UNA EXTENSION DE HIPERMEDIA PARA LA ORGANIZACION DE DOMINIOS COMPLEJOS

EL MODELO MHEF Y SU APLICACION EN EL CAMPO DE LAS REDES DE COMPUTADORAS

C.C. María Alejandra Schiavoni

Director: Lic. Francisco Javier Díaz

A mis padres

AGRADECIMIENTOS

Quiero mencionar en forma especial a Claudia Queiruga, quien colaboró aportando su tiempo y su valiosa dedicación, fundamentales para llevar adelante esta tarea. Agradezco profundamente su inestimable apoyo y cariño, brindados durante todo el desarrollo de este trabajo.

Además, quiero agradecer a Gustavo Mariani y Gustavo Sagasti, quienes desinteresadamente aportaron su valioso asesoramiento.

No puedo dejar de agradecer al personal del CeSPI, en particular a: Cecilia Pertino, Pedro Brissón, Néstor Castro, Alejandra Osorio por brindarme su ayuda en todo momento.

Por último, agradezco enormemente a quien fue el director de este proyecto el Lic. Javier Díaz por su sincera confianza, su constante colaboración y su apoyo incondicional que me ayudaron a superar los momentos más difíciles y me permitieron llevar a cabo este proyecto en un clima basado en el afecto y la cordialidad. Gracias por haber confiado en mí, en una etapa difícil de mi carrera.

A todos, GRACIAS.

UNA EXTENSION DE HIPERMEDIA PARA LA ORGANIZACION DE DOMINIOS COMPLEJOS

El modelo MHEF y su aplicación en el campo de las redes de computadoras

INTRODUCCION

DISEÑO DE MHEF: MODELO HIPERMEDIAL CON ESTADISTICAS Y FILTROS

1. HIPERTEXTO/HIPERMEDIA

2. UNA EXTENSION DEL MODELO DE HIPERMEDIA PARA LA ORGANIZACION DE DOMINIOS COMPLEJOS

3. APLICACION DEL MODELO A UN DOMINIO CONCRETO: REDES DE COMPUTADORAS

IMPLEMENTACION DE IMHEF: UN PROTOTIPO FUNCIONAL BASADO EN MHEF

4. ANALISIS DE LAS HERRAMIENTAS UTILIZADAS

5. ARQUITECTURA DEL PROTOTIPO IMHEF

6. DISEÑO DE LA INTERFAZ DEL USUARIO

7. IMPLEMENTACION DE IMHEF

INDICE DE TABLAS

CONCLUSIONES

REFERENCIAS

INTRODUCCION

El presente trabajo arranca la línea de investigación que integra los temas Interfaces hombre-computadora y Redes de datos.

Cómo describir adecuadamente las redes, cómo organizar el dominio y posibilitar su comprensión por un amplio espectro de público fueron las motivaciones principales de este proyecto. Tengamos en cuenta que actualmente las posibilidades de usar redes no están restringidas a profesionales informáticos o al uso para actividades de investigación, sino que llega a un público mucho más amplio.

A fin de elegir un enfoque para difundir las redes se consultó con gente especializada en la atención de usuarios (ver agradecimientos). Tratando de responder el tipo habitual de consultas de los usuarios con las posibilidades (medios) de una herramienta clásica para desarrollar hipermedia, se llega a la etapa donde se analizaron plenamente las posibilidades y limitaciones de hipermedia, como herramienta para organizar información.

Se analizaron en forma cuidadosa las necesidades para cumplir satisfactoriamente la meta de construir un sistema que permitiese a un usuario cualquiera, conocer y aprender sobre redes de datos comprendiéndolas a nivel de sus capacidades e intereses. Con este objetivo surge la necesidad de brindar una estructura de base muy completa, con un gran volumen de información, pero con ayudas que asistan y orienten una navegación parcial.

En este punto se focalizó en los problemas y posibles soluciones para la organización y consulta de extensos volúmenes de información. Esto dio lugar a un modelo extendido cuya aplicabilidad fue presentada en algunos congresos de la especialidad, tanto nacionales como internacionales [Díaz93] [Díaz94a] [Díaz94b] [Schi94] [Díaz95]. Este modelo fue bautizado MHEF (**M**odelo **H**ipermedial con **E**stadísticas y **F**iltros), resaltando las dos características distintivas fundamentales de la propuesta, por un lado el mantenimiento y uso de estadísticas de visitas, y por el otro lado, filtros que posibilitan la obtención de espacios de búsqueda más reducidos que el lector puede manejar y explorar con facilidad.

Por último y a fin de testear la aplicabilidad de la propuesta se construyó un prototipo completamente operacional: IMHEF (Instanciación de **MHEF**), que permitió testear prácticamente los objetivos teóricos del modelo MHEF. El prototipo ilustra dos aspectos, el primero respecto de la factibilidad (complejidad e integración de herramientas) de la implementación de MHEF en un producto estandard y el segundo la usabilidad del sistema resultante donde se combinan aspectos de descripción del dominio (por ejemplo técnicas de interacción tales como animaciones, dependencias entre objetos, etc,) con la manipulación de las nuevas facilidades propuestas.

El proyecto demuestra que es posible mejorar los mecanismos básicos de hipermedia (navegación y visualización de nodos y enlaces) teniendo un éxito muy notable y posibilitando la aplicación en dominios cuya estructura y contenidos dificulta, o más aún impide una organización textual o de bases de datos tradicional.

A continuación resumo la organización en capítulos del informe a fin de orientar la lectura y resaltar los objetivos de cada capítulo.

Este informe se divide en dos partes: una teórica (que comprende los tres primeros capítulos) y otra de implementación.

En el capítulo *Uno* se introducen conceptos básicos de Hipermedia, ventajas y desventajas de usar esta herramienta y se establecen las diferencias entre hipermedia y multimedia. El objetivo de este capítulo es hacer conocer al lector la esencia de esta herramienta, sobre la cual se basa el modelo descripto en este informe.

El capítulo *Dos* trata en primer lugar las características y limitaciones de hipermedia en dominios complejos, para luego describir en detalle el diseño y funcionalidad del modelo MHEF. La finalidad de este capítulo es que el lector entienda cada una de las facilidades propuestas por el modelo como una solución a las limitaciones de hipermedia expuestas previamente.

En el capítulo *Tres* (último de esta sección), se describe en detalle el dominio de Redes de computadoras, elegido por su extensión, complejidad y por contar con un espectro de usuarios muy variado. Se muestran los principios básicos para la aplicación de las facilidades de MHEF a este dominio. El lector puede observar la funcionalidad del modelo, a través de la instanciación de cada una de sus herramientas en el campo de aplicación elegido.

En el capítulo *Cuatro* se hace un análisis exhaustivo de las herramientas utilizadas para la implementación del prototipo funcional. Se describe en detalle el software de base utilizado: Multimedia Toolbook 1.53, junto con su lenguaje de programación (OpenScript) y su posibilidad de extensión, usando librerías escritas en otros lenguajes (Pascal, C, etc.). Además, se explica cómo surgió la necesidad de usar un manejador de bases de datos del tipo dBase.

El capítulo *Cinco* describe detalladamente la arquitectura básica del prototipo IMHEF, las librerías implementadas y las bases de datos utilizadas. Con el contenido de este capítulo el lector tiene una visión global de la implementación de IMHEF.

El capítulo *Seis* trata sobre las características de la interfaz del usuario y los objetivos tenidos en cuenta durante el proceso de diseño de la misma. Se explican cada una de las técnicas utilizadas, mencionando sus ventajas y fundamentando el uso de las mismas. Además, se mencionan distintos aspectos de usabilidad de la interfaz diseñada.

Por último, en el capítulo *Siete* se explica en forma detallada todo el proceso llevado a cabo para la construcción del prototipo: desde la recopilación y organización de la información, hasta la implementación final. Se describen cómo fueron implementadas cada una de las facilidades del modelo MHEF, y se muestra una especificación de todas las librerías y bases de datos construídas.

Si el lector desea tener mayor información ya sea sobre la esencia y funcionalidad del modelo MHEF, como sobre las características detalladas de la implementación del prototipo IMHEF, tenga a bien contactarse con el LINTI (Laboratorio de Investigación en Nuevas Tecnologías Informáticas).

DISEÑO DE MHEF: MODELO HIPERMEDIAL CON ESTADISTICAS Y FILTROS

1. HIPERTEXTO / HIPERMEDIA

1.1. Conceptos generales

El texto "chato" tradicional expresado a través del papel obliga a escribir párrafos en forma secuencial determinando una única secuencia de lectura. A medida que las computadoras fueron evolucionando, surgieron nuevas posibilidades que permitieron una organización más compleja de la información. Surgieron nuevos mecanismos que permitieron una manipulación directa de referencias o vínculos entre distintos bloques de texto; fueron provistas nuevas interfaces que permitieron a los lectores interactuar con dichos bloques de información y establecer nuevas relaciones entre ellos. Ted Nelson (uno de los pioneros en el tema) definió hipertexto como una combinación entre un texto en lenguaje natural y la posibilidad de almacenarlo y recuperarlo en forma no secuencial a través de una computadora [Nels67].

Hipertexto, básicamente, consiste de un conjunto de piezas de texto interrelacionadas. Cada pieza de información se denominada **nodo**, **notecard** [Trig86] o **card** y contiene una colección de datos relacionados con un tópico en particular. Cada uno de estos nodos puede tener punteros a otras unidades de información, y estos punteros se denominan **links**. El número de links de cada nodo no es fijo, depende del contenido del nodo. Existen nodos que están relacionados con muchos otros nodos y, por consiguiente tienen muchos links, sin embargo existen nodos que sirven solamente como destino de links. En un principio, los nodos contenían sólo información textual. Actualmente, pueden contener información proveniente de diferentes medios: gráficos, imágenes animadas, audio, video, etc., por lo cual se utiliza el término *hipermedia*. En este informe usaré indistintamente los términos hipertexto e hipermedia.

El concepto de hipertexto es simple: **ventanas** en pantalla que están asociadas a objetos (llamados nodos) pertenecientes a una base de datos y **vínculos** (llamados links) entre estos objetos. Los vínculos existentes entre estos objetos son: 1) vínculos gráficos, que representan tokens con nombre y 2) vínculos en la base de datos, que son punteros entre los nodos [Conk87] (Fig. 1).

Desde un punto de vista simple e informal, un *hipertexto* es un texto desplegado de una manera no lineal. Esta organización no secuencial de la información le permite al lector acceder a ella fácilmente "saltando" de tópico en tópico. El usuario no necesita leer el texto en forma rígida y secuencial [Niel90] [Seye91].

Desde un punto de vista formal, hipertexto es un *método de base de datos* que provee una nueva forma de acceso directo a los datos, diferente a los esquemas tradicionales de consulta. A su vez, hipertexto es un *esquema de representación*, una clase de red semántica que combina texto informal con operaciones y procesos mecanizados más formales. Finalmente, hipertexto es

una *modalidad de interface* que incluye "botones de control" definidos por el autor (íconos que representan el origen de un vínculo). La funcionalidad de hipertexto resulta una fusión de estas tres metáforas.

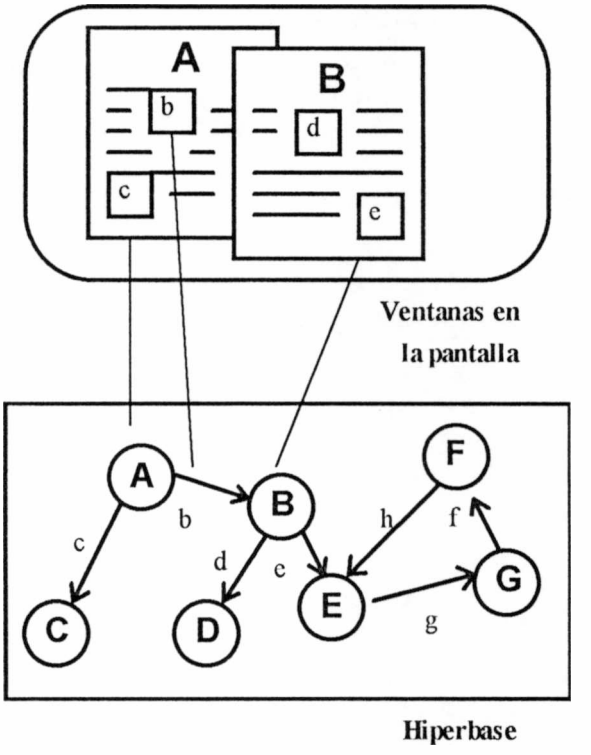


Fig. 1. Correspondencia entre las ventanas y vínculos de la pantalla y los nodos y vínculos de la base de datos.

1.2. Constructores básicos de un hipertexto

1.2.1 Nodos

Los nodos consisten en fragmentos de texto, gráficos, animaciones, audio o video. El tamaño de un nodo (granularidad) varía desde un único gráfico o unas pocas líneas de texto hasta un extenso documento [Jona90]. Los nodos representan la unidad básica de almacenamiento de información. Sin embargo, no existe un acuerdo total con respecto a lo que realmente constituye un nodo [Niel90]. El autor de un hipertexto utiliza los nodos para expresar ideas.

En el proyecto que se describe en este informe, cada nodo constituye una unidad conceptual. Los nodos son utilizados para representar, a través de texto y gráficos, cada uno de los tópicos del dominio en cuestión.

1.2.2. Links

La característica más distintiva de hipertexto, es el mecanismo de soporte de "navegación" a través de los links o vínculos. Un link es información propia de cada nodo, que lo conecta a otro nodo de la red.

Los links que parten de un nodo se visualizan a través de "botones" (un botón es la representación visual de un link en un nodo).

Los links pueden ser referenciales u organizacionales [Conk87]. Los links referenciales apuntan a información contenida en otro nodo y generalmente le permiten al usuario retornar al nodo origen a través del mismo link. Los links organizacionales implementan información jerárquica, conectan cada nodo con sus hijos formando un árbol estricto dentro de la red de hipertexto. Además, los links pueden ser de *valor*, de *texto* o de *léxico* [Coll87]. Los links de valor parten desde un nodo y apuntan a otro nodo. Los links de texto parten de un texto y llegan a un nodo. Por último, los links de léxico parten de una región de un texto y llegan a un nodo. Dependiendo de la naturaleza o tipo del link, se desencadenan distintas acciones.

En el modelo desarrollado, se implementan links tanto referenciales como organizacionales para vincular los nodos. Además, en cada uno de los nodos, los links tienen como origen un texto, porción de texto o un gráfico y como destino, un nodo.

1.3. Ventajas del uso de hipertexto

Hacer referencias entre textos no es un método nuevo. La importancia de hipertexto radica en el hecho de que provee mecanismos para soportar estas referencias [Nels67]. En la literatura tradicional, como en hipertexto, se establecen relaciones entre ideas y los conceptos se organizan jerárquicamente. El medio usado en la literatura tradicional, que es el papel, restringe el flujo de la lectura a una lectura totalmente secuencial. Sin embargo, es fundamental poder acceder a referencias externas aún en el medio impreso. Toda persona que haya realizado investigación sabe que gran parte del esfuerzo radica en la búsqueda de trabajos de referencia, obtención de referencias cruzadas, búsqueda de términos en el diccionario y glosarios, construcción de anotaciones relacionadas, etc. Aún en una simple lectura, una persona está continuamente buscando referencias en otros capítulos o secciones (a través del índice de contenidos o de referencias explícitas que aparecen en el texto), en referencias bibliográficas, en notas al pie de página, en tablas y figuras, etc. Frecuentemente, el autor del texto le indica al lector que saltee una sección si no está interesado en obtener información con más grado de detalle.

Según Conklin [Conk87], los métodos tradicionales presentan algunos problemas, solucionados con la técnica de hipertexto:

- No se puede retroceder a través de las referencias. Un lector no puede encontrar fácilmente dónde se hizo referencia a un artículo o libro determinado. De la misma manera, a un autor le resulta difícil encontrar el lugar donde fue referenciado su artículo.

• A medida que el lector se "anida" o profundiza buscando referencias, le resulta difícil mantener en su memoria los documentos por los que pasó.

• El lector debe ubicar sus anotaciones en el margen del texto que está leyendo, o en un documento separado.

• Finalmente, acceder a referencias entre documentos requiere un gran esfuerzo y una pérdida de tiempo (aún si el lector está trabajando en el marco de una buena biblioteca y dispone de un gran número de libros, artículos y publicaciones).

Nuevas posibilidades para autoría y diseño

Hipertexto ofrece nuevas formas de escritura y diseño de documentos. El proceso de autoría es visto tradicionalmente como una actividad a nivel de procesamiento de texto. Sin embargo, la autoría de documentos está muy relacionada con el proceso de estructuración de ideas, orden de presentación, y exploración conceptual. La unidad en el nivel de autoría es la idea o concepto. Hipertexto soporta esta metodología de trabajo, ya que una idea específica puede ser expresada en un nodo. A medida que el autor construye nuevas ideas, puede escribirlas en sus propios nodos, y luego asociarlos con ideas o nodos ya existentes, o dejarlos aislados si es demasiado prematuro para establecer alguna asociación.

Nuevas posibilidades para la lectura y recuperación de información

Hipertexto ofrece, además, nuevas posibilidades para el acceso a grandes volúmenes de información. Un documento lineal sólo puede ser accedido en forma secuencial. La ventaja esencial de hipertexto es la posibilidad que brinda de poder recuperar la información desde diferentes puntos de vista. La utilidad de hipertexto radica fundamentalmente en la posibilidad de establecer enlaces entre ideas relacionadas, permitiendo a los lectores un acceso más libre a la información [Niel90b].

Otra ventaja importante es que en un ambiente de hipertexto resulta natural suspender temporariamente la lectura de un tema en particular, para buscar un ejemplo, un tema relacionado o profundizar más en detalle sobre algún aspecto del tema en cuestión.

Ventajas operacionales de hipertexto

Conklin en su artículo [Conk87] sintetiza las ventajas operacionales de hipertexto de la siguiente manera:

• Facilidad de acceso a referencias externas: el mecanismo de soporte de links permite acceder fácilmente a las referencias hechas en cada documento, como así también el retroceso a partir de ellas.

• Facilidad para la creación de nuevas referencias: los lectores de hipertexto pueden hacer crecer la red de nodos o simplemente agregar anotaciones o comentarios.

• Estructuración de la información: la técnica de hipertexto permite organizar información tanto jerárquica como no-jerárquica.

• Creación de documentos personalizables: la posibilidad de acceder a un documento a través de caminos distintos permite que el mismo documento esté destinado a diferentes tipos de usuarios.

• Modularización de la información: en hipertexto un mismo segmento de texto puede ser accedido desde diferentes puntos, lo cual evita que las ideas o conceptos aparezcan duplicados.

• Consistencia de la información: las referencias existentes en una porción de texto, se conservan aún si el texto es movido dentro del documento o llevado a otro documento.

• Colaboración entre distintos autores: hipertexto permite que varios autores trabajen en forma cooperativa sobre el mismo documento.

• Acceso rápido a los conceptos: hipertexto permite acceder en forma rápida a los conceptos o ideas que son de interés para el lector.

1.4. Desventajas en el uso de hipertexto

Existen dos problemas clásicos asociados con la aplicación y uso de hipertexto: la desorientación del lector durante la navegación del hiperdocumento y la sobrecarga de información [Conk87] [Niel90][Simp90].

Desorientación durante la navegación

Cuando la información es organizada de una manera no secuencial o lineal, aparece el problema conocido como "desorientación o pérdida en el hiperespacio". A medida que el lector va recorriendo la red de nodos, comienza a formularse algunas preguntas tales como: Dónde estoy? o Cómo llegar a un determinado punto de la red (que sabemos o pensamos que existe)? [Simp90]. El problema de la desorientación también existe en los documentos organizados en forma secuencial, pero en estos casos el lector tiene sólo dos opciones: buscar el texto de interés en las páginas anteriores y en las siguientes. Hipertexto, en cambio, ofrece un mayor grado de libertad y un espacio con más dimensiones en donde el lector puede moverse; por consiguiente está más proclive a perderse dentro del hiperespacio.

Básicamente, existen dos técnicas que permiten visualizar y manipular el hiperespacio intentando solucionar el problema de la desorientación: *browsers gráficos* y *mecanismos de consulta y búsqueda en la base de datos*. Los

browsers gráficos están vinculados al sistema visual humano. La ubicación de los nodos y links en un espacio de dos o tres dimensiones aportan a los browsers características útiles, permitiendo una diferenciación visual entre los distintos tipos de nodos y links existentes en el hiperespacio. El browser ayuda al lector a orientarse durante la navegación, pues es una representación gráfica de la red de nodos. Sin embargo, para redes de hipertexto grandes y complejas, resulta muy difícil mostrar en forma clara el hiperdocumento a través de un browser.

Otras de las formas de solucionar el problema de la desorientación es usando técnicas standard de búsqueda y consulta en bases de datos para localizar el/los nodos que resultan de interés para el usuario. Generalmente se usan búsquedas por palabra clave o por strings dentro del contenido de los nodos. Sin embargo, en la mayoría de los casos los mecanismos standard de consulta no resultan adecuados para esta forma de organización de la información.

Sobrecarga de información

Desde el punto de vista del autor, resulta difícil acostumbrarse al esfuerzo mental adicional que representa el proceso de crear links, definirle nombres y recordar todas las referencias realizadas en el documento. Encontrar una palabra o frase significativa que sintetice el nodo o concepto referenciado no es una tarea fácil. El autor debe definir una frase que sea a la vez descriptiva y sugerente como para que el lector pueda decidir si le interesa acceder a esa referencia. Además, al autor se le presenta la posibilidad de darle al link un nombre que sugiera el contenido del nodo referenciado o la clase de relación entre el nodo origen y el destino. Una vez establecida una referencia a un nodo nuevo, el autor también debe tener en cuenta si no existen otros puntos (nodos) en la red, a partir de donde conviene establecer una referencia a dicho nodo.

El problema de la sobrecarga de información aparece también durante el proceso de recorrido de un hipertexto. Se le presentan al lector un gran número de alternativas, a partir de las cuales tiene que decidir con qué link continuará navegando por la red. El lector, en cada momento, se enfrenta con muchas opciones de recorrido, lo que provoca una sobrecarga mental a nivel de toma de decisiones. Esta sobrecarga no se produce cuando es el autor quien toma a priori estas decisiones. En el momento de encontrar un link, el lector se pregunta: seguir el camino definido por el link, no me distraerá del tema que estoy investigando?, el nombre del link dice lo suficiente del nodo al que referencia?.

Los problemas descriptos previamente no son nuevos en hipertexto, ni tampoco propios de él. Los escritores, científicos, artistas, diseñadores se enfrentan con el problema de que la mente crea ideas, más rápido de lo que ellos puedan escribirlas o expresarlas verbalmente. Hipertexto, simplemente ofrece una herramienta o "lápiz" sofisticado que permite combinar la riqueza, variedad y la capacidad de interrelación del pensamiento creativo humano.

1.5. Herramientas de navegación provistas por algunos sistemas de hipertexto

Los sistemas de hipertexto proveen, en general, un conjunto de herramientas que asisten al lector durante la navegación.

Browsers

Un browser muestra total o parcialmente el hiperdocumento como un grafo, permitiendo al lector tener una visión contextual y espacial de su ubicación dentro de la red [Conk87]. Esta visión le permite al lector tener una idea de qué nodo está viendo, cuáles son los nodos vecinos y cómo están relacionados unos con otros. Para hipertextos reducidos (menos de un centenar de nodos), un browser gráfico resulta muy útil. Sin embargo, cuando el dominio de aplicación es muy extenso, se obtiene un hiperdocumento con gran cantidad de nodos y vínculos, que aparece representado en un grafo muy complejo. En estos casos, puede resultar confuso mostrar en pantalla un grafo con tales características [Garz91].

Actualmente, existen muy pocos sistemas de hipertexto que generan automáticamente un browser del hiperdocumento (Intermedia [Yank88] y NoteCards [Hala87a] son dos de ellos).

Herramientas de consulta

La forma usual de buscar información en una red de hipertexto, ofrecida por algunos sistemas existentes, es consultar un "nodo de control" o un "nodo índice", éste último ofrece una visión general del sistema total [Seye91]. Sin embargo, en algunos casos resulta más rápido si se realiza una búsqueda por una cierta frase o palabra clave en todos los nodos. Al construir un hipertexto, algunos autores incorporan la búsqueda de string tradicional dentro del diseño para satisfacer las demandas de los lectores. Por otro lado, es posible combinar el mecanismo de búsqueda de texto tradicional con hipertexto [Niel90].

Historia de recorrido (Trails)

Un "trail" es un registro de los nodos accedidos por el usuario durante la navegación del hiperdocumento. Lo ideal es que esta lista de nodos contenga también las anotaciones creadas por el lector. Podría resultar útil que un usuario tenga acceso a trails o recorridos de otros usuarios [Jona90].

El registro de la historia de recorrido puede ser útil para refinar la red de hipertexto. El hecho de que un autor pueda analizar los recorridos realizados por los usuarios durante un proceso de búsqueda de información o de aprendizaje de un material educativo, puede resultar útil para redefinir algunos nodos y las relaciones entre ellos.

Tours

Un tour o recorrido guiado es un soporte computacional provisto por el autor para ofrecer pistas o ayudas a los lectores. Los tours constituyen una facilidad interactiva basada en un grafo, que sintetiza la estructura presentacional de la red de hipertexto [Mars89].

Conceptualmente, un tour es un grafo construído por el autor, donde cada nodo representa una parada y las aristas, vínculos entre estas paradas [Trig88].

Algunos sistemas de hipertexto proveen varias maneras de construir tours. Una forma de crear un tour es abrir un hiperdocumento y acceder a todos los nodos que se quieren incluir en el tour. El sistema irá registrando cada uno de estos nodos y los convertirá en paradas del recorrido. Por otro lado, durante la construcción de un hipertexto, algunos autores incorporan herramientas para la generación de tours dinámicos, basados por ejemplo en información estadística o en respuesta a una consulta del lector.

1.6. Multimedia

1.6.1. Conceptos Generales

El término *multimedia* se refiere a aquellas aplicaciones que interactúan con el usuario, usando simultáneamente diferentes medios como audio, imágenes estáticas, animaciones, gráficos y texto obteniendo de esta forma una comunicación más efectiva. El punto importante en las aplicaciones multimedia radica en el sincronismo necesario entre los distintos medios para representar una idea.

El concepto de multimedia tiene mayor impacto en aplicaciones desarrolladas para usuarios que no tienen ningún conocimiento sobre procesamiento de datos, por ejemplo puntos automatizados de venta, aplicaciones típicas de escritorio, teleconferencias y correo electrónico, aplicaciones para capacitación y entrenamiento, documentación o proyectos basados en hipermedia, aplicaciones basadas en trabajo cooperativo, sistemas para edición de videos, etc.

El estudio de sistemas multimedia se torna interesante no sólo por el potencial que le otorga a las nuevas aplicaciones, sino también porque cuestiona las soluciones tradicionales que dependen fuertemente del uso de datos alfanuméricos. En las aplicaciones multimedia existen tres puntos fundamentales [Gome92]:

- la adquisición, representación y presentación de objetos multimedia (objetos que representan audio, imágenes, etc.);

- el almacenamiento y recuperación de objetos multimedia;

- la transmisión de objetos multimedia, especialmente audio e imágenes en movimiento, que exigen transmisión en tiempo real.

