo anterior como $X = (X_1/X_2)^*X_3$, mientras que otro puede "ver" $X = X_1/(X_2{}^*X_3)$. Otra fuente potencial de ambigüedad es el uso no estándar de identificadores que tienen tipos de datos implícitos.

Una falta de uniformidad y la ocurrencia de ambigüedad psicológica normalmente se dan juntas. Si un lenguaje de programación muestra los aspectos negativos de estas características, el código fuente será menos legible y la traducción desde el diseño más problemática.

Lo compacto que sea un lenguaje de programación es un indicativo de la cantidad de información orientada al código que se debe retener en la memoria humana. Entre los atributos del lenguaje que miden lo compacto que es, se encuentran:

- el grado en que un lenguaje soporta las construcciones estructuradas
- los tipos de palabras clave y de abreviaturas que se pueden usar
- la variedad de tipos de datos y las características implícitas
- el número de operadores aritméticos y lógicos
- el número de funciones incorporadas.

Las características de la memoria humana tienen un fuerte impacto en la forma en que se usa un lenguaje. La memoria y el reconocimiento humano se pueden dividir en dominios *sinestético* y secuencial. La memoria sinestética nos permite recordar y reconocer las cosas como un todo. La memoria secuencial proporciona una forma de reconocer el siguiente elemento de una secuencia. Cada una de estas características de la memoria afecta a las características de los lenguajes de programación denominadas *localización* y *linealidad*.

La linealidad es una característica psicológica que se asocia con el concepto de mantenimiento de un dominio funcional; o sea, la percepción humana se facilita

cuando se encuentra una sec encia lineal de operadores lógicos. Las grandes ramificaciones violan la linealidad del procesamiento. De nuevo, la implementación directa de las construcciones estructuradas ayudan a la linealidad de un lenguaje de programación. Nuestra capacidad para aprender un nuevo lenguaje de programación se ve afectada por la *tradición*.

La tradición también afecta al grado de innovación durante el diseño de un lenguaje de programación. Aunque frecuentemente se proponen nuevos lenguajes, las formas de los nuevos lenguajes evolucionan muy despacio.

Shneiderman desarrollo un *modelo sintáctico/semántico* del proceso de programación que tiene relevancia al considerar el paso de codificación. Cuando un programador aplica los métodos (análisis, diseño, codificación, etc.), que son independientes del lenguaje de programación aprovecha el conocimiento semántico. Por otro lado, el conocimiento sintáctico es dependiente del lenguaje, centrándose en las características de un lenguaje específico.

De estos tipos de conocimiento, el conocimiento semántico es el más difícil de adquirir y el que intelectualmente es más importante de aplicar. El paso de codificación aplica el conocimiento sintáctico debido a que es "arbitrario e instructivo" y aprendido de manera rutinaria. Cuando se aprende un nuevo lenguaje de programación se añade a la memoria nueva información sintáctica. Puede aparecer cierta confusión cuando la sintaxis de otro lenguaje; sin embargo, se debe hacer notar que un nuevo lenguaje de programación puede servir para motivar a aprender la nueva información semántica.

Características de los lenguajes

Una visión general de la ingeniería del software sobre las características de los lenguajes de programación se centra en las necesidades que puede tener un proyecto específico de desarrollo de software. Aunque se podrían derivar algunos requerimientos para el código fuente, se puede establecer un conjunto general de características de ingeniería:

1. facilidad de traducción del diseño al código
2. eficiencia del compilador
3. portabilidad del código fuente
4. disponibilidad de herramientas de desarrollo y
5. facilidad de mantenimiento

El paso de codificación comienza tras haber definido, revisado y modificado el diseño. El grado de *facilidad de la traducción del diseño al código* proporciona una indicación de cómo se aproxima un lenguaje de programación a la representación del diseño, un lenguaje que implemente directamente las construcciones estructuradas, que incluya estructuras de datos sofisticadas, E/S

especializada, posibilidades de manipulación de bits y manejo de cadenas, hará que la traducción del diseñó al código fuente sea mucho más fácil.

La *portabilidad del código fuente* es una característica de los lenguajes de programación que se puede interpretas de tres formas.

1. El código fuente puede ser transportado de un procesador a otro y de un compilador a otro sin ninguna o muy pocas modificaciones.

2. El código fuente permanece inalterado cuando cambia su entorno de funcionamiento.

3. El código fuente puede ser integrado en diferentes paquetes de software sin que prácticamente se requieran modificaciones debidas a las características propias del lenguaje de programación.

De las tres interpretaciones de portabilidad, la primera es la más frecuente. La estandarización (por la ISO) y/o el ANSI continúa siendo el principal esfuerzo para la mejora de la portabilidad de los lenguajes de programación.

La *disponibilidad de herramientas de desarrollo* puede acortar el tiempo requerido para la generación del código fuente y puede mejorar la calidad del código. Muchos lenguajes de programación pueden ser adquiridos con un conjunto de herramientas que incluyen: compiladores con depuradores, ayudas de formato para el código fuente, facilidades de edición incorporadas, herramientas, para control del código fuente, bibliotecas que ayuden al desarrollo de software, etc.

La *facilidad de mantenimiento* del código fuente es importante para cualquier esfuerzo no trivial de desarrollo de software. El mantenimiento no se puede llevar a cabo hasta que no se entienda, ya que, la facilidad de traducción del diseño al código es un elemento importante en la facilidad de mantenimiento del código fuente. Además, las propias características de documentación de un lenguaje tienen una fuerte influencia sobre el mantenimiento.

Elección de un lenguaje

La elección de un lenguaje de programación para un proyecto específico debe tener en cuenta las características mencionadas, aunque, el problema asociado con la elección puede desaparecer si sólo se dispone de un lenguaje o si el cliente demanda uno en particular.

Entre los criterios que se aplican durante la evaluación de los lenguajes disponibles están.

1. Área de aplicación general
2. Complejidad algoritmica

3. Entorno en el que se ejecutará el software
4. Consideraciones de rendimiento
5. Complejidad de las estructuras de datos
6. Conocimiento de la plantilla de desarrollo de software
7. Disponibilidad de un buen compilador

4.4 Codificación

Todos los pasos que se han presentado en los capítulos anteriores han tenido un sólo objetivo: traducir los requerimientos del software a una forma que pueda ser comprendida por una computadora, llegándose al paso de codificación que es un proceso que transforma el diseño en un lenguaje de programación; sin embargo, hay que considerar, que tanto las características del lenguaje de programación como el estilo de programación pueden afectar profundamente a la calidad y al mantenimiento del software.

El paso de codificación traduce una representación del software dada por un diseño detallado a una realización en un lenguaje de programación. El proceso de traducción continúa cuando un compilador acepta el *código fuente* como entrada y traduce como salida un *código objeto* y posteriormente la salida del compilador es traducida a *código máquina*.

El paso inicial de codificación (del diseño detallado al lenguaje de programación) es un punto fundamental dentro del contexto de ingeniería del software; ya que, la interpretación equivocada de las especificaciones del diseño detallado puede conducir a un código fuente erróneo. La complejidad o las restricciones de un lenguaje de programación pueden conducir a un código fuente muy confuso que resulte difícil de probar y mantener.

Las características del lenguaje tienen un impacto directo sobre la calidad y la eficiencia de la codificación (también llamada *programación*). Generalmente la programación depende de la habilidad individual, la atención al detalle y el conocimiento de cómo utilizar los instrumentos disponibles de la mejor manera. La buena programación es un proceso independiente del lenguaje. Los lenguajes de alto nivel simplifican el proceso de convertir un diseño en una aplicación, debido a las características que ofrecen.

El proceso de desarrollo de un programa a partir del diseño puede enfocarse de dos formas: el desarrollo *descendente* y el *ascendente*. Si el programa se considera como una jerarquía de componentes (módulos), el desarrollo descendente implica iniciar en el tope de la jerarquía y trabajar hacia abajo, mientras que en el desarrollo ascendente se empieza en el nivel más bajo y se prosigue hacia arriba, como se muestra en la siguiente figura:

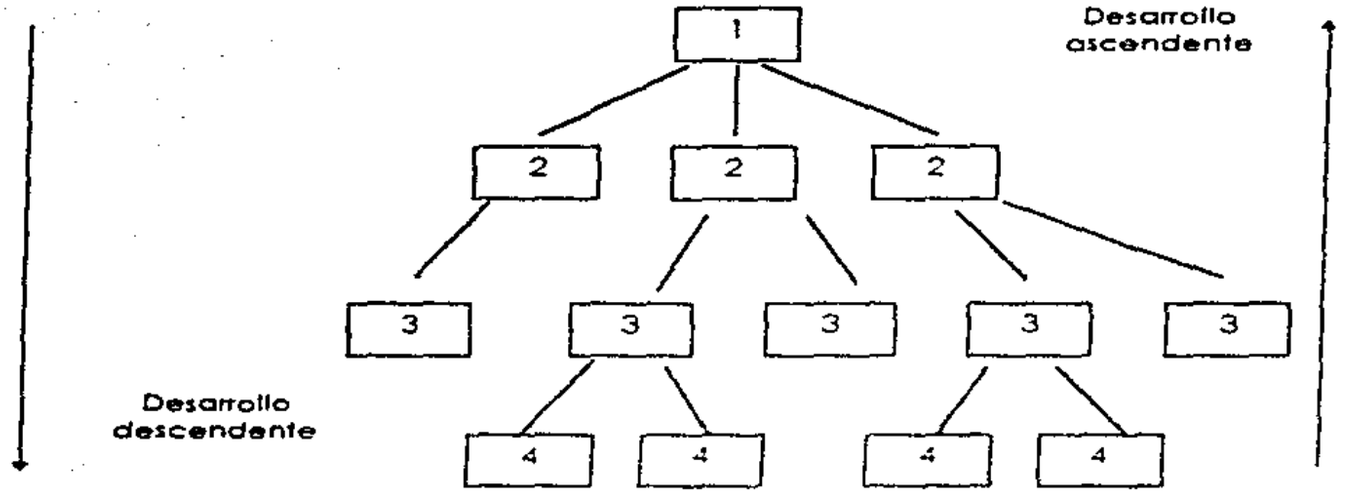


Figura 1 Desarrollo descendente y ascendente

El desarrollo descendente es análogo al proceso descendente que debe seguirse en el diseño del sistema cuando la estructura del programa es jerárquica. Generalmente, se elaboran o aplican los niveles más altos del diseño y se representan los niveles inferiores con módulos que simulan su función de manera simplificada. Al acabar la aplicación de un nivel, el programador pasa al siguiente nivel inferior y lo aplica a partir de sus subniveles. Por último, el nivel más bajo del sistema se aplica mediante el empleo de las características básicas del lenguaje de programación.

