Soporte Hipermedia en la Investigación Conceptual

Carlos Alberto Ballesteros

Director: Dr. Gustavo Rossi

Tesis presentada al Departamento de Informática de la Universidad Nacional de La Plata, como parte de los requisitos para la obtención del título de Magíster en Ingeniería de Software

Índice

Agradecimientos

"Muchas veces me doy cuenta que mi propia vida y sus logros se han construido gracias al trabajo de las personas que me rodean. También comprendo, cuanto debo esforzarme para darles, en correspondencia, tanto como he recibido."
Albert Einstein

Es imposible establecer un orden o una medida para la gratitud, como así también lo es agradecer a todas las personas meritorias que indirectamente han ayudado a la concreción de mi trabajo. Solamente quiero pedirles disculpas a quienes aquí estén ausentes, a los que espero les alcance con mis "¡Gracias!" en forma personal y con la eterna gratitud de mi corazón.

Quiero agradecer a mi director de Tesis, el Dr. Gustavo Rossi, en primer lugar por haberme involucrado en el tema de investigación, y en segundo lugar por su experiencia y fundamentalmente su paciencia, para guiarme a través del trabajo. Del mismo modo quiero agradecer a mi colega y amigo el Lic. Guillermo Lafuente, por su interés en mi trabajo y su ayuda en la etapa final del mismo.

Quiero agradecer además a las autoridades de la Facultad de Ingeniería de la Universidad Nacional de La Pampa, en especial al Mg. Heman Prieto, quien con su paciencia y comprensión ha permitido que trabaje en el clima necesario como para poder concretar una tarea que demandó de mi mayor esfuerzo.

En forma especial quiero agradecer al Mg. Hugo Berti, quien con su ayuda invaluable y desinteresada se convirtió en mi co-director, consejero y amigo. Más aún, quiero dar gracias a Dios por haberme cruzado en tiempo y espacio con él, ya que su escala de valores y sentido humano lo convierte en una persona que ayuda a creer que un mundo diferente es posible, un mundo en el cual la solidaridad y el amor al prójimo estén por encima de las mezquindades.

Para finalizar quiero agradecer nuevamente a Dios, quien en cada camino oscuro puso una estrella que me guió hacia la salida, y a mi familia por su amor y contención, en especial a mi hija Carolina y a mi esposa Cecilia, quienes sin más remedio tuvieron que soportarme (aunque debo admitir que "heroicamente"), y muy en especial a mi hijo Andrés quien con su reiterada preocupación reflejada en la pregunta "¿Papá te falta mucho para la Tesis?", redoblaba mi esfuerzo y compromiso con el trabajo.

Resumen

En este trabajo se identifican las necesidades de un grupo de investigadores que son difíciles (algunas imposibles) de cubrir con la tecnología y herramientas actuales. Se trata de un grupo que coopera en la etapa de investigación conceptual, en la cual enfrentan la tarea de manejarse con una gran cantidad de material compartido e interrelacionado, posiblemente generado por distintas aplicaciones e incluso almacenado sobre distintas plataformas.

En esta etapa los investigadores necesitan representar y compartir distintos tipos de relaciones, las cuales constituyen en sí conocimiento agregado acerca de la interpretación del material en estudio y además proveen las conexiones para permitir consultas eficientes. Por tal motivo se realiza un análisis detallado de las actividades desarrolladas dentro del grupo y se identifican las necesidades individuales y grupales que deben ser satisfechas, así como también las características técnicas que un soporte de software debiera proveer.

Del análisis de las actividades del grupo se determina además la relación existente entre sus necesidades y una de las principales metas de la Hipermedia: almacenar y consultar eficientemente el conocimiento acumulado. La clave para que el material pueda ser consultado eficientemente está en la posibilidad de contar con las relaciones apropiadas que conecten dicho material. En la convicción de que la Hipermedia es la mejor forma de abordar el soporte para el grupo de investigación presentado, y que ésta constituye en general una forma natural para representar y compartir conocimiento, se realiza un estudio profundo de los trabajos de investigación y desarrollo de la comunidad Hipermedia.

Con el propósito de obtener una mejor interpretación del estado actual del arte, se realiza una reseña histórica de la Hipermedia que incluye las principales contribuciones. Dentro de la reseña se presenta y discute la evolución de los Sistemas, incluyendo el representante actual que más popularizó la Hipermedia, la Web. Dentro de esta discusión se destacan las virtudes y falencias de los distintos sistemas a la hora de soportar la representación y comunicación de conocimiento requerida en el grupo de investigación presentado.

Del estudio de los Sistemas Hipermedia más relevantes y de las cuestiones pendientes de investigación surge que: la Web, los Sistemas Hipermedia y las aplicaciones de escritorio no brindan un ambiente propicio para satisfacer las necesidades identificadas en el grupo, por lo que los investigadores se ven obligados a utilizar herramientas alternativas y complementarias para intentar cubrirlas. Muchas veces estas alternativas entorpecen la representación de las relaciones y obligan a duplicar material, con los consabidos problemas de consistencia y mantenimiento que esto provoca. Por lo tanto se justifica la

necesidad de desarrollar un nuevo sistema que cubra las falencias de funcionalidad de las herramientas actuales.

Con el propósito de abordar el soporte en forma rigurosa, se realiza una investigación de los trabajos más relevantes acerca de la conceptualización subyacente en el diseño de Sistemas Hipermedia, y de la infraestructura arquitectural necesaria para soportarlos. Es así que se discute en primer lugar el Modelo de Referencia de Hipertexto de Dexter, de Halasz y Schwartz de 1990, considerado por este trabajo como una de las mayores contribuciones en lo que a conceptualización se refiere. En esta discusión se describen las conclusiones a las que arribó el grupo que desarrolló el modelo, las dificultades de los desarrolladores Hipermedia para cumplir con dicho modelo, y las conclusiones arribadas en este trabajo.

Una vez finalizada la discusión del Modelo de Dexter, y en base al análisis de los trabajos de investigación y desarrollo más relevantes, se describe la operación básica de los Sistemas de Hipermedia Abierta (concepción actual de la Hipermedia) y la Arquitectura conceptual necesaria para soportarlos. Luego se realiza una descripción detallada de las entidades conceptuales subyacentes en el diseño de dichos Sistemas. Esta descripción incluye la conceptualización básica para el almacenamiento y el manejo del tiempo de ejecución, e involucra además detalles de la evolución de las entidades conceptuales, con las distintas interpretaciones y funcionalidades otorgadas a cada una. También se describen las consideraciones básicas para la navegación y la presentación de relaciones.

Este trabajo cierra con la propuesta de un Sistema de Hipermedia Abierta, que aprovecha las principales contribuciones analizadas, e incorpora nuevas funcionalidades a las herramientas favoritas de los investigadores, para cubrir las necesidades identificadas. La propuesta consta de una memoria descriptiva del Sistema, una arquitectura para soportar dicho sistema y un diseño orientado a objetos que contiene: el modelo del dominio, los requerimientos funcionales y no funcionales, la descripción de los casos de uso y el diseño detallado del Servicio Hipermedia. Se incluye además una operación de travesía que soporta comportamientos avanzados de navegación.

Resulta necesario destacar que la propuesta de este trabajo establece además bases de diseño útiles para proveer soporte en ámbitos de representación y manejo de conocimiento en general, por lo que sirve de plataforma para futuros trabajos de investigación y desarrollo orientados en esta dirección.

Capítulo 1

Presentación e Introducción

> *"There is a growing mountain of research results; the investigator is bombarded with the findings and conclusions of thousands of parallel workers which he cannot find time to grasp as they appear, let alone remember; specialization becomes increasingly necessary for genuine progress, and effort to bridge between disciplines correspondingly superficial. Still we adhere rather closely, in our professional efforts, to methods of revealing, transmitting, and reviewing results which are ... now inadequate for their purpose."*
>
> Vannevar Bush, 1945 [3]

Resumen

En este capítulo se presentan las actividades llevadas a cabo en un ambiente de investigación conceptual que vislumbraron la necesidad de un soporte tecnológico adecuado y motivaron la realización de este trabajo. Luego se realiza una introducción general a las principales áreas involucradas en la estrategia para brindar el soporte al ambiente presentado, evidenciando como en dicho soporte encaja en la Hipermedia Abierta. Las áreas involucradas se presentan, partiendo desde la visión general de la comunicación digital, para luego definir e introducir las áreas específicamente implicadas: Hipertexto, Hipermedia y su visión actual, la Hipermedia Abierta. Para finalizar se explicita el objetivo de este trabajo y se describe la estructura de la tesis.

1.1 Investigación Conceptual

Hay profundas diferencias entre lo que hoy es el proceso de investigación y lo que era dicho proceso una década atrás. La investigación que alguna vez fue una larga y solitaria actividad conducida por investigadores independientes, en la actualidad requiere del trabajo de grupos interdisciplinarios de investigadores que aborden la creciente complejidad de los problemas.

La comunicación digital ha permitido entre otras cosas que estos grupos de investigadores, distribuidos geográficamente, puedan trabajar en forma cooperativa en la generación de nuevo conocimiento. Dentro del proceso de investigación, y previa a la etapa de investigación empírica, se encuentra la etapa de investigación conceptual. La investigación conceptual es una búsqueda a nivel cognitivo en escritos científicos de actualidad dentro de un área de conocimiento determinada.

Durante esta etapa de investigación conceptual los investigadores enfrentan la difícil tarea de manejarse con una gran cantidad de trabajos escritos (material en formato electrónico) en los que se basa su propio trabajo. Estos escritos corresponden a trabajos propios, trabajos del grupo de investigación al que pertenecen y trabajos de terceros. Sobre este material digital, los investigadores realizan lo que se denomina lectura activa, es decir, a medida que van leyendo van realizando comentarios o anotaciones sobre el mismo y van creando vínculos o relaciones entre escritos asociados. Tanto los comentarios como las relaciones creadas por cada investigador, conforman metadatos útiles para los demás investigadores.

Si además se considera que el material está compuesto por distintos tipos de documentos (texto, planillas de cálculo, gráficos, etc.), generados por distintas aplicaciones que podrían correr sobre distintas plataformas, y que el trabajo en grupo demanda la necesidad de compartir e intercambiar (material, anotaciones y relaciones), así como también realizar escrituras conjuntas, surge una demanda de soporte tecnológico que brinde un ambiente propicio de trabajo.

Luego de un estudio de los objetivos que ha tenido la Hipermedia, partiendo de los trabajos de sus pioneros (Vannevar Bush, 1945 [3]), se revela la relación existente entre estos objetivos y las necesidades del ambiente de investigación planteado, esto es, ambos tienen que ver con el manejo de conocimiento. La técnica fundamental de Hipermedia, crear links asociativos entre material en línea, se vislumbró como la base del soporte para el ambiente de investigación presentado.

Se realiza a continuación, una descripción general de las principales áreas estratégicas involucradas en la búsqueda de soporte al ambiente de investigación presentado. Dicha descripción tiene como objetivo evidenciar la relación entre el soporte tecnológico necesario y las áreas seleccionadas para

brindarlo. Partiendo desde el área general involucrada, la comunicación digital, luego se introducen el Hipertexto y la Hipermedia, para finalmente introducir el área estratégica específica seleccionada, la Hipermedia Abierta.

1.2 Comunicación digital

La comunicación digital constituye uno de los campos de estudio y práctica más excitantes y de más rápida expansión en el mundo, solo basta observar el incremento de sitios Web y de usuarios de Internet, como así también la publicación y utilización de discos compactos multimedia en colegios, hogares, universidades y empresas.

El lenguaje y los conceptos de la vida digital son hoy centrales en la cultura popular. En el ciberespacio los roles de escritor y lector ya no son estáticos, sino dinámicos; el concepto de texto ya no es fijo sino fluido. La tecnología ha entregado una herramienta potente para la creación, presentación e intercambio de texto, imágenes, video y audio. Esta tecnología permite ambientes de información que integran los distintos tipos de medios en forma transparente.

Se está presenciando una revolución en la comunicación y en el aprendizaje, de la misma magnitud que la que hubo con posterioridad a la imprenta de Gutenberg. Este cambio revolucionario en la forma de expresarse, comunicarse y hasta de pensar, obliga a buscar herramientas, modelos y soporte en general que saquen provecho de su potencial.

1.3 Hipertexto e Hipermedia

> *It is tempting to describe the essence of hypertext as its ability to perform high speed branching transactions on textual chunks. But this is a little like describing the essence of a great meal by listing its ingredients. Perhaps a better description would focus on hypertext as a computer based medium for thinking and communication.*
>
> Conklin 1987, p. 32 [1]

Aunque el concepto de Hipertexto está omnipresente en la Web, es difícil encontrar una única y clara definición. Se presentan, a continuación, algunas definiciones que permiten apreciar el concepto desde distintas perspectivas, y las distintas concepciones que ha tenido a través de su historia, de las cuales surgen evidencias para su utilización en la búsqueda del soporte requerido:

- Ted Nelson (creador del término Hipertexto), 1965 [2]: Colección de documentos que contienen referencias cruzadas, las cuales, con la ayuda de un programa interactivo, permiten al lector moverse en forma sencilla desde un documento a otro.

- Real Academia Española: "Texto que contiene elementos a partir de los cuales se puede acceder a otra información".

- Diccionario Oxford: *"Text which does not form a single sequence and which may be read in various orders; specially text and graphics ... which are interconnected in such a way that a reader of the material (as displayed at a computer terminal) can discontinue reading one document at certain points in order to consult other related matter. "*

Lowe and Hall en el Capítulo dos del libro "Hypermedia & the Web" proveen la definición de Hipermedia y características específicas:

- Es una base de datos que tiene referencias cruzadas interactivas y que permiten el salto.

- Denota un medio de información que vincula información verbal y no verbal.

- Puede ser almacenado, leído, buscado y modificado y contiene conexiones a otros documentos.

Además, en la página 32, Lowe and may, dan la siguiente definición de Hipermedia:

> *"An application which uses associative relationship among information contained within multiple media data for the purpose of facilitating access to, and manipulation of, the information encapsulated by the data".*

Dado que se están presentando definiciones de palabras, vale aclarar, que Hipermedia es la evolución del Hipertexto, Hipertexto se refería a relacionar elementos textuales, mientras que la Hipermedia permite relacionar cualquier tipo de medio (sonido, imagen, video, etc.).

La convergencia de Hipermedia y las telecomunicaciones hicieron posible el nacimiento de la Web: información multimedia que puede ser interconectada en forma distribuida, de modo que diferentes piezas de información puedan estar en diferentes computadores distribuidas por todo el mundo.

Se podría pensar que a través de la Web los Sistemas Hipermedia han alcanzado la cima, sin embargo, los conceptos básicos de Hipermedia resultantes de las investigaciones llevadas a cabo desde antes del

advenimiento de la Web, y con posterioridad al mismo, tienen mucho más para ofrecer a la comunicación digital y al manejo de conocimiento.

En la actualidad el término Hipermedia está directamente relacionado a la Hipermedia Abierta. En forma indistinta se habla de Hipermedia o de Hipermedia Abierta debido a que en la concepción actual son inseparables.

1.4 Hipermedia Abierta

La palabra "abierta" tiene diferentes significados en diferentes contextos. En general, un sistema de software es catalogado como abierto si especifica y publica interfaces entre sus componentes. Vale aclarar que, dadas las distintas concepciones que han tenido conceptos como ancla, link y seguimiento a través de la evolución de la Hipermedia, en esta sección se presentan algunas definiciones que podrían resultar básicas, sin embargo, se considera necesario dicha presentación a fin de permitir apreciar las distintas concepciones y de interpretar las concepciones adoptadas por la Hipermedia Abierta.

Antes de describir a qué hace referencia el término "abierta" en el contexto de la Hipermedia, es necesario introducir los conceptos contenido y estructura. Se denomina contenido a los datos representados sobre distinto tipo de medios (gráficos, sonidos, videos), en general al texto. Y se denomina estructura a las distintas relaciones creadas entre fragmentos de contenido, el ejemplo más típico de estructura es el link.

En el contexto de Sistemas Hipermedia, un Sistema Hipermedia que impone un formato especifico para el contenido y la estructura es considerada cerrado, ya que las aplicaciones tienen que respetar dichos formatos para poder acceder a la Hipermedia. Un Sistema de Hipermedia Abierta, en contraste, especifica formato sólo para la estructura.

A fin de clarificar la distinción entre Sistemas Abiertos y Cerrados, en el contexto de Hipermedia, se presentan a continuación dos operaciones de seguimiento de links, una dentro del contexto de un Sistema Cerrado, como la Web y otra en el contexto de Sistemas Abiertos.

La Web es un Sistema Cerrado, ya que impone a sus navegadores el formato HTML para el contenido (HTML involucra contenido y estructura ya que los links están incrustados en el contenido). Se describe a continuación la operación de travesía[1] (seguimiento) de un link en un Sistema cerrado como lo es la Web:

[1] Travesía: al seguimiento de un link en Hipermedia se le denomina travesía, ya que se utiliza la frase "atravesar un link" en lugar de "seguir un link". En el capitulo 5 se hace referencia al motivo de tal distinción.

1. El usuario hace clic sobre un ancla[2] en un archivo HTML mostrado en un navegador. La URL, del otro extremo del link, está incrustada en el archivo HTML mostrado.
2. El navegador, basado en la información de la URL, envía un requerimiento *get* al servidor Web que mantiene el archivo en el otro extremo del link.
3. El servidor Web envía el archivo al navegador.
4. El navegador, basado en el tipo de archivo devuelto por el servidor Web, hará una de las siguiente tareas:
    a. Si se trata de un archivo HTML: el contenido del archivo será interpretado y mostrado con sus links salientes, si los tuviera. El usuario puede seguir estos links a otros archivos.
    b. Si se trata de otro tipo de archivo: el navegador puede mostrar algunos tipos de archivos, pero esos tipos de archivo no tienen link salientes (solo los archivos HTML mantienen información de links). Si el navegador no puede mostrar el archivo disparara un *plug in* o una aplicación separada que lo pueda mostrar. Estas aplicaciones no están habilitadas para soportar links, por lo que en este caso, el usuario ha alcanzado un punto muerto en la Hipermedia (sin links para poder seguir).

Desde la perspectiva de sistemas de software en general, la Web es abierta ya que la interfaz entre navegador y servidor está bien definida (protocolo HTTP). Sin embargo, desde la perspectiva de la Hipermedia, el navegador debe construirse para interpretar y mostrar archivos HTML.

Para integrar una aplicación existente, que opere como un navegador, la aplicación debería modificarse para utilizar HTML como su formato básico de modelo de dato. Esto requeriría en la mayoría de sus casos una reescritura de dicha aplicación. Por lo tanto desde esta perspectiva, la Web es un Sistema Hipermedia cerrado.

En general, un Sistema Hipermedia cerrado, esta caracterizado por manejar sólo un conjunto cerrado de formatos de modelo de dato y por tener un conjunto cerrado de aplicaciones habilitadas para Hipermedia que participen en forma activa. Una aplicación que es disparada sólo para mostrar un archivo, no puede ser considerara una aplicación habilitada para Hipermedia, ya que esta aplicación no es "consciente" del Sistema Hipermedia que la disparó.

Permitir que las aplicaciones de un Sistema de Hipermedia carguen contenidos en diferentes formatos, potencialmente desde afuera del Sistema Hipermedia, es un requerimiento básico en la Hipermedia Abierta. De este modo se pueden integrar y utilizar aplicaciones existentes. A continuación se describe la operación conceptual general de travesía en un ambiente de Hipermedia Abierta:

---

[2] Ancla: en el caso más simple un ancla es el origen o el destino de un link. Un link típico une dos anclas, el ancla origen y el ancla destino.

1. Una aplicación, podría ser la favorita del usuario, comunica el requerimiento de atravesar un link al Sistema de Hipermedia Abierta (SHA) debido a alguna acción, la que puede ser disparada por un clic del usuario en un ancla, o por algún otro evento que tomó lugar en la aplicación.
2. El SHA resuelve el link y determina el o los archivos (los SHA soportan links n arios, vinculan múltiples documentos) del otro extremo del link. En caso de un link con múltiples destinos, el paso 3 es repetido para cada archivo del otro extremo del link.
3. El SHA puede hacer una de las siguientes tareas para mostrar el archivo:
    a. Requerir que una aplicación habilitada para Hipermedia que estaba corriendo muestre el archivo.
    b. Disparar una aplicación habilitada para Hipermedia que muestre el archivo.
    c. Disparar, en caso que ninguna aplicación habilitada para Hipermedia pueda mostrar el tipo de archivo dado, una aplicación no habilitada para Hipermedia que lo muestre. En este caso, el usuario ha alcanzado un punto muerto, sin links disponibles. De ser así los puntos 4 a 8 no se ejecutan para este archivo. (En general nuevos formatos de modelo de dato pueden ser soportados en un ambiente de Hipermedia Abierta, habilitando una aplicación existente que pueda manejar el formato requerido).
4. Basado en la información obtenida del SHA, la aplicación habilitada para Hipermedia, abre el archivo requerido.
5. La aplicación habilitada para Hipermedia envía un mensaje al SHA, requiriendo las anclas del archivo mostrado.
6. El SHA responde con una lista de anclas.
7. La aplicación habilitada para Hipermedia muestra las anclas y resalta el ancla particular que fue conectada por el link atravesado.
8. El usuario puede seguir los links disponibles desde el nuevo archivo mostrado.

Los SHA almacenan la estructura separada de los contenidos. La estructura es superimpuesta sobre el contenido, en el momento en que éste se muestra, y la aplicación habilitada para Hipermedia continúa manejando el contenido. Por lo tanto, el conjunto de aplicaciones que forman el ambiente de Hipermedia Abierta pueden vincular desde y hacia documentos en línea de distintos tipos de fuentes (incluyendo fuentes de solo lectura).

Por lo expuesto, en un Sistema de Hipermedia Abierta:

- Los usuarios pueden crear sus propias estructuras sobre cualquier tipo de medio, incluso sobre material de solo lectura.
- Las estructuras se almacenan en bases de dato separadas respecto del contenido.

- Múltiples estructuras pueden convivir, formando capas sobre el mismo grupo de documentos.
- Los usuarios acceden a la funcionalidad Hipermedia sin abandonar sus aplicaciones favoritas.

1.5 Objetivo del trabajo

Es necesario y factible contar con un soporte tecnológico que permita crear un espacio de trabajo ideal para el ambiente de investigación planteado en la sección 1.1, de modo que las necesidades involucradas sean satisfechas en forma sencilla y efectiva.

La comunicación digital a través de la Web ha abierto posibilidades de trabajo cooperativo entre usuarios distribuidos geográficamente, con material también distribuido, almacenado y manejado por distintas aplicaciones, sobre distintas plataformas. Dentro de este ambiente de comunicación surgen necesidades que, ni la Web, ni los Sistemas Hipermedia tradicionales pueden cubrir. Son pocos los usuarios que disfrutan de soporte Hipermedia integrado dentro de su ambiente de trabajo, es decir, en sus editores y herramientas favoritas. Trabajar con una buena funcionalidad Hipermedia que provea el soporte necesario al ambiente planteado, requiere un conjunto separado de herramientas.

Un soporte tecnológico adecuado debería aumentar la funcionalidad de las aplicaciones favoritas de los investigadores con soporte Hipermedia avanzado, a fin de contar con un ambiente más propicio para el proceso de investigación. Hay mucho camino construido en la integración de material, aplicaciones y plataformas dentro del campo de la Hipermedia Abierta, este trabajo aprovecha ese camino construido, utilizando las investigaciones más relevantes del área, en la búsqueda de una solución específica.

En un convencimiento de la potencia que tiene el enfoque de Sistemas de Hipermedia Abierta para integrar aplicaciones a través de redes, plataformas y ambientes heterogéneos, el objetivo de este trabajo es diseñar un Sistema de Hipermedia Abierta que considere las investigaciones más relevantes en el área y brinde un soporte que cubra las necesidades de los investigadores dentro del ambiente de investigación conceptual cooperativa planteado en la sección 1.1. Además dicho soporte, debe estar integrado al ambiente de trabajo de cada investigador, es decir, respetando sus editores, aplicaciones y herramientas favoritas.

1.6 Estructura de la tesis

Luego de este capítulo en el que se realizó la presentación de un ambiente de investigación que motivó la búsqueda de un soporte tecnológico, se describieron las áreas involucradas en la estrategia para abordar una solución y se explicitó el objetivo de este trabajo, en el capitulo 2 se describen en forma detallada las necesidades de un grupo de investigadores que trabajan

en forma cooperativa, y el conjunto de características técnicas que debería poseer un Sistema Hipermedia que cubra dichas necesidades.

En el capítulo 3 se describe la evolución histórica de la Hipermedia, a través de un relevamiento de los Sistemas Hipermedia más destacados que podrían servir de soporte al ambiente planteado. Éste capitulo cierra con las conclusiones acerca de los sistemas analizados y con la justificación de la búsqueda de un nuevo sistema, motivo de este trabajo, que brinde el soporte requerido.

El capítulo 4 describe la base de los fundamentos utilizados en el diseño del nuevo sistema, el Modelo de Referencia de **Dexter**. Dado el impacto de dicho modelo tuvo en la Hipermedia actual, se dedica al mismo un capítulo especial, en el cual se presentan además las conclusiones del modelo y se describen las cuestiones pendientes.

En el capítulo 5 se describen los fundamentos de la Hipermedia Abierta sobre los cuales esta basada la propuesta de este trabajo. En particular se realiza una descripción del funcionamiento básico de los sistemas Hipermedia Abierta, de la Arquitectura conceptual que utilizan y, en forma detallada, las entidades conceptuales involucradas en los SHA.

En el capítulo 6 se describe la propuesta de este trabajo que consta de: una memoria descriptiva del sistema, la arquitectura requerida para soportarlo, el método de desarrollo adoptado y el diseño orientado a objetos del sistema propuesto.

Para finalizar en el capítulo 7 se presentan las conclusiones finales obtenidas y se plantean los trabajos futuros previstos.

Capítulo 2

Requerimientos en un ambiente de Investigación

Resumen

En este capítulo se presenta el ambiente de trabajo de un grupo de investigadores durante la etapa de investigación conceptual junto con algunos escenarios que resaltan sus necesidades individuales y grupales. Luego estos escenarios, motivadores de la búsqueda de una solución, son utilizados para detallar los requerimientos técnicos que debe tener un soporte de software que brinde un ambiente ideal de trabajo.

2.1 Ambiente de trabajo en la Investigación Conceptual Cooperativa

La investigación conceptual es una búsqueda a nivel cognitivo, en escritos científicos de actualidad dentro de un área de conocimiento determinada. En la metodología de la investigación, la investigación conceptual está relacionada a la etapa en la que se enfrenta un problema delimitado con el acervo bibliográfico correspondiente (estado del arte) y es donde surgen los primeros postulados e hipótesis que luego conllevarán a una investigación empírica.

Tal cual se ha expresado en el capítulo anterior, la comunicación digital ha permitido, entre otras cosas, que grupos de investigadores distribuidos geográficamente puedan trabajar en forma cooperativa, en particular, durante esta etapa de investigación conceptual. Dentro de esta etapa los investigadores enfrentan la difícil tarea de manejarse con una gran cantidad de material

compartido e interrelacionado, posiblemente generado por distintas aplicaciones e incluso almacenado sobre distintas plataformas.

El primer paso de este trabajo consistió en identificar y analizar las necesidades individuales y grupales que surgen dentro de esta etapa de investigación. La identificación surgió a partir de un análisis de las prácticas propias y de las prácticas de colegas que se encontraban en un ambiente de investigación como el planteado.

2.1.1 Necesidades individuales identificadas

El material con el que trabaja cada investigador está compuesto por distinto tipo de documentos, los cuales son generados por distintas aplicaciones (ver figura 2.1). Estos documentos (propios y de terceros) no son islas, están relacionados y no solamente por relaciones asociativas. Cada investigador tiene necesidades de agrupar, anotar y relacionar el material. Estas relaciones constituyen en sí conocimiento agregado acerca de la interpretación del material en estudio, y son metadatos sumamente valiosos para el resto de los investigadores. Se detallan a continuación las necesidades típicas de un investigador durante esta etapa:

a. Crear relaciones entre materiales heterogéneos

Un investigador que, por ejemplo, necesita establecer una relación entre una frase en un documento textual y una parte de un modelo representada por un objeto dentro de un documento gráfico.

En general los investigadores trabajan con distinto tipo de materiales, algunos de los cuales se detallan a continuación y se ilustran en la figura 2.1:

- Gráficos
- Planillas de cálculo
- Correos electrónicos
- Notas personales
- Documentación de terceros de solo lectura o que no deba ser alterada. (estándares, documentación de calidad, etc.)
- Planificaciones
- Animaciones
- Videos
- Fotos

b. Realizar lectura activa:

La investigación conceptual, así como también otras actividades humanas, está basada en un ciclo de lectura y escritura de documentos. Las anotaciones soportan este ciclo sobre documentos individuales, en

donde un lector instantáneamente se convierte en escritor y el documento se convierte en un artefacto que permite realizar esta actividad en forma productiva.

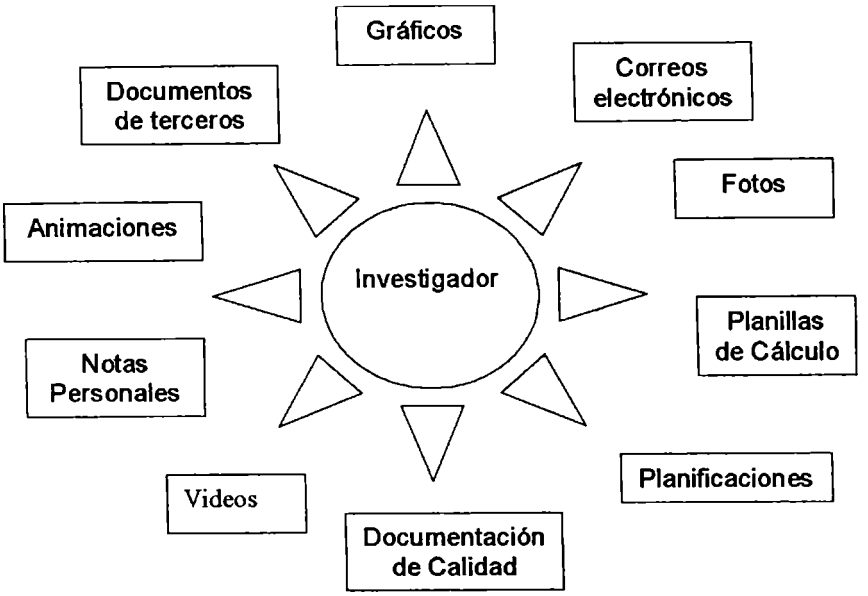


Figura 2.1: Materiales de trabajo de un investigador.

Leer en forma activa, es decir leer un material (al cual incluso podría carecerse del derecho de escritura) y realizar anotaciones relacionadas al mismo, es una actividad común en el ambiente de investigadores planteado. En ambientes donde varios investigadores cooperan sobre un conjunto de documentos, la posibilidad de agregar anotaciones a partes seleccionadas del material es importante ya que permite agregar valor a la base de conocimientos formada por escritos científicos de un área específica.

Otra consideración es la lectura activa de medios no textuales. Por ejemplo, la lectura activa de un video implica conectar hechos con otros, comparar secuencias, etc. Para hacer esto el usuario necesita crear notas y vincularlas a segmentos de video.

c. Establecer relaciones entre materiales almacenados sobre distintas plataformas

Un investigador podría tener que cambiar de plataforma para realizar distintas tareas y puede haber relaciones entre materiales de las distintas plataformas. Por ejemplo, un investigador podría necesitar vincular su material mantenido dentro de una plataforma Windows NT con documentación de calidad, de solo lectura, almacenada en un servidor sobre una plataforma UNIX.

d. Establecer otro tipo de relaciones, además de la simple asociación

Un investigador podría necesitar crear relaciones diferentes de un simple vínculo asociativo, por ejemplo, agrupar un conjunto de documentos relacionados (colección). Si un investigador necesita agrupar un conjunto de *PDFs* que relacionan la Computación Estructural con otras áreas, normalmente coloca todos estos documentos en el mismo directorio o carpeta. Sin embargo si alguno o varios de estos documentos a su vez pertenecen a otras colecciones de documentos relacionados, sería necesario mantener múltiples copias en distintas carpetas, con los problemas de mantenimiento y almacenamiento que la redundancia conlleva. Por lo tanto resulta de gran utilidad poder crear estructuras diferentes de la simple asociación, permitiendo por ejemplo, mantener agrupados conjuntos de documentos relacionados.

e. Establecer relaciones que involucren material incluido en documentos de terceros

Generalmente cuando se necesitan crear relaciones con documentos de terceros, a los que no se tiene derecho de escritura, se crea un vínculo hacia el documento completo. Hay situaciones en las que es necesario establecer un vínculo desde un documento propio a una posición específica de un documento de un tercero. Por ejemplo, crear un vínculo entre una parte de un documento Microsoft Word propio, y un punto específico dentro de un estándar de la W3C sobre una página HTML. Permitir realizar este tipo de vínculos, incrementaría la eficiencia en el acceso a este material.

f. Establecer relaciones con origen en documentos de terceros

Hay situaciones en las que es necesario crear un vínculo o relación con origen en un documento al cual no se tiene derecho de escritura. Si un investigador, por ejemplo, está leyendo un trabajo de un tercero, y determina que existe relación entre este trabajo y otro, debería poder establecer tal relación.

g. Crear distintos puntos de vista o contextos sobre el mismo material

Los investigadores necesitan crear distintos tipos de relaciones sobre el mismo material, es decir tener distintas vistas o contextos sobre el mismo grupo de documentos. Por ejemplo, un conjunto de escritos podría estar interrelacionado desde el punto de vista de la Computación Estructural y a su vez, algunos de estos documentos, podrían estar relacionados con otros documentos desde la óptica de la Hipermedia Abierta. Es necesario poder crear relaciones y anotaciones desde distintos puntos de vista sobre el mismo conjunto de documentos.

h. Determinar los documentos relacionados a un documento específico.

Hay situaciones en que es útil que un investigador pueda determinar los documentos que son origen de alguna relación y que tienen como destino a un documento específico (ver figura 2.2), es decir, dado un documento poder determinar el o los documentos que tienen como destino de una relación a un fragmento o a todo el documento dado. Si un investigador considera, por ejemplo, que un trabajo analizado es irrelevante para la investigación en curso y desea eliminarlo, sería útil poder determinar previamente si no existe alguna relación que tenga como destino el trabajo a eliminar.

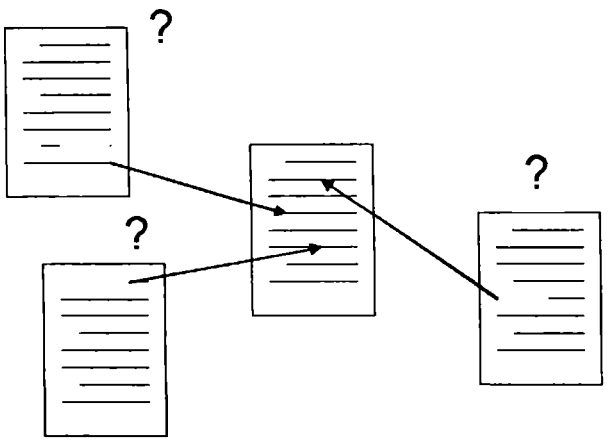


Figura 2.2: Documentos que son orígenes de relaciones, que tienen como destino un documento dado.

Además de estas necesidades individuales, están las necesidades que surgen por el hecho de que cada investigador trabaja en proyectos que involucran a otros colegas que pueden estar distribuidos geográficamente.

2.1.2 Necesidades grupales identificadas

Tal cual se planteó, cada investigador puede utilizar distintas aplicaciones para la generación de sus escritos, incluso distintas plataformas. El trabajo en grupo implica compartir e intercambiar material, anotaciones y relaciones así como también realizar escrituras conjuntas. En este entorno surgen necesidades de cooperación, coordinación y distribución de trabajo. A continuación se detallan las necesidades grupales identificadas:

a. Soportar acceso en línea al material, anotaciones y estructuras compartidas

A fin de reducir la necesidad de información redundante, lo que también reduciría el trabajo de mantenimiento y el espacio de almacenamiento, es necesario que todos los investigadores tengan acceso en línea, al material y a sus relaciones y anotaciones. Tener redundancia implicaría mantener todos los cambios en todas las copias del material redundante actualizadas,

de modo que se garantice la consistencia del mismo, y además involucraría un desperdicio del espacio de almacenamiento.

b. Realizar distribución y seguimiento de trabajo:

Hay situaciones en las que, por ejemplo, un investigador líder distribuye escritos especializados entre un grupo de colegas y necesita tener un seguimiento del material distribuido. El líder del grupo, por ejemplo, recibe una revista electrónica en la que hay artículos relacionados con trabajos en marcha, distribuye los artículos por pertinencia y necesita mantener las relaciones (y un recordatorio semanal) de quien es responsable de responder, comentar o resumir cada artículo distribuido.

c. Crear distintos puntos de vista o contextos sobre el mismo material:

Distintos investigadores comparten el mismo material, cada investigador puede crear sus anotaciones y relaciones entre dicho material. Si todos los vínculos, de todos los investigadores, se colocan sobre el mismo material, podría haber yuxtaposición de vínculos e inconsistencias, por otro lado si cada investigador tuviera una copia del material existiría redundancia. Es necesario poder mantener distintos conjuntos de relaciones y anotaciones, que representen distintos puntos de vista, sobre el mismo conjunto de documentos.

d. Producir nuevos escritos en forma conjunta

Durante la producción conjunta de trabajos, se realizan escrituras y se crean anotaciones y relaciones. En este contexto se identifican otras necesidades, las cuales se detallan a continuación:

d1. Bloquear acceso a documentos.

Cuando un investigador está editando un escrito compartido, debe impedirse que otro investigador pueda acceder al mismo con derecho de escritura.

d2. Notificar cambios.

Hay situaciones en las que una vez modificado el escrito compartido es necesario notificar a uno o varios colegas acerca de los cambios realizados. Las modificaciones pueden incluir cambios en un texto, en las anotaciones y en las relaciones. Por ejemplo, si se modifica un documento de calidad o algún documento que es un referente dentro de la investigación, es necesario que todo el grupo tome conocimiento de dicha modificación.

d3. Identificar el autor de comentarios y/ o relaciones.

Debe poder determinarse quién fue el autor de un comentario o de una relación a fin de poder contactar a la persona responsable y contribuyente del mismo.

e. Consensuar un vocabulario común.

Los integrantes del grupo deberían ser capaces de poder compartir una ontología o un vocabulario común consensuado y de poder acceder fácilmente desde el material bajo estudio a las definiciones que forman parte de ese vocabulario consensuado.

f. Mantener espacios de trabajo privados.

También es necesario que cada investigador pueda mantener un conjunto de documentos, relaciones y anotaciones en forma privada.

g. Utilizar el modelo argumentativo de Toulmin en debates y discusiones.

Los investigadores podrían necesitar formalizar sus debates y discusiones utilizando el modelo argumentativo de Toulmin. La argumentación está asociada a actos verbales epistémicos interesados en la generación y producción de ideas (conocimientos). El argumento de Toulmin consta de cuatro componentes principales (conclusión, fundamento, garantía y respaldo) que deben mantenerse relacionadas, y no por relaciones asociativas, sino más bien agrupadas. Además la relación entre las componentes no es de tipo origen y destino, como en el caso de relaciones asociativas, sino más bien del tipo: el fundamento "sustenta" la conclusión.

Cabe destacar que para muchas de las necesidades identificadas, generalmente, se utilizan soluciones paliatorias e incluso algunas necesidades ni siquiera son consideradas como tales, debido a la imposibilidad o dificultad de las herramientas con las que trabajan los investigadores para poder darles una solución efectiva. En general, el investigador se adecua a las posibilidades que le brindan las herramientas con que trabaja, en lugar de que las herramientas se adecuen a las necesidades del investigador.

Las herramientas en sí, obstaculizan la posibilidad de que el investigador busque mejores formas de comunicarse. Por ejemplo, si un investigador, que está escribiendo un documento, necesita establecer una relación con un objeto que pertenece a un documento gráfico (necesidad individual identificada como "a"), en general, pega el gráfico completo dentro del documento que está editando, lo que implica redundancia de información. Por otro lado, una necesidad que no es considerada como tal, es la de poder determinar qué material tiene como destino de una relación, al documento con el que se está trabajando (necesidad individual identificada como "h"). Esta última necesidad

no es considerada simplemente porque resulta imposible de realizar con las herramientas que se utilizan habitualmente.

El siguiente paso de este trabajo consistió en adoptar una tecnología y determinar las características técnicas requeridas que un soporte debería proveer a fin de crear un ambiente de trabajo ideal en el cual todas las necesidades identificadas puedan ser satisfechas.

2.2 Tecnología y características técnicas requeridas

Dentro de un ambiente como el planteado, rebrota la inquietud de Vannaver Bush (reconocido como el pionero de la Hipermedia), escrita en "*As we may think*" en 1945, donde establecía las dificultades para almacenar y consultar eficientemente la gran cantidad de conocimiento acumulado. Si bien actualmente se dispone de la tecnología necesaria, la clave para que el material pueda ser consultado eficientemente está en la posibilidad de contar con las relaciones apropiadas que conecten dicho material.

La Hipermedia tiene el potencial de contribuir en la mayoría de las áreas de trabajo con material e información digital. Cada área representa desafíos para esta tecnología y para cómo ésta debe ser diseñada. Se considera que la tecnología Hipermedia es la mejor forma de abordar el soporte para cubrir las necesidades identificadas en el ambiente de investigación planteado, y que la Hipermedia en general es una forma natural de representar y compartir conocimiento. La utilización de links y otros mecanismos de estructuración pueden representar efectivamente las relaciones descriptas entre los distintos tipos de documentos. Se detallan a continuación las características técnicas que un soporte tecnológico debería proveer:

A. Links entre documentos generados por distintas aplicaciones.

   Dado que distintos investigadores pueden utilizar distintas aplicaciones para crear y editar su material (editores de texto, editores gráficos, etc.) es necesario contar con links entre documentos generados por distintas aplicaciones.

B. Links entre distintas plataformas

   Diferentes investigadores pueden trabajar bajo distintas plataformas, incluso un mismo investigador puede tener material bajo distintas plataformas tal cual fue ejemplificado anteriormente.

C. Estructuras diferentes al simple link asociativo.

   Para cubrir necesidades como la ilustrada en el punto "d", de las necesidades individuales, en las que un investigador agrupaba documentos relacionados. Esto permite, entre otras cosas, crear grupos de documentos relacionados, sin necesidad de ubicarlos en la misma carpeta y por lo tanto

sin la necesidad de tener material redundante. Del mismo modo, es un requerimiento necesario, para proveer soporte al modelo argumentativo de Toulmin planteado en el punto "g" de las necesidades grupales.

D. Almacenamiento de links y de otras formas de estructuración en forma separada al material.

Para cubrir necesidades como la ilustrada en el punto "g" de las necesidades individuales, la cual requería el mantenimiento de distintas vistas sobre el mismo material. También para cubrir necesidades como la del punto "h" de las necesidades individuales, en la que debe poder determinarse los documentos que tienen como destino de una relación a un documento dado.

E. Estructura y servicios de estructuración a las aplicaciones favoritas.

Un ambiente de trabajo ideal debería cubrir las necesidades planteadas sin que para ello el investigador debiera mudarse de sus aplicaciones favoritas. Por lo tanto debe proveer servicios de estructuración y anotación a las aplicaciones que normalmente utiliza cada investigador. Por ejemplo, para aquellas aplicaciones que no permitan la creación de links, proveer un servicio, que permita dicha creación. Esto también está relacionado con mantener las estructuras separadas del material, ya que las aplicaciones se hacen cargo del almacenamiento y mantenimiento del material, mientras que las estructuras y anotaciones agregadas se almacenan y mantienen por separado a través del servicio de estructuración.

F. Links bidireccionales y con múltiples puntos extremos.

Para cubrir necesidades como la descripta en el punto "e", de las necesidades grupales, en las que se comparte un vocabulario común. Por ejemplo si una frase en un documento "D1" está relacionada con otro documento "D2", y a su vez esa misma frase esta definida dentro de un diccionario compartido, el investigador debería tener la posibilidad de seleccionar dicha frase y elegir el destino de la travesía del link (o bien hacia el documento "D2" o bien hacia la definición en el diccionario). Para este caso específico se requiere de un link con un origen y dos destinos.

G. Links con puntos extremos calculados.

Para cubrir necesidades grupales como la "e", en las que resulta necesario que cualquier ocurrencia de una palabra, sirva como origen de un link hacia una definición de la misma. En este caso el origen de un link no es un punto fijo, sino más bien, un texto que puede aparecer en cualquier documento. El punto extremo origen, en este caso, debe implementarse como un punto extremo que se calcula mediante una búsqueda de la palabra en cuestión sobre cualquier documento del material. El punto extremo fijo, es la definición de la palabra en el diccionario que representa el vocabulario común consensuado.

H. Links con atributos.

Dentro del punto "d" de necesidades grupales, se identificó la necesidad de poder determinar quién era el autor de una relación. Para esto es necesario que el link tenga atributos, entre los cuales esté el autor del mismo y la fecha de creación.

I. Bloqueo de acceso a documentos.

Tal cual lo expresado dentro de las necesidades grupales "d", de escritura conjunta, los documentos y estructuras deben estar almacenados en bases de datos que permitan bloquear acceso a los mismos cuando sea necesario.

J. Notificación de eventos.

También relacionada a las necesidades de escritura conjunta, en las que una modificación en el material o en las relaciones compartidas requiera que otros investigadores se pongan al tanto de las mismas.

Existen distintos Sistemas Hipermedia que podrían ser considerados en la búsqueda de una solución que cree un ambiente de trabajo ideal, sin embargo, algunos de éstos obligarían a los investigadores a mudarse de sus aplicaciones favoritas; otros no proveerían, o al menos no con facilidad, muchas de las características técnicas requeridas. Por tal motivo, y tal cual se concluye en el capítulo 3, se propone el diseño de un nuevo Sistema Hipermedia que brinde el soporte requerido.