1.6.2. Multimedia vs. Hipermedia

Aunque muchas aplicaciones de hipertexto tienen efectos multimedia, no necesariamente un sistema multimedia tiene la información organizada en una red de nodos y links, permitiendo la navegación a través de ella. La combinación de texto y gráficos no es suficiente para construir una hipermedia [Niel90]. Muchos sistemas de multimedia se basan en la proyección de films a un usuario pasivo, que no tiene la posibilidad de navegar en un espacio de información. Sólo cuando los usuarios pueden activar links dinámicos entre distintas unidades de información de forma interactiva, estamos en presencia de un sistema de

hipermedia. La diferencia entre multimedia e hipermedia es similar a la que existe entre observar un film de un viaje y ser un turista.

Una clase de sistema multimedia que frecuentemente se confunde con hipermedia es el video interactivo. Es muy común la utilización de efectos de video interactivo en la interface de un sistema de hipermedia, pero muchos de los sistemas de video interactivo no son lo suficientemente "interactivos" para clasificarlos de hipermedia. Muchos de estos sistemas de video le otorgan al usuario un rol de observador pasivo, quien sólo puede seleccionar video clips de un menú. La razón fundamental de por qué este diseño de sistemas no es hipermedia, es que el usuario no tiene forma de interactuar con el video una vez que éste comenzó a proyectarse. En otras palabras, la interacción con el sistema no le permite al usuario sentir que puede navegar, explorando el espacio de información.

El hecho de permitirle al usuario más opciones de navegación implica el uso de una interface más sofisticada, que simplemente mostrarle un menú para que seleccione una opción.

2. UNA EXTENSION DEL MODELO DE HIPERMEDIA PARA LA ORGANIZACION DE DOMINIOS COMPLEJOS

2.1. Características esenciales de hipermedia en dominios complejos

Como se ha descripto en las secciones anteriores, hipermedia es una herramienta que facilita tanto la organización como la recuperación de la información, debido a la flexibilidad y libertad que brinda en cuanto a la forma de expresar los conceptos y las relaciones entre ellos. La característica esencial de hipermedia es la posibilidad de navegar a través de un hiperdocumento siguiendo los vínculos o relaciones entre los conceptos. La navegación aparece como el único mecanismo de recuperación de información, cuando dicha información está organizada de una manera no secuencial [Hala87b][Ichi93a].

Desde el punto de vista de la organización de los datos, la gran ventaja de hipermedia es la posibilidad de generar información altamente interconectada y con gran número de referencias cruzadas [Utti89]. Esto permite fácilmente mantener y manejar información no estructurada, en una base de datos. Sin embargo, cuando el usuario accede a un sistema de hipermedia, examina los nodos de una manera secuencial, debido a su forma habitual de acceso a un documento [Ichi93b]. En otras palabras, los usuarios de los sistemas de hipermedia se detienen a examinar el contenido de cada nodo en detalle, para luego determinar con qué nodo continuar el recorrido. En cada momento de la sesión de lectura, los usuarios deben involucrarse en un proceso de toma de decisiones que no es habitual para ellos [Bego90]. En el caso de redes de hipermedia complejas, este mecanismo de recuperación de información puede resultar completamente improductivo para el usuario.

Por otro lado, los dominios amplios y complejos generan en términos de hipermedia un grafo con gran cantidad de nodos y múltiples relaciones entre ellos. Cuando un lector se enfrenta con un universo de tales características, le resulta difícil encontrar la información de su interés, si sólo dispone del mecanismo de navegación.

Además, este paradigma de recorrido propuesto por hipermedia, puede no ayudar a los lectores que no están familiarizados con el tema presentado, ya que la lectura de los tópicos podría hacerse en un orden no adecuado. Es decir, los grandes hipermedios pueden incorporar nuevas fuentes de confusión y distracción a los intereses de un lector, además de requerir un esfuerzo mental mayor que en el uso de documentos lineales [Egan89]. A medida que el tamaño del hipertexto crece, el problema de "perderse en el hiperespacio" limita la eficiencia y usabilidad de los sistemas de hipermedia.

En los sistemas hipermediales aplicados sobre dominios vastos, donde la navegación es el único mecanismo de recuperación de información, se "compromete" la utilidad y productividad de los mismos.

En el punto siguiente se explicarán cuáles son las limitaciones que presenta hipermedia cuando el dominio de aplicación es muy extenso y complejo. Además, se proponen a modo introductorio algunas posibles soluciones a los problemas planteados.

2.2. Limitaciones de hipermedia en dominios complejos

En el uso de sistemas de hipermedia, aparecen dos problemas típicos (ya mencionados anteriormente en este informe) que son: la "desorientación del usuario" durante la navegación y la "sobrecarga de información", éste último relacionado con la clase (calidad) de la información que el lector va recuperando cuando recorre el hiperespacio. La desorientación se produce cuando el lector se encuentra totalmente perdido en el hiperespacio y no es capaz de volver a un nodo previamente visitado o de seguir un camino o secuencia de nodos que le resulte interesante. Cuando un lector se enfrenta con documentos extensos organizados bajo una red de nodos y vínculos, es usual que llegue un punto en el cual se encuentra perdido; es decir no puede responder las preguntas: dónde estoy?, qué nodos visité? , cómo puedo llegar a un determinado lugar? [Simp90] [Paru89]. El lector puede no ser capaz de seguir un camino de nodos que contengan información de su interés, ni de retornar al nodo previamente visitado [Guin92]. Los lectores pueden perder su ubicación contextual, ya sea por una interrupción externa, como por la tarea de acceder momentáneamente a vínculos que le resulten interesantes, y que pueden llegar a ser terminales [Bern88]. Para tratar de solucionar este problema, muchos sistemas de hipermedia proveen mapas (esquemas gráficos de la estructura de la red) que resultan útiles cuando la cantidad de nodos que componen la red no excede el centenar. Sin embargo, cuando el dominio es extenso (abarca centenares o miles de nodos) se deben proveer estrategias para obtener vistas con distintos niveles de abstracción (vistas de grafos parciales).

En relación al segundo punto, que trata la sobrecarga de infomación (cognitive overhead) experimentada por los lectores, aparece el problema de poder recuperar fácilmente información relevante en una hipermedia. Esto está relacionado con el hecho de que el lector pueda recuperar la información que considera importante de acuerdo a sus necesidades, sin detenerse en datos o conceptos que no son de su interés. Si un usuario está navegando por el hiperespacio con el único propósito de encontrar información interesante o potencialmente útil, entonces el mecanismo de navegación definido por hipermedia resulta adecuado. Sin embargo, si un usuario tiene una finalidad específica e intenta encontrar datos concretos y precisos dentro de la red de nodos, le va a resultar difícil o casi imposible hacerlo si no cuenta con pistas o ayudas que lo orienten en la búsqueda [Guin92][Wate91].

Una propuesta para solucionar este problema es potenciar la navegación incorporando mecanismos de acceso basados en consultas. La funcionalidad de los métodos tradicionales de recuperación de información por selección de documentos o porciones de texto, en respuesta a una consulta del usuario, ha sido aprovechada desde hace mucho tiempo [Salt89]. La técnica de acceso y manipulación de datos propuesta por hipermedia, basada en la navegación a través de un universo de nodos y vínculos, es comparativamente reciente. La primera es útil para satisfacer las necesidades del usuario, cuando estos saben exactamente qué están buscando; mientras que la última es más efectiva cuando los usuarios no tienen un objetivo preciso y sólo desean recorrer el hiperdocumento [Wate91]. Entre estos dos extremos, existen necesidades del usuario que tienen componentes de ambas técnicas, y es por ello que surgen propuestas para combinar hipermedia con los métodos tradicionales de acceso a la información.

La posibilidad de que el lector cuente con *caminos*, constituídos por un conjunto acotado de nodos y enlaces, puede ayudar a solucionar estos problemas, ya que se reduce el número de opciones disponibles para el lector. Los usuarios se encuentran menos desorientados cuando recorren caminos predefinidos, en lugar de navegar libremente por el hiperdocumento. Estos caminos ofrecen pocas alternativas de recorrido (en algunos casos ofrecen sólo una), lo cual ayuda al lector, por un lado a no perderse en el universo total de nodos y, por otro lado, a no acceder a una gran cantidad de nodos lo que podría resultarle totalmente confuso. Estas herramientas permiten explorar el grafo total y disminuir su complejidad pues el lector tendrá acceso a un conjunto reducido de nodos y enlaces. Además, la construcción de esta clase de caminos le permite al autor del hiperdocumento organizar secuencias de presentación adecuadas para distintos tipos de audiencias [Zell91]. La existencia de caminos no intenta suplantar el mecanismo de navegación propio de hipermedia, sino que propone nuevas alternativas de recorrido.

2.3. MHEF: Modelo Hipermedial con Estadísticas y Filtros.

En la sección anterior se describieron las limitaciones que ofrece hipermedia cuando el dominio de aplicación es vasto y complejo. Los dominios con tales características generan en términos de hipermedia una red de información altamente interconectada y con gran número de referencias cruzadas. En la mayoría de las disciplinas, los conceptos a desarrollar tienen una naturaleza jerárquica y con asociaciones bien definidas. Cada uno de estos conceptos puede presentarse a través de una definición intuitiva y simple o también dando una descripción detallada y rigurosa con múltiples asociaciones con otros conceptos. La recuperación de información de interés en este tipo de grafos es una tarea dificultosa, que desalienta al lector en la búsqueda de los conceptos que considera relevantes según sus necesidades [Díaz93]. El lector cuenta con muchas alternativas para continuar su tarea y pocas ayudas que las prioricen. A un lector que está interesado en obtener una visión informativa del tema en cuestión, le es absolutamente improductivo y confuso recuperar información con excesivo grado de detalle, además de desalentarlo en la búsqueda de los temas de su interés.

El objetivo de este proyecto es el desarrollo del modelo MHEF (**M**odelo **H**ipermedial con **E**stadísticas y **F**iltros) basado en hipermedia, que permite la organización y recuperación de grandes volúmenes de información, apuntando a solucionar las limitaciones propias de hipermedia cuando se lo aplica a dominios extensos.

> *El modelo propuesto tiene como finalidad ofrecerle al lector un conjunto de herramientas, que lo ayuden a reducir la sobrecarga cognitiva de la navegación en dominios complejos y que puedan ser combinadas en secuencia y en alternativas, según sus necesidades y gustos.*

Objetivos del modelo:

- **Proveer ayudas para simplificar la exploración:** ofrecerle al lector pistas y simplificaciones de su entorno de trabajo, para evitar que deba enfrentarse con la totalidad del dominio o hiperespacio.

- **Permitirle al lector encontrar sin dificultad los conceptos en los que está realmente interesado:** la intención del lector no sólo debe ser implícita, sino que describirla explícitamente puede permitirle al sistema mostrarse desde una perspectiva más adecuada a sus propósitos.

- **Proveer mecanismos para facilitar la comprensión del tema tratado:** además de favorecer la exploración, la intención del modelo es facilitar la lectura y comprensión de temas.

Teniendo en cuenta estos objetivos, el modelo MHEF propone enriquecer hipermedia incorporando facilidades que posibiliten, por un lado, simplificar y facilitar la búsqueda de la información (reduciendo la sobrecarga cognitiva de la navegación); y por otro, facilitar la visualización de la información a nivel de base de datos (hiperespacio) y a nivel de nodo. Estas facilidades permiten obtener *vistas reducidas* del hiperespacio y están basadas en simplificaciones estructurales y cognitivas del mismo [Díaz94a] [Díaz94b].

Además, el modelo provee herramientas que le permiten al lector mantener información privada dentro del mismo ambiente, y de esta manera obtener su propia versión del documento que está leyendo.

La funcionalidad provista por MHEF lo convierte en una herramienta adecuada para manejar de una manera flexible grandes volúmenes de información. Las facilidades propuestas por el modelo planteado posibiltan:

- Particionar el dominio de acuerdo al perfil del usuario.

- Agregar filtros que permitan obtener una visión reducida del hiperdocumento.

- Proveer circuitos sugeridos por el autor.

- Realizar anotaciones personalizadas en los nodos.

- Mantener un registro de los nodos más visitados.

Para ilustrar la adecuación de MHEF se construyó un prototipo funcional llamado IMHEF (Instanciación de **MHEF**) sobre el dominio de redes de datos, que cuenta con las características de extensión y complejidad requeridas.

En la siguiente sección describiré detalladamente cada una de las facilidades provistas por MHEF, y en secciones posteriores mostraré su aplicación en el dominio elegido.

2.3.1. Funcionalidad del modelo propuesto

Como se ha descripto anteriormente, el modelo MHEF provee mecanismos sofisticados de visualización y recuperación de la información que apuntan a solucionar las limitaciones que presenta hipermedia cuando el campo de aplicación es extenso y complejo. Las facilidades propuestas por MHEF permiten manipular grandes volúmenes de nodos y enlaces a través de la generación de subgrafos construídos bajo distintos criterios y la visualización esquemática de los mismos.

2.3.1.1. Filtros: para simplificar la visualización del hiperespacio

El mecanismo de filtrado apunta a disminuir la complejidad del espacio con el que trata el usuario, mostrando sólo los nodos y vínculos que son de su interés.

Se definieron dos métodos de filtrado, uno más general que depende del perfil e intereses del lector y otro más específico basado en la selección de palabras claves.

- **Filtrado de acuerdo al modelo del usuario**

En los dominios amplios el volumen de información que se maneja es muy grande y resulta difícil mostrar todos los tópicos de una manera ordenada y clara. Esta clase de dominios generan en términos de hipermedia un grafo con gran cantidad de nodos y vínculos entre ellos. Cómo se le puede mostrar a un usuario un grafo con estas características?. La alta densidad de nodos y enlaces puede tornar imposible la visualización en una ventana de una red con centenares o miles de nodos. Si el usuario tiene necesidad de recorrer una red de información demasiado extensa y altamente interconectada, es factible que se pierda en todo ese cúmulo de información y no encuentre los datos que satisfagan sus necesidades.

Por otro lado, en la mayoría de las disciplinas cada uno de los conceptos puede presentarse con diferentes niveles de complejidad, a través de una definición intuitiva y simple, como también dando una descripción detallada y rigurosa con múltiples asociaciones con otros conceptos. A un lector que carece de especialización técnica y sólo está interesado en obtener una visión informativa del tema en cuestión, le resultará totalmente confuso e improductivo recuperar información con tanto grado de detalle, además de desalentarlo en la búsqueda de los tópicos en los que realmente está interesado.

Cuando los dominios son descriptos exhaustivamente, la gran cantidad de información disponible cubre intereses absolutamente divergentes. Las necesidades de los distintos usuarios varían en un amplio espectro, desde conceptos muy generales hasta aspectos muy técnicos. Por esta razón, se propone un método de filtrado que se basa en las características y necesidades de los distintos usuarios. Este método consiste en la clasificación de los potenciales usuarios en categorías bien definidas, y a partir de esta categorización particionar el dominio. De esta manera, la información se organiza en conjuntos de nodos y enlaces esencialmente disjuntos, que se

corresponden con cada categoría de usuario. Estos conjuntos, generalmente, comparten muy pocos tópicos. Para el usuario final, este efecto no se diferencia de navegar distintos hiperdocumentos (uno por cada nivel), sin embargo a los fines de la autoría, crear y mantener una única estructura evita incoherencias y simplifica el diseño (Ver Tabla 2).

El lector autodefine su propósito de lectura al comenzar una sesión de trabajo y solamente tendrá acceso a la versión del hiperespacio correspondiente al nivel elegido durante toda la sesión. Este mecanismo de filtrado "mejora" la navegación ya que el espacio de búsqueda se reduce en cantidad y complejidad (el lector tiene disponible sólo la información adecuada a su nivel) (Ver Tabla 1). Esta herramienta es particularmente útil cuando el dominio de aplicación es muy amplio, ya que evita que un lector acceda a información que pueda confundirlo y distraerlo de sus intereses.

A continuación se dará una definición rigurosa del alcance de la facilidad descripta previamente:

Sea $\mathcal{H}$ el grafo que representa el hiperespacio, $\mathcal{H} = (V,A)$ y $A \subset V \times V$.

Sea $\mathcal{H}_{Nivel}=(V_{Nivel},A_{Nivel})$, el hiperespacio que *navega* un lector.

Siendo $V_{nivel} = \{ v : v \in V \wedge Nivel \in Nivel(v) \}$

$A_{Nivel} = A \cap V_{Nivel} \times V_{Nivel}$ y Nivel(n) el nivel/niveles asociados al nodo n.

- **Filtrado por palabra clave**

El lector tiene varias posibilidades de filtrar la información de su interés. Una de ellas, es el método de filtrado por palabra clave que permite recuperar los nodos relacionados a un tópico específico. Este mecanismo de filtrado está basado en una clasificación de los nodos por parte del autor, según el contenido de los mismos (Ver Tabla 2).

Para la realización de este método de filtrado, además del hiperdocumento se cuenta con un *índice jerárquico de conceptos* (según la terminología de árbol) en forma similar al sistema OpenBook [Ichi93b]. Cada nodo del hiperdocumento es clasificado de acuerdo a su contenido [Hala87b] y se le asocia una o más palabras claves o conceptos. Estos están organizados usando la relación IS-A, que establece una jerarquía entre ellos. Los niveles inferiores en una jerarquía representan mayor especialización de los conceptos, mientras que los niveles superiores representan una mayor generalización.

El resultado de una consulta realizada por el lector es la recuperación de la subred correspondiente a un criterio específico de selección, eligiendo los nodos clasificados por una palabra clave específica y por aquellas contenidas conceptualmente en dicha palabra clave (Fig. 2). Además, este mecanismo realiza un filtrado sobre los vínculos, habilitando sólo aquellos vínculos que relacionan los nodos seleccionados. Por ejemplo, supongamos que un usuario

quiere recuperar los nodos que tratan sobre el concepto c2, por lo cual activa el mecanismo de filtrado y elige dicho concepto. El filtro busca los nodos clasificados por c2, c21 y c22. El resultado de la búsqueda son los nodos {N2,N4} y durante el estado de filtro, se habilitará sólo el vínculo que relaciona estos dos nodos.

Esta herramienta le posibilita al usuario una recuperación selectiva de los nodos y le ofrece un espacio de búsqueda acotado, en el que solamente tiene acceso a aquellos nodos recuperados a través del filtro. Este método de filtrado permite realizar un acceso directo a los nodos y vínculos del hiperespacio, ya que se le agrega a la navegación un mecanismo adicional de búsqueda directa (Ver Tabla 1).

La técnica de filtrado descripta provee un medio para simplificar el acceso del usuario a un gran conjunto de posibilidades, permitiéndole reducir el espacio total de información. Esto resulta particularmente adecuado para los dominios que cuentan con una clasificación clara y perfectamente definida de sus conceptos.

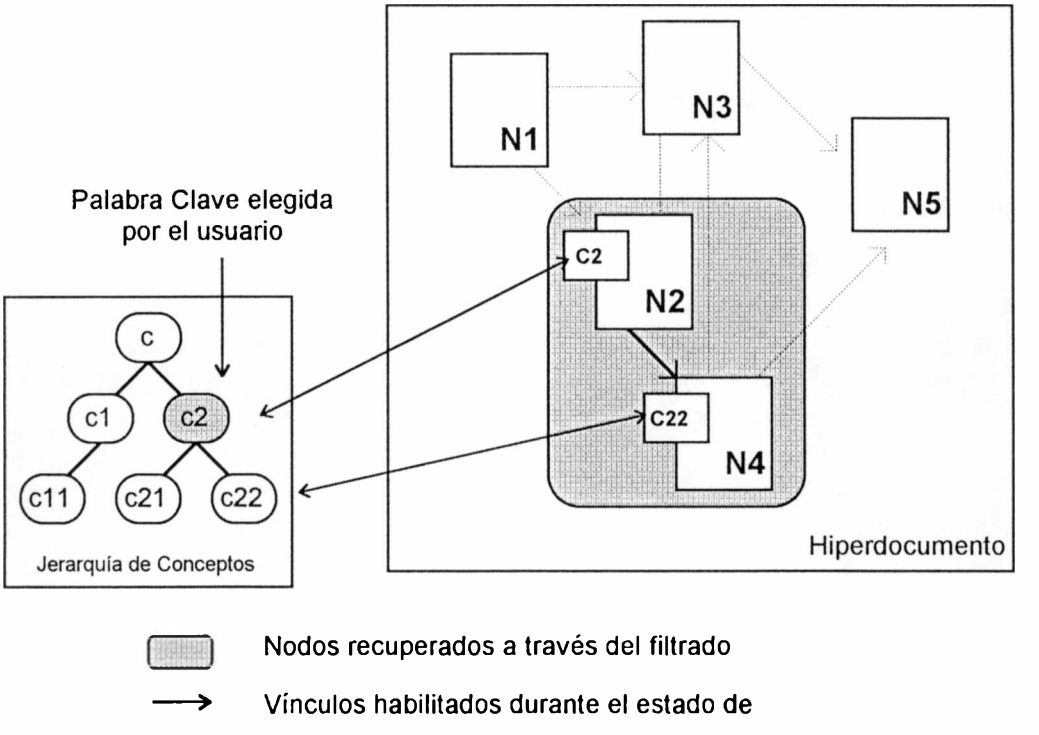


Fig. 2. Nodos y vínculos recuperados a través del método de filtrado por palabra clave.

Para precisar los alcances de esta característica se dará un definición formal para la misma:

Sea $\mathcal{H}$ el grafo que representa el hiperespacio, $\mathcal{H}$ = (V,A) y A $\subset$ V x V.

Sea $\mathcal{H}_{Nivel}=(V_{Nivel},A_{Nivel})$, el hiperespacio que *navega* un lector, después de haber aplicado el filtro de acuerdo al modelo del usuario.

Sea $\mathcal{F}_{pal}(\mathcal{H}_{Nivel})= (\mathbf{VF}_{pal}, \mathbf{AF}_{pal})$, tal que $\mathbf{VF}_{pal}$ y $\mathbf{AF}_{pal}$ se definen de la siguiente manera:

$$\mathbf{VF}_{pal} = \{n: n \in V_{Nivel} \wedge pal \in ClavesDerivadas(n)\}$$

$$\mathbf{AF}_{pal} = \{(a,b): a,b \in VF_{pal}\}$$

$$ClavesDerivadas(n) = \begin{cases} Claves\ (n) \\ Derivar\ (Claves\ (n)) \end{cases}$$

Siendo Claves(n) las palabras claves asociadas al nodo n, y Derivar(Claves(n)) las palabras claves dependientes de Claves(n) según la relación ES_UN (definida previamente) en el Thesaurus correspondiente al dominio del hipertexto.

Comparación entre ambos métodos de filtrado:

En el filtrado de acuerdo al modelo del usuario se ha mejorado la búsqueda de la información de interés, pues **se redujo el espacio de búsqueda en cantidad y complejidad**, a pesar de ser la navegación el único mecanismo de recorrido del hiperdocumento. Este mecanismo de filtrado es obligatorio.

El filtrado por palabra clave permite **acceder en forma directa** a los nodos que tratan específicamente sobre el tópico de interés del lector. Este método consiste en brindarle al lector un mecanismo adicional de búsqueda directa en el hiperespacio. A diferencia del método de filtrado anterior, el filtrado por palabra clave se construye por demanda del lector, y lo ayuda a encontrar dentro del hiperespacio los nodos correspondientes a su foco de atención.

2.3.1.2. Vistas de la hipermedia: para evitar la "desorientación del usuario"

Las facilidades descriptas en esta sección constituyen herramientas que le permiten al lector obtener visiones reducidas del hiperespacio, y de esta manera disminuir el problema de la desorientación. Con cada una de ellas se obtienen estructuras de recorrido simplificadas (árboles y caminos lineales) construídas según distintos criterios de selección. Estas estructuras ofrecen menos alternativas de navegación que el espacio total, lo cual ayuda al lector a no perderse durante la navegación.

- **Tours Guiados**

Un tour guiado es un soporte computacional a través del cual el autor brinda pistas a los lectores. Un tour guiado es básicamente una facilidad de características interactivas basada en la representación gráfica de un grafo que representa la estructura de presentación de la red de nodos [Mars89].

La metáfora de proveer un tour guiado en un sistema de hipertexto fue propuesta en primer término por Hammond y Allinson [Hamm87], y luego utilizada por Trigg [Trigg88]. En esta oportunidad, los tours guiados eran creados por el autor del hiperdocumento y constituían una facilidad completamente estática y fija.

En forma similar a la propuesta de Guinan [Guin92], en este modelo la construcción de un tour está basada en información estadística. Los nodos y vínculos pertenecientes al tour guiado no son seleccionados por el autor, sino que son seleccionados automáticamente, teniendo en cuenta los nodos más visitados y los vínculos más accedidos [Frei92] [Belk92] (Ver Tabla 2).

Para la construcción del tour guiado, se lleva un registro con el número de visitas de cada nodo y se calcula el *promedio total de visitas* del grafo al momento de generar el tour. El promedio total de visitas es la suma del número de visitas de todos los nodos dividido el total de nodos del grafo. Los nodos cuya frecuencia de visitas superen el valor promedio, formarán parte del tour guiado.

Como ya fue mencionado, en un principio los tours guiados eran una herramienta previamente definida por el autor del hiperdocumento y constituían un camino con un recorrido fijo. En el modelo propuesto, el tour que se le presenta al lector puede cambiar de una sesión a otra, a medida que cambia la frecuencia de visitas de los nodos.

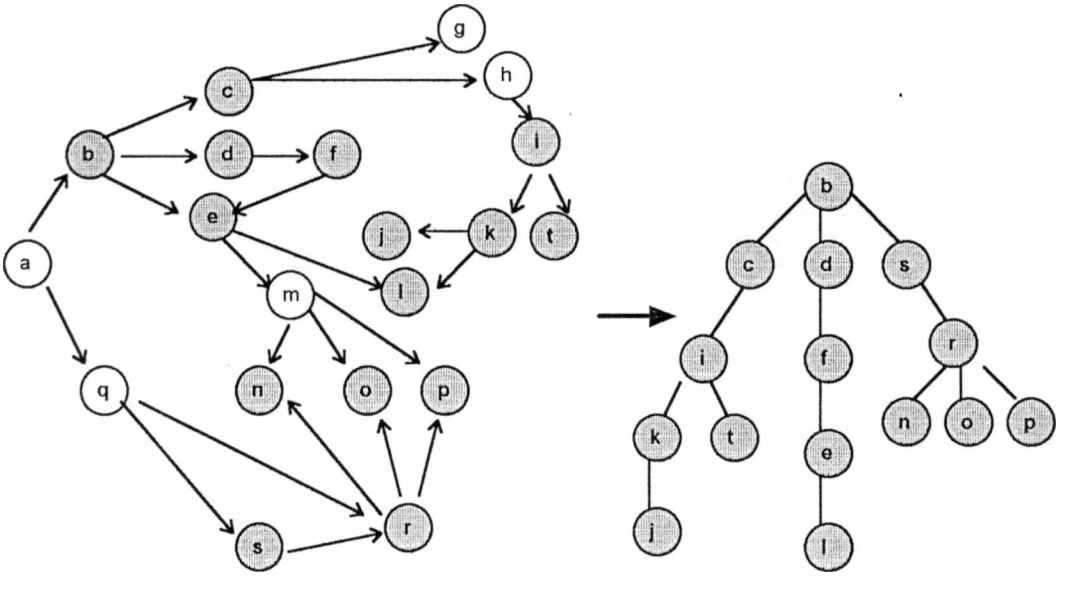


Fig. 3 a. Digrafo resultante de estadísticas de uso.**b**.Jerarquización del digrafo

El método de construcción del tour consiste en una *jerarquización dinámica* del grafo del hipertexto, construyendo un espacio de navegación, cuya base es un **árbol** de nodos. Esto se logra transformando los subgrafos, obtenidos a través de la selección estadística, en subárboles, que serán organizados de acuerdo a un criterio de cercanía entre ellos. La raíz del árbol resultante será la raíz del subárbol más cercano a la raíz del grafo del hiperdocumento.