El desarrollo ascendente es el inverso de este proceso. La aplicación comienza con los niveles más bajos del sistema y éste se construye hacia arriba hasta alcanzar el nivel más alto del diseño. El programador crea los componentes básicos y los utiliza para construir bloques más complejos, que a su vez se emplean para los niveles más altos del sistema.

Autores como Wirth, Dijkstra y Naur[6] sostienen que el desarrollo descendente es una metodología superior porque da como resultado la creación de programas más legibles y confiables que los aplicados con técnicas ascendentes. El desarrollo ascendente, se dice, tiende a producir mejoras locales a costa de la calidad del sistema, pues el programador nunca tiene la oportunidad de ver al

[6] Sommerville, I. *Ingeniería del software*, pag. 118

sistema como un todo, sino que siempre se le presenta como una colección de partes.

4.5 Interfaz del usuario

La interfaz del usuario es la vara con la que el usuario mide la calidad del sistema. Si un sistema tiene una interfaz mal diseñada, está expuesto al rechazo, independientemente de las posibilidades que ofrezca. Una interfaz mal diseñada puede hacer que el usuario cometa errores potencialmente graves. Si la información se presenta de una forma confusa o engañosa, el usuario puede interpretar mal el significado de un elemento de información y a partir de ahí iniciar una secuencia de acciones peligrosas.

Las interfaces del usuario se pueden clasificar en interfaces interactivas en linea y en interfaces fuera de linea[7]. En una interfaz en línea, el usuario interactúa en forma directa con el sistema a través de su computadora. Una interfaz fuera de línea, se basa en que el usuario prepara por separado una entrada legible para la máquina y después la presenta al sistema. Los resultados de procesar esa entrada se devuelven al usuario a menudo en forma de reporte. Podríamos considerar otro tipo de interfaz de usuario, en la cual el usuario interactúa con el sistema de acuerdo a las opciones que éste presente y las necesidades que requiera el usuario.

El diseño de una interfaz adecuada requiere conocer tanto los procedimientos del usuario como presentar un modelo de lo que se entendió, de las necesidades del usuario.

La utilización de una interfaz de diseño pobre o inadecuada puede tener graves consecuencias. En el mejor de los casos, una interfaz inadecuada producirá un costo de adiestramiento elevado y una mayor proporción de errores de usuario y en el peor de ellos, dará como resultado que se abandone completamente el sistema.

---

[7] *Idem*, pág. 251.

CAPÍTULO 5

DICCIONARIO DE TÉRMINOS ADMINISTRATIVOS

5.1 Introducción

Este capítulo presenta el desarrollo del sistema Diccionario de términos administrativos aplicando cada una de las metodologías descritas en los capítulos anteriores.

La razón por la cual se expuso primero las metodologías a seguir, así como las consideraciones necesarias para elegir las herramientas necesarias para desarrollar software (aplicaciones) es que considero importante señalar inicialmente el fundamento sobre el cual esta basado todo mi diseño, posiblemente no sea la mejor recopilación de información pero contiene los pasos necesarios para llevar a cabo un buen desarrollo.

Primeramente se expone cómo surge la idea de desarrollar un diccionario para computadoras personales y la interfaz empleada para desarrollarlo, seguidamente del análisis y diseño estructurado, así como del análisis de bases de datos.

5.2 Planteamiento del problema

La idea de realizar esta aplicación surgió por parte del Lic. Benjamín Franklin Fincowsky quien es el autor del diccionario que lleva por nombre *"Diccionario para la administración de negocios"*. El desarrollo de un diccionario para computadoras personales persigue la unificación de conceptos que pueden ser empleados por cualquier estudiante o persona interesada en el área administrativa y contable.

El sistema a desarrollar se pretendió fuera una aplicación que trabajara en ambiente Windows para que fuera fácil de interactuar con otras aplicaciones que el usuario utilizara frecuentemente (como Word, Excel, Power Point, WordPerfect, Quattro Pro, dBase, Lotus, entre otras); además, porque Windows está predominando de manera general entre los usuarios de computadoras personales.

La objetivo fundamental de esta aplicación es proporcionar al usuario una herramienta a través de la cual le sea fácil efectuar búsquedas de palabras, muy similar a la de un diccionario como texto, pero con la ventaja de poder actualizar su diccionario de manera personal.

5.3 Herramientas de desarrollo

La herramienta elegida para el desarrollo de esta aplicación fue Visual Basic 3.0, porque es uno de los lenguajes de desarrollo diseñado para crear aplicaciones gráficas de forma rápida y sencilla; proporcionando una gran fuente de utilerias para diseños gráficos y un lenguaje de alto nivel.

Visual Basic está centrado en dos tipos de objetos, ventanas y controles que permiten diseñar con el mínimo de código permitiendo un mecanismo de comunicación (interfaz) para una aplicación; además de incluir soporte para intercambio de datos con otras aplicaciones (DDE - dynamic data exchange) y soporte para establecer enlaces con Windows y con rutinas escritas en otros lenguajes (DLL - dynamic link libraries).

Teniendo como lenguaje de programación Visual Basic, se utilizó Foxpro 2.5 como manejador de base de datos, debido a la fácil conexión con Visual Basic, no se utilizó Access, que es el manejador de bases de datos nativo de Visual Basic, porque no se obtuvieron las librerías necesarias para accesar a la versión 2.0 de Access desde Visual Basic 3.0

Para crear los discos de instalación de empleo el programa Freeman Installer 2.99, este programa instalador crea los discos riecesarios para instalar el diccionario en cualquier computadora personal, ésto incluye la creación de un directorio ***(dicwin)*** en el cual se graba tanto el archivo ejecutable y las archivos que contienen la base de datos. Hace una copia en el directorio c:\windos\system de las librerías necesarias para que Visual Basic haga la conexión con los archivos de FoxPro.

Como herramienta de desarrollo para el análisis y diseño se empleo el análisis y diseño estructurado de Edward Yourdon, debido a que es una de las metodologías más eficientes para el desarrollo de sistemas. A partir del siguiente punto se describen los pasos a seguir para el desarrollo de sistemas con el uso de dicha metodología.

5.4 Objetivo

Proporcionar información veraz y oportuna para adquirir conocimientos que ayuden en el proceso de la toma de decisiones.

5.5 Lista de eventos

1. Usuario solicita la definición de una palabra.
2. El sistema proporciona la definición correspondiente a la palabra solicitada.
3. El usuario actualiza el diccionario agregando, eliminando o modificando las palabras y/o definiciones.

5.6 Diccionario de datos

PALABRA	=	*conjunto de letras que designan una cosa o idea*
ÍNDICE	=	1{letra}
LETRA	=	*lista del abecedario latino*
TEMA	=	1{tópico}
TÓPICO	=	*asunto relativo a algo en particular*
CONSULTA_DICCIONARIO	=	palabra+indice+tema
PROPORCIONA_DEFINICION	=	consulta_palabra+topico+idioma+grupo
GRUPO	=	*palabras asociadas entre sí, de acuerdo a su definición*
IDIOMA	=	[ingles | español | frances]
CONSULTA_PALABRA	=	palabra+definicion
DEFINICIÓN	=	*explicacion del significado de una palabra*
ACTUALIZA_DICCIONARIO	=	agregar+eliminar+cambiar
AGREGAR	=	*aumentar el diccionario con una nueva palabra*
CAMBIAR	=	[texto | definicion]
ELIMINAR	=	*disminuir el diccionario en una palabra*
PALABRA_SELECCIONADA	=	*palabra seleccionada por el usuario para ser buscada por el sistema*
LISTA_PALABRAS	=	*conjunto de palabras*
INDICE_PALABRA	=	*conjunto de palabras que inician con una letra especifica*

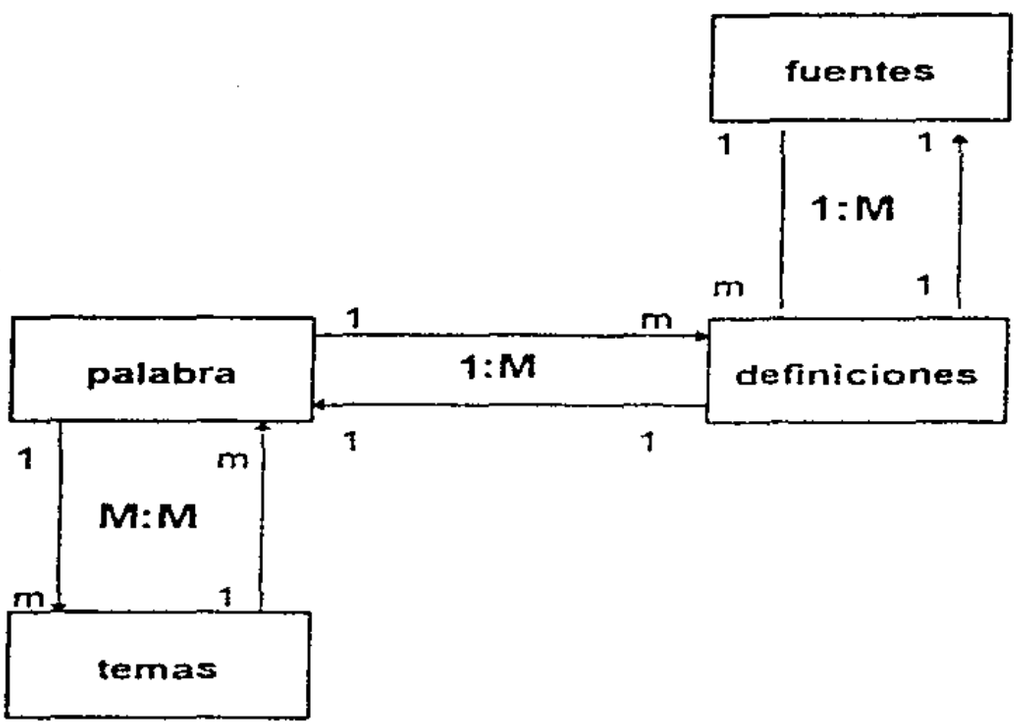
TEMA_PALABRA -- "conjunto de palabras que pertenecen a un tema especifico"

5.7 Diseño de Bases de Datos

Las entidades identificadas para el diseño de la base de datos son las siguientes:

- palabras
- definiciones
- temas
- fuentes

- **Asociaciones :**



Palabras - definiciones: Una palabra puede tener una o más definiciones pero una definición sólo puede tener una palabra.