Capítulo 3

Historia y Trabajos Relacionados

Resumen

En este capítulo se presenta inicialmente una reseña histórica, en la cual se muestran los principales precursores del nacimiento de la Hipermedia y las distintas líneas de investigación surgidas. Posteriormente se describe la evolución de los Sistemas Hipermedia, presentando los Sistemas más relevantes junto con sus fortalezas y debilidades y se arriba a algunas cuestiones pendientes de investigación. El capitulo cierra con la conclusión de los sistemas analizados y la justificación de la propuesta de diseño de un nuevo Sistema Hipermedia que brinde el soporte tecnológico requerido al ambiente de investigación planteado en el capitulo 2.

3.1 Introducción

Desde los años 60 muchas variantes de los conceptos fundamentales de la Hipermedia han sido propuestas e implementadas. Los sistemas resultantes han soportado la creación y navegación de redes de texto, e información multimedia, interrelacionado.

Actualmente hay dos ejemplos de Hipermedia y multimedia dominantes. El primero es la Wide World Web (Web), el mayor sistema para organizar y navegar información sobre Internet, y el segundo lo constituyen los paquetes multimedia educacionales, enciclopedias, novelas y todos los sistemas que se distribuyen en CD.

Se podría pensar que a través de la Web los Sistemas Hipermedia han alcanzado la cima, sin embargo, hay mucho por mejorar considerando los valiosos aportes, generados incluso desde antes del advenimiento de la Web, que ha realizado la comunidad de investigadores de la Hipermedia. En la actualidad muy pocos usuarios disfrutan de una buena funcionalidad Hipermedia integrada dentro de su ambiente de trabajo, trabajar con esta funcionalidad Hipermedia aún requiere un conjunto separado de herramientas.

La Hipermedia debería proveer soporte para material compartido, organizado sobre distintas estructuras, las que no sólo incluyan el simple vínculo asociativo, brindando de esta forma un soporte efectivo a las necesidades de un ambiente como el planteado en el capítulo 2. Además el soporte debería estar integrado al ambiente de trabajo del usuario, es decir a sus editores y herramientas favoritas. La necesidad de integración constituye uno de los desafíos más grandes de los diseñadores de Hipermedia.

Dado que el estudio que este trabajo llevó a cabo sobre los precursores de la Hipermedia, permitió un mejor entendimiento e interpretación del origen de éste área, de sus metas y del estado actual del arte, se presentan a continuación los principales pioneros conjuntamente con sus trabajos y visiones.

3.2 Precursores de la Hipermedia

Vannevar Bush

"A Memex is a device in which an individual stores his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory..."

"...It affords an immediate step to associative indexing, the basic idea of which is a provision whereby an item may be caused at will to select immediately and automatically another. This is the essential feature of the Memex."

Vannevar Bush - **Memex** (*Memory Extension*) – 1945 [3]

El Ingeniero norteamericano Vannevar Bush, director de la "*U. S. Government's Office of Scientific Research and Development*", trabajaba para el presidente Roosevelt en la coordinación de las actividades de los científicos líderes de Estados Unidos. En julio de 1945, probablemente inspirado por la novela "*World Brain*" de H. G. Wells[1], publica un artículo titulado *"As we may*

[1] H. G. Wells: Escritor británico conocido por sus novelas de ciencia ficción, en 1938 publicó una colección de ensayos acerca de la organización futura del conocimiento y la educación. Entre estos ensayos estaba "*The Idea of a Permanent World Encyclopaedia*" en el cual planteaba un recurso distribuido para estudiantes de todo el mundo.

think" en *"The Atlantic Monthly"*, en el cual instaba a los científicos, luego de finalizada la segunda guerra mundial, a dedicar sus esfuerzos no sólo a extender los poderes físicos del ser humano (microscopios que potencien los ojos o maquinas de destrucción que potencien sus puños) sino también los poderes de su mente.

La preocupación central de Vannevar Bush era la extensión en proporciones prodigiosas de la experiencia y el conocimiento humano y las consiguientes dificultades de los investigadores para la utilización, almacenamiento, recuperación y consulta eficiente de toda esa información. Vale aclarar en este punto que, gran parte de la motivación de este trabajo surgió a partir de la convicción de que esta preocupación de Bush sigue siendo válida, a pesar de la Web y de toda la tecnología que se dispone.

Para atender su inquietud, Bush, ideó un aparato, el cual bautizó con el nombre de *Memex*, consistía en una especie de mesa con superficies translúcidas, teclado, palancas y botones que podían buscar rápidamente archivos en forma de *microfilms*. Pero además, y lo que era decisivo, el lector podía añadir notas marginales y comentarios mediante un sistema de fotografía seca que permitía incluir las notas en la película del *Memex*. La clave de este dispositivo es que funcionaría imitando los procesos de la mente humana que trabaja por asociación, de acuerdo con un intrincado tejido de senderos construido por las células del cerebro. Bush proclamaba que aunque no se pudiera reproducir totalmente este proceso artificialmente, se podía aprender de él. De esta forma, la selección debía ser por asociación y no por la ubicación mecánica de temas en un índice alfabético.

Quien consultaba dicho aparato, podría construir caminos (*Trails*) de lectura de acuerdo con su interés, seleccionando y enlazando los artículos que deseara a través del laberinto de materiales disponibles, y podría modificar esta configuración cuando lo deseara. Con su máquina del futuro, Bush proponía no sólo una nueva forma de manejar la información sino también, una nueva forma de escribirla y de leerla, es decir, por asociación de documentos, además de contemplar la posibilidad de la intervención del lector quien mediante sus comentarios sería, a su vez, autor.

Imaginaba, mucho antes del desarrollo de la computadora, un aparato que, a la manera de un suplemento de nuestra memoria, facilitaría el acceso y la vinculación de la información acumulada.

> *"In fact, every time one combines and records facts in accordance with established logical processes, the creative aspect of thinking is concerned only with the selection of the data and the process to be employed, and the manipulation thereafter is repetitive in nature and hence a fit matter to be relegated to the machines."*
>
> Vannevar Bush - *Memex* – 1945 [3]

Planteaba que la manipulación de los datos era repetitiva en naturaleza y podía ser delegada a las maquinas, además planteaba la necesidad de poder crear asociaciones (*links*) y caminos (*trails*):

> *"Any two items in the Memex could be coded for associative selection – a trail, allowing linking and personalization... Trails are not simple a joining of two documents – they refer to a coherent trail through a number of documents on a single trail"*
>
> Vannevar Bush - Memex – 1945 [3]

Los ítems podían formar parte de muchos caminos, se podrían comprar "libros" con caminos preestablecidos (lo que hoy conocemos como tours guiados) y la existencia de organizaciones y profesionales dedicados a la producción de estos caminos.

La máquina que Bush ideó no llegó a materializarse nunca, pero sentó las bases para la Hipertextualidad. Este mecanismo, permitía crear rastros de pensamiento y evitaría el problema de la memoria transitoria humana. La mente humana, cuando no puede conectar una información a otra ya establecida en nuestra memoria, la pierde. El *Memex* evitaría este problema, ya que se podría consultar esa información sin necesidad de recordarla y evidentemente, sin que se pierda.

Vannevar Bush estaba convencido de la necesidad de anotar comentarios, a medida que se leía un texto. Esto, redefinía el concepto de lectura como proceso activo que implica escritura. Este trabajo reconoce en Bush a un creador, un visionario que con sus ideas motivó y movilizó a investigadores que le sucedieron, como Douglas Engelbart y Theodor Nelson, quienes "despertados" por éstas ideas dieron pasos fundamentales hacia lo que hoy es la Hipermedia, la Web y en gran medida la Informática en general.

Douglas Engelbart

> *"By 'augmenting human intellect' we mean increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to problems."*
>
> Douglas Engelbart 1963 [38]

Douglas Engelbart, trabajaba como operador de radar en las Filipinas durante la Segunda Guerra Mundial, se inspiró en el artículo de Vannevar Bush para buscar la forma de utilizar las computadoras para mejorar la sociedad.

Cuando terminó la guerra, y siguiendo con esta idea, renunció a su trabajo como ingeniero y se fue a estudiar a UC Berkeley. En Mayo del 1962, trabajando en el "*Stanford Research Institute*", le escribió una carta a Vannevar

Bush solicitando su autorización para citar algunos párrafos de su artículo sobre el *Memex*; Por ese entonces, Engelbart, trabajaba en lo que seria **NLS** (*oN-Line System*) / **Augment** (*Augmentation of the human intellect*) [38], el primer Sistema en línea conocido y el inicio de los Sistemas Hipermedia. Como dato curioso, Engelbart previamente había presentado *"Off Line System"* (OSL), irónicamente por falta de soporte económico para su proyecto, por éste motivo posteriormente eligió la sigla *NLS* para el sistema definitivo.

El proyecto *Augment*, que permitía a un grupo de trabajadores compartir información dentro de una red de computadoras, constituyó una etapa fundamental en la historia que llevó a la construcción de las actuales máquinas digitales interactivas; Engelbart presentó una parte de su sistema *NLS* durante la "*Fall Joint Computer Conference*" de 1968, en esa ocasión demostró de manera práctica los resultados alcanzados. Este innovador sistema proveía al usuario una interfaz de múltiples ventanas en la cual todo podía ser vinculado y la principal interacción se realizaba a través del *mouse* (otra de las innovaciones de Engelbart).

En forma paralela a estos desarrollos Engelbart participaba en las reuniones que iban definiendo el proyecto Arpanet, red militar y versión previa de Internet, o sea la creación de una red de computadoras financiada por la agencia estatal ARPA (*Advanced Research Projects Agency*) basada en la transmisión de los denominados paquetes de datos. El "*Stanford Research Institute*" fue uno de los primeros nodos de esta red pionera que interconectaba algunas universidades y centros de investigación de los Estados Unidos.

La incesante actividad en el campo tecnológico desarrollada por Engelbart, quien es considerado unánimemente el Thomas Edison de la informática, lo ha llevado a la proyección de numerosos dispositivos de interacción que revolucionaron la forma de comunicarnos con las máquinas digitales. Engelbart no sólo ha inventado el mouse (conocido originalmente como "indicador de posición X-Y para sistemas con monitor"), un componente fundamental de las actuales interfaces gráficas, lo cual bastaría para adjudicarle un lugar preeminente en la historia de la informática; sino que ha desempeñado un rol central en el estudio y desarrollo de mecanismos que agilizan el aprendizaje y la utilización, sobre todo a nivel grupal, de los computadoras.

En sólo dos décadas las revolucionarias tecnologías nacidas en el laboratorio de Engelbart pasaron a formar parte de la dotación estándar de cualquier computadora personal. Douglas Engelbart prefiguró un mundo de acceso directo a la información y un nuevo modo de trabajar en grupo, con el objetivo de potenciar (*augment*) las capacidades individuales de cada uno de sus integrantes, que hoy ya pertenecen a nuestra realidad cotidiana.

Theodor Nelson

"Hypertext is a collection of documents containing cross-references which, with the aid of an interactive browser program, allow the reader to move easily from one document to another."

Theodor Nelson 1965 [4]

Por otro lado Ted Nelson, quien en 1960 se inscribía en el "*Graduate School*" de Harvard, en su primer año trabajó en un proyecto para crear un sistema de escritura similar a un procesador de texto, pero que permitía que distintos documentos sean vinculados en forma no lineal por asociación (basándose en las ideas del *Memex*). Nelson no terminó con este proyecto pero continuó trabajando en el tema. En 1965 presentó un escrito en la "*Association for Computing Machineryin*" donde inventaba el término Hipertexto en referencia a escritos no secuenciales, una combinación de lenguaje textual con la capacidad de la computadora para saltar y mostrar dinámicamente, que coordinaran la presentación de cualquier tipo de información, texto e imágenes. En 1967, basándose en la teoría de Bush y en el proyecto *Augment* de Engelbart, Nelson nombró y creó su propio proyecto, el proyecto **Xanadu**. (http://xanadu.com/)

Xanadu tenía como objetivo la construcción de una gran red hipertextual que permitiera almacenar y enlazar todo lo que haya sido escrito en todos los tiempos para ser consultado por los usuarios de forma sencilla. Una vez creada, esta gran red, debería permitirse que nuevos textos, a través de links, ampliaran e interrelacionaran la información existente. Era una forma revolucionaria de lectura y escritura. Preveía un universo de documentos, *docuverse* (Nelson 1980) [5], a lo largo del mundo y sobre los cuales, los usuarios pudieran reutilizar material y agregaran sus propios vínculos y anotaciones a lo que leían. Como curiosidad se señala que *Xanadu* es una palabra que procede de un poema titulado "Kubla Khan" y que se refiere a una ciudad mágica y onírica.

Tim Berners Lee

"The World-Wide Web (W3) initiative is a practical project to bring a global information universe into existence using available technology."
Berners Lee 1992 [6]

El último gran precursor de la Hipermedia actual (quien materializó su globalización) es Tim Berners Lee, quien liderando un grupo en la CERN (*Centre Européen pour la Recherche Nucléaire*), creó la Wide World Web (Web) [6]. Esto permitió que varias de las ideas de los pioneros se popularizaran y llegaran a interconectar el mundo. La meta de Berners Lee, según lo declara en su libro "*Weaving the Web*" era desarrollar un sistema con el espíritu del *Memex*, donde la vinculación de documentos fuera posible, pero preveía que no solo documentos de bibliotecas pudieran ser parte de este

sistema. Berners Lee además quería que la creación de contenido la pudiera realizar cualquiera (y no solo la creación de links a documentos existentes), esto significó un importante avance respecto de las ideas originales de Bush.

Para Berners Lee, las herramientas para crear y modificar contenido Web, eran tan importantes como las que servían para navegarla. Este inglés es, actualmente, el director del consorcio que lleva el nombre de su proyecto W3C (*World Wide Web Consortium*), el cual lidera programas para que empresas y compañías potencien la utilización de Internet y de la Web.

El avance más importante de Lee fue unir hipertexto con Internet. En su libro "*Weaving The Web*", explica además que sugirió repetidamente a ambas comunidades técnicas (las de hipertexto e Internet) que era posible y necesaria una unión entre estas dos tecnologías, pero dado que nadie atendió esta invitación, encaró él mismo el proyecto. Este trabajo considera que éste fue un punto de inflexión en la historia de la Hipermedia, que ya debido a la falta de una invitación formal o al menos más específica por parte de Lee y/ o debido a la falta de una actitud más cooperativa por parte de la comunidad Hipermedia de esa época, esta unión de tecnologías no se dio, y con ello se desaprovechó un gran camino recorrido por los investigadores de la Hipermedia, abordándose la Web como una línea de desarrollo e investigación separada. De no ser así, probablemente hoy contaríamos con una Web dueña de una funcionalidad Hipermedia mucho más rica que la que posee actualmente.

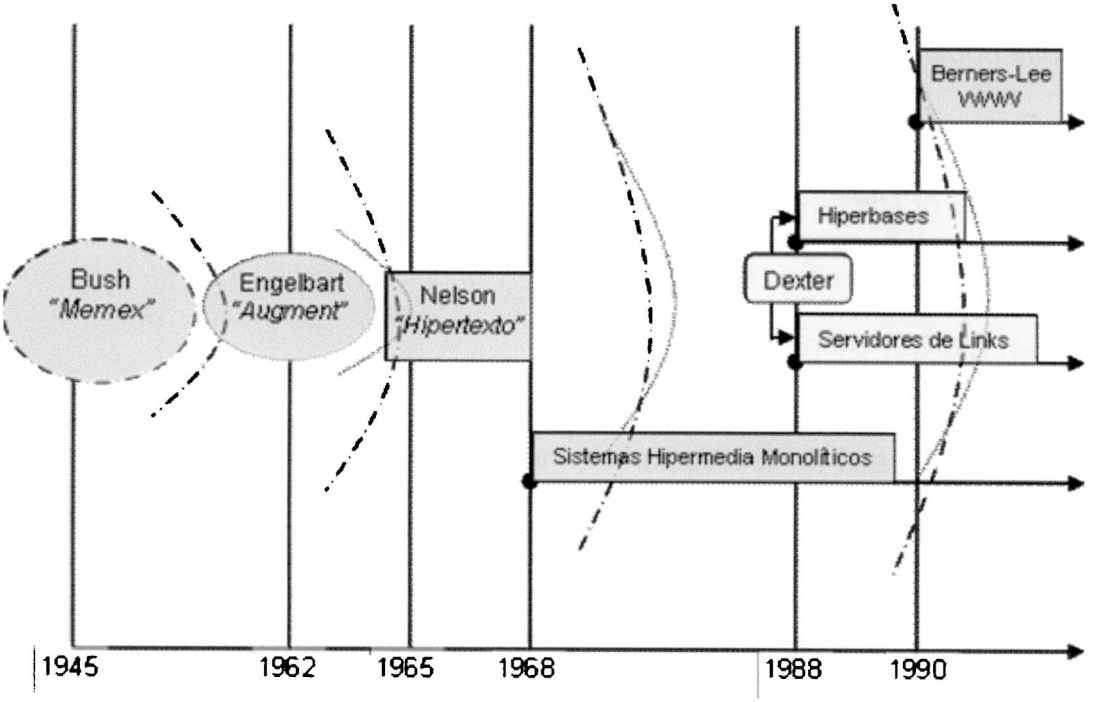


Fig. 3.1: Precursores y líneas de investigación surgidas.

Vannevar Bush, Douglas Engelbart, Theodor Nelson y Berners Lee produjeron hitos que impactaron, desde una visión general, en toda la

sociedad, ya que cambiaron la forma de comunicarse, de leer, de escribir y en gran parte, hasta de pensar. En la figura 3.1 se presenta un gráfico en el que se ilustra la evolución cronológica de los precursores de la Hipermedia. En dicho gráfico se ilustra además, cómo las ideas de los precursores influenciaron en su posterioridad.

3.3 Precursores en el diseño y desarrollo de Hipermedia

Si se consideran los hitos de la historia de la Hipermedia desde el punto de vista del diseño y desarrollo de sistemas, es necesario hablar del modelo de Referencia de **Dexter** (1990) [7]. Este modelo fue creado por un grupo de destacados desarrolladores de Sistemas Hipermedia, quienes conceptuaron por primera vez todo lo que debía subyacer bajo los sistemas Hipermedia para poder soportar la filosofía de comunicación planteada por los precursores. Este grupo de dieciséis desarrolladores (entre los cuales estaba Engelbart) comenzó a reunirse en 1988 y formalizó su propuesta a través de Frank Halasz y Mayer Schwartz en 1990.

Este trabajo considera que la conceptuación realizada por este grupo, fue un paso necesario y decisivo para poder avanzar sobre conceptos y principios que deben subyacer en el diseño y desarrollo de Sistemas Hipermedia, que permitió plantearse metas hacia las cuales dirigirse. Se considera que, si bien experimentar implica implementar herramientas que concreten las propuestas, trabajar sobre conceptos y principios es la única forma de poder "ver más allá" de lo que las herramientas pueden brindar.

3.4 Evolución de los Sistemas Hipermedia

> *"The field of Hypermedia concerns the design and use of systems that support authoring, managing, and navigating networks of interlinked textual and multimedia information"*
>
> Jacob Nielsen 1990 [39]

La palabra interconectado (*interlinked*) es esencial; el avance de la Hipermedia más allá de la creación de multimedia y de sistemas de manejo de documentos, es su explícito soporte para representar interconexiones entre material en línea. En lo más simple, estas interconexiones toman la forma de links que conectan una fuente con un destino. Como veremos, los Sistemas Hipermedia han evolucionado para soportar una variedad de estructuras (además de estos vínculos asociativos clásicos); explícitas y calculadas.

Revisando la historia desde el punto de vista del desarrollo de Sistemas Hipermedia, se puede identificar un corrimiento desde el diseño de sistemas de ambientes integrados o monolíticos, que incluían todos los editores de texto, bases de datos y demás, que un usuario pudiera necesitar; hacia sistemas abiertos que provean estructuras (no solamente *links*) Hipermedia para editores, visores y aplicaciones estándares.

A continuación se describen los Sistemas relevantes a lo largo de la evolución de la Hipermedia. No se intenta una descripción minuciosa de todos los sistemas ni un análisis profundo de los mismos (lo cual implicaría un trabajo en si mismo), sólo se plantean los sistemas más relevantes y se destacan las virtudes y las falencias para brindar el soporte al ambiente planteado en el capítulo 2. El apéndice A contiene una descripción más detallada de los sistemas descriptos.

3.4.1 Sistemas Monolíticos

El primer sistema de Hipertexto en funcionamiento fue [38] **NLS/ Augment**, un sistema monolítico que intentaba soportar todas las aplicaciones que el usuario necesitara, incluyendo e-mail, herramientas de programación y editores de texto. Hasta ese momento ensamblar aplicaciones existentes no era viable. Pocas aplicaciones podían superar las herramientas provistas por *Augment*. Más aún, no había disponible infraestructura para la integración de aplicaciones, por lo que *Augment* no tuvo otro remedio que proveer la propia. Sin embargo, *Augment* presagiaba la Web 25 años antes (Engelbart 1984 [8]).

Varios sistemas posteriores compartieron el enfoque monolítico (proveer un ambiente integrado) de *Augment*. Por ejemplo **KMS** [9], el primer sistema en trabajar con grandes hipertextos (utilizado por la marina americana), proveía su propio e-mail y editores de texto y gráfico, además permitía importar y exportar soporte para algunos formatos de dato. **NoteCards** [10], ideado como un ambiente monousuario de propósito general, era profundamente adaptable, pero podía integrarse con cualquier aplicación escrita dentro del ambiente Lisp en que corría. En cierto sentido, era el ambiente Lisp, más que el Sistema Hipermedia, el que soportaba la integración de aplicaciones. Sin embargo, Notecard, contaba con una característica muy interesante, el navegador de tarjetas, esto es, una representación gráfica de la estructura de los *links* (permitía la edición de dicha estructura) que a su vez era una entidad que podía ser vinculada a cualquier otra tarjeta. Las lecciones aprendidas en el desarrollo de *Notecards* fueron cristalizadas en el famoso escrito de Halasz "*seven issues*" en 1988 [11], donde identificaba áreas que debían ser cubiertas por los nuevos Sistemas Hipermedia.

El mismo argumento que *Notecards* tenia **Intermedia** [12], desarrollado en la universidad de Brown con propósitos educacionales, el cual estaba construido en el framework para aplicaciones MacApp, y por lo tanto podía integrar solo aplicaciones basadas en ese framework.

3.4.2 Sistemas que separan la estructura del contenido

Un paso crucial en la evolución de la Hipermedia Abierta fue el reconocimiento de la importancia de separar estructura del contenido. El sistema *Intermedia*, a pesar de ser monolítico, logró esta separación con sus

bases de dato *Web* (base de datos que almacenaba los *links*), que permitían que los vínculos sean almacenados separados del texto y del material gráfico.

Intermedia fue construido desde un framework común orientado a objetos y soportaba la construcción de distintos editores para documentos de tipos específicos. El framework aseguraba que los editores soportaban el mantenimiento de bloques de dato (lo que luego se denominara ancla) identificables en diferentes tipos de documentos. Los bloques eran objetos con un identificador y alguna especificación de una ubicación en un documento dado.

Los vínculos en *Intermedia* eran relaciones simétricas entre objetos de bloques; ambos, los bloques y los vínculos, eran cargados en la base de datos *web*. De este modo, *Intermedia*, soportaba una estructura de vínculos privada para cada usuario, los que solapaban su estructura sobre una estructura de información pública. Por ejemplo, *Intermedia*, posibilitaba que los usuarios de biblioteca pública de solo lectura, mantengan una *web* personal con notas y referencias. *Intermedia* también soportaba acceso multiusuario parcial a las *webs* permitiendo al usuario entrar en nodos y *links* en la base de datos, en modo lectura, anotación o escritura.

Intermedia representa el primer paso hacia la apertura, pero del mismo modo que *NoteCards*, ambos, las webs y el material o dato, debían vivir dentro del ambiente de *Intermedia*. Esta debilidad fue señalada por uno de sus desarrolladores, Meyrowitz, en 1989 en el artículo de "*missing link*" [13]. Sin embargo la idea del bloque y el conjunto de características avanzadas que poseía, han tenido una gran influencia en el modelo de *Dexter* y en los Sistemas de Hipermedia Abierta posteriores. Se describen a continuación las características avanzadas (cruciales en la evolución) de *Intermedia*.

- Acceso multiusuario a documentos y estructuras Hipermedia.
- Derechos de acceso: lectura, escritura y anotación.
- Colecciones de documentos que denominaba "*corpus*"
- Estructuras Hipermedia ("*Webs*") almacenadas externamente
- *Links* bidireccionales
- Aplicaciones WYSIWYG[2] totalmente desarrolladas
- Navegadotes gráficos de *links*
- *Framework* orientado a objetos para facilitar su extensibilidad.

La separación de estructura y contenido se conceptualiza a fines de los 80, cuando un grupo de diseñadotes destacados (Halasz y Schwartz entre otros) reunidos en Dexter Inn en Sunapee, capturaron los conceptos e ideas fundamentales de los Sistemas Hipermedia más relevantes, y desarrollaron un Modelo de Referencia de Hipertexto (presentado en la sección 3.3) que

[2] WYSIWYG, *What You See Is What You Get*: Procesador de texto o sistema de publicación en el cual la pantalla muestra el texto exactamente en la forma en que será impreso.

combinaba las mejores ideas de estos sistemas junto con unos pocos conceptos decisivos.

Luego de varios encuentros Halasz y Schwartz se pusieron a trabajar en las conclusiones del grupo para armar el modelo formal, el cual presentaron en 1990 en el Instituto Nacional de Estándares y Tecnología [7]. Dada la importancia que éste modelo tuvo en la conceptualización y en los principios para el diseño y desarrollo de Sistemas de Hipermedia, y dado que el modelo forma parte de los fundamentos de este trabajo, en el capítulo 4 se presenta dicho modelo en forma exclusiva.

3.4.3 Servicios de Hipermedia Abierta

> *"Carefully examining the systems created to date, uncovers a single common thread that explains why hypertext/ hypermedia systems have not caught o: virtually all systems have been insular, monolithic packages that demand the user disown his o her present computing environment.*
>
> Meyrowitz 1989 [13]

Con la revolución de las computadoras personales (PC) surgieron una gran cantidad de nuevas aplicaciones, y los desarrolladores de Hipermedia (hasta ese entonces, en su mayoría, de sistemas monolíticos) no pudieron manejarse con esa cantidad de nuevos editores soportados por las PC. El resultado fue que las aplicaciones Hipermedia no fueron tan ampliamente utilizadas como lo podrían haber sido. Meyrowitz notó esto y sugirió que el curso correcto era cubrir las aplicaciones de terceros y ofrecer funcionalidad Hipermedia a dichas aplicaciones, idealmente hasta el punto en que, la funcionalidad Hipermedia estuviera tan omnipresente como el *"copy"* y el *"paste"*. Esta idea comenzó a ser conocida como Hipermedia Abierta, la aumentación de las aplicaciones de terceros con Hipermedia.

Una ventaja adicional de la integración de aplicaciones era que el usuario podía crear relaciones entre documentos manejados por diferentes programas que, de otra forma no podían ser integrados. Estas diferentes aplicaciones se mantenían unidas a través de un Sistema de Hipermedia Abierta (SHA).

Los SHA tienen muchas características en común, lo que es de esperar dado el similar espacio de problemas que cubren. Debido a la necesidad de integrar aplicaciones de terceros con diferentes (generalmente cerrados) formatos de documento, es imposible almacenar *links* y demás estructuras dentro de los documentos y por ello los SHA utilizan almacenamiento externo para las estructuras Hipermedia.

El almacenamiento de las estructuras es manejado generalmente por un servidor Hipermedia. Del mismo modo, es necesario un cliente que recupere

estas estructuras Hipermedia y las muestre de algún modo en conjunción con el documento al que pertenecen. Las áreas en las que usualmente difieren los SHA son las formas en que abordan la distribución, colaboración, marcado de *links* y extensibilidad.

A fines de los ´80 y principios de los ´90, un conjunto de Servicios de *Link* y de Sistemas de Hiperbase emergieron. Estos sistemas aplicaban variantes de los principios básicos del modelo de *Dexter* (Meyrowitz integraba el grupo Dexter), es decir, los vínculos y otras estructuras Hipermedia debían estar separadas de los datos.

El primer intento de salir de la naturaleza monolítica de *NoteCards* e *Intermedia* fue tomar la forma de servicios de *link*. Un servicio de *link* es un servidor independiente que provee vínculos Hipermedia para editores y visualizadores de terceros a través de un canal de comunicación. Por ejemplo, el servicio de *link* de Sun, **Sun Link Service** [15] (considerado como el primer SHA y sobre cuyas conclusiones se basaron los SHA posteriores más relevantes) y **LinkWorks** [16] soportaban vínculos utilizando un servidor separado.

En Sun Link Service y LinkWorks, el servicio de *link* era provisto por medio de un servidor de *links* y APIs que soportan la comunicación entre el servicio y las aplicaciones de terceros para la creación y travesía de *links*. La falla de estos sistemas fue, en gran parte, debido a que el éxito de los mismos dependía de que los desarrolladores modificaran sus aplicaciones. Otros ejemplos de este tipo de servicios de *links* fueron **Multicard** [17] y **Chimera** [18] (ver apéndice A).

Con el mismo propósito, de salir de los sistemas monolíticos hacia sistemas abiertos, y bajo la misma influencia del modelo *Dexter* surgieron, en forma contemporánea a los servicios de *links*, los sistemas de Hiperbase. Una Hiperbase es una base de dato de propósito especial que maneja el almacenamiento de objetos Hipermedia y, en algunos casos, contenidos de dato. Una Hiperbase puede soportar transacciones de corto y largo plazo, control de versión, colaboración y distribución de objetos Hipermedia. Ejemplos de sistemas de Hiperbase son **HAM** [19] y **HB1-4**[20].

Gran parte del grupo de desarrolladores, e investigadores, de estas dos líneas de investigación (Servicios de *Link* e Hiperbases) de la era post Dexter, se unen en 1994 (ver figura 3.2) para formar el Grupo de Trabajo de Sistemas de Hipermedia Abierta, OHSWG (http://www.ohswg.org/).

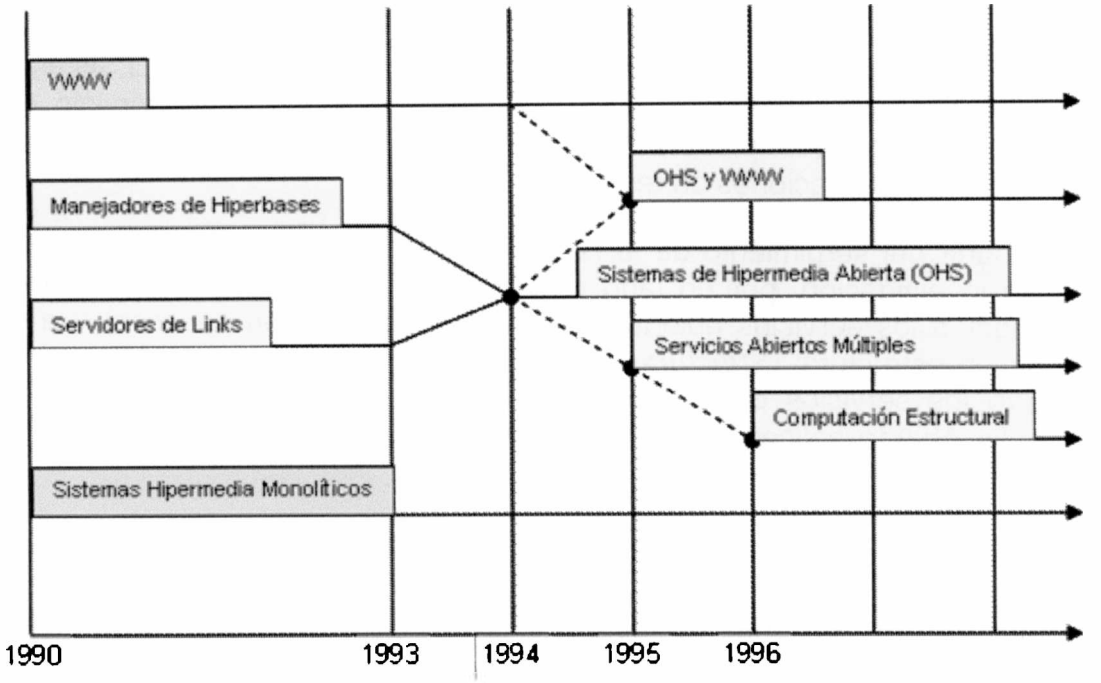


Fig. 3.2: Líneas de investigación surgidas a partir de 1990.

Una de las principales metas iniciales del OHSWG era la integración de aplicaciones de terceros, y este ha sido siempre el gran desafío para el desarrollo de Sistemas de Hipermedia Abierta. Además de esta meta inicial, también deseaban identificar qué constituía realmente la Hipermedia Abierta y buscaban crear un estándar común de modelo de dato, de modo que los Sistemas de Hipermedia Abierta pudieran interoperar.

El trabajo del OHSWG condujo a la creación de un modelo de dato y un Protocolo de Hipermedia Abierta (OHP). En la actualidad el OHP abarca un creciente numero de protocolos que cubren distintos dominios (espacial, taxonómico, etc.) y cuestiones de colaboración. El alcance de este trabajo no permite una descripción minuciosa del estándar, sin embargo en el apéndice B se presenta su DTD (*Data Type Definitions*) y se puede acceder a una descripción del mismo en http://users.ecs.soton.ac.uk/hcd/ohp/ohp.htm.

Los servicios de Hipermedia Abierta evolucionaron respecto de los servicios de vínculo y las Hiperbases, integrando aplicaciones estándar que no estuvieran escritas específicamente para el uso con Sistemas Hipermedia. Esta profunda integración de aplicaciones fue quizás alcanzada primero por **Microcosm** [21]. En este sistema la posibilidad de ubicar *links* dentro de un documento dependía del grado de "conocimiento o respeto" de *Microcosm* por parte de la aplicación controlada.

Una característica importante de *Microcosm* era que proveía *links* genéricos o globales (un punto extremo origen fijo y varios puntos extremos destino calculados) y con éstos un grado significativo de funcionalidad Hipermedia para cualquier aplicación sobre la plataforma de la estación de

trabajo. *Microcosm* avanzó respecto del requerimiento del servicio de *links* de Sun que imponía que las aplicaciones soportaran ciertos protocolos. El Sistema Hipermedia Devise Hypermedia (**DHM**) [22] es otro ejemplo de servicio de Hipermedia Abierta que proveía soporte para un conjunto de estructuras Hipermedia para aplicaciones de terceros.

A partir del surgimiento de la Hipermedia Abierta se generaron nuevos líneas de investigación, por un lado relacionadas con la Web y por el otro, dando lugar a los servicios abiertos múltiples (*Multiple open services*) y a la computación estructural (*Structural computing*). Cabe destacar en este punto que, tanto los servicios de *link*, los sistemas de Hiperbase y los servicios Hipermedia Abierta aplican variantes de los principios básicos del modelo de referencia de *Dexter*.

La investigación sobre los Sistemas de Hipermedia Abierta basados en componentes (*Component Based – Open Hypermedia System*) creció fuera de las primeras investigaciones de los SHA. Existen dos tipos distintos de CB-OHS. Por un lado el trabajo de los "*Multiple open services*" discutido primero en 1995 en conexión con **HyperDisco** [23], el cual proveia servicios de diferentes tipos de *link*, integración, distribución, y colaboración. Por otro lado el trabajo de la "*Structural computing*" fue discutido primero en 1996 en el escrito inicial de Numberg, **HOSS** [24]. En 1997, el proyecto *HOSS* reportó los primeros resultados en la provisión de diferentes tipos de servicio de estructura de Hipermedia (asociativa, espacial y taxonómica) respaldados por un "*hypermedia operating system*" (HMOS).

Sin embargo, los frutos de las dos líneas de investigación surgidas de la Hipermedia Abierta (CB-OHS y Computación estructural), presentan sus dificultades. Por un lado, y tal cual lo fundamenta Karousos en el 2004 [25], los CB-OHS no solo tienen el problema del costo, sino la dificultad en su utilización por parte de los desarrolladores para insertar funcionalidad Hipermedia en sus aplicaciones. Por otro lado *HOSS*, al llevar la funcionalidad Hipermedia a nivel del sistema operativo, está suponiendo que los usuarios migrarán de sus plataformas favoritas.

3.4.4 Cuestiones Pendientes

La posibilidad de proveer estructuras Hipermedia para tipos de dato arbitrarios, aunque es un importante avance, no es suficiente para las prácticas de la vida real. Si bien el soporte para adaptabilidad, colaboración y distribución no es una cuestión nueva para los investigadores de Hipermedia, ha comenzado a ser cada vez más vital. Las siguientes tres secciones revisan parte de la investigación histórica de estas cuestiones.

Adaptabilidad.

Los sistemas adaptables soportan modificaciones y mejoras de un sistema instalado por parte de los usuarios o de los desarrolladores que no

estaban involucrados en el desarrollo original del sistema. Ha sido considerada una característica crucial en muchos de los mejores Sistemas Hipermedia.

Los sistemas adaptables son capaces de aumentar o integrar nuevas aplicaciones con un servicio de Hipermedia Abierta instalado. Tal integración generalmente involucra adaptar las nuevas aplicaciones y el servicio de Hipermedia. La aplicación se aumenta con nuevos elementos de interfaz de usuario y procedimientos para comunicarse con el servicio Hipermedia, y el servicio se aumenta con objetos que modelan las nuevas aplicaciones

NoteCards [10] proveía un rango de opciones de adaptabilidad, desde hojas de preferencia hasta la posibilidad de crear nuevos tipos de tarjetas especializadas para algunas aplicaciones. **Intermedia** [12] también ofrecía soporte para la integración de nuevas aplicaciones aunque requería compilar nuevamente el núcleo del sistema. **HyperCard** (Bill Atkinson, Apple) empleaba el lenguaje de programación HyperTalk, que soportaba una amplia variedad de adaptabilidad. Por último **KMS** [9] incluía un lenguaje de *script*, como lo hacían **MultiCard** [17] y los navegadores y servidores de la Web.

Cooperación

Augment fue el primer sistema en ofrecer soporte para trabajo cooperativo, un video (http://sloan.stanford.edu/mousesite/) muestra a Engelbart, su creador, en 1968 realizando escritura cooperativa en tiempo real en una capa Hipermedia compartida con un colega a una distancia de 30 millas. *KMS* también posibilitaba la colaboración operando en una especie de modo cliente-servidor que permitía al usuario compartir parte de su estructura Hipermedia. *Intermedia* y *NoteCards* hicieron avances en colaboración asincrónica, incluyendo anotaciones Hipermedia en escritos de coautoría y reorganización cooperativa de materiales de proyectos compartidos. Trabajos posteriores incluyeron soporte para colaboración asincrónica, como en **Sepia** [26] y para notificaciones de eventos, como las notificaciones de Hill en 1991 [27].

Sin embargo, el soporte de colaboración ha sido poco investigado en ambientes integrados (monolíticos), y los Sistemas de Hipermedia Abierta actuales son principalmente sistemas monousuario con poco o ningún soporte para manipulación de colaboración de estructuras y contenidos Hipermedia.

Distribución.

Antes del arribo de la Web la Hipermedia distribuida fue casi ignorada. A parte de un corto período en los '70 cuando *Augment* estaba disponible para autores colaborando sobre lo que era Arpanet, los Sistemas Hipermedia han usado generalmente almacenamiento centralizado. Esto es tan cierto para servicios de *link* de hoy en día, como *Microcosm*, como lo fue para los sistemas monolíticos de la era precedente. *Xanadu* preveía Hipermedia distribuida a lo

largo del mundo desde los ´60, pero no hay ningún sistema trabajando más allá de prototipos.

Con el advenimiento de la Web, se vio un renovado enfoque hacia Hipermedia distribuida. Los sistemas abiertos sobre las estaciones de trabajo se comenzaron a familiarizar con la Hipermedia global. Los primeros experimentos en distribuir estructuras Hipermedia a través de la infraestructura de la Web aparecieron en sistemas como **Hyperwave** [28], *Microcosm* y **Device Hypermedia** [22].

Los mayoría de los sistemas están aún en su infancia respecto a estas cuestiones de investigación, el anhelo de la comunidad Hipermedia, y de este trabajo, es que estos servicios de Hipermedia distribuidos cambien la orientación de la Web, desde sistemas de navegación, a ambientes Hipermedia totalmente desarrollados con soporte para colaboración y para el diseño de estructuras Hipermedia.

3.5 La Web

> *"My definition of the Web is a universe of network-accessible information, a means of human-to-human communication, and a space in which software agents can, through access to a vast amount of everything which is society, science and its problems, become tools to work with us."*
>
> Berners Lee 1996 [37]

La WWW (*World Wide Web* o como se ironiza en la comunidad Hipermedia, "*What is Wrong with the Web*"), apareció en escena a principios de los ´90, si bien es el Sistema Hipermedia más exitoso y ampliamente conocido representa una línea diferente de desarrollo que no tuvo en cuenta uno de los principios básicos de la evolución de la Hipermedia, la separación de estructura y contenido. En la Web las estructuras (solo *links* binarios asociativos unidireccionales) Hipermedia están incrustadas en los datos, es decir, en el material. Sin embargo, casi de repente, la simplicidad de la Web llevó la Hipermedia a usuarios finales de todo el mundo, pero los conceptos básicos de la Hipermedia tienen mucho más para ofrecer que direcciones incrustadas en texto HTML.

La Web utiliza *links* unidireccionales incrustados, mientras que diseñadores de Sistemas de Hipermedia Abierta proveen *links* externos bidireccionales y n-arios. La mayor ventaja de los *links* incrustados en la Web es su simplicidad, no hay necesidad de un servidor de *links* especializado; la Web solo tiene que manipular archivos ASCII con etiquetas. Esta simplicidad tiene su costo, solo el dueño de un documento puede crear *links* desde los documentos. Al mismo tiempo *links* a partes especificas de un documento se pueden realizar sólo si hay etiquetas destino en el punto deseado del documento, y para un dado documento solo puede haber un conjunto de *links*.

Si dos usuarios desean tener distintos *links* desde el mismo documento, tienen que mantener copias separadas del documento, lo que implica mantenimiento extra y potencial inconsistencia. Además si un documento no es HTML, debe ser convertido (lo delata la existencia de convertidores de RTF a HTML, por ejemplo) antes de ser usado en un contexto Web.

Los *links* externos son considerablemente más complicados, pero ofrecen numerosas ventajas. Por ejemplo, el documento original no se afecta por la creación de *links*, los usuarios pueden crear, mantener y compartir sus propios *links* de un documento dado, en forma sencilla. Como los *links*, en un sistema basado en el modelo de *Dexter*, son objetos de primera clase en una base de datos, se pueden seguir *links* desde y hacia un documento consultando la base de dato de *links*. Los *links* pueden tener nombre y tipo. Además los *links* externos pueden tener más de un destino (n-arios). Finalmente como los *links* se manejan de modo externo al documento, los documentos pueden tener distintos formatos.

Luego del advenimiento de la Web, se ha buscado mezclar las ideas de servicios de *link* e Hiperbases con la Web. La línea de Sistemas de Hipermedia Abierta y Web trata las cuestiones de diseño emergidas por la tensión entre sofisticadas nociones y la funcionalidad de la Hipermedia Abierta basadas en el modelo de *Dexter* y el más simple (pero de mínima funcionalidad Hipermedia) y altamente escalable enfoque de la Web.

Asimismo nuevas recomendaciones emitidas por la W3C como XLink [29] y XPointer [30] han tratado de abordar la pobreza funcional Hipermedia de la Web. Sin embargo, por un lado, existen pocos navegadores e implementaciones que los soporten y por otro la comunidad de Hipermedia Abierta se ha expresado en favor de formatos más estudiados y difundidos dentro de dicha comunidad como FOHM [31] y OHIF [33].

Gronbaek y otros en el año 2000 [33], destacaron el formato de intercambio de Hipermedia Abierta OHIF por sobre Xlink, argumentando que el primero tiene un modelo de dato más rico, el cual permite mecanismos de estructuración y localización más sofisticados. En este trabajo, Gronbaek destaca que Xlink no soporta estructuración taxonómica ni espacial, ni localización en video, audio, hojas de cálculo, etc. Además OHIF ha sido adoptado por Sistemas de Hipermedia Abierta relevantes como **WebVise** [34] y **Arakne** [35], y existen trabajos realizados por la comunidad de Hipermedia Abierta que permiten un mapeo entre OHIF y Xlink, tal es el caso de Xspect de Christensen en 2003 [36].

3.6 Conclusiones

Los precursores de la Hipermedia (Bush, Engelbart y Nelson) preveían un universo de material que incluyera soporte para distintas formas de vinculación (dinámicas y distribuidas) y que permitiera navegar, buscar, comentar, estructurar y colaborar. Sin embargo aún hoy son pocos los usuarios que disfrutan de un soporte Hipermedia que permita esta funcionalidad.

Los sistemas monolíticos como la Web, los CDs multimedia de autoría y los Sistemas Hipermedia clásicos analizados (*NLS/ Augment, KMS, Intermedia, HyperCard* y *NoteCards*) requieren casi total control sobre el formato de dato, los algoritmos de generación y la interfaz del usuario. Todos estos sistemas tienen limitaciones y anomalías en la forma en que soportan el trabajo con conocimiento llevado a cabo por una comunidad de investigadores. Se destacan a continuación las tareas que son difíciles, sino imposibles, de realizar con este tipo de sistemas:

- Catalogar o manipular información de los *links* independientemente de los documentos en los cuales son activados. Esto impide mantener, actualizar y ampliar el sistema o, por ejemplo, proveer facilidades para una navegación gráfica.
- Insertar *links* en documentos existentes. Se requiere derechos de autoría, siendo que, en general, los documentos están disponibles en formato digital antes de que puedan convertirse a hipertexto.
- Integrar otras aplicaciones. Corren, generalmente, como programas independientes y no tienen comunicación bidireccional con otras aplicaciones. Esto impide que el usuario utilice sus aplicaciones favoritas.
- Agregar vínculos hacia fuera del sistema con la posibilidad de "volver" al mismo. Hay ocasiones en que es posible crear un *link* hacia fuera del sistema, pero se arriba a un punto muerto, es decir, no es posible continuar con funcionalidad Hipermedia.
- Integrar material al cual se tenga solo derecho de lectura. No es posible agregar vínculos en CDROMs y archivos de solo lectura.
- Integrar documentos producidos por otro sistema. Tienen un formato de documento determinado por el sistema.