Por ejemplo, supongamos que después de varias sesiones de trabajo, los nodos más visitados son los que aparecen en la Fig. 3a. Los subgrafos conexos son transformados en subárboles y luego organizados en una estructura de árbol general, como se muestra en la Fig. 3b.

Esta herramienta apunta a solucionar el problema de la desorientación pues le ofrece al lector un espacio de búsqueda en el que existen menos alternativas de navegación que en el espacio total (Ver Tabla 1). Una manera de "ubicar" a un lector en un hipertexto es imponiéndole una estructura de datos sobre la totalidad del hiperdocumento y de esta manera facilitarle la lectura y la comprensión de los temas tratados [Rivl94]. Una estructura natural para este propósito es la jerárquica. El lector al navegar sobre un árbol tiene menos posibilidades de *perderse* durante la navegación.

Esta forma de generar el tour guiado es independiente del dominio de discurso y está basado en el uso empírico del sistema. La principal característica de este tipo de tour guiado es que evoluciona con el tiempo, esto es, el grafo se mostrará de diferentes maneras dependiendo del uso del sistema a lo largo del tiempo.

A continuación se dará una definición rigurosa del alcance del tour guiado:

Sea $\mathcal{H}$ el grafo que representa el hiperespacio, $\mathcal{H} = (V,A)$ y $A \subset V \times V$.

Sea $\mathcal{H}_{Nivel}=(V_{Nivel},A_{Nivel})$, el hiperespacio que *navega* un lector, después de haber aplicado el filtro de acuerdo al modelo del usuario.

Sea Δ_{Nivel} el nodo origen de $\mathcal{H}_{Nivel}$.

A continuación se definirán algunas funciones sobre el grafo $\mathcal{H} = (V,A)$:

1.-Función incidencia:

$i_A: V \rightarrow Nat$, tal que $i_A(n) = \left|\{ (d,n) \in A \}\right|$

2.-Función camino:

$\mathbf{Camino}_A: V \times V \rightarrow Boolean$

$$\mathbf{Camino}_A(a,b) = \begin{cases} 1, \text{ si } \exists\, x_1,..,x_n \in V \text{ tq. } x_1 = a \text{ y } x_{n+1} = b \text{ y } (x_i, x_{i+1}) \in A \\ 0, \text{ en caso contrario} \end{cases}$$

3.-Función visitas:

Visitas$_V$: V $\rightarrow$ Nat y $\bar{s} = \sum$ **Visitas**$_V$**(n) /** |V|

4.-Función distancia:

$\mathbf{d}_\Lambda$: VxV$\rightarrow$ Nat

$\mathbf{d}_\Lambda$(n,n) = 0

$\mathbf{d}_\Lambda$(a,b)= 1 sii $\exists$ (a,b)$\in$ A

$\mathbf{d}_\Lambda$(a,b) = **Mínimo** $\{$ n: si $\exists$ $x_1,..,x_n \in V$ tq. $x_1 = a$ y $x_{n+1} = b$ y $(x_i, x_{i+1}) \in A$ $\}$

Definición del subgrafo de $\mathcal{H}_{\text{Nivel}}$ asociado a los nodos más visitados:

Sea W = $\{$ n: n $\in V_{\text{Nivel}} \wedge$ **Visitas**$_{V\text{Nivel}}$ (n) > $\bar{s}$ $\}$

Se define $\mathcal{S}$ = (W, $A_{\text{Nivel}} \cap$ (W x W))

Después de haber analizado los dos algoritmos más conocidos de construcción de árboles minimales a partir de grafos (Prim y Kruskal) [Aho83] [Weis92], y dado que las aristas del grafo $\mathcal{S}$ no tienen peso, se usó un recorrido breadth-first search a partir de los vértices de menor incidencia, para construir los árboles de cada una de las componentes conexas.

Sean $T_1,.....,T_n$ los árboles generadores del grafo $\mathcal{S}$.

Cada T_i se puede expresar como ($\mathcal{V}(T_i)$, $\mathcal{A}(Ti)$), donde $\mathcal{V}(T_i)$ son los vértices de T_i y $\mathcal{A}(Ti)$ son las aristas de T_i.

Podemos expresar a W de la siguiente manera: $W = \bigcup_{i=1}^{n} \mathcal{V}(T_i)$.

Se definirán n-1 aristas para interconectar los árboles entre sí.

Sea r_i la raíz de T_i.

Se elige k tal que: $\mathbf{d}_{\Lambda\text{Nivel}}(\mathcal{S}_{\text{Nivel}}, r_k) \leq \mathbf{d}_{\Lambda\text{Nivel}}(\mathcal{S}_{\text{Nivel}}, r_i) \quad \forall i$

Tomamos como árbol inicial a T_k, y llamamos $Y_0 = T_k$

Una vez definido Y_i , definimos Y_{i+1} de la siguiente manera:

a.- Si i+1 = k , entonces $Y_{i+1} = Y_i$

b.- Si i+1 ≠ k , se pueden presentar dos casos:

1er. Caso: $\exists$ v $\in Y_i$, tal que: $\mathbf{Camino}_{\mathbf{A}\text{Nivel}}(v, v_{i+1})=1$

Entonces se elige x $\in Y_i$, tal que $\mathbf{d}_{\text{ANivel}}(x, v_{i+1}) \leq \mathbf{d}_{\text{ANivel}}(t, v_{i+1})$ $\forall$ t $\in Y_i$.

Se define $Y_{i+1} = (\mathcal{V}(Y_i) \cup \mathcal{V}(T_{i+1}),\ \mathcal{A}(Y_i) \cup \mathcal{A}(T_{i+1}) \cup (x, v_{i+1}))$

2do. Caso: $\mathbf{Camino}_{\mathbf{A}\text{Nivel}}(v, v_{i+1})=0$ $\forall$ v $\in Y_i$

Se define $Y_{i+1} = (\mathcal{V}(Y_i) \cup \mathcal{V}(T_{i+1}),\ \mathcal{A}(Y_i) \cup \mathcal{A}(T_{i+1}) \cup (v_k, v_{i+1}))$

Al cabo de n-1 pasos se habrán tomado todos los T_i , y el árbol Y_n será el árbol definitivo.

El árbol asociado al tour guiado será: $\mathcal{TG} = \mathbf{Y_n}$.

- **Circuitos de lectura recomendada**

Otra herramienta incluída en el modelo son los llamados circuitos de lectura recomendada, cuyo objetivo es aliviar el esfuerzo requerido por el lector en la comprensión de los conceptos impartidos. Cuando el dominio de aplicación es amplio, el espectro de conceptos a abordar es muy extenso y variado. Cada concepto puede presentarse desde distintos puntos de vista, los que satisfacen los requerimientos de las diferentes clases de usuarios. Además de dar una definición formal y las principales características de cada uno de los tópicos, se muestra información adicional como ejemplos, comentarios, datos estadísticos que es posible que no resulten de interés para todos los lectores. Cada usuario puede detenerse y profundizar en los temas que él desee de acuerdo a sus preferencias. La posibilidad de estructurar dominios complejos bajo una red de hipertexto permite a cada usuario tener su propia configuración de recorrido y, por lo tanto, una gran libertad de acceso a la información. Sin embargo, se genera una red con un gran número de nodos y relaciones entre ellos, lo que ocasiona que al lector le resulte difícil encontrar los conceptos relevantes. Por esta razón, se construyeron circuitos de lectura

recomendada formados por nodos conceptualmente importantes (Ver Tabla 2). Estos nodos son seleccionados por el autor, de acuerdo a su importancia dentro del tema tratado. De la misma manera que en el tour guiado, en el circuito de lectura recomendada se le impone al lector una estructura de datos más simple y comprensible que el grafo original. Esta estructura de datos es una *linearización estática* del hiperdocumento, es decir una secuencia de nodos con un orden preestablecido y una única alternativa de recorrido (Ver Tabla 1). Es el autor quien establece el orden de visita para dichos nodos, es decir la secuencia que el lector debe seguir para poder completar los requisitos mínimos de cada tópico tratado. Los nodos que forman el circuito recomendado son seleccionados con el criterio de que son esenciales para obtener una comprensión global del tema en cuestión. Para cada categoría o nivel de usuario se define un circuito recomendado, de manera que un lector pueda acceder rápidamente a los conceptos que el autor considera importantes para ese nivel. Estos circuitos constituyen el conjunto mínimo de nodos del hiperespacio necesarios para que un lector pueda tener una comprensión global acerca del tema tratado, sin acceder a información no esencial.

A fines de precisar los alcances del circuito de lectura recomendada se dará una definición formal para el mismo:

Sea $\mathcal{H}$ el grafo que representa el hiperespacio, $\mathcal{H} = (V,A)$ y $A \subset V \times V$.

Sea $\mathcal{H}_{Nivel}=(V_{Nivel},A_{Nivel})$, el hiperespacio que *navega* un lector, después de haber aplicado el filtro de acuerdo al modelo del usuario.

Sea $\aleph_{Nivel}$ el nodo origen de $\mathcal{H}_{Nivel}$.

Sea **VC** $\subset V_{Nivel}$ el conjunto de nodos conceptualmente importantes elegidos por el autor y $\aleph_{Nivel} \in$ **VC**.

1.-El autor define un orden total sobre el conjunto VC (es decir un isomorfismo de VC con los naturales), tal que:

Orden: VC $\rightarrow \{0,1,.... |VC|-1\}$, siendo **Orden** una función biyectiva y **Orden**($\aleph_{Nivel}$)=0.

2.-A partir de un nodo n $\in$ VC, el lector sólo podrá acceder al nodo derivado directo según la función **Orden**:

a.-Función Siguiente de un nodo n:

Siguiente(n): $VC \rightarrow \varnothing \cup VC$

$$\textbf{Siguiente}(n) = \begin{cases} \varnothing, \text{ si Orden } (n) = |VC| - 1 \\ m \in VC \ \wedge \ \text{Orden } (m) = \text{Orden } (n) + 1 \end{cases}$$

b.-Función Anterior de un nodo n:

Anterior(n): $VC \rightarrow \varnothing \cup VC$

$$\textbf{Anterior}(n) = \begin{cases} \varnothing, \text{ si } n = \aleph_{Nivel} \\ m \in VC \ \wedge \ \text{Orden}(m) = \text{Orden}(n) - 1 \end{cases}$$

Siendo **TC**=$\{(a,b)$: a,b $\in$ VC $\wedge$ **Orden**(b)= **Orden**(a)+1$\}$, la secuencia asociada al circuito de lectura recomendada se define como: $\mathcal{CR}(\mathcal{H}_{Nivel})=(VC,TC)$.

	Criterio de selección definido	Vista resultante	Forma de lectura
Filtros según el perfil del lector	el autor clasifica y el lector se autoclasifica	estática	navegación
Filtros por palabra clave	el autor clasifica los nodos y el lector elige palabras claves	semiestática	búsqueda + navegación
Tours guiados	estadístico	dinámica	navegación reducida
Circuitos recomendados	autor	estática	recorrido

Tabla 1. Comparación entre filtros y vistas del hiperespacio

2.3.1.3. Anotaciones personalizadas: para mantener información privada

La facilidad descripta en esta sección tiene como objetivo primordial permitirle al lector mantener información privada dentro del mismo ambiente de consulta, brindándole la posibilidad de agregar sus propias anotaciones o comentarios.

El modelo MHEF ofrece una herramienta a través de la cual el lector tiene la posibilidad de crear anotaciones en los nodos que él elija. Esta facilidad permite contar con un ambiente de consulta personalizable de acuerdo a las necesidades y preferencias del lector. Las anotaciones o comentarios pueden agregarse en cualquier momento y en todos los nodos, y sólo son accedidas por la persona que las realizó. Tener la posibilidad de realizar anotaciones le permite al lector agregarle al contenido de los nodos sus propios comentarios, esto es, frases, ilustraciones, cuadros, gráficos, etc. De esta manera, el lector puede enriquecer la semántica de los nodos del hiperdocumento.

	A nivel de grafo	**A nivel de nodo**
Filtros según el perfil del lector	Activa particiones disjuntas del hiperdocumento.	---
Filtros por Palabra Clave	Genera subgrafos del dominio, de acuerdo al contenido, permitiendo el acceso a los nodos por contenido. Simplifica la búsqueda de información.	Restringe los vínculos activables.
Tours Guiados	Genera subgrafos del dominio, según estadísticas, de acuerdo a la frecuencia de lectura de los usuarios.	Establece nuevos vínculos de recorrido (sustituyendo los vínculos existentes).
Circuitos Recomendados	Permite al autor explicitar subgrafos minimales del dominio, de acuerdo a los conceptos impartidos. Facilita la comprensión de las ideas presentadas.	Restringe los vínculos activables. Establece vínculos secuenciales de recorrido (anterior y siguiente).
Anotaciones Internas al nodo	---	Personaliza el contenido.
Anotaciones de Referencia Terminal	Modifica la estructura. Genera un grafo personalizado.	Genera un nuevo vínculo a un nodo terminal.

Tabla 2. Descripción conceptual de las facilidades según niveles

En el modelo se proveen dos tipos de anotaciones: anotaciones internas y anotaciones de referencia terminal.

Anotaciones internas: estas anotaciones son internas al nodo en que se realizan. El usuario le puede asociar a cada nodo tantas anotaciones internas como desee. Ellas están estructuradas en una especie de agenda, donde cada página representa una anotación distinta. Una vez creadas, el usuario puede consultarlas tantas veces como desee o agregar una nueva anotación. Esta clase de anotación resulta útil, por ejemplo, cuando el lector quiere agregarle referencias bibliográficas a un nodo del documento, o datos actualizados del tema tratado en ese nodo.

Anotaciones de referencia terminal: estas anotaciones significan la creación de un enlace desde un nodo existente a un nodo creado por el lector con este propósito. Esto significa que la anotación de referencia terminal es un nodo externo, al que se accede a través de botones ubicados en el nodo origen (donde se realizó la anotación). El lector puede crear sólo una anotación externa en cada nodo. Las anotaciones de este tipo permiten crear un nodo nuevo, donde el usuario puede ingresar información textual.

3. APLICACION DEL MODELO A UN DOMINIO CONCRETO: REDES DE COMPUTADORAS

3.1. Redes de computadoras: definición y objetivos

> *Una red de computadoras es una combinación de equipos y programas interconectados usados para transportar información entre distintos puntos donde la misma podrá ser procesada, almacenada o usada.*

Los objetivos primordiales de las redes de computadoras son [Tane91]:

- Compartir recursos: el objetivo es hacer que todos los programas, datos y equipos estén disponibles para cualquier usuario de la red que así lo solicite, sin importar la localización física del recurso y del usuario. En otras palabras, el hecho de que el usuario se encuentre a 1000 km de distancia de los datos, no debe evitar que éste los pueda utilizar como si fueran originados localmente.

- Alta fiabilidad: las redes proporcionan una alta fiabilidad al contar con fuentes alternativas de suministro. Por ejemplo, todos los archivos podrían duplicarse en dos o tres máquinas, de tal manera que si una de ellas no se encuentra disponible (como consecuencia de una falla), podría utilizarse alguna de las otras copias. Para aplicaciones bancarias, militares, de control de tráfico aéreo y muchas otras, es muy importante la capacidad de los sistemas para continuar funcionando a pesar de existir problemas de hardware.

- Ahorro económico: las computadoras pequeñas tienen una mejor relación costo/rendimiento, comparada con la ofrecida por las máquinas grandes. Estas son, a grandes rasgos, diez veces más rápidas que el más rápido de los microprocesadores, pero su costo es miles de veces mayor.

- Poderoso medio de comunicación: con el empleo de una red es relativamente fácil para dos o más personas que viven en lugares separados, escribir un informe juntas. Cuando un autor hace un cambio en un documento que se mantiene en línea, los otros pueden ver el cambio de inmediato, en lugar de esperar varios días para recibirlo por carta. Esta rapidez hace que la cooperación entre grupos de individuos que se encuentran alejados, y que anteriormente había sido imposible de establecer, pueda realizarse ahora.

3.2. Por qué la elección del tema Redes de computadoras ?

En la actualidad es muy común la utilización de computadoras para procesar información (por ej. utilizando procesadores de palabra, planillas de cálculo y bases de datos). De esta forma, ya no sólo residen en la computadora datos para

procesos numéricos sino múltiples representaciones de información. Todo este cúmulo de datos es susceptible de ser transmitido a un número cada vez mayor de personas que utilizan computadoras. Esta tendencia hace que tengan interés las redes de computadoras como medio de comunicación de amplio espectro. Por esta razón, las redes se encuentran presentes en la mayoría de las instituciones, desde las académicas y gubernamentales hasta las organizaciones comerciales. Las instituciones que utilizan redes difieren en el tiempo y en el espacio de los centros de trabajo convencionales. Las comunicaciones basadas en redes son extremadamente rápidas y eficientes si las comparamos con el teléfono o el servicio postal. Las redes permiten intercambiar información con el otro extremo del mundo en cuestión de minutos; este intercambio puede involucrar a múltiples sitios. Las redes posibilitan que el tiempo no transcurra. Los mensajes electrónicos pueden conservarse indefinidamente, y así los usuarios pueden leerlos tantas veces como quieran, copiarlos, modificarlos o enviarlos a otros usuarios. Los científicos utilizan las redes académicas para intercambiar ideas, escribir artículos técnicos y mantener contacto con otros grupos de investigación.

Dada la versatilidad de los servicios brindados por las redes electrónicas de datos, éstas cuentan con una amplia gama de usuarios provenientes de diferentes ámbitos: desde ingenieros que intervienen en la especificación de los protocolos de comunicación; especialistas en informática que usan la red para el desarrollo de aplicaciones distribuídas; científicos que usan el correo electrónico; empleados de empresas que se comunican con sucursales ubicadas en lugares distantes e intercambian información que acelera la toma de decisiones hasta niños que acceden a enciclopedias distribuídas a través de redes.

Para mostrar la complejidad del dominio de Redes podemos decir que existen actualmente más de 1800 documentos llamados RFC (Request for Commnets) que abordan distintos tópicos dentro de este dominio y están orientados a un amplio espectro de usuarios. Los temas abarcan desde especificación de protocolos (por ejemplo RFC 661, RFC 959, RFC 1780, RFC 1792, RFC 1805 por nombrar sólo algunos), estándares para los mensajes enviados a través del correo electrónico (RFC 822, RFC 934, RFC 1049, RFC 1341, etc.); hasta respuestas para usuarios tanto novatos como expertos en Internet (RFC 1207, RFC 1462, RFC 1463, RFC 1594 etc.). Estos RFC's pueden ser consultados por cualquier usuario a través del correo electrónico.

Como se ha descripto en las secciones anteriores, el modelo de hipermedia permite descomponer un tema en una jerarquía semántica, donde se establecen distintos tipos de vínculos: a) entre conceptos relacionados, por ejemplo cuando el usuario se está informando sobre los medios físicos para la transmisión de información multimedial le resultará útil poder hacer una analogía entre los soportes usados en redes LAN y WAN, y b) para obtener información adicional, por ejemplo cuando se quiere conocer el significado de una sigla o de un término técnico muy especifico.

Esta técnica posibilita estructurar temas complejos y amplios, como es el de redes de computadoras, en un reticulado de conceptos, donde cada nodo representa un subtema y los vínculos, las relaciones semánticas descriptas. Desde el punto de vista del lector, este tipo de organización le facilita el acceso a la información evitando enfrentarse con una gran cantidad de conceptos simultáneamente y sólo recuperar aquellos datos que son de su interés,

construyendo de esta manera su propio enfoque del tema. Por ejemplo, si un lector quiere informarse sobre los servicios ofrecidos por las redes, puede filtrar rápidamente aquellos nodos que traten sobre este tópico.

Por otro lado la flexibilidad con respecto a la presentación de la información alienta al lector a la compenetración en el tema y ayuda a comprender de una manera más efectiva cada uno de los conceptos; por ejemplo, es más didáctico observar una animación sobre el correo electrónico que muestre la forma en que se envía una carta, que un simple texto que lo explique.

3.3. Limitaciones de hipermedia para la implementación de un sistema de consulta sobre redes de computadoras

Como fue explicado previamente, hipermedia es una herramienta adecuada para organizar la información de una manera flexible, permitiéndole al usuario acceder solamente a la información que él considera relevante. El diseño de un sistema de consulta sobre redes de computadoras basado en hipermedia, cuya funcionalidad consiste en poder recuperar información sobre los conceptos básicos, la forma de uso y los servicios provistos por las redes de computadoras en general y en particular por las redes académicas, requiere de facilidades adicionales al modelo básico de hipermedia.

El dominio de Redes tiene una amplia difusión tanto en lo que hace a la diversidad de público usuario (BBS, Internet, etc.) como por la gran variedad de servicios que ofrecen (desde Mosaic hasta video por demanda) y la constante efervescencia de tecnologías emergentes (ATM, SMDS, etc.). Por lo expuesto, resulta obvio que la única herramienta idónea para organizar esta información es mediante un hiperdocumento, sin embargo cabe destacar que si no se provee la funcionalidad que caracteriza a MHEF, entonces la tarea de búsqueda de conceptos específicos y la lectura de información actualizada sería prácticamente imposible.

El dominio de redes de datos es amplio y complejo y genera un grafo con gran cantidad de nodos y múltiples vínculos entre ellos. El lector está proclive a "perderse" o "desorientarse" al acceder a un espacio de búsqueda con estas características. Los conceptos que componen el dominio de esta aplicación tienen una naturaleza jerárquica y con relaciones bien establecidas (por ejemplo el concepto de capas en comunicaciones), de cada uno de ellos puede darse desde una definición intuitiva y simple hasta una descripción completamente rigurosa y con múltiples asociaciones con otros conceptos. Al tratar, por ejemplo, el tema Protocolos de comunicación puede presentarse simplemente una enumeración de los protocolos existentes o llegar a una descripción detallada de las capas que componen a cada uno de ellos, de acuerdo al modelo OSI. A un lector que carece de especialización técnica, obtener tal grado de detalle le resultará totalmente confuso y lo desalentará en la investigación del tema, resultando completamente improductivo para sus necesidades. El lector puede internarse en la red de nodos y llegar a un punto donde le resulte casi imposible ubicarse conceptualmente dentro del tema, sin poder recuperar las ideas que realmente le interesan. Por ejemplo, un lector que está interesado en saber cómo usar el servicio de e-mail provisto por las redes, seguramente no le interesa conocer el medio físico usado para la transmisión ni los formatos de dirección propios de cada protocolo.

Dado el gran número de conceptos que se deben abordar al tratar el tema redes de computadoras, resulta de escasa utilidad construir un sistema de consulta que no cuente con herramientas que asistan a los lectores durante el acceso a la información. Estas herramientas deben facilitar la búsqueda de la información de interés y evitar que el lector acceda a un conjunto de datos completamente irrelevantes para él.

La construcción de un ambiente de consulta sobre redes de datos es un tema interesante y nada sencillo, ya que involucra un análisis de los distintos tipos de usuarios y una cuidadosa recopilación y organización de los tópicos a desarrollar. La información debe estar organizada de manera que satisfaga las inquietudes de los distintos usuarios, sin que estos se enfrenten con un espacio de información que les resulte difícil de controlar.

Por lo expuesto anteriormente, se eligió el dominio de redes de computadoras para construir el prototipo funcional IMHEF (Instanciación de MHEF), basado en el modelo descripto anteriormente.

En la próxima sección describiré la instanciación de cada una de las facilidades provistas por el modelo, en el prototipo implementado.

3.4. Principios básicos para la aplicación de las facilidades de MHEF al dominio elegido

El objetivo de este proyecto es presentar el modelo MHEF, basado en hipermedia, que provee un conjunto de herramientas para la organización y recuperación de grandes volúmenes de información. Este modelo apunta a solucionar las limitaciones propias de hipermedia cuando es aplicada a dominios extensos.

Para mostrar la adecuación del modelo propuesto, se construyó IMHEF, un prototipo funcional sobre el dominio de redes de computadoras que, como fue explicado previamente, cuenta con características de extensión y complejidad que lo tornan imposible de representar usando un modelo básico de hipermedia.

En esta sección describiré la aplicación de cada una de las facilidades provistas por el modelo sobre el dominio elegido.

Las herramientas propuestas en MHEF, por un lado, simplifican la visualización y recuperación de la información a través de la generación de subgrafos construidos bajo distintos criterios y, por otro lado permiten obtener un ambiente de consulta personalizado.

Para la descripción de cada una de las facilidades seguiré la clasificación expuesta anteriormente:

- **Filtros: para simplificar la visualización del hiperespacio**

  Filtrado de acuerdo al modelo del usuario: la gran cantidad de información disponible en el campo de la tecnología de redes cubre intereses absolutamente divergentes. Las necesidades de los distintos usuarios varían en un amplio espectro, desde conceptos muy generales hasta aspectos muy técnicos (como información de detalle sobre protocolos de comunicación) pasando por aprender a usar los servicios ofrecidos por las redes (como

correo electrónico, mensajes interactivos, teleconferencias, grupos de interés, etc.).

Teniendo en cuenta los intereses de los potenciales usuarios de nuestro sistema, se establecieron tres categorías de lectores: de difusión o coloquial, práctico o de uso común y teórico o de referencia. El nivel coloquial implementa el aprendizaje de los conceptos básicos de redes en un lenguaje no técnico. El nivel de uso está basado en los aspectos de usabilidad de las redes, centrándose en el uso práctico de los diferentes servicios ofrecidos por las redes. Por último, el nivel de referencia presenta los aspectos más técnicos y especializados de redes.

La organización planteada, tiene como consecuencia inmediata la división del dominio en tres conjuntos de nodos y enlaces esencialmente disjuntos, en donde sólo se comparten muy pocos tópicos.

En el momento que el lector comienza la sesión de trabajo, él mismo elige el nivel que considera más adecuado de acuerdo a sus intereses o preferencias [Soko88]. Por ejemplo, un científico que solamente usa algunos de los servicios ofrecidos por las redes como correo electrónico y teleconferencias, seguramente no estará interesado en información sobre medios físicos de transmisión o protocolos internos de comunicación, con lo cual carece de sentido presentarle un hiperdocumento que contenga este tipo de información. Por otro lado, si el lector es un profesional de informática estará interesado en aspectos técnicos como por ejemplo niveles de seguridad y de performance en la transmisión de la información, a tener en cuenta para instalar una red.

Filtrado por palabra clave: este mecanismo de filtrado permite recuperar los nodos relacionados con un tópico específico. Para ello, se cuenta con un índice jerárquico de conceptos y una clasificación de los nodos por parte del autor, según el contenido de los mismos.

El lector elige una palabra clave y el resultado de la consulta es la recuperación de la subred integrada por los nodos clasificados por dicha palabra clave y sus términos derivados.