Palabras - temas: Una palabra puede encontrarse en uno o más temas y un tema puede tener una o muchas palabras.

Definiciones - fuentes. Una definición puede haber sido citada por una autor y un autor puede citar más de una definición.

Tablas y atributos

Palabra

cve_pal	palabra

PK

Definición

cve_def	definición	cve_fuente	cve_pal

PK

Temas

cve_tema	tema

PK

TemaPal

cve_tema	cve_pal

PK

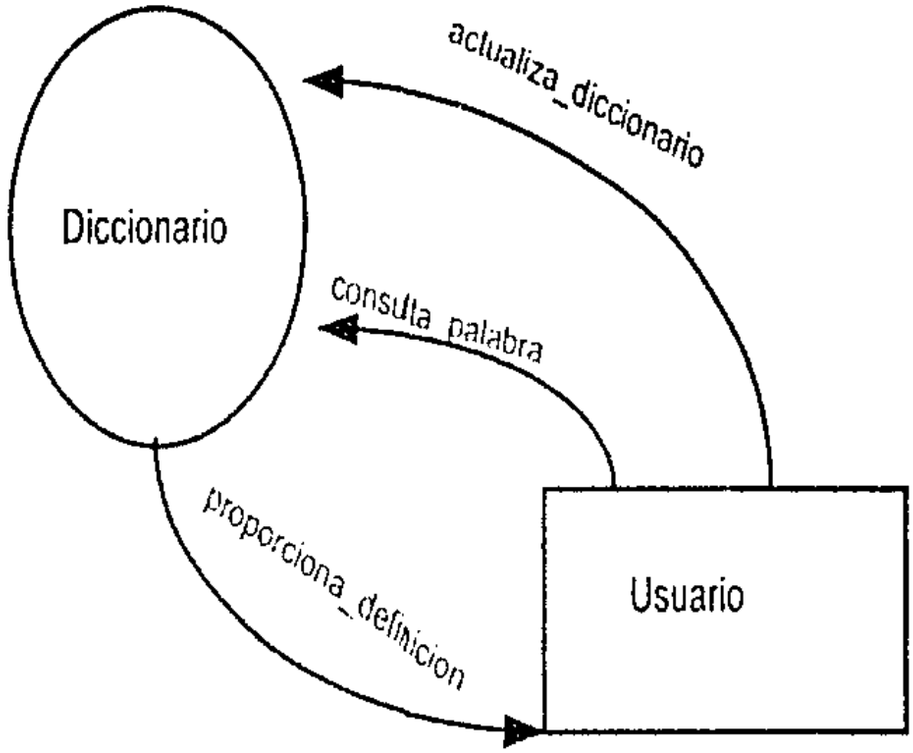
Fuente

cve_fuente	fuente

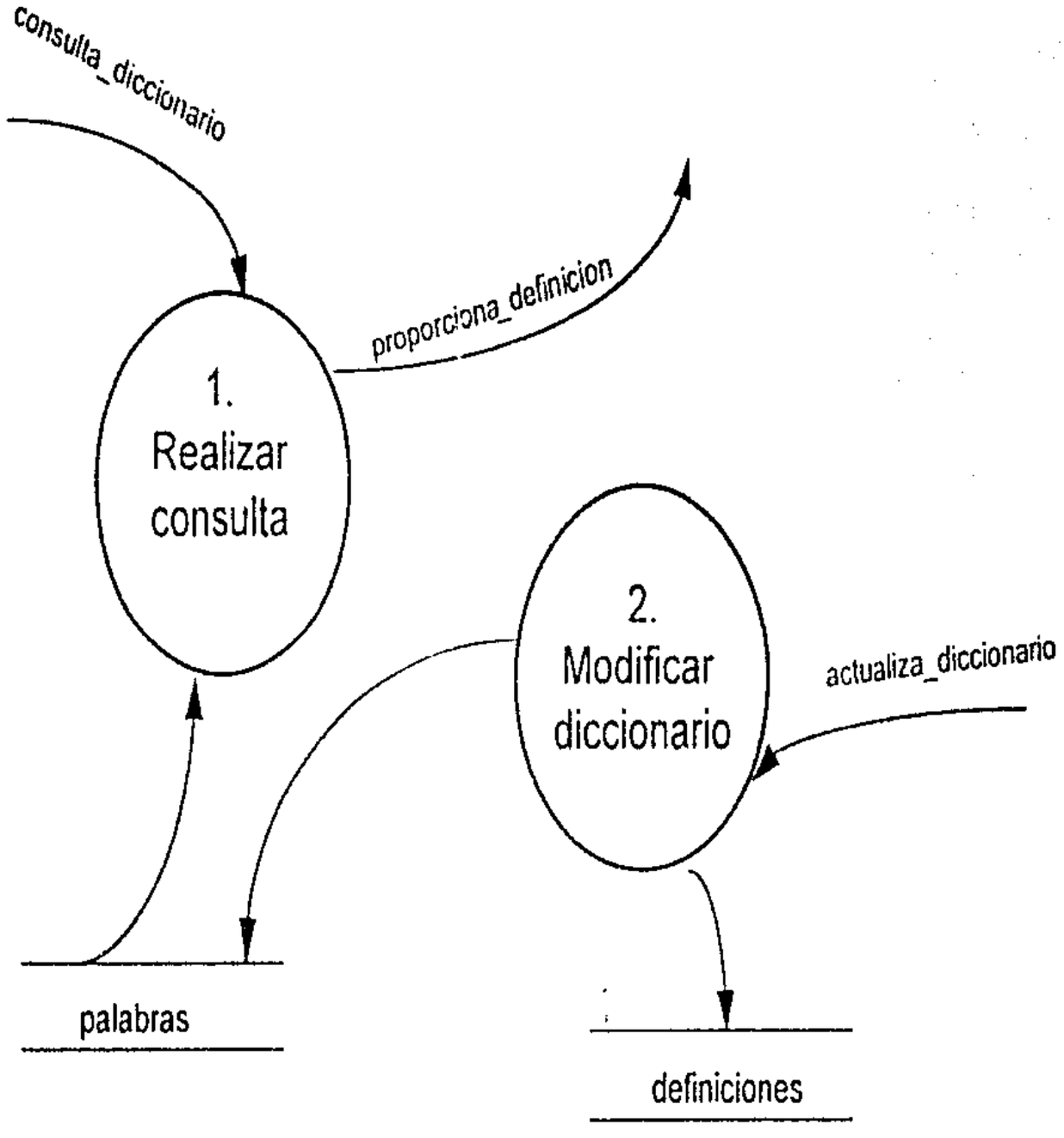
PK

5.8 Diagramas de flujo de datos

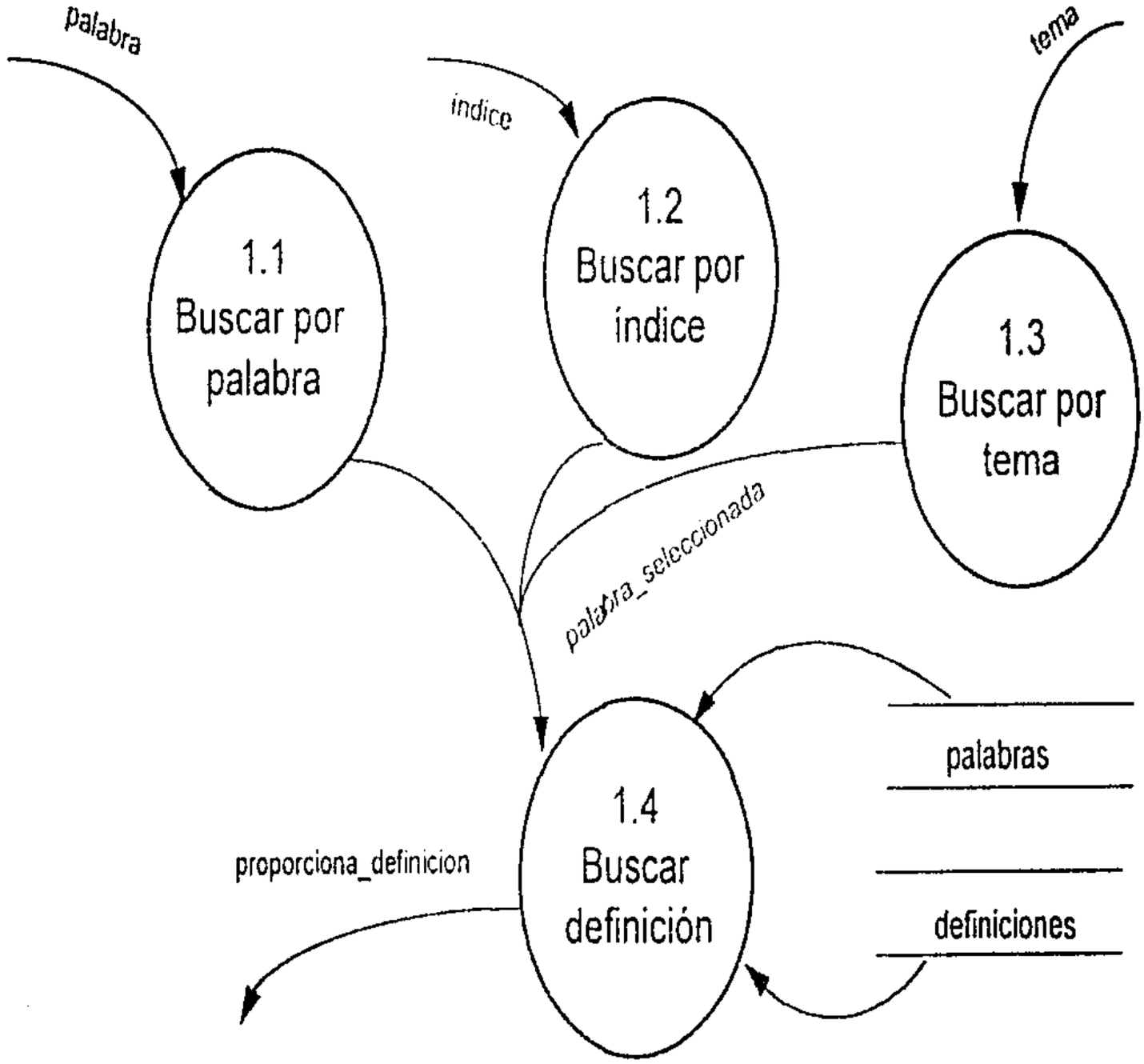
Diagrama de contexto



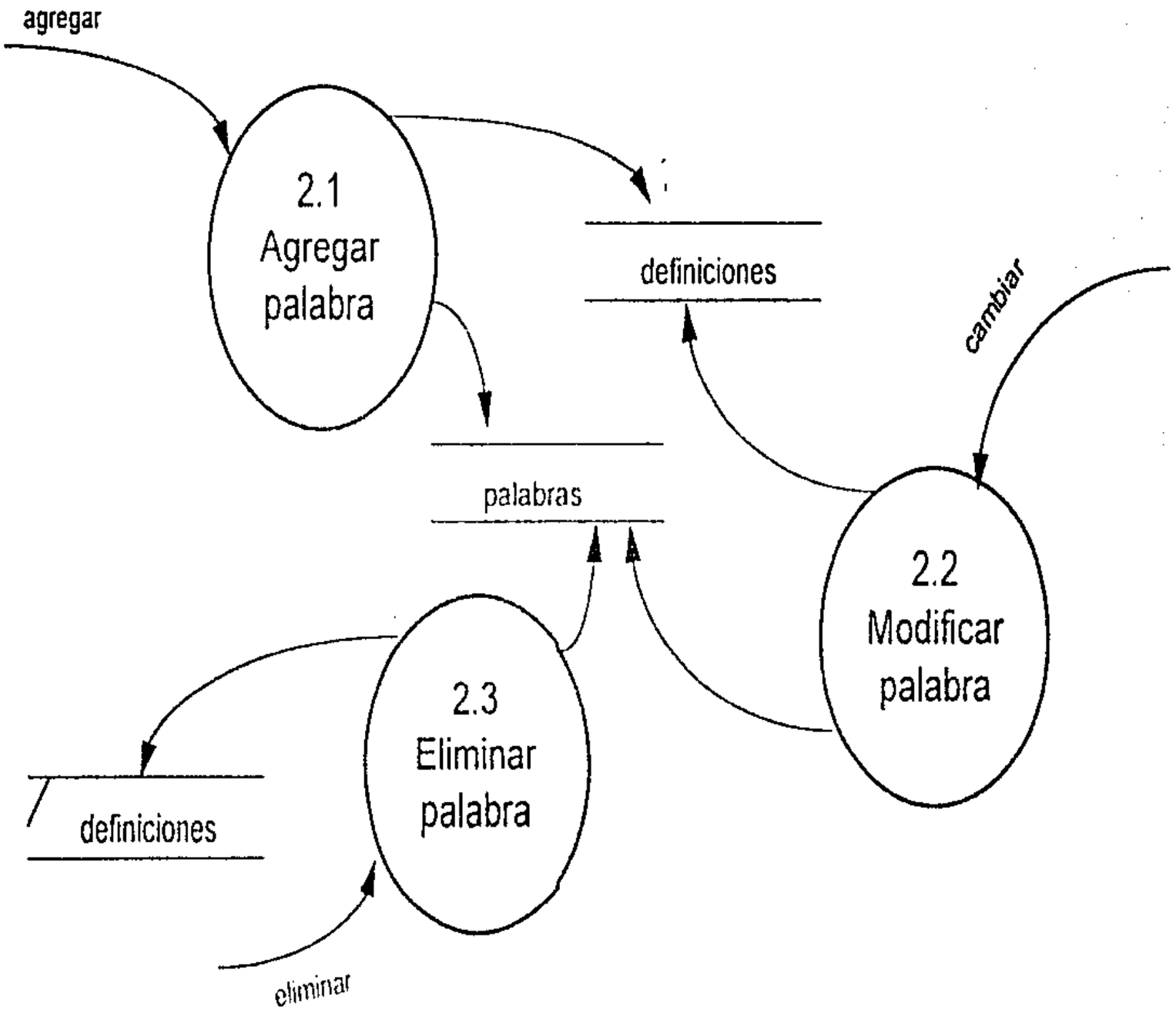
DFD 0


consulta_diccionario
proporciona_definicion
1.
Realizar
consulta
2.
Modificar
diccionario
actualiza_diccionario
palabras
definiciones

DFD 1
Realizar consulta



DFD 2
Modificar diccionario



5.9 Miniespecificaciones

1.1 Buscar por palabra

Para cada palabra proporcionada por el usuario hacer:

1. Seleccionar del archivo **palabras** todas aquellas palabras que cumplan con la siguiente condición:

   *SELECT **
   FROM palabras
   WHERE palabra like %variable%

2. Mostrar el conjunto de palabras, resultado de la búsqueda anterior.

1.2 Buscar por indice

Por cada letra del alfabeto seleccionado por el usuario hacer:

1. Buscar en el archivo **palabras** todas aquellas palabras cuya letra inicial sea igual a la letra seleccionada por el usuario.

```
SELECT *
FROM palabras
WHERE substring(palabra,1,1) like 'A'
```

2. Mostrar el conjunto de palabras que resulte de esta búsqueda en una lista.

1.3 Buscar por tema

Para cada tema seleccionado por el usuario hacer:

1. Buscar en el archivo **tomapal** las palabras que correspondan al tema seleccionado de acuerdo a la siguiente condición:

```
SELECT T1.cve_palabra, T1 palabra
FROM palabras T1, temapal T2
WHERE T1.cve_palabra = T2.cve_palabra
AND T2.cve_tema = variable_tema
```

2. Mostrar el conjunto de palabras que resulte de esta búsqueda en una lista.

1.4 Buscar definición

Para cada palabra seleccionada a buscar:

1. Obtener su clave al momento de seleccionarla.

2. Buscar en el archivo **palabras** todas las definiciones cuya clave de palabra sea igual a la clave de la palabra seleccionada; además buscar en el archivo **fuentes** el autor que cita dicha definición

```
SELECT T1.definicion, T2.autor
FROM definicion T1, fuentes t2
WHERE T1.cve_palabra = variable_palabra
AND T1.cve_autor = T2.cve_autor
```

3. Buscar con la clave de la palabra seleccionada en el archivo **temas**, los temas en donde se encuentra contenida la palabra.

```
SELECT T1.tema
FROM temas T1, temapal T2
WHERE T2.cve_palabra = variable_tema
AND T2.cve_tema = t1.cve_tema
```

4. Mostrar la palabra seleccionada sus posibles definiciones (con sus respectivas fuentes) y los temas en los que se encuentra contenida la palabra.

2.1 Agregar palabra

Para cada una de las nuevas palabras:

1. Guardar la palabra de el archivo **palabra** y asignarle una clave única

2. Guardar cada una de las definiciones de la palabra en el archivo **definicion**, asignándole una clave única a cada definición , guardar la clave de la palabra a la cual pertenece y almacenar la clave del autor que cita dicha definición

3. Almacenar en la archivo **palabras** la clave de la palabra y la clave del tema al cual se este relacionando la palabra.