En particular la Web, a través de las recomendaciones de XLink y XPointer emitidas por la W3C, ha tratado de abordar su pobreza de funcionalidad Hipermedia, pero tal cual se planteó en la sección 3.5, por un lado existen pocos navegadores e implementaciones que los soporten y por otro la comunidad de Hipermedia Abierta ha mostrado las ventajas de formatos más estudiados y difundidos como FOHM y OHIF por sobre estas recomendaciones.

Los Sistemas de Hipermedia Abierta, y la investigación que en ellos subyace, son los más adecuados para brindar soporte al ambiente planteado en el capítulo 2. Sin embargo tal cual se planteó en la sección 3.4.4, están en la infancia en cuestiones de adaptabilidad, cooperación y distribución. Además,

en un intento de brindar soporte utilizando alguno de los SHA analizados, estos revelaron tener algunas o varias de las siguientes dificultades:

- Son productos comerciales costosos, lo que dificulta el acceso a los mismos.

- Están preparados para un sistema operativo y un conjunto específico de aplicaciones a integrar.

- Intentan brindar máxima funcionalidad Hipermedia, lo que conlleva una gran y compleja infraestructura con funcionalidad extra innecesaria para el soporte específico requerido en el ambiente de investigadores planteado.

- Presentan problemas de usabilidad, ya que en sus interfaces de usuario utilizan terminología como: *global link, global anchor* y *composite*, a la cual el usuario común no está acostumbrado.

- No permiten la creación cooperativa de contenidos y estructuras, además la base de dato Hipermedia debería no soporta el monitoreo de cambios y la notificación distribuida a usuarios que necesiten estar informados.

- Son productos de código cerrado. Esta es una dificultad crucial para este trabajo (y para cualquier trabajo de investigación), ya que se busca el logro de experticia en un área de conocimiento, y esto en una ciencia fáctica conlleva no sólo conceptualización y teorías, sino desarrollos e implementaciones. El código abierto permite además experimentar con nuevas propuestas, realizar adaptaciones para nuevos clientes, plataformas y funcionalidades, y sirve de base para trabajos de investigación futuros.

Dadas las carencias y dificultades expuestas de los distintos Sistemas analizados y presentados, entre los cuales se incluyen las aplicaciones de escritorio (manejadoras de contenidos), se propone abordar el diseño detallado de un soporte tecnológico que considere las principales contribuciones de investigación y desarrollo de la comunidad Hipermedia, e incorpore las funcionalidades que cubran las dificultades identificadas en el ambiente de investigación presentado en el capítulo 2. Esto es, en el convencimiento de la potencia que tiene el enfoque de la Hipermedia Abierta para el manejo de conocimiento y para la integración de aplicaciones a través de redes, plataformas y ambientes heterogéneos.

Capítulo 4

El modelo Dexter de Referencia de Hipertexto

Resumen

Dado el impacto que el modelo Dexter produjo en el diseño de Sistemas de Hipermedia Abierta y dado que forma parte de los fundamentos de este trabajo, en este capítulo se presenta la conformación del grupo Dexter y se brinda una perspectiva de las metas del grupo. Seguidamente se presentan las conclusiones obtenidas por el grupo, en forma de requerimientos, y los problemas que diseñadores relevantes de Hipermedia enfrentaron al intentar desarrollar sistemas que cumplieran con el modelo. El capítulo cierra con las conclusiones del modelo a las cuales arribó este trabajo.

4.1 Introducción

"The Dexter Hypertext reference Model provides standard hypertext terminology coupled with a formal model of the important abstractions commonly found in a wide range of hypertext systems. Thus, the Dexter Model serves as a standard against which to compare and contrast the characteristics and functionality of various hypertext (and non hypertext) systems. The Dexter Model also serves as a principal basis on which to develop standards for interoperability interchange among hypertext systems."

Halasz and Schwartz, 1994 [48]

A través de la historia de la Hipermedia, hubo varios intentos de estandarización, desde el nivel conceptual y desde el nivel de intercambio e interoperatividad. La estandarización es importante por varias razones. Primero, acordando claramente qué es Hipermedia, es más sencillo abordar y comparar trabajos de investigación. Segundo, distintos Sistemas Hipermedia pueden intercambiar documentos entre sí e incluso pueden trabajar en conjunción. Tercero, si existe un estándar adoptado por varios Sistemas Hipermedia, resulta atractivo para los desarrolladores adoptar dicho estándar, con lo que se extiende soporte de Hipermedia para más aplicaciones y refuerza el desarrollo de sistemas, tal cual lo describe Whitehead (1999) [49].

En cuanto a la estandarización a nivel conceptual, gran parte de la historia del diseño Hipermedia fue capturada en el modelo de referencia de Dexter. Este innovador modelo conceptual incluye una especificación formal de entidades hipertexto y sus relaciones, como así también una descripción informal y sus motivaciones. Dado que la mayor parte de los integrantes del grupo Dexter tiene gran peso propio dentro de la Hipermedia, y dado que el modelo influenció el trabajo posterior de cada uno, se presentan a continuación sus integrantes.

4.2 Conformación del Grupo Dexter

Los encuentros del grupo comenzaron en 1988 con John Leggett y Jan Walker quienes invitaron a diseñadores de Hipermedia, los que hubieren participado en desarrollo de Sistemas Hipermedia innovadores (ver tabla 4.1), a la posada Dexter en Sunapee (USA). Los integrantes eran desarrolladores de 17 sistemas (13 diseñadores) los cuales cubrían una amplia variedad de filosofías y estrategias de diseño de Hipermedia. (Los sistemas más relevantes que no estuvieron representados fueron Guide, 1987 [50] y Xanadu, de Nelson, 1981.

Participantes	Sistemas Hipermedia
Organizadores: John Legget, Jan Walker	
Rob Akscyn, Don McCracken	ZOG, KMS
Doug Engelbart	NLS, Augment
Steve Feiner	IGD
Mark Frisse	Dynamic Medical Handbook
Frank Halasz, Randy Trigo	NoteCard, TextNet
Norm Meyrowitz, Karen Smith	FRESS, Intermedia
Tim Oren	HyperCard
Amy Peral	Sun Link Service
Catherine Plaisant, Bill Weiland	HiperTies
Mayer Schwartz	Neptune, HAM
Jan Walter	Concordia, Document Examiner

Tabla 4.1: Participantes de los encuentros en Dexter y sistemas que representaban.

Debido a la diversidad de formas de abordar y diseñar de estos sistemas fue razonable definir qué era Hipermedia y qué tenían en común estos sistemas.

Luego del primer encuentro hallaron sorprendentes conexiones entre sus trabajos. En el segundo encuentro hicieron una representación gráfica de la historia en el campo de la Hipermedia. El gráfico que hicieron era en si mismo una estructura Hipermedia. Cada sistema se representaba con una línea cuyo largo correspondía a su duración en tiempo. Dos tipos de flechas conectaban las líneas, una representa el ancestro directo y otra los sistemas que tuvieron menos influencia directa.

Luego de varios encuentros Halasz y Schwartz se pusieron a trabajar en las conclusiones del grupo para armar un modelo formal de referencia, el cual se presentó en el Instituto nacional de estándares y tecnología en 1990 [7].

4.3 Metas del grupo Dexter

En un principio la meta no era el diseño de tal modelo, sino la de comparar experiencias y arribar a una terminología compartida que pudiera acompañar la variedad de aparatos conceptuales en el campo. En el proceso el grupo discutió la historia y lo que se estaba llevando a cabo en los proyectos Hipermedia que ellos estaban desarrollando.

Fue más tarde que se avocó a la tarea de diseñar, en conjunto, un modelo de referencia de hipertexto. Eventualmente sus intenciones fueron más allá de la creación de una terminología común y un marco de trabajo conceptual, hacia:

- Unificar las nociones de nodo y link.
- Aumentar las redes basadas en link con otras estructuras.
- Integrar Hipermedia con ambientes y editores de contenido existentes
- Representar e imponer diferentes modelos de datos y comportamiento de tiempo de ejecución.
- Almacenar propiedades de presentación de nodos (*pSpecs*).
- Intercambiar hipertexto entre sistemas.
- Definir a qué se considera Hipertexto / Hipermedia.

4.4 El decálogo Dexter para el diseño de Hipertexto

"Los que se enamoran de la práctica sin la teoría son como los pilotos sin timón ni brújula, que nunca podrán saber a dónde van."

Leonardo Da Vinci

El modelo de referencia que resultó de la deliberación del grupo cumple más o menos con las metas planteadas. Esta sección lista las principales conclusiones del modelo de referencia de Dexter. Dada la importancia y la cantidad de conclusiones a que arribó el grupo, en este trabajo se las

denomina "decálogo". El grupo articuló estas conclusiones como requerimientos [7] [51], y explica las premisas que las subyacen. Algunas de las conclusiones fueron controversiales, unas pocas fueron nuevas y la mayoría eran heredadas de uno o más sistemas existentes y generalizadas para su aplicación. Se presentan a continuación dichas conclusiones:

1 - No se puede esperar que los Sistemas Hipermedia manejen el contenido del material cuyas interconexiones soporta. En la mayoría de los casos, esta es una cuestión de aplicaciones que están fuera del control del Sistema Hipermedia.

Fue claro para el grupo que mientras la mayoría de los sistemas utilizaban el termino *link*, el nombre de las entidades conectadas por los *links* del Sistema Hipermedia variaba de un sistema a otro. Estas entidades eran llamadas nodos, tarjetas, documentos, archivos, etc. Considerando la meta fundamental de la Hipermedia, la integración, era poco sorprendente que mucho del material a ser integrado existiera y tuviera su propia terminología.

Dado el reconocimiento, por parte del grupo, de la variedad de material preexistente a ser interconectado, determinó que el manejo de ese material estaba fuera del alcance del Sistema Hipermedia. Estas observaciones surgieron de Pearl Amy quien experimentó con el Sun Link Service, posiblemente el primer Sistema de Hipermedia Abierta integrador.

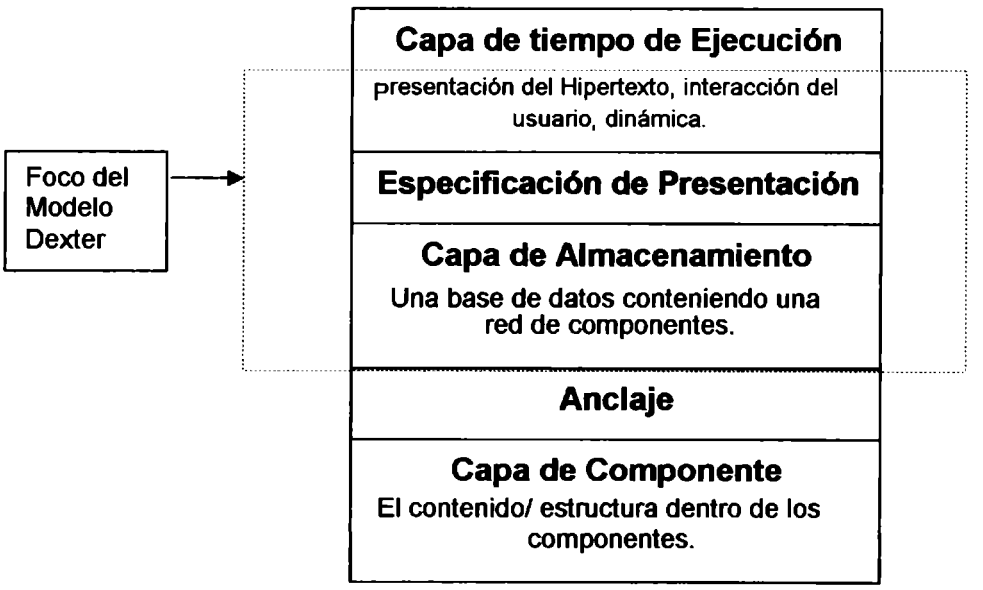


Figura 4.1: Arquitectura de 3 capas de Dexter.

La arquitectura de alto nivel del modelo Dexter (figura 4.1) refleja el reconocimiento del grupo de las limitadas posibilidades de un Sistema Hipermedia. Adoptando la terminología de sistemas operativos, el grupo definió la capa de componentes, "*within component layer*", que es la parte del

ambiente en línea (documentos "conectados" al hipertexto) con el usuario que esta bajo el control de programas de aplicación (componentes fuera del alcance del Sistema Hipermedia). Esta decisión de diseño, marcó el primer momento en el que la teoría de Hipermedia tomó una postura antimonolítica.

2 - Un marco conceptual para Hipermedia debe distinguir entre aspectos del almacenamiento del material electrónico y su comportamiento de tiempo de ejecución.

Otro aspecto importante para el grupo, y para la mayoría de los investigadores de ciencias de la computación, lo constituye la necesidad de distinguir el almacenamiento de la información, de las interfaces de tiempo de ejecución de esa información, en otras palabras, entre la persistencia de materiales electrónicos y sus comportamientos.

Nuevamente el grupo concibió esta distinción en términos de capas. En este modelo las capas son divisiones arquitecturales entre partes del sistema como así también entre áreas de competencia. En particular el modelo distingue entre una capa de almacenamiento y una de tiempo de ejecución. En la sección 4.5 se plantea (basándose en trabajos de la comunidad Hipermedia) cómo el concepto de capas lleva a hacer problemáticas asociaciones al lector del modelo.

3 - Anclaje de links: conectar los puntos extremos de los links al contenido, constituye gran parte de la acción en un Sistema Hipermedia. Por lo tanto, el anclaje debe estar explícitamente representado en el modelo.

Uno de los problemas más importantes que tomo el grupo fue el anclaje de links. Esto es, dada la premisa *1* del decálogo, que concluía que el sistema no podía controlar el contenido del material: ¿Cómo se crean y manejan links que conecten puntos dentro de ese contenido?

Con este fin el grupo generalizó la noción de bloque de Intermedia, una estructura de dato representando puntos o fragmentos en el material de los documentos en línea. El resultado fue el ancla, un objeto abstracto que divide los limites entre las capas de almacenamiento y de componente.

4 - Ciertos aspectos particulares del comportamiento de tiempo de ejecución de los links y los nodos deben persistir a través de los usos del sistema y por lo tanto deben ser almacenados con el contenido y la estructura Hipermedia.

El grupo definió la especificación de la presentación (*pSpecs*) como la entidad asociada con todo componente, que retiene información relacionada al comportamiento en tiempo de ejecución del componente y de sus vecinos vinculados.

Al igual que las anclas, las *pSpecs* dividen los límites entre las capas de tiempo de ejecución y de almacenamiento. El ejemplo más comúnmente utilizado para justificar las *pSpecs* involucra una variación de la interfaz del

material dependiendo del link que el usuario atraviese para acceder al mismo. La figura 4.2 ilustra un escenario de una tutoría asistida por computadora, mostrando dos formas de arribar a una animación, una utilizada por un estudiante que debe visualizar la animación, y otra por el profesor quien por defecto debe editarla.

Para el ejemplo de la figura, las *pSpecs* almacenadas con cada *link* capturan la distinción para representar las dos opciones. De este modo, las *pSpecs* proveen una forma de recordar y almacenar opciones realizadas sobre comportamiento de ejecución. Hay una *pSpec* para el profesor, quien ve la animación en un editor de animaciones y otra *pSpec* para el estudiante, quien ejecuta la animación.

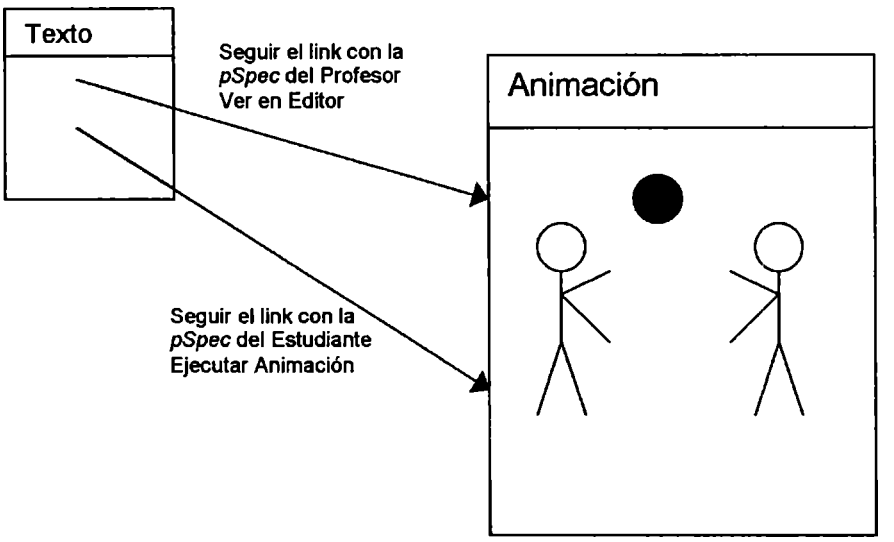


Figura 4.2: Dos links con diferentes instrucciones (*pSpecs*) para la interfaz destino.

5 - Las estructuras, diferentes a los *links, deben ser representadas como partes totalmente desarrolladas del modelo.*

Frank Halasz afirmaba en su escrito "*seven issues*" (un hito en 1988) que la mayoría de las aplicaciones Hipermedia requerían estructuras que iban más allá de las redes puramente basadas en links. Estas estructuras incluían jerarquías, tablas, relaciones de bases de datos, objetos con atributos, etc. Más aún, argumentaba la construcción de esas representaciones en forma separada de los links. Proponía una nueva entidad Hipermedia denominada compuesto (*composite*), para representar estructuras que no estuvieran basadas en *links*. La noción de compuesto fue tomada por el grupo y se convirtió en una de sus características más conocidas. En la visión del modelo Dexter los compuestos y los links son equivalentes en status.

6 - Los distintos objetos involucrados en la representación de estructuras Hipermedia deben ser especializaciones de una única entidad genérica, la cual modele: el material involucrado, los links utilizados para interconectarlo, y los compuestos utilizados para estructurarlo.

En lugar de adoptar el término más utilizado a la fecha, nodo, el grupo eligió el término "componente" para expresar un concepto que unificara las ideas de link y de nodo (ver figura 4.3). Esta consolidación en el modelo facilitó la tarea de capturar los elementos comunes entre componentes links y componentes no links, suministrando una simple entidad con la cual modelar los aspectos estructurales y de comportamiento comunes a todos los objetos modelados. La asignación de identificadores únicos podía entonces ser definida una única vez para el objeto Componente, asegurando así la cobertura de todos los objetos en la Hipermedia. Además los links podrían conectar otros links o componentes no link.

Esta idea era atractiva a muchos investigadores, aunque aplicaciones prácticas de la misma eran difíciles de imaginar en ese momento. Finalmente el componente era utilizado también para modelar compuestos (material organizado con estructuras no link).

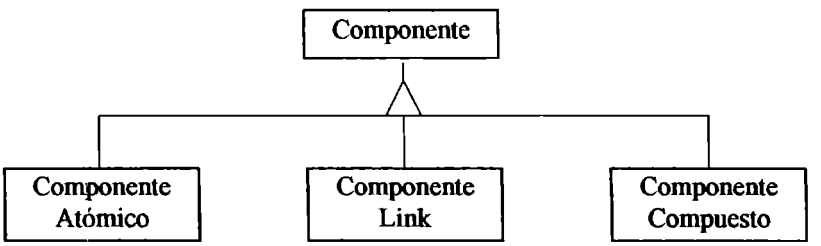


Figura 4.3: una descripción orientada a objeto de los conceptos originales de Dexter.

El modelo distingue tres tipos fundamentales de componentes: atómicos, link y compuestos. Los atómicos representan contenido que podría, generalmente, ser manejado por una aplicación de tercero. Eran llamadas atómicas para resaltar el hecho de que la estructura de ese contenido no era visible para el Sistema Hipermedia. Los links y los compuestos eran las estructuras manejadas por el Sistema Hipermedia.

7 - Los Links con puntos extremos calculados se modelan mejor separando el trabajo de especificar una localización, del de acceder al material de esa localización.

A través de los años, muchos investigadores de la comunidad de Hipermedia habían argumentado en favor de la noción de un link cuyo destino sea calculado. En lugar de depender de puntos extremos fijos, los links calculados ejecutan un pequeño programa para determinar el resultado de una

travesía[1]. Aunque pocos sistemas hasta esa fecha habían utilizado el concepto, el grupo acordó que cualquier modelo comprensivo, completo, debía ser capaz de representar *links* calculados y *links* fijos. El grupo arribó a una generalización potente que tendría una amplia aplicación.

Primero, decidieron que el cálculo esté asociado con los puntos extremos de los *links* (en lugar de estar asociados al link completo). Esto significa que un origen de un link puede ser calculado o que sus múltiples puntos extremos pueden ser calculados simultáneamente.

Segundo, el modelo distingue entre especificar localizaciones y acceder a ellas. De hecho, estas actividades están plasmadas en el modelo por dos funciones: resolver (*resolver)* y acceder (*accessor*). Resolver modela la identificación de puntos extremos a través del procesamiento de *scripts* ejecutables como así también la identificación de puntos extremos fijos. El resultado del cálculo es un identificador único de un componente (UID), el que opcionalmente incluye un identificador de ancla para localizar una posición dentro del contenido de ese componente. El UID es manejado por la función acceder, la cual devuelve el componente y opcionalmente el ancla en caso de que el punto extremo no identifique la componente completa.

Este modelo de cálculo fue particularmente apropiado para soportar links cuyos cálculos de puntos extremos involucraban búsquedas en la Hipermedia por componentes con ciertos atributos.

8 - La direccionalidad debe estar asociada con los puntos extremos del link, además los links deben tener un número ilimitado de puntos extremos.

En un intento de cubrir los sistemas de los participantes del grupo, discutieron la semántica y el comportamiento al atravesar los links.

Algunos sistemas como HyperCard implementaban *links* como sentencias *goto* incrustadas. En este Sistema cada *link* era calculado, en el sentido de que su destino se hallaba corriendo el *script* Hypertalk correspondiente. Estos *links* eran unidireccionales dado que la travesía hacia atrás, desde el destino al origen del link, era casi imposible. Este también es el caso de los links HTML en la Web, lo que provocó que algunos se refirieran a la Web como la versión distribuida de HyperCard. Por otro lado Intermedia representa el otro extremo, los links eran objetos bidireccionales separados, en los cuales ninguna dirección era privilegiada.

Como con el caso de links calculados, el grupo decidió asociar direccionalidad con punto extremo de link y adoptaron cuatro constantes de direccionalidad: hacia, desde, ninguna y ambas (*TO, FROM , NONE, y BOTH*).

[1] Travesía: En Hipermedia un link es una entidad separada y hay que atravesarla para ir desde un punto extremo origen a un punto extremo destino. Comúnmente en la Web se habla se seguimiento en lugar de travesía, dado que la dirección de destino esta incrustada en el html.

Aunque el modelo no ofrecía una semántica para todas las posibles asignaciones de direccionalidad de los puntos extremos, el grupo fue capaz de modelar la semántica de HyperCard, Intermedia y la de los otros sistemas. Además el grupo decidió incluir links con más de dos puntos extremos en el modelo, una decisión que consideraron importante para sistemas futuros.

La figura 4.4 representa un componente link con tres puntos extremos que interconectan tres componentes. El componente atómico de la parte izquierda actúa como origen del link (el punto extremo tiene direccionalidad *FROM*) mientras que las otras dos componentes actúan como destinos (direccionalidad *TO*).

En la figura 4.4 hay un componente de cada tipo. Las flechas entre componente (compuesta y atómica) y componente base indican la función acceder. Resolver a, identifica un componente que puede tener ciertos atributos y un identificador de ancla de esa componente. Toda componente tiene atributos, *pSpec* y anclas, aunque en la componente link, anclas esta vacía.

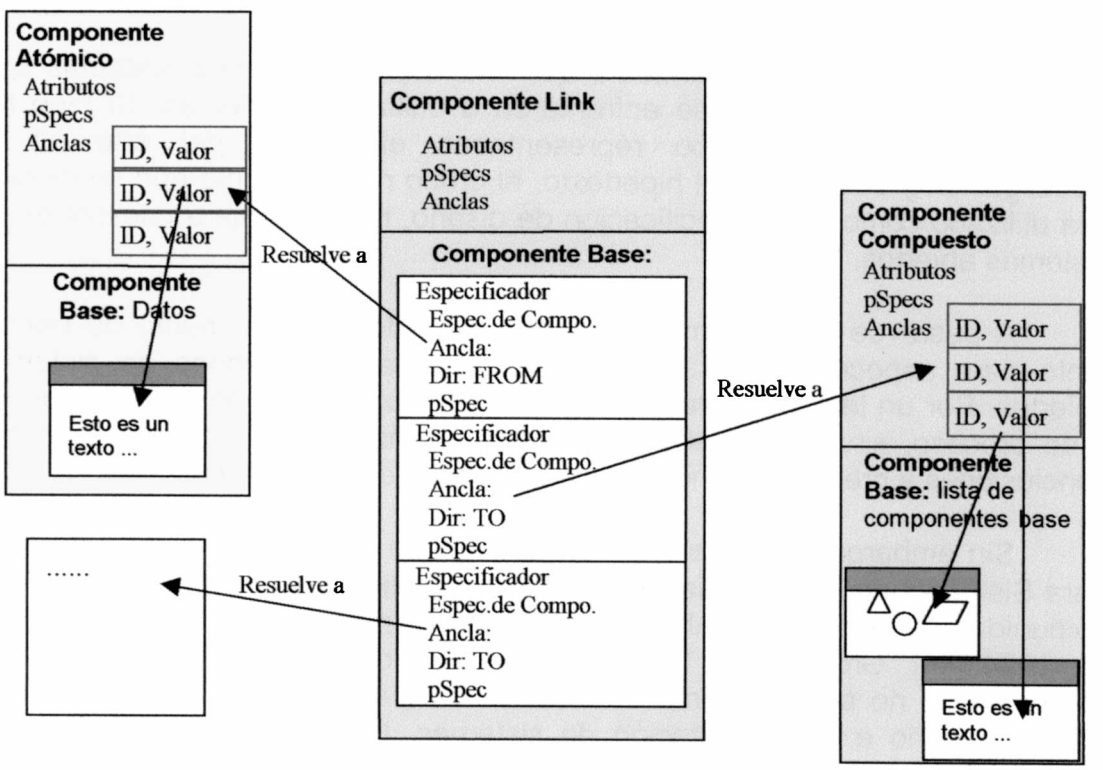


Figura 4.4: Un link en Dexter y sus relaciones con otros componentes.

9 - Cualquier Sistema Hipermedia debe ser capaz de exportar una representación de la estructura y el contenido, en una forma estándar que permita que otros Sistemas Hipermedia importen esa representación.

Una meta clave en el diseño del modelo fue soportar el intercambio entre Sistemas Hipermedia. Sin embargo, ningún participante tenía experiencia en el traslado de representaciones entre sistemas. El primer experimento en el tema fue de Leggett y Schnase en 1994 [47].

10 - Todas estas premisas (conclusiones) deben estar representadas formalmente en un modelo de referencia que permita que los Sistemas Hipermedia sean comparados, en términos de sus capacidades, entre sí y con respecto a un estándar definido.

El grupo intentaba que su modelo provea un buen "sello de aprobación", de modo que el diseño de futuros Sistemas Hipermedia pudieran ser comparados con el modelo Dexter con una especie de chequeo de cumplimiento. Para posibilitar tal comparación el grupo representó formalmente el modelo utilizando Z, una sintaxis formal para modelos de referencia [46]. Además de representar el marco de trabajo conceptual, este formalismo también expresaba ciertas reglas de buen comportamiento Hipermedia, la más conocida era la de los links colgados ("*dangling links*").

4.5 Problemas y cuestiones pendientes

Los diseñadores Hipermedia que intentaron desarrollar sistemas que cumplieran con el modelo se enfrentaron a distintos problemas. Si bien las recomendaciones del grupo representaban el estado del arte en la investigación y desarrollo del hipertexto, el grupo nunca intentó que su modelo sea utilizado como una especificación de diseño, ni concernía a cuestiones de sistemas abiertos.

A pesar de esto, el modelo Dexter ha sido un buen punto de partida tanto para especificaciones de diseño como para cuestiones de sistemas abiertos. Por un lado su clara articulación de recomendaciones arquitecturales y de proceso, sirve como entrada para el diseño; y por otro, varias de sus conclusiones y premisas coinciden con la agenda de los sistemas abiertos.

Sin embargo, el modelo Dexter tal cual se planteaba no era adecuado para Sistemas de Hipermedia Abierta. En tal sentido, hay varios trabajos de la comunidad de Hipermedia Abierta (Gronbaek y Trigg, 1996 [45], Gronbaek et. Al, 1994 [44], Gronbaek y Trigg, 1994 [43], Gronbaek, 1994 [40]) que han comprobado, no solo las fortalezas del modelo, sino también las debilidades para el diseño e implementación de sistemas. Luego de su publicación, el modelo fue extendido para soportar prácticas de trabajo cooperativo y distribuido y, lo más importante, para soportar sistemas abiertos e integración.

Se detallan a continuación los problemas y cuestiones pendientes que han sido identificados y reconocidos por la comunidad Hipermedia (trabajos citados en el párrafo anterior) al intentar diseñar e implementar sistemas que cumplieran con el modelo. Vale la pena aclarar nuevamente, que a pesar de la validez de las críticas que se describen a continuación, el grupo nunca intentó

que su modelo sea utilizado como una especificación de diseño ni concernía con Sistemas Abiertos.

1- ¿Cuál es el rol del modelo de referencia de Dexter en el diseño?

El hecho de que el modelo fue formalizado utilizando la sintaxis de Z posibilita al lector comparar las capacidades y comportamiento de su sistema. Pero el sólo uso de Z, no sirve como una especificación para el diseño de sistemas.

2- ¿Qué es una capa?

Modelar un Sistema Hipermedia con las tres capas mostradas en la figura 4.1 fue un paso importante para soportar integración con aplicaciones existentes. Pero esta arquitectura es incómoda e inadecuada como base para una arquitectura de sistemas abiertos concretos.

El primer problema encontrado por la comunidad de Hipermedia fue el término capa, que muchos asociaron con sistemas operativos. Las capas de almacenamiento, componentes y tiempo de ejecución nombran los tres niveles más altos concernientes al modelo Dexter. Aunque éstas se solapan en formas interesantes, no están ordenadas por grado de abstracción como lo están por ejemplo las capas ISO del sistema operativo.

Además hay problemas con la yuxtaposición de capas en el modelo. Aunque la ubicación de almacenamiento entre las capas de tiempo de ejecución y de componentes ayuda a recalcar los límites entre estas y las interacciones y objetos utilizados para manejar esos limites, imponen dos limitaciones importantes.

La primera limitación es que las capas de tiempo de ejecución y de componentes están muy separadas en el espacio conceptual, lo que implica que las aplicaciones involucradas para los componentes son responsables de la definición y el almacenamiento de contenido, pero no del comportamiento en tiempo de ejecución de las componentes. Sin embargo, la forma integradora de abordar de los sistemas abiertos requiere que las aplicaciones se puedan hacer cargo además, de las interfaces de tiempo de ejecución (las paginas HTML o los documentos de texto tienen comportamiento de tiempo de ejecución suministrado por los navegadores y los procesadores de texto respectivamente).

Una segunda limitación del modelo reside en la ubicación establecida al concepto ancla sobre el borde y entre las capas de almacenamiento y de componente, que sugiere que las anclas no están involucradas en comportamiento de tiempo de ejecución. De hecho una cuestión importante de la investigación de Hipermedia Abierta involucra el manejo apropiado de actualizaciones en tiempo de ejecución para las anclas, cuando los usuarios modifican los contenidos de los componentes (por ejemplo, agregan texto a un

componente y las anclas cambian su localización). Las nociones de componente y ancla, deben estar incorporadas en las capas de almacenamiento y de tiempo de ejecución.

3- ¿Deben permitirse componentes híbridos?

Aunque la unificación al considerar nodos y links como componentes brindaba nuevas posibilidades a diseñadores de Hipermedia, como por ejemplo, la posibilidad de links a links. Irónicamente, la noción de componente incluía la restricción de que un objeto Hipermedia debía ser: un link, un compuesto o una componente atómica. De hecho había experiencia dentro del grupo Dexter con componentes híbridos, como en NoteCards que empleaba las *FileBox*, unos componentes compuestos que representaban estructuras jerárquicas, pero también atómicas, ya que podían ser utilizadas como un componente de texto. Tales componentes híbridas son difíciles de representar en el modelo Dexter.

4- ¿Quién almacena el contenido de un componente?

Uno de los avances del grupo Dexter fue el modelado de componentes como envoltorios, que ayudaban a distinguir las responsabilidades de los Sistemas Hipermedia de aquellas responsabilidades de las aplicaciones (que manejaban el contenido). Sin embargo, el modelo pasó por alto el hecho de que algunas aplicaciones necesitan manejar el almacenamiento de sus contenidos. Las aplicaciones multimedia actuales son un buen ejemplo, generalmente necesitan almacenar grandes archivos de datos que requieren de algoritmos de compresión complejos. Sería inapropiado requerir que el Sistema Hipermedia se hiciera cargo del manejo de tales formas de datos. Una aproximación ideal para Hipermedia Abierta debe ofrecer soporte de almacenamiento solo para aquellas aplicaciones que lo necesiten, de otro modo deben manejar solamente el almacenamiento de envoltorios de componentes, así como también de links y compuestos.

5- ¿Exactamente, qué compone un componente compuesto?

Uno de las contribuciones más conocidas del grupo fue la elevación del compuesto, utilizado para modelar estructuras no basadas en link, al mismo nivel que el link.

Los compuestos contienen sus componentes constitutitos. Desafortunadamente esta definición de compuesto refleja solo un tipo de compuesto, sin tener en cuenta ciertos casos comunes importantes. El compuesto de Dexter fue pensado para modelar componentes que tuvieran estructura interna, dentro de la cual se pudieran requerir links. El ejemplo favorito del grupo, fue un componente gráfico, cuyo editor maneja la distribución y construcción de grupos de objetos gráficos. La intención fue

permitir el anclaje de links en objetos gráficos particulares (objetos independientes dentro del gráfico) como se muestra en la figura 4.5.

Sin embargo este limitado sentido al compuesto no soporta el modelado de otro tipo de componentes. Por ejemplo, estructuras solapadas de otro tipo de componentes al estilo de los "*tours guiados*" de NoteCards. Las colecciones como compuestos, aparecen con Halasz en su escrito "*Seven Issues*", aunque no con ese nombre. Una definición más completa de compuesto debería modelar no solo la noción de la estructura de intra componente sino la de inter componente representada por compuestos de tipo colección.

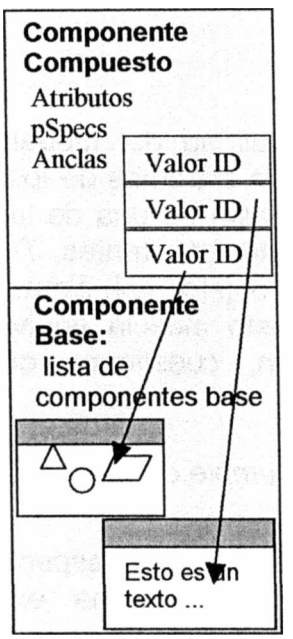


Figura 4.5: El componente compuesto de Dexter.

6- ¿Qué es un ancla?

El anclaje fue ampliamente reconocido como un valioso aporte del modelo. El grupo reveló que en el concepto de anclaje estaba el corazón de las metas fundamentales de la Hipermedia: integrar información en línea. Sin embargo la definición y utilización de anclas en el modelo muestra algunos inconvenientes.

Uno de esos inconvenientes es que, aunque el grupo soporta el cálculo para hallar los componentes identificados por un punto extremo de un link, se asumía que la localización del ancla dentro de esos componentes no involucraba ningún cálculo. Esto significa que el modelo no tiene en cuenta una forma particular y muy útil de ancla que especifica búsquedas dentro del contenido de su componente, sumamente importante en Sistemas Abiertos en el que el contenido puede cambiar fuera del alcance del Sistema Hipermedia.

Otro inconveniente es que la definición de ancla es deficiente aún para anclas que especifican su localización directamente (sin cálculo). El caso de un compuesto con varias capas de estructuras anidadas ilustra el problema. La definición de Dexter no dice nada acerca de cómo identificar un objeto anclado anidado dentro de una estructura. Por ejemplo, supongamos que estamos implementando un compuesto colección. Una de las opciones es cargar anclas locales en cada componente de la colección, que entonces son identificadas por anclas en el compuesto contenedor. Esta forma de abordar tiene la ventaja de mantener anclas locales a los componentes a las que se refiere. Tal referenciado indirecto desde unas anclas del compuesto a anclas de sub componentes, no es soportada por el modelo Dexter.

7- ¿Qué camino toma un link?

La noción de direccionalidad del modelo asocia apropiadamente un atributo dirección con los puntos extremos de los links, y no con los links como un todo. Desafortunadamente esta es una de las áreas en las que el modelo aporta más confusión. Las cuatro constantes, *TO, FROM, BOTH* y *NONE* son utilizadas para distinguir entre objetos link almacenados y el estilo *goto* de los links incrustados. Debido a esto mezcla equivocadamente las nociones de direccionalidad e incrustación, cuestiones que deberían ser totalmente independientes.

8- ¿Que Sistema Hipermedia cumple con el modelo Dexter?

El modelo Dexter representa una especie de estándar contra el cual comparar. Sin embargo, ningún sistema existente, incluyendo aquellos desarrollados por los miembros del grupo, cumplía con Dexter. El grupo consideró la posibilidad de cumplimiento parcial con el modelo, por ejemplo reconociendo que el requerimiento de que los links nunca pudieran estar "colgados", ni siquiera temporalmente, era muy estricto. Pero nunca se especificó una semántica de cumplimiento parcial con el modelo.

Las reglas de comportamiento forzadas por el Modelo son imposibles de cumplir y poco razonables en contextos de sistemas abiertos. Volviendo al ejemplo más notorio, la regla contra los links colgados, el modelo requiere que la eliminación de un componente dispare la eliminación de todos los links conectados a ésta. En un ambiente distribuido no se puede forzar tal requerimiento, al menos para aquellos elementos almacenadas en servidores inaccesibles. Más aún, la eliminación puede suceder fuera del alcance del Sistema Hipermedia.

Además de esta dificultad planteada para la regla contra los links colgados, en el contexto de Sistemas Abiertos los links colgados son útiles en ciertas situaciones. Por ejemplo se podría crear un link antes que su componente destino, o antes del fragmento del componente donde el link tendrá su punto extremo anclado. Los links colgados son útiles también cuando

se copian subredes de un Hiperespacio (subhiperespacios). Más que prohibirlos, los Sistemas de Hipermedia Abierta soportan su utilización, por ejemplo, manejándolos como recordatorios de trabajo por realizar.

4.6 Conclusiones del modelo Dexter

A pesar de los problemas y las cuestiones encontradas por los desarrolladores Hipermedia que intentaron utilizarlo, el modelo representó y representa una base fundamental y un punto de partida para el diseño de Sistemas de Hipermedia Abierta. Las aportes mas trascendentes del modelo fueron:

- El planteamiento de una arquitectura general de Hipermedia.
- La clara especificación de qué responsabilidades le corresponden a un Sistema Hipermedia.
- La concientización de que nodos, links y compuestos son componentes de primera clase (y todos pueden ser apuntados por un link).
- El reconocimiento de la importancia del anclaje para la integración.
- La generalización del link, de binario a n-ario (multicabeza).
- La generalización del compuesto.
- El reconocimiento de la necesidad de separar aspectos de almacenamiento del material, de los aspectos de comportamiento.

Más allá de las extensiones y modificaciones que el modelo requirió para ser utilizado en el diseño de Sistemas, este trabajo reconoce que el grupo Dexter marcó un hito trascendental para la Hipermedia, de hecho ha sido fuente de inspiración y fundamento de diversos trabajos de relevancia ([47] Leggett and Schnase 1994, [41] Gronbaek 1993, [42] Gronbaek et. Al 1993, [34] Gronbaek et. al 1999, [40] Gronbaek y Malhotra 1994, [32] Bouvin et. Al 1997, [44] Gronbaek et. al 1994).

Capítulo 5

Fundamentos de Diseño en Hipermedia Abierta

Resumen

En este capítulo se presentan los fundamentos de diseño, los cuales han sido fruto de años de investigación y desarrollo de la comunidad de Hipermedia Abierta. Se describe en primer término la operación básica de los SHA y luego la arquitectura conceptual para soportar dicha operación. Seguidamente se describe la conceptualización subyacente en el diseño, incluyendo detalles de la evolución de las entidades conceptuales descriptas. El capítulo cierra con la consideración de las entidades conceptuales involucradas en el comportamiento de un Sistema en tiempo de ejecución, describiendo además las consideraciones básicas para el manejo de la presentación y de la travesía.

5.1 Operación básica de los SHA

Desde el punto de vista conceptual, Hipermedia concierne a encontrar, crear y seguir conexiones de material en línea. En Hipermedia Abierta este material ya existe y además pertenece a un ambiente que tiene tiempo de ejecución independiente del Sistema Hipermedia. El comportamiento agregado por un SHA involucra la creación y navegación de conexiones, representadas por estructuras como links, comentarios y compuestos. Las aplicaciones continúan siendo responsables del manejo del contenido de los documentos.

Los SHA aumentan las aplicaciones favoritas de los usuarios a través de un servicio de Hipermedia, el cual provee estructuras almacenadas en una

base de datos separada del contenido representado en los distintos documentos generados por esas aplicaciones. Esta separación permite que el usuario trabaje con sus aplicaciones favoritas y además pueda crear dinámicamente las estructuras, e incluso pueda colaborar y compartir estos útiles metadatos con otros usuarios.

El almacenamiento de las estructuras en una base de datos separada implica la localización vía anclas (objetos separados referenciados por los links), las cuales permiten encapsular información de localización dentro de los documentos, incluso sobre cualquier tipo de medio (texto, gráfico, sonido, etc.). Este tipo de localización sólo está limitada por los métodos que las aplicaciones, utilizadas por los usuarios, provean para acceder a fragmentos de los documentos que manejan. Además dado que los documentos originales no se alteran por la creación de estructuras, el usuario puede crear, mantener y compartir sus estructuras sobre cualquier tipo de documento, incluso sobre documentos de solo lectura.

Otra ventaja de la separación de la estructura y el contenido es que se pueden utilizar formas más ricas de estructuración que el simple link binario unidireccional asociativo, clásico de la Web. Esto permite que el usuario pueda crear estructuras como colecciones y jerarquías, e incluso pueda relacionar documentos almacenados localmente con documentos basados en la Web.

Las estructuras son recuperadas de la base de datos y mezcladas dinámicamente con los documentos de las aplicaciones de terceros, es decir, las localizaciones de los links y los comentarios son resaltadas en forma similar a como se resaltan los links en la Web. Para estructuras tales como las colecciones, resulta necesario que el SHA provea alguna interfaz de modo que el usuario pueda accederlas y editarlas.

El hecho de almacenar las estructuras por separado de los contenidos de los documentos implica, por un lado una infraestructura adicional (un Servidor y una Base de datos Hipermedia), y por el otro genera problemas de inconsistencia potencial de links y demás estructuras. Por ejemplo, si un documento es modificado fuera del alcance del SHA, se pueden generar puntos extremos de links "colgados". Este problema es una cuestión general de los SHA y se pueden adoptar diferentes criterios de diseño para abordarla.

Por último, los SHA son adaptables, lo que permite integrar nuevas aplicaciones con el Servicio de Hipermedia. Tal integración involucra adaptar las nuevas aplicaciones de terceros a ser integradas y adaptar el Servicio de Hipermedia. La aplicación es adaptada aumentándola con nuevos elementos de interfaz de usuario que proveen acceso a la funcionalidad Hipermedia e invocan procedimientos del Servidor. El Servicio de Hipermedia se aumenta con objetos que modelan las nuevas aplicaciones. Por lo expuesto, parte del éxito de los SHA depende de la apertura de las aplicaciones que se integran.

Dado lo crucial que resulta la operación de seguimiento de link en un entorno de Hipermedia Abierta, y a modo de brindar información que ayude a interpretar tal operación en el entorno de un SHA, a continuación se describen los pasos conceptuales de la travesía de un link:

1. La aplicación utilizada por el usuario comunica el requerimiento de atravesar un link al SHA, a causa de un clic del usuario en un ancla o por algún otro evento que tomó lugar en la aplicación.

2. El SHA "resuelve" el link y determina el o los archivos del otro extremo del link que deberán ser presentados. Tal cual se ha discutido, los SHA permiten links con múltiples puntos extremos, por lo que se pueden vincular múltiples documentos con un origen dado, en cuyo caso, se repite desde el paso 3 para cada archivo del otro extremo del link.

3. El SHA puede realizar algunas de las siguientes tareas para mostrar el archivo:
   a. Requerir que una aplicación habilitada para Hipermedia (integrada), que estaba en ejecución, muestre el archivo.
   b. Disparar una aplicación habilitada para Hipermedia que muestre el archivo.
   c. En caso de que ninguna aplicación habilitada para Hipermedia pueda mostrar el tipo de archivo dado, el SHA dispara una aplicación no habilitada para Hipermedia que lo muestre. En este último caso el usuario ha alcanzado un punto muerto, es decir, un documento sin links disponibles, por lo que los pasos 4 a 7 no se ejecutan para este archivo.

4. Basado en la información obtenida del SHA, la aplicación habilitada para Hipermedia abre el archivo requerido (por ejemplo recuperándolo desde el sistema de archivo).