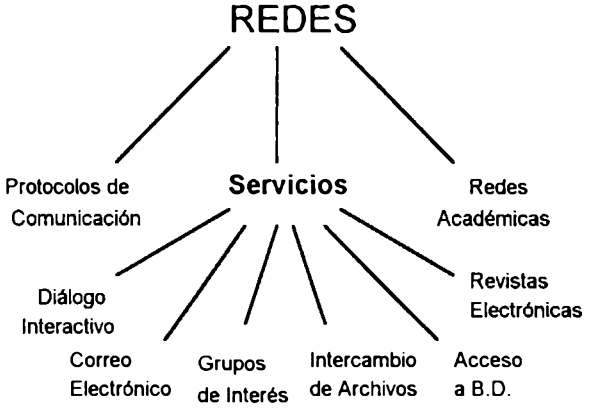


Fig. 4. Parte del índice jerárquico de conceptos para el nivel coloquial

Al aplicar esta funcionalidad en el prototipo implementado, se creó un índice de conceptos para cada uno de los niveles de usuario descriptos anteriormente. Por ejemplo, si un lector del nivel coloquial desea navegar por aquellos nodos que traten el tema "Servicios ofrecidos por las redes", deberá filtrar el grafo por la palabra clave "Servicios" y el resultado será un subgrafo donde solamente aparezcan los nodos relacionados con la palabra clave elegida y sus derivadas, como por ejemplo "Diálogo interactivo", "Correo Electrónico", etc..

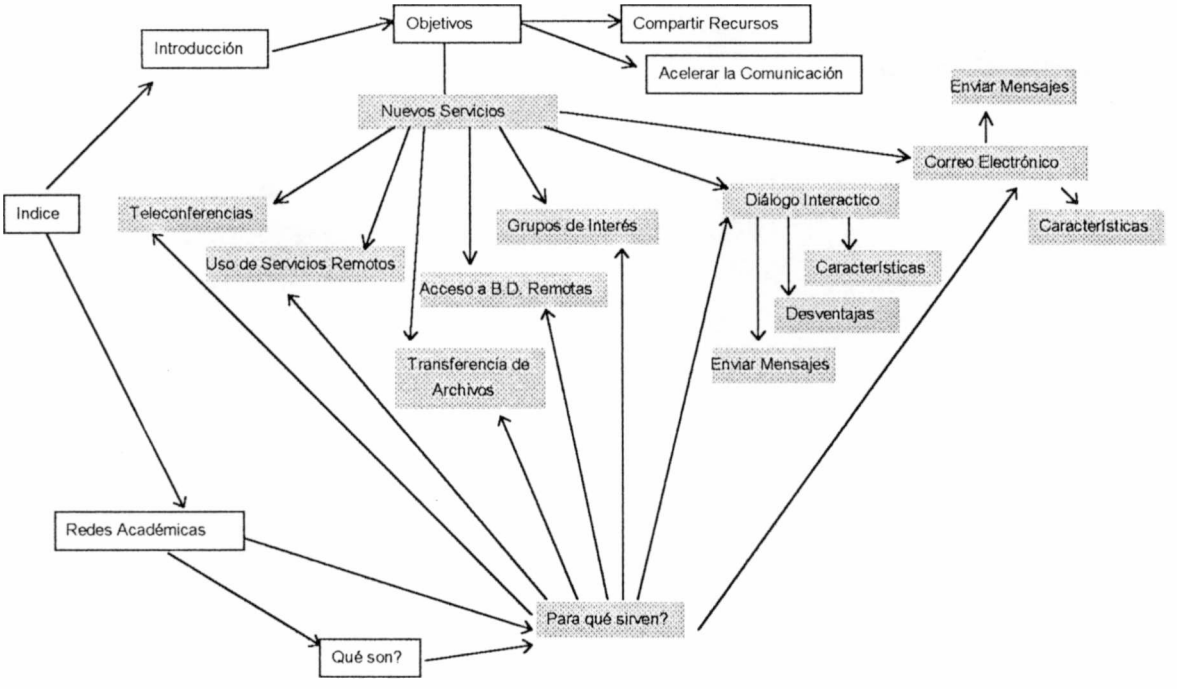


Fig. 5. Digrafo resultante del filtro por la palabra clave Servicios

- **Vistas de la hipermedia: para evitar la "desorientación del usuario"**

  - Tours Guiados

Esta herramienta permite obtener vistas reducidas del hiperespacio, a través de un criterio estadístico de selección de los nodos. Para cada nivel de usuario se construye dinámicamente un tour diferente, seleccionando los nodos más visitados y los vínculos más accedidos de dicho nivel. Una vez recuperados los nodos y enlaces que formarán el tour, estos se organizan en una estructura jerárquica (árbol), que facilita la navegación.

Esta forma de generar el tour guiado es independiente del dominio y está basada en el uso del sistema a lo largo del tiempo.

Supongamos que después de varias sesiones de trabajo, los nodos más visitados son los que se muestran en la Fig. 6. Estos nodos se organizan en una estructura de árbol como muestra la Fig. 7. En las próximas secciones explicaré detalladamente el algoritmo usado para la transformación de los subgrafos obtenidos en árboles.

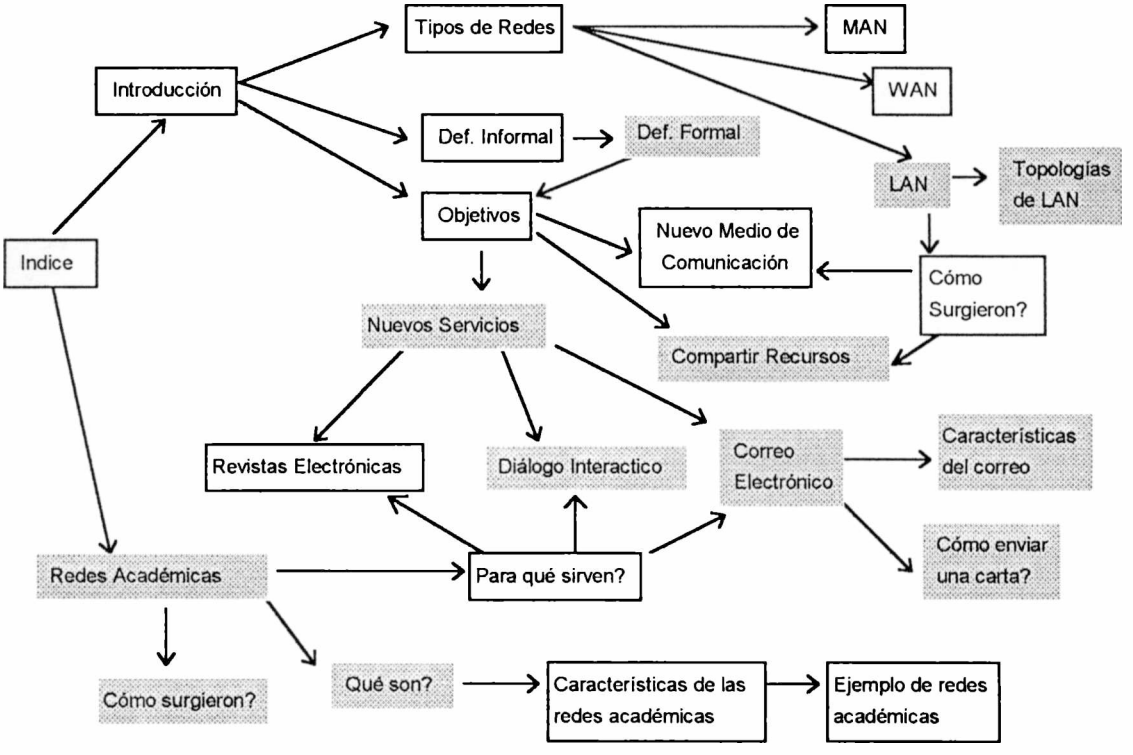


Fig. 6. Digrafo resultante de estadísticas de uso

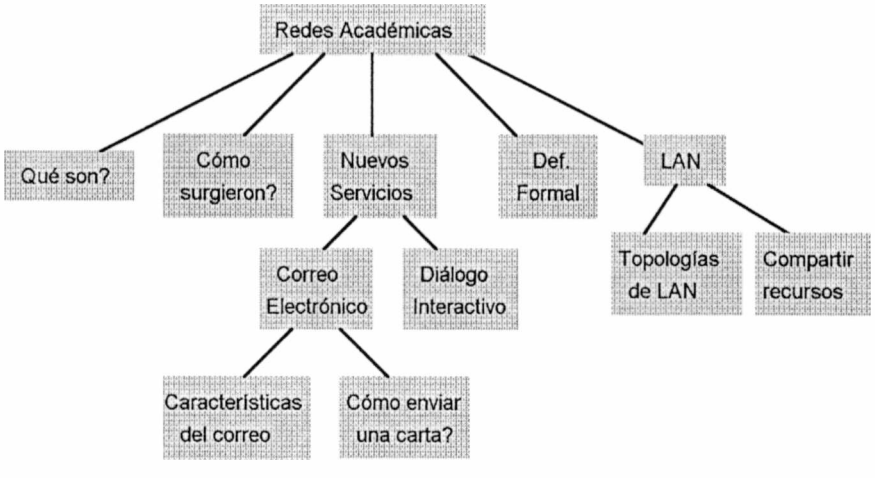


Fig. 7. Jerarquización del digrafo resultante de las estadísticas de uso

- Circuitos de lectura recomendada

Los circuitos de lectura recomendada están formados por nodos relevantes, seleccionados por el autor de acuerdo a la importancia que tienen dentro del tema tratado. Estos circuitos representan caminos en los cuales los nodos son presentados en un orden preestablecido por el autor. El lector recorre estos circuitos en forma secuencial.

Para cada nivel de usuario definido anteriormente, se construyeron circuitos de lectura recomendada conteniendo los tópicos conceptualmente importantes de cada nivel. De esta manera, un lector puede recorrer fácilmente los conceptos esenciales y así obtener una comprensión global del tema Redes de Computadoras. Por ejemplo, para el nivel coloquial, el circuito está formado por los siguientes nodos: Introducción a las redes, Definición formal de redes, Objetivos, Nuevos Servicios, Correo electrónico, Intercambio de archivos, Grupos de interés, Revistas electrónicas y Redes académicas.

- **Anotaciones personalizadas: para mantener información privada**

- Anotaciones en nodos

El objetivo de proveer anotaciones en los nodos es personalizar el ambiente de consulta de acuerdo a las necesidades y preferencias del lector. A través de ellas, un lector puede agregar sus propios comentarios en cualquier nodo del hiperdocumento, y así obtener su propia versión de los nodos.

a) *Anotaciones internas*: fundamentalmente el lector las usará para asociarle a un nodo comentarios breves, tal como referencias bibliográficas, datos estadísticos actualizados, etc. Por ejemplo, el lector puede asociarle a un nodo que trata sobre el tema Topología de anillo, una anotación que hace referencia a un libro que él leyó y donde según su criterio el tema fue tratado en forma muy interesante.

b) *Anotaciones de referencia terminal*: a través de estas anotaciones, el lector modifica el grafo del hipertexto agregándole nuevos nodos y vínculos. Las anotaciones de referencia terminal permiten agregar asociaciones personales a los temas tratados en el hiperdocumento. Por ejemplo, en un nodo que trata sobre el tema Topologías de redes locales, el lector puede crear un nuevo vínculo y asociarle una anotación que contenga una tabla comparativa de los costos y beneficios de cada una de ellas.

IMPLEMENTACION DE IMHEF: UN PROTOTIPO FUNCIONAL BASADO EN MHEF

4. ANALISIS DE LAS HERRAMIENTAS UTILIZADAS

4.1. Estudio de la herramienta: Multimedia Toolbook versión 1.53

4.1.1. Qué es Multimedia Toolbook 1.53 ?

Multimedia ToolBook versión 1.53 (de ASYMETRIX Corporation) es una herramienta de construcción de software, que opera sobre plataforma Microsoft Windows versión 3.1.

La creación de aplicaciones en Multimedia ToolBook 1.53, es una tarea sencilla, debido a que provee una potente interfaz gráfica con el usuario y un rico lenguaje de programación de scripts (OpenScript). La interfaz combina metáforas visuales (como íconos, manejo de color, menúes y ventanas) con manipulación directa de objetos. En Multimedia ToolBook también es simple "usar" las aplicaciones creadas por otras personas.

Una aplicación en Multimedia ToolBook 1.53 consiste en uno o más libros diseñados para un propósito específico, como por ejemplo capacitación, manejo de información administrativa, entretenimiento, etc. Los libros que se pueden crear, varían desde la simple presentación de datos, interfaces con otras aplicaciones (las usualmente denominadas "front-end") hasta aplicaciones autocontenidas.

La principal característica de Multimedia ToolBook 1.53 es la capacidad que tiene para combinar sencillamente información textual y gráfica. Muchas aplicaciones son potenciadas en su efectividad cuando es posible incorporarles dibujos, sonido, imágenes estáticas y en movimiento, además de texto.

4.1.2. Qué se puede hacer con Multimedia Toolbook 1.53 ?

Generar información para compartir con otros lectores: los libros construidos por un lector pueden ser leídos por otros lectores, por lo tanto, Multimedia ToolBook 1.53 es una herramienta adecuada para crear por ejemplo: manuales, folletos de divulgación científica, etc.

Desarrollo de paquetes de software para capacitación, para cursos y para presentaciones: la capacidad de Multimedia ToolBook 1.53 para construir sencillamente gráficos y animaciones, hace posible generar interfaces sofisticadas en forma simple y rápida.

Recopilación de información personal: Multimedia ToolBook 1.53 es una herramienta propicia para construir agendas personales, catálogos de productos donde se pueden incluir fotos o gráficos ilustrativos, etc..

Crear prototipos de futuras aplicaciones de software: Multimedia ToolBook 1.53 es una plataforma adecuada para el testeo y prueba de ideas a implementar en futuras aplicaciones. Por un lado, la facilidad para crear interfaces gráficas con el usuario, y por otro las caracteristicas de la programación en OpenSript, hacen que rápidamente se pueda construir un prototipo con todas las funcionalidades previstas.

4.1.3. Componentes de una aplicación en Multimedia Toolbook 1.53.

Una aplicación en Multimedia ToolBook 1.53 consiste en uno o más libros diseñados para un propósito especifico. De la misma manera que un libro impreso, un libro en Multimedia ToolBook 1.53 está dividido en *páginas*. Un libro se construye creando sus páginas, vinculándolas y asociándole scripts que son los que le definen el comportamiento al libro.

Cada libro tiene asociado una única ventana y se muestra de a una sola página a la vez. Se puede recorrer el libro, desplegando las diferentes páginas, siempre en la misma ventana.

Las páginas incluyen campos de texto, botones y gráficos, los que se denominan genéricamente *objetos*. Los objetos, se pueden disponer en las páginas de diferentes formas: dispersos, alineados o superpuestos. Los objetos tienen *propiedades* asociadas, que determinan la forma en que se mostrarán en la página, por ejemplo el tipo de borde de un botón. Las propiedades de los objetos pueden cambiarse y así permiten modificar la manera que se muestran en las páginas. Sencillamente se puede alterar el tamaño y la ubicación de los objetos, sin afectar a los restantes objetos de la página. Los objetos, también se pueden agrupar y posteriormente ser manipulados como una única entidad.

Todos los objetos en Multimedia ToolBook 1.53 pueden tener asociado scripts, que son los que definen el comportamiento del objeto. Las acciones que llevan a cabo los scripts varían desde algo muy simple como por ejemplo ir a la página siguiente del libro, o mover de lugar un objeto de una página, hasta realizar acciones complejas como crear objetos o páginas nuevas, enviar mensajes a otros objetos para que realicen cálculos numéricos o solicitar servicios a otros lenguajes de programación. Los scripts están formados por sentencias escritas en OpenScript (lenguaje de programación de Multimedia ToolBook 1.53).

En Multimedia Toolbook existen dos niveles de trabajo: de autor y de lector, y con una simple selección de comandos, es posible pasar de un nivel a otro nivel. En el nivel de lector se puede recorrer un libro, atravesando las páginas, editar los campos de texto y activar los botones presentes en cada una de las páginas. El nivel de autor provee herramientas y comandos para crear nuevos libros, crear y modificar cualquiera de los objetos de las páginas y asociarles un comportamiento a dichos objetos.

4.1.4. Lenguaje de programación de Multimedia Toolbook 1.53: OpenScript

OpenSript es un lenguaje de programación de scripts orientado a eventos. Un evento es una acción realizada por el lector o el autor sobre el mouse o el teclado, o una acción interna de Multimedia Toolbook 1.53 (por ejemplo arribar a

una página determinada). OpenScript también tiene algunas características de orientación a objetos: encapsulamiento de comportamiento (mediante la definición de scripts), mecanismo de herencia dado por una jerarquía preestablecida (fija) de objetos (no hay posibilidades de definir nuevas clases), y un mecanismo rudimentario de polimorfismo (mediante el uso de un comando nativo es posible implementar polimorfismo).

Las aplicaciones en OpenScript se construyen creando los objetos: libros, páginas, botones, campos de texto y gráficos, y escribiendo las instrucciones que definen su comportamiento. Estas instrucciones se encapsulan en scripts. Esto significa que cada uno de los objetos de una aplicación puede tener asociado un script, que es en donde se define su comportamiento.

Un script asociado a un objeto está compuesto de unidades llamadas *manejadores* ("handlers") que agrupan las acciones que se desencadenan en respuesta a ciertos eventos. Cuando ocurre un evento, por ejemplo el lector "cliquea" el botón izquierdo del mouse sobre un objeto, Multimedia Toolbook le envía un mensaje al objeto que corresponde. Cuando dicho objeto recibe el mensaje se dispara el manejador correspondiente a ese evento, que se encuentra definido en su script.

Si un objeto recibe un mensaje y su script no tiene definido el manejador que pueda responderlo, dicho mensaje es enviado al objeto superior en la jerarquía de objetos definida en Multimedia Toolbook. La *jerarquía de objetos* determina el orden en que los mensajes son transferidos desde un objeto a otro hasta encontrar el manejador correspondiente. Por ejemplo, si el script de un gráfico no incluye el manejador correspondiente a un mensaje recibido, dicho mensaje es enviado al grupo al que pertenece el gráfico (si perteneciera a algún grupo), luego a la página que incluye el gráfico y así siguiendo hacia arriba la jerarquía de objetos.

Los mensajes siempre recorren la jerarquía de objetos hacia arriba comenzando por el objeto que recibió el mensaje. El autor de una aplicación puede ubicar un manejador en el lugar que considere más adecuado en la jerarquía para evitar repetir el mismo manejador en los scripts de varios objetos.

Multimedia ToolBook 1.53 provee herramientas que permiten definir scripts sin tener que programar en OpenScript: una de ellas es la opción de "asociación" (linking) de la caja de diálogo utilizada para crear botones, que hace posible la creación de scripts de navegación, con sólo identificar la página a la que se desea ir cuando el lector "cliquea" en el botón; la otra es el grabador de scripts, que permite grabar casi todas las acciones que se pueden realizar y automáticamente se traducen a código OpenScript (es una especie de grabador de macros). Ambas opciones ofrecen una buena alternativa para aprender a programar acciones sencillas. Sin embargo, para la programación de scripts más complejos es necesario usar OpenScript. Este método permite construir scripts más flexibles y a la vez más poderosos.

4.1.5. Características de OpenScript

OpenScript permite de una manera sencilla crearle a las aplicaciones interfaces gráficas con el usuario: debido a que cada uno de los objetos del libro (libros, páginas, botones, campos de texto, gráficos) tiene asociado un script, es sencillo incluirle a los scripts instrucciones que afecten a la manera en que se

presentarán los objetos en la pantalla. Esta característica diferencia favorablemente a OpenScript de los lenguajes de programación tradicionales. Es por este motivo que Multimedia Toolbook es una herramienta adecuada para la construcción de interfaces para otras aplicaciones, usualmente denominadas "front-end".

OpenScript no tiene tipos de datos: los datos son interpretados de acuerdo a los comandos, funciones, y operadores donde se utilizan.

Esta es una muy buena cualidad para la prototipación, pues agiliza la prueba y testeo de ideas. Es mucho más rápido y sencillo programar en un lenguaje que no provee tipos de datos, que en uno que si los tiene. Sin embargo, cuando el tamaño de la aplicación crece en complejidad esta característica pasa a ser una desventaja.

OpenScript facilita la programación modular: los scripts son programas modulares, pues el script de un objeto está separado de los restantes scripts y tiene un objetivo muy bien definido: proveerle comportamiento al objeto. Los scripts están divididos en módulos llamados *manejadores* y cada uno de ellos tiene una función específica (responde a la ocurrencia de un evento determinado). La modularidad hace que la programación sea más sencilla.

OpenScript es un lenguaje extensible: por un lado, es posible programar nuevas funciones y manejadores en OpenScript, y por otro lado también se puede extender la funcionalidad del lenguaje programando procedimientos y funciones en C, Pascal o Assembler y acceder a ellos mediante Windows Dynamic Link Libraries (DLLs). Esta última alternativa potencia al lenguaje, pues permite hacer más solvente el soporte de las aplicaciones. Definiendo librerías se pueden manipular estructuras de datos más complejas, trabajar con archivos de registros, etc..

OpenScript no tiene ciclos de compilación largos: la compilación se realiza incrementalmente cada vez que se escribe o se define un script. Esto es una buena capacidad para la programación de aplicaciones de pequeña escala, pero se transforma en una desventaja cuando se trata de la programación de "grandes" aplicaciones.

OpenScript incluye herramientas de "debugging": Multimedia ToolBook 1.53 provee herramientas nativas para el seguimiento de la ejecución de los scripts, para la búsqueda y corrección de errores.

4.1.6. Capacidades de Multimedia Toolbook 1.53

Como ya fue explicado previamente, Multimedia ToolBook 1.53 se caracteriza fundamentalmente por la capacidad que provee para poder construir de una manera sencilla interfaces de usuarios gráficas, para aplicaciones finales. Las interfaces que se pueden construir, le brindan al usuario los siguientes servicios:

Multitarea, mediante la definición de múltiples ventanas. De esta manera, se pueden realizar agrupamientos particulares de información y trabajar simultáneamente con varias tareas.

Uso de metáforas visuales, mediante el empleo de iconos, colores, bordes, títulos, animaciones, incorporación de fotos, uso de fichas para incorporar datos, etc.

Uso de menúes descolgables, en donde se pueden organizar según su función las acciones a realizar por la aplicación.

Proveer ayudas y documentación en línea.

Desde el punto de vista del programador de la aplicación, Multimedia ToolBook 1.53 *facilita el acceso a otras herramientas de construcción de software* mediante el empleo de librerías dinámicas en Microsoft Windows 3.1.

4.1.7. Limitaciones de Multimedia Toolbook 1.53

Sin embargo, a pesar de la riqueza de los servicios que ofrecen las interfaces construíbles en Multimedia ToolBook 1.53, el soporte que tienen las aplicaciones respecto de la manipulación de los datos y de la programación a mediana y gran escala es muy pobre.

Construir en Multimedia ToolBook 1.53 una aplicación en donde sea necesario definir algoritmos complejos que manipulen estructuras de datos avanzadas (por ejemplo grafos), que requieran del uso de módulos subsidiarios o de la utilización de bases de datos, es una tarea impracticable, ya que OpenScript carece de primitivas y de constructores apropiados.

OpenScript es un lenguaje de programación que presenta las siguientes falencias en la implementación de aplicaciones como las descriptas:

No provee declaración ni tipado de variables.

No provee ningún mecanismo para encapsular datos (datos + comportamiento).

No provee mecanismos de acceso implícito a la información (por ejemplo la recuperación de las páginas cuyo contenido cumple una condición determinada).

No permite compilar ni poner a punto módulos de programa en forma separada (para poder guardarlos en bibliotecas de programas y reutilizarlos cuando se lo desee).

No provee ningún mecanismo para operar con la estructura de datos resultante de la organización del libro (por ejemplo, si se usa para implementar hipertextos, no provee ningún mecanismo que construya el grafo resultante de la interconexión de las páginas).

4.2. Extensión de Multimedia Toolbook 1.53 con facilidades adicionales

Como fue mencionado anteriormente, el lenguaje OpenScript provisto por Multimedia Toolbook 1.53 puede extenderse usando Windows Dynamic Link Libraries (DLLs) que son programas externos construídos en otros lenguajes

como Pascal, C y Assembler. De esta manera se pueden llevar a cabo acciones más complejas y establecer una comunicación con aplicaciones no-Windows. Los procedimientos y funciones definidos en las DLLs pueden invocarse usando instrucciones especificas de OpenScript.

Una DLL puede también describirse como una interfaz entre Multimedia Toolbook y un archivo o aplicación no-Windows.

Para invocar funciones de una DLL en un script, primeramente se debe "linkear" la DLL a Multimedia Toolbook e incluir una declaración de las funciones que serán usadas.

La posibilidad de utilizar DLLs, permite que varios libros puedan compartir programas externos y de esta manera extender la potencialidad de Multimedia Toolbook. Las librerías se compilan en forma separada de la aplicación, se incluyen una única vez y se pueden usar tantas veces como sea necesario. Además estas librerías por ser "módulos" independientes de la aplicación que interactúan solamente a nivel de parámetros, pueden ser utilizas por otras aplicaciones, extendiendo a Multimedia ToolBook 1.53 en otra dimensión: *reusabilidad de software*.

Después de muchos estudios y pruebas con varios lenguajes (librerías provistas por Multimedia Toolbook, librerías escritas en C, etc.) en la implementación de IMHEF, el prototipo funcional sobre Redes de Computadoras, se usaron DLLs construídas en Borland Pascal para Windows versión 7.0 Copyright 1992 (Borland International, Inc.). Borland Pascal 7.0 es un lenguaje de programación de propósito general, muy rico ya que permite definir tipos de datos (estructurados y no estructurados), que provee chequeo estático de tipos, mecanismos para encapsular sintácticamente datos y comportamiento y compilación separada de módulos.

Las librerías construídas permitieron, por un lado, el uso de estructuras de datos más avanzadas: listas encadenadas, árboles y grafos y, por otro lado, llevar a cabo funciones más complejas que van desde recorrer un grafo o un árbol y buscar caminos de distancia mínima entre dos nodos, hasta construir árboles a partir de las componentes conexas de un grafo.

4.3. Interfaz con otros productos

Además de la construcción de las librerías en Borland Pascal 7.0, fue necesario la utilización de archivos de datos implementados en dBase III Plus (Ashton-Tate Corporation) para almacenar la estructura del grafo de la hipermedia (nodos y aristas) y el árbol jerárquico de conceptos.

Si bien Borland Pascal 7.0 es un lenguaje de próposito general, no fue diseñado para la manipulación de grandes volúmenes de datos. El manejo de archivos provisto por Pascal hace muy compleja la definición de nuevos archivos y la actualización de los datos. Por lo tanto, para el almacenamiento de los datos se usó dBase III Plus y para su acceso un "motor" gerenciador de archivos (técnicamente denominado *engine*) del tipo Xbase (esto es, compatible con el formato de grabación utilizado por dBase) [Mari90].

Dbase III Plus es un administrador de bases de datos que provee herramientas tales como: editores de datos y archivos, indexación, consultas, etc., que resultan adecuadas para el fácil manejo de los datos y archivos.