2.2 Modificar palabra

Para modificar el contenido de la definición de una palabra:

1. Mostrar todas las palabras existentes.

2. El usuario seleccionará una de ellas para poder modificar el contenido de la palabra, así como sus definiciones y temas.

3. Guardar la palabra modificada en el archivo **palabras**.

4. Guardar las modificaciones a las **definicion**.

5. Guardar las relaciones con los temas de la palabra en el archivo **temapal**.

2.3 Eliminar palabra

Para borrar una palabra del diccionario:

1. Borrar la palabra seleccionada y su clave del archivo **palabras**.

2. Borrar del archivo **definicion** todas las definiciones cuya clave de palabra pertenezcan a la palabra seleccionada.

3. Borrar del archivo **temapal** todas las claves de palabra que pertenezcan a la palabra a borrar.

Conclusiones

En una sociedad tan cambiante, en la que las formas de operación (procesos) constituyen la base de las organizaciones, la informática surge no tan solo como una herramienta de desarrollo, sino como una herramienta de análisis que ayuda a detectar inconsistencias en los procesos de operación de las organizaciones.

Para llegar a un análisis a nivel de procesos es necesario conocer una metodología que nos ayude a identificar el medio ambiente en el cual se opera y poder especificar las necesidades y requerimientos a solucionar.

Como profesionistas debemos de brindar soluciones alternativas que mejoren los proceso manuales, pero esto se logra eficientemente, si primeramente se hace un análisis administrativo del desarrollo actual del proceso y posteriormente, teniendo conocimiento de la operación se elabora un análisis del sistema; éste análisis debe basarse en una metodología que conlleve a la solución del problema.

El análisis y diseño estructurado de Edward Yourdon, actualmente es el más difundido y podría decirse que el más simple en cuanto a las etapas involucradas para su desarrollo. El propósito de desarrollar sistemas bajo la guía de una metodología, es la de reducir el margen de error en la interpretación de lo que debería hacer el sistema, involucrarse con el mismo sistema, y tener una idea más precisa al momento de programar.

Otro aspecto importante durante el desarrollo, es el análisis de las base de datos, en el caso del sistema Diccionario, su base de datos no era demasiado grande; sin embargo, actualmente los volúmenes de información son cada vez más grandes, lo cual involucra, diseñar base de datos que soporten estos volúmenes de información y que además operen eficientemente. Aunado a esto, hay que poner atención sobre las herramientas (lenguajes de programación que se va a emplear para poder optimizar los recursos con los que se cuentan.

Bibliografía

ALONSO, F. & Segovia F.J. Entornos y metodologías de programación. 1a. ed. Ed. Paraninfo, España, 1995, 463 p.

BARRIOS, V. Marisela. Introducción a las bases de datos. 1a ed. Editado por DGSCA-UNAM, México, 1995, 86 p.

BATINI, Ceri Navathe. Diseño conceptual de bases de datos. 1a. ed. Ed. Addison Wesley, México, 1995, 545 p.

BISLAND, Ralph. Database management. 2a ed. Ed. Mc Graw Hill, Arizona USA,1993, 419 p.

BURCH, John & Grudnitsk, Gary. Diseño de sistemas de información. 1a. ed. Ed. Grupo Noriega Editores, México, 1992. 985 p.

CORNELL, Gary. Manual de Visual Basic 3 para Windows. 1a ed. Ed.. Mc Graw Hill, México, 1995, 792 p.