5. La aplicación habilitada para Hipermedia envía un mensaje al SHA requiriendo las anclas del archivo mostrado.

6. El SHA responde con una lista de anclas.

7. La aplicación habilitada para Hipermedia muestra las anclas y posiciona el documento en el ancla particular que fue conectada por la travesía del link. A partir de lo cual, el usuario puede seguir los links disponibles desde el nuevo archivo mostrado.

5.2 Arquitectura conceptual de los SHA

La arquitectura física de los SHA ha sido ampliamente discutida por la comunidad de Hipermedia, incluso en trabajos específicos como el de Madsen y otros en 1994 [44] y el de Gronbaek y Wiil en 1997 [22]. La coincidencia es que, más allá de la arquitectura física, para poder operar y proveer las funcionalidades descriptas en la sección anterior, la arquitectura debe estar

básicamente organizada en cuatro capas conceptuales. Estas cuatro capas, utilizadas por la mayoría de los Servicios de Hipermedia Abierta descriptos en el capítulo 3 (sección 3.4.3), se describen a continuación:

- Aplicación: representa el espacio de aplicaciones de terceros que pueden integrarse con el Sistema de Hipermedia Abierta. Dada la importancia otorgada al contenido existente, y a los editores de contenido existentes, este ambiente se modela en forma independiente del SHA.

- Almacenamiento: almacena y recupera los objetos Hipermedia que deben persistir. Representa una base de datos que contiene una red de nodos (*componentes*) y vínculos (*links* y demás estructuras)

- Tiempo de ejecución: permite que los usuarios puedan navegar y manipular las estructuras Hipermedia, incluso en forma cooperativa. Se encarga de la presentación de las estructuras, de la interacción con el usuario y captura la funcionalidad para manipular y acceder a las estructuras almacenadas. Es la responsable de manejar links, anclas y estructuras en tiempo de ejecución. Esta capa tiene responsabilidad solamente del comportamiento dentro de lo que es Hipermedia, incluyendo navegación y presentación, y es en el mayor grado posible, independiente de las interfaces de las aplicaciones.

- Comunicación: encapsula la comunicación con las aplicaciones de terceros, es decir con las que radican en la capa de aplicación, proveyendo una interfaz uniforme para dichas aplicaciones.

5.3 Conceptualización en el diseño de SHA

> *"Our experience... has encouraged us to practice a design philosophy of voluntary simplicity striving to make do with fewer concepts and mechanisms."*
> Akscyn et al. 1988, p.834. [44]

El diseño de un Sistema de Hipermedia Abierta incluye no solamente links y demás estructuras agregadas a material existente, sino también una cuidadosa atención a cuestiones conceptuales subyacentes tales como las de localización, anclaje e interconexión.

5.3.1 Localización, anclaje e interconexión

Un tema crucial en Hipermedia Abierta involucra los detalles de cómo los links y demás formas de estructurar (almacenados en forma externa) se refieren a (localizan) fragmentos de material y cómo son accesibles desde (anclajes) el material manejado por las aplicaciones. En esta sección se introducen tres conceptos que son fundamentales para el trabajo de integrar Hipermedia con aplicaciones y material: localización, anclaje e interconexión.

Una localización es un puntero que identifica, más o menos precisamente, puntos o fragmentos de material. Por medio de tales direcciones, las estructuras Hipermedia son capaces de referirse al material.

Un anclaje es un mecanismo de interfaz de usuario que provee acceso a las estructuras Hipermedia (a fin de atravesarlas) desde fragmentos del material. El anclaje generalmente necesita estar resaltado, de modo que resulte visible y accesible para el usuario.

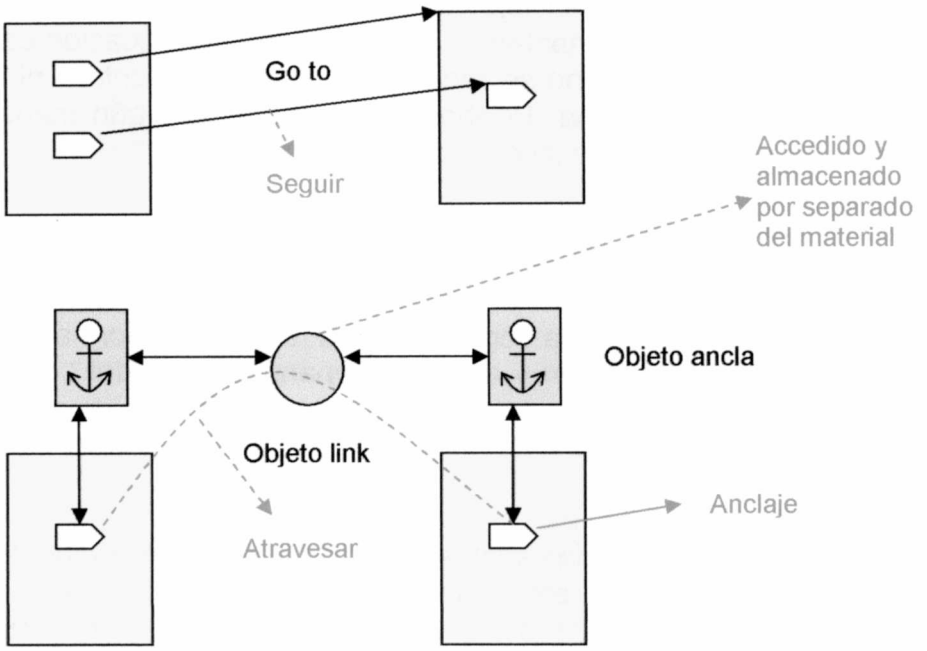


Figura 5.1: Dos tipos de interconexiones: direcciones incrustadas y objetos links externos.

Una interconexión es la forma de brindar instancias de estos dos conceptos fundamentales (localización y anclaje) para crear estructuras atravesables. En Hipermedia Abierta se habla de estructuras atravesables y de atravesar links (en lugar de seguir links como se acostumbra en la Web) debido a que los links y demás formas de estructuración se almacenan en forma separada al material. En la figura 5.1 se aprecia gráficamente la diferencia semántica que existe entre "seguir" y "atravesar" un link, en la Web y en un Sistema de Hipermedia Abierta respectivamente.

En cada una de las siguientes secciones se describe cómo estos tres conceptos elementales interactúan para proveer los fundamentos de diseño de un Sistema de Hipermedia Abierta.

5.3.2 Separando link de anclaje

Los primeros Sistemas Hipermedia, como Augment, HyperCard y KMS (incluso la Web), se enfocaron en el trabajo de localizar fragmentos de material en línea. No se consideraba el problema del anclaje dado que la entidad

estructural (link) estaba directamente incrustada en el material (como una URL incrustada en HTML). En estos sistemas para agregar un link, se debía tener acceso de escritura al material, pero no existía el problema del anclaje, se accedía directamente a las entidades de estructuración.

Una de las grandes contribuciones del grupo Dexter consistió en la determinación de que el uso de links para modelar diferentes tipos de conexiones debía separarse de la cuestión del anclaje. En el modelo las interconexiones se capturan en estructuras cargadas en forma separada del material, de este modo los anclajes pasan a ser referencias no incrustadas. La localización del anclaje se mantiene utilizando una especificación calculable (un localizador). En este caso no se requiere derecho de escritura ni modificar el material, aunque esta forma de abordaje requiere de algún mapeo desde el material a las referencias no incrustadas y viceversa.

La figura 5.1 muestra la diferencia entre estas dos formas de interconectar. En la parte superior de la figura, toda la información acerca del link se carga en el lugar de su origen (dentro del contenido del material). En la parte de inferior se muestra la segunda forma, similar a la forma de Intermedia, en la cual los links se cargan y acceden separadamente del contenido de los componentes vinculados. En este último caso los links interconectan objetos anclas, y estos objetos anclas manejan las localizaciones de los puntos extremos del link.

Las entidades llamadas anclas, como se definen en el modelo de Dexter, actúan como objetos de referencia para los links y son responsables de encapsular la información de localización de fragmentos de material dentro de un componente. Un Ancla consiste en un identificador, único dentro del componente al que pertenece, y un valor fijo que especifica un lugar o fragmento (*span*) en el material del componente. Los *links span-to-span* son aquellos que van de un fragmento de un componente a otro fragmento de otro componente. Para conseguir estos *links*, es que el modelo Dexter proveyó de una entidad de direccionamiento indirecto denominada ancla. El grupo Dexter considera que la forma de localizar varía de una aplicación a otra y por lo tanto no especifica tipos de valores para las anclas.

Un inconveniente con el modelo Dexter es que si bien reconoce la importancia del cálculo en Hipermedia, asocia soporte de cálculo sólo a los puntos extremos de los links, no a las anclas. Esto genera inconvenientes de inconsistencia cuando se modifica el contenido del material desde afuera del Sistema de Hipermedia. Además, en Dexter, no estaba prevista la especificación componentes enteros como anclas.

5.3.3 Localización avanzada: locSpecs y refSpecs

El grupo Dexter, además, separó y diferenció la operación para identificar un componente, de la operación para acceder a un componente. Para esto definieron dos operaciones abstractas que denominaron acceder y

resolver. La operación acceder es la que accede a un componente (almacenado en una base de datos) dado su ID, mientras que la función resolver permite, a partir de las especificaciones del componente, obtener los ID correspondientes. Por lo tanto, acceder constituye un direccionamiento directo a los componentes, mientras que resolver se utiliza para un direccionamiento indirecto a partir de especificaciones.

La especificación de localización que utiliza resolver puede ser cualquier atributo de un componente (por ejemplo, el nombre) o un *script* que realiza una búsqueda sobre la base de alguna consulta provista por el usuario. El grupo Dexter dejó abierto el formato preciso de las especificaciones ya que dependen del tipo de material en cuestión. Separar las operaciones resolver y acceder les permitió entremezclar cálculo (resolver) con direccionamiento directo (acceder) y asociar cálculo a los puntos extremos de los links en lugar de asociarlo a los links como un todo.

Gronbaek y Trigg en 1996 [45] presentaron una extensión del modelo Dexter para soportar, entre otras cosas, cálculo en anclas y compuestos[1]. Simplemente reprodujeron el formalismo de especificación de los puntos extremos propuestos por el grupo Dexter, en las anclas y en los compuestos. Para ello utilizaron una nueva entidad, la *locSpec*, especificación de localización, y reconstruyeron el interior de las entidades fundamentales del modelo Dexter para utilizar esta nueva forma uniforme de especificar localizaciones.

Una *locSpec* es la especificación de una localización que puede "vivir" en forma independiente del Sistema Hipermedia y del material, por lo tanto se parece a una URL, sin embargo, está diseñada para soportar cálculo, localización sobre distinto tipo de medios, localizaciones robustas y tiene un formato que brinda facilidades para ser empaquetada dentro de un objeto y para ser referenciada en un programa.

Del mismo modo que una URL puede transportase como *strings* de texto e incorporarse dentro de un Sistema Hipermedia (así como una URL puede incluirse en un correo electrónico). En principio una *locSpec* puede también incluir atributos arbitrarios. Esta forma de localización permite modelar localizaciones incrustadas, como en el caso de la Web, y localizaciones externas, como en el caso de los Sistemas de Hipermedia Abierta.

La localización a través de *locSpec* permite identificar fragmentos dentro del material, aún para material representado en distintos medios (texto, gráfico, video, etc.), y posibilita la localización estática (fija) y dinámica (que pueda calcularse en cada acceso). La localización dinámica es de suma utilidad para la integración de aplicaciones de terceros, ya que el material puede modificarse fuera del alcance del Sistema Hipermedia.

---

[1] Compuesto: componente que está compuesto por otros componentes. Permite estructurar material y se describe en detalle en la sección 5.3.4.

Una *locSpec* envuelve varios tipos de convenciones de nombres para localizaciones, y está compuesta de dos partes, una para localización dentro de un componente y otra para localización del componente entero. La localización dentro de componente provee tres formas de especificar una posición o fragmento:

- Identificador (ID) de fragmento: especifica una localización a través de un nombre (único dentro del componente y conocido por la aplicación que lo maneja), por ejemplo, un ID de un ancla, un nombre definido en HTML o un ID de una selección o de un objeto dentro de un gráfico.

- Descripción de estructura: es una forma de especificar localización que describe cualquier estructura que el material en el componente pueda tener, por ejemplo, para un fragmento de texto (*span*) podría ser: desde el carácter 148 al carácter 192; o, en caso de un video, desde el *frame* 1240 al *frame* 1244.

- Especificación de cálculo: provee una forma de empaquetar cálculo en una *locSpec*. Puede ser una consulta en la forma de un *string* (buscar un texto), una descripción de formato o en forma general un *script* ejecutable como podría ser un *applet* de Java.

Para *locSpecs* simples, en general, se utiliza solo una de estas 3 formas de localización dentro de componente, sin embargo, la especificación de la misma localización de múltiples formas también es valiosa, y es lo que permite una especificación robusta. Por ejemplo, se podría identificar un fragmento de texto en un componente a través de la especificación de su estructura y además, utilizando la especificación de cálculo con una búsqueda del texto correspondiente. De este modo se podría detectar si el documento ha sido modificado fuera del control del Sistema Hipermedia e incluso reparar las inconsistencias que esto podría provocar.

Además de las tres formas de especificar localizaciones dentro de componentes, las *locSpecs* incluyen tres formas de especificar los componentes completos (como un todo):

- Un identificador (ID) de componente: único dentro de un particular Hiperespacio[2].

- Una descripción de estructura: se utiliza cuando hay estructuras entre componentes, es decir, cuando los componentes forman parte de otro componente (Generalmente para identificar el compuesto al que pertenecen).

[2] Hiperespacio: red de material en línea. En la sección 5.3.4 se describe con mayor detalle.

- Una especificación de cálculo: posibilita la ejecución de, por ejemplo, consultas de búsqueda por atributos de componentes (fecha de creación). También se pueden especificar *scripts* generales para localizar uno o más componentes. Una *locSpec* podría identificar los componentes creados en una fecha determinada, es decir, podría identificar varios componentes.

Como en el caso de localización dentro de componentes se pueden utilizar estas tres especificaciones en forma redundante para localizar el mismo componente o conjunto de componentes.

A lo sumo una de las dos partes de la *locSpec* puede estar vacía, ya sea la de especificación del componente o la especificación de localización dentro del componente. Si la parte vacía corresponde a la especificación del componente significa que la *locSpec* es aplicable a cualquier componente, si la parte vacía corresponde a la especificación de adentro de componente, significa que la *locSpec* identifica el componente entero.

Un ejemplo de instancia de una *locSpec* con un solo valor en cada una de sus dos partes constitutivas (la de componente y la de dentro de componente), podría ser: www.pape.com/home.html#titulo2, la cual tiene:

- Un identificador de componente, la página www.pape.com/home.html.
- Un identificador de objeto dentro de componente, titulo2.

Gronbaek y Trigg definieron además una nueva entidad denominada *refSpec* (especificación de referencia), que posibilitó la utilización de *locSpecs* dentro de un Sistema Hipermedia. La *refSpec*, si bien es independiente del material, existe solamente si el Sistema Hipermedia está en ejecución, es decir, es una entidad del Sistema Hipermedia que empaqueta (envuelve) una entidad (*locSpec*) que existe independientemente del mismo.

Una *refSpec* podría ser un objeto definido de acuerdo a la jerarquía de clases en un diseño de un Sistema de Hipermedia. Las *refSpecs* proveen el fundamento para estructurar, es decir, son la base para las anclas y para las referencias de puntos extremos de links y compuestos. Permiten construir estructuras que "vivan" aparte del material y constituyen la forma de empaquetar las *locSpecs* y almacenarlas en bases de dato separadas.

Estas nuevas formas de localización, *locSpec* y *refSpec*, ampliamente aceptadas[3] por la comunidad de Hipermedia Abierta, forman la base para cuestiones fundamentales de la Hipermedia Abierta, como lo son localización, anclaje e interconexión, y representan la línea divisoria entre el almacenamiento, y lo que en Dexter se denominaba capa de interior de componente (capa de aplicación).

---

[3] El trabajo en que fueron presentadas [45] estuvo nominado para el premio Douglas Engelbart al mejor *paper* en HYPERTEXT '96.

Las *locSpecs* y *refSpecs* enriquecieron las entidades del modelo Dexter, asociándose a los puntos extremos de los links y a las anclas (tal cual se presentan en la sección 5.3.4), permitiendo de este modo que las anclas y los compuestos sean calculables, algo no previsto por el modelo Dexter y necesario para mantener la consistencia de las anclas cuando el material es modificado fuera del Sistema Hipermedia y para los compuestos que requieren cálculo. Estas nuevas entidades proveyeron además la base para discusiones sobre travesías y estructuras, en Hipermedia Abierta.

5.3.4 Entidades Conceptuales en el diseño de SHA

En esta sección se describen las distintas entidades conceptuales que subyacen en el diseño de SHA, junto con la evolución y las distintas concepciones que han sufrido. Las entidades presentadas han sido propuestas por la comunidad de Hipermedia Abierta (www.ohswg.org) y surgen de un relevamiento de:

- La propuesta del OHP 2.0 (*Open Hypermedia Protocol*, Davis y otros en 1996).

- El modelo Dexter y las distintas extensiones realizadas al mismo (Gronbaek y Trigg en el 1996 [45], Gronbaek y otros en 1993 [41]; Leggett y Schnase en 1994 [47]; Gronbaek y Trigg en 1994 [43]; Gronbaek en 94 [14]).

- Los Sistemas de Hipermedia Abierta más relevantes (Anderson y otros 1994 [18], Hall y otros 1996 [53], Gronbaek y otros 1997 [32] [34], Nurenberg y otros 1996 [54], Wiil y Legget 1996 [23], Bouvin [52]).

Componentes

Los sistemas monolíticos tienen una vista de la Hipermedia centrada en el link. De hecho generalmente llaman "dato" al cuerpo de material en línea subyacente, como si éste existiera sólo para servir a la operación de vinculación o como si la Hipermedia consistiera de contenedores vinculados que necesitan ser rellenados. Esto encubría la variedad de material inherente y reflejaba la meta de los sistemas monolíticos, soportar todos los trabajos de los usuarios bajo el Sistema Hipermedia.

En la Hipermedia centrada en el link, los diseñadores reflejaban su visión de cómo el material debía ser organizado. Desde material dividido en trozos relativamente pequeños, muchas veces denominados "tarjetas" que se vinculaban como un todo, hasta material dividido en uno o más trozos de tamaño arbitrario que podían inspeccionarse a través del "*scroll*". En este último caso el vinculado interconecta fragmentos (*spans*) de material, y el material era sencillo de imprimir en el orden que tenía por omisión.

El grupo Dexter, en lugar de especificar una metáfora (nodo, tarjeta, dato, etc.) o estilo de interacción, introdujo la noción de componente atómico.

Esto dejaba la responsabilidad del material en manos de los diseñadores de las aplicaciones. Un componente atómico podía, entonces, envolver un cuerpo de material de tamaño arbitrario. De esta forma proponía una división de tareas que separaban las responsabilidades de los desarrolladores de las aplicaciones de aquellas de los desarrolladores del Sistema Hipermedia.

El componente atómico de Dexter proveía una entidad que podía vincularse y además comunicarse con los programas responsables de abrir, navegar y editar su contenido. En algunas ocasiones borraban la línea que dividía el Sistema Hipermedia de la aplicación, por ejemplo, reconocían la necesidad de que el Sistema Hipermedia influenciara el comportamiento de la aplicación que manejaba el contenido del documento. Un ejemplo de la Web es el soporte de HTML para especificar la ventana o *frame* dentro de la cual se mostrará el destino de un link.

Independientemente del potencial que ofreció la propuesta de Dexter (utilizar el componente atómico como envoltorio), suponía que el Sistema Hipermedia se responsabilizaba del almacenamiento del contenido de los componentes y del comportamiento de tiempo de ejecución de los mismos, utilizando presentadores/ editores a los que denominaba instancias.

En una propuesta integradora, como en la Hipermedia Abierta, hay que considerar que las aplicaciones de terceros proveen sus propios ambientes para manejar presentación y edición de contenido, y además manejan el almacenamiento del mismo. De modo que un Sistema de Hipermedia Abierta debe posibilitar la estructuración dentro de interfaces existentes. Otro inconveniente con Dexter es que requiere que el material sea atomizado en tipos de entidades con capacidades mutuamente excluyentes, sin considerar la posibilidad de componentes híbridos.

En los Sistemas de Hipermedia Abierta la meta es la integración, por lo tanto se libera del problema de cómo conceptualizar de forma separada la variedad de material en línea que los Sistemas Hipermedia esperan vincular. En lugar de esto se considera cómo y hasta qué punto extender el sistema para comunicarse con las aplicaciones que manejan el almacenamiento y comportamiento del material.

En Hipermedia Abierta se concibe el componente como la entidad básica a través de la cual el material en línea se agrupa, interrelaciona y estructura, y se la denomina generalmente, al igual que en Dexter, componente atómico. Los componentes atómicos son entidades del Sistema Hipermedia que empaquetan[4] (incluyen o referencian) material que existe en forma independiente del Sistema, y reflejan la organización preexistente del material.

[4] Empaquetar: envolver un recurso, un nodo en este caso, dentro de un componente atómico, de modo que este último actúe como un *proxi* al recurso. Si, por ejemplo, cambia la URL del recurso solo se modifica un Nodo, y no todas las anclas que apuntaban a ese recurso.

Un componente atómico puede incluir, por ejemplo, un párrafo, en ocasiones se hace referencia al párrafo como el contenido del componente. Las entidades estructurantes, link y compuestos, también son considerados componentes, y se presentan como especializaciones de un componente genérico. Los componentes estructurantes permiten capturar estructuras que no estén reflejadas en el material en línea.

Una ventaja importante de esta noción unificada de componente es que las decisiones concernientes a la división de trabajo y el manejo del comportamiento de tiempo de ejecución, pueden realizarse sobre la base de la aplicación que maneja cada tipo de componente. Por ejemplo los componentes link son almacenados por el Sistema Hipermedia, mientras que la mayoría de los editores/visores de texto involucrados por los componentes atómicos textuales, son responsables del almacenamiento de los mismos.

Todos los componentes pueden incluir una colección estructurada de anclas, así como también contenido, el cual es un objeto de dato o una referencia a un objeto de dato almacenado en forma separada. Además, todos los componentes, correspondan o no a la estructura inherente del material, pueden tener tipos y atributos que permitan diferenciar la semántica y el comportamiento de los mismos. Por ejemplo, cualquier componente puede configurarse como temporal cambiando el valor de un atributo, en cuyo caso el componente no será almacenado en la base de datos Hipermedia.

Anclas

Las anclas soportan la vinculación desde y hacia puntos dentro del contenido de documentos manejados por aplicaciones de terceros. Un ancla (utilizando las nociones de localización avanzada) tiene empaquetada o asociada una *refSpec* que apunta al interior del contenido de un componente y es la responsable de mantener actualizada la localización dentro del componente al que pertenece, cuestión sumamente valiosa ya que el material del componente puede ser modificado fuera del alcance del SHA.

Las *refSpecs* pertenecientes a anclas nombran explícitamente el componente al que pertenecen a través de la especificación de su ID, ya que las anclas sólo pueden pertenecer a un componente. Las anclas tienen un ID, al igual que el resto de las entidades conceptuales, único dentro del componente al que pertenecen y además empaquetan una especificación de presentación, *pSpec* (propuesta de Dexter), que permite configurar la forma en que la misma se presentada al usuario.

Esta forma de concebir las anclas (dada la naturaleza extensible de las *locSpecs*), permite que las mismas puedan identificar fragmentos en distintos tipos de medios y que estos puedan ser lugares fijos o calculados. Esto resuelve el inconveniente de inconsistencia potencial del ancla de Dexter que tenía un valor fijo asociado. Sin embargo, dado que el ancla es una

especificación que necesita ser resuelta, es necesario algún tipo de mapeo entre esta referencia no incrustada y el material.

A diferencia de las anclas concebidas como direcciones incrustadas, en donde la misma aplicación que maneja el material mantiene la integridad de las mismas (si el material que incluye un ancla es borrado, el ancla es borrada), cuando se utilizan referencias no incrustadas, *refSpec*, la integridad de las anclas debe ser mantenida por el Sistema Hipermedia.

Las anclas actúan como puntos de entrada (conexiones potenciales) para estructurar. De este modo puede evitarse que los componentes estructurantes apunten directamente dentro del material, con las consecuencias que esto conllevaría para el mantenimiento. Sin embargo, pueden utilizarse también en componentes no atómicos, referenciando puntos de componentes estructurantes. Por ejemplo, un ancla en un componente link podría referirse a uno de sus puntos extremos, de esta forma, un Sistema Hipermedia podría soportar no solo links a otros links, sino que también links a puntos extremos de otros links, cuestión que tiene un gran potencial para formas de estructuración más complejas.

Otro inconveniente del modelo Dexter era su falta de soporte para anclas temporales, es decir, anclas que persisten por la duración de una operación de navegación. Un típico ejemplo, es un punto extremo origen calculado de un link genérico, que sucede cuando el usuario hace una selección en un editor y esta selección no corresponde a un ancla existente. El sistema, en este caso, debe crear un ancla temporal y buscar por links que tengan puntos extremos con origen coincidente con el ancla temporal creada. El ancla puede borrarse luego de que los links coincidentes se atraviesen. Con el diseño de ancla descripto, solo basta agregarle un atributo que indique su temporalidad.

Links: distintas perspectivas y cuestiones de travesía.

Los links han sido interpretados en una variedad de formas a través de los años y de los sistemas. Generalmente un mismo sistema soporta más de una perspectiva, por ejemplo, la Web utiliza links como punteros entre dos puntos asociados, y como una forma, generalmente jerárquica, de estructurar documentos. Dada la importancia del link en la Hipermedia, y con el objetivo de ayudar a interpretar el diseño actual de los mismos, se presenta a continuación una enumeración de las distintas concepciones y perspectivas de funcionalidad que se le han asignado a través de los años y de los distintos sistemas (para una descripción más detallada incluyendo los sistemas asociados, consulte el apéndice A):

- Dirección de destino incrustada en el material.
- Dirección legible.
- Localización a resolver a través de cálculos y/o *scripts*.
- Vinculación a material fuera de línea, por ejemplo a un archivo, cuando el archivo está disponible (en línea) se notifica al usuario.

- Links como botones, texto subrayado, objetos de un gráfico, etc.
- Links bidireccionales, sin ninguna dirección privilegiada (asociaciones).
- Links fríos, calientes y tibios.
    - Fríos: asociaciones tradicionales
    - Calientes: por ejemplo, asociación entre un texto y una celda de una planilla de cálculo. .El valor de la celda en el texto, se mantiene actualizado.
    - Tibios: asociaciones que se actualizan bajo el control del usuario.
- Links para capturar distintos tipos de estructura, la más común es la jerárquica. Se fuerzan reglas estructurales en tiempo de vinculación, por ejemplo, no puede haber ciclos ni componentes desvinculados a la jerarquía.
- Links al componente más cercano en una distribución física (gráfica) de componentes (involucra cálculo).
- Links para estructuras de argumentación, en la que se mantienen agrupados sus componentes. Para este tipo de utilización se le asignaban tipos a los links.
- Links genéricos, globales o calculados: generalmente para material textual. La idea es que cualquier palabra hallada en cualquier documento puede servir como origen de un link. El seguimiento del link causa que la palabra de origen sea utilizada en una consulta sobre una base de datos de material vinculado, por ejemplo, en un diccionario.
- Links que en sus puntos extremos tienen una semántica de travesía. No solo las semánticas de punto origen y destino clásicas de los links, sino distinto tipo de relaciones, como por ejemplo: la afirmación en el componente del punto extremo origen "es soportada" por el argumento del componente del punto extremo destino. En este caso la travesía del link en sentido inverso, requiere cambiar la semántica de "soporta" a "es soportado".

Uno de los objetivos en el diseño de Hipermedia Abierta es utilizar una entidad conceptual que permita implementar una rica variedad de perspectivas, como las descriptas anteriormente, acerca de la funcionalidad del link.

Independientemente de la interpretación de los links, todos soportan alguna forma de travesía. Hay varias cuestiones a considerar en cuanto a la travesía. Una es la de si una travesía debe causar que el destino del link reemplace el material que está siendo mostrado o no. Esta opción puede especificarse en el mismo link asociándole (como al resto de los componentes) una *pSpec*, por ejemplo, que por defecto indique reemplazar (o mostrar) el destino en una nueva ventana. Hay una idea más compleja que es mantener dos ventanas vinculadas sincronizadas durante procesos como el *scroll*.

Otra cuestión importante para el diseño de soporte de travesía es lo que George Landow en 1987 [55] denominó retórica de salida y de arribo de un link. Un ejemplo de la retórica de salida es la vista previa, un resumen del destino del link que puede mostrarse rápidamente para ayudar al usuario a decidir si

atravesar el link o no. Una variante más simple es la de la muestra de los navegadores Web, de la URL de destino del link al poner el cursor sobre el origen. Otra variante es representar visualmente, en un navegador gráfico, los vecinos locales del documento activo, en este caso, el documento actual se muestra junto con los documentos que están a un link de distancia.

Por otro lado, la retórica del arribo concierne a la mejor forma de proveer contexto, para un usuario que ha seguido un link y ha arribado a un nuevo lugar en el material interrelacionado. Aunque las paginas Web pueden incluir extensiva e informativa retórica de salida, las representaciones de contexto de arribo raramente se soportan. Una aproximación es proveer un navegador gráfico del tipo "usted está aquí", que muestre la estructura de la red que circunda, similar al mapa local de Intermedia. Tales navegadores, generalmente, se construyen y mantienen en forma manual por los *webmasters*, pero son usualmente accesibles sólo desde los nodos raíces de un sitio, además son muy costosos de mantener.

Muchas de las distintas funcionalidades y cuestiones de travesía se abordaban utilizando los clásicos links binarios diseñados para soportar asociaciones. Sin embargo Halasz, el diseñador original de *NoteCards*, en su escrito "*seven issues*", argumentaba en contra de la sobrecarga de los links con semántica para representar estructuras distintas a las referencias y asociaciones para los que originalmente estaban diseñados, en su lugar proponía soportar distintas estructuras utilizando lo que denominó compuestos. Los compuestos (se describen más adelante) son los que se utilizan actualmente en Hipermedia Abierta para manejar este tipo de estructuras y son además la base de diseño para los navegadores gráficos que permiten abordar las retóricas de salida y arribo planteadas en los párrafos anteriores.

Links: la aproximación Dexter.

La aproximación del grupo Dexter para soportar direccionalidad constituyó un avance arquitectural, aunque su colección de tipos de direccionalidad podía malinterpretarse. El avance arquitectural innovador radicó en asignar direccionalidad a los puntos extremos de los links en lugar de asignárselos al link como un todo. Esto es especialmente valioso cuando se generaliza la noción de direccionalidad hacia links multicabeza.

El modelo Dexter provee 4 constantes de direccionalidad, TO, FROM, BOTH y NONE. Éstas pueden representar a links como direcciones incrustadas y links con un origen y múltiples destinos, así como también comportamientos para travesía bidireccional. Sin embargo, este modelo de direccionalidad dejaba algunas cuestiones pendientes, por ejemplo, ¿Cuál es el significado de un link con dos puntos extremos FROM? ¿Cómo debe interpretarse un link con dos puntos extremos NONE? Estas cuestiones fueron abordadas en Hipermedia Abierta y se describen en el punto "links en sistemas de Hipermedia Abierta".

Otra de las contribuciones importantes del grupo Dexter consistió en asociar cálculo con puntos extremos de links. Esto significa que el calculo puede dispararse sobre cualquier (o todos) los puntos extremos de un link. De hecho el grupo utilizaba el término "*especificación*" en lugar de "*punto extremo*" para subrayar la cualidad potencialmente calculable de la localización de un punto extremo. Una especificación podía entonces ser un identificador absoluto o una descripción de un cálculo a ser ejecutado en tiempo de travesía.

La generalidad de la arquitectura de Dexter disparó además otras cuestiones relacionadas a la travesía. Por ejemplo: ¿Cuál debe ser la semántica exacta de seguir un link multicabeza? ¿Puede seguirse un link hacia atrás? Estas cuestiones también fueron abordadas y se describen en el siguiente punto.

Links en Sistemas de Hipermedia Abierta

Hay dos variantes en lo que a interconexiones respecta, es decir en cómo acceder desde un punto origen en el material a uno o más destinos. La primera variante es la que utilizaban los primeros sistemas y la que utiliza la Web, esto es, directamente incrustar en el material una entidad que contenga una o más *locSpec*s que capturen el o los destinos de la interconexión con ese origen.

La segunda variante, utilizada por los Sistemas de Hipermedia Abierta, es la interconexión externa, es decir, estructuras separadas del material. Para esto se utilizan las *refSpec*s, que empaquetan *locSpec*s, y que son la base para construir tales estructuras. Este diseño de interconexiones externas soporta cálculo y requiere, como ya se ha planteado, de un servidor de estructura y por lo tanto de una sobrecarga para la vinculación. Sin embargo, de esta forma (además de los beneficios obtenidos ya descriptos) permiten soportar una amplia variedad de las funcionalidades descriptas.

En Hipermedia Abierta, entonces, un link se diseña como un componente especializado para estructurar información que tiene cero o más puntos extremos (*refSpec*s) asociados, cada uno con una direccionalidad de travesía. Dado que cada punto extremo puede calcularse en forma independiente, es posible soportar una gran variedad de perspectivas de link, por ejemplo, los links genéricos consisten de uno o varios puntos extremos orígenes calculables y un punto extremo destino fijo.

La direccionalidad en Hipermedia Abierta se utiliza exclusivamente para modelar comportamiento de travesía, de este modo se evitan posibles confusiones con el concepto de incrustación. El conjunto posible de valores de direccionalidad puede ser extensible, e incluye al menos origen (FROM), destino (TO), ambos (BOTH) y ninguno (NONE). Se utiliza ninguno como una forma de ocultar temporalmente ciertos puntos extremos en la travesía. Además se utilizan atributos asociados a los links y/o puntos extremos para soportar otras formas de semántica de dirección.

Puntos Extremos

Las entidades estructurales en un Sistema de Hipermedia Abierta incluyen *locSpecs* para cada punto extremo, empaquetadas en *refSpecs*. Por lo tanto, los puntos extremos de un link están formados por una *refSpec*, una direccionalidad y una *pSpec* que determina la forma en que el punto extremo se presentará al usuario. Cuando un usuario sigue un link a un ancla, el sistema combina la *pSpec* del ancla y del punto extremo del link para manejar la presentación del material vinculado.

Una *locSpec* dada puede ser referenciada por varios puntos extremos (*refSpecs*), pero un punto extremo pertenece a un único link. La diferencia crucial entre una *refSpec* asociada a un punto extremo y a un ancla, es que la *locSpec* subyacente en la *refSpec* del ancla está restringida a identificar una localización dentro del contenido del componente al que pertenece, mientras que la asociada a un punto extremo no tiene restricción, incluso puede identificar un componente a través de un cálculo (la cual podría cambiar en cada acceso).

Con el propósito de ayudar a interpretar las entidades conceptuales descriptas hasta el momento, se presenta las figura 5.2 que ilustra una representación de un Hiperespacios, en la que se muestran dos componentes atómicos (el componente B incluye a su contenido y el componente A lo referencia) y un link.

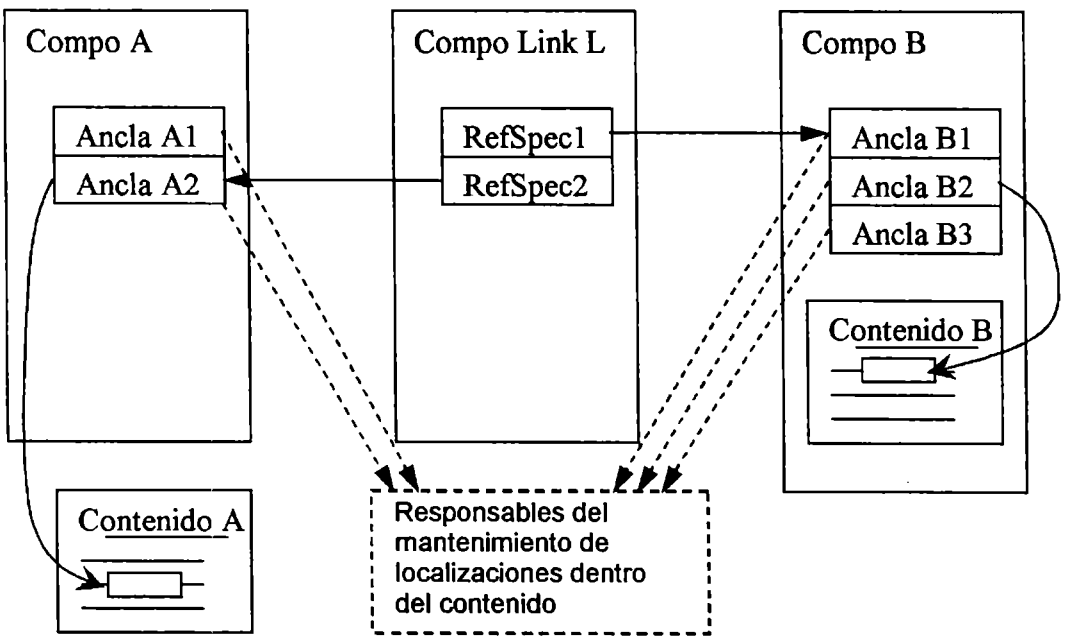


Figura 5.2: Dos componentes atómicos interconectados por un link.

Siempre que es posible, y tal cual se representa en la figura 5.2, las *refSpecs* de puntos extremos apuntan (identifican) a anclas en lugar de apuntar directamente dentro del material, ya que las anclas están diseñadas para mantener actualizadas las localizaciones dentro del componente al que pertenecen. Por lo tanto si el material cambia, el punto extremo sigue apuntando al mismo ancla (no es necesario actualizarlo), incluso múltiples puntos extremos podrían apuntar al mismo ancla.

Compuestos

La estructuración ha formado y forma parte de los Sistemas Hipermedia, las jerarquías de Augment, los *frames* de KMS y las relaciones de Aquanet dan cuenta de ello. Estas estructuras jerárquicas y relacionales estaban dirigidas a suplementar y complementar las estructuras de red provistas por los links asociativos clásicos.

Una crítica recurrente dirigida al diseño de páginas Web, radica en la falencia de soporte para estructuras, por ejemplo, estructuras jerárquicas que podrían servir para reunir páginas Web en documentos más grandes. De este modo los usuarios podrían seguir links a páginas individuales, y además la estructura jerárquica podría posibilitar otras actividades tales como impresión de todo el material. Hasta el momento, el diseño de páginas Web se ve forzado a elegir entre crear largas páginas fáciles de imprimir o paginas cortas interrelacionadas fáciles de navegar.

La Web tiene un solo tipo de componente, llamado página. Las páginas soportan selecciones permanentes (etiquetas NAME de HTML) que pueden ser apuntadas por URLs. Algo parecido al efecto de un compuesto se puede obtener utilizando páginas con múltiples URLs. Sin embargo, no se soporta el verdadero estructurado del compuesto, es decir, las páginas no pueden anidarse, todos los componentes son igualmente accesibles desde un rejunte plano.

Otra estructura perdida en la Web, pero bien conocida por los investigadores de Hipermedia, es el navegador gráfico de links. Aunque insuficiente por sí solo para resolver el problema de la desorientación dentro de un Hiperespacio, los navegadores de link proveen generalmente una ayuda significativa. Los navegadores gráficos de link pueden crearse y mantenerse manualmente, de hecho algunos sitios Web lo hacen, pero esto es sumamente costoso. Como se describe más adelante, los navegadores de link requieren soporte sofisticado de estructuración y la posibilidad de poder calcular estas estructuras (compuestos calculados) según demanda.

Dentro del campo de la Hipermedia el soporte para tales estructuras no ha estado en tela de juicio, pero si fue cuestión de debate el cómo implementar dichas estructuras utilizando links. En sistemas como KMS, Intermedia, y NoteCards la facilidad básica del link, ideado para referencias y asociaciones,

soporta además otro tipo de estructuras. Generalmente se utilizaban links con tipo para soportar otras formas de estructuración.

En su escrito "*seven issues*" [11], Halasz argumentaba en contra de la utilización de los links para representar estructuras diferentes a redes. Argumentaba que ciertas estructuras no tienen un componente principal natural que pueda actuar como el destino de un link, que debería conectar a la estructura como un todo. Además hay ocasiones en que la definición de la estructura está distribuida a través de una red de vínculos, en lugar de ser accesible a través de un solo componente cualquiera. Un ejemplo clásico que muestra estos inconvenientes son las estructuras que representan los argumentos de Toulmin.

Halasz destacaba además, que si bien los links eran apropiados para representar estructuras irrepetibles (estructuras con una raíz y varias hojas, sin nodos intermedios), eran "artificiales" en casos de patrones estructurales repetitivos, como en árboles. Además, como tales estructuras incluyen links generados por el sistema para mantener dicha estructura y links explícitamente creados por los usuarios, se generan inconvenientes a la hora de distinguirlos. La implementación de tales estructuras utilizando links, hace que sean costosas de crear y mantener.

Para abordar este problema, Halasz propuso utilizar los compuestos, dándoles la misma jerarquía que a los componentes atómicos y links. Los compuestos capturan las organizaciones de información no basadas en red, haciéndolas parte explícita de la funcionalidad Hipermedia. Halasz también introdujo las nociones relacionadas de virtualidad y cálculo en compuestos.

Un compuesto virtual se representa por una descripción o especificación que puede instanciarse según demanda para resultar en un componente compuesto. Un ejemplo muy citado es el compuesto de búsqueda, una consulta sobre un Hiperespacio que devuelve un conjunto de componentes y links. La intención es que tales compuestos sean creados y calculados según la demanda en tiempo de ejecución, pero no sean almacenados.

Luego que apareciera al escrito de Halasz, el grupo Dexter propuso una realización concreta de estructurado no basado en link. El compuesto de Dexter fue definido para ser un componente de la misma jerarquía que el link y los componentes atómicos, cuyo contenido consistiera de múltiples instancias de "datos" atómicos. Aunque fue un primer paso crucial, el compuesto de Dexter tenía algunos inconvenientes notados en primer lugar por aquellos que trataron de implementar la idea en un sistema concreto. Por ejemplo no estaba claro, cómo las anclas de compuestos podían apuntar al contenido de componentes anidados dentro de un compuesto.

La limitación primaria del compuesto de Dexter involucra cuestiones de encapsulamiento, referencia y contención (Gronbaek y Trigg, 1994 [43]). De acuerdo al grupo Dexter, un compuesto puede solo "contener" objetos de dato encapsulados llamados "componentes base". Tal cual el mismo grupo lo

describía, este tipo de compuesto podía modelar estructuras como *canvas* gráficos, consistentes de componentes simples conteniendo diferentes tipos de dato en su contenido. Sin embargo, tales componentes no pueden ser utilizados para organizar y estructurar otras entidades Hipermedia. Resulta necesario un componente que pueda contener o referenciar otros componentes o anclas de componentes.

Idealmente, los compuestos deberían soportar un conjunto estándar de tipos de estructura (conjuntos, listas, tablas, árboles) que pueda ser extendido con nuevas estructuras de dato de propósito especial. Además un compuesto debería estar capacitado para mantener objetos de dato (un documento de texto, por ejemplo), otros componentes (objetos del Sistema Hipermedia), referencias dentro de un componente, o una combinación de estas entidades.

Compuestos y links: Aspectos de comportamiento

Por lo expuesto, los compuestos y los links realizan agrupamiento; esto es, proveen formas de juntar múltiples localizaciones del material disponible. La distinción entre ellos tiene dos aspectos: uno es, si el agrupamiento es más una contención que una conexión y el otro es, si el comportamiento es más una apertura que una travesía. Contención y apertura sugieren un compuesto, mientras que conexión y travesía sugieren un link.

Otra diferencia la envuelve el rol en tiempo de ejecución del constructor. Un link es normalmente atravesado; es poco frecuente que se presente un link para inspección en una interfaz de usuario, a menos que su definición o puntos extremos estén editándose. Generalmente los links son representados como segmentos de líneas en un navegador gráfico o como texto marcado en un editor. En contraste un compuesto, en general, se presenta para inspección y edición en una interfaz.

Compuestos: Aspectos de cálculo y temporalidad

Hay dos cualidades muy relacionadas en el compuesto virtual de Halasz: cálculo y temporalidad. Como para los links, un compuesto incluye una especificación que puede calcularse en tiempo de ejecución para reconstruir su contenido. Halasz llamó a este tipo de compuestos, compuestos virtuales, ya que se almacenaban las descripciones en lugar del contenido de los mismos. En Hipermedia Abierta se denomina compuesto calculable, ya que el compuesto podría almacenar los resultados de cálculos previos y podría entonces comportarse como un compuesto normal para la mayoría de los propósitos.

En la Web, las URLs a veces incluyen una consulta, como un argumento para un *script* de búsqueda. En tales casos, se podría simplemente almacenar la URL de modo de volver a realizar la búsqueda posteriormente. Sin embargo, si la URL apunta a un formulario HTML que permita cargar los parámetros de la

búsqueda, el almacenamiento de la URL sería insuficiente para recrear la búsqueda.

En Hipermedia Abierta las consultas que describen cálculo se asocian a las *pSpecs* del componente calculable. Luego la decisión de recalcular en la próxima presentación del componente, se determina por la *pSpec* del componente, aunque la *pSpecs* de un link o compuesto desde el cual el usuario arribó a ese componente puede también contribuir a la decisión.

La denominación compuesto calculado, también podría malinterpretarse. En la mayoría de los casos, los puntos extremos de los link o de los compuestos son los calculados, en lugar de que el componente entero sea el calculado. Para un link calculado, por ejemplo, los puntos extremos origen son generalmente los calculados mientras que el punto extremo destino es el fijo. Para un compuesto calculado, todos los puntos extremos podrían ser resultado de un cálculo.