Esta combinación de productos (dBase III Plus + Pascal) ofrece un esquema de trabajo sólido y eficiente, ya que provee, por un lado, la flexibilidad y facilidad

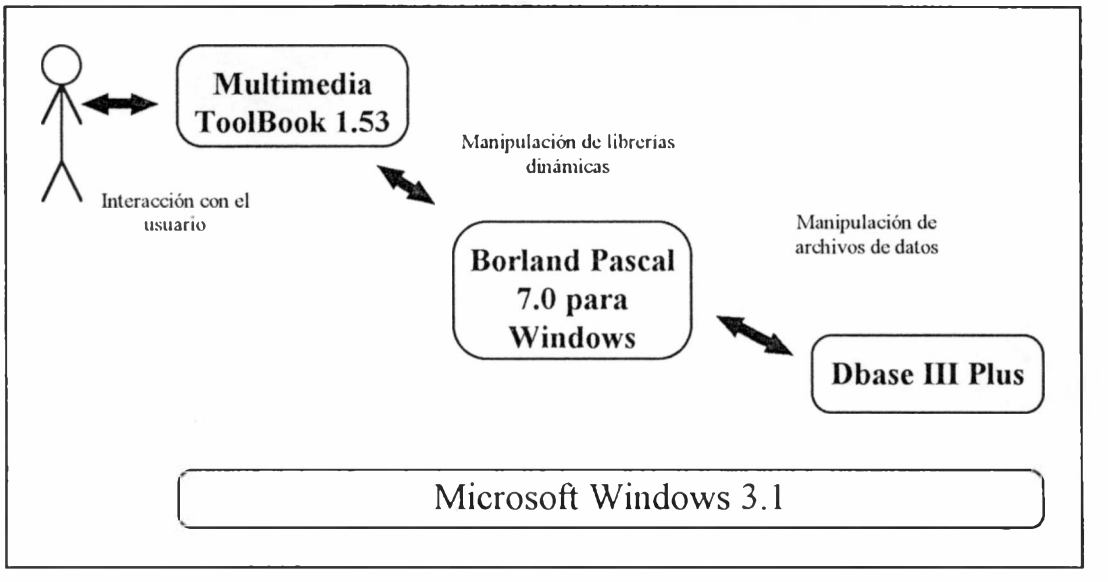
en la actualización y almacenamiento de los datos aportada por dBase III Plus y, por otro lado, la eficiencia en el acceso y manipulación de estructuras dinámicas provista por Pascal.

En la próxima sección describiré en detalle la arquitectura del prototipo IMHEF, mostrando cómo se integraron estas herramientas.

5. ARQUITECTURA DEL PROTOTIPO IMHEF

5.1. Descripción de la arquitectura básica

Para mostrar la funcionalidad del modelo MHEF descripto en este informe, se construyó IMHEF (Instanciación de **MHEF**), un prototipo basado en el dominio de Redes de computadoras. El prototipo desarrollado está implementado combinando diferentes herramientas de construcción de software: Multimedia ToolBook 1.53, Borland Pascal 7.0 y Dbase III Plus, bajo plataforma Microsoft Windows versión 3.1. Fue desarrollado sobre un computador personal con microprocesador 80486, 16 Mb de RAM y monitor SVGA. En el siguiente esquema, que muestra la arquitectura básica del sistema, se puede observar cómo interactúan cada una de las herramientas usadas.



Multimedia Toolbook versión 1.53 fue utilizado en la construcción de la interfaz del usuario, que representa el nivel de presentación del modelo HAM [Camp88] y en la implementación completa del mecanismo de navegación de hipermedia.

Dado que Multimedia Toolbook 1.53 permite trabajar con más de una instancia o "libro" simultáneamente, esta herramienta se utilizó no sólo para la presentación de la hipermedia sino también para la implementación de las anotaciones internas y de referencia terminal. Además, se utilizó para mostrar el grafo de la hipermedia usado en las facilidades de filtro, tour guiado y circuito recomendado y permitir un acceso directo desde los nodos del grafo.

Sin embargo, para aquellas facilidades tales como filtro por palabra clave y tour guiado, que manipulan el grafo de la hipermedia, fue necesario utilizar librerías dinámicas implementadas en Borland Pascal 7.0. Como fue explicado anteriormente, estas librerías aumentan la potencialidad del lenguaje nativo de Multimedia Toolbook, OpenScript, permitiendo el uso de estructuras de datos más complejas como listas encadenadas, árboles y grafos. Mediante el empleo de estas librerías se superan algunas de las falencias que presenta OpenScript, como por ejemplo el encapsulamiento de datos, la puesta a punto y la

compilación separada de módulos. En estas librerías es posible implementar abstracción a nivel de procedimientos y funciones, y también de datos.

5.2. Descripción de las librerías construídas

Para la implementación del filtro por palabra clave, el tour guiado y el circuito de lectura recomendada fue necesario la implementación de librerías dinámicas ya que estas funciones, por un lado, realizan consultas sobre la estructura del grafo de la hipermedia, el árbol jerárquico de conceptos y sobre los atributos de los nodos, y por otro lado, construyen estructuras de datos adicionales (grafos derivados, árboles, etc.).

Las funciones definidas en las librerías permiten recorrer el grafo de la hipermedia y el árbol de conceptos, obtener componentes conexas, buscar caminos de distancia mínima entre dos nodos, construir el árbol minimal de un grafo, etc. y además efectuar consultas sobre los atributos de los nodos (palabras claves y frecuencia de visitas).

5.2.1. Librería Grafo

La librería GRAFO permite implementar y manejar a un grafo como una abstracción de datos: solamente en el interior de la librería se "conoce" y se puede manipular la estructura interna; sin embargo, la única interacción que se tiene con el exterior es mediante la invocación a los procedimientos y funciones que implementan su comportamiento.

Mediante el uso de esta librería se implementa el grafo asociado a la hipermedia, permitiendo realizar inferencias acerca de la estructura de conexiones entre los nodos del hipertexto. De acuerdo a lo descripto previamente, Multimedia ToolBook 1.53 facilita la presentación o visualización del hipertexto, sin embargo no provee ningún mecanismo propio para generar y manipular la estructura del hipertexto (grafo).

Se definió al grafo como una abstracción de datos, que tiene dos componentes fundamentales: *vértices* y *aristas*. En los vértices se almacenan los datos (nombre del vértice, palabras claves por las que está calificado el vértice y su registro de visitas), y en las aristas se representa la relación o conexión existente entre ellos (nombre del botón que conecta los nodos y una etiqueta identificatoria de la arista).

Esta librería provee funciones que permiten consultar y actualizar tanto los datos como las relaciones entre los vértices y construir nuevas estructuras de datos a partir del grafo de la hipermedia.

5.2.1. Librería Arbol

En el mismo sentido que la librería Grafo, la librería ARBOL permite abstraer la estructura interna y el comportamiento de un árbol general.

Los datos se almacenan en los nodos y se establecen relaciones de tipo jerárquicas entre ellos mediante las aristas. Las aristas además están etiquetadas.

Esta librería fue utilizada para la implementación de la jerarquía de conceptos o palabras claves usada en el método de filtrado por palabra clave. Además, fue usada en la implementación de la estructura de datos asociada al tour guiado, permitiendo controlar el mecanismo de navegación por los nodos que pertenecen a dicha jerarquía.

Las funciones que se definieron permiten consultar los datos almacenados en los nodos, recuperar la información acerca de los descendientes de un nodo, agregar descendientes, etc..

5.2.1. Librería Bosque

La librería BOSQUE permite la manipulación de un conjunto de árboles en una única estructura, aportando un mayor grado de abstracción. Dado que cada componente del conjunto es un árbol, se usan las funciones de la librería Arbol definida previamente.

Esta librería es subsidiaria de la librería Arbol y fue utilizada en el procesamiento de los árboles intermedios que surgen durante la implementación de la función tour guiado.

Las funciones incorporadas en esta librería permiten la recuperación, el agregado y la eliminación, de cada uno de los árboles que integran el bosque.

5.2.1. Librería Lista de enteros

La librería LISTA DE ENTEROS fue construída para optimizar el procesamiento de datos enteros durante la ejecución de las facilidades provistas.

La manipulación de los nodos del grafo se realiza mediante un mecanismo de indexación de los mismos. Se utilizan listas de enteros para el soporte de los índices y, de esta manera, el recorrido y la búsqueda dentro del grafo resultan más eficientes.

Esta librería implementa un mecanismo de búsqueda dicotómico, por lo cual las listas de enteros resultan útiles para el almacenamiento de cálculos auxiliares, por ejemplo cuando se debe mantener ordenado un conjunto de distancias mínimas entre nodos del grafo.

Las funciones provistas por esta librería permiten la búsqueda, el agregado y la eliminación de datos enteros a una lista.

5.3. Descripción de las bases de datos

Como fue explicado anteriormente, se utilizaron archivos de datos dBase III Plus para almacenar el grafo de la hipermedia y el árbol jerárquico de conceptos.

La estructura del grafo es almacenada en dos bases de datos: una denominada *Vértices* que contiene toda la información relativa a los nodos o vértices del grafo y otra llamada *Aristas* que contiene información de las aristas o vínculos existentes entre los nodos.

En cada registro de la base de datos Vértices se almacena el nombre del nodo, su descripción, las palabras claves por las que está clasificado y su frecuencia de visitas. Cada registro del archivo Aristas refleja el nodo origen y el

destino de la arista, el botón que conecta el origen con el destino y el nombre del enlace.

La información del árbol de conceptos se almacena en una base de datos denominada *Concepto*, que refleja la estructura jerárquica usando la representación de hijo izquierdo - hermano derecho. Cada registro de este archivo contiene el nombre del concepto y los índices donde se encuentran su hijo izquierdo y su hermano derecho.

6. DISEÑO DE LA INTERFAZ DEL USUARIO

6.1. Descripción de la interfaz utilizada

Al construir un sistema de hipermedia se debe poner especial interés en el diseño de la interfaz a utilizar. Es necesario elegir, por un lado, la forma más adecuada de presentar la información contenida en los nodos, y por otro lado, la manera de representar claramente las asociaciones o enlaces existentes entre los nodos. Es fundamental que el origen de los enlaces estén representados en forma clara y precisa para que el lector reconozca instantáneamente el "lugar" de la pantalla que le permitirá "explotar" un vínculo y de esta manera navegar a través de la hipermedia. El diseño del mecanismo de enlaces debe permitir que la navegación resulte simple, intuitiva y consistente a través de toda la hipermedia [Shne92].

En el prototipo IMHEF implementado fue necesario, además, analizar cuidadosamente la forma de representar las funciones adicionales provistas por el modelo: filtro por palabra clave, tour guiado, circuito de lectura recomendada y anotaciones. Estas funciones son independientes del contenido de los nodos y por lo tanto deben presentarse en forma separada de éste.

Al diseñar la interfaz de IMHEF, se combinaron elementos de **interfaces gráficas o visuales** con elementos de **interfaces icónicas** [Hara93].

En las interfaces visuales se emplean gráficos, imágenes y secuencias de animaciones para ilustrar o presentar los datos y entidades que intervienen en el sistema, así también como las entradas y los resultados (o salidas) a mostrar. Todos los objetos intervinientes en el sistema tienen asociada una representación visual.

Las interfaces icónicas son una clase más específica de interfaces visuales, en donde se utiliza como medio de representación visual exclusivamente al ícono. Un icono es una imagen, un gráfico o un símbolo que representa un concepto [Gitti86] [Roge89]. Se usa la representación icónica para modelar todas las componentes del sistema y sus funcionalidades, los datos de entrada, de salida, estados y entidades.

En las interfaces visuales el usuario se "comunica" o interactúa con el sistema mediante la utilización de un diálogo visual. El usuario final le muestra al sistema qué hacer mediante el desplazamiento y la manipulación de las representaciones visuales de los objetos que componen la interfaz. Esta clase de interfaz se caracteriza por la manipulación directa de los objetos y el mouse es un dispositivo físico apropiado para esto. Poder trabajar con representaciones de objetos reales resulta más "natural" y está ligado a la capacidad innata del ser humano (en las etapas evolutivas del hombre, el entrenamiento visual aparece antes que el lenguaje) [Shne92].

Las ideas o conceptos contenidos en los nodos de la hipermedia se expresaron mediante la combinación de información textual, gráfica, animaciones simples e imágenes. Cada pantalla fue diseñada de manera que el contenido resulte fácilmente comprensible para el lector. En la definición de la mayoría de los conceptos aparece asociado un gráfico o una animación explicativa. Los usuarios pueden reconocer y recordar las imágenes rápidamente, y detectan

fácilmente cambios en el tamaño, color, forma y textura de los objetos representados [Shne92]. Además, la representación gráfica de objetos reales se acerca al mundo tridimensional en el que vivimos, y por lo tanto permite explicar los conceptos en una forma más clara y familiar.

Como nodo inicial o raíz del hiperdocumento se presenta una especie de "tabla de contenidos", que muestra los conceptos principales desarrollados. Tener una estructura global de la información contenida en el hiperdocumento, le permite a los lectores construir su propio mapa mental de los tópicos tratados. Esto facilita la tarea del lector al iniciar la navegación, ya que resulta importante que el lector sepa qué conceptos no están incluídos en el hiperdocumento. Los lectores pueden sentirse "terriblemente" frustrados si piensan que algún concepto de interés está en la base de información a la que están accediendo y no pueden llegar a él [Shne92].

Para la representación de las asociaciones o enlaces existentes entre los nodos de la hipermedia, se utilizaron botones o íconos de enlace. Se puso especial énfasis en el diseño del tipo de presentación elegida para los botones, ya que estos deben aparecer como guías fácilmente reconocibles para el lector. Se analizó cuidadosamente la frase de texto o gráfico asociado a cada botón, para que realmente sintetice la relación entre los nodos que vincula. Los botones representan el origen de los vínculos o relaciones establecidas por el autor, por lo tanto es fundamental que el lector los reconozca instantáneamente (para facilitar esta tarea, al "pasar" con el mouse por cada botón se cambia el formato del cursor). La navegación a través de la red de nodos debe requerir el mínimo esfuerzo por parte del lector.

Por otro lado, se eligió la representación icónica para expresar las acciones que llevarán a cabo las funciones que provee el sistema, como el glosario de palabras y la ayuda, las facilidades provistas para la recuperación de información sobre el hiperdocumento (filtros, tours guiados, circuitos de lectura recomendada y anotaciones), y las clásicas de recorrido (retroceder de nodo o "backtrack", nodo anterior y siguiente e ir al nodo índice). Los íconos que representan estas acciones integran un "panel de funciones" ubicado en la parte inferior de todos los nodos de la hipermedia.

Cabe destacar que en el diseño de cada una de las pantallas, se mantuvo la consistencia en lo que respecta a la estructura, ubicación de los objetos y terminología utilizada. Los colores fueron usados, no sólo en forma decorativa, sino para cumplir funciones específicas, tales como resaltar determinados nodos del grafo y agrupar conceptos relacionados.

6.2. Descripción del ambiente multi-ventana

Esta técnica ofrece un poder significativo para la organización de la información y el control de la presentación (con títulos, bordes y colores de fondo que denotan agrupamientos particulares). Entre sus principales ventajas se encuentran: la optimización del uso del estado real de la pantalla; la posibilidad de definir distintos contextos y la facilidad de movimiento entre ellos, para poder trabajar simultáneamente en diferentes tareas; la posibilidad de contar con

múltiples fuentes de información al mismo tiempo; y, por último, la posibilidad de que, a través de múltiples ventanas, el sistema pueda entablar diálogos efectivos con los usuarios debido a la capacidad de síntesis y mantenimiento del contexto que la caracteriza.

Por otro lado, el diseño de un ambiente multi- ventana permite coordinar las ventanas según tareas realizadas por el usuario. Esto significa que las ventanas pueden abrirse, cambiar el contenido o cerrarse como resultado directo de una tarea llevada a cabo por el usuario en la ventana principal [Norm86] [Shne86].

En el prototipo IMHEF desarrollado sobre "Redes de Computadoras" se definieron las siguientes ventanas de trabajo:

- *la ventana de la hipermedia*, en donde se presenta la información y los enlaces (empleando botones) de cada uno de los nodos de la hipermedia, utilizando representaciones visuales (descripto anteriormente). Además, en la parte inferior de esta ventana aparece un panel único de funciones, propias del sistema y simbolizadas por íconos. En este panel están representadas las funciones provistas por el modelo MHEF: filtros, tour guiado, circuito de lectura recomendada y anotaciones; las herramientas clásicas de recorrido: ir al nodo siguiente y anterior, ir al nodo visitado previamente (backtrack) e ir al nodo índice; y además, aparecen dos íconos adicionales: la ayuda y un glosario de palabras. En la parte inferior derecha de la ventana se puede visualizar el ícono de salida. Este diseño de la pantalla se mantiene similar a lo largo de todo el sistema (Fig. 8).

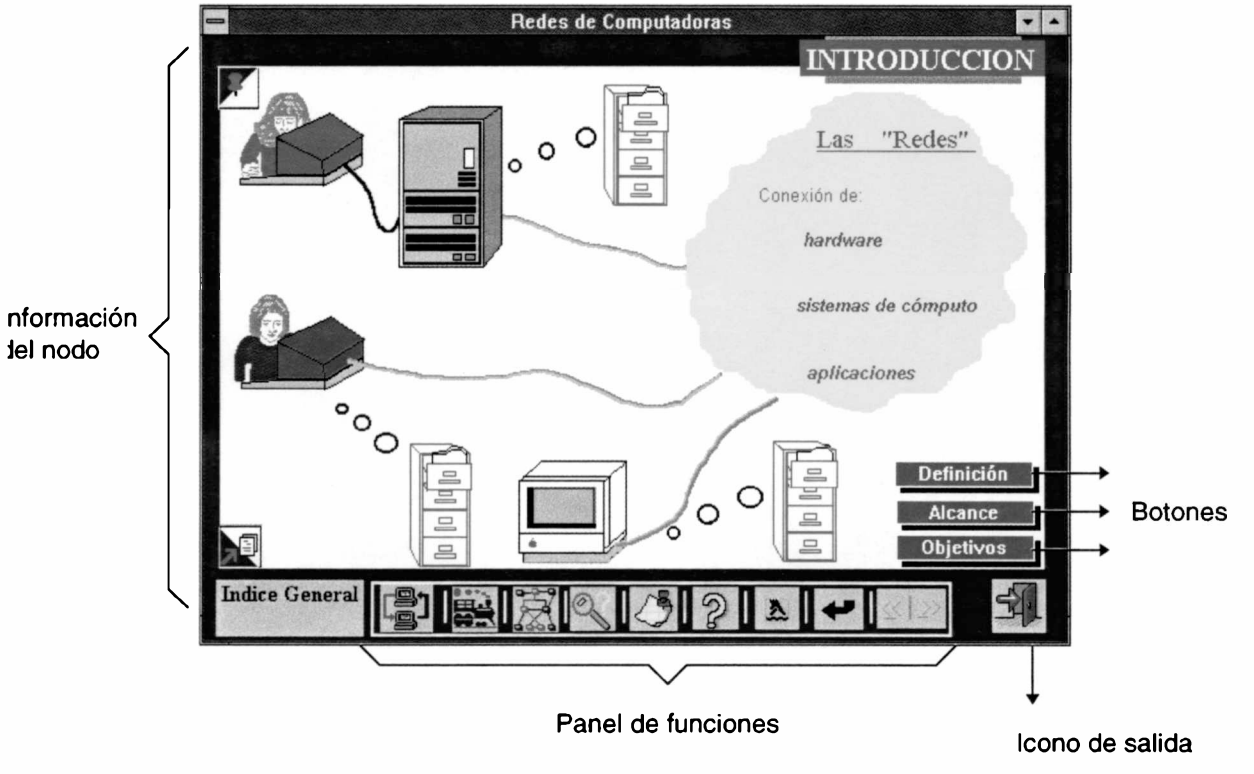


Fig. 8. Ventana de la hipermedia

La ventana de la hipermedia constituye la tarea principal del sistema, en el sentido de que es la única ventana que se abre (en la parte central de la pantalla) al comenzar una nueva sesión de trabajo, y permanece abierta durante toda la sesión.

- *la ventana del grafo*, que se utiliza para presentar gráficamente la estructuración de la red de nodos y enlaces de la hipermedia. Los nodos están representados a través de rectángulos etiquetados y los enlaces, a través de líneas que conectan los nodos. Los enlaces se muestran en distinto color según se trate de enlaces existentes (creados por el autor) o enlaces nuevos (que aparecen al construir el tour guiado o el circuito recomendado). Esta ventana está asociada a las siguientes funciones: filtro por palabra clave, tours guiados y circuitos de lectura recomendada.

La ventana del grafo se abre en la parte superior derecha de la pantalla cuando el lector elige alguna de las funciones nombradas anteriormente. De esta manera, la ventana de la hipermedia se desplaza hacia abajo y a la izquierda de la pantalla para que el área de superposición entre las dos ventanas disminuya y el contenido de la ventana principal no sea cubierto por la nueva ventana (Fig. 9).

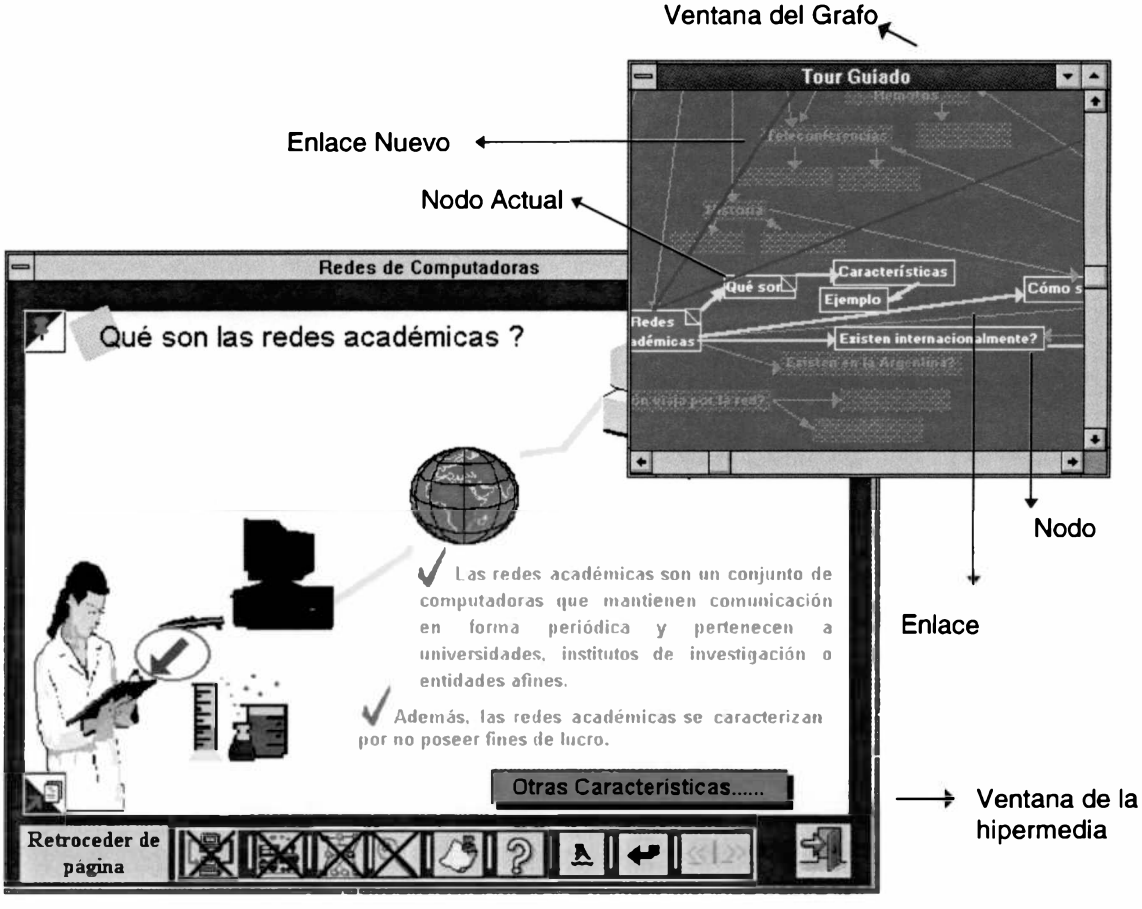


Fig. 9. Ventana de la hipermedia y del grafo (correspondiente al Tour Guiado)

- *las ventanas de anotaciones internas y de referencia terminal*, que se utilizan, para crear, editar y mostrar las anotaciones internas y externas de un nodo. La ventana de las anotaciones internas se abre en la parte inferior derecha de la pantalla, permaneciendo abierta la ventana de la hipermedia, y permite crear y editar las anotaciones en una especie de agenda. Esta ventana presenta un campo de texto y botones que permiten crear una nueva anotación, ir a una anotación determinada y finalizar con las anotaciones (Fig. 10).

Por otro lado, la ventana de anotaciones externas o de referencia terminal "reemplaza" a la ventana principal de la hipermedia. Dado que la anotación externa es considerada un nuevo nodo del grafo, cuando el lector está editando o simplemente viendo una anotación de este tipo, sólo estará abierta esta ventana, en la misma ubicación y con el mismo tamaño de la ventana principal. La ventana de anotaciones externas presenta un campo de texto donde el lector puede escribir el contenido del nodo que está creando. Además, presenta en la parte inferior un panel de funciones para edición de texto: fijar el tamaño de la letra y el estilo del texto.

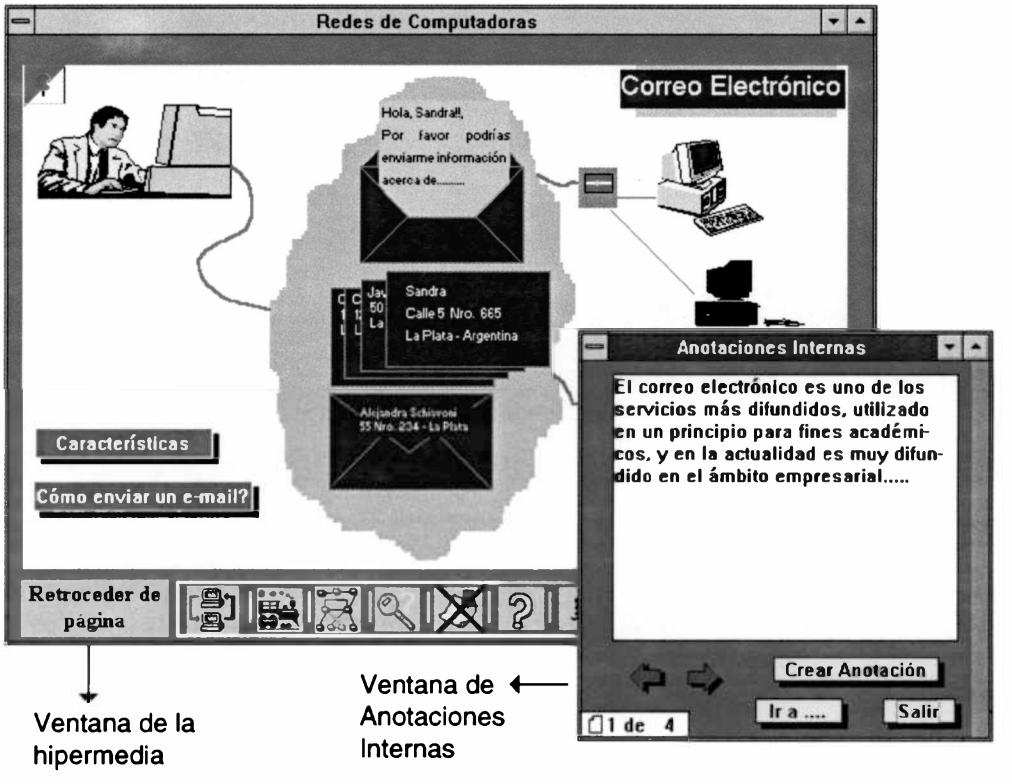


Fig. 10. Ventana de la hipermedia y de anotaciones internas

Tanto la ventana del grafo como las de anotaciones internas y externas, se abrirán en el momento que el lector seleccione el ícono correspondiente a la función que desea aplicar. De esta manera, el lector tiene la posibilidad de interactuar con la tarea principal, por ejemplo navegando a través de los nodos del hipertexto y, simultáneamente interactuar con las ventanas que implementan las facilidades provistas, por ejemplo realizando anotaciones sobre un nodo o utilizar los mecanismos de acceso directo a los nodos que provee el filtro por palabra clave.