DE AMESCUA Seco Antonio & García Sánchez Luis. Ingeniería del software de gestión: análisis y diseño de aplicaciones. 1a. ed. Ed. Paraninfo, España, 1995, 327 p.

HAMMER, Michael & Champy, James. Reingeniería. 1a ed. 5a reimpresión, Ed. Norma, Colombia, 1994, 226 p.

HICKS, Herbert G. Administración de organizaciones. 2a. ed. De. McGrawHill, México, 587 p.

JENNINGS, Roger. Database developer's with Visual Basic 3, 1a. ed. Ed. SAMS Publishing, Indianapolis USA, 1994, 1133p.

MANN, Anthony. Programming with Visual Basic. 1a. ed. Ed. . SAMS Publishing, Indianapolis USA, 1995, 884 p.

MURDICK, Robert & Munson, John. Sistemas de información administrativa. 2a ed. Ed. Prentice Hall, México, 1986, 722p.

PRESSMAN, Roger S. Ingeniería del software. 2a de. De Mc Graw Hill, México, 1990, 628 p.

SOMMERVILLE, I. Ingeniería de software. 2a ed. Ed. Addison Wesley Iberoamericana, México, 1988, 362 p.

STONER, James. Administración. 3a. ed. Ed. McGrawHill, México, 1989, 826 p.

YOURDON, Edward. Análisis estructurado moderno. 1a ed. Ed. Prentice Hall, México, 1993, 735 p.

Apéndice A
Procedimientos básicos

El código que se muestra en este apéndice corresponde a algunos de los principales procedimientos que se emplearon en el programa de Visual Basic. Debido al extenso código generado sólo se muestran algunos de los procedimientos para llevar a cabo un proceso mayor.

Procedimientos para actualizar la definición de las palabras y la asociación con otros temas.

```
Sub Form_Load ()
   Form8.Move 720, 290, 8010, 5625
   FormaActivo = 3
   label1.Caption = "Palabras existentes:"
   text1.Visible = False
   Text2.Visible = False
   Anterior.Visible = False
   Siguiente.Visible = False
   Agregar.Visible = False
   Aceptar.Visible = False
   ModifTema.Visible = False
   SQLst = "Select * From TabPal Order by palabra"
   Set db = OpenDatabase(CurDir$, False, False, conn$)
   Set MiSet = db.CreateDynaset(SQLst)
   Do While Not MiSet.EOF
     list1.AddItem MiSet!palabra
     list1.ItemData(list1.NewIndex) = MiSet!cve_pal
     MiSet.MoveNext
   Loop
End Sub


Sub ModificaDef ()
   definiciones(Sig) = Text2.Text  'Guarda los cambios en la definicion actual
   MisetDef.MoveFirst
   For i = 1 To Numreg
      If MisetDef!definicion <> definiciones(i) Then
         MisetDef.Edit
           MisetDef!definicion = definiciones(i)
         MisetDef.Update
      End If
      If definiciones(i) = "" Or Len(definiciones(i)) <= 5 Then
         MisetDef.Delete
      End If
      MisetDef.MoveNext
   Next i
   If NuevoNumReg > Numreg Then   ' Si se agregaron nuevas definiciones
     Set mitabla = db.OpenTable("TabDef")
     For j = Numreg + 1 To NuevoNumReg
        If definiciones(j) <> "" Then
           mitabla.MoveLast
           clavedef = mitabla!cve_def + 1
           mitabla.AddNew
             mitabla("cve_def") = clavedef
             mitabla("definicion") = definiciones(j)
             mitabla("cve_pal") = Clave
           mitabla.Update
        End If
     Next j
     mitabla.Close
   End If
End Sub
```

```
Sub ModificaPal ()
    If text1.Text <> PalActualizar Then
      PalModificada = text1.Text
      Call ConverMayus(PalModificada)
      MiSet.Bookmark = MarcaPal
      MiSet.Edit
        MiSet!palabra = PalModificada
      MiSet.Update
    End If
End Sub


Function VerModif () As Integer
  MisetDef.MoveFirst
  If MisetDef!definicion <> Text2.Text Then
    VerModif = 1
    Exit Function
  End If
  For i = 1 To NuevoNumReg
    If MisetDef!definicion <> definiciones(i) Or NuevoNumReg > Numreg Then
      VerModif = 1
      Exit Function
    Else
      VerModif = 0
    End If
    MisetDef.MoveNext
  Next I
End Function
```

Procedimientos para eliminar una palabra de la base de datos, (tanto la palabra como su definición o posibles definiciones y pertenencia a uno o más temas)

```
Sub Form_Load ()
    Form7.Move 890, 952, 7830, 4500
    CerrarMiset = 0
    PalSeleccionada = 0
    Etiqueta.Visible = False
    Combo1.Visible = False
    ventana = 4
End Sub


Sub OpPal_Click (Value As Integer)
    PalSeleccionada = 0
    list1.Clear
    Combo1.Visible = False
    Etiqueta.Visible = True
    Set db = OpenDatabase(CurDir$, False, False, conn$)
    Set MiSet = db.CreateDynaset("Select * from TabPal order by palabra")
    Do While Not MiSet.EOF
      list1.AddItem MiSet!palabra
      MiSet.MoveNext
    Loop
    MiSet.Close
    db.Close
    list1.SetFocus
    CerrarMiset = 0
End Sub


Sub Aceptar_Click ()
    Call ProcesoBorrar
End Sub


Sub Elimina ()
    ClavePal = MiSet!cve_pal    'Guarda la clave de la palabra
    MiSet.Delete                'Borra la palabra de la tabla TabPal
    MiSet.MoveNext
    MiSet.Close                 'Cierra la tabla

    'Borra las definiciones de la palabra correspondiente
    SQLst = "Select * from TabDef where cve_pal=" + ClavePal + ""
    Set MiSet = db.CreateDynaset(SQLst)
    Do While Not MiSet.EOF
      MiSet.Delete
      MiSet.MoveNext
    Loop
    MiSet.Close
    'Borra la relacion entre la palabra y el tema al que pertenece
    SQLst = "Select * from TemaPal where cve_pal=" + ClavePal + ""
    Set MiSet = db.CreateDynaset(SQLst)
    Do While Not MiSet.EOF
      MiSet.Delete
      MiSet.MoveNext
    Loop
End Sub
```

```
Sub ProcesoBorrar ()
   Dim NL$, msg As String
   Dim Aceptar As Integer
   Dim VarPalabra As String
   If PalSeleccionada = 1 Then
     NL$ = Chr$(13) + Chr$(10)
     msg = "¿Está seguro de querer eliminar" + NL$ + "la palabra seleccionada?"
     Aceptar = MsgBox(msg, 0 + 1, "¡ Precaución !")
     If Aceptar <> 2 Then        'El valor 2 es para cancelar
       Form7.MousePointer = 11
       VarPalabra = list1.List(list1.ListIndex)
       Set db = OpenDatabase(CurDir$, False, False, conn$)
       SQLst = "Select * from TabPal where palabra = '" & VarPalabra & "'"
       Set MiSet = db.CreateDynaset(SQLst)
       Call Elimina
       MiSet.Close
       db.Close
       Form7.MousePointer = 0
       ventana = 0
       Unload Me
     End If
   Else
     Beep
     MsgBox "Seleccione una palabra de la lista", , "¡ Cuidado !"
   End If
End Sub
```

Procedimientos para hacer una búsqueda que muestre todas las palabras referentes a una letra del alfabeto.

```
Sub Indice (SQLst As String)
   Form3.MousePointer = 11
   PalSeleccionada = 0
   list1.Clear
   SQLst = " SELECT * FROM TabPal WHERE ucase(left(TabPal!palabra,1)) ='" & letra & "'
order
    by palabra"
   Set db = OpenDatabase(CurDir$, False, False, conn$)
   Set MiSet = db.CreateDynaset(SQLst)
   If MiSet.EOF And MiSet.BOF Then
      MsgBox "No hay palabras registradas con esa letra", , ""
   Else
      MiSet.MoveLast
      max = MiSet.RecordCount
      MiSet.MoveFirst
      For i = 1 To max
         list1.AddItem MiSet!palabra
         MiSet.MoveNext
      Next i
   End If
   MiSet.Close
   db.Close
   Form3.MousePointer = 0
End Sub


Sub VerSignificado ()
   ventana = 1 'el 1 es para saber que la ventana a cerrar es la de indice Form3
   If PalSeleccionada = 1 Then
      MDIForm1.MousePointer = 11
      buscar = list1.List(list1.ListIndex)
      Call BuscarPalabra(buscar)
      Form3.Enabled = False
      form1.Show
   Else
      Beep
      MsgBox "Seleccione una palabra de la lista", , "¡ Cuidado !"
      list1.SetFocus
   End If
End Sub


Sub LetraA_Click ()
   letra = "A"
   Call indice(SQLst)
   list1.SetFocus
End Sub
```