En cuanto a la temporalidad, la mayoría de los componentes en un Hiperespacio se crean para almacenarse; es de esperar que su tiempo de vida sea mayor que el de una simple sesión[5]. Sin embargo, algunos se crean en el momento y en forma temporal, como las paginas Web que son resultados de búsquedas. La duración exacta de estos componentes depende de la configuración del sistema operativo como el tamaño de la *cache* del usuario, pero generalmente, no se almacenan más allá de la duración de una sesión. Sin embargo el soporte para hacer tales componentes temporales como permanentes no existe. En la mayoría de los navegadores, el usuario puede almacenar el componente temporal en un servidor Web local. En este caso, la página es permanente y sus links salientes son preservados. Sin embargo, el problema es que cualquier link entrante queda colgado.

Compuestos en Sistemas de Hipermedia Abierta

En la Hipermedia Abierta se conciben a los compuestos como entidades capaces de estructurar a través de colecciones de *refSpecs*. Estas *refSpec* pertenecen a los puntos extremos que indican qué elementos forman el compuesto y qué relación existe entre estos componentes que lo conforman. Tales compuestos pueden construir estructuras Hipermedia externas al material y pueden fácilmente referirse a localizaciones dentro del material (directamente, a través de anclas) y también pueden referenciar distintos tipos de elementos como otros componentes estructurantes y/o localizaciones arbitrariamente calculadas.

Estos compuestos basados en puntos extremos de *refSpecs*, aunque son flexibles en términos de los elementos que contienen, no son tan flexibles para soportar la forma de inclusión o contención estricta, ya que en lugar de contener componentes, como era el caso de los compuestos de Dexter, referencian componentes.

---

[5] Sesión: entidad que representa un Hiperespacio en tiempo de ejecución.

Si bien los links también son entidades Hipermedia basadas en colecciones estructuradas de puntos extremos de *refSpecs*, tal cual se describió anteriormente, difieren con los compuestos respecto a su utilización. Dado que los links están diseñados para ser atravesados incluyen atributos de direccionalidad en sus puntos extremos, en contraste los compuestos están diseñados para ser abiertos y agregados jerárquicamente, por lo que incluyen atributos para determinar relación de estructura. Además las *refSpecs* de puntos extremos de compuestos, en general, apuntan a componentes completos, es decir, tienen vacía la especificación de dentro de componente.

En cuanto al cálculo se utiliza el campo de expresión calculable de las *locSpecs* empaquetadas en las *refSpec*. Como ejemplo típico de un compuesto calculado, es uno creado por la ejecución de una consulta. Un atributo del compuesto, generalmente la *pSpec*, contiene la información utilizada para realizar el cálculo. El contenido del compuesto puede luego recalcularse en cada acceso, según demanda o automáticamente.

En cuanto a la temporalidad, dado que los compuestos son componentes, tienen un conjunto de atributos, uno de los cuales puede identificar si el compuesto es o no temporal. Normalmente, los componentes se almacenan cuando el Hiperespacio al que pertenecen se almacena. Sin embargo, los componentes identificados como temporales se almacenan solo si hay estructuras (*refSpecs*) apuntando a ellas. Cuando resulte necesario, el usuario puede también marcar un componente temporal como permanente.

Vale aclarar que aunque Halasz entendía temporalidad y cálculo como aplicables principalmente a compuestos, el comportamiento descrito anteriormente para los compuestos de Hipermedia Abierta, se aplica en teoría a cualquier tipo de componente.

Hiperespacios: opciones de diseño

La entidad superior en el fenómeno Hipermedia es la red Hipermedia en sí, que en muchos escritos se la denomina Hipertexto o Hipermedia. En Hipermedia Abierta generalmente se utiliza el término Hiperespacio para evitar confusiones. Algunas de las más viejas y persistentes cuestiones para la investigación y desarrollo Hipermedia han estado puestas en este nivel:

- ¿Qué tan grande es un Hiperespacio?
- ¿Hay un Hiperespacio para todo el mundo o una colección?
- ¿Puede un Hiperespacio contener otros Hiperespacios?
- ¿El Hiperespacio es de solo lectura, o cualquier usuario puede extenderlo y modificarlo?
- ¿Debe un Hiperespacio distribuirse a través de varias bases de datos, o se deben interconectar Hiperespacios separados a través de links entre Hiperespacios?

- ¿Son necesarios los subHiperespacios temporales, de modo que se permita mover y copiar los mismos?

Subyaciendo estas opciones de diseño hay dos vistas distintivas y no inmediatamente reconciliables del Hiperespacio: como un medio de información totalmente abarcador o como un recurso para estructurar información de proyectos y grupos de trabajo.

En los ´70 y los ´80, visionarios como Nelson mantenían la aproximación del medio abarcador, mientras que desarrolladores de Sistemas Hipermedia en la investigación y en la industria tomaban la aproximación de Hiperespacio limitado, en respuesta a lo que notaban en los requerimientos prácticos del trabajo de cada día (McCracken y Akscyn, 1984 [56]). Estos desarrolladores de sistemas creían que los Hiperespacios limitados, eran el mejor lugar para invertir recursos y que en cualquier momento, cuando se convirtiera en práctico, estos Hiperespacios podían vincularse sobre redes más amplias.

La aparición de la Web fue un avance histórico (acordado por todos los investigadores Hipermedia), el primer sistema de hipertexto ampliamente disponible de la historia. Los usuarios de la Web tienen acceso a todas las páginas sin importar su ubicación, realizando de esta forma parte de la visión de Xanadu. Sin embargo, aún cuando la Web da la visión de Hiperespacio totalmente abarcador, ésta también afirma la necesidad de interrelacionar Hiperespacios más pequeños que soporten varias formas de acceso, de materiales y de estructuras.

A pesar del advenimiento de la Web, muchos trabajos posteriores de Hipermedia han sido sistemas que soportan Hiperespacios limitados. Generalmente, estos sistemas interrelacionan material relevante en un simple proyecto, o para un simple usuario o grupo de usuarios. Los identificadores de material, con el propósito de vinculación, son generalmente únicos dentro del Hiperespacio cerrado. Raramente se confronta, en estos casos, el problema de crear grandes Hiperespacios vinculando Hiperespacios limitados.

Hiperespacios: una propuesta integradora

Un diseño integrador debe ser capaz de soportar ambas opciones de Hiperespacio simultáneamente, al menos bajo ciertas condiciones. La Web puede ser nuestro docuverso[6], pero necesita intervincular otros Hiperespacios encapsulados, como lo son las intranets (que por seguridad permanecen detrás de *firewalls*) y los Hiperespacios basados en estaciones de trabajo. Los límites entre las intranets representan problemas adicionales de copiado y movido de subHiperespacios desde un sistema a otro. La noción de las estructuras almacenadas por separado resulta útil, pero tiene que ser lo suficientemente flexible para trabajar en múltiples contextos.

[6] Docuverso: Traducción de la denominación original de Ted Nelson, *docuverse*, para referirse a un universo de documentos.

Otra cuestión para pensar en Hiperespacios limitados, es que resulta muy poco probable que las aplicaciones de todos los días, como editores de texto y planillas de cálculo vayan a migrar a la Web a corto término. Por lo tanto las visiones actuales de Hipermedia deben ser capaces de vincular entre aplicaciones sobre una estación de trabajo y también entre aplicaciones y la Web.

El trabajo del grupo Dexter no descartaba esta visión híbrida de la Hipermedia. El concepto de Dexter de Hipertexto modelaba un espacio de direcciones limitado dentro del cual el trabajo de resolver punteros Hipermedia podía tomar lugar. En este sentido, los Hiperespacios locales y totalmente abarcadores cumplían en principio con el modelo de Dexter. Por otro lado el grupo Dexter no abordaba la cuestión de vinculación entre Hiperespacios ni cuestiones de copiado de Hiperespacios y subHiperespacios.

Dado que la mayoría de los Hiperespacios viven o vivirán bajo la Web, el problema de intercambio de subHiperespacios puede ser transformado en uno de distribución e interoperatividad remota. En tal escenario una parte de un Hiperespacio podría ser identificable (en lugar de ser copiado y pegado) y el destinatario distante podría entonces leerlo o manipularlo remotamente.

Hiperespacios en Sistemas de Hipermedia Abierta

En Hipermedia Abierta se adopta la opción de Hiperespacios limitados, considerando que las estructuras se almacenan en forma separada y que se deben cubrir (integrar) tanto las aplicaciones que corren en estaciones de trabajo, como los Hiperespacios detrás de *firewalls*. El Hiperespacio limitado puede pensarse como un compuesto que contiene (en lugar de referencia) otros componentes. La contención estricta implica que si el Hiperespacio es eliminado, se eliminan todos los componentes contenidos.

De este modo, cada Hiperespacio tiene su propio espacio de nombres dentro del cual los componentes (y *refSpecs*) están unívocamente identificados y las *locSpec*s restringidas al rango de direcciones del Hiperespacio al que pertenecen. Si se requiere referenciar componentes que pertenezcan a otro Hiperespacio, es necesario proveer el ID del Hiperespacio externo.

La Web bajo esta visión, puede ser tomada como una colección de Hiperespacios, uno por cada servidor Web. Entonces la URL estándar puede verse como compuesta por dos partes, una identificador de Hiperespacio (un servidor Web) y un *locSpec* única dentro de ese Hiperespacio (el resto de la URL). De acuerdo a esto las URLs actúan como punteros a través de Hiperespacios, pero solamente dentro de la colección limitada de Hiperespacios que comprende la Web.

En la figura 5.3 se sintetizan, a través de un mapa conceptual, las entidades subyacentes en el diseño de un Sistema de Hipermedia Abierta y sus relaciones. Estas entidades forman la base del diseño propuesto en el capítulo

6, en la búsqueda de un soporte para el ambiente de investigación planteado en el capitulo 2.

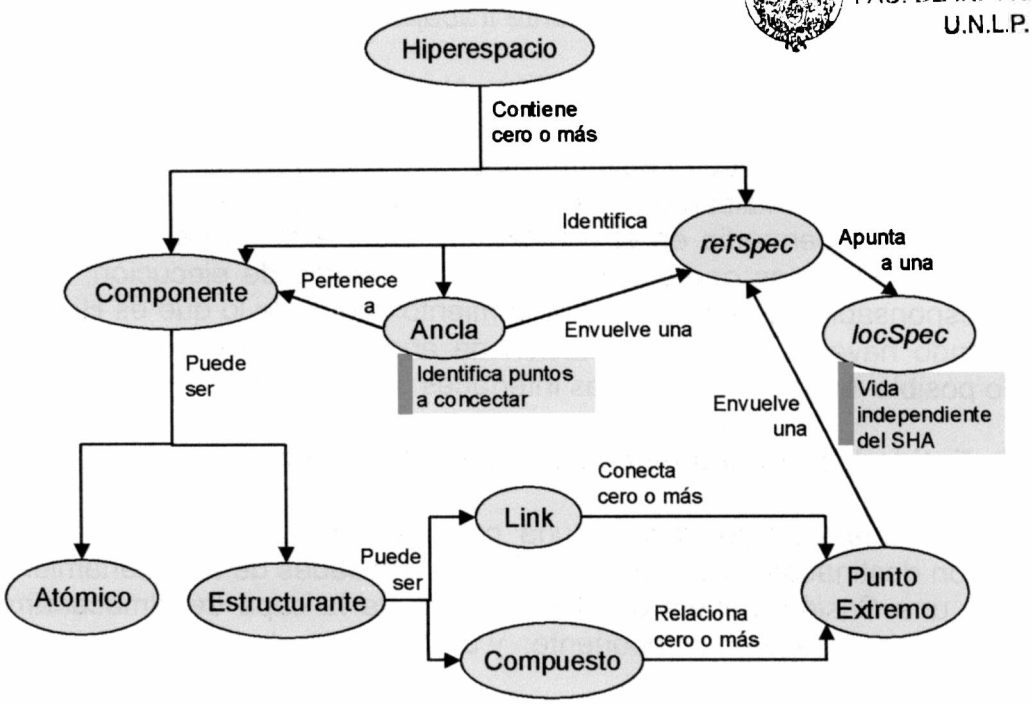


Figura 5.3: **Mapa conceptual de las entidades subyacentes en el diseño de un SHA.**

5.4 Conceptualización en el diseño del tiempo de ejecución

Un objetivo de diseño primario para los proyectos Hipermedia radica en hallar la forma apropiada de distinguir dos partes del sistema, una que almacene y recupere estructuras, y otra a través de la cual los usuarios naveguen y manipulen dichas estructuras. Hasta aquí se han descrito las entidades fundamentales de diseño de Hipermedia y las formas a través de las cuales estructuran el material en línea. En esta sección se presenta la conceptualización subyacente en el diseño del comportamiento de los Sistemas Hipermedia en tiempo de ejecución.

5.4.1 Separando almacenamiento de tiempo de ejecución

El hecho de desenganchar tiempo de ejecución del almacenamiento permite que los usuarios puedan colaborar, mientras ejecutan aplicaciones sobre distintas plataformas y comparten un servidor de almacenamiento común. El grupo Dexter, como se describió en el capítulo 3, realizó una división en tres capas para abordar esta cuestión.

La división de capas de Dexter, sin embargo, presentaba un inconveniente, el cual subyacía en la separación de responsabilidades entre la capa de tiempo de ejecución y la capa de interior de componente. Este modelo suponía que las aplicaciones responsables de manejar componentes

individuales no tenían demanda sobre la capa de tiempo de ejecución. Sin embargo, tal cual se describió en el capítulo 4 (sección 4.5) las aplicaciones además de manejar el almacenamiento y la edición de material, también presentan una interfaz cuidadosamente trabajada.

Las aproximaciones de Sistemas Abiertos reconocen la existencia del ambiente e interfaz de una aplicación, requiriendo que el diseño se enfoque en la división de tareas entre el Sistema Hipermedia y las aplicaciones de terceros. La división puede realizarse a través de un conjunto de APIs que pongan la funcionalidad Hipermedia en el contexto de las interfaces de las aplicaciones existentes y sistemas operativos. La capa de tiempo de ejecución, entonces, tiene responsabilidad solo del comportamiento dentro de lo que es Hipermedia, incluyendo navegación y presentación de anclas de links, y es en el mayor grado posible independiente de las interfaces de las aplicaciones.

5.4.2 Entidades Conceptuales del tiempo de ejecución

El modelo Dexter incluye una colección de conceptos de tiempo de ejecución destinados a corresponder con las entidades de almacenamiento. En Dexter, una *Sesión* maneja un Hiperespacio de la capa de almacenamiento; una *Instancia* maneja un componente; y un *Marcador de link*, un ancla.

Esencialmente, estos mapeos entre entidades de tiempo de ejecución y de almacenamiento de Dexter son muchos a uno, es decir, un simple componente puede tener múltiples *Instancias* correspondientes a distintas vistas, y en diferentes vistas, un ancla de un componente podría representarse utilizando distintos estilos de *Marcadores de link*. Del mismo modo un Hiperespacio podría tener múltiples *Sesiones* corriendo simultáneamente para diferentes usuarios sobre diferentes plataformas.

La *Instancia* de Dexter es la presentación de un componente al usuario. El funcionamiento es similar al de una *cache*: una copia (*Instancia*) de un componente se pasa al usuario para que la visualice y/ o edite. Esta copia será luego escrita nuevamente en la capa de almacenamiento. Puede haber más de una *Instancia* de un componente a la vez, por ello, a cada *Instancia*, Dexter le asignaba un identificador único que denominó IID (*instantiation identifier*).

Como en un momento dado, un mismo usuario, puede estar visualizando y/o editando varias *Instancias* de componentes, el modelo utilizó la entidad Sesión, que permite saber momento a momento la relación entre los componentes y sus *Instancias*. De este modo cuando un usuario pretende acceder a un Hiperespacio, lo hace abriendo una sesión.

En cuanto al *Marcador de link* de Dexter, es difícil sino imposible, en desarrollar en SHA un modelo general de Marcador de link, dado que pueden aparecer como objetos incrustados en material cuyas interfaces se manejan por aplicaciones de terceros. Además el modelo supone una correspondencia entre Marcadores de link y anclas, que sugiere que cualquier ancla en un

sistema está visiblemente marcada en la correspondiente *Instancia* del componente. Como se ha visto este no es siempre el caso, por ejemplo, las anclas que se calculan o resuelven corresponden a selecciones arbitrarias del usuario en un editor que pueden no estar marcadas previamente.

Además del inconveniente con el Marcador de link, el modelo carece de una entidad abarcadora para supervisar el manejo de múltiples Hiperespacios, y dado que en Hipermedia Abierta se adopta la utilización de Hiperespacios limitados, resulta necesario contar con alguna entidad que pueda manejar esta cuestión.

Por lo expuesto anteriormente, en SHA se descarta la utilización del Marcador de link de Dexter y se agrega una nueva entidad para manejar múltiples Hiperespacios, comúnmente denominada Manejador de Sesión, que es la entidad abarcadora de tiempo de ejecución destinada a manejar múltiples *Sesiones*. La *Sesión* continúa como la entidad que maneja el comportamiento de un Hiperespacio y la *Instancia* continúa manejando la vista y edición de un componente particular.

5.4.3 Manejo de Presentación

Las estructuras Hipermedia se brindan a través de un tiempo de ejecución que soporta la interacción de usuario. En este contexto resulta necesario representar visualmente cada entidad estructural del sistema en una interfaz de usuario. Las presentaciones pueden modificar la apariencia del material envuelto por un componente o pueden requerir la creación de una nueva interfaz gráfica, por ejemplo para un componente compuesto. Las propiedades de tal presentación dependen de una variedad de factores estáticos y dinámicos.

La noción de especificación de presentación, *pSpec*, de Dexter ayuda a cruzar los límites entre las capas de almacenamiento y tiempo de ejecución. Las *pSpecs* ofrecen una forma de representar un conjunto de preferencias para la apariencia de anclas, puntos extremos y componentes. De este modo, las *pSpecs* se almacenan junto con los demás atributos de las entidades, en el momento en que el Hiperespacio se almacena. Es la forma de representar y almacenar preferencias acerca de la apariencia de las entidades estructurales

Además se pueden aplicar múltiples *pSpecs*, por ejemplo, el material envuelto por un componente puede ser manejado por la *pSpec* del componente y por la *pSpec* del link que fue atravesado para arribar a ese componente. Por lo tanto se combinan los efectos de las *pSpecs* y se priorizan sus efectos cuando hay conflictos.

La forma de las *pSpecs*, su sintaxis y estructura, estaba completamente abierta en el modelo Dexter, aunque la tendencia era implementarlas como listas de pares atributo-valor. Una posibilidad interesante de extender esta noción, reconocida por diseñadores Hipermedia como Ackscyn y McCracken

en 1993 [57], consiste en utilizar la misma Hipermedia como un medio de representar las *pSpec*.

La utilización de la Hipermedia para representar *pSpecs* es más natural para los casos en que una *pSpec* de un componente hace referencia a otro componente. Por ejemplo, un link podría requerir que su componente destino se brinde dentro de un contexto, requiriendo que un compuesto particular, que encierra el componente destino, sea abierto junto con dicho componente. En este caso la *pSpec* del punto extremo del link, contiene una *refSpec* del contexto de destino que identifica el compuesto que encierra el componente destino.

Las *pSpecs* pueden también utilizar atributos para especificar comportamiento en la travesía de un link. Por ejemplo, como ya se planteó anteriormente, estos atributos podrían indicar al sistema de ventanas reemplazar la ventana que representa el componente actual, con la ventana del componente destino, o para otras necesidades, yuxtaponer las ventanas. Este comportamiento podría ser alternado según la situación, y nuevamente las *pSpecs* serían las encargadas de manejar tal alternancia.

5.4.4 Manejo de Travesía

Seguir un link, el comportamiento de tiempo de ejecución más característico de un Sistema Hipermedia, requiere atravesar los límites entre las capas de almacenamiento y la de tiempo de ejecución, y entre la capa de tiempo de ejecución y la de aplicación. Se pasan las especificaciones de los puntos extremos del link a identificadores únicos de componentes y se recuperan los componentes con tales identificadores desde el almacenamiento. Por el otro, el concepto de ancla modela el origen y destino del link, los cuales están incrustados en material dentro de un componente (capa de aplicación).

Analizando la travesía desde el punto de vista conceptual (enfoque ampliamente utilizado en la Hipermedia Abierta), los usuarios experimentan lo que se denomina localidad, es decir, la interfaz los posiciona en lugares particulares del Hiperespacio, y resulta necesario soportar cambios de localidad. En Hipermedia Abierta estos cambios de localidad se soportan brindando el material referenciado por una o más *refSpecs* de una estructura, la cual es accedida desde la actual localidad del usuario.

En el caso más simple, el usuario experimenta el cambio de localidad con el seguimiento de un link clásico, es decir, haciendo clic en lugares resaltados en el material de componentes. Alternativamente el usuario puede identificar una localización en el material que no estaba resaltada, y el sistema busca estructuras que coincidan en alguno de sus puntos extremos con esa localización, en otras palabras, busca *refSpecs* de componentes estructurantes que resuelven a esa localización y lleven al usuario a otro lugar del material en línea.

Los links y los compuestos son los componentes estructurales vistas que posibilitan los cambios de localidad, brindando dos tipos de navegación: travesías y aperturas respectivamente. En ambos tipos de navegación están involucrados tres elementos: un origen, un componente estructural y un destino. A su vez, cada uno de estos tres elementos puede ser múltiple, por ejemplo podría haber múltiples destinos. En general al componente estructural (link o compuesto) se lo denomina componente de mapeo, ya que mapea desde un conjunto de *refSpecs* a otro.

Capítulo 6

Diseño de Soporte para la Investigación Conceptual

Resumen

En este capítulo se presenta una propuesta de diseño, basada en los fundamentos de la Hipermedia Abierta planteados en el capítulo 5, para proveer soporte al ambiente de investigación conceptual cooperativa discutido en el capítulo 2. La propuesta consta de: una memoria descriptiva que detalla cómo operará el sistema, una arquitectura que lo soporta y un diseño orientado a objetos de las dos capas relevantes de la arquitectura (almacenamiento y tiempo de ejecución), el cual incluye una descripción detallada de la operación más característica del SHA, la travesía. Se propone también, desde el comienzo del capítulo, una metodología de desarrollo sobre la cual se basa la especificación del diseño propuesta.

6.1 Introducción

Debido a que, tal cual se ha fundamentado, se considera a la Hipermedia Abierta como la mejor forma de abordar un soporte para el ambiente de investigadores planteado en el capítulo 2, y en modo general a la Hipermedia como una forma natural para representar y compartir conocimiento, en este capítulo se presenta una propuesta de diseño, basada en los fundamentos de Hipermedia Abierta descriptos en el capítulo 5, de un Sistema que cubre las necesidades del ambiente de investigación planteado. Se trata de un diseño centrado principalmente en la obtención de una clara visión de la funcionalidad del sistema, de la arquitectura necesaria para soportarla y de las clases de diseño involucradas.

Además se propone abordar el proceso de desarrollo del sistema mediante una aproximación experimental y cooperativa. Experimental de tipo prototipado, es decir que consiga implementaciones parciales tempranas, y cooperativa en el sentido de involucrar fuertemente en el desarrollo a los usuarios potenciales. Debido a esta elección de proceso de desarrollo, es que el diseño presentado no está orientado hacia una especificación excesiva.

Cabe destacar que la elección de un desarrollo de tipo prototipado se debe a lo inadecuados que han resultado los procesos tradicionales orientados hacia la especificación, principalmente en sistemas de la envergadura como el propuesto en los que generalmente prevalecen las metodologías ágiles. Por otro lado, en cuanto a la elección de un desarrollo cooperativo, se considera que el éxito de un sistema depende del grado en el cual su desarrollo se alimenta de los intereses y de la participación activa de los usuarios, ya que en definitiva, los sistemas deben ser útiles para las personas cuyo trabajo no tiene nada que ver con Hipermedia, ingeniería, *frameworks*, etc.

6.2 Memoria Descriptiva del Sistema propuesto

El Sistema de Hipermedia Abierta propuesto, amplía las aplicaciones favoritas de los investigadores con un Servicio de Hipermedia Abierta que permite la creación dinámica de estructuras y comentarios (almacenados en una base de datos separada de los documentos) y la colaboración necesaria para poder compartir estos útiles metadatos. La ampliación de las aplicaciones se produce a través de nuevos elementos de interfaz de usuario (generalmente con barras de menú o menús contextuales) que proveen acceso a la funcionalidad Hipermedia e invocan procedimientos del Servidor de Hipermedia.

Tal cual se planteó en el capítulo 5 sección 5.1, el almacenamiento de estructuras en base de datos separada, implica la localización vía anclas, las cuales encapsulan la información de localización dentro de los documentos sobre cualquier tipo de medio e incluso sobre documentos de solo lectura, ya que el documento original no es alterado por la creación de estructuras. Como se describió también en el capítulo 5, la única limitación para este tipo de localización, son los métodos que las aplicaciones, favoritas de los usuarios, provean para acceder a partes de los documentos que manejan.

Los links, notas y demás estructuras se mezclan dinámicamente con los documentos (páginas Web y documentos de aplicaciones de escritorio) presentados por las aplicaciones, es decir, las localizaciones de links y anotaciones son resaltadas en forma similar a como se muestran los links en la Web. Los compuestos, colecciones y argumentos de Toulmin, se acceden a través de navegadores gráficos manejados por una aplicación Hipermedia Cliente (según la arquitectura propuesta descripta en la sección 6.5) que corre sobre cada estación de trabajo de cada investigador. El investigador puede

entonces crear y utilizar las estructuras externas como capas transparentes sobre páginas Web y documentos arbitrarios.

El comportamiento extra que agrega el SHA involucra la creación y la navegación a través de estructuras, sin embargo las aplicaciones favoritas de los investigadores, las cuales tienen su ambiente de tiempo de ejecución propio, siguen haciéndose responsables del manejo del contenido de los documentos.

6.2.1 Interfaz de usuario del Sistema

Se provee al investigador de acceso a la funcionalidad Hipermedia (y con ello a las estructuras externas) de la siguiente forma: como se planteó en la sección anterior a través de menús agregados a las aplicaciones favoritas de los usuarios y a través de una aplicación de escritorio denominada Cliente de Hipermedia Abierta. Cada investigador tendrá abierta en su plataforma una ventana con el Cliente de Hipermedia Abierta. Esta aplicación se encarga además, de la comunicación con el Servicio de Hipermedia Abierta, es decir, actúa como *middleware* entre las aplicaciones de la estación de trabajo y el servidor Hipermedia, tal cual se describe en la sección 6.5.

Los menús agregados a las aplicaciones favoritas de los investigadores proveen acceso a las funciones Hipermedia que se pueden realizar sobre los documentos individuales, es decir constan de opciones tales como: crear link, agregar ancla, atravesar link, crear nota, etc. A partir de la descripción de casos de uso de la sección 6.4, se pueden obtener el resto de las funcionalidades agregadas a las aplicaciones.

Por otro lado, el Cliente Hipermedia provee una interfaz de usuario que permite acceder a funcionalidad Hipermedia aplicable al Hiperespacio general (no solo a documentos individuales), es decir, permite acceder a funciones tales como "visualizar todas las colecciones del Hiperespacio". Además provee acceso a la edición y visualización de las estructuras agregadas por el Sistema Hipermedia, por ejemplo, a colecciones y argumentos de Toulmin, las cuales están compuestas de varios documentos.

La interfaz del usuario del Cliente Hipermedia mostrada en la figura 6.1, consiste de una ventana principal que permite que el usuario cree y edite Hiperespacios, los cuales dentro de esta propuesta se denominan contextos. Se ha elegido el término contexto, en lugar de Hiperespacio, ya que las estructuras creadas sobre el contenido representan un punto de vista (o contexto) desde el cual el material es analizado, anotado e interrelacionado. La interfaz del Cliente muestra además una lista (navegador) con los nodos (documentos, colecciones y argumentos de Toulmin) y otra con los links pertenecientes al contexto actual.

Al hacer doble clic sobre un nodo, en el navegador de nodos, se activa la aplicación asociada y se muestra el documento. Si se trata de una colección o

argumento de Toulmin, es el Cliente Hipermedia el que muestra una interfaz gráfica para visualizar y editar este tipo de nodos. Para mostrar los links que apuntan desde y hacia un nodo dado, el usuario debe hacer clic derecho sobre el nodo en cuestión y debe seleccionar "mostrar links" a partir de un menú contextual, en respuesta a esta acción, se muestran los links en el navegador de links.

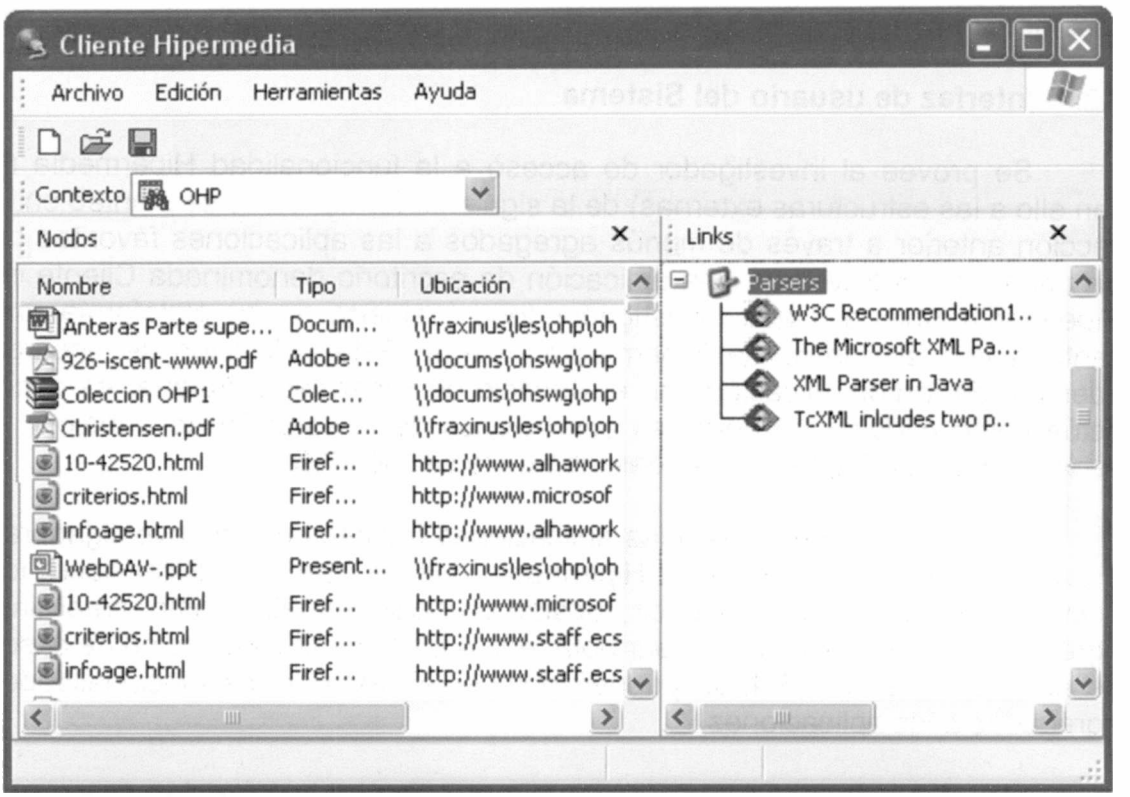


Figura 6.1: Ventana Principal del Cliente Hipermedia.

Los nodos, notas, links y anclas que referencian documentos son creados desde las aplicaciones a través de los menús agregados a cada una de las mismas. Las colecciones y los argumentos de Toulmin se crean desde el menú del Cliente Hipermedia. Para agregar un documento a una colección se copia una referencia del mismo desde el navegador de nodos. Cuando se crea un argumento de Toulmin, es el Cliente Hipermedia quien solicita cada una de sus cuatro componentes.

6.3 Modelo del Dominio

El modelo del dominio propuesto, ilustrado en la figura 6.2, muestra los conceptos existentes en el dominio del problema planteado en el capítulo 2 y sus relaciones. A modo de recordatorio, en el capítulo 2 se planteó un ambiente en el cual un grupo de investigadores distribuidos geográficamente trabaja en forma cooperativa durante la etapa de investigación conceptual. Dentro de esta

etapa los investigadores se manejan con una gran cantidad de material compartido e interrelacionado, posiblemente generado por distintas aplicaciones e incluso almacenado sobre distintas plataformas.

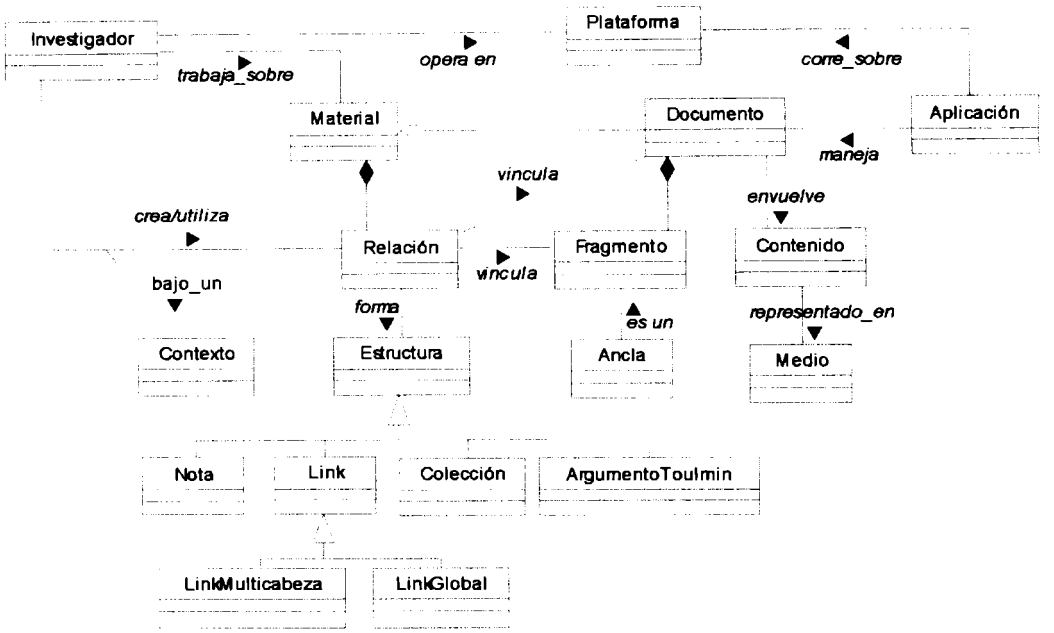


Figura 6.2: Modelo del Dominio.

6.3.1 Conceptos y relaciones

A continuación se describen en forma textual los distintos conceptos y relaciones representados en el modelo del dominio propuesto. El objetivo de tal descripción es complementar el modelo del dominio, brindando una mayor especificidad conceptual que permita el manejo de un vocabulario común y un mejor entendimiento del dominio del problema.

- **Material:** escritos científicos dentro de un área de conocimiento sobre los cuales los investigadores realizan una búsqueda a nivel cognitivo. El material está representado por un conjunto de **documentos** que pueden estar sobre distintos tipos de **medios**, distribuidos sobre distintas plataformas. La organización del material en distintos documentos constituye es en sí, una forma de estructurar a tal organización, se la denomina estructura inherente del material. Por lo tanto el material tiene un **contenido** (dato) y una **estructura**. El contenido está distribuido (organizado) en distintos documentos. El investigador utiliza y edita contenido y crea y/o atraviesa (navega) estructuras.

- **Medio:** la variedad de material (heterogeneidad) implica que puede estar formado por un conjunto de documentos, cada uno de los cuales puede pertenecer a un medio distinto, por ejemplo: un documento CAD, una

planilla de cálculo, un gráfico, un texto, etc. Por lo que, cada documento puede ser manejado por una **aplicación** distinta.

- **Documento:** material científico que envuelve contenido. Se pueden seleccionar partes o fragmentos (**anclas**) de documentos a fin de crear relaciones. Se permiten además relaciones entre documentos completos. En ocasiones se utiliza el término archivo para denominar a un documento.

- **Contenido:** son los datos sobre los que se investiga. Tal cual se describió, el contenido puede estar representado en distinto tipo de medios.

- **Aplicación:** son manejadores de contenidos que permiten crear, editar, visualizar y almacenar dicho contenido, es decir, manejan almacenamiento y comportamiento de documentos.

- **Plataforma:** ambiente sobre el cual corren las aplicaciones.

- **Relación:** los investigadores realizan lectura activa de documentos, creando vínculos entre documentos, y entre fragmentos de documentos. Las relaciones no son solamente asociativas, también hay **colecciones** y relaciones estructurales entre elementos, tal es el caso de los **argumentos de Toulmin**. Las relaciones forman estructuras sobre los documentos.

- **Contexto:** es un punto de vista desde el cual el material es analizado y a partir del cual se crean relaciones y notas. Permiten tener distintas perspectivas sobre un mismo conjunto de documentos. El investigador puede crear y navegar estructuras externas sobre distintos contextos, como si fueran capas trasparentes puestas encima de los documentos. Además, potencialmente, puede ver los documentos desde uno o más contextos o sin estructura externa impuesta. Varios investigadores, generalmente, comparten un mismo contexto o punto de vista.

- **Estructura:** conjunto de relaciones sobre el material, establecidas según un punto de vista o contexto determinado. Estas relaciones son metadatos útiles que representan conocimiento y brindan un valor agregado al contenido. El vínculo asociativo o link es la forma más común de estructurar el material, sin embargo tal cual se ha descrito, hay otras formas de estructurar como las jerarquías, las colecciones, las taxonomías, etc. Para el ambiente de investigadores presentado las estructuras son: links, links globales, colecciones, notas y argumentos de Toulmin.

- **Link:** relación asociativa entre documentos. La relación más común es la binaria, es decir, un link con un ancla origen y un ancla destino. Además se pueden utilizar links multicabeza, que representan relaciones entre múltiples anclas.

- **Link global:** los links globales poseen un ancla destino fija y múltiples anclas origen calculadas. Este es el caso de relaciones entre una palabra o frase que aparece en distintos documentos y su definición en un diccionario.

- **Colección:** conjunto de documentos relacionados, por ejemplo, un conjunto de escritos científicos sobre el formato de intercambio de Hipermedia abierta (OHIF) que normalmente se colocaría en un mismo directorio bajo una organización de carpetas, en el ambiente planteado, estos documentos formarán parte de una misma colección.

- **Argumento de Toulmin:** es un modelo que explica, desde un punto de vista lógico, la estructura a la cual debe responder un texto argumentativo. En dicho modelo, un sujeto argumentador presenta explícitamente una tesis o conclusión y expone una serie de razones lógicas que desembocan y confirman la tesis propuesta. El modelo consiste de cuatro componentes principales:

    - **Conclusión o tesis:** una declaración que clarifica un asunto en discusión, o una posición respecto de un tema que quien argumenta trata de defender. Es una aserción.
    - **Fundamento o evidencia:** son los datos o la información sobre los cuales se basa la **conclusión**. es el sustento.
    - **Garantía:** la conclusión y el fundamento no son suficientes para establecer una argumentación sólida, faltan otros elementos que indiquen cómo a partir de una evidencia se obtiene la conclusión. Tal elemento es la garantía, parte esencial del argumento, que permite evaluar si la conclusión se basa en el fundamento, es decir, autoriza el paso del fundamento a la conclusión. Normalmente las garantías se presentan bajo la forma de reglas, principios o patrones
    - **Respaldo:** el respaldo asegura que las **garantías** sean fidedignas y aplicables al contexto presente. Puede ser un estudio científico, una estadística o una creencia firmemente arraigada dentro de una comunidad.

- **Nota:** comentario realizado sobre material durante la lectura activa del mismo.

- **Ancla:** fragmentos de material referenciados por links. Pueden ser fragmentos de texto, *frames* de video, objetos de un gráfico, filas de una BD e incluso un ancla puede referenciar un documento completo.

6.4 Requerimientos y Casos de Uso

En esta sección se presentan los requerimientos funcionales y no funcionales. Los requerimientos no funcionales fueron obtenidos a partir de los requerimientos técnicos del sopote de *software*, planteados en el capítulo 2, y del análisis de los Sistemas Hipermedia presentados en el capítulo 3, mientras

que los requerimientos funcionales se obtuvieron a partir de las necesidades de los investigadores planteadas en el capítulo 2.

6.4.1 Requerimientos No Funcionales

A fin de garantizar el éxito del Sistema, el mismo debe cumplir los siguientes requerimientos no funcionales:

- Respetar los principios de diseño de la Hipermedia Abierta: estos principios han sido fruto de años de investigación de la comunidad de Hipermedia Abierta. Teniendo en cuenta este requerimiento se han mapeado los conceptos presentados en el capítulo 5 a las clases de diseño que se presentan en la sección 6.6.
- Implementar el Sistema utilizando código abierto: la intención es que los prototipos a desarrollar sirvan de plataforma de investigación para desarrollos futuros y brinden facilidades a la hora de realizar extensiones y ampliaciones.
- Respetar las aplicaciones favoritas de los usuarios: La integración de las aplicaciones favoritas de los usuarios es un requisito general de los SHA y de cualquier sistema que pretenda ser exitoso. El sistema no debe forzar a los usuarios a cambiar su ambiente de trabajo.
- Proveer la mínima funcionalidad Hipermedia requerida: de modo que soporte el ambiente planteado en el capítulo 2, pero minimice la complejidad de la infraestructura y de la interfaz del usuario.
- Utilizar estándares de la comunidad de Hipermedia Abierta: de modo de aprovechar los estudios más relevantes en el área y de permitir la comunicación del SHA desarrollado con otros SHA relevantes. Con tal motivo se ha utilizado un modelo de dato que permite la utilización del formato de intercambio estandarizado OHIF (*Open Hypermedia Interchange Format*) [33].
- Permitir la accesibilidad del Servidor de Hipermedia desde distintas aplicaciones que corran sobre distintas plataformas: debe ser posible que el servidor sea remoto, esto permite además minimizar el desarrollo necesario para cada plataforma específica. El servicio debe estar disponible en la Web vía URL.
- Proveer un mecanismo de seguridad para acceder al sistema: el sistema debe permitir el acceso sólo a los integrantes del grupo de investigación. Alternativamente, el sistema podrá dar un acceso calificado, con derechos de sólo lectura, para integrantes que tengan otros roles.
- Simplificar el mecanismo de instalación: la instalación debe ser sencilla y debe estar preconfigurada en el mayor grado posible. Este es un requisito de usabilidad básico.
- Almacenar links y demás estructuras en bases de dato Hipermedia Orientada a Objetos sobre un servidor específico: esto facilita el trabajo cooperativo permitiendo que el bloqueo de acceso y la notificación de eventos se puedan realizar a nivel de estructuras individuales.

Un requisito no funcional que merece un par de párrafos especiales es el de hacer que los conceptos de Hipermedia Abierta tales como colecciones, links multicabeza y links globales, sean conocidos y accesibles para los usuarios. Para este fin se debe proveer una interfaz amigable con mínima jerga de Hipermedia Abierta y un sistema de ayuda que trabaje los nuevos conceptos a través de metáforas conocidas y de preguntas frecuentes. Hay que tener en cuenta que las nuevas características (funcionalidades) son aceptadas sólo cuando son fáciles de utilizar y se construyen sobre conceptos conocidos.

A modo de contribución se describe a continuación una nueva característica construida a partir de un concepto conocido: existe en el trabajo diario de los usuarios un concepto similar en su comportamiento al link multicabeza, éste concepto es el menú contextual. El menú contextual se despliega, por ejemplo, cada vez que el usuario hace clic derecho sobre un ícono del escritorio, a partir de allí el usuario puede elegir una opción entre un conjunto de opciones presentadas. Este comportamiento es idéntico al seguimiento de un link multicabeza, en el cual el usuario debe elegir un destino determinado entre varios destinos presentados como opciones.

6.4.2 Requerimientos Funcionales

El sistema brinda servicios a sus usuarios. Los usuarios pueden ser investigadores u otros sistemas (Clientes de Hipermedia Abierta) que interactúen con los servicios brindados. A continuación se detallan los requerimientos funcionales que el sistema debe proveer.

El sistema debe permitir al usuario:

- Ingresar una dirección de correo electrónico
- Editar una dirección de correo electrónico

- Crear un contexto.
- Abrir un contexto existente
- Agregar un documento a un contexto

- Crear un link
- Editar un link

- Agregar un ancla

- Crear un link global
- Editar un link global

- Agregar un ancla global

- Atravesar un link
- Atravesar un link global

- Crear una colección de documentos
- Agregar un documento a una colección
- Eliminar un documento de una colección

- Abrir una colección
- Editar una colección

- Crear una nota.
- Editar / leer una nota.

- Consultar los links que apuntan a un documento

- Crear un argumento de Toulmin
- Editar un argumento de Toulmin
- Abrir un argumento de Toulmin

6.4.3 Descripción de Casos de Uso

Las interacciones con los usuarios se representan con casos de uso, los que permiten describir la funcionalidad del sistema y proporcionarán un hilo conductor en el proceso de desarrollo. Sin embargo, dado que tal cual se planteó en la introducción, se busca un desarrollo de tipo experimental y participativo, es decir no excesivamente orientado a la especificación, se describen a continuación solamente los tres casos de uso más relevantes:

Caso de uso: Crear un link.

Actor: Usuario.

Propósito: Que el usuario pueda crear una relación asociativa desde un documento manejado por su aplicación favorita.

Descripción: el usuario selecciona un fragmento de material (una palabra o frase si se trata de un documento textual) con el objetivo de crear una relación asociativa entre este fragmento y otro/s.

Precondición: El usuario debe tener el Cliente de Hipermedia Abierta (descripto en la sección 6.2.1) en ejecución, para lo cual debió ingresar al sistema utilizando su dirección de correo electrónico. Además debe tener un contexto abierto sobre el cual se creará el link.

Poscondición: Se crea un link con origen en el fragmento seleccionado y se resalta el mismo como un ancla. En este caso es un link con un solo punto extremo, es decir un link colgado, lo cual es permitido y ha sido fundamentado en la comunidad de Hipermedia Abierta. La presencia de links

colgados puede monitorearse en forma manual o automática. Si el documento donde se crea el link no pertenece al contexto actual, éste es agregado automáticamente al mismo. El link creado, pasa a ser el actual en el navegador de links del Cliente Hipermedia.