6.2.1. Caja de diálogo

El aspecto de un caja de diálogo es muy parecido al de un formulario de oficina. En las cajas de diálogo, el usuario puede especificar opciones y acciones usando íconos y controles, entrar texto en campos de entrada, ver mensajes y contar con una asistencia permanente a través de ejemplos.

En el prototipo IMHEF, se utilizó esta técnica de interacción, para implementar el ingreso por parte del lector, de la palabra clave, por la que desea realizar el filtrado del grafo de la hipermedia. A continuación se muestra el empleo de cajas de diálogo, para ingresar palabras claves (Fig. 11):

Fig. 11. Caja de diálogo usada para ingresar palabras claves

Cómo funciona la caja de diálogo para ingresar palabras claves?: en el momento que el lector selecciona el ícono correspondiente al filtro por palabra clave, el sistema presentará una caja de diálogo organizada de la siguiente manera: 1) en la parte superior izquierda un menú deslizable donde se presentan todas las palabras claves disponibles ordenadas de acuerdo a su inicial; 2) en la parte superior derecha, un panel fijo donde se presentan todas las letras del alfabeto (asistirá al usuario en la localización de las palabras claves) y; 3) en la parte inferior, íconos que simbolizan las funciones de salida de la caja de diálogo. A partir de esta caja de diálogo, el lector elige una de las palabras claves disponibles, utilizando exclusivamente el menú donde se listan todas las palabras, o la combinación del panel de localización de palabras claves a partir de su inicial junto con el menú. El lector también puede anular la operación de selección de palabras claves, seleccionando el ícono de cancelación de la caja de diálogo.

6.3. Paradigma de interacción: técnicas usadas

6.3.1. Dependencias entre objetos

Una dependencia especifica una relación causal que debe ser siempre mantenida. Es una herramienta muy poderosa que permite que el sistema, ante la manipulación de un objeto determinado ajuste todas las partes, como así también, los objetos que dependen de él.

En el prototipo funcional IMHEF que se está describiendo, se utilizó la técnica de dependencias en los siguientes puntos:

1) Según lo explicado previamente, en la ventana de la hipermedia se presenta la información contenida en los nodos (datos y enlaces), y en la ventana del grafo se presenta, mediante el empleo de una metáfora visual, la estructuración de los nodos y enlaces del hipertexto, usando rectángulos etiquetados y líneas, respectivamente. Dependiendo de la función que tenga asociada la ventana del grafo (tour guiado, filtro por palabra clave o circuito de lectura recomendada), se mostrarán coloreados adecuadamente los rectángulos (nodos) y las líneas (enlaces) asociados a los subgrafos resultantes de la aplicación de la función, distinguiéndose así de los que no fueron seleccionados. Cuando el lector elige ("cliquea") un rectángulo coloreado (nodo) de esta ventana, automáticamente se mostrará el contenido de dicho nodo en la ventana de la hipermedia, produciéndose un cambio en el contenido de la misma. El nodo elegido en la ventana del grafo queda marcado como visitado.

Por otro lado, cuando el lector arriba a un nodo explotando un botón en la ventana de la hipermedia, automáticamente en la ventana del grafo se realiza un corrimiento ("scrolling") para que el nodo accedido aparezca visible dentro de la ventana y además, al nodo del grafo se le agrega una marca de visitado.

Por lo tanto, la ventana del grafo de la hipermedia y el contenido de la ventana principal (contenido del nodo actual) están en relación de dependencia (Fig. 12).

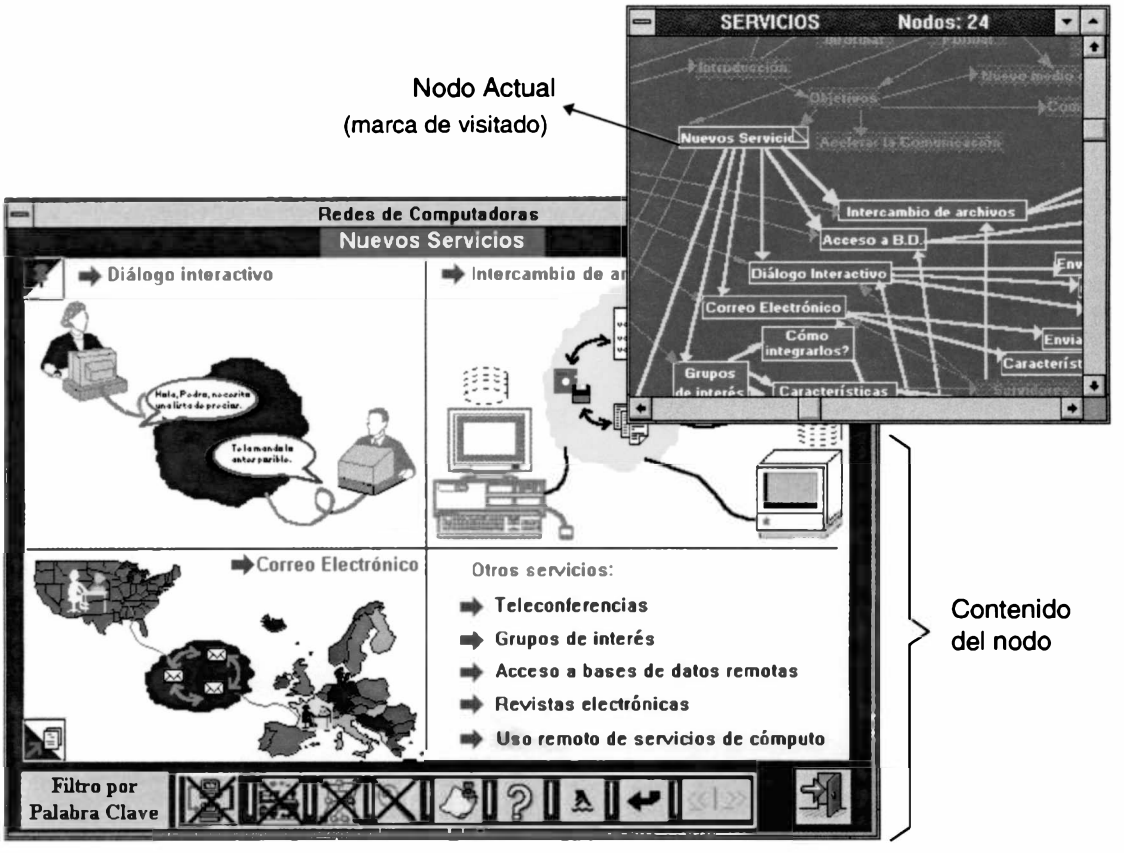


Fig. 12. Dependencia entre el grafo de la hipermedia y el contenido del nodo actual

2) En la ventana de la hipermedia, a la izquierda del panel de funciones aparece un campo de texto que, dependiendo de la ubicación del cursor respecto de los íconos de dicho panel, presenta una leyenda informando qué función tiene asociada el ícono sobre el que está posicionado el cursor. Esta técnica de explicar brevemente el significado del ícono tiene como propósito ayudar al lector en la identificación de las funciones que representa cada ícono. El uso integrado de íconos y texto resulta más efectivo en la mayoría de los sistemas [Norm91]. Por ejemplo, cuando el cursor "pasa" por el ícono que simboliza la función filtro por palabra clave, el campo de texto se actualizará con la leyenda "Filtro por Palabra Clave". Por lo tanto, existe una relación de dependencia entre el campo de texto en donde se presenta la ayuda y los íconos del panel de funciones (Fig. 13).

3) Cuando el lector crea una anotación (interna o de referencia terminal) en un nodo, aparece una marca representada por un ícono nuevo que simboliza la presencia de una anotación. Este ícono representa la esquina de la hoja doblada, con un "clip" para el caso de anotaciones internas o una flecha que indica un enlace para las anotaciones externas. La incorporación de este ícono al contenido

del nodo permite que el lector reconozca fácilmente la existencia de una anotación. En el momento que el lector "pasa" por este ícono aparece un campo de texto, que describe su significado. Este campo de texto muestra una leyenda que dice "Hay una anotación interna" o "Hay una anotación externa", según corresponda. Aquí se tiene otra dependencia, entre el ícono que simboliza la marca de anotación y el campo de texto que describe su función (Fig. 14).

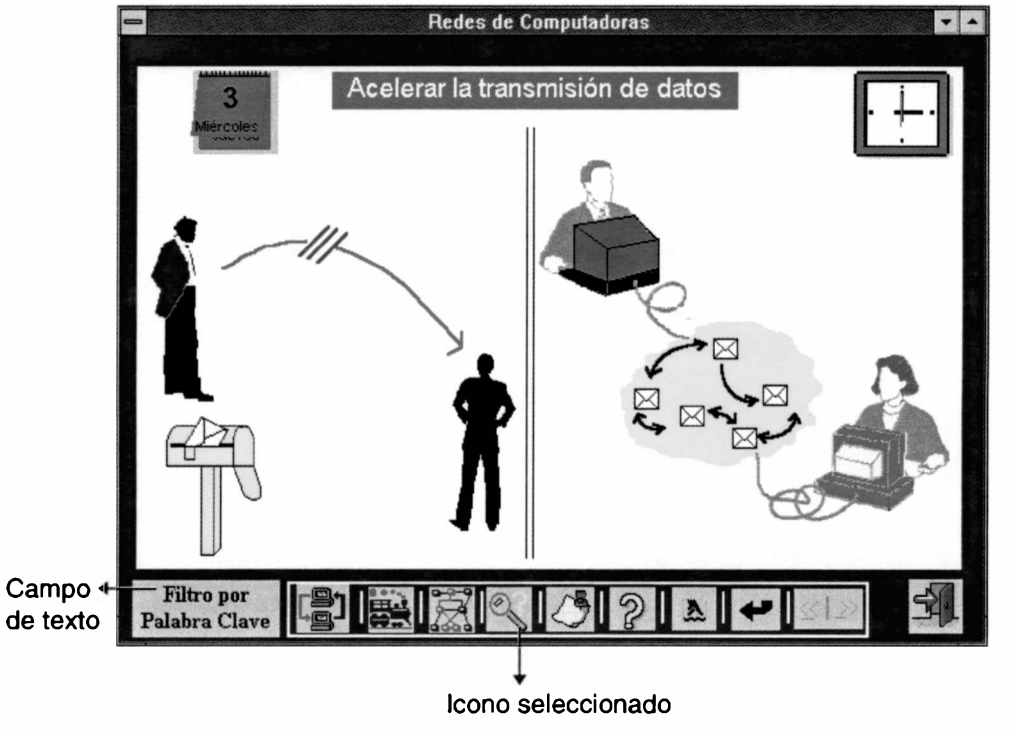


Fig. 13. Dependencia entre los íconos del panel de funciones y el campo de texto

6.3.2. Retorno o "feedback"

La técnica denominada retorno o "feedback" le permite al sistema mantener informado al usuario. Al proveer feedback la interfaz da una respuesta gráfica o textual en la pantalla frente a una acción del usuario, indicando el estado en que se encuentra [Hara93].

El feedback es el proceso de reflejar sobre la pantalla el resultado de alguna operación realizada por el usuario. Existen dos tipos de retornos: uno "superficial", en donde únicamente se suministra información propia de la entrada/salida (por ejemplo, confirma que un dato fue aceptado); y otro "semántico" en donde se extrae información de la aplicación para lograr una respuesta más apropiada, más completa relacionada al contexto (por ejemplo, visualiza el estado de la aplicación).

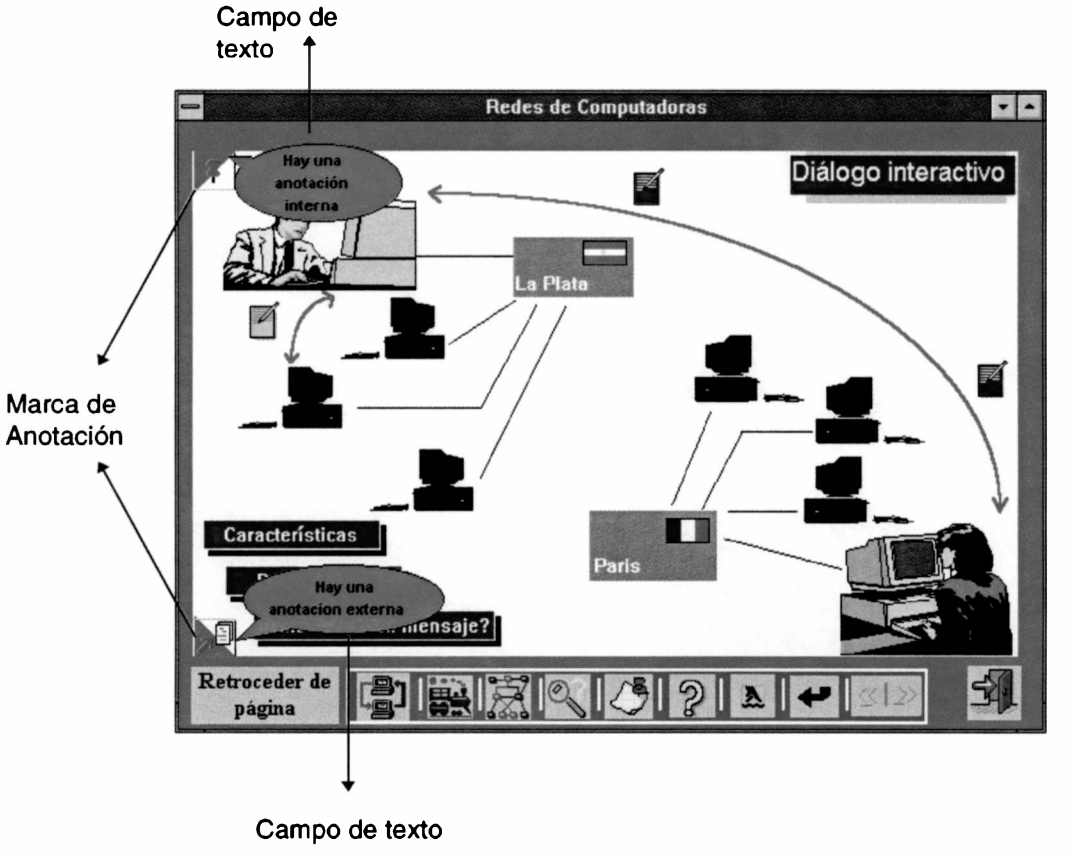


Fig. 14. Dependencia entre la marca de anotación y el campo de texto

En el diseño del prototipo hipermedial IMHEF que se está describiendo, se utilizó esta técnica de interacción, para reflejar en la pantalla: 1) *la inhabilitación de botones de navegación:* cambiando la forma asociada al cursor del mouse cuando el lector "pasa" sobre él, indicando que dicho enlace no se puede "explotar". Esta situación se presenta cuando el lector está navegando por el subgrafo generado a partir del filtro por palabra clave, por el árbol construído en el tour guiado o por el circuito recomendado; 2) *la inhabilitación de funciones adicionales:* superponiéndole una tachadura al ícono que simboliza la función. Esta tachadura indica que si el lector está usando una función específica, tiene inhabilitadas otras funciones, pues en la mayoría de los casos la aplicación de funciones no se puede componer. Por ejemplo, si se está utilizando el tour guiado, no se puede usar filtro por palabra clave, tours guiados ni circuitos de lectura recomendada (Fig. 9); 3) *la inhabilitación de las acciones de recorrido, nodo anterior y siguiente:* sombreando al ícono que simboliza la acción. De la misma manera que en el punto anterior, el sombreado simboliza que el lector tiene "prohibida" la ejecución de dicha acción. Por ejemplo, cuando el usuario está "leyendo" un nodo determinado y ese nodo no tiene conexiones del tipo secuencial, entonces los íconos que llevan a cabo las acciones de ir al nodo siguiente y al anterior están sombreados; 4) *la habilitación de nodos y enlaces, desde la ventana del grafo:* dependiendo de la función que el lector haya aplicado sobre la hipermedia (filtro por palabra clave, tour guiado, circuito de lectura recomendada), los símbolos gráficos que representan los nodos y vínculos del grafo de la hipermedia (rectángulos y líneas), se colorean

apropiadamente, indicando que pertenecen al subgrafo generado, después de haber aplicado la función (Fig. 12).

6.4. Características generales de la interfaz

Los objetivos del diseño de la interfaz del prototipo IMHEF fueron, por un lado, asegurar la uniformidad y coherencia en el tipo de interacción provista por el sistema, y por otro lado, garantizar la completitud desde el punto de vista funcional, pues se abarcan todas las funciones que se pueden aplicar sobre la hipermedia.

- ***Flexibilidad***: le permite al lector de la hipermedia elegir, seleccionar opciones, investigar a partir de las asociaciones establecidas entre los nodos, sin seguir una secuencia de lectura (o de acciones) prefijada. De esta manera, se sitúan los dos agentes que interactúan, el hombre y la computadora, a un mismo nivel (evitándose los diálogos dominantes por la computadora). Además, la utilización de metáforas visuales para representar la información, combinada con la representación icónica para simbolizar las funciones que se pueden aplicar sobre la hipermedia, permiten disminuir el esfuerzo mental requerido por el lector para trasladar sus pensamientos o tareas en mente, a las acciones del sistema.

- ***Confiabilidad***: permite expresar los términos y conceptos en forma segura, sin ambigüedades ni falsas interpretaciones. Se puso especial dedicación en el diseño de los íconos que expresan las funciones aplicables sobre la hipermedia, así también como en el diseño de los botones. Se utilizaron, siempre que fue posible, representaciones visuales para los conceptos a impartir (gráficos y animaciones simples), teniendo en cuenta que las imágenes condensan ideas (se dice que un gráfico dice más que 1000 palabras...). La utilización de mucho texto en la pantalla le provoca al lector pérdida del interés e indiferencia, y el uso de poco texto puede traer aparejado ambigüedades.

- ***Transparencia***: este punto está relacionado a la confiabilidad de la interfaz, pero tiene características propias, que hacen referencia a la forma de expresar y visualizar los estados del sistema y sus transiciones. Todos los estados del sistema son transparentes para el lector. Esta característica esta muy relacionada con el concepto de retorno, ya que se le muestra al lector, no sólo las entradas y salidas sino también los estados temporales de una acción.

6.5. Aspectos de usabilidad

En el diseño de la interfaz del prototipo desarrollado se puso especial énfasis en los aspectos de *usabilidad* del mismo. Cabe destacar que la usabilidad de una aplicación hipermedial está determinada por la combinación entre la usabilidad del sistema de hipermedia subyacente, la usabilidad del contenido y estructura de la base de información hipermedial y la forma en la que estos dos componentes pueden acoplarse adecuadamente [Niel90b].

Según Nielsen [Niel90b], la usabilidad está tradicionalmente asociada a cinco parámetros:

- Facilidad de aprendizaje: la interfaz fue diseñada de manera que resulte comprensible y fácil de aprender para todos los lectores. Los botones que representan el origen de los enlaces pueden ser reconocidos instantáneamente, facilitando la tarea de navegación. Por otro lado, en el panel de funciones aparecen íconos con un texto explicativo, que simbolizan cada una de las facilidades adicionales provistas por el sistema, tales como filtros, tour guiado y circuitos recomendados. Estas facilidades han sido implementadas de manera que el lector pueda ejecutarlas simplemente "clickeando" sobre el ícono correspondiente (sin necesidad de escribir ningún comando). Todas estas características hacen que un lector pueda aprender a usar el sistema hipermedial en forma rápida y natural.

- Eficiencia en el uso: después de la etapa de aprendizaje en el uso del sistema, el objetivo es lograr que el lector tenga un alto nivel de productividad. Con respecto al contenido de los nodos, éste es mostrado en forma clara y precisa para facilitar su comprensión por parte del lector. Por otro lado, las funciones adicionales muestran los resultados resaltando los nodos y enlaces en mapas esquemáticos del grafo. En un ambiente hipermedial, esto es de mucha utilidad, ya que permite a los lectores tener una ubicación contextual dentro de la red de nodos.

- Facilidad para recordar: el objetivo en el diseño de la interfaz es que un lector pueda usar nuevamente el sistema después de un período de no hacerlo, sin tener que repetir la etapa de aprendizaje. Este punto está relacionado con el primero, ya que una interfaz clara y bien definida resultará fácil no sólo de aprender, sino también de recordar.

- Ocurrencia de pocos errores: otro objetivo importante tenido en cuenta al diseñar la interfaz de IMHEF, es que los lectores cometan pocos errores durante el uso del mismo, y que en caso de producirse uno puedan recuperarlo fácilmente. Si un lector siguió erróneamente un enlace, le resultará sencillo regresar al nodo del cual partió (mecanismo de "backtrack"). Por otro lado, si ha seguido un camino que ya no le interesa, en cada pantalla tiene disponible un ícono para ir en forma directa al nodo índice.

- Agradable para usar: en el desarrollo del prototipo funcional se trató de lograr una interfaz que resulte agradable y placentera para los lectores. La posibilidad que brinda hipermedia de "navegar" por la información, le ofrece a los lectores gran libertad para construir su propia secuencia de lectura. La combinación del mecanismo de navegación y del uso de una interfaz icónica para implementarlo, dan como resultado un sistema totalmente amigable para el lector.

7. IMPLEMENTACION DE IMHEF

7.1. Recopilación de información

Esta etapa consistió en la búsqueda, selección y organización de la información disponible sobre Redes de datos.

Se partió de la recopilación de información dispersa, utilizando documentos originales publicados en congresos, artículos de revistas técnicas, libros generales, documentos distribuídos a través del correo electrónico como RFC's (Request for Comments), FAQ (Frequently Asked Questions) y FYI (For Your Information). Las fuentes de información fueron seleccionadas por su actualidad, compatibilidad y porque reunían el nivel de información requerida.

Además, se consultaron especialistas en el tema que aportaron no sólo conocimientos técnicos, sino también elementos fundamentales para la estructuración de la información según niveles.

Para la organización y presentación de la información se analizaron cuidadosamente los intereses y preferencias de los lectores de acuerdo a los niveles establecidos (coloquial, de uso común y de referencia).

Dado que en el prototipo IMHEF la selección de nivel condiciona todas las otras herramientas, se consideró que bastaba con implementar sólo un nivel. Se eligió el nivel coloquial, en cuya presentación de información se introdujeron ejemplos, animaciones y metáforas gráficas para ilustrar los conceptos. Las definiciones incorporadas fueron analizadas detalladamente para que puedan ser comprendidas por lectores que carecen de conocimientos sobre el tema.

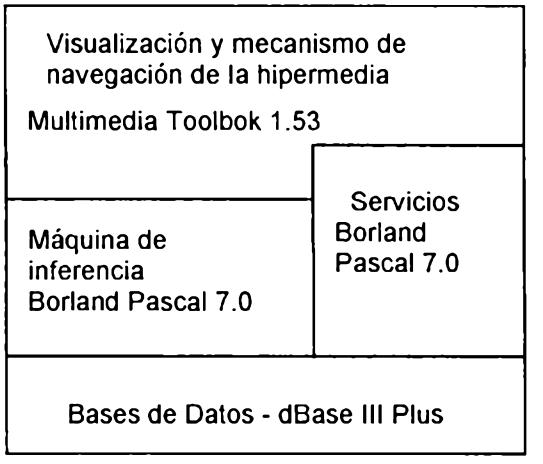
Esta etapa denominada de autoría consistió en: la recopilación de información sobre Redes de datos, la selección del material según los diferentes propósitos y la posterior estructuración de acuerdo al paradigma de trabajo adoptado: *hipermedia*. Este último punto representa la organización de los conceptos en nodos y la definición de vínculos entre dichos conceptos. La tarea de creación de enlaces entre nodos fue realizada cuidadosamente para que los lectores puedan conocer las relaciones existentes entre los conceptos.

7.2. Construcción del prototipo IMHEF

El prototipo IMHEF, tal como se explicó en el capítulo Cinco, fue implementado integrando varias herramientas de construcción de software. En la parte de visualización de la hipermedia e implementación del mecanismo de navegación se utilizó Multimedia Toolbook 1.53. Para la implementación de las funciones que forman el modelo MHEF, se utilizó un lenguaje de propósito general: Borland Pascal 7.0 y un administrador de bases de datos: Dbase III Plus.

En la implementación de IMHEF podemos observar tres niveles claramente definidos: el nivel superior representado por Multimedia Toolbook, un nivel inferior que comprende el almacenamiento en bases de datos de la estructura del hipergrafo y del árbol de conceptos, y un tercer nivel que actúa de enlace entre los dos niveles anteriores. El nivel intermedio comprende una máquina de inferencia construída en Borland Pascal 7.0 y un conjunto de servicios provistos por librerías también escritas en Pascal.

En el siguiente esquema se puede observar la estructura interna de IMHEF.



En los puntos siguientes se describirá detalladamente cómo fueron implementados cada uno de los tres niveles.

7.2.1. Visualización de la hipermedia

La visualización de la hipermedia fue implementada en Multimedia ToolBook 1.53, aprovechando las facilidades que provee, para la construcción de interfaces gráficas o visuales.

De acuerdo a la terminología de Multimedia ToolBook 1.53, se construyó un "libro" para cada uno de los niveles de usuario definidos previamente (coloquial, de uso común y de referencia). En el prototipo implementado se completó sólo el libro del nivel coloquial. Este libro tiene una única ventana de trabajo asociada y es la ventana principal del sistema (la que permanece abierta durante toda la sesión) (Fig. 8). Está conformado por páginas, que son las que representan los nodos de la hipermedia, y por objetos de tipo botón, que son la representación gráfica de los enlaces y los que implementan el mecanismo de navegación. Las páginas se muestran de a una por vez en la ventana asociada al libro.

En la implementación del libro correspondiente al nivel coloquial, se definieron tres capítulos (o grupos de páginas): "Que son las redes de datos?", "Cómo funcionan?" y "Servicios provistos por las redes de datos". Las páginas de cada uno de los capítulos tienen asociado un color de fondo particular; las del capítulo "Que son las redes de datos?" tienen como color de fondo el azul, las del capítulo "Cómo funcionan?" el color amarillo y las del capítulo "Servicios provistos por las redes de datos" el color verde. Los colores del fondo de las páginas se corresponden con los colores que aparecen en el índice. Esta agrupación por colores, tiene como objetivo ayudar al lector en la "ubicación" mental respecto del tema tratado.

Por otra parte, las funciones adicionales que provee el sistema, están simbolizadas por íconos que se ubicaron alineados en la parte inferior de todas las páginas. Cabe destacar que las páginas en Multimedia ToolBook 1.53, están formadas por un background (fondo de la página) y un foreground (frente de la página). En el background se ubican todos los objetos de la interfaz que son comunes a las páginas que comparten ese background y en el foreground sólo

los que son particulares a cada una de las páginas. Por lo tanto, el panel de funciones que es el mismo para todas las páginas del libro se ubicó en el background de las páginas.