Procedimientos para guardar una nueva palabra en la base de datos, (palabra, definición(es) y temas).

```
Sub Aceptar_Click ()
   No_definicion(j) = Text2.Text
   If (j = 0 And No_definicion(j) = "") Or Text1.Text = "" Or Not SelTema Then
      Beep
      NL$ = Chr$(13) + Chr$(10)
      msg = "Faltan datos para poder " + NL$ + "registrar la palabra "
      Call EnviaMensaje(msg, "¡ Error !")
   Else
      MDIForm1.MousePointer = 11
      cont = 0
      Nueva_Palabra = Text1.Text
      Call ConverMayus(Nueva_Palabra)
      'Revisa la existencia de la palabra en la base de datos
      Call BuscarPalabra(Nueva_Palabra)
      If Not miset.EOF And Not miset.BOF Then
         MDIForm1.MousePointer = 0
         Beep
         Call EnviaMensaje(" La palabra ha sido dada de alta", " ¡ Cuidado !")
         miset.Close
      Else
         miset.Close
         Call GuardarPalabra
      End If
      FormaActiva = 0
      MDIForm1.MousePointer = 0
      Unload Me
   End If
End Sub


Sub Cancelar_Click ()
   If i > 0 Or No_definicion(i) <> "" Then
      Respuesta = MsgBox("¿ Desea abandonar los cambios ?", 0 + 1, "")
      If Respuesta <> 2 Then        'Valor 2 = Cancelar
         FormaActiva = 0            'Valor 1 = Aceptar
         Unload Me
      End If
   Else
      FormaActiva = 0
      Unload Me
   End If
End Sub


Sub GuardarPalabra ()
   'Registra la nueva palabra en la tabla TabPal
   Set mitabla = db.OpenTable("TabPal")
   mitabla.MoveLast
   clavepal = mitabla!cve_pal + 1
   mitabla.AddNew
     mitabla("cve_pal") = clavepal
     mitabla("palabra") = Nueva_Palabra
   mitabla.Update
   mitabla.Close

   'Registra la(s) definición(es) en la tabla TabDef
   Set mitabla = db.OpenTable("TabDef")
   If j = 0 And No_definicion(j) <> "" Then 'La palabra solo contiene un significado
```

```
    mitabla.MoveLast
    clavedef = mitabla!cve_def + 1
    mitabla.AddNew
     mitabla("cve_def") = clavedef
     mitabla("definicion") = No_definicion(cont)
     mitabla("cve_pal") = clavepal
    mitabla.Update
    'Call llenar_tabla(cont, clavepal)
  Else
    For X = 0 To j
      If No_definicion(X) <> "" Then
        mitabla.MoveLast
        clavedef = mitabla!cve_def + 1
        mitabla.AddNew
         mitabla("cve_def") = clavedef
         mitabla("definicion") = No_definicion(X)
         mitabla("cve_pal") = clavepal
        mitabla.Update
        'Call llenar_tabla(cont, clavepal)
        End If
    Next X
  End If
  mitabla.Close

  'Registra la clave de la palabra y los diferentes temas a los que pertenece
  Set mitabla = db.OpenTable("TemaPal")
  For Y = 0 To List1.ListCount - 1
    If List1.Selected(Y) Then
      NombreTema = List1.List(Y)
      ClaveTema = BuscarClaveTema(NombreTema)
      mitabla.AddNew
       mitabla("cve_tema") = ClaveTema
       mitabla("cve_pal") = clavepal
      mitabla.Update
    End If
  Next Y
  mitabla.Close
  db.Close
End Sub
```

Procedimientos para mostrar una serie de palabras referentes a un tema seleccionado por el usuario.

```
Sub MostrarSignificado ()
   ventana = 2                          'la ventana a cerrar es la de tema Form4
   If PalSeleccionada = 1 Then
      MDIForm1.MousePointer = 11
      Form4.WindowState = 1
      buscar = List1.List(List1.ListIndex)  'Lee la palabra a buscar
      Call BuscarPalabra(buscar)            'Busca el significado de la palabra
      Form1.Show
   Else
      Beep
      MsgBox "Seleccione una palabra de la lista", , "¡ Cuidado !"
      List1.SetFocus
   End If
End Sub


Sub temapal (SQLst As String)
   Dim SetTema As dynaset
   PalSeleccionada = 0
   List1.Clear
   Form4.MousePointer = 11
   Set db = OpenDatabase(CurDir$, False, False, conn$)
   Set SetTema = db.CreateDynaset(SQLst)
   If SetTema.EOF And SetTema.BOF Then
      MsgBox "No hay palabras asociadas al tema seleccionado", , ""
   Else
      SetTema.MoveLast
      max = SetTema.RecordCount
      SetTema.MoveFirst
      For i = 1 To max
         List1.AddItem SetTema!palabra
         SetTema.MoveNext
      Next i
   End If
   SetTema.Close
   db.Close
   Form4.MousePointer = 0
   List1.SetFocus
End Sub


Sub Administración_Click (Value As Integer)
   tema = "ADMN"     'Administración
   SQLst = " SELECT tabpal.cve_pal, palabra,cve_tema FROM TabPal,TemaPal WHERE
TemaPal!cve_pal=TabPal!cve_pal and cve_tema = '" & tema & "' order by palabra"
   Call temapal(SQLst)
End Sub


Sub Aceptar_Click ()
   Call MostrarSignificado
End Sub
```

Apéndice B
Ambiente gráfico

A continuación se presentan algunas de las pantallas usadas en el Diccionario de Terminos Administrativos, el orden de las figuras es de acuerdo a la secuencia en que un usuario podria necesitarlas.

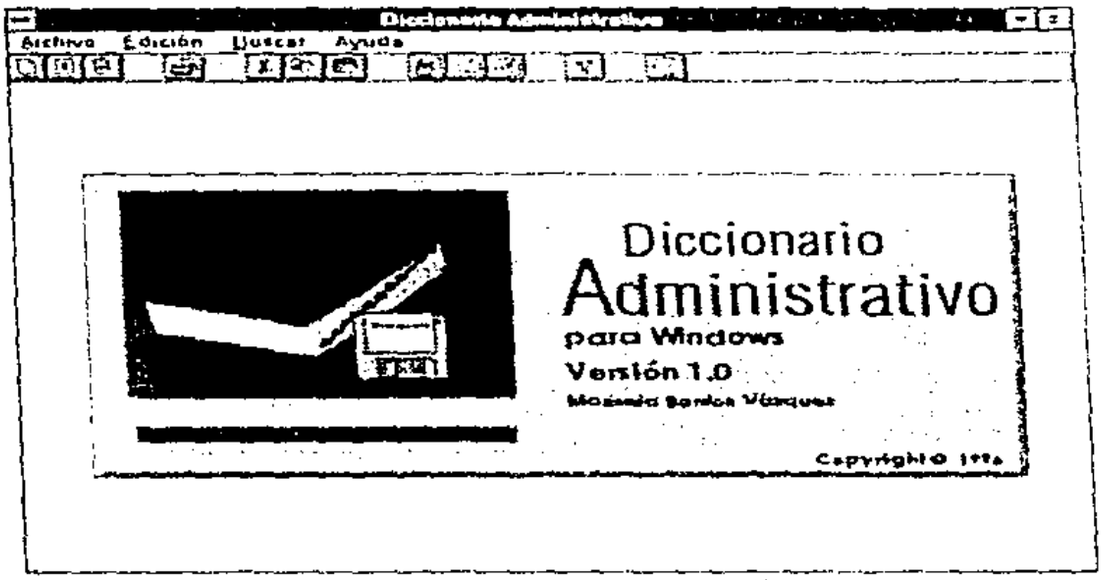


fig. 1 Inicio de la aplicación.

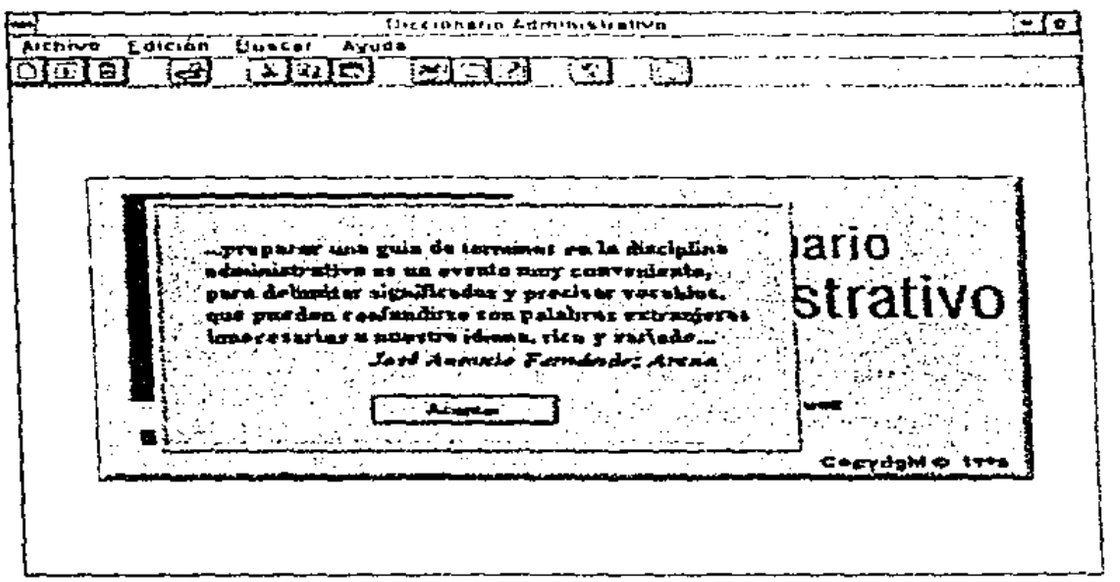


Fig. 2 Mensaje activado al hacer clic sobre el dibujo.

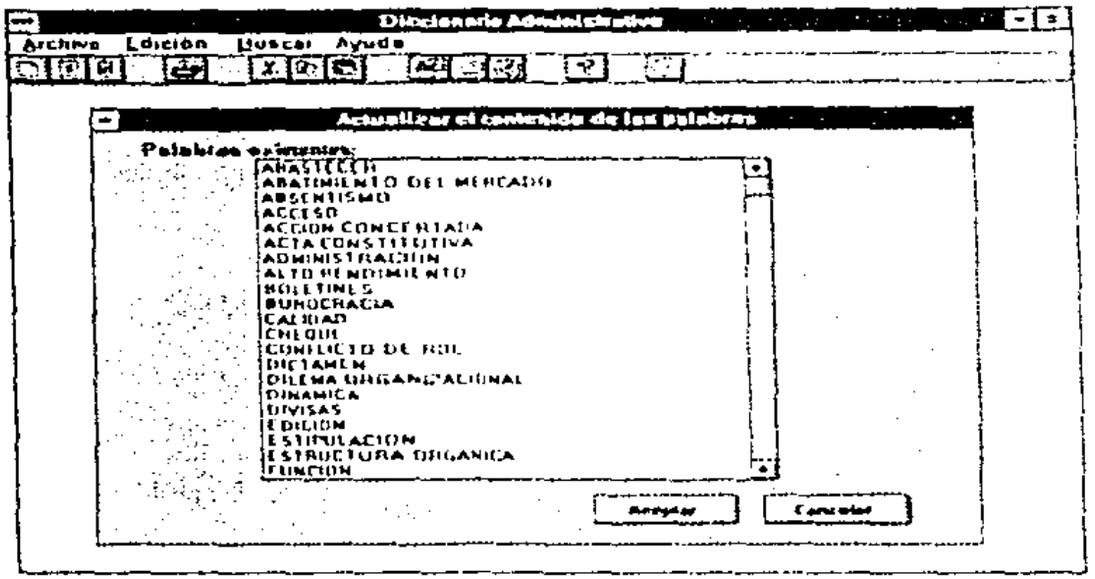


Fig 3. Listado de todas las palabras susceptibles a ser modificadas.