Flujo principal:

I. El usuario selecciona un fragmento de material, por ejemplo una palabra o frase en un texto.
II. El usuario hace clic derecho sobre el fragmento seleccionado.
III. El sistema despliega un menú contextual con la funcionalidad Hipermedia agregada a la aplicación favorita del usuario.
IV. El usuario selecciona "crear un link" desde el menú contextual.
V. El sistema crea un link, dentro del contexto actual, con un solo punto extremo, el cual es el ancla origen seleccionada por el usuario.
VI. El sistema agrega el link con dicho punto extremo a la base de datos Hipermedia.
VII. El sistema resalta (con color y subrayado) el ancla origen del link.

Flujos alternativos:

I. El usuario no tiene en ejecución el Cliente Hipermedia
   - La opción "crear link" del menú contextual se encuentra deshabilitada. El sistema debe notificar al usuario y permitirle que corra el Cliente Hipermedia para poder realizar esta operación.

II. El usuario no tiene abierto ningún contexto
   - La opción "crear link" del menú contextual se encuentra deshabilitada. El sistema debe permitir que el usuario cree un contexto o abra uno existente.

III. El documento donde crea el link no pertenece al contexto actual
   - El sistema agrega automáticamente el documento al contexto actual.

Caso de uso: Mostrar los links que apuntan a un documento.

Actor: Usuario.

Propósito: Que el usuario pueda consultar los links que apuntan a un documento dado.

Descripción: el usuario, mientras visualiza/edita un documento, selecciona la opción "mostrar links que apuntan al documento" del menú Herramientas del Cliente Hipermedia. Esta opción se encuentra en el menú del Cliente Hipermedia, y no en el menú contextual (clic derecho), dado que la consulta es sobre un contexto completo.

Precondición: El usuario debe tener el Cliente de Hipermedia Abierta en ejecución, para lo cual debió ingresar al sistema con su dirección de correo electrónico. Además debe tener un contexto en uso, al cual pertenece el documento al que desea realizar la consulta. Además dicho documento debe

ser el que se está editando/visualizando actualmente, es decir, el actual en el navegador de nodos del Cliente Hipermedia.

Poscondición: El sistema muestra un navegador de links, en una ventana secundaria, con los links que tienen como destino al documento en cuestión.

Flujo principal:

I. El usuario selecciona la opción "mostrar links que apuntan al documento" del menú herramientas del Cliente Hipermedia.
II. El sistema consulta, en la base de datos Hipermedia, los links que tienen como destino al documento en cuestión.
III. El sistema muestra un navegador de links temporal en una ventana secundaria, con los links que resultaron de la consulta.
IV. El usuario, a través del navegador de links, puede consultar los links mostrados e incluso acceder a los documentos vinculados con el documento actual, es decir, a los documentos que tienen las anclas origen del link que apunta al documento actual.

Flujos alternativos:

I. El usuario no tiene en ejecución el Cliente Hipermedia
   - El sistema debe notificar al usuario y permitirle que corra el Cliente Hipermedia para poder realizar esta operación.
II. El usuario no tiene abierto ningún contexto
   - El sistema debe permitir que el usuario cree un contexto o abra uno existente.
III. El usuario no tiene un documento seleccionado
   - El sistema debe tener deshabilitada la opción "mostrar links que apuntan al documento" del menú herramientas del Cliente Hipermedia.

Caso de uso: Atravesar un link.

Actor: Usuario.

Propósito: Que el usuario pueda acceder a el o los destinos de un link.

Descripción: el usuario hace clic derecho sobre un fragmento de material resaltado como ancla y elige la opción "atravesar link" del menú contextual desplegado por el sistema. Esta opción lleva al usuario al destino del link en cuestión, previa selección de un destino en caso de tratarse de un link multicabeza.

Precondición: El usuario debe tener el Cliente de Hipermedia Abierta en ejecución, para lo cual debió ingresar al sistema con su dirección de correo electrónico. Además debe tener un contexto en uso, al cual pertenece el documento sobre el cual desea atravesar el link.

Poscondición: El sistema muestra un navegador con los destinos del link, en caso de que haya más de un destino, a partir del cual el usuario puede seleccionar el destino deseado. El documento destino seleccionado por el usuario es presentado y el cursor posicionado en el ancla destino correspondiente.

Flujo principal:

I. El usuario hace clic derecho sobre un fragmento de material resaltado como ancla.
II. El sistema muestra el menú contextual correspondiente.
III. El usuario selecciona, desde el menú contextual, la opción "atravesar link".
IV. El sistema, previa consulta en la base de datos Hipermedia, muestra un navegador con el o los destinos del link en cuestión.
V. El usuario selecciona uno de los posibles destinos.
VI. El sistema muestra el documento destino posicionado en el lugar correspondiente al ancla destino.

Flujos alternativos:

I. El usuario no tiene en ejecución el Cliente Hipermedia
   - Las opciones del menú contextual Hipermedia se encuentran deshabilitadas. En este caso, las anclas mostradas no pertenecen al Sistema Hipermedia.

II. El usuario no tiene abierto ningún contexto.
   - Las opciones del menú contextual están deshabilitadas. Las anclas mostradas no pertenecen al Sistema Hipermedia.

III. El link no tiene destinos.
   - El sistema informa al usuario.

6.5 Descripción de la Arquitectura

Las cuatro capas conceptuales propuestas por la comunidad de Hipermedia Abierta y descriptas en la sección 5.2 del capítulo 5: capa de aplicación, capa de comunicación, capa de tiempo de ejecución y capa de almacenamiento, se mapean en esta propuesta, a tres capas físicas: la capa de Aplicaciones, la capa del Servicio Hipermedia (SH) y la capa de Base de Datos Hipermedia (BDH). En la figura 6.3 se ilustra una representación gráfica de la arquitectura en la cual se muestran por izquierda las capas físicas mapeadas y por derecha las capas conceptuales.

A fin de capturar las características particulares de cada plataforma y de independizar la implementación del servicio respecto de la misma, se diseña un Cliente Hipermedia. El Cliente Hipermedia corre sobre cada estación de trabajo y actúa como *middleware* entre las aplicaciones de terceros y el Servicio Hipermedia. El Cliente Hipermedia involucra parte de la interfaz de usuario y los mecanismos de comunicación con el SH. Dicha comunicación se utiliza para transmitir información de localizaciones e invocar operaciones del servicio.

Para permitir que las estructuras Hipermedia sean compartidas entre muchos usuarios y clientes, se separa la Base de Datos Hipermedia del Servicio de Hipermedia, es decir, del programa que instancia y manipula objetos Hipermedia almacenados. Esta separación se realiza suministrando las interfaces, y permite que múltiples usuarios compartan las mismas estructuras Hipermedia a través de un Servicio Hipermedia común. En definitiva, ambos procesos (los del SH y los de la BDH) son capaces de distinguir entre varios usuarios.

Tal cual se planteó en los requerimientos no funcionales se propone utilizar protocolos estándar de comunicación como HTTP y TCP/IP para la comunicación entre las capas de la arquitectura. Luego de la figura 6.3, en la que se ilustran las capas de la arquitectura, se describen las tres capas físicas en que se ha dividido la misma.

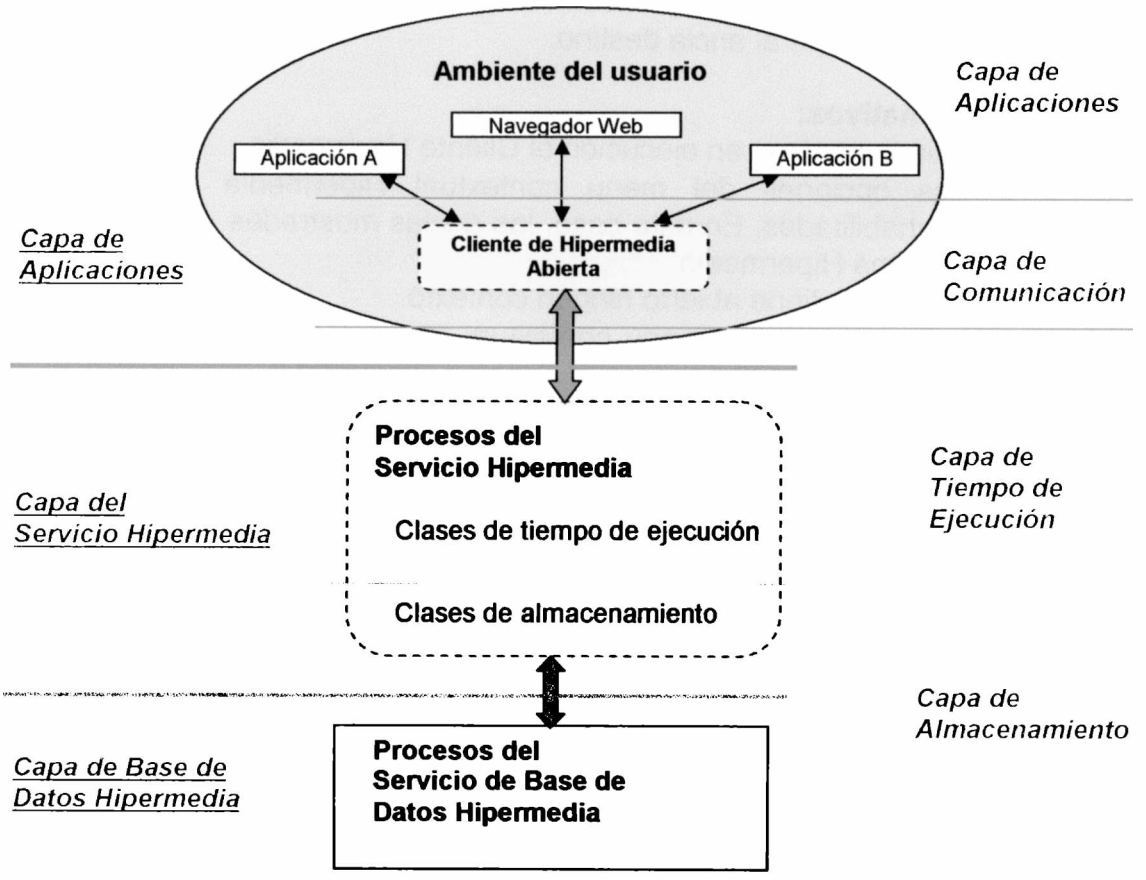


Figura 6.3: Arquitectura del SHA propuesto.

6.5.1 Procesos de la capa de Aplicación

Los procesos de esta capa, integran las aplicaciones del usuario (visores y editores responsables de mostrar y manipular contenido) con el Servicio de Hipermedia, e incluyen navegadores de estructuras como el navegador de colecciones y el de argumentos de Toulmin. Las aplicaciones de terceros

manejan tipos específicos de objetos de dato, por ejemplo, texto correspondiente al contenido de un componente textual o dibujos correspondientes al contenido de componentes CAD (*Computer Aided Design*).

Los objetos de dato son almacenados por las mismas aplicaciones en archivos separados, fuera de la Base de Datos Hipermedia. Las manipulaciones internas de los objetos de dato de las aplicaciones son operaciones sobre objetos de dato realizadas por dichas aplicaciones. Por lo tanto, la funcionalidad de la aplicación pertenece a la capa homónima, y la funcionalidad Hipermedia de las aplicaciones depende de la comunicación que realiza el Cliente Hipermedia con los correspondientes objetos instanciados de la capa de tiempo de ejecución del SH.

Una aplicación presenta el contenido para un tipo específico de componente y se comunica con los Procesos del Servidor Hipermedia a través del Cliente Hipermedia. Los procesos del Servicio Hipermedia se comunican con los de la aplicación, a través del Cliente Hipermedia, y dicen cómo mostrar un componente. Las aplicaciones comunican *locSpecs* de anclas a los objetos del almacenamiento a través del Cliente Hipermedia, e interpretan *pSpecs* provistas por el Servicio. Es decir, las *pSpecs* hacen de interfaz entre la capa de almacenamiento y la de tiempo de ejecución, y las *locSpecs* hacen de interfaz entre la de almacenamiento y la de aplicación. Las *pSpecs* modelan el almacenamiento de comportamiento de tiempo de ejecución de las estructuras Hipermedia.

El Cliente de Hipermedia, tal cual se introdujo en la sección 6.2.1, es responsable de la comunicación entre los Procesos del Servicio Hipermedia y las aplicaciones de terceros. Se propone que esta aplicación de escritorio implemente el protocolo de Hipermedia abierta OHIF, propuesto por la comunidad de Hipermedia Abierta, de modo de facilitar la comunicación entre aplicaciones de una plataforma dada y el SH y la comunicación con otros SHA relevantes que utilizan dicho protocolo.

Se presenta a continuación una discusión más detallada, desde el punto de vista arquitectural, de la aplicación Cliente Hipermedia y de su actuación para imponer las estructuras sobre documentos Web y documentos de aplicaciones de escritorio.

Cliente de Hipermedia Abierta

Para cada aplicación extendida con servicios de Hipermedia, el Cliente implementa su correspondiente envoltorio. Este envoltorio (*wrapper)* maneja la comunicación con dicha aplicación, lo que incluye el disparo de la misma y el requerimiento de que ésta abra y muestre un documento en una localización particular del mismo. La creación y presentación de anclas dentro de documentos es manejada por las aplicaciones y es muy dependiente de la apertura de las mismas.

El Cliente Hipermedia expone los servicios a través de una interfaz. Los servicios expuestos son un subconjunto simplificado de los servicios ofrecidos por el SH. Los servicios están simplificados, en el sentido de que las aplicaciones de los usuarios pueden utilizar algunos de los servicios sin tener que proveer toda la información requerida por el SH, ya que el Cliente Hipermedia es el responsable de proveer dicha información.

Esto permite que el Cliente Hipermedia requiera información de *locSpec* a partir de los documentos y que decore dinámicamente los documentos con las estructuras externas (links, notas, anclas) creadas por el usuario. También permite que el Cliente muestre selectivamente solo algunos de los puntos extremos en un documento dado, por ejemplo, el conjunto de puntos extremos para el contexto activo. Además determina el deterioro de los links, esto se debe a que tal deterioro es descubierto generalmente cuando se decora el link (resalta un punto extremo).

El Cliente se comunica con el servidor de estructura a través del protocolo de Hipermedia Abierta OHIF. Este protocolo está implementado como un protocolo textual sobre TCP/IP, en el que los mensajes son codificados utilizando XML. La comunicación a través de OHIF permite la interoperatividad e intercambio con otros Sistemas de Hipermedia Abierta. La elección de OHIF surgió luego de investigar un formato de intercambio que tendiera a imponerse como estándar, de modo que el SHA propuesto pueda exportar las estructuras creadas y que los SHA actuales puedan importar las mismas.

El formato de intercambio de Hipermedia Abierta OHIF, es soportado por Sistemas de Hipermedia Abierta actuales relevantes como: Webvise, Arakne Environment, Ariadne y Chimera de la Universidad de Colorado, y está definido formalmente por un DTD (http://www.daimi.au.dk/~les/ohif/ohif.dtd). OHIF fue derivado del modelo navegacional propuesto por el Grupo de Trabajo de Sistemas de Hipermedia Abierta, OHSWG, y además de soportar Hipermedia navegacional, es decir links, soporta compuestos y tours guiados.

6.5.2 Procesos del Servicio Hipermedia

Son los procesos que proveen Servicios Hipermedia para los Clientes Hipermedia y se encargan de manejar sesiones y permitir la colaboración coordinada sobre estructuras Hipermedia compartidas. Estos procesos corren sobre un *host* compartido por varios Clientes Hipermedia.

Dentro de esta capa física se encuentra la capa conceptual de tiempo de ejecución, que implementa el núcleo del comportamiento Hipermedia especificado por las clases genéricas descriptas en la sección 6.6.2, es decir, las clases de tiempo de ejecución están físicamente implementadas dentro de los procesos del Servicio Hipermedia. Éstos son los procesos que manejan las estructuras Hipermedia en tiempo de ejecución.

La capa del SH es responsable del manejo de links, anclas y componentes en tiempo de ejecución, y de interactuar por un lado con el Cliente Hipermedia y por otro lado con el servidor de Base de Datos Hipermedia. El SH crea instancias de las clases que implementan los conceptos de la capa de tiempo de ejecución y provee operaciones independientes de las aplicaciones para crear y manipular los objetos que implementan los conceptos de la capa de almacenamiento.

La otra capa conceptual, que está dentro de esta capa física, es la capa de almacenamiento, consistente de dos elementos: el esquema de base de dato, que implementa las clases de almacenamiento descriptas en la sección 6.6.1, y el sistema de base de datos Hipermedia físico (BDH), el cual almacena los objetos. Por lo tanto, la capa (conceptual) de almacenamiento está físicamente ubicada en dos partes, dentro del SH y en la BDH, en la cual están las estructuras de dato y los procesos de base de datos que mantienen la persistencia de las estructuras Hipermedia.

6.5.3 Procesos del Servicio de Base de Dato Hipermedia

El servidor de base de dato Hipermedia provee almacenamiento físico persistente para los objetos Hipermedia, las estructuras. Los objetos almacenados son instancias de especializaciones de las clases genéricas que implementan los conceptos de la capa de almacenamiento. Dado que se implementarán tanto el SH como la BDH con orientación a objetos, las clases pueden ser declaradas físicamente en cualquiera de las dos partes (BDH o SH) y utilizadas por los procesos de ambas.

Con la propuesta de implementar la BDH utilizando una OODB (base de datos orientada a objetos), las características tales como *locking* y notificación son meta propiedades que no necesitan especificarse cuando se utiliza el modelo de dato. De este modo no es necesario predecir qué objetos o clases de objetos necesitaran soportar *locking* o notificación de eventos. En lugar de ello, las aplicaciones pueden ser adaptadas en cualquier momento para suscribir a notificación de eventos para objetos o clases arbitrarias que estén almacenadas en la base de datos.

6.6 Diseño del Servicio Hipermedia

En esta sección se describe la propuesta de diseño de las dos capas conceptuales que forman la capa física del Servicio de Hipermedia Abierta, esto es, la capa de tiempo de ejecución y la capa de almacenamiento. Cabe reiterar que este diseño, además de cubrir los casos de uso descriptos y satisfacer los requerimientos planteados, saca provecho de las experiencias de diseño de la comunidad de Hipermedia Abierta, ya que las clases genéricas de diseño fueron obtenidas mapeando las entidades conceptuales descriptas en el capítulo 5, secciones 5.3 y 5.4.

6.6.1 Diseño de la Capa de Almacenamiento

En el capítulo 5 se introdujeron las entidades conceptuales básicas subyacentes en un sistema Hipermedia. A modo de resumen se introdujeron las *locSpecs*, que permiten identificar puntos a conectar, y las *refSpecs* que proveen un envoltorio para los *locSpecs* y permiten construir distinto tipo de estructuras, no solamente las típicas relaciones asociativas. También se introdujeron los componentes atómicos (documentos), como aquellos manejados por aplicaciones de terceros que correspondían a agrupamientos presentes en el material y formaban lo que se denomina estructura inherente del mismo. De este modo las estructuras compartían el mismo espacio conceptual que el material. El resto de las estructuras (links, notas y compuestos) tomaban la forma de agregaciones de *refSpecs*.

Clases de diseño: Capa de almacenamiento

A continuación se describen las clases de diseño de la capa de almacenamiento representadas gráficamente utilizando diagramas de modelado orientados a objeto en notación UML. Se comienza con la descripción de las clases básicas, *locSpecs* y *refSpecs*, a partir de las cuales se construyen el resto de las estructuras Hipermedia.

Especificación de Localización: LocSpec

Tal cual se describió en el capítulo 5, la especificación de localización, *locSpec*, es la entidad de nivel más bajo definida, que a diferencia del resto de los objetos Hipermedia puede "vivir" fuera del sistema como un texto legible. Del mismo modo que las URLs, las *locSpecs* pueden ser transportadas como *strings* de texto e incorporadas dentro de un Sistema Hipermedia.

Las *locSpecs* están presentes en las referencias de los puntos extremos de los links y de las anclas, en las cuales identifican localizaciones dentro del componente al cual pertenecen. Las *locSpecs* proveen 3 formas de identificar una localización dentro de un componente, las cuales, en este trabajo, se mapean a tres atributos a los que se denominará como atributos de localización de contenido:

- ObjetoEnCompoID: el ID (identificador) del objeto. Nombre único, de una localización dentro de un componente.
- ObjetoEnCompoDescEstruc: la especificación de estructura, que es una forma de localizar que describe cualquier estructura que el material, dentro del componente, pueda tener.
- ObjetoEnCompoEspecCalc: la especificación de cálculo, que provee una forma de introducir cálculo dentro de una especificación de localización.

Para *locSpecs* simples se utiliza solo una de estas 3 formas de localizar, pero en este trabajo se utilizará la especificación de la misma localización de

múltiples formas como una herramienta que permita detectar si un documento ha sido modificado fuera del control del Sistema Hipermedia. Por ejemplo, para documentos de texto se utilizará la descripción de estructura, que localizará un fragmento fijo dentro del documento, y la descripción de cálculo, como una búsqueda del texto en cuestión dentro del componente. Si el texto de la búsqueda no coincidiera con el texto determinado por la descripción de estructura (cuestión que será detectada por el Cliente Hipermedia), se detecta una inconsistencia en el ancla y se notifica al usuario.

Además de las tres formas de identificar localizaciones dentro de componente descripta, como se presentó en el capítulo 5, se requiere una identificación de los componentes que encierran esas localizaciones, es decir, de los componentes como un todo, esta especificación se mapea a los siguientes tres atributos:

- CompoID: el ID de componente el cual es único dentro de un Contexto particular. Un contexto (se describe en detalle más adelante) es una estructura completa armada sobre un material, bajo un punto de vista determinado.
- CompoDescEstruc: el descriptor de estructura se utilizará cuando haya estructuras entre componentes, por ejemplo, para identificar un componente dentro de un argumento de Toulmin.
- CompoDescCalc: la especificación de cálculo posibilita, por ejemplo, la ejecución de consultas de componentes respecto a sus atributos.

A estos últimos tres elementos se los denomina atributos de localización de componente. Como en el caso de localización dentro de componente se utilizarán especificaciones redundantes para localizar el mismo componente o conjunto de componentes. La figura 6.4 muestra la clase *locSpec* utilizando notación UML.

A lo sumo una de las dos partes de la *locSpec* (la del componente o la de dentro de componente) podrá estar vacía. Si la parte vacía es la del componente significa que la *locSpec* es aplicable a cualquier componente. Por otro lado, si la parte vacía es la de dentro de componente, significa que la *locSpec* identifica el componente entero.

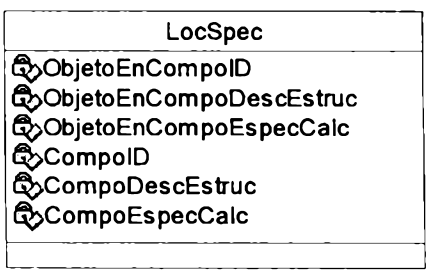


Fig. 6.4: clase *LocSpec* en notación UML.

Especificación de Referencia: RefSpec

La especificación de referencia envuelve una *locSpec* y una especificación de presentación, *pSpec*, y forma la base para referenciar anclas y puntos extremos de links y compuestos (colecciones y argumentos de Toulmin). De este modo permite que la *locSpec* (que puede tener "vida independiente" del Sistema Hipermedia) quede envuelta dentro de un objeto del Sistema Hipermedia. La clase *refSpec* se diseña como una clase abstracta de la que especializan anclas y puntos extremos.

Cada *refSpec* tiene un puntero a un objeto *locSpec*, lo que permite que varias *refSpecs* compartan la misma *locSpec*. Además cada *refSpec* incluye un ID único dentro del contexto al que pertenece. La *pSpec* es un atributo virtual (atributo abstracto) que determina cómo va a ser presentada la *refSpec*. Las *pSpec* junto con las especificaciones de cálculo de las *locSpecs* permiten especificar comportamiento de las entidades Hipermedia en tiempo de ejecución, que requiere ser almacenado. Las *pSpecs* almacenan una especificación ligera (pares atributo-valor) de atributos para la aplicación encargada de presentar un objeto Hipermedia en tiempo de ejecución.

La clase *refSpec* incluye el método *IgualLocSpec* que acepta una *locSpec* como argumento y determina si coincide con la *locSpec* envuelta por la *refSpec*. *IgualLocSpec* es utilizada durante el seguimiento de links. La operación resolver se debe implementar como un AND entre los seis elementos que constituyen la *locSpec*, es decir, todos los especificados deben ser verdaderos. Resolver toma una locSpec y devuelve el o los componentes correspondientes. La figura 6.5 muestra la clase *refSpec* utilizando notación UML.

Los atributos Fecha y Autor están diseñados para mantener los datos acerca del usuario que crea las anclas y puntos extremos, y las fechas de creación. Esta información es necesaria para satisfacer los requerimientos funcionales, en los que se planteó la necesidad de poder determinar el autor y la fecha de cada entidad estructurante. Tal cual está diseñado será posible además determinar el autor y la fecha de creación de cada ancla y punto extremo.

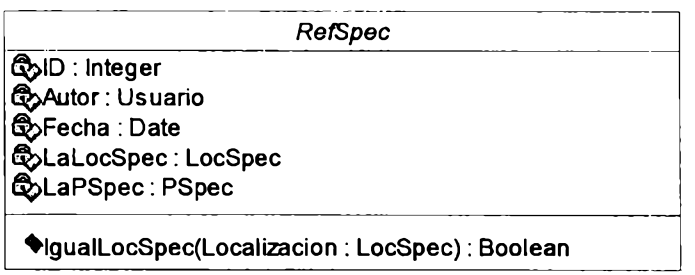


Fig. 6.5: clase *RefSpec* en notación UML.

Ancla

Las anclas estarán asociadas a componentes atómicos, mientras que los puntos extremos, como se ve más adelante, estarán asociados a componentes estructurantes. El ancla se diseña como una especialización de la *refSpec*, que identifica una localización en el contenido del componente al cual pertenece. Modela la conexión de estructuras Hipermedia con contenido existente. El ID de la *refSpec* ancla es único dentro del componente al que pertenece.

La *pSpec* está limitada a un *anclaPSpec,* que determina cómo se presenta el ancla en el contenido del componente. Debido a que un ancla está limitada a una componente particular, la *locSpec* del ancla identifica explícitamente el componente donde está incluida, utilizando el ID del mismo. Sin embargo, en la especificación de localización dentro del componente, el objeto puede ser calculado.

La *locSpec* del ancla, localiza contenido y constituye un origen o destino potencial para un link. El atributo Tipo está diseñado para indicar si se trata de un ancla común o una nota de usuario, ya que las notas se diseñan como anclas con un texto relacionado. En caso de ser un ancla tipo nota, la *anclaPSpec* se utiliza para determinar el texto de la nota y como ésta es mostrada al usuario. La figura 6.6 muestra la clase *ancla* utilizando notación UML.

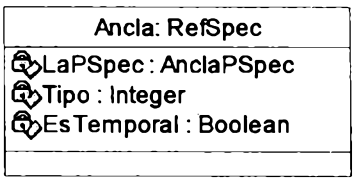


Fig. 6.6: clase *Ancla* en notación UML.

Solo las anclas pueden ser temporales, no los puntos extremos, por lo tanto se diseña el atributo EsTemporal para las anclas y no para las *RefSepcs*. El comportamiento característico de la Hipermedia, seguir links y abrir compuestos, depende de la capacidad para hacer coincidir anclas con puntos extremos.

Nota

Tal cual se describió en el punto anterior, las notas se diseñan como anclas con una *pSpec* que describe la forma en que será presentada en el contenido (reemplazar, *popup*, prefijo, postfijo). Las anclas se crean cada vez que el usuario agrega un ancla a un link, por lo que casi todas las anclas son apuntadas por algún punto extremo de link. Sin embargo, las anclas pertenecientes a notas, no son apuntadas por ningún punto extremo a pesar de estar resaltadas del mismo modo que un ancla común.

Punto Extremo

Un punto extremo es una especialización de una *refSpec* asociado con un link o un compuesto. En el caso más simple es una *refSpec* de un link y apunta a un ancla. El Tipo del punto extremo se diseña para especificar la direccionalidad o relación estructurante asociada con ese punto extremo, es decir, cómo se comporta dicho punto extremo. Para puntos extremos de links, el Tipo puede ser Origen, Destino o Ambos, mientras que, para puntos extremos de compuestos que representen argumentos de Toulmin, el tipo puede ser Conclusión, Fundamento, Garantía o Respaldo.

En la *refSpec* de un punto extremo, la identificación del componente no tiene ninguna restricción (a diferencia de las *refSpecs* de anclas), es decir, puede ser cualquier componente incluso un componente que se obtiene a través de un cálculo. Por este motivo se diseña un atributo Padre, que determina el link o compuesto al que pertenece dicho punto extremo. El ID del punto extremo es único dentro del componente estructurante al que pertenece. Otra consideración a tener en cuenta es que un punto extremo calculado, como es el caso de uno perteneciente a un link global, no apunta a un ancla, sino a cualquier ocurrencia de un texto en cualquier componente.

Este diseño de punto extremo, en el que los puntos extremos apuntan a anclas (para lugares fijos del material), en lugar de apuntar directamente al material, permite que si el material cambia, el punto extremo siga apuntando al mismo ancla, incluso múltiples puntos extremos pueden apuntar al mismo ancla, y lo único que cambia ante cambios en el material es el ancla, no todos los puntos extremos que apuntan al ancla.

A continuación se describe la política que se seguirá para interpretar los valores de direccionalidad mínimos previstos, durante una travesía:

- Atravesar hacia delante abre todos los puntos extremos etiquetados como destino o ambos.
- Atravesar hacia atrás abre aquellos etiquetados como origen o ambos.

El diseño deja abierta la posibilidad de agregar puntos extremos con direccionalidad Ninguna. Los puntos extremos con direccionalidad Ninguna podrían estar ocultos a las dos políticas de travesía descriptas y se podrían utilizar para mantener puntos extremos ocultos durante las travesías. Un editor de link invocaría una operación separada para abrir todos los puntos extremos sin importar su direccionalidad, cuando sea necesario.

La *pSpec* está limitada a una *pSpec* de punto extremo que determina cómo el punto extremo es presentado en el contenido del componente, incluso se utilizará para determinar si, durante una travesía, el material apuntado por el punto extremo reemplaza al material existente o se presenta en una nueva ventana. Dicha *pSpec* se utiliza además en los visores/editores de links y

compuestos provistos por el Cliente Hipermedia. La figura 6.7 muestra la clase *PuntoExtremo* utilizando notación UML.

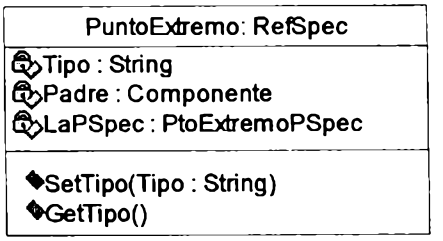


Fig. 6.7: clase *PuntoExtremo* en notación UML.

Contexto

Dado que para el ambiente de investigadores presentado se trabaja con un Hiperespacio limitado, como lo es un conjunto de escritos científicos, y sumado a que sobre cada conjunto de escritos se prevén crear distintas estructuras que definan distintas vistas sobre el mismo material, en esta propuesta a los Hiperespacios se los denomina contextos. Estos contextos se diseñan de modo que no puedan contener otros contextos. Cualquier usuario, registrado, puede extender y modificar un contexto. Por lo tanto, un contexto constituye una Hipermedia limitada a representar un punto de vista sobre un conjunto de escritos científicos compartidos por una comunidad de investigadores.

Cada contexto es un contendor de componentes y *refSpecs*. La contención implica que si el Contexto es eliminado, se eliminan todos los componentes y *refSpecs* contenidos. De este modo, cada Contexto tiene su propio espacio de nombres dentro del cual los componentes (y *refSpecs*) están unívocamente identificados y las *locSpecs* restringidas al rango de direcciones del contexto al que pertenecen. La resolución de una referencia en una *locSpec* se realiza en relación al contexto activo, es decir las IDs de componentes son únicas dentro del contexto, por lo que las *locSpec* están limitadas a localizar en ese rango de IDs.

Por lo expuesto, un contexto es una colección indexada de otros objetos Hipermedia: componentes y *refSpecs* (de puntos extremos y anclas). Cada contexto se identifica a través de un ID. En tiempo de ejecución un contexto es manejado por una sesión. Los componentes y *refSpecs* son creados y borrados a través de llamadas a la sesión, quien a su vez invoca las operaciones respectivas del Contexto: AgregarCompo, EliminarCompo, AgregarRefSpec y EliminarRefSpec. La sesión también maneja el trabajo de travesía de estructuras.

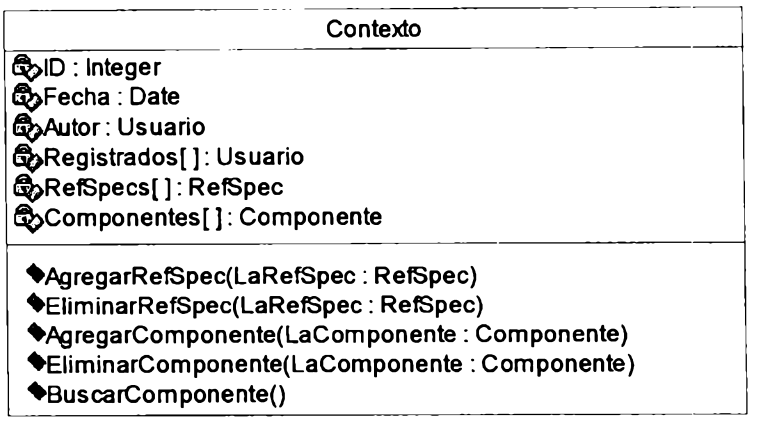


Fig. 6.8: clase *Contexto* en notación UML.

Las aplicaciones de terceros tomarán la responsabilidad de almacenar los contenidos de los componentes atómicos. La Hipermedia brindada consistirá de envoltorios para estos documentos, y de estructuras para vincular y organizar ese contenido. La figura 6.8 muestra la clase *Contexto* utilizando notación UML.

Componente

El componente se diseña para ser instanciado directamente como un componente atómico y al mismo tiempo, sirve como superclase para componentes links y compuestos. Los componentes son identificados por un ID que es único dentro del Contexto.

Dado que el objetivo es diseñar clases para estructuración simples con roles y capacidades claramente identificables, se utiliza una especialización diferente de componente para cada rol funcional. Por lo tanto, links y compuestos se implementan como clases separadas, ya que por un lado son interconexiones atravesables y por el otro son estructuras orientadas a la apertura.

La *pSpec* del componente maneja las características de presentación por defecto y es consultada cuando un componente es abierto. Si la apertura resulta de seguir un link (o de abrir un compuesto), entonces la *pSpec* del componente es combinada con la *pSpec* del punto extremo del link (o del compuesto) y con la *pSpec* del ancla destino. Además, las *pSpecs* de componentes, almacenan información para arrancar y configurar (localizarla en un lugar correspondiente del contenido del componente y marcar las anclas correspondientes) la aplicación manejadora del mismo.

Todas los componentes (atómicos, links y compuestos) están diseñados para incluir una colección estructurada de anclas y un contenido. El contenido es un objeto de dato o una referencia a un objeto de dato almacenado separadamente. Si el componente contiene el objeto de dato que representa el contenido, si se borra el componente, se borra su contenido. Los componentes

incluyen operaciones para agregar y eliminar anclas. Las anclas son las responsables de mantener la consistencia de las *locSpecs* que apuntan al contenido del componente.

También se prevé, a través de un atributo, que el componente pueda ser temporal y/o calculado. De este modo cualquier componente en el sistema puede hacerse temporal poniendo el atributo en un valor correspondiente, en cuyo caso el componente no será almacenado en la base de datos. Sin embargo, el componente temporal será almacenado si un link, u otro componente, lo referencian.

Si el contenido referenciado por un componente no está disponible debido a cuestiones que están fuera del alcance del Sistema Hipermedia (fue eliminado el documento referenciado), la operación de travesía debe capturar la excepción del sistema de archivos y pasar al usuario una excepción de link colgado.

Componente atómico

Los componentes atómicos, instancias directas de la superclase componente, pueden tener una estructura que no es visible por el Sistema Hipermedia, por ello se denominan de esta forma. El contenido de un componente atómico es un documento manejado por una aplicación de tercero. Los componentes atómicos reflejan la estructura preexistente del material y proveen un envoltorio Hipermedia para una pieza de información.

Dentro de los atributos de los componentes atómicos está la fecha de última modificación. Este atributo se diseña, entre otras cosas, para mantener la consistencia de las anclas. Si la fecha de última modificación del contenido apuntado por el componente es posterior a la fecha de última modificación del componente atómico, puede implicar recalcular sus anclas para mantener la consistencia de las mismas.

Las anclas son una colección estructurada de *refSpecs* que localizan fragmentos de material de lo apuntado por el contenido. El contenido puede ser, por ejemplo, una URL que especifica una página Web, en cuyo caso todas las anclas que tienen el ID del componente, no cambian si la página es movida, solo se cambia la URL del componente que apunta al contenido. La figura 6.9 muestra la clase *componente* utilizando notación UML.

La *PSpec* del componte determina cómo se presentará visualmente cada entidad estructural del Sistema Hipermedia en la interfaz del usuario. Esto puede requerir modificar la apariencia del material envuelto por el componente o crear una nueva interfaz gráfica, por ejemplo para presentar un compuesto al usuario. La *pSpec* de un componente atómico dispara la aplicación correspondiente, mientras que la de una colección dispara la interfaz de usuario manejada por el Cliente Hipermedia.

Dado que los componentes se diseñan con atributos que indican el autor de las mismas, es posible determinar el autor de cada estructura, incluso se pueden filtrar, dentro de un contexto, las estructuras creadas por un investigador determinado.

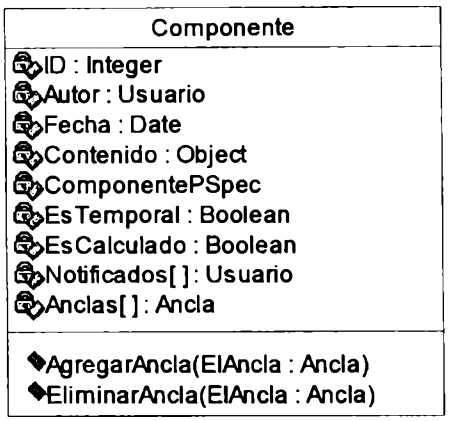


Fig. 6.9: clase *Componente* en notación UML.

Componente link

Tanto los links como los compuestos son componentes especializados (componentes para estructurar) e incluyen colecciones de puntos extremos así como también operaciones para agregar y eliminar puntos extremos. Dado que el link es una especialización de componente, tiene anclas y métodos para agregar y eliminar anclas, lo cual permitirá que puedan ser referenciados por puntos extremos de otros links, es decir se pueden utilizar links a links. Los links también pueden tener contenido o apuntar a un contenido directamente.

Los links contienen una colección estructurada de *refSpecs* aumentadas con atributos, entre los cuales está el atributo direccionalidad. A través de la utilización de *refSpecs* se soportan links calculados y estáticos. La figura 6.10 muestra la clase *link* utilizando notación UML.

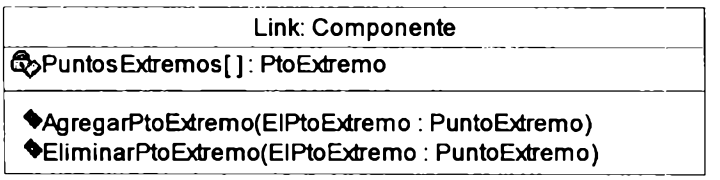


Fig. 6.10: clase *Link* en notación UML.

Componente link global

Dado que, tal cual se ha planteado, el objetivo es diseñar clases para estructuración simples con roles y capacidades claramente identificables, se diseña el link global como una especialización del componente link. Los links globales están diseñados para soportar la utilización de un vocabulario común

consensuado compartido entre los investigadores. Son conceptualmente algo diferentes a los links comunes, cuando uno de estos links es activado, es atravesado desde un punto extremo global hacia una localización especificada por un punto extremo destino fijo.

Los links globales se especifican con un punto extremo destino fijo, con sus elementos asociados (ancla y nodo), y una colección de puntos extremos origen denominados anclas globales. Estas anclas globales no tienen un componente asociado, sino que especifican un texto que puede pertenecer a cualquier componente. Nótese que la globalidad del link no esta asociada con el link completo, sino con el punto extremo origen.

Componente compuesto

Al igual que los links, los compuestos son componentes especializados que incluyen colecciones de puntos extremos así como también operaciones para agregar y eliminar puntos extremos. Aunque links y compuestos son casi idénticos con respecto al diseño, el comportamiento en tiempo de ejecución es diferente. Cuando un compuesto (colección o argumento de Toulmin) es abierto, éste brinda los componentes que sus puntos extremos referencian, en lugar de esto, los links son atravesados desde y hacia uno o más puntos extremos. Los puntos extremos de un compuesto indican qué componentes lo forman y qué relación de estructura (utilizando otra concepción del atributo direccionalidad de los links) existe entre estos componentes.

Cualquier compuesto puede ser el resultado de un cálculo o puede ser creado manualmente por el usuario. Las colecciones son un ejemplo de un compuesto creado por el usuario y los compuestos creados por el sistema a partir de la ejecución de una consulta, son ejemplos de compuestos calculados. En este último caso, un atributo perteneciente a la *pSepc* del compuesto mantiene la información utilizada para realizar el cálculo. El contenido del compuesto puede ser recalculado según demanda o automáticamente.

Para este trabajo, se utilizan tres especializaciones de la clase compuesto. Los compuestos que representan argumentos de Toulmin, los que representan colecciones y los que representan los distintos navegadores utilizados para la interfaz del usuario. Tales navegadores, que tal cual están diseñados pueden almacenarse, brindan una visión gráfica de una colección o de un argumento de Toulmin o permiten mostrar el resultado de consultas realizadas sobre un Contexto. La *pSpec* del compuesto es utilizada para representar visualmente el mismo en una interfaz de usuario. La figura 6.11 muestra la clase *compuesto* utilizando notación UML.

El cálculo en un compuesto puede estar asociado, además de a los puntos extremos, al compuesto completo. Por ejemplo un compuesto que incluye los componentes creados en el 2000, tiene un conjunto de 0 a n puntos extremos, calculados en base a una especificación incluida en la *pSpec* del mismo. En este caso es un compuesto calculado, ya que el compuesto

completo se calcula, no solo sus puntos extremos. Se mantiene como clase de almacenamiento, ya que con este diseño el usuario puede decidir almacenar el resultado de este tipo de consultas.

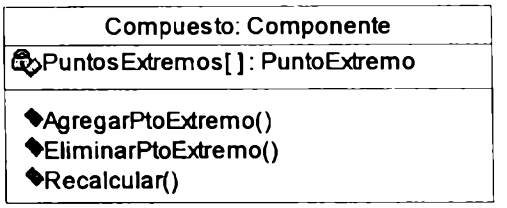


Fig. 6.11: clase *Compuesto* en notación UML.

Compuesto Colección

Se diseña como una especialización de la clase genérica compuesto, en la cual los puntos extremos referencian componentes atómicos, es decir, los elementos apuntados por los puntos extremos están limitados en su tipo. Además la localización de los componentes del compuesto se realiza a través de referencias, permitiendo de este modo que un componente pueda pertenecer a más de una colección. No se impone límite a la cantidad de elementos de este compuesto, y se diseña como una colección desestructurada de elementos, es decir, la única relación que existe entre los componentes es que pertenecen a la misma colección.

Esta especialización, se diseña además, para que la relación entre el compuesto y sus miembros tenga una sola dirección, es decir, los componentes de un compuesto colección no saben de quién, ni de cuántos, compuestos son miembro. Esto facilita la eliminación de compuestos colección, ya que no se dejan punteros colgados desde los componentes miembros a sus respectivos compuestos. Otra ventaja de este tipo de relación es que se pueden agregar o eliminar componentes a un compuesto sin bloquear el acceso a los componentes, ya que solo el compuesto colección es modificado en tal operación, esto facilita el trabajo cooperativo.

El inconveniente de este tipo de relación, en una dirección, se presenta cuando se elimina un componente que puede pertenecer a varios compuestos, en cuyo caso la solución prevista es utilizar una operación que determine los componentes compuesto que apuntan a un componente dado.

Compuesto Argumento de Toulmin

Se parece a un link multicabeza con puntos extremos con nombre. Se diseña como un compuesto contenedor de componentes, en lugar de referenciar componentes como en el caso de colecciones. Por lo tanto, la eliminación del compuesto implica la eliminación de sus componentes. En este

caso, la cantidad de componentes del compuesto está limitada a cuatro y existe relación entre los componentes (garantiza, sustenta, etc.).