Las páginas se construyeron combinando los distintos objetos gráficos y de texto provistos por Multimedia ToolBook 1.53. También se incorporaron a las páginas animaciones simples, construídas a partir de secuencias de figuras. En el nivel coloquial se emplearon varias metáforas gráficas con el objetivo de aliviarle al lector la comprensión de los temas abordados.

Siguiendo la terminología de Multimedia Toolbook 1.53, la ventana del grafo que se abre al ejecutar el filtro por palabra clave, el tour guiado o el circuito recomendado corresponde a otro libro. El libro del Grafo tiene sólo una página cuyo contenido es el esquema del grafo de la hipermedia.

Las anotaciones internas que se realizan en cada nodo están implementadas a través de libros independientes. Al crearse una anotación en una página, automáticamente se le asocia su propio libro de anotaciones.

7.2.2. Mecanismo de navegación

El mecanismo de navegación fue implementado en Multimedia ToolBook 1.53, utilizando la flexibilidad que provee el lenguaje OpenScript para definir el comportamiento de los botones que se usan para vincular páginas (o nodos).

Los botones, que son la representación visual de los enlaces de la hipermedia, están contenidos en las páginas (o nodos de la hipermedia), y tienen asociados scripts, mediante los que se implementa la navegación. Estos scripts tienen definidos manejadores (handlers), que atienden los diferentes eventos producidos por los lectores sobre el botón, el más usual es "cliquear" sobre el botón (denominado en OpenScript *buttonUp*). Cuando el lector "cliquea" sobre un botón de una página, Multimedia ToolBook 1.53 ejecuta el manejador correspondiente del script del botón, que lo transportará a la página que corresponda. De esta manera, el lector puede navegar por la hipermedia, con la simple selección de un botón. Algunas páginas tienen, además, habilitados los íconos de recorrido secuencial (presentes en el panel de funciones). Los manejadores de los scripts de estos íconos, transportan al lector a la página anterior y a la siguiente.

Dado que el prototipo implementa funciones adicionales que enriquecen el mecanismo de navegación, fue necesario definir manejadores en los scripts de los botones que permitan, por un lado, verificar el estado de navegación actual (navegación usual, por filtro por palabra clave, por tour guiado o por circuito de lectura recomendada) y, por otro lado, habilitar o deshabilitar los enlaces de los nodos según corresponda. Esto se llevó a cabo implementando funciones especiales que se ejecutan cada vez que el lector "pasa" sobre un botón con el mouse (evento denominado en OpenScript *mouseEnter*). Estas funciones verifican el estado de navegación actual y modifican una de las propiedades del botón, el "nombre", para reflejar su estado (habilitado o deshabilitado). Cuando el estado de navegación no es el tradicional y el lector pasa por primera vez por un botón, se le agregan al nombre de ese botón caracteres especiales que permiten identificar su estado de habilitación. Al ejecutarse el evento de "cliquear" (buttonUp) sobre un botón, éste ya tiene actualizado su nombre (indicando si

está o no habilitado). El manejador del script que atiende la selección de dicho evento, verifica el contenido del nombre del botón y dependiendo de este resultado ejecuta alguno de los comandos que implementan la navegación o inhabilita el enlace que parte de ese botón.

Es importante tener en cuenta que todos los botones del sistema, tienen el mismo comportamiento (detallado previamente). Por otro lado, OpenScript provee un mecanismo de jerarquía de objetos para la atención de los eventos producidos por los lectores. Esto significa, que si todos los objetos de una misma página requieren la misma respuesta, ante la ocurrencia de un mismo evento, se debe definir el manejador correspondiente, en el script de la página a la que pertenecen los objetos. Este concepto se extiende para todos los objetos de distintas páginas que tienen el mismo comportamiento; en ese caso los manejadores que implementan el comportamiento compartido se codifican en el script del libro. Esta característica permite factorizar el código escrito en los manejadores de los distintos objetos. Aprovechando esta característica de orientación a objetos de OpenScript, se codificaron las funciones requeridas para verificar el estado de navegación actual de la hipermedia y para determinar la habilitación de los botones, una única vez y en el script del libro. De esta manera el evento de "pasar" por un botón (mouseEnter) sólo dispara un manejador que se encargará de habilitar o deshabilitar a ese botón (modificándole el nombre) de acuerdo al estado de navegación actual. Por otro lado, el evento de "cliquear" sobre él (buttonUp), verifica el estado del botón (habilitado o deshabilitado) inspeccionando el nombre del mismo.

A continuación se muestra el script de los botones, y los manejadores del script del libro utilizados, ilustrando los conceptos previamente descriptos:

1.- Script de los botones:

```
to handle mouseEnter
  system identObjeto

  set identObjeto to getUniqueName (uniqueName of the target)
  send mouse identObjeto
end


to handle buttonUp
  system identObjeto

  if character 1 of (name of identObjeto) is not "/" then
                    send goToPage "caracteristicas e_mail"
  end if
end


to handle mouseLeave
  set sysCursor to 1
end
```

2.- Manejadores del script del libro que implementan la habilitación de botones:

```
to get getUniqueName unUniqueName

  --Controla que el UniqueName recibido tenga nombre,
     si no tiene busca el padre
  set nombre to name of unUniqueName
  set unObjeto to unUniqueName

  while nombre = ""
        set unObjeto to parent of unObjeto
        set nombre to name of unObjeto
  end while

return unObjeto

end getUniqueName


to get cambiarNombre aUniqueName

  system botonHabilitado
  local nombreOriginal

  put name of aUniqueName into nombreOriginal
  set botonHabilitado to testearBotones(aUniqueName)

  if not botonHabilitado then
        put ("/"&nombreOriginal) into nombreOriginal
  end if
  put (nombreOriginal&"*") into nombreOriginal

  return nombreOriginal

end cambiarNombre


to get testearBotones aUniqueName

  system listaBotones
  local nombreBoton,nombrePag,i,esta

  --Recupero el nombre del boton de aUniqueName
  set nombreBoton to name of aUniqueName

  --Recupero el nombre de la pagina donde esta nombreBoton
  set nombrePag to name of this page

  set i to 1
  set esta to false
```

```
    While i<=itemCount(listaBotones) and not esta
          if upperCase(item i of listaBotones)=upperCase(nombreBoton) and\
            upperCase(item i+1 of listaBotones)=upperCase(nombrePag) then
                                                    set esta to true
          end if
          increment i by 2
    end While

  return esta

  end testearBotones

  to handle mouse aUniqueName

    system svFiltro, svCircuito, botonHabilitado, svTour, listaBotonesVisitados

    send recupEstadoFiltro
    send recupEstadoTour
    send recupEstadoCircuito

    conditions
        when svCircuito is true
          if name of aUniqueName contains "/" is false then
           put name of aUniqueName into unNombre
           put "/"&unNombre&"*" into unNombre
           set name of aUniqueName to unNombre
           push aUniqueName onto listaBotonesVisitados
          end if
           put 25 into sysCursor -- boton deshabilitado
        when svFiltro is true or svTour is true
          conditions
           when name of aUniqueName contains "/"
                  put false into botonHabilitado
           when name of aUniqueName contains "*"
                  put true into botonHabilitado
           else
                  set name of aUniqueName to cambiarNombre(aUniqueName)
                  --Apilo los botones visitados para restaurar el nombre
                  push aUniqueName onto listaBotonesVisitados
           end conditions
          if not botonHabilitado then
           put 25 into sysCursor -- boton deshabilitado
          else
           put 6 into sysCursor --boton habilitado
          end if

        else -- svFiltro, svTour y svCircuito contienen false
          put 6 into sysCursor
        end conditions

  end mouse
```

7.2.3. Funciones adicionales

Estas funciones fueron implementadas en Multimedia ToolBook 1.53, utilizando funciones subsidiarias, implementadas en librerías dinámicas, en Borland Pascal versión 7.0 para Windows. A continuación describiré en detalle la implementación de cada una de las facilidades adicionales propuestas por el modelo MHEF para facilitar la búsqueda y visualización de la información. Cabe destacar que en la implementación de dichas funciones se puso especial énfasis en facilitar la tarea del lector al momento de utilizarlas.

7.2.3.1. Filtros: para simplificar la visualización del hiperespacio

Filtro de acuerdo al modelo del usuario: esta facilidad es la más simple de implementar, ya que consiste en acceder al subgrafo de nodos correspondiente al nivel de lectura elegido por el usuario. En este caso no se requiere la construcción de estructuras de datos auxiliares, ya que el lector navega por la totalidad del subgrafo perteneciente a su nivel.

Por otro lado, el lector tiene habilitados todos los enlaces que vinculan los nodos del subgrafo, por lo cual no es necesario analizar si un botón está o no habilitado durante la navegación.

Al comenzar una sesión de trabajo, el lector autodefine su propósito de lectura y sólo tendrá acceso a la versión del hiperespacio correspondiente al nivel elegido durante toda la sesión. La forma de implementación de este método de filtrado es transparente al usuario, ya que el subgrafo obtenido representa para él la totalidad del espacio de navegación.

Filtro por palabra clave: la aplicación de un filtro sobre un hiperdocumento, consiste básicamente en realizar un tamizado del hiperdocumento de acuerdo a una palabra clave. El resultado será la obtención de un subgrafo del grafo original de la hipermedia, donde el conjunto de nodos estará formado solamente por aquellos nodos que estén calificados por la palabra clave elegida y sus derivadas, y el conjunto de aristas, por todas aquellas que conecten nodos seleccionados a través del filtrado. La estructura de datos recuperada se utilizará como soporte para la navegación por filtrado.

La secuencia de acciones a realizar para implementar este método de filtrado, es la siguiente:

- recuperar del árbol de conceptos, el subárbol que tiene como raíz a la palabra clave elegida. Este subárbol se utilizará para obtener las palabras claves derivadas.

- recorrer exhaustivamente el grafo original de la hipermedia y recuperar todos aquellos nodos que estén calificados por la palabra clave dada o por alguna de sus derivadas, y todas las aristas que relacionan dichos nodos. Para poder realizar este chequeo es necesario hacer búsquedas en el árbol de palabras claves obtenido en el paso anterior. El subgrafo resultante de

filtrar el grafo original de la hipermedia por la palabra clave dada, estará formado por los nodos y aristas recuperados.

- una vez que se construyó el subgrafo asociado al filtrado por palabra clave, según los puntos anteriores, la navegación de la hipermedia estará reducida a los nodos y enlaces que pertenecen a esta nueva estructura. Cuando al lector se le presenta un nodo correspondiente al filtro seleccionado, solamente tendrá habilitado aquellos enlaces que lo transporten a nodos pertenecientes al subgrafo recuperado.

Desde el punto de vista de la visualización, el filtro por palabra clave tiene asociado una ventana de trabajo propia, en donde se muestra esquemáticamente un grafo, con los nodos y enlaces pertenecientes al subgrafo filtrado coloreados apropiadamente (Fig. 12).

El lector tiene disponible dos formas de navegación: a) la usual en hipermedia, restringida a los botones habilitados y b) a partir del mapa, accediendo en forma directa a los nodos calificados por la palabra clave elegida. A pesar de la sobrecarga cognitiva originada por la tarea de diferenciar cuáles son los vínculos habilitados según el filtro aplicado (la forma a) de navegación), el lector cuenta con una herramienta de consulta poderosa que le posibilitará el acceso a los nodos que tratan directamente el tema de su interés, evitando abordar aquellos nodos que no lo hacen. Por otro lado, si bien la posibilidad de contar con un mapa esquemático del grafo filtrado introduce un cambio en la interfaz y en la interacción con la aplicación, el lector dispone de una herramienta que le permite "ubicar" contextualmente el tema de su interés respecto de la totalidad de los temas abordados en el hiperdocumento y acceder en forma directa a alguno de los nodos recuperados por la consulta.

Como ya fue mencionado anteriormente, en términos de Multimedia Toolbook 1.53 la ventana del grafo asociada al filtro por palabra clave, se implementa como un libro separado del libro de la hipermedia.

7.2.3.2. Vistas de la hipermedia: para evitar la "desorientación del usuario"

Tours Guiados: básicamente crear un tour guiado consiste en construir una estructura jerárquica (un árbol) a partir del grafo de la hipermedia. Esta nueva estructura de datos es una simplificación del hiperdocumento original y será el soporte de la navegación cuando el lector elija esta facilidad.

Para construir esta nueva estructura de datos se llevan a cabo la siguiente secuencia de acciones:

- crear a partir del grafo original de la hipermedia, un subgrafo donde el conjunto de vértices está formado por todos aquellos nodos que fueron estadísticamente más visitados por los lectores del sistema, y el conjunto de aristas por todas aquellas aristas que conecten nodos que cumplan la condición de más visitados enunciada.

- el subgrafo resultante del punto anterior estará formado por varias componentes conexas. A partir de dicho subgrafo se van construyendo árboles tomando como raíz los nodos de mínima incidencia. De esta manera,

cada componente conexa se transforma en un árbol. En cada uno de estos árboles se conservan respecto de la componente conexa correspondiente todos los vértices y se eliminan algunas aristas. Por lo tanto, respecto del subgrafo de vértices más visitados se eliminaron solamente asociaciones entre conceptos.

- a partir de los árboles obtenidos se construye un bosque generador.

- con el bosque generador creado en el punto anterior, se construye un único árbol aplicando un criterio de distancias mínimas respecto de la raíz del grafo original de la hipermedia.

Se calculan las distancias mínimas, desde la raíz del grafo original de la hipermedia a cada una de las raíces de los árboles del bosque generador. El árbol que contenga la raíz de distancia mínima será el árbol definitivo inicial. A este árbol se le insertarán los restantes árboles del bosque aplicando también un criterio de distancias mínimas. Este criterio consiste en: para cada uno de los restantes árboles del bosque generador, obtener el nodo del árbol definitivo inicial cuya distancia a la raíz del árbol en cuestión, sea mínima y, a partir de dicho nodo se insertará el nuevo árbol. Al finalizar este proceso el árbol definitivo contendrá todos los árboles generadores obtenidos en el punto anterior.

El criterio de cercanía aplicado para construir el árbol asociado al tour guiado, se basa en el hecho que, si un nodo es adyacente a otro nodo en el grafo del hipertexto, significa que existe una asociación conceptual entre los contenidos de cada uno de los nodos. El mejor de los casos sería que siempre se puedan encontrar nodos adyacentes. De no ser así, se establecen nodos de distancia mínima, teniendo en cuenta que esta cercanía favorecerá a la nueva asociación conceptual que establecerá el árbol que se está construyendo.

- una vez que se construyó el árbol asociado al tour guiado, según los puntos anteriores, la navegación de la hipermedia estará reducida a los nodos y enlaces que pertenecen a esta nueva estructura. Cuando al lector se le presenta un nodo del tour guiado, solamente tendrá habilitado aquellos enlaces que también pertenezcan al árbol del tour guiado.

Desde el punto de vista de la visualización, el tour guiado tiene asociado una ventana de trabajo propia, en donde se muestra esquemáticamente un grafo, con los nodos y enlaces pertenecientes al tour coloreados apropiadamente (los distintos colores permiten distinguir los enlaces nuevos de los ya existentes) (Fig. 9).

De la misma manera que en el filtro por palabra clave, el lector dispondrá de dos formas de navegación: a) la usual en hipermedia, a través de los botones habilitados y de los botones que representan los nuevos enlaces, y b) a partir del mapa, accediendo en forma directa a los nodos más visitados. En forma similar al mecanismo de filtrado descripto en el punto anterior, si bien esta facilidad introduce una componente de cambio en la interacción e interfaz de la aplicación, se le ofrece al lector un espacio de exploración simplicado y

transitorio en donde se han eliminado ciertos conceptos y algunas asociaciones entre conceptos.

En términos de Multimedia Toolbook 1.53, la ventana del grafo asociada al tour guiado, se implementa como un libro separado del libro de la hipermedia.

Circuito de lectura recomendada: esta facilidad, como ya fue explicado previamente, tiene como objetivo aliviar el esfuerzo del lector en la comprensión de los temas tratados, en el sentido que es el autor quien define los tópicos que contendrá y la secuencia de visita de los mismos. Es una facilidad de características estáticas, que consiste en la construcción de una estructura lineal a partir del grafo de la hipermedia.

La secuencia de acciones que se llevan a cabo para construir el circuito de lectura recomendada es la siguiente:

- recuperar la lista de nodos asociada al circuito de lectura recomendada, según la definió el autor.

- una vez obtenida la secuencia de nodos, la navegación de la hipermedia estará reducida a esta nueva estructura lineal. Los únicos botones que el lector tendrá habilitados son los que lo llevan al nodo anterior y siguiente en dicha secuencia.

De la misma manera que las dos facilidades anteriores, el circuito de lectura recomendada tiene asociada una ventana de trabajo propia, en la que aparece el grafo de la hipermedia con los nodos y enlaces de la secuencia coloreados apropiadamente (en ella se distinguen los nuevos vínculos secuenciales).

En forma similar a las dos facilidades anteriores (filtro por palabra clave y tour guiado), el lector cuenta con dos formas de navegación: a) la usual en hipermedia, a través de los botones que representan los nuevos enlaces secuenciales, y b) a partir del mapa, accediendo en forma directa a los nodos pertenecientes al circuito. Esta facilidad introduce cambios en la interacción e interfaz de la aplicación, pero le provee al lector un mecanismo de recorrido secuencial a través del circuito. Esta forma de recorrido disminuye la sobrecarga cognitiva propia del mecanismo de navegación, ya que desde un nodo el lector sólo puede acceder al nodo siguiente o al anterior (no debe tomar decisiones durante el recorrido).

7.2.3.3. Anotaciones personalizadas: para mantener información privada

Esta facilidad se implementó íntegramente en Multimedia Toolbook 1.53, y consistió en la definición de un libro para cada uno de los dos tipos de anotaciones provistas por el modelo.

Anotaciones internas: para esta clase de anotaciones se definió un libro formado por un conjunto de páginas organizadas en una especie de agenda o cuaderno. Cada nodo tendrá asociado un libro de anotaciones independiente (el nombre del libro representa el nodo al que pertenece). Inicialmente esta agenda tiene una sola página, y a medida que el lector le va incorporando

nuevas anotaciones a un nodo se van agregando páginas a la agenda. El foreground de cada página contiene un panel de escritura, en donde el lector realiza sus anotaciones. El panel de escritura tiene la funcionalidad de un editor de texto simplificado. El background de las páginas de la agenda, contiene un panel de funciones, representadas por íconos, que permiten el recorrido a través de las anotaciones realizadas, la localización directa de una de ellas y salir de la agenda de anotaciones (Fig. 10).

Anotaciones de referencia terminal: estas anotaciones se implementan en un libro separado donde cada página representa la anotación asociada a un nodo (recordemos que el lector puede crear sólo una anotación externa en cada nodo). El nombre de la página en el libro de anotaciones externas representa el nodo al que pertenece esa anotación. El lector puede usar esta ventana en dos modos diferentes: edición y visualización. En el modo de edición el lector puede incorporar texto a la página y darle el formato que desee (tipo de letra, tamaño, resaltado, etc.), mientras que en el modo de visualización sólo puede ver el contenido de la página (no modificarlo). La página contiene en la parte inferior un panel de funciones de edición de texto. Además, presenta dos íconos: uno de salida de las anotaciones y otro a través del cual el lector puede cambiar el modo de trabajo.

7.2.4. Implementación de las librerías

7.2.4.1. Librería Grafo

Teniendo en cuenta como premisa, disminuir la complejidad del peor de los casos de la ejecución de los procedimientos y funciones, se definió el concepto de identificador único para cada uno de los vértices del grafo [Aho83]. En términos de la implementación de las funciones este identificador permite realizar acceso directo a la información asociada a cada uno de los vértices, y así las operaciones que manipulan los vértices (como por ejemplo, conectar_vertices, pertenece, comenzar_adya, proximo_adya, fin_adya, etc.) tienen orden de ejecución constante (que es el mejor de los casos), y por lo tanto se mejorará la complejidad de las operaciones que se puedan componer a partir de estas funciones.

La estructura de datos elegida para representar el grafo es la denominada lista de adyacencia. Consiste en definir una secuencia de nodos o vértices (un arreglo o una lista encadenada), donde cada uno de ellos tiene almacenado un contenido y una referencia a la lista de vértices adyacentes.

A continuación se describe la estructura de datos empleada:

```
const
    maxLong = 500;
    maxVisita =200;
    cantAdya=15;
    maxNodos = 100;
```

```
type
    listaAdya = ^Nodo;
    nodo  = Record
                ind : 1..maxLong;
                peso: record
                        link,boton: Pchar;
                      end;
                sgte: listaAdya;
              end;

    Grafo = Record
                vertices : array[1..maxLong] of  record
                                                   nombre: Pchar;
                                                   descripcion: Pchar;
                                                   visita: 0..maxVisita;
                                                   clave1,clave2,clave3: Pchar;
                                                 end;
                adya     : array[1..maxLong] of record
                                                   cabeza: ListaAdya;
                                                   actual: ListaAdya;
                                                 end;
                total_de_vertices: 0..maxLong;
                vert_actual:1..maxLong;
              end;
```

function crear(var pg:Pointer):Byte
Crea un grafo vacío y lo devuelve en la variable pg (en términos de implementación significa inicializar la variable puntero pg con la constante nil).

function destruir(var pg:Pointer):Byte
Libera el espacio de memoria en donde está alocada la variable pg.

function inicGrafo(var pg:Pointer):Byte
Esta función devuelve en la variable de tipo pointer pg, el grafo inicializado con la información correspondiente a los vértices y las aristas. La inicialización del grafo se realiza a partir de los archivos dBase: Vértices y Aristas. En la implementación de esta función fue necesario utilizar el "motor" gerenciador de archivos XBase descripto en la sección arquitectura del sistema.

function cant_vertices(var pg:Pointer):Integer
Esta función devuelve la cantidad de vértices del grafo referenciado por la variable puntero pg.

function agregar_Vertice(var pg:Pointer; nom:Pchar):Byte
Esta función agrega un vértice, almacenado en la variable nom, al conjunto de vértices del grafo referenciado por la variable pg.

function pertenece(var pg:Pointer;indice:Integer):Byte
Esta función verifica si un vértice pertenece al conjunto de vértices del grafo referenciado por la variable pg. Devuelve 1 (uno) si el vértice que tiene asociado identificador índice pertenece al grafo y 0 (cero) en caso contrario.

function conectar_vertices(var pg:Pointer; indice1, indice2:Integer; link, boton:Pchar): Byte
Esta función agrega una arista al conjunto de aristas del grafo referenciado por la variable pg. La arista a agregar, conecta los vértices que tienen identificador índice1 e índice2 respectivamente, y tiene asociada como peso el contenido de la variable link y como botón el contenido de la variable botón.

function esta_Conectado(var pg:Pointer;indice1,indice2:Integer):Byte
Esta función verifica si dos vértices dados del grafo referenciado por la variable pg, están conectados. Devuelve 1 (uno) si los vértices que tienen identificador indice1 e índice2 están conectados y 0 (cero) en caso contrario.

function identificador(var pg:Pointer;nombre:Pchar):Integer
Esta función devuelve el identificador de un vértice en el grafo referenciado por la variable pg. El vértice en cuestión está almacenado en el parámetro nombre.

function vertice(var pg:Pointer;indice:integer):Pchar
Esta función devuelve el contenido de un vértice en el grafo referenciado por la variable pg, a partir de su identificador (almacenado en el parámetro índice).

function obtener_lista_vertices(var pg,pl:Pointer):Byte
Esta función recupera la lista de vértices del grafo referenciado por la variable pg. Dado un grafo, devuelve en la variable de tipo puntero pl, la lista de vértices del grafo dado.

function grabarVisitas(var pg:Pointer):Byte
Esta función actualiza el archivo Vértices con el registro actual de visitas, de cada uno de los vértices del grafo referenciado por la variable pg. En términos de implementación esta función recorre el grafo y para cada uno de sus vértices graba en el archivo Vértices la cantidad actual de visitas.

function incrementar(var pg:Pointer; indice:Integer):Integer
Esta función incrementa en 1 (uno) la cantidad de visitas del vértice que tiene identificador indice, y devuelve la cantidad actual de visitas de dicho vértice. El grafo está referenciado por la variable de tipo puntero pg.

function obtener_Visitas(var pg:Pointer;indice:integer):Integer
Esta función recupera la cantidad de visitas registradas sobre el vértice que tiene identificador índice, en el grafo referenciado por pg.

function link(var pg:Pointer;indice1,indice2:Integer):Pchar
Esta función devuelve el peso de la arista que conecta dos vértices dados, en el grafo apuntado por la variable pg. Dado un grafo y dos identificadores de vértices indice1 e índice2, devuelve el peso de la arista que los conecta.

function boton(var pg:Pointer;indice1,indice2:Integer):Pchar
Esta función devuelve el nombre del botón que conecta dos vértices dados, en el grafo referenciado por la variable pg. Dado un grafo y dos identificadores de vértices índice1 e índice2, devuelve el nombre asociado al botón que los conecta.

function comenzar_vertice(var pg:Pointer):Integer
Esta función implementa el comienzo de la iteración sobre los vértices del grafo referenciado por la variable pg. El efecto que tiene es inicializar el apuntador actual de la lista de vértices en la cabeza o inicio de la misma.

function proximo_vertice(var pg:Pointer):Integer
Esta función tiene dos efectos: 1) devolver el identificador del vértice asociado al apuntador actual de la lista de vértices del grafo referenciado por la variable pg y; 2) actualizar el apuntador actual en el siguiente ítem de la lista de vértices.

function fin_vertice(var pg:Pointer):Byte
Esta función devuelve 1 (uno) si se tomaron todos los ítems de la lista de vértices del grafo pg, y 0 (cero) en caso contrario.

function comenzar_adya(var pg:Pointer;indice:Integer):Byte
Esta función implementa el comienzo de la iteración sobre la lista de vértices adyacentes del vértice que tiene identificador índice, del grafo apuntado por la variable puntero pg. El efecto que tiene es inicializar el apuntador actual de la lista de vértices adyacentes del vértice que tiene identificador índice, en la cabeza o inicio de la misma.

function proximo_adya(var pg:Pointer;indice:Integer):Byte
Esta función tiene el siguiente efecto, actualizar el apuntador actual de la lista de adyacencia del vértice que tiene identificador índice, del grafo referenciado por la variable pg.

function obtener_adya(var pg:Pointer;indice:Integer):Integer
Esta función devuelve el identificador del vértice correspondiente al apuntador actual de la lista de adyacencia del vértice que tiene identificador índice, en el grafo referenciado por la variable puntero pg.

function fin_adya(var pg:Pointer;indice:Integer):Byte
Esta función devuelve 1 (uno) si se tomaron todos los ítems de la lista de vértices adyacentes del vértice que tiene identificador índice, en el grafo referenciado por la variable pg, y 0 (cero) en caso contrario.

function obtener_link(var pg:Pointer;indice:Integer):Pchar
Esta función devuelve el peso de la arista que conecta el vértice que tiene identificador índice, con el vértice adyacente actual de la lista de adyacencia del vértice con identificador índice, en el grafo referenciado por la variable pg.

function obtener_boton(var pg:Pointer;indice:Integer):Pchar
Esta función devuelve el nombre del botón (enlace) que conecta el vértice que tiene identificador índice, con el vértice adyacente actual de la lista de

adyacencia del vértice con identificador índice, en el grafo referenciado por la variable pg.

function obtener_descrip(var pg:Pointer;indice:Integer):Pchar
Esta función devuelve la descripción asociada al vértice que tiene identificador índice, en el grafo referenciado por pg.

function obtener_clave(var pg:Pointer;indice:Integer;
indClave:Integer):Pchar
Esta función devuelve la clave con índice indClave del vértice que tiene identificador índice, en el grafo referenciado por pg.

function cant_incidentes(var pg:Pointer;indice:Integer):Integer
Esta función devuelve la cantidad de incidentes del vértice que tiene identificador índice, en el grafo referenciado por pg.

function planar(var pg:Pointer;var plista:Pointer;indice:Integer;
var ptrArbol:Pointer):Byte
Esta función transforma un grafo en un árbol. Dado un grafo (referenciado por la variable pg) y un identificador de un vértice (índice), devuelve el árbol que se puede construir a partir del vértice dado. Esta función además elimina de la lista de vértices del grafo (referenciada por plista) los vértices que forman parte del árbol que se construyó.

function dijkstra(var pg:Pointer;iorigen,idestino:integer):Integer
Esta función obtiene la longitud del camino de costo mínimo entre dos vértices dados de un grafo. Dado un grafo referenciado por la variable pg, y dos identificadores de vértices iorigen e idestino, esta función devuelve el camino de longitud mínima entre los vértices dados, utilizando el algoritmo de Dijkstra.