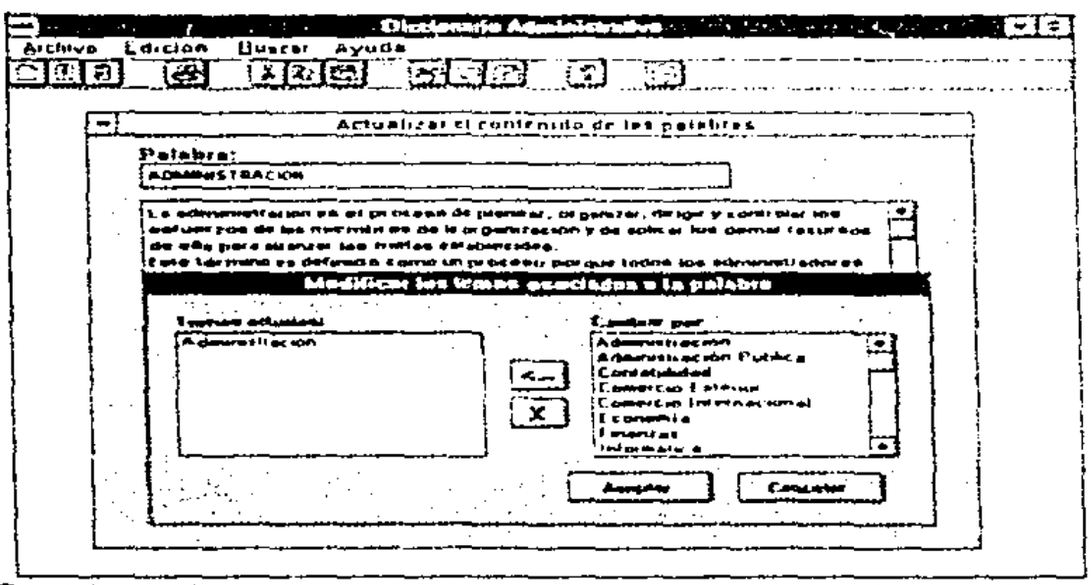


Fig. 4 Pantalla que permite modificar el contenido de la palabra.

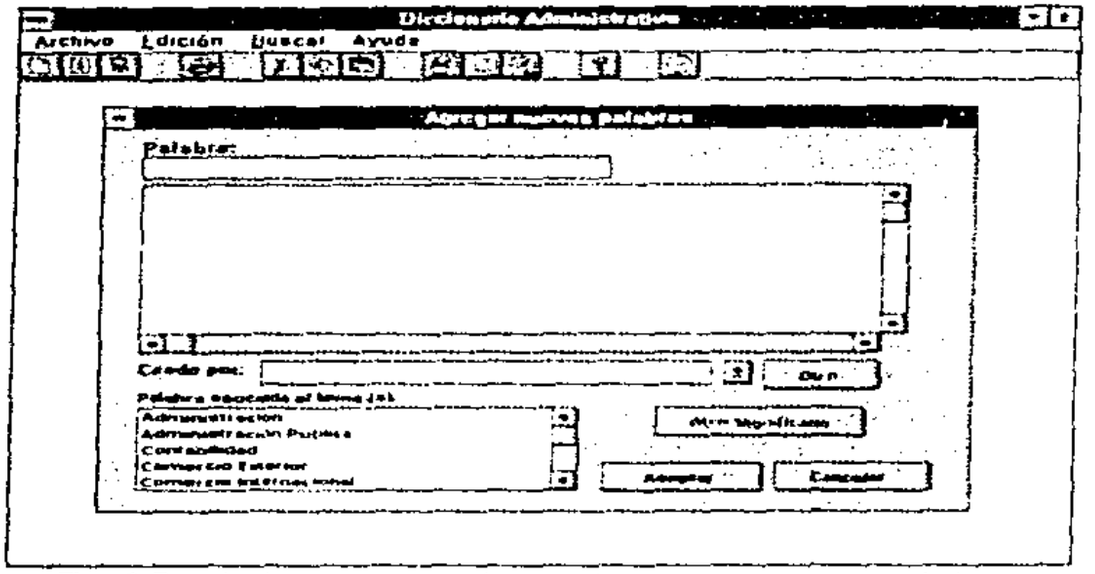


Fig. 5 Pantalla que permite capturar una nueva palabra.

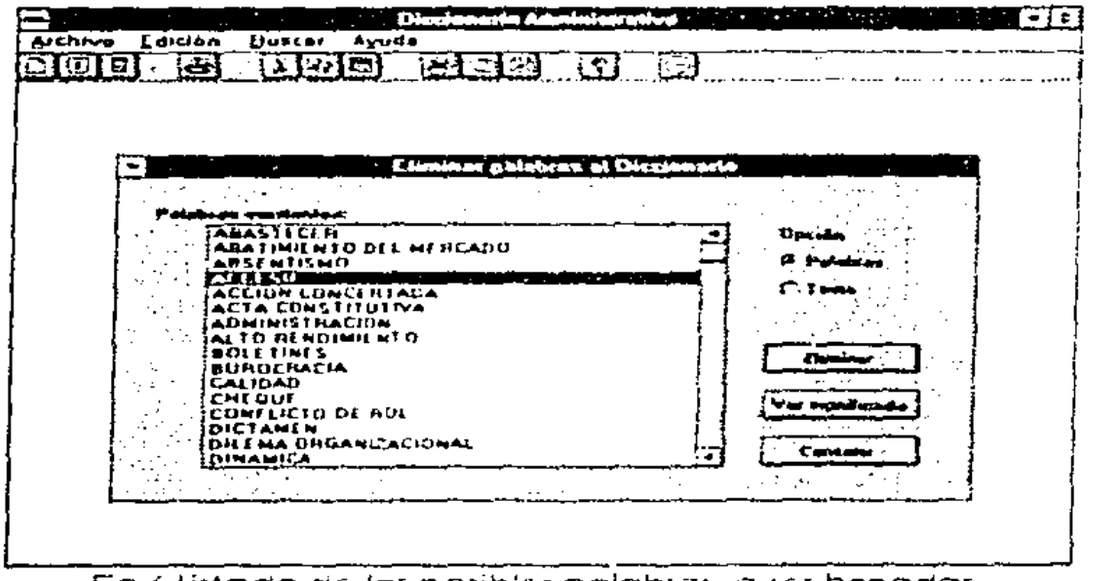


Fig.6 Listado de las posibles palabras a ser borradas.

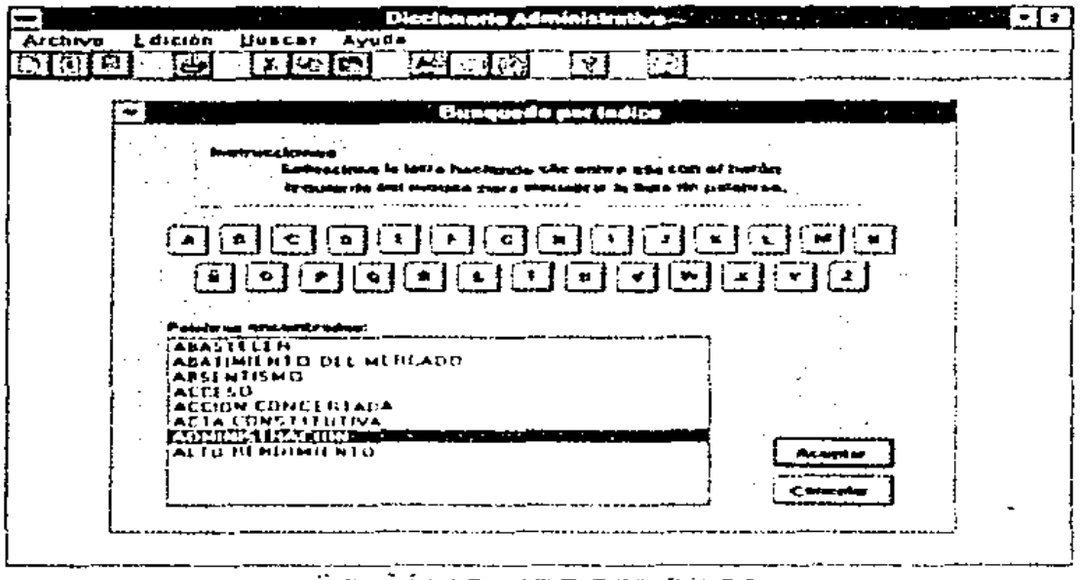


Fig. 7 Búsqueda por índice.

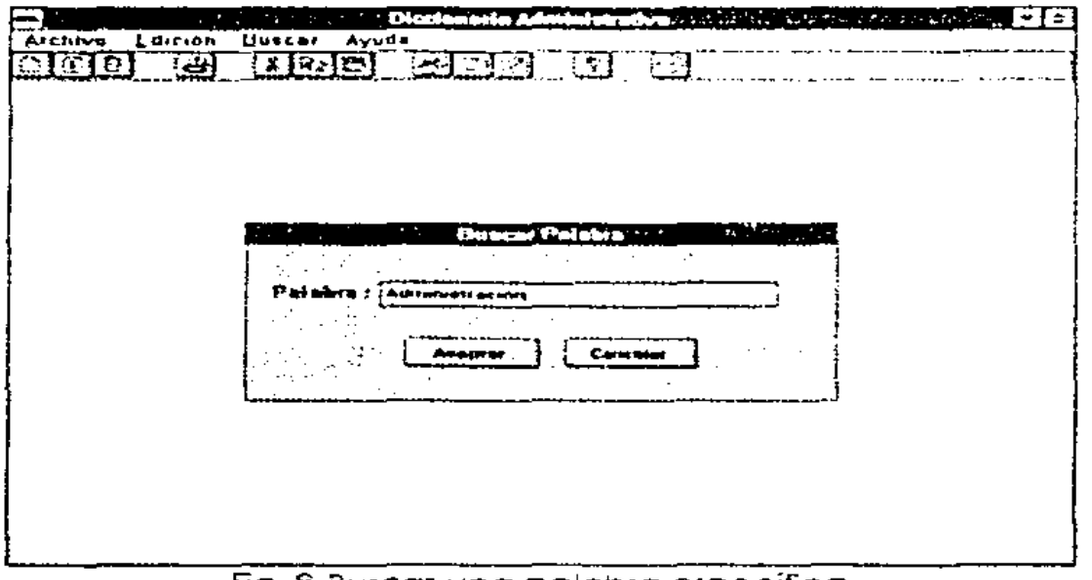


Fig. 8 Buscar una palabra específica.