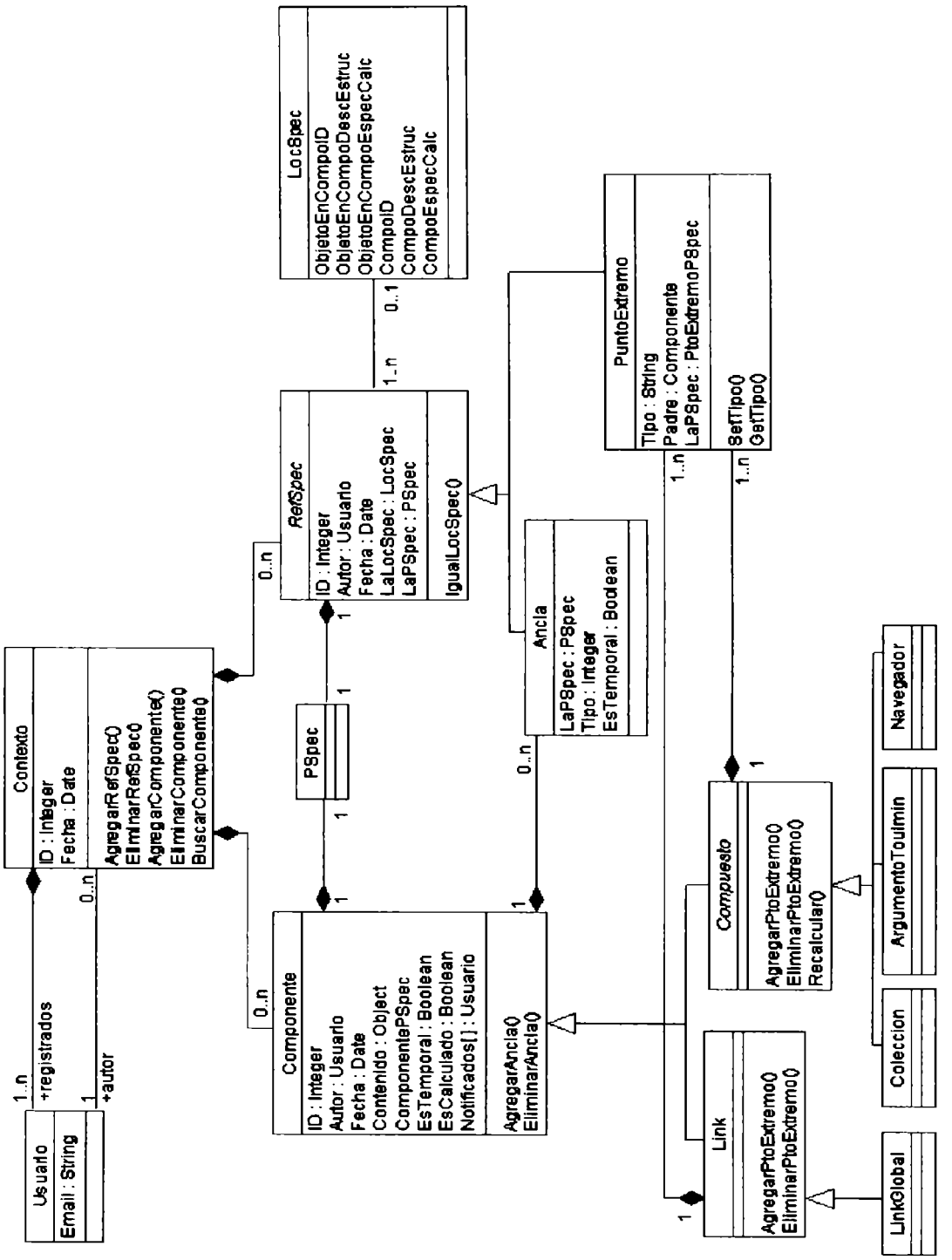


Fig. 6.12: modelo del Diseño de la Capa de Almacenamiento.

Como en el caso de colecciones, los compuestos Argumentos de Toulmin también apuntan a componentes completos, es decir, con la especificación de dentro de componente vacía. Además, también como en el caso de compuestos colección, apuntan a componentes atómicos, es decir, no se permiten subestructuras dentro de ninguna de las especializaciones de los componentes compuestos.

Compuesto Navegador

El compuesto navegador modela, además de los distintos navegadores del Cliente Hipermedia (navegador de nodos y navegador de links), los navegadores resultantes de consultas realizadas por los investigadores. Un ejemplo de este último caso es el navegador de links que muestra todos los links del contexto que apuntan a un documento dado. En este caso los puntos extremos del compuesto apuntan a componentes de tipo link, los resultantes de una consulta sobre el contexto, además la cantidad de puntos extremos de este navegador no es fija, y puede ser de cero o varios puntos extremos. El atributo *PSpec* del compuesto mantiene la información para realizar el cálculo, y el contenido del mismo puede ser recalculado a demanda o automáticamente.

RecalcCompos es instanciada en esta especialización de la clase compuesto, dicha operación recalcula los componentes del compuesto. Esta operación es invocada automáticamente cuando se presenta el compuesto al usuario o puede ser invocada por el investigador a través de la interfaz de usuario. El navegador de link del Cliente Hipermedia, por ejemplo, es un compuesto navegador que filtra su contenido para que los elementos sean sólo los componentes link que pertenecen al contexto activo.

En la figura 6.12 se muestra el modelo de diseño de la capa de almacenamiento del Servicio Hipermedia. Cabe destacar que en dicho modelo, y dado que se trata de una fase de iniciación, se incluyeron las clases y relaciones más relevantes a fin de facilitar la legibilidad del mismo. Por ejemplo no se han incluido las distintas especializaciones de la clase *PSpec*, tal es el caso de AnclaPSpec, PtoExtremoPSpec, etc.

6.6.2 Diseño de la Capa de Tiempo de Ejecución

Tal cual quedó evidenciado en la descripción de la arquitectura, en el diseño del SHA se distinguen dos partes, una que almacena y recupera material Hipermedia (la capa de almacenamiento descripta), y otra a través de la cual los usuarios navegan y manipulan esas estructuras. Tal distinción posibilita que los investigadores colaboren, trabajando sobre diferentes plataformas, y compartan un servidor de almacenamiento común.

La capa de tiempo de ejecución tiene responsabilidad sólo del comportamiento dentro de lo que es Hipermedia, incluyendo navegación y presentación de anclas de links, y es en el mayor grado posible, independiente de las interfaces de las aplicaciones de terceros.

Clases de diseño: Capa de Tiempo de Ejecución

A continuación se describen las clases de diseño de la capa de tiempo de ejecución representadas gráficamente utilizando diagramas de modelado orientados a objeto en notación UML. El manejo de comportamiento Hipermedia en tiempo de ejecución es realizado por tres entidades básicas: el manejador de sesión, la sesión y la instancia. El manejador de sesión es el objeto abarcador de tiempo de ejecución que maneja múltiples sesiones, cada una en control de un simple contexto. Dentro de cada sesión hay instancias que manejan la vista y edición de los componentes particulares.

Manejador de sesiones

El manejador de sesiones es la interfaz primaria de tiempo de ejecución. Incluye una colección de sesiones, así como también operaciones para abrir nuevas sesiones y cerrar existentes. El manejador de sesión invoca además operaciones tales como CrearContexto y EliminarContexto. Un mismo contexto puede tener varias sesiones abiertas manejadas por distintos investigadores. En la figura 6.13 se muestra la clase *ManejadorDeSesiones* utilizando notación UML.

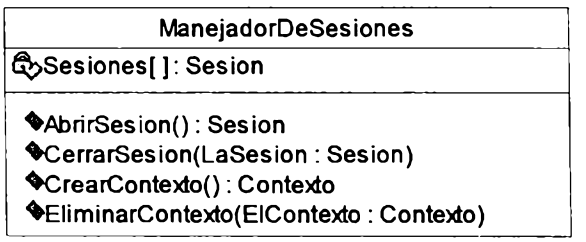


Fig. 6.13: clase *ManejadorDeSesiones* en notación UML.

Sesión

Como en un momento dado un mismo usuario puede estar visualizando y/o editando varias instancias de componentes, se utiliza la entidad sesión, la cual permite saber, momento a momento, la relación entre los componentes y sus instancias. De este modo, cuando un usuario pretende acceder a un contexto realiza la apertura de una sesión.

Por lo tanto la sesión es la entidad de tiempo de ejecución que maneja la actividad dentro de un contexto particular y corresponde a dicha entidad de la capa de almacenamiento. Un Contexto podría tener múltiples sesiones en ejecución simultánea para distintos usuarios sobre distintas plataformas.

Una sesión incluye el contexto accedido, una colección de instancias que pertenecen a los componentes de la capa de almacenamiento que se utilizan en la sesión y una colección de *refSpecs*. Además es necesario que la sesión mantenga una historia que contenga las operaciones que se han hecho

desde que se abrió la misma. En la figura 6.14 se muestra la clase *Sesion* utilizando notación UML.

Del mismo modo que se disponía de un conjunto de operaciones que permitían acceder y modificar la capa de almacenamiento, también se dispone de un conjunto de operaciones que lo permiten sobre la capa de tiempo de ejecución. Las operaciones mínimas requeridas son:

- Abrir una Instancia: que devolverá, a partir de un componente de la capa de almacenamiento, una instancia del mismo como parte de la sesión.
- Cerrar una instancia: una función "inversa" de la función AbrirInstancia, es decir, a partir de una instancia devolver un componente que "refleje" el estado actual de la instancia.
- Agregar y Eliminar Componentes y *refSpecs*: los componentes y las *refSpecs* son agregadas y borradas a través de llamadas a la sesión, quien a su vez invoca a las operaciones respectivas del contexto.
- Búsqueda de componentes y *refSpecs*: estas operaciones proveen una forma de acceder (Base de Datos) a los componentes y *refSpecs* que pertenecen al Contexto.
- Travesía: utilizada durante la travesía de estructuras. Dada una *locSpec*, esta operación devuelve *refSpecs* que devuelven *igualLocSpec* en verdadero. A partir de allí se obtienen todos los links aplicables, esto es, aquellos link en los que un punto extremo coincide con la *locSpec* dada.

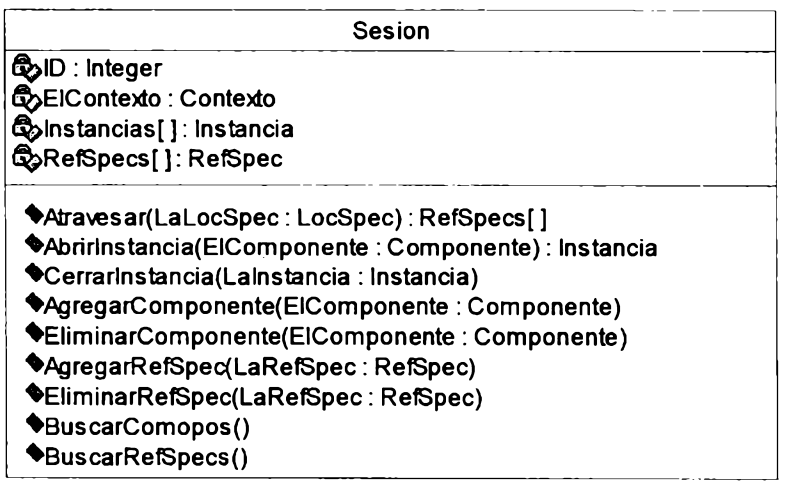


Fig. 6.14: clase *Sesion* en notación UML.

Instancia

Las instancias manejan el comportamiento de tiempo de ejecución de los componentes del contexto que han sido accedidas. Dado que puede haber más de una instancia de un mismo componente, a cada instancia se le asigna un identificador único. Tal cual se ha descripto, la manera en que el componente

se presenta al usuario depende de las preferencias que tenga definidas en las especificaciones de presentación, *pSpecs*, para el mismo.

La instancia, que como se planteó maneja la vista, edición y comportamiento de un componente, incluye operaciones para crear, eliminar y buscar anclas, esta última busca en la colección de anclas asociadas al componente. Para las Instancias de componentes link también se especifican las formas a través de las cuales una travesía se mueve desde uno o más puntos extremos a otros. Además incluye una operación, Abrir, que permite brindar el componente posicionado en un ancla particular. A partir de la clase instancia habrá especializaciones para manejar cada tipo de componente, es decir para componentes atómicos, links y compuestos.

Para manejar el comportamiento de los componentes miembros de un compuesto, se diseña una subclase de instancia, que a diferencia de una instancia común, no invoca una aplicación para presentar el contenido del componente. En lugar de ello provee una interfaz para acceder a atributos del componente y para invocar una instancia real cuando sea necesario. Esta subclase de instancia aparecerá en la interfaz del usuario como un icono en la ventana del compuesto. El icono permitirá al usuario ver y consultar los componentes pertenecientes al compuesto en una interfaz gráfica sin tener que invocar las aplicaciones que manejen sus componentes.

Además se diseñan especializaciones de instancias para manejar los tres tipos de compuestos implementados (colecciones, argumentos de Toulmin y navegadores). La especialización para el compuesto navegador se diseña de modo que restrinja sus componentes a un tipo particular (links por ejemplo) y es calculado. Esta subclase de compuesto es utilizada en navegadores y consultas que recolecten, por ejemplo links desde un contexto. Además es un compuesto temporal, es decir, se crea en tiempo de ejecución. Para esto se invoca la operación abrir instancia, la cual crea una instancia de compuesto temporal. Este compuesto no existe a nivel base de datos, y si permanece temporal, al cerrar la sesión no es almacenado. Por otro lado, si se configura como permanente, se invoca a agregar componente de la sesión y se almacena. En la figura 6.15 se muestra la clase *Instancia* utilizando notación UML.

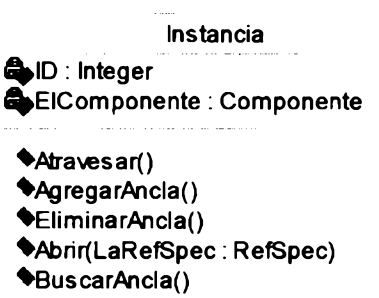


Fig. 6.15: clase *Instancia* en notación UML.

Como las *pSpec* son aplicadas a los puntos extremos de los compuestos, permiten capturar información como posición y tamaño en la ventana en que se muestran los miembros de un compuesto. Como cada componente tiene además *pSpec*, ésta puede ser combinada con la del punto extremo del compuesto. Tomando como ejemplo la posición en la ventana del compuesto, un componente puede almacenar la posición por defecto en su *pSpec*.

En la figura 6.16 se muestra el modelo de diseño de la capa de tiempo de ejecución del Servicio Hipermedia. Cabe destacar que en dicho modelo, y dado que se trata de una fase de iniciación, se incluyeron las clases y relaciones más relevantes a fin de facilitar la legibilidad del mismo.

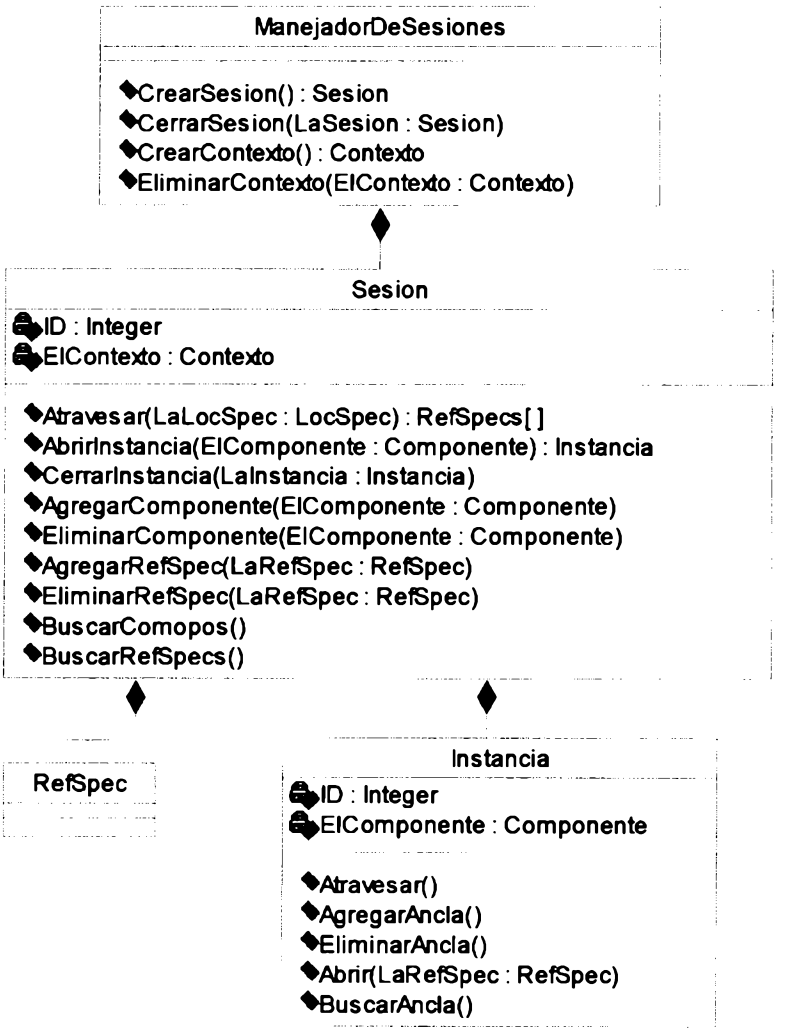


Fig. 6.16: modelo del Diseño de la Capa de Tiempo de Ejecución.

6.7 Diseño de la operación de Travesía

Como se planteó desde el principio del capítulo, se ha elegido un método de desarrollo orientado al prototipado, por lo que tanto en las secciones anteriores como en esta sección, no se realizan especificaciones excesivas. Por tal motivo, en esta sección se describen solamente las actividades llevadas a cabo durante la operación de travesía y las consideraciones más relevantes a tener en cuenta a la hora de implementarla.

Se ha elegido la especificación de esta operación debido a lo crucial que resulta en el contexto de la Hipermedia. Además, a partir del diseño descripto, se puede inferir el diseño de operaciones de travesía más específicos. El algoritmo descripto, al igual que el resto del diseño, está basado en las investigaciones de la comunidad de Hipermedia Abierta y soporta comportamientos avanzados de vinculación.

En esta propuesta, la operación de travesía está involucrada en las clases sesión e instancia, pero solo para instancias que representan componentes de estructuración. No existe esta operación a nivel manejador de sesión, ya que no se permiten links entre contextos. Se describe a continuación los pasos detallados de la operación:

1. **Crear o hacer clic sobre un ancla origen:** identificar o crear una *locSpec* origen. Esto puede realizarse de alguna de las siguientes cuatro formas:

   a. El usuario realiza una selección sobre el material, la cual no estaba previamente resaltada, por ejemplo, un fragmento de texto o un objeto de un gráfico. A partir de esta selección se construye, en el momento, una *locSpec* temporal (este ancla temporal dura solo por esta travesía). El ancla temporal pertenece a la instancia del componente en cuestión. Este caso a, es el utilizado para el seguimiento de links globales.

   b. El usuario escribe la representación textual de una *locSpec*, de forma similar a como escribe una URL en un navegador. En este caso, habrá una interfaz Hipermedia que permita al usuario escribir la ID o algún otro atributo de una *locSpec*. Esa *locSpec* se utiliza como ancla origen.

   c. El usuario identifica un ancla en algún componente Hipermedia. Éste es el caso más común, en el que el usuario hace clic en lugares resaltados en el material. La *locSpec* del ancla se utiliza como *locSpec* de origen.

   d. El usuario identifica un componente entero. En este caso, una *locSpec* temporal es creada, identificando el componente entero, es decir, omitiendo la especificación de dentro de componente de la *locSpec*. Este es el caso, por ejemplo, utilizado para determinar los links que apuntan a un componente dado.

2. **Buscar componentes estructurantes que contengan el ancla origen:** dentro del contexto actual, se mapea desde la *locSpec* origen a una colección de *refSpecs* coincidentes, para lo cual se invoca a la travesía de la clase sesión. Se busca, en la colección de *refSpecs*, por *refSpecs* que devuelvan *igualLocSpec* en verdadero, al recibir como argumento la *locSpec* origen. Se debe tener en cuenta que un ancla origen puede ser punto extremo de múltiples links. Los "componentes mapeados" (links o compuestos) que contienen estas *refSpecs* serán utilizadas para identificar los destinos de la operación de travesía. Puede haber refSpecs coincidentes que sean anclas. En este caso el componente mapeado es un componente atómico, por lo que se debe determinar qué puntos extremos resuelven a estas anclas. Si se trata de un link global, la *locSpec* es buscada del mismo modo que para links "comunes", en la colección de *refSpec*, en la que previamente haya sido registrada el ancla global origen del link global.

3. **Obtener los puntos extremos destino de los componentes estructurantes:** se invoca la operación travesía de cada una de las instancias de los componentes mapeados. Esta operación se implementa como un mapeo desde alguno de los puntos extremos (origen y ambos, generalmente) a otros (destinos y ambos generalmente). Aquí es donde la direccionalidad de un punto extremo entra en juego. Para cada componente mapeado, se pueden obtener varios puntos extremos destino. En este caso el SHA puede mostrar una interfaz para que el usuario seleccione uno de los destinos. Dependiendo el tipo de link, la semántica de esta suboperación toma una de las siguientes dos formas:

   a. Link multicabeza: considerando, por ejemplo, un link con 5 puntos extremos, dos de los cuales tienen direccionalidad etiquetada como Origen, dos Destino y uno Ambos. La semántica de travesía, si se parte desde un punto extremo origen, consiste en brindar los puntos extremos destino y los puntos extremos ambos. Por otro lado si se parte desde un punto extremo destino (para travesías en reversa), brinda los dos puntos extremos origen y el punto extremo ambos. Por último, si se parte desde el punto extremo ambos, se brindan los otros cuatro puntos extremos como resultado.

   b. Link global: el punto extremo origen es una *refSpec* cuya *locSpec* tiene especificado un texto, el seleccionado por el usuario. Esta *locSpec* es buscada en todos los puntos extremos de los links del contexto y hallada en el punto extremo origen calculado de algún link global. El resultado es brindar el punto extremo destino.

4. **Resolver las refSpecs destino a localizaciones en componentes destino:** dadas las *refSpecs* resultantes del paso anterior, resolver (calculando si es necesario) a los componentes y anclas dentro de componentes. Para cada punto extremo, se utiliza el contexto activo, es decir, se utiliza buscarCompos y buscarRefSpecs de sesión, para traer los

componentes destino. En definitiva, dada una *locSpec* se siguen las instrucciones implicadas en sus atributos para buscar el componente resultante o ancla resultante, o para realizar el cálculo dictado por sus atributos. El resultado es una colección de componentes y/o localizaciones dentro de componentes.

5. **Mostrar el o los componentes destino en sus respectivas localizaciones:** se invoca la operación Abrir de la instancia de cada componente destino (el usuario previamente puede filtrar los componentes destino), la cual abre el componente, probablemente invocando primero la aplicación (de tercero si es componente atómico o una interfaz provista por el Cliente Hipermedia si es un compuesto) que lo maneja, y lo posiciona en la *refSpec* destino (resaltando el ancla destino). Es decir, acceder (recuperar de BD) y mostrar o resaltar el material identificado como resultado de la resolución de los puntos extremos del punto cuatro. En este paso se utilizan las *pSpecs* relevantes, es decir, las *pSpecs* de los puntos extremos, de los componentes y anclas destino, combinándolas si es necesario.

Este algoritmo de travesía descripto es general, y sirve de base para implementar travesías más específicas. Por ejemplo, cuando se mueve desde anclas a links que apuntan a éstas, se ejecutan sólo los pasos 1 y 2. En este caso se puede utilizar una interfaz del Cliente Hipermedia, como un navegador de links, que muestra el resultado. Otro caso es comenzar desde un componente link en lugar que desde un ancla origen, ejecutando sólo los pasos 3, 4 y 5. Este último es el caso de la simple resolución de una *locSpec* que flota libremente (no empaquetada dentro de una refSpec) a la localización del componente al que referencia. Dos ejemplos típicos de este último caso son: el seguimiento de link en la Web y la apertura de compuestos en un SHA.

A continuación se describen algunos detalles de diseño a tener en cuenta en la implementación de la operación de travesía:

- La operación de búsqueda de *locSpecs* (generalmente origen) dentro de las *refSpec* de un contexto, es una operación que se lleva a cabo para cada operación de travesía de la sesión, por lo que necesita ser implementada en forma eficiente. Por lo tanto, se propone implementarla como una búsqueda en una tabla *hash*. De modo que, todas las *refSpecs* de un contexto deben ser registradas en dicha tabla. La tabla se debe mantener actualizada a medida que las *refSpecs* se agregan, modifican y eliminan. A partir de las *refSpecs* devueltas, a través del punto extremo que las contiene, se accede al componente link o compuesto al que pertenecen.

- El *test* de coincidencia de una *locSpec* origen en el conjunto de refSpec del contexto, es manejado por el procedimiento *igualLocSpec* de las *refSpecs*. El comportamiento exacto de esta coincidencia es otra cuestión de implementación. Sin embargo se describen, con ejemplos, un par de las características que deben tenerse en cuenta. Si una *refSpec* potencialmente

coincidente tiene un ObjetoEnCompoID (objeto dentro de componente) no vacío, entonces la *locSpec* dada, debe también tener el ObjetoEnCompoID no vacío. Por otro lado, si la *refSpec* provee una búsqueda, y la búsqueda coincide con la *locSpec* dada, entonces no importa si la *locSpec* tiene además un ObjetoEnCompoID. En otras palabras un punto extremo de un link global puede coincidir con un ancla fija.

- El comportamiento de la travesía de un link, en cuanto a si al atravesarlo el nuevo material por defecto reemplaza la configuración actual o se solapa a la actual, también depende de la implementación del sistema y las necesidades de la comunidad de los usuarios. Las *pSpecs* se utilizarán para determinar este comportamiento en caso de que sea necesario cambiar dinámicamente entre ambas alternativas.

Capítulo 7

Conclusiones y Trabajos Futuros

La complejidad actual requiere en general de trabajos de investigación grupales e interdisciplinarios. La comunicación digital ha permitido, entre otras cosas, que grupos de investigadores distribuidos geográficamente puedan trabajar en forma cooperativa en la elaboración de nuevo conocimiento.

Dentro de un marco de investigación conceptual actual, los investigadores trabajan cooperativamente con material (producciones científicas) propio y de terceros, representado sobre distinto tipo de medios (textos, gráficos, videos, etc.). El material compartido por el grupo está interrelacionado, es generado por distintas aplicaciones y posiblemente almacenado sobre distintas plataformas.

Los investigadores realizan una lectura activa sobre el material, es decir, crean notas y relaciones a medida que leen. Estas relaciones constituyen en sí conocimiento agregado (metadatos) acerca de la interpretación del material en estudio. Estos metadatos resultan sumamente útiles para el resto del grupo de investigadores y constituyen una herramienta necesaria para la consulta efectiva de dicho material. Por lo que resulta imperioso y factible que los investigadores puedan crear notas y relaciones sobre el material, y además puedan compartirlos.

La Hipermedia desde sus inicios ha tenido como meta el manejo del conocimiento, basta con ver los escritos de pioneros como Vannaver Bush para constatarlo. La aparición de la Web, si bien ha permitido popularizar la Hipermedia de manera insospechada y ha provocado el mayor hito en la comunicación de la era pos Gutenberg, no ofrece la riqueza funcional requerida

para la representación y comunicación de conocimiento, que por otro lado si estaba presente en el ámbito de la comunidad Hipermedia.

En respuesta a las falencias de funcionalidad Hipermedia de la Web para el manejo de conocimiento, la comunidad de Hipermedia, principalmente a través del grupo de trabajo de Sistemas de Hipermedia Abierta (OHSWG), ha investigado y desarrollado sistemas que permiten complementar la Web y las aplicaciones manejadoras de contenido, agregando a estas la funcionalidad carente.

7.1 Conclusiones

Este trabajo ha sido fruto del análisis de los trabajos de investigación y desarrollo realizados por la comunidad de Hipermedia, y propone el diseño detallado de un Sistema que provee soporte a un grupo de investigadores durante la etapa de investigación conceptual. Esta propuesta, además de estar soportada con conocimiento acumulado en el área, muestra cómo se abordan cuestiones críticas de diseño. Cabe destacar que se abordó la solución en forma rigurosa, para lo cual se siguieron los principios, protocolos y normas propuestos por la comunidad Hipermedia, en la búsqueda de la solución.

Se describen a continuación las principales contribuciones de este trabajo, las que están directamente relacionadas con los objetivos planteados en el capítulo 1 (sección 1.5):

- **Selección y justificación de la metodología de desarrollo:** se propone un desarrollo experimental y cooperativo.
    - Experimental: de tipo prototipado, es decir, que consiga implementaciones parciales tempranas. Se propone el prototipado dado lo inadecuados que han resultado los procesos tradicionales orientados hacia la especificación, principalmente en sistemas de la envergadura como el planteado, para los cuales generalmente prevalecen las metodologías ágiles.
    - Cooperativo: en el sentido de involucrar fuertemente en el desarrollo a los usuarios potenciales. Se considera que el éxito de un sistema depende del grado en el cual su desarrollo se alimente de los intereses y de la participación activa de los usuarios. Hay que considerar que en definitiva, los sistemas deben ser útiles para las personas cuyo trabajo no tiene nada que ver con Hipermedia, ingeniería, *frameworks*, etc.
- **Descripción de la operación básica del Sistema:** en la cual se especifica cómo el sistema propuesto amplía las aplicaciones favoritas de los investigadores con un servicio de Hipermedia Abierta, lo que permite la creación dinámica de estructuras y comentarios, y la colaboración necesaria para compartir esos útiles metadatos. Además se describe cómo se recuperan y muestran los metadatos almacenados en una base de datos separada de los documentos. Se realiza también una descripción textual y gráfica de la interfaz del usuario con el sistema, en la cual se presenta la

ventana principal del Cliente de Hipermedia Abierta y se detalla la funcionalidad básica del servicio y del cliente.

- **Descripción del dominio del problema:** en la que se incluye el modelo del dominio y se describen detalladamente en forma textual los distintos conceptos y las relaciones entre los mismos. Esta descripción complementa el modelo y brinda mayor especificidad en pos de un manejo común de vocabulario y una correcta interpretación del dominio del problema.
- **Especificación de los requerimientos funcionales y no funcionales:** los requerimientos funcionales son obtenidos a partir de las especificaciones de necesidades de los investigadores (individuales y grupales) planteadas en el capítulo 2. Los requerimientos no funcionales incluyen cuestiones que hacen a la rigurosidad en el abordaje de la solución, como lo son la adhesión a los principios de Hipermedia Abierta, la utilización de estándares y cuestiones claves de usabilidad como por ejemplo, que las nuevas herramientas se utilizan si se basan sobre metáforas conocidas y si el beneficio de utilizarlas es mayor que el esfuerzo requerido.
- **Descripción de los casos de uso más significativos:** en la que se seleccionan los casos de uso más relevantes, y se describen en forma detallada. Dado que se propone un desarrollo de tipo experimental, se describen solamente los casos de uso más significativos, dado que éstos sientan las bases para la correcta interpretación del resto de los casos de uso y que usuarios potenciales estarán involucrados durante el desarrollo.
- **Especificación y justificación de la Arquitectura:** se especifica una arquitectura de tres capas físicas basadas en las cuatro capas conceptuales propuestas por la comunidad de Hipermedia Abierta. Se describen además las funcionalidades de los procesos involucrados en cada capa, y se presenta un gráfico que muestra la relación entre las capas físicas y conceptuales. La especificación incluye la propuesta de los protocolos de comunicación a utilizar entre las capas, por ejemplo, se propone y justifica la utilización de OHIF, formato estándar de intercambio de Hipermedia Abierta, para la comunicación entre los Clientes Hipermedia y el Servicio.
- **Diseño detallado del Servicio de Hipermedia Abierta:** el cual consta de las clases de diseño para las capas de almacenamiento y de tiempo de ejecución. Este diseño cubre los casos de uso descriptos y satisface los requerimientos funcionales planteados. Cada clase base de diseño surge de mapear la conceptualización propuesta por la comunidad de investigación y desarrollo en Hipermedia Abierta. Para las representaciones gráficas se utiliza *Rational Rose*, herramienta que permite sentar las bases para la implementación futura del prototipo. Además de la representación gráfica de cada clase, se incluye una descripción textual de cada una que complementa los gráficos e incluye cuestiones específicas y/o de implementación que no surgen directamente de la representación UML.
- **Descripción detallada de la travesía:** se elige la descripción de esta operación, dado que es fundamental de cualquier Sistema Hipermedia. Se incluyen las clases de diseño involucradas y el algoritmo, al igual que el resto del diseño, está basado en las investigaciones más relevantes en el área. Cabe destacar además que soporta comportamientos avanzados de vinculación como lo son la apertura de compuestos y colecciones, links

globales y links multicabeza. En el final de la descripción se proponen también, abordajes de diseño e implementación para cuestiones críticas de las travesías, como:

- Utilización de tablas Hash para las localizaciones: una propuesta de implementación para las localizaciones de modo que las búsquedas, realizadas en cada operación de travesía, resulten más eficientes.
- Consideraciones para implementar la coincidencia entre una localización buscada y las localizaciones almacenadas para un contexto.
- Propuesta de diseño para soportar la retórica de arribo en la travesía de un link.

La propuesta presentada en este trabajo, además de alcanzar los objetivos planteados, sienta las bases para brindar soluciones a grupos de trabajo similares al presentado en el capítulo 2, ya que brinda una metodología de desarrollo y un marco conceptual que permiten extender el diseño presentado, marcando y facilitando el camino para el abordaje de soporte a ambientes orientados al manejo de conocimiento como, por ejemplo, lo son:

- La revisión y comparación de sitios: algunos diarios basados en la Web proveen revisiones y comparaciones de sitios, para lo cual un periodista revisor puede utilizar el Servicio Hipermedia para crear links y caminos (con comentarios) entre detalles de los sitios bajo revisión. Luego los lectores pueden tener acceso a estos metadatos agregados, a través de un Servicio Hipermedia restringido a solo lectura.
- Las consultoras basadas en la Web: que ayudan a los usuarios en el entendimiento de cuestiones como por ejemplo, leyes y directivas, publicadas vía Web por el gobierno. A través de la creación de links, por parte de los consultores, desde las directivas y leyes a textos explicativos y/o sitios alternativos que ayuden al usuario. Además varios consultores podrían trabajar cooperativamente en la producción de la información de consulta.

Además de estas contribuciones principales que eran el objetivo de este trabajo, hay un conjunto de logros secundarios que han tenido lugar a lo largo del mismo, que se detallan a continuación:

- Determinación de la relación existente entre el manejo de conocimiento, la Hipermedia (desde sus pioneros) y las actividades llevadas a cabo en un grupo de investigación durante la etapa de investigación conceptual. A partir de esta determinación se propuso abordar el soporte para el ambiente de investigadores utilizando la tecnología Hipermedia.
- Desarrollo de un resumen de la evolución histórica de la Hipermedia (capítulo 3), analizando la misma desde distintos puntos de vista, como el conceptual, el tecnológico y desde el diseño y desarrollo de sistemas. La evolución presentada incluye además una descripción gráfica y textual,

acerca de cómo las distintas contribuciones e investigadores, influenciaron a sus sucesores.
- Dominio de un área de conocimiento, la Hipermedia Abierta.

Uno de los logros secundarios más valiosos que ha dejado este trabajo, y que requiere un par de párrafos especiales, es el reconocimiento de la importancia de conceptualizar para poder plantear nuevas y mejores alternativas. La Web, el Sistema Hipermedia que domina el mundo, vista desde el enfoque conceptual brindado por la comunidad Hipermedia, muestra sus debilidades.

La conceptualización permite plantear y proponer mejoras que sean independientes de la tecnología y del estado actual. Basta con observar la diferencia entre la definición de link presentada en la W3C y la presentada por la comunidad de Hipermedia Abierta, para notar la diferencia entre una definición "atada" a la tecnología disponible, y una definición independiente, que amplía el horizonte.

7.2 Trabajos Futuros y Cuestiones Pendientes

Se ha arribado a un estado en el que resulta imperiosa la implementación de un prototipo, a fin de refinar y concretar el diseño con cuestiones que aparecen durante la implementación, por lo que éste es considerado como el trabajo futuro inmediato. El prototipo debe implementarse conforme al diseño obtenido y utilizando la metodología de desarrollo propuesta.

Será necesario discutir los conceptos y elementos claves que el primer prototipo deba considerar, y restringir el alcance del mismo, definiendo características que serán implementadas en cada versión. Por lo que la planificación del desarrollo del prototipo deberá contemplar:

- El entendimiento de la tecnología necesaria para el desarrollo del mismo.
- La definición y el orden cronológico de las características que serán desarrolladas. En principio se deben fijar prioridades y solamente implementar las características más esenciales.
- La definición de las restricciones del alcance del prototipo, es decir, se necesitará una lista de cosas que el prototipo aun no hará.
- La planificación de desarrollo, consistente de un número de puntos de chequeo. Estas metas parciales definirán qué será capaz de hacer el prototipo en cada momento e Indicarán el orden en el cual las características deberán ser implementadas.
- La comprensión de que la planificación no será fija, debido a dificultades tecnológicas y/o nuevas ideas que surjan durante el desarrollo, el plan debe permitir cambios en caso de que resulten necesarios.

En cuanto a cuestiones que han quedado pendientes y que habrá que abordar en el desarrollo de futuros prototipos están:

- Los cambios en el material, por fuera del Sistema Hipermedia: cuando un documento se borra, renombra o mueve por fuera del alcance del Sistema Hipermedia, el seguimiento de un link hacia ese documento genera una excepción. La operación de travesía debería considerarlo, capturando la excepción y pasándosela al investigador junto con atributos almacenados en el componente atómico, de modo de informar al usuario acerca del tipo de error.
- El manejo de versiones, para documentos y estructuras: se podría considerar el manejo de versiones para las estructuras, sin embargo esto disparará inconvenientes en cuanto a las versiones sobre los documentos, por ejemplo: ¿Dónde debe apuntar un link, a la versión original del documento que apuntaba, o la última versión? ¿Qué sucede si la ultima versión no contiene el punto extremo del link en cuestión?

Apéndice A

Reseña de Sistemas Hipermedia

En esta reseña, se presentan (por orden alfabético) las perspectivas generales de los trabajos de Hipermedia considerados como los más relevantes. Los sistemas, diseños y modelos elegidos varían en cuanto a su punto de vista y van: desde prototipos de investigación para un grupo de usuarios, hasta productos populares para el mercado general, desde soporte de infraestructuras de bajo nivel hasta integraciones a nivel de aplicación y desde programas centrados en estaciones de trabajo hasta programas para arquitecturas cliente-servidor.

Se busca que este apéndice exprese la amplitud de trabajos de diseño y desarrollo que esté relacionado implícita o explícitamente con la Hipermedia Abierta. Vale destacar que la importancia de varios de estos sistemas se extiende más allá del campo de la Hipermedia Abierta.

AHM: Modelo de Hipermedia de Ámsterdam
Conocido como AHM, extiende el modelo de Dexter para soportar estructuras multimedia basadas en el tiempo (Hardman y otros, 1994). En particular, AHM generaliza el compuesto de Dexter para soportar sincronizaciones de granulado grueso. Además este modelo ha sido utilizado para implementar la herramienta CMIFed (Rossum y otros, 1993), cuya utilidad es el diseño de presentaciones multimedia interactivas.

Aquanet
En el diseño de Aquanet (Marshall y otros, 1992), se abordó un problema que Halasz identificó en su escrito "Seven Issues" de 1998. El problema era, el inadecuado soporte para estructuras que tenían la mayoría de los sistemas Hipermedia. En el sistema Aquanet los links son virtualmente inexistentes, en su lugar, los componentes están Interconectadas a través de relaciones estructuradas que tienen tipo y son configurables. Un aspecto crucial de cada relación es su representación gráfica en la interfaz del usuario, para lo cual utiliza el color y una distribución de dos dimensiones para expresar los roles que juegan los componentes en estructuras mayores. Debido a que la naturaleza formal de las estructuras creaba problemas para algunos usuarios, Marshall abordo más tarde este inconveniente en el sistema VIKI, que soportaba estructuras informales emergentes.

Augment (NLS)
Douglas Engelbart se anticipó dos décadas a la mayoría de los investigadores de Hipermedia y de CSCW (*Computer Supported Cooperative Work*). Comenzando en 1960, con los sistemas NLS y Augment, los cuales eran plataformas de una serie de herramientas innovadoras para el estructurado y la navegación de información. Las contribuciones de Engelbart incluyen el *mouse* y el *chordset*, la video conferencia, la interfaz de usuario con ventanas y la

edición de estructuras fuera de línea. Aunque las aplicaciones dentro de Augment lo hacían un sistema esencialmente monolítico, varias de sus innovaciones hay influenciado en gran medida el trabajo dentro de la Hipermedia Abierta. A través de ARPAnet, Augment ofrecía el primer medio para autoría cooperativa en red (Engelbart, 1984). Además las especificaciones y comportamiento de travesía de los links de Augment, inspiraron la especificación de presentación del modelo de Dexter. En www.bootstrap.org/librar.htm, se encuentra un resumen de las contribuciones de Engelbart.

Chimera

Chimera (Anderson y otros, 1994), desarrollado en la Universidad de California, es un sistema de Hipermedia Abierta que integra aplicaciones de terceros. Con una aproximación similar a la de *Sun Link Service*, provee librerías y APIs para utilizar a nivel de código fuente. Chimera ha sido integrada con varias aplicaciones, incluyendo Framemaker y navegadores Web.

Desde sus comienzos, su principal meta eran los grandes proyectos de desarrollo de software, lo que involucraba entre otras cosas, el trabajo sobre escalabilidad. La arquitectura de Chimera es de tres capas con una base de datos relacional para el almacenamiento. Una característica interesante es la posibilidad de atender *applets* Java de los usuarios, de modo que éstos puedan acceder y manejar el sistema sin instalar software especial.

Una característica única de este sistema es el concepto de vista. Los sistemas de Hipermedia Abierta en general, manejan editores o visores para mostrar un documento y utilizan un visor para cada tipo de medio. Las anclas en estos casos están asociadas al documento y su presentación depende de las posibilidades del visor. Chimera aborda esto en forma diferente, asociando las anclas a vistas, es decir, a la combinación de un visor y un documento. La implicancia es que puede presentar el mismo documento en forma diferente (con diferentes anclas), dependiendo del visor utilizado. Un ejemplo podría ser una planilla de cálculo presentada como una matriz o como un grafo.

La interfaz para el manejo de estructuras Hipermedia es responsabilidad del cliente o visor (dependiendo del nivel de integración). Chimera no ejerce control de documentos y puede, como la mayoría de los sistemas Hipermedia, tener problemas con la consistencia de los links. Como las anclas son específicas de la vista, su apariencia y ubicación depende de las capacidades del visor.

Construct

Es uno de los resultados del proyecto Coconut. Es el código base descendiente de HOSS, HyperDisco y DHM, diseñado y desarrollado por personas que trabajaron en estos sistemas. Cumple con estándares del OHSWG, a través de varias extensiones, tales como Hipermedia espacial y de composición. Es también el *backend* de Arakne Environment.

Tiene un diseño de tres capas con un conjunto de servicios extensible. La arquitectura es simétrica, respecto a las tres capas. Los clientes se comunican con el servidor de estructura a través de un servidor virtual, los servidores se comunican con los clientes a través de clientes virtuales y así sucesivamente. El núcleo mantiene la funcionalidad actual, por lo tanto, un cliente de Hipermedia navegacional utiliza el núcleo *Nav* para operar sobre componentes navegacionales.

La arquitectura con clientes y servidores virtuales, provee multiplexado y deja espacio para otras capas de transporte. Construct utiliza XML sobre *sockets* TCP/IP. Soporta Hipermedia navegacional, especial y de composición. Dado a su naturaleza extensiva, este conjunto puede crecer.

Soporta cooperación a nivel de servidor, principalmente a través de la noción de "sesión". Para manejar la consistencia de los links, el modelo navegacional del OHSWG, tiene un conjunto de especificadores de localización flexible y extensible. Construct no soporta manejo de versiones.

Device Hypermedia

DHM (Goenbaek y Malhorra, 1994) es un sistema de Hipermedia Abierta que se ejecuta sobre los sistemas operativos UNIX, Macintosh y Windows, inspirado en el modelo de Referencia de Dexter. Este sistema es un framework orientado a objeto que extiende el modelo de referencia de Hipertexto de Dexter. Utiliza un servidor de base de datos orientado a objetos para almacenar las estructuras Hipermedia e integra varias aplicaciones técnicas y de oficina entre si, y con la Web

Ha sido utilizado como un prototipo de investigación, y es un sistema Hipermedia entre plataformas, cooperativo, adaptable a través de un interprete incluido, y en su actual versión, *Webvise* basado en Microsoft Windows, integra Microsoft Office y el Internet Explorer.

A nivel estructura soporta links n-arios, bidireccionales y tour guiados. Pone énfasis en la interfaz del usuario, de lo cual resultan interfaces integradas a aplicaciones Microsoft en las cuales la funcionalidad Hipermedia está disponible a través de menús, y las anclas se resaltan sobre los distintos documentos.

DHM tiene especificadotes de anclas robustas. Si bien los documentos están fuera del control del sistema lo cual lleva a la posibilidad de inconsistencia de

links, esta especificación permite en numerosas ocasiones solucionar este inconveniente. DHM no provee soporte de versiones y si bien DHM es un sistema cooperativo y distribuido, su actual presentación Webvise, no lo es.

Dexter: Modelo de referencia de Hipertexto

El modelo de Referencia de Hipertexto de Dexter (Halasz y Schwartz 1994) no esta asociado con una implementación particular de sistema; modela un rango de sistemas existentes desde el punto de vista arquitectural y en términos de su comportamiento de tiempo de ejecución. Es conocido por la división de responsabilidades de un sistema Hipermedia entre las capas de almacenamiento, tiempo de ejecución y componente, y por los conceptos de ancla y de especificación de presentación. En el capítulo 4 se proveyó una detallada introducción al modelo.

HAM

La máquina abstracta de Hipertexto (Campbell y Goodman, 1988) fue el primer modelo de dato abstracto para Hipertexto. Antes de la publicación del modelo de Dexter, HAM era utilizado para caracterizar, comparar y contrastar los modelos de dato de KMS, NoteCards e Intermedia. Junto con Intermedia y Xanadu, inspiro la separación en capas de ejecución y almacenamiento del modelo de Dexter. HAM fue también utilizada para construir un sistema de ingeniería de software, Neptune, que por primera vez abordaba el problema de manejo de versiones en Hipermedia (Delisle y Schwartz, 1986).

HOSS (HB1-HB4)

HOSS (*Hypermedia Operating System Services*, Nurnberg y otros, 1996) representa el último desarrollo de las series HBx/ SPx (Leggett y Schnase, 1994) de sistemas Hipermedia de la Universidad A&M de Texas. HOSS separa la funcionalidad de los sistemas Hipermedia en primitivas básicas a ser provistas a nivel de sistema operativo. Las primitivas están divididas en datos, estructuras y comportamientos. Los datos son contenidos de información de aplicaciones Hipermedia; las estructuras reflejan la rica variedad de relaciones entre los datos; y los comportamientos son cálculos acoplados a la estructura. HOSS provee una arquitectura de capas sobre el tope de un sistema operativo existente, *Sun Solaris*.