7.2.4.2. Librería Arbol

La estructura de datos elegida para el representar el árbol es la conocida como hijo izquierdo y hermano derecho. Consiste en almacenar para cada nodo del árbol un contenido, una referencia a su hijo izquierdo (que también es un árbol) y una referencia a su hermano derecho (que también es un árbol). A continuación se detalla la estructura de datos definida:

```
type
  Arbol= ^Nodo;
  Nodo  = Record
            dato: Pchar;
            Hi: arbol;
            linkI: record
                     etiqueta,boton:Pchar;
                   end;
            HerD: arbol;
            linkD: record
                     etiqueta,boton:Pchar;
                   end;
          end;
```

function crearArbol(var pa:Pointer):Byte
Crea un árbol vacío y lo devuelve en la variable pa (en términos de implementación significa inicializar la variable puntero pa con la constante nil).

function inicializa(var Pa:Pointer;nombre:Pchar):Byte
Esta función inicializa un árbol del tipo hoja (sin descendientes). Inicializa el árbol referenciado por la variable pa, con el contenido almacenado en el parámetro nombre.

function obtener_dato(var pa:Pointer):Pchar
Esta función recupera el contenido de la raíz del árbol referenciado por la variable pa.

function masCercano(var ptrArbol, pgrafo, arbolMasCerca:Pointer; dest:Pchar):Integer
Esta función devuelve el subárbol de un árbol dado, que tiene como raíz al nodo más próximo a un nodo dado. La distancia mínima entre estos nodos se obtiene aplicando el algoritmo de Dijkstra sobre un grafo también dado como parámetro. El árbol del que se obtendrá el subárbol está referenciado por el parámetro ptrArbol; el grafo sobre el que se calcula la distancia mínima está referenciado por el parámetro pgrafo; y el nodo dado como destino para obtener la distancia mínima por el parámetro dest.

function vacio(var pa:Pointer):Byte
Esta función verifica si el árbol referenciado por el parámetro pa está vacío (no contiene nodos). Devuelve 1 (uno) si está vacío y 0 (cero) en caso contrario.

function hijo_izquierdo(var pa:Pointer;var hijolz:Pointer):Byte
Esta función devuelve el subárbol izquierdo del árbol referenciado por el parámetro pa.

function hermano_derecho(var pa:Pointer;var herderecho:Pointer):Byte
Esta función devuelve el subárbol que es hermano derecho del árbol referenciado por el parámetro pa.

function obtener_linklz(var pa:Pointer):Pchar
Esta función devuelve la etiqueta (peso) de la arista que conecta al árbol referenciado por el parámetro pa con su subárbol izquierdo.

function obtener_botonlz(var pa:Pointer):Pchar
Esta función devuelve el nombre del botón que vincula el árbol referenciado por el parámetro pa, con su subárbol izquierdo.

function obtener_linkDer(var pa:Pointer):Pchar
Esta función devuelve la etiqueta (peso) de la arista que vincula el árbol referenciado por el parámetro pa con el subárbol que es su hermano derecho.

function obtener_botonDer(var pa:Pointer):Pchar
Esta función devuelve el nombre del botón que vincula el árbol referenciado por el parámetro pa con el subárbol que es su hermano derecho.

function agregarHijo(var pa:Pointer;var phi:Pointer;
nlink,nboton:Pchar):Byte
Esta función agrega al árbol referenciado por el parámetro pa, un árbol referenciado por el parámetro phi. La arista que los conecta tiene asociada como etiqueta el valor dado por el parámetro nlink y como botón el valor dado por el parámetro nboton.

function inicLinkIz(var pa:Pointer;nombre:Pchar):Byte
Esta función inicializa el valor de la etiqueta de la arista que conecta el árbol referenciado por el parámetro pa con su subárbol izquierdo, con el valor dado por el parámetro nombre.

function inicBotonIz(var pa:Pointer;nombre:Pchar):Byte
Esta función inicializa el valor del botón de la arista que conecta el árbol referenciado por el parámetro pa con su subárbol izquierdo, con el valor dado por el parámetro nombre.

function inicLinkDer(var pa:Pointer;nombre:Pchar):Byte
Esta función inicializa el valor de la etiqueta de la arista que conecta el árbol referenciado por el parámetro pa con el subárbol que es su hermano derecho, con el valor dado por el parámetro nombre.

function inicBotonDer(var pa:Pointer;nombre:Pchar):Byte
Esta función inicializa el valor del botón de la arista que conecta el árbol referenciado por el parámetro pa con el subárbol que es su hermano derecho, con el valor dado por el parámetro nombre.

function es_hoja(pa:Pointer):Byte
Esta función verifica que el árbol referenciado por el parámetro pa no tiene hijos. Devuelve 1 (uno) si el árbol dado no tiene hijos y 0 (cero) en caso contrario.

function esta(var pa:Pointer;dato:Pchar;var psubArbol:Pointer):Byte
Esta función verifica que el árbol referenciado por el parámetro pa contenga el valor almacenado en el parámetro dato. Si el valor dado está en el árbol, el parámetro psubArbol contendrá el subárbol que tiene como raíz a dato. La función devuelve 1 (uno) si el valor dado está en el árbol y 0 (cero) en caso contrario.

function inicArbol(var pa:Pointer):Byte
Esta función inicializa el árbol referenciado por el parámetro pa con valores almacenados en el archivo dBase Concepto.dbf. En términos de implementación, esta función usa el "motor" gerenciador de archivos descripto en la sección arquitectura del sistema.

function destruirArbol(var pa:Pointer):Byte
Esta función libera el espacio de memoria alocado por el parámetro pa.

7.2.4.3. Librería Bosque

La estructura de datos utilizada para representar el bosque el simple: un arreglo donde cada una de sus componentes contiene punteros a zonas de memoria ocupadas por árboles.

A continuación se detalla la estructura de datos empleada:

```
const
   max=100;

type
   tipo_bosque= record
                    arboles: array[1..max] of pointer;
                    cant: 0..max;
                    ite: 1..max;
                end;
```

function CrearBosque(var pb:Pointer):Byte;
Esta función crea un bosque vacío y lo devuelve en el parámetro pb. En términos de implementación consiste en inicializar la variable pb con la constante nil.

function AgregarArbol(var pb, pa:Pointer):Byte;
Esta función agrega al bosque referenciado por el parámetro pb, un árbol referenciado por el parámetro pa.

function ComenzarBosque(var pb:Pointer):Byte;
Esta función inicializa el iterador sobre la secuencia de árboles referenciada por el parámetro pb, con la cabeza o el comienzo de esta lista.

function ProximoArbol(var pb:Pointer;var proxArbol:Pointer):Byte;
Esta función avanza un lugar el iterador de la secuencia de árboles referenciada por el parámetro pb. Devuelve en el parámetro proxArbol el árbol correspondiente a la iteración actual.

function FinBosque(pb:Pointer):Byte;
Esta función verifica si el iterador sobre el bosque referenciado por el parámetro pb, llegó al final. Devuelve 1 (uno) si se llegó al final y 0 (cero) en caso contrario.

function RecupArbol(var pb:Pointer;pos:Byte;var parbol:Pointer):Byte;
Esta función devuelve en el parámetro parbol el árbol correspondiente a la posición pos en la secuencia de árboles referenciada por el parámetro pb.

function DestruirBosque(var pb:Pointer):Byte;
Esta función libera la zona de memoria alocada al parámetro pb.

7.2.4.4. Librería Lista de Enteros

La estructura de datos utilizada para representar la lista de enteros es un arreglo donde cada una de sus componentes contiene un número entero.

A continuación se detalla la estructura de datos empleada:

```
const
    maxLong = 1000;

type
    vecNodos= record
                etiquetas: array[1..maxLong] of integer;
                cant     : 0..maxLong;
                ite      : 1..maxLong;
              end;
```

function crearListaInt(var pl:Pointer):Byte
Esta función crea una lista de números enteros vacía. En términos de implementación consiste en inicializar el parámetro pl con la constante nil.

function agregarItemInt(var pl:Pointer;dato:integer):Integer
Esta función agrega un número entero almacenado en el parámetro dato, en la lista referenciada por el parámetro pl.

function agregarItemIntEn(var pl:Pointer;pos,dato:integer):Byte
Esta función agrega un número entero almacenado en el parámetro dato, en la lista referenciada por el parámetro pl en la posición pos.

function eliminarItemInt(var pl:Pointer;dato:Integer):Byte
Esta función elimina de la lista referenciada por el parámetro pl, un elemento almacenado en el parámetro dato.

function elimItemIntDeLaPos(var pl:Pointer;indice:integer):Byte
Esta función elimina de la lista referenciada por el parámetro pl, el elemento ubicado en la posición referenciada por el parámetro índice.

function primerItemInt(var pl:Pointer):Byte
Esta función inicializa el iterador sobre la lista referenciada por el parámetro pl, con la cabeza de la misma.

function proximoItemInt(var pl:Pointer):Integer
Esta función devuelve el ítem actual (el que apunta el iterador) de la lista referenciada por pl e incrementa el iterador de la lista en uno.

function finListaInt(var pl:Pointer):Byte
Esta función verifica el iterador de la lista referenciada por el parámetro pl. Si el iterador llegó al final, devuelve 1 (uno) y en caso contrario devuelve 0 (cero).

function perteneceListaInt(var pl:Pointer;indice:Integer):Byte
Esta función verifica si el elemento almacenado en el parámetro índice pertenece a la lista referenciada por el parámetro pl. Si pertenece a la lista devuelve 1 (uno) y 0 (cero) en caso contrario.

function itemDeLaPos(var pl:Pointer;indice:integer):Integer
Esta función devuelve el elemento ubicado en la posición índice de la lista referenciada por el parámetro pl.

function posicion(var pl:Pointer;indice:integer):Integer
Esta función devuelve la posición del elemento contenido en índice, en la lista referenciada por el parámetro pl.

function cantItemInt(var pl:Pointer):Integer
Esta función devuelve la cantidad de ítems de la lista referenciada por el parámetro pl.

function destruirListaInt(var pl:Pointer):Byte
Esta función libera el espacio de memoria asignado a la variable de tipo puntero pl.

7.2.5. Bases de datos

Como fue explicado anteriormente, la información identificatoria del hiperdocumento o grafo de la hipermedia: nodos o vértices y enlaces entre ellos, y la información acerca de la jerarquía de conceptos se almacena en archivos de datos implementados en DBASE III Plus.

La información del grafo se almacena en un **archivo de nodos** denominado *Vértices* y en **un archivo de aristas** (que representan las conexiones entre los nodos) denominado *Aristas*.

En cada registro de la base de datos *Vértices*, se guarda la siguiente información del nodo: nombre del nodo (que coincide con el nombre de la página en Multimedia Toolbook 1.53), su descripción (una breve leyenda que describe el contenido del nodo), las palabras claves por las que está clasificado (a lo sumo tres) y su registro de visitas.

En cada registro del archivo de *Aristas*, se almacena: el nombre del nodo origen de la arista (o página origen), el número de registro dentro del archivo de nodos del nodo destino de la arista (de esta manera es posible realizar acceso directo a los datos del nodo destino), el nombre del botón que conecta el nodo origen con el nodo destino y el nombre del enlace (esta información es usada para graficar los vínculos habilitados en la ventana del grafo, durante el filtrado por palabra clave, tour guiado y circuito de lectura recomendada).

La información del árbol de conceptos se almacena en un **archivo de conceptos**, denominado *Conceptos*, donde cada registro contiene el nombre del concepto, el número de registro del archivo de conceptos donde está su hijo izquierdo y el número de registro del archivo de conceptos donde está su hermano derecho.

A continuación se describirán esquemáticamente las bases de datos definidas:

Vértices.dbf

Nodo	nombre del nodo o de la página (es un nombre único)
Clave1	palabra clave por la que está calificado el nodo
Clave2	palabra clave por la que está calificado el nodo
Clave3	palabra clave por la que está calificado el nodo
Visitas	cantidad de visitas del nodo

Aristas.dbf

Origen	nombre del nodo origen de la arista
Adyacente	nro. de registro del archivo de Vértices donde está almacenado el nodo destino de la arista
Link	nombre del enlace que vincula el nodo origen y el destino
Boton	nombre del botón que vincula el nodo origen y el destino

Conceptos.dbf

Nodo	nombre del concepto
Hi	nro. de registro en el archivo de Conceptos donde está almacenado el hijo izquierdo del concepto
Hder	nro. de registro en el archivo de Conceptos donde está almacenado el hermano derecho del concepto

7.3. Reflexiones sobre la implementación

La etapa de testeo de las hipótesis de trabajo del modelo MHEF fue realizada obteniéndose un producto que refleja la funcionalidad buscada.

La interfaz del usuario final es manejable lográndose una visión unificada, evitando desconcertar al lector.

La extensa descripción de funciones auxiliares necesarias para implementar los mecanismos propuestos da una idea detallada de las limitaciones del modelo básico de hipermedia y cómo debe extendérselo para proveer búsquedas y visiones parciales del hipergrafo.

El prototipo IMHEF está constituído por aproximadamente 70 nodos y 120 vínculos que relacionan dichos nodos. En IMHEF sólo se desarrolló el nivel coloquial, que implementa el aprendizaje de los conceptos básicos de redes en un lenguaje no técnico y usando ejemplos y animaciones. Este prototipo abarca los conceptos introductorios sobre redes de computadoras: explica qué son las redes en general y las redes académicas en particular, presenta una clasificación de las redes según su alcance y muestra una breve reseña histórica de cómo surgieron las redes. Además, realiza una descripción de los servicios ofrecidos por las redes: correo electrónico, diálogo interactivo, acceso a bases de datos remotas, intercambio de archivos, etc, explicando para qué sirven y cómo funcionan cada uno de ellos.

CONCLUSIONES

El trabajo de grado descripto abrió dos líneas de trabajo vinculadas pero con aristas independientes. Por un lado el modelo MHEF tiene un conjunto de características que resuelven una serie de problemas de búsqueda y exploración de grandes volúmenes de información con posibilidades de mayor ampliación tanto en lo que hace a incorporar modelos de usuario, mayores posibilidades de consulta de las estadísticas de uso, técnicas de evaluación y asistencia en los procesos de autoría y monitoreo del sistema resultante.

El modelo ha sido analizado y evaluado en sus características específicas con resultados superadores respecto de una hipermedia, con los mecanismos básicos de visualización y navegación.

Por otro lado, el prototipo IMHEF no sólo sirvió para testear la factibilidad del modelo antes mencionado, sino que además permitió construir una herramienta de utilidad en el dominio de las redes.

De la implementación realizada surge una línea de trabajo completando el sistema a través de la carga de más información (en los niveles de servicios y de detalle técnico) y distribuyendo el mismo para su utilización masiva para consulta.

La IMHEF a su vez ha sido evaluada respecto de criterios de diseño, tiempos de respuesta y aceptabilidad de la interfaz del usuario, siendo muy favorable la aceptación del mismo.

REFERENCIAS

[Aho83] A. Aho, J. Hopcroft, J. Ullman. *Data Structure and Algorithms*. Addison-Wesley Publishing Company, 1983.

[Asym89] Asymetric Corporation. *Using Toolbook. A Guide to Building and Working with Books* y *Using OpenScript*. 1989.

[Bego90] J. A. Begoray. *An Introduction to Hypermedia Issues, Systems and Application Areas*. Int. J. Man-Machine Studies, Vol. 33, pp.121-147, 1990.

[Belk92] N. Belkin, W. Croft. *Information Filtering and Information Retrieval: Two Sides of the Same Coin?*. Communications of ACM, Vol. 35, No. 12, December, 1992.

[Bern88] M. Bernstein. *The Bookmark and the Compass: Orientation Tools for Hypertext Users*. ACM SIGOIS Bull, Vol. 9, No. 4, 1988.

[Camp88] B. Campbell, J. Goodman. *HAM: A general purpose Hypertext Abstract Machine*. Communications of the ACM 31, pp.856-861, July, 1988.

[Coll87] G. Collier. *Thoth II: Hypertext with explicit semantics*. Proceedings of Hypertext '87 Conference. Chapel Hill, NC: University of North Carolina, Computer Science Department, 1987.

[Cond90] C. Condon and Yale Computer Center. *The BITNET Services Library. BITNET USERHELP*. October, 1990.

[Conk87] J. Conklin. *Hypertext: An introduction and survey*. IEEE, Computer 20, 9, 1987.

[Díaz93] J. Díaz, C. Queiruga, M. A. Schiavoni. *An Experience: Extending the hypermedia model for training proposes*. Proceedings of MCAT'93 (3rd Australian Multi-Media Communications, Applications and Technology Workshop), Australia, July, 1993.

[Díaz94a] J. Díaz, C. Queiruga, M. A. Schiavoni. *A proposed extension to the hypermedia model for training in networking technology*. Proceedings of BIWIT'94 (Basque International Workshop on Information Technology), Biarritz, Francia, February,1994.

[Díaz94b] J. Díaz, C. Queiruga, M. A. Schiavoni. *Uso empírico de herramientas de hipermedia para la organización de dominios complejos*. Memorias de CLEI - PANEL '94 (XX Conferencia Latinoamericana de Informática), México, Septiembre, 1994.

[Díaz95] J. Díaz, C. Queiruga, M. A. Schiavoni. *MHEF: Modelo Hipertextual con Estadísticas y Filtros*. Memorias de las III Jornadas Chilenas de Computación, XI Conferencia Internacional de la SCCC (Sociedad Chilena de Ciencias de la Computación), Arica, Chile, Noviembre,1995.

[Egan89] D. E. Egan, J. R. Remde, et al. *Formative Design-Evaluation of SuperBook*. ACM Trans. On Information Systems, Vol. 7, No. 1, pp.30-57, 1989.

[Frei92] H. Frei, D. Stieger. *Making Use of Hypertext Links when Retrieving Information*. Proceedings of Echt '92, Fourth ACM Conference on Hypertext, Milano, Italy, pp.102-111, November, 1992.

[Garz91] F. Garzotto, P. Paolini, D. Schwabe, M. Bernstein. *Tools for Designing Hyperdocuments*. Hypertext/Hypermedia Handbook, Emily Berk and Joseph Devlin (Eds.), McGraw - Hill, New York, N.Y., 1991.

[Gitti86] D. Gittins. *Icon-based Human-computer Interaction*. International Journal for Man-Machine Studies 24, pp.519-543, 1986.

[Gome92] L.F. Gomes Soares, L. Tucherman. *Fundamentos de Sistemas Multimídia*. VIII Escola de Computaçao, Gramado, RS, Brasil, 3-12 Agosto, 1992.

[Guin92] C. Guinan, A. Smeaton. *Information Retrieval from Hypertext Using Dynamically Planned Guided Tours*. Proceedings of Echt '92, Fourth ACM Conference on Hypertext, Milano, Italy, pp.122-130, November, 1992.

[Hala87a] F. Halasz, T. Morgan, R. Trigg. *Notecards in a nutshell*. Proceedings of the ACM Conference on Human Factors in Computing Systems, Toronto, Canadá, April, 1987.

[Hala87b] F. Halasz. *Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems*. Proceedings of Hypertext '87, 1987.

[Hala94] F. Halasz, M. Schwartz. *The Dexter Hypertext Reference Model*. K. Grønbæk, R. Trigg, Eds., Communications ACM, Vol. 37, No. 2, pp.40-49, February, 1994.

[Hamm87] N. Hammond, L. Allinson. *The Travel Metaphor as Design Principle and Training Aid for Navigating around Complex Systems*. Proceedings of the 3rd. Conference of the British Computer Society Human-Computer Interaction Specialist Group, pp.75-90, 1987.

[Hara93] I. Harari. *Informe de Beca de Estudio aprobado por la Comisión de Investigaciones Científicas (CIC)*. 1993.

[Ichi93a] S. Ichimura, Y. Matsushita. *A PilotCard-Based Shared Hypermedia System Supporting Shared and Private Databases*. Proceedings ACM COOCS, November 1993.

[Ichi93b] S. Ichimura, Y. Matsushita. *Another dimension to Hypermedia Access*. Proceedings of Hypertext '93 Conference, Seattle, Washington, pp.14-18, November, 1993.

[Jona90] D. Jonassen, R. Grabinger. *Problems and Issues in Designing Hypertext/Hypermedia for Learning*. Designing Hypermedia for Learning, D. H. Jonassen, H. Mandl (Eds.), Springer - Verlag, Berlin, 1990.

[Mari90] G. Mariani, G. Sagasti, A. Jacobs. *Trabajo realizado para la materia Proyecto de Software de la Licenciatura en Informática*. 1990.

[Mars89] C. Marshall, P. Irish. *Guided Tours and on-line presentations: How authors make existing hypertext intelligible for readers*. Proceedings of Hypertext '89, 1989.

[Nels67] T. H. Nelson. *Getting it out of our System*. Information Retrieval: a Critical review, G. Schechter, de. Thompson Books, Washington, D.C., 1967.

[Niel90] J. Nielsen. *Hypertext and Hypermedia*. Academic Press, Inc., New York, 1990.

[Norm86] K. L. Norman, L. Weldon, B. Shneiderman. *Cognitive layouts of windows and multiple screens for user interfaces*. International Journal of Man-Machine Studies 25, pp.229-248, 1986.

[Norm91] K. Norman. *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface*. Ablex, Norwood, NJ, 1991.

[Paru89] H. V. D. Parunak. *Hypermedia Topologies and User Navigation*. Proceedings ACM Hypertext '89, 1989.

[Quei94] C. Queiruga. *Informe de Beca de Perfeccionamiento aprobado por la Comisión de Investigaciones Científicas (CIC)*. 1994.

[Rivl94] E. Rivlin, R. Botafogo, B. Shneiderman. *Navigating in Hyperspace: Designing a structure-based Toolbox*. Communications of the ACM, Vol. 37, No. 2, pp.87-96, February, 1994.

[Roge89] Y. Rogers. *Icons at the Interface: Their usefulness*. Interacting with Computers 1, 1, pp.105-117, 1989.

[Salt89] G. Salton. *Automatic Information Retrieval: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[Save93] K. Savetz. *Internet Services Frequently Asked Questions and Answers*. 1993.

[Schi94] M. A. Schiavoni. *Enciclopedia Hipermedial de redes de datos*. Memorias de las 2das Jornadas de Investigación del Grupo Montevideo, Salto Grande, Septiembre, 1994.

[Seye91] P. Seyer. *Understanding Hypertext Concepts and Applications*. Windcrest Books McGraw - Hill, 1991.

[Shne86] B. Shneiderman, P. Shafer, R. Simon, L. Weldon. *Display strategies for program browsing: Concepts and experiment.* IEEE Software 3, 3, pp.7-15, May 1986.

[Shne92] B. Shneiderman. *Designing the User Interface. Strategies for Effective Human-Computer Interaction.* Second Edition, Addison-Wesley Publishing Company, 1992.

[Simp90] A. Simpson. *Lost in Hyperspace: how can designers help?.* Intelligent Tutoring Media, Vol. 1, No. 1, 1990.

[Soko88] T. Sokolnicki. *Intelligent Tutoring System -Craft or Technology-.* Linkoping University, Tech Rep. LiTH-IDA-R-88-33, 1988.

[Tane91] A. S. Tanenbaum. *Redes de Ordenadores.* Segunda Edición, Prentice-Hall Hispanoamericana, S.A.

[Trig86] R. Trigg, L. Suchman, F. Halasz. *Supporting collaborations in NoteCards.* Proceedings of the Conference on Computer-Supported Cooperative Work, 1986.

[Trigg88] R. Trigg. *Guided tours and Tabletops: Tools for communicating in hypertext environment.* ACM Transactions on Office Information Systems, 1988.

[Utti89] K. Utting, N. Yankelovich. *Context and Orientation in Hypermedia Networks.* ACM Trans. On Information Systems, Vol. 7, No. 1, pp.58-84, 1989.

[Wate91] J. A. Waterworth, M. H. Chignell. *A Model for Information Exploration.* Hypermedia, 3(1), pp.35-58, 1991.

[Weis92] M. Weiss. *Data Structures and Algoritm Analysis.* The Benjamin/Cummings Publishing Company, Inc., 1992.

[Yank88] N. Yankelovich, B. Haan, N. Meyrowitz, S. Drucker. *Intermedia: The Concept and Construction of a Seamless Information Environment.* IEEE Computer, 21, pp.91-96, 1988.

[Zell91] P. Zellweger. *Structure, Navigation and Hypertext: The status of the navigation problem.* Proceedings of Hypertext '91 Conference, San Antonio, Texas, 15-18 December, 1991.

INDICE DE RFC (REQUEST FOR COMMENTS)

RFC #661 J. Postel. *Protocol information.* 23/11/1974.

RFC #822 D. Crocker. *Standard for the Format of ARPA Internet Text Messages.* 13/08/1982.

RFC #934 M. Rose, E. Stefferud. *Proposed standard for message encapsulation.* 01/01/1985.

RFC #959 J. Postel, J. Reynolds. *File Transfer Protocol.* 01/10/1985.

RFC #1049 M. Sirbu. *Content-type header field for Internet messages.* 01/03/1988.

RFC #1207 G. Malkin, A. Marine, J. Reynolds. *Answers to Commonly asked "Experienced Internet User" Questions.* 28/02/1991.

RFC #1325 G. Malkin, A. Marine. *FYI on Questions and Answers. Answers to Commonly asked "New Internet User" Questions.* 15/05/1992 (FYI 4).

RFC #1341 N. Borenstein, N. Freed. *MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies.* 11/06/1992.

RFC #1462 E. Krol, E. Hoffman. *FYI on "What is the Internet?".*27/05/1993.

RFC #1463 E. Hoffman, L. Jackson. *FYI on Introducing the Internet--A Short Bibliography of Introductory Internetworking Readings for the Network Novice.* 27/05/1993.

RFC #1594 A. Marine, J. Reynolds, G. Malkin. *FYI on Questions and Answer Answers to Commonly asked "New Internet User" Questions.* 11/03/1994.

RFC #1780 J. Postel. *INTERNET OFFICIAL PROTOCOL STANDARDS.* 28/03/1995.

RFC #1792 T. Sung. *TCP/IPX Connection Mib Specification.* 18/04/1995.

RFC #1805 A. Rubin. *Location-Independent Data/Software Integrity Protocol.* 07/06/1995.