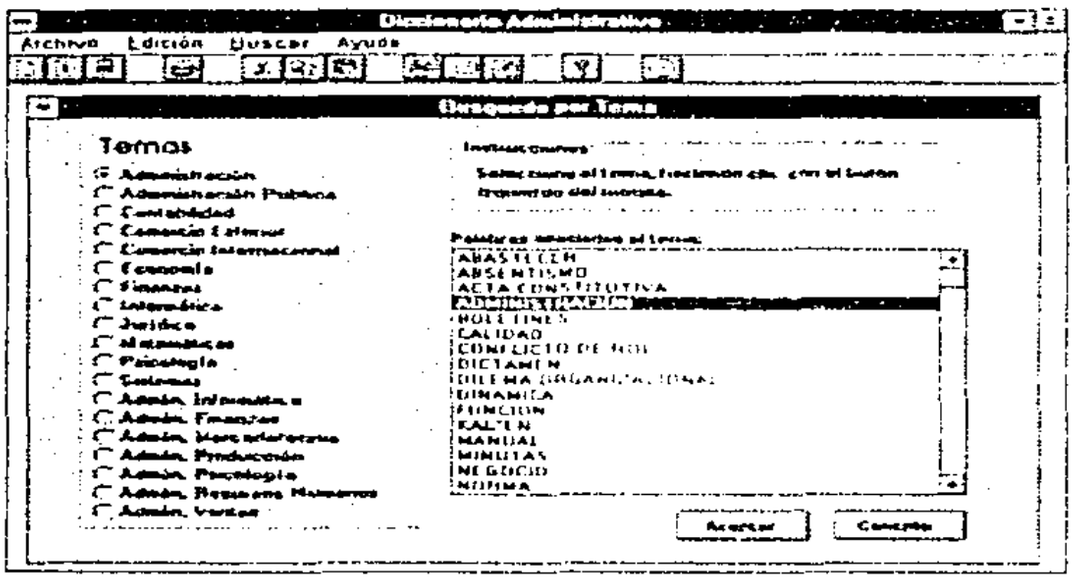


Fig. 9 Busqueda por temas.

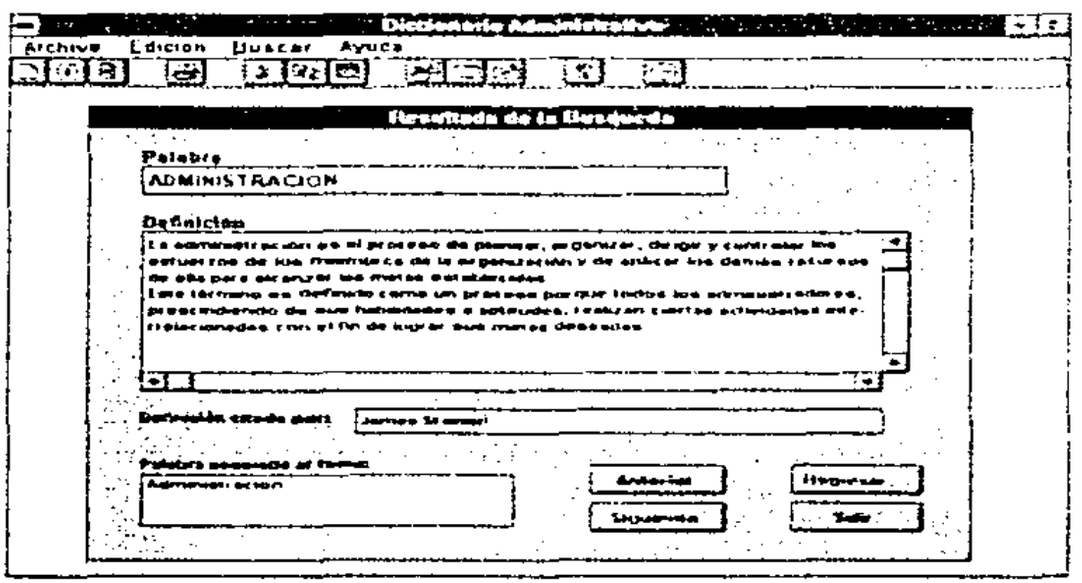


Fig. 10 Pantalla, resultado de cualquier búsqueda (índice, tema, palabra).

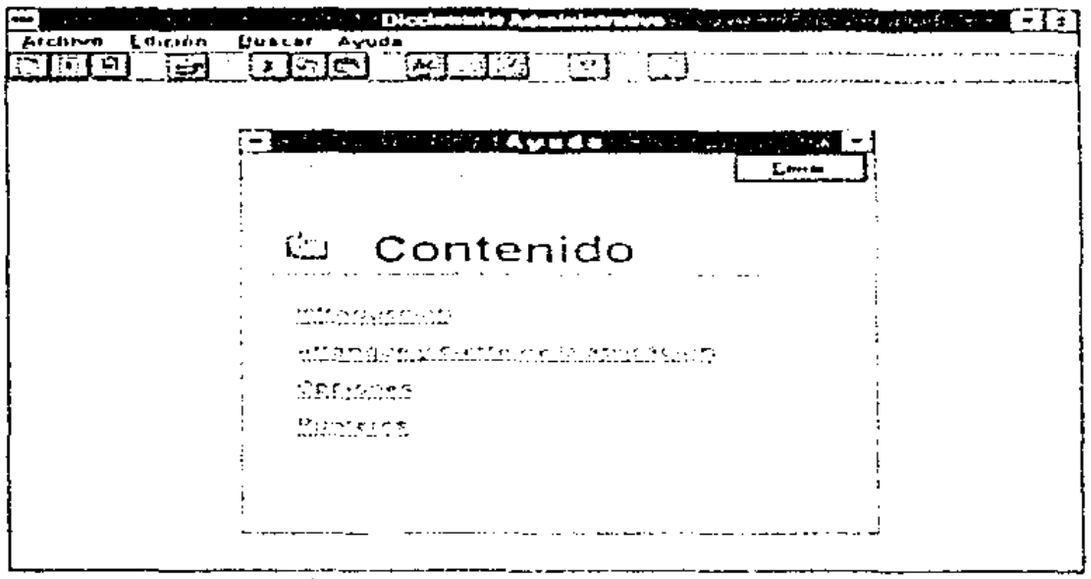


Fig. 11 Ayuda en línea del diccionario.

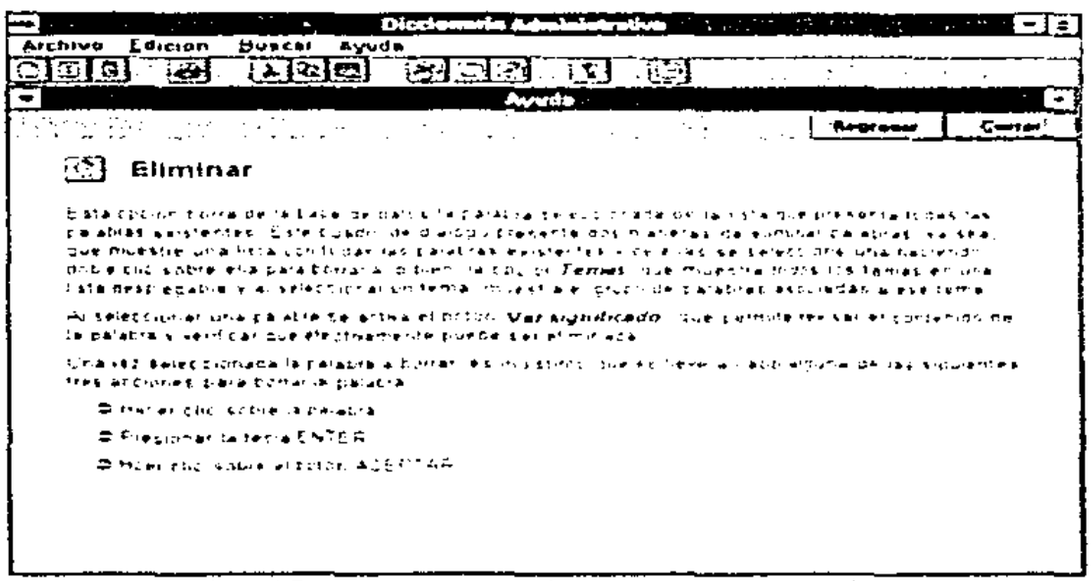


Fig. 12 Ayuda específica de una acción.

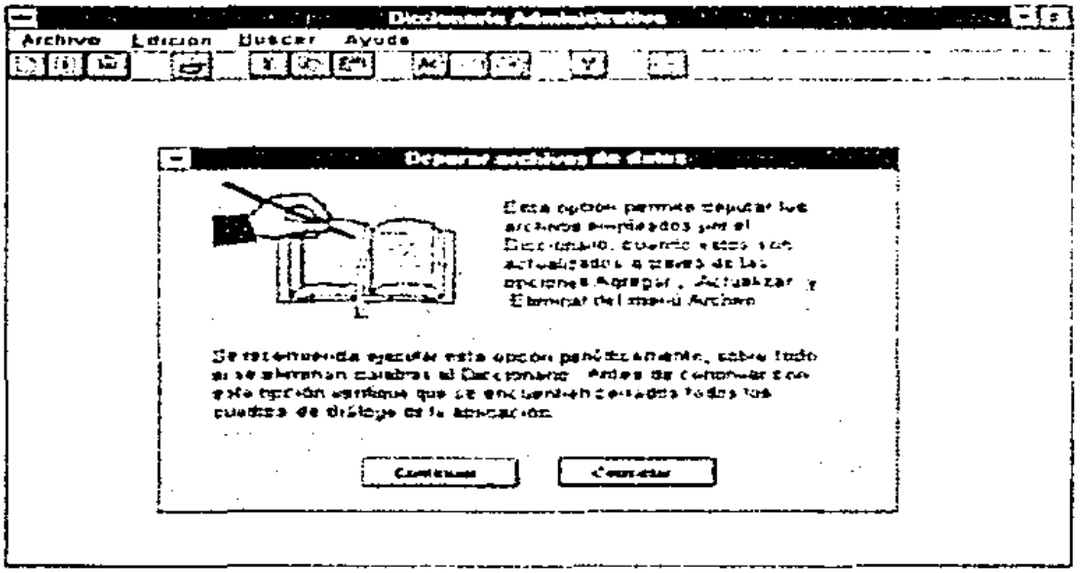


Fig. 13 Depuración de los registros borrados de la base de datos.