HOSS fue un paso más allá de una serie de hiperbases y sistemas Hipermedia desarrollados en la Universidad de Texas A&M: PROXHY, HB1/SP1, HB2/SP2 y HB3/SP3. Una característica relevante de HOSS es su enfoque de proveer funcionalidad Hipermedia a nivel de sistema operativo, principio que manejaba la comunidad de "*Structural Computing*". Generalmente los sistemas Hipermedia eran considerados como pertenecientes al nivel de aplicación, dando al usuario herramientas para interactuar y estructurar información.

Otra innovación de HOSS es la separación entre datos, estructura y comportamiento. La separación de datos y estructura era conocida en el ambiente Hipermedia, pero este sistema fue un paso mas allá permitiendo que

el comportamiento sea sutilmente separado. Separando el comportamiento y relegándolo junto con la estructura a nivel de sistema operativo, se abren posibilidades sustanciales. Un sistema operativo Hipermedia podría ser muy flexible y podría proveer a sus usuarios de una performance superior en el trabajo con estructuras, realizando una carga previa de los datos y las estructuras, manejando automáticamente estructuras, etc. Un prototipo de este sistema operativo ha sido implementado en HOSS, este sistema corre bajo *SunOS* y provee:

- Servicios para manejar datos, estructura y comportamiento.
- Un conjunto abierto de comportamientos (*Sprocs*),
- Hipermedia navegacional, taxonómica y anotaciones.
- Herramientas para animación, revisión y edición de texto espacial.
- Invocación para herramientas no integradas.
- Almacenamiento de objetos para herramientas integradas. La capa de almacenamiento de HOSS ofrece control de versión y está disponible para estructuras y datos.
- Manejo de distribución.
- Consistencia de links, para los documentos guardados en su almacenamiento de objetos.

Además de la idea de integrar Hipermedia a nivel sistema operativo, HOSS provee su propia herramienta de desarrollo. Los diseñadores de HOSS destacaban la importancia de un conjunto de servicios abiertos y con este fin proveyeron al desarrollador, el PDC (*Protocol Definition Compiler*), que genera librerías C para IPC (*interprocess communication*) entre las componentes de HOSS. Esto acelera el tiempo de desarrollo, ya que los programadores se liberan del trabajo tedioso de desarrollar ese código.

HyperCard

Originalmente un ambiente de programación gráfico que permitía conectar fragmentos de información. Una pila formada por una colección de tarjetas, y botones que vinculan cualquier tarjeta de la pila. Fue el primer Sistema Hipermedia ampliamente utilizado en las PCs (Goodman, 1987). Hasta la aparición de la Web, HyperCard tuvo probablemente la comunidad de usuarios mas grande de sistemas Hipermedia. De hecho, debido a sus modelos similares de vinculación y navegación, el primer navegador Web se parece a HyperCard con el agregado de texto etiquetado. HyperCard tiene poco soporte para integración de aplicaciones o colaboración, es por esto que no ha formado parte en las discusiones de Hipermedia Abierta presentadas en este trabajo. Sin embargo, su lenguaje de *script*, HyperTalk (para asociar eventos con acciones), confirma, al menos para la comunidad de Macintosh, una potencia y flexibilidad de Hipermedia que es profundamente adaptable.

HyperForm e HyperDisco.

HyperForm (Wiil y Leggett, 1992) e HyperDisco (Wiil y Leggett, 1997) son plataformas de Hipermedia Abierta que generalizan hiperbases previas, tales como HAM, que estaban diseñadas para soportar un modelo de dato

Hipermedia fijo y un conjunto específico de servicios de bases de datos. En contraste, HyperForm e HyperDisco pueden manejar el almacenamiento de una variedad de modelos de dato Hipermedia. Además, las plataformas permiten a los diseñadores de Hipermedia, crear nuevas combinaciones de herramientas de bases de datos y colaboración, para soportar control de acceso, manejo de versiones y notificación de eventos. También proveen servicios de Hipermedia Abierta a través de herramientas denominadas integradoras. Provee además soporte para distribución de estructuras Hipermedia.

HyperDisco viene de una familia de sistemas Hipermedia desarrollados en la Universidad de Aalborg: Hyperbase 0–2, Hyperform, el Emacs HyperText System (EHTS) e HyperDisco. Una característica de estos sistemas es la orientación arquitectural hacia la distribución, colaboración y los sistemas de base de datos Hipermedia. Pocos sistemas, salvo HOSS, se enfocan tanto sobre la infraestructura Hipermedia. De hecho HyperDisco puede ser considerado un sistema de meta Hipermedia, tal cual lo expresan sus autores:

> *"The HyperDisco project is about design, development, deployment and assessment of OHSs. The overall goals of the project are to work towards innovative new OHSs and to try to influence the development of future generations of OHSs for the Internet"* (Wiil y Leggett, 1992).

El sistema Hipermedia HyperDisco tiene varias propiedades interesantes. Es esencialmente una arquitectura de tres capas con herramientas (integradas o nativas), integradores de herramientas (en general uno por usuario) y espacios de trabajo. Un espacio de trabajo incluye una hyperbase, funcionalidad cooperativa y puede estar distribuido sobre Internet. Los usuarios pueden elegir con qué espacio de trabajo operar, asumiendo que tienen los permisos apropiados. En cuanto a las cuestiones identificadas por Pearl en *Sun Link Service*, Hyperdisco aborda varias de estas cuestiones.

- Integra diez aplicaciones muy conocidas, entre las cuales está XEmacs, http://www.xemacs.org/ con menú y marcado de anclas.
- Maneja documentos almacenados fuera del espacio de trabajo, y para los que están dentro del espacio de trabajo ejerce control de documentos.
- Responde en gran medida a la inconsistencia de links. Por cierto si el documento se mantiene bajo el control de HyperDisco, la consistencia de links está asegurada. Como las anclas son extensibles, la posibilidad de re generar anclas depende de los clientes.
- Maneja control de versiones a nivel de modelo de dato proveyendo una clase "*Versión Control*" que a través de la herencia múltiple puede ser sub clasificada para cualquier componente. Esto es utilizado principalmente para control de versión de nodos, no así para estructuras.
- Soporta distribución y cooperación. Los espacios de trabajo pueden estar distribuidos en Internet y los usuarios del sistema pueden cooperar libremente dentro de esos espacios.

Hytime
Hytime es un estándar para representación e intercambio de documentos estructurados (DeRose y Durand, 1994). Construido en SGML, agrega soporte para vinculación y sincronización de documentos multimedia. Aunque no está orientado al desarrollo de sistemas, ha influenciado en la comunidad de de Hipermedia Abierta. Por ejemplo, su formato estructurado y personalizado para localizar dentro y entre documentos, incluye una de las definiciones más comprensivas de *locSpec*.

Intermedia
Junto con KMS y NoteCards, fue uno de los sistemas Hipermedia más influyentes de los 80s, con usuarios dedicados y un diverso conjunto de investigadores (Yankelovich entre otros, 1988). Intermedia fue el primer sistema en soportar links bidireccionales, basados en anclas y cargados en webs separadas de los contenidos de los documentos. El grupo de Intermedia condujo una investigación de amplio rango e innovadoras en áreas tales como multimedia (Catlin y Smith, 1988), colaboración (Catlin y otros, 1989), navegación gráfica (Utting y Yankelovich, 1989), y aplicaciones educacionales de Hipermedia (Ess, 1991). Otras características sobresalientes son la arquitectura orientada a objeto de Intermedia (MeyroWitz, 1986) y la utilización de bases de datos genéricas para las estructuras (Haan y otros, 1992).

KMS (ZOG)
KMS es la mayor y mas ilustre historia de los sistemas Hipermedia aún en uso. Comenzó con ZOG en la Universidad de Carnegie Mellon en 1960. Luego de moverse fuera de la universidad hacia el mundo comercial en 1980, se llamó KMS y tiene uso activo aun hoy (Akscyn y otros, 1988). Corre bajo plataformas UNIX, tiene una población de usuarios dedicados que aprecian su interfaz de usuario, su elegante travesía de links y su editor de componentes tipo *canvas*. KMS soporta colaboración y manejo de versiones, esta última una rara característica en sistemas Hipermedia actuales. Como la Web, KMS tiene un tipo de componente simple el *frame*, pero a diferencia de la web, KMS puede representar fácilmente interconexiones jerárquicas, así como también vínculos.

Memex
La mayoría de los investigadores Hipermedia coincide que el Memex de Bush (1945) es el progenitor de la noción moderna de Hipertexto como información Interrelacionada navegable. Por ejemplo, Engebert y Nelson reconocen al Memex como la primera mayor influencia. Curiosamente, para Bush el camino (*trail*) más que el link fue la entidad estructurante fundamental (Trigg, 1991). Más allá de los mecanismos para la creación y navegación de caminos, Bush tenía una visión de escolares cooperando. En particular, la idea de que un escolar podía pasar el resultado de una investigación en la forma de un camino a otro estudiante. El camino lleva consigo copias del contenido que estructuraba, y es por lo tanto un ejemplo perfecto del copiado y movido de subhiperespacios.

Microcosm

El grupo de investigadores y desarrolladores de la universidad de Southampton y Multicosm Ltd., han articulado e implementado la visión de Hipermedia Abierta tan bien como nadie (Hall y otros, 1996). Su principal contribución desde la perspectiva de Hipermedia Abierta, es un conjunto de herramientas para interconectar aplicaciones de terceros, incluyendo aquellas que son poco adaptables. Microcosm soporta la integración de aplicaciones cerradas a través del denominado "visor universal", que remienda la interfaz de usuario de una aplicación arbitraria con funcionalidad, para capturar selecciones e invocar servicios de link. El servicio de link distribuido (*Distributed Link Service*, Carret y otros, 1995) es una extensión de Microcosm que traslada muchas de sus ideas al contexto de la Web. Como intermedia, los investigadores de Microcosm, han también abordado problemas de integración multimedia (Lewis y otros, 1996)

Es un sistema que aun estaba en desarrollo y uso en el 2000. El grupo de la universidad de Southampton ha sido exitoso en la integración de muchas aplicaciones de terceros. Una de las innovaciones de este sistema fueron los links genéricos (links con un punto extremo destino con ubicación específica y con un origen que sólo especifica el contenido de un punto extremo origen, en lugar de su localización).

Uno de los problemas con la vinculación Hipermedia es que el autor del link tiene que crear todos los links, de modo que si se agregan nuevos documentos, éstos están inicialmente, desvinculados. Al crear links basados, por ejemplo en palabras clave, hace posible la reutilización de links existentes en nuevos documentos.

Microcosm abogaba por una inclusión agresiva de editores (*viewers*) de terceros, incluso de aquellos que no tuvieran "conciencia" de Microcosm. Para soportar este tipo de aplicaciones, alguna funcionalidad tal como resaltado automático de anclas eran evitadas. En lugar de esto, las anclas en un documento podían ser halladas por demanda o mostradas en ventanas separadas. El diseño de Microcosm tiene algunas interesantes consecuencias.

- Interfaz sensata: los links genéricos implican que generalmente no hay marcado de links. Ciertamente, los creadores de Microcosm citan esto como una fortaleza del sistema. Por otro lado al no haber marcado, es mas fácil integrar editores y con ello expandir el numero de aplicaciones a las que se puede agregar funcionalidad. Microcosm agrega un botón Hipermedia a todas las aplicaciones, desde donde el usuario accede a su funcionalidad.
- Control de documentos: puede ser manejado por un manejador de documentos, que aborda algunas de las cuestiones con respecto del control de documentos. Sin embargo si los documentos pueden ser recuperados por fuera del sistema, la posibilidad de documentos corrompidos permanece. Pero este problema es menor en Microcosm, dado el enfoque de los links genéricos. Un link genérico trabajara en todos los documentos, mientras haya una selección que concuerde con el link, el documento destino puede no estar.

- Localización de punto extremo: en Microcosm está circunscripta, ya que muchos puntos extremos, especialmente a editores sin conciencia Microcosm, son componentes completas. Para anclas específicas, utiliza desplazamiento de *bytes* para localizarlas. Tal forma de abordar es frágil, y puede ser suplementada, por ejemplo con macros para localizar la selección.
- Con Microcosm TNG (*The Next Generation*), Microcosm puede manejar documentos, links y localización de servicio distribuido. TNG incorpora además la noción de sesión, con las que los usuarios pueden compartir y colaborar sobre aplicaciones y servicios seleccionados, esto es, nodos y colecciones de links.
- Microcosm permite que el usuario acceda libremente a nodos y links de diferentes versiones, aunque no soporta mezcla (entre distintas versiones para links y nodos)

Actualmente este proyecto se encuentra comercialmente bajo el nombre de *Active Navigation* (www.activenav.com).

MultiCard

MultiCard (Risk y Sauter, 1992), desarrollado en INRIA (Francia) es un sistema de Hipermedia Abierta cuyo conjunto de herramientas Hipermedia permite que los programadores creen y manipulen estructuras Hipermedia distribuidas. Las aplicaciones a ser integradas deben soportar M2000, un rico protocolo basado en librerías para integración en tiempo de desarrollo. También incluye un avanzado lenguaje para escribir *scripts*, que puede ser asignado a nodos, grupos (compuestos) y anclas. Finalmente, MultiCard provee un conjunto de herramientas de autoría de propósito general para Hipermedia.

NoteCards

Fue uno de los Sistemas Hipermedia de investigación más conocidos de los 80´ (Halasz y otros, 1987). Basado en una metáfora de tarjetas de notas y cajas de archivos, intentaba cubrir las necesidades de trabajadores de información, desde análisis de inteligencia a autores de disertaciones doctorales. El tamaño de la comunidad de usuarios de NoteCard estaba limitado por su plataforma de tecnología, el ambiente de Xerox InterLisp. Sin embargo, para aquellos que accedían a esa plataforma, les proveía una forma de integrar texto estándar y editores gráficos en una forma extensible y adaptable (Trigg y otros, 1987). Varias de las herramientas de Hipermedia desarrolladas por NoteCards influenciaron en la comunidad Hipermedia, incluyendo el navegador de red gráfico, utilizado para visualizar y manejar redes de cientos de tarjetas y links (Irish y Trigg, 1987); los tours guiados, las tarjetas de TableTop (Trigg, 1988); y la historia de tarjetas, utilizada para monitorear trabajo cooperativo asincrónico entre múltiples autores (Irish y Trigg, 1989).

Desde la perspectiva de los diseñadores de Hipermedia Abierta, la contribución más importante de NoteCard es su mecanismo extensible de tipo de tarjeta, un precursor para los tipos de componente extensibles. También relevante para

este trabajo, pero menos precursor, son sus links entre *notefiles* y las operaciones para demarcar, copiar y mover sub redes de *notefiles*.

Para finalizar, una mención especial para el escrito brillante de Frank Halasz "*Seven Issues*" (1988), quizás el más citado articulo sobre Hipermedia, el cual ha trazado las mayores direcciones de las investigaciones de los 90, sobre la base de un cuidadoso análisis de las debilidades y fortalezas de NoteCards. En un campo que cambia tan rápido como éste, es un testamento, dado que más de 10 años después, sigue siendo una de las lecturas populares más requerida.

OLE/ DDE
Las librerías y APIs de OLE y su predecesor DDE, soportan integración entre aplicaciones sobre la plataforma Windows (Brockschmidt, 1995). El intercambio de datos entre programas y la invocación de comandos, posibilitan que los usuarios construyan links calientes o tibios, los cuales sirven como canales activos o pasivos para transferencia de datos entre programas, por ejemplo, desde una celda seleccionada en una planilla de calculo, a una tabla en un documento Word. OLE también soporta HyperLinks, que son links Hipermedia desde un contenedor OLE a otro. Como en HTML, estos links están embebidos en los datos de su contenedor de origen.

SEPIA
SEPIA (Streitz y otros, 1992) es un ambiente de autoría cooperativa de Hipermedia. Provee almacenamiento de datos persistente y compartido, un modelo de dato de Hipermedia con compuestos, funcionalidad de autoría comprensiva y soporte para trabajo cooperativo. SEPIA suscribe al modelo para autoría basado en espacios de actividades, que provee vistas de actividades dependientes del hiperdocumento que está siendo creado. SEPIA soporta cooperación sincrónica y asincrónica entre múltiples autores, en la forma de accesos compartidos a hiperdocumentos, y conferencia integrada de audio y video. Las herramientas de autoría son editores *buit-in* y navegadores gráficos; pero no soporta la integración de aplicaciones de terceros.

Sun Link Service
Aunque nunca desarrollo una gran comunidad de usuarios (Pearl, 1989), es generalmente considerada como el primer servicio de link basado en red para aplicaciones de terceros (el primer Sistema de Hipermedia Abierta). Su especificación de anclas y protocolos para comunicación entre las aplicaciones de servicio de link, inspiraron la separación de responsabilidades modeladas por el grupo Dexter de la capa de componente. Requiere que los desarrolladores de aplicaciones integren una pequeña librería Hipermedia estándar con sus aplicaciones, en tiempo de desarrollo. Esta librería provee, entre otras cosas, comunicaciones con el servicio de link y marcado de link estandarizado.

El servicio de link era una parte integrada del sistema operativo SunOS y del sistema de ventanas OpenLook, y estaba disponible para todas las

aplicaciones que corrieran sobre esta plataforma. Proveía a los usuarios de colecciones privadas o compartidas de links, y soportaba "*garbage collection*" de links automática o explicita.

Sun Link Service no tuvo la popularidad que debiera, porque el éxito del sistema dependía de que los desarrolladores activamente modificaran sus aplicaciones, sin embargo Pearl identificó varias cuestiones importantes para el desarrollo de sistemas de Hipermedia Abierta.

- Interfaz de usuario: por definición un sistema de Hipermedia Abierta depende de editores de terceros para manejar documentos. Hay dos aspectos de interfaz en esto: primero, debe haber una interfaz para interactuar con el servicio de link; segundo, debe haber algún "resaltado" para identificar los objetos vinculados.

- Control de documento: otro tema común en Hipermedia Abierta es la carencia fundamental de control de documentos y su paradero. A menos que el documento resida en un sistema manejador de documentos, éstos pueden ser modificados, borrados o movidos sin que el sistema Hipermedia se entere.

- Consistencia de link: esto está relacionado con el control de documentos. Los links pueden volverse inconsistentes si uno de sus documentos constituyentes desaparece, es borrado, movido, modificado o no esta disponible.

- Ubicación de puntos extremos: aparte del problema de que el documento desaparezca, localizar un punto extremo en un documento de un tipo no diseñado para tal cuestión, puede ser un problema en sí, especialmente si el documento fue modificado.

- Distribución: los sistemas Hipermedia monolíticos generalmente funcionaban dentro del contexto de un sistema de archivos. Un sistema de Hipermedia Abierta puede contener documentos residiendo en otros, posiblemente remotos, sistemas de archivo. Esto emerge cuestiones con respecto al control de documento y la consistencia de links: ¿Debe el link ser marcado como inválido (o borrado) si se pierde la conexión con el servidor?

- Colaboración: es natural la colaboración sobre el material de una aplicación Hipermedia, y esto debe ser soportado por el sistema. Esto hace emerger cuestiones como: compartir, permisos, bloqueo de acceso, etc.

- Manejo de versiones de documentos y de estructuras Hipermedia: mientras que el manejo de versiones de estructuras Hipermedia, presumiblemente debe estar bajo el control del sistema Hipermedia, el del documento puede ser un problema, por ejemplo: ¿Dónde debe apuntar un link, a la versión original del documento que apuntaba, o la última versión? ¿Qué sucede si la última versión no contiene el punto extremo del link en cuestión?

Esta última cuestión se relaciona con el servidor y el cliente de Hipermedia. Las cuestiones de interfaz de usuario y de localización de puntos extremos son problemas del lado del cliente. La consistencia del link, en la mayoría de las implementaciones de sistemas de Hipermedia Abierta, son también cuestiones del cliente, éste es quien determina el deterioro de los link, ya que esto es generalmente descubierto cuando se lo decora (resalta su punto extremo). Si el servidor es capaz de almacenar especificaciones de localización lo suficientemente generales, el cliente será capaz de intentar regenerar el link buscando el ancla perdida. Por otro lado si la especificación de localización del modelo Hipermedia emplea una especificación de localización muy específica, tal como un desplazamiento en *bytes*, esto no será posible.

El problema de la consistencia de links relacionado a la carencia de control de documentos es una característica general de los sistemas. Distintos sistemas Hipermedia abierta adoptan distintas decisiones de diseño para abordarlo.

VIKI

El sistema de Marshall (Marshall y Shipman, 1995) introdujo la noción de Hipermedia espacial, una forma implícita de determinar estructura basándose en las características de la distribución en la representación de los componentes. Dado un arreglo de iconos de componentes y compuestos en una ventana, VIKI puede automáticamente agrupar componentes de acuerdo a su proximidad, color, forma y otras características. Estas estructuras determinadas espacialmente, son un ejemplo perfecto de los compuestos calculados y temporales.

Web y HTML

Inventada hace "pocos" años por Berners-Lee, la Web dio al término Hipermedia amplio reconocimiento. Más importante, es el hecho de que fundamentalmente alteró las posibilidades (en un sentido amplio) para el acceso distribuido a la información. El éxito de la Web, se debe en parte a su adopción de ciertos principios de Hipermedia Abierta, incluyendo el vinculado entre *hosts* de Internet, protocolos extensibles y el *script*. Sin embargo, la Web y el HTML no son suficientes en lo que respecta al soporte de Hipermedia Abierta a las necesidades de los usuarios. Links uni direccionales embebidos en etiquetas HTML y un pobre soporte para anotaciones y autoría cooperativa, son entre otros, los tópicos que necesitan ser abordados.

Afortunadamente, la comunidad Web, particularmente la W3C, está haciendo progresos en estas y otras áreas. Quizás lo más excitante desde la perspectiva de Hipermedia Abierta es el trabajo hacia un estándar para la distribución y el manejo de versiones basado en la Web, denominado WebDAV (www.ics.uci.edu/~ejw/authoring/). Una vez estandarizado, el protocolo resultante soportaría anotaciones distribuidas sobre Internet.

Xanadu

Junto con el Memex de Bush, este sistema es el más influyente de los sistemas Hipermedia nunca construido. Creado por Nelson (1981), junto con los términos Hipertexto e Hipermedia, la visión de Xanadu inspiro muchos de esos campos de hoy en día. El diseño incluye, un manejo sofisticado de versiones de links y contenido, identificadores universales únicos para fragmentos de contenido, y una clara separación de funcionalidad entre *frotnend* y *backend*. La *transclusión* de Xanadu sugiere una nueva noción del link, como un mantenedor dinámico de identidad entre copias de contenido (Nelson, 1995). Xanadu fue concebido basado en un modelo económico de "pague por utilizar", pero nunca hubo una implementación completa del mismo, aunque prototipos parciales han sido demostrados.

Una de las más conocidas visiones de un hiperespacio totalmente abarcador, fue la articulada en su libro clásico, "Computer Lib/Dream Machines" de 1974. Luego Nelson, acuño el termino docuverso para denotar un universo de documentos intervinculados en red (1981).Una de las contribuciones del grupo del proyecto Xanadu de Nelson, fue la invención de una técnica para identificar unívocamente material arbitrario cargado en un docuverso. La técnica, empleaba representaciones compactas llamadas *tumblers* que se parecen a los *strings* numéricos del sistema decimal de Dewey y a la del numero IP de un *host* de Internet. La universalidad de esta aproximación supone, en principio, que cualquier documento almacenado en el docuverso, puede ser unívocamente referenciado en cualquier momento (los tumblers pueden también referenciar versiones previas de material creado en el docuverso). La publicación en Xanadu, era una cuestión de conceder acceso a una porción del docuverso previamente mantenida privada por su autor.

Apéndice B

DTD del Protocolo OHP de Hipermedia Abierta

En este apéndice se presenta el DTD del Protocolo de Hipermedia Abierta desarrollado por el OHSWG (*Open Hypermedia System Working Group*). Se puede acceder a una descripción detallada del mismo en http://users.ecs.soton.ac.uk/hcd/ohp/ohp.htm.

```
<!--

        OHP Navigational Document Type Definition for defining OHPNav messages.

        This DTD defines the on-the-wire specification for OHP-Nav as being
        used for the demo at Hypertext '99, Darmstadt.

        Navigational Middleware Services and fronted entities are expected to
         communicate these definitions using plain TCP/IP sockets where the
         first 19 bytes denote the length of a message. E.g.

                0000000000000000377<OHP><MESSAGESET><MESSAGE><MESSAGEHEADER>...

        ************************************************************************
        Last change: May 26 1999 16:45 GMT

        Updates to May 17 1999 12:00 GMT
        - IDSET in context has been renamed to MEMBERIDSET as
          suggested by Lennert

        Updates to Apr 13 1999 22:30 GMT
        - some element types for the loc spec definition have not been defined.
This
          has been fixed.

        Updates to Apr 12 1999 12:00 GMT
        - contextsetretrieved, nodesetretrieved and endpointsetretrieved now all
          return sets (to be consistent as Lennert pointed out)

        Updates to Feb 18 1999 10:22 GMT
        - As suggested by Lennert, PSPECSET is now called PSPECIDSET and
          only IDs are transferred.

        Updates to Feb 16 1999 18:26 GMT
        - REFERENCE in locspec has been changed to ID
        - xSET is used consistently throughout the DTD
        - all message names have nounVerb syntax, e.g. LinkCreate, LinkCreated,
etc.

        Updates to Feb 11 1999 18:00 GMT
        - OHPNAVVersion now with a hyphen between component protocol and actual
version
        - entity not internal (rather than external) to allow inclusion in xml
examples

        Updates to Feb 10 1999 13:00 GMT
```

```
        - CONTEXT had two compIDs, changed to one
        - ANCHORTEMPLATE now has %locspec; instead of LOCSPEC

        Updates to Jan 15 1999, 16:06 GMT
        - computation defined according to new style, i.e. not using attributes
          but plain elements. Some element types for computation were missing
-->

<!ENTITY OHPNAVVERSION                  "OHPNAV-1.3-Darmstadt 1999">

<!--
        Pete, Henning and Sigi decided to support only four types.
        DELETE means to not only remove all values but to remove
        the whole object (i.e. attribute). CREATE sets the value even
        if the object does not exist yet (it should then be created
        by the component). It should be noted that all of these
        operations should always be processed gracefully, e.g.
        deleting something that is not there should not result in
        an error but simply be ignored by the server. As the resulting
        object is returned anyway this would always give the client
        (= requesting component) a chance to check whether its request
        has been successful. -->
<!ENTITY % modType "REMOVE | CREATE | ADD | DELETE">

<!--    The following is the complete list of message to be used in
        OHP. For details see below. -->
<!ENTITY % allMessNames  "CONTEXTCREATE   | CONTEXTCREATED     |
                          LINKCREATE      | LINKCREATED        |
                          ENDPOINTCREATE  | ENDPOINTCREATED    |
                          ANCHORCREATE    | ANCHORCREATED      |
                          NODECREATE      | NODECREATED        |
                          PSPECCREATE     | PSPECCREATED       |

                          CONTEXTDELETE   | CONTEXTDELETED     |
                          LINKDELETE      | LINKDELETED        |
                          ENDPOINTDELETE  | ENDPOINTDELETED    |
                          ANCHORDELETE    | ANCHORDELETED      |
                          NODEDELETE      | NODEDELETED        |
                          PSPECDELETE     | PSPECDELETED       |

                         CONTEXTMODIFY   | CONTEXTMODIFIED    |
                          LINKMODIFY      | LINKMODIFIED       |
                          ENDPOINTMODIFY  | ENDPOINTMODIFIED   |
                          ANCHORMODIFY    | ANCHORMODIFIED     |
                          NODEMODIFY      | NODEMODIFIED       |
                          PSPECMODIFY     | PSPECMODIFIED      |


                         CONTEXTRETRIEVE   | CONTEXTRETRIEVED    |
                          LINKRETRIEVE      | LINKRETRIEVED       |
                          ENDPOINTRETRIEVE  | ENDPOINTRETRIEVED   |
                          ANCHORRETRIEVE    | ANCHORRETRIEVED     |
                          NODERETRIEVE      | NODERETRIEVED       |
                          PSPECRETRIEVE     | PSEPCRETRIEVED      |

                         LINKFOLLOWFROMLOCSPEC     | LINKFOLLOWED              |
                          LINKFOLLOWFROMENDPOINT    | GETENDPOINTSETFROMNODE    |
                           ENDPOINTSETRETRIEVED      | GETNODESETFROMCONTEXTSET  |
                           NODESETRETRIEVED          | DISPLAYANCHOR             |
```

```
                          GETCONTEXTSET                    | CONTEXTSETRETRIEVED
|
                         DISPLAYNODE               | ERROR ">

<!ENTITY % hmObject       "CONTEXT       | LINK            | CC                  |
                           ENDPOINT      | ANCHOR          | NODE          |
                           COMPUTATION   | PSPEC">

<!ENTITY % abstractObject "SESSIONSTATE  | SESSIONRECORD   | LOCATIONRECORD |
                          SUBSCRIPTION  | %hmObject;">

<!ENTITY % locSpec      "(NALOC | AXISLOC)">
<!--    Dave & Sigi: MYTYPE is optional because it should only be used
        if the actual type is different, e.g. Dave'sEndpoint, etc.
        -->
<!ENTITY % abObjInfo   "MYTYPE?, DESCRIPTIONSET?, CHARACTERISTICSET?">

<!--    PSPEC
        Pete and Sigi: change for demo: PSPECIDSET is optional because
        nobody is using it anyway. -->
<!ENTITY % hmObjInfo   "(%abObjInfo;), COMPUTATIONID?, PSPECIDSET?">
<!ENTITY % locSpecInfo        "SELECTION?, SELECTIONCONTEXT?, SELECTIONTYPE?">

<!--
        SELECTION is the actual piece of data that makes the content of an anchor.
        SELECTIONCONTEXT is the surrounding text (or any data) that is used for
        (re-)finding an anchor in case the content has been changed.
        SELECTIONTYPE denotes the MIMETYPE of the selection.
-->
<!ELEMENT SELECTION    (#PCDATA)>
<!ELEMENT SELECTIONCONTEST (#PCDATA)>
<!ELEMENT SELECTIONTYPE        (#PCDATA)>

<!ELEMENT ID           (#PCDATA)>
<!ELEMENT MEMBERIDSET  (ID, ID*)>
<!ELEMENT MYTYPE       (#PCDATA)>

<!--    MESSAGESET
        This is the basic structure of an OHP message to be sent over the
        wire as text (with 19 leading bytes expressing the length of
        the message). An OHP document consists of one message plus any number
        of additional messages. -->
<!ELEMENT OHP          (MESSAGESET)>
<!ELEMENT MESSAGESET   (MESSAGE, MESSAGE*)>

<!--    MESSAGE STRUCTURE
        An OHP message consists of a message header  plus a message body which
        is one of the messages. -->
<!ELEMENT MESSAGE       (MESSAGEHEADER, (%allMessNames;))>

<!--    USER DETAILS
        Currently we only support a userid and an optional name.
        Further extensions might be needed for collaboration. -->
<!ELEMENT ACCOUNT      (USERID, NAME?)>
<!ELEMENT USERID       (#PCDATA)>
<!ELEMENT NAME         (#PCDATA)>

<!--    MESSAGEHEADER
        The message header contains the following fields.
```

```
        - SENDER (optional): the sending component
        - RECEIVER (optional): the receiving component. Both fields
          could be (are) used to allow routing of messages.
        - SERIAL: the unique message ID
        - RETURNSERIAL (optional: used if a message is referring to a previous
          serial.
        - SESSION (optional): a session identifier
        - ACCOUNT (optional): user data
        - MNAME: the name of the message. This field allows e.g. routing
          of messages by looking at the header only (the content could
          be encrypted).
        - PROTOCOL: this basically is version information about the
          protocol spoken.
        - CERT (optional): an optional certificate that can be used
          for security reasons.
        - CONTEXTIDSET: the set of contexts that this message applies to.
        - PERFORMATIVE (optional): a performative as known from agent
          communication languages that allows dealing with messages
          without actually understanding their content. -->

<!ELEMENT MESSAGEHEADER  (SENDER?, RECEIVER?, SERIAL, RETURNSERIAL?, SESSION?,
                ACCOUNT?, MNAME, PROTOCOL, CERT?, CONTEXTIDSET, PERFORMATIVE?)>

<!ELEMENT SENDER       (#PCDATA)>
<!ELEMENT RECEIVER     (#PCDATA)>
<!ELEMENT SERIAL       (#PCDATA)>
<!ELEMENT RETURNSERIAL (#PCDATA)>
<!ELEMENT SESSION      (#PCDATA)>

<!--    MESSAGE NAME (MNAME)
        We cannot call this "message" in XML because of name space
        clashes (we could resolve this, but wan't to keep
        the DTD simple). -->
<!ELEMENT MNAME        (#PCDATA)>
<!ELEMENT PROTOCOL     (#PCDATA)>
<!ELEMENT CERT         (#PCDATA)>
<!ELEMENT CONTEXTIDSET (CONTEXTID, CONTEXTID*)>
<!ELEMENT CONTEXTID    (#PCDATA)>
<!ELEMENT PERFORMATIVE (#PCDATA)>

<!--    CREATE MESSAGES
        Actually the returns, i.e. the creatEDs, deleteDs, etc. should
        be objects and the requests, i.e. delete, retrieve, etc. should
        be object IDs (to improve performance). Returning complete objects
        will also be better for notificational purposes. However, we
        will have to provide a set of flags for what has been changed as
        well (in particular for modification messages). -->

<!--    CREATES -->
<!ELEMENT CONTEXTCREATE              (CONTEXT)>
<!ELEMENT CONTEXTCREATED       (CONTEXT)>
<!ELEMENT LINKCREATE           (LINK)>
<!ELEMENT LINKCREATED          (LINK)>
<!ELEMENT ENDPOINTCREATE       (ENDPOINT)>
<!ELEMENT ENDPOINTCREATED      (ENDPOINT)>
<!ELEMENT ANCHORCREATE               (ANCHOR)>
<!ELEMENT ANCHORCREATED        (ANCHOR)>
<!ELEMENT NODECREATE           (NODE)>
<!ELEMENT NODECREATED          (NODE)>
```

```
<!ELEMENT PSPECCREATE          (PSPEC)>
<!ELEMENT PSPECCREATED                 (PSPEC)>

<!--    DELETES
        Note that if notification is used complete objects might be
        returned. -->
<!ELEMENT CONTEXTDELETE                (ID)>
<!ELEMENT CONTEXTDELETED       (ID)>
<!ELEMENT LINKDELETE           (ID)>
<!ELEMENT LINKDELETED          (ID)>
<!ELEMENT ENDPOINTDELETE       (ID)>
<!ELEMENT ENDPOINTDELETED      (ID)>
<!ELEMENT ANCHORDELETE         (ID)>
<!ELEMENT ANCHORDELETED        (ID)>
<!ELEMENT NODEDELETE           (ID)>
<!ELEMENT NODEDELETED          (ID)>
<!ELEMENT PSPECDELETE          (ID)>
<!ELEMENT PSPECDELETED                 (ID)>

<!--    MODIFY
        Note again, when using notification the changed values
        might be returned (see above). -->
<!ELEMENT CONTEXTMODIFY                (ID, CONTEXT, CONTEXTTEMPLATE, MODTYPE)>
<!ELEMENT CONTEXTMODIFIED      (ID)>
<!ELEMENT LINKMODIFY           (ID, LINK, LINKTEMPLATE, MODTYPE)>
<!ELEMENT LINKMODIFIED         (ID)>
<!ELEMENT ENDPOINTMODIFY       (ID, ENDPOINT, ENDPOINTTEMPLATE, MODTYPE)>
<!ELEMENT ENDPOINTMODIFIED     (ID)>
<!ELEMENT ANCHORMODIFY                 (ID, ANCHOR, ANCHORTEMPLATE, MODTYPE)>
<!ELEMENT ANCHORMODIFIED       (ID)>
<!ELEMENT NODEMODIFY           (ID, NODE, NODETEMPLATE, MODTYPE)>
<!ELEMENT NODEMODIFIED                 (ID)>
<!ELEMENT PSPECMODIFY          (ID, PSPEC, PSPECTEMPLATE, MODTYPE)>
<!ELEMENT PSPECMODIFIED        (ID)>

<!--    RETRIEVE
        If the template is not there the linkserver
        (or whatever component is dealing with the message) should
        return the whole thing. -->
<!ELEMENT CONTEXTRETRIEVE      (ID, CONTEXTTEMPLATE?)>
<!ELEMENT CONTEXTRETRIEVED     (CONTEXT)>
<!ELEMENT LINKRETRIEVE                 (ID, LINKTEMPLATE?)>
<!ELEMENT LINKRETRIEVED        (LINK)>
<!ELEMENT ENDPOINTRETRIEVED    (ID, ENDPOINTTEMPLATE?)>
<!ELEMENT RETRIEVEDENDPOINT    (ENDPOINT)>
<!ELEMENT ANCHORRETRIEVE       (ID, ANCHORTEMPLATE?)>
<!ELEMENT ANCHORRETRIEVED      (ANCHOR)>
<!ELEMENT NODERETRIEVE                 (ID, NODETEMPLATE?)>
<!ELEMENT NODERETRIEVED        (NODE)>
<!ELEMENT PSPECRETRIEVE        (ID, PSPECTEMPLATE?)>
<!ELEMENT PSPECRETRIEVED       (PSPEC)>

<!--    QUERIES
        We (Pete & Sigi) decided not to support queries in this version
        of the DTD because we'd rather do them properly than doing a
        quick hack and we are simply using some basic queries for the
        demo. Hence, we agreed on the following queries. -->

<!--    GETCONTEXTSET
```

```
        This is the boot message that a client would send to a server in order
        to get the available linkbases (i.e. contexts). Expected reply see next
        line. The complete list of contexts is returned. -->
<!ELEMENT GETCONTEXTSET         EMPTY>
<!ELEMENT CONTEXTSETRETRIEVED   (CONTEXTSET)>
<!ELEMENT CONTEXTSET            (CONTEXT, CONTEXT*)>

<!--    GETNODESETFROMCONTEXTSET
        Use this message once you have retrieved the available contexts. Expected
        reply see second line.
        We (Lennert, Pete, Sigi) decided that for the demo we would not have any
        really huge files so for the time being we return (complete) nodes rather
        than IDs. -->

<!ELEMENT GETNODESETFROMCONTEXTSET (CONTEXTIDSET)>
<!ELEMENT NODESETRETRIEVED      (NODESET)>
<!ELEMENT NODESET               (NODE, NODE*)>


<!--    GetEndpointSetFromNode
        This message retrieves all endpoints given a node. Expected
        reply see below. -->
<!ELEMENT GETENDPOINTSETFROMNODE (ID)>
<!ELEMENT ENDPOINTSETRETRIEVED (ENDPOINTSET)>
<!ELEMENT ENDPOINTSET           (ENDPOINT, ENDPOINT*)>

<!--    OTHERS -->
<!ELEMENT DISPLAYANCHOR                 (ANCHOR)>
<!ELEMENT DISPLAYNODE           (NODE)>
<!ELEMENT ERROR                 (#PCDATA)>

<!--    DESCRIPTIONS
        Descriptions are single value; it is planned to provide a
        language attribute for foreigners for future use.
        Change by Sigi in Aarhus: descriptionsets can now actually be
        empty; this is useful when the descriptionset tag is used in
        a template, e.g. the nodetemplate. -->
<!ELEMENT DESCRIPTIONSET        (DESCRIPTION*)>
<!ELEMENT DESCRIPTION           (NAME, VALUE)>
<!--    future use: this could be used to display things in different languages
        ATTLIST DESCRIPTION
        language        (en | de | dk | tx | us | fr) #IMPLIED -->

<!--    CHARACTERISTICS
        Characteristics are multiple value (as opposed to single
        value descriptionset.
        Change by Sigi in Aarhus: characteristicsets can now actually be
        empty; this is useful when the characteristicset tag is used in
        a template, e.g. the nodetemplate. -->
<!ELEMENT CHARACTERISTICSET     (CHARACTERISTIC*)>
<!ELEMENT CHARACTERISTIC        (NAME, VALUESET)>

<!--    PSPECIDSETS
        Change by Sigi in Aarhus: PSEPCIDSETs can be empty to be used in e.g.
        templates  -->
<!ELEMENT PSPECIDSET            (PSPEC*)>

<!--    NODE -->
<!ELEMENT NODE                  (ID, NAME?, CONTENTSPEC, (%hmObjInfo;))>
```

```
<!ELEMENT NODETEMPLATE                (ID?, NAME?, MYTYPE?, DESCRIPTIONSET?,
                                CHARACTERISTICSET?, COMPUTATIONID?, PSPECIDSET?,
CONTENTSPEC?)>

<!--    CONTENTSPEC
        Change by Sigi in Aarhus: VERSION and MIMETYPE can be optional to allow
        usage in templates. -->
<!ELEMENT CONTENTSPEC           (URL?, CONTENT?, CHARACTERISTICSET?, VERSION?,
MIMETYPE?)>
<!ELEMENT CONTENT               (#PCDATA)>
<!ELEMENT URL                   (#PCDATA)>
<!ELEMENT VERSION               (#PCDATA)>
<!ELEMENT MIMETYPE              (#PCDATA)>

<!--    ENDPOINT
        Note: actually this should be a reference (i.e. an ID) to an
        anchor rather than an anchor objects itself. However, implementation
        experience has shown that it is not efficient to pass around IDs,
        hence we have an anchor object. (might be changed ...)
        Note also, that endpoints only can have one linkID -->
<!ELEMENT ENDPOINT              (ID, NAME?, LINKID?, DIRECTION,
                                 (%hmObjInfo;), ANCHORID)>
<!ELEMENT LINKID                (#PCDATA)>
<!--    direction is either BIDIRECTIONAL or SOURCE or DESTINATION -->
<!ELEMENT DIRECTION             (#PCDATA)>
<!ELEMENT ANCHORID              (#PCDATA)>

<!ELEMENT ENDPOINTTEMPLATE      (ID?, NAME?, MYTYPE?, DESCRIPTIONSET?,
                                 CHARACTERISTICSET?, COMPUTATIONID?, PSPECIDSET?,
                                 LINKID?, DIRECTION?, ANCHORID?)>

<!--    ANCHOR
        ParentID is the "thing" that is actually linked to, i.e. it
        does not have to be a node but it could be anything else
        with an ID (e.g. an endpint, link, etc.). -->
<!ELEMENT ANCHOR                (ID, NAME?, PARENTID?, (%locSpec;), (%hmObjInfo;))>
<!ELEMENT ANCHORTEMPLATE        (ID?, NAME?, MYTYPE?, DESCRIPTIONSET?,
                                 CHARACTERISTICSET?, COMPUTATIONID?, PSPECIDSET?,
(%locSpec;)?)>

<!--    NALOC
        This is short for not accessible location, e.g. a book on a shelf. -->
<!ELEMENT NALOC                 ((%locSpecInfo;), SPEC, VERSION, ID)>

<!--    AXISLOC
        Location via axes. The basic idea follows the HyTime standard. -->
<!ELEMENT AXISLOC               ((%locSpecInfo;), FWDAXISSET, REVAXISSET, VERSION,
                                 ID, OVERRUN)>
<!ELEMENT SPEC                  (#PCDATA)>
<!ELEMENT FWDAXISSET            (AXIS*)>
<!ELEMENT REVAXISSET            (AXIS*)>
<!ELEMENT AXIS                  (NAME?, TYPE, VALUESET)>

<!ELEMENT VALUESET              (VALUE*)>
<!ELEMENT VALUE                 (#PCDATA)>

<!--    LINK
        Links link a set of endpointIDs together. -->
<!ELEMENT LINK                  (ID, NAME?, (%hmObjInfo;), ENDPOINTIDSET?)>
```

```
<!ELEMENT ENDPOINTIDSET                (ID, ID*)>

<!ELEMENT LINKTEMPLATE                 (ID?,  NAME?, MYTYPE?, DESCRIPTIONSET?,
                                    CHARACTERISTICSET?, COMPUTATIONID?, PSPECIDSET?,
                                    ENDPOINTIDSET?)>

<!--    CONTEXT
        A context object is like a set of related IDs. Hence, e.g.
        linkbases are represented as contexts. -->
<!ELEMENT CONTEXT                   ((%hmObjInfo;), MEMBERIDSET, ID?, NAME?)>
<!ELEMENT COMPUTATIONID                (#PCDATA)>

<!ELEMENT CONTEXTEMPLATE            (ID?, NAME?, MYTYPE?, DESCRIPTIONSET?,
                                     CHARACTERISTICSET?, COMPUTATIONID?, PSPECIDSET?,
IDSET?)>

<!--    PSPEC
        This is short for presentation specification. -->
<!ELEMENT PSPEC                     (ID, NAME?, VERSION?, (%hmObjInfo;), OPAQUESPEC)>
<!ELEMENT OPAQUESPEC                (#PCDATA)>

<!ELEMENT PSPECTEMPLATE                (ID?, NAME?, MYTYPE?, DESCRIPTIONSET?,
                                    CHARACTERISTICSET?, COMPUTATIONID?, PSPECIDSET?,
OPAQUESPEC?)>

<!--    Non-paradigmatic operations
        The following set of operations needs to be supported by
        OHP compliant components in addition to the basic
        data model operations. -->

<!--    LinkFollowFromLocSpec
        The structure server finds the anchors with a locSpec that equals
        the locSpec that is given as input parameter. From the set of anchors
        the structure server finds the associated set of endpoints and their
        associated links. For each of the links the structure server returns
        the other endpoints of the links that have a matching direction.
        The server only looks for anchors in nodes that are members of the
        contexts.
        ComputationId identifies a computation that should be run. When (i.e.
        at which time) is not yet specified. We would need a proper
        event model. If compudationId is the null identifier no computation
        will take place.
        LinkFollowed is the server's reply to the client's request
        Linkfollowfromlocspec. For efficiency reasons a set of
        endpoint/anchor pairs is returned.
        Issue here: in most situations you will need to retrieve the parent node
        (via parentID) but there are cases where the parent would not be a node,
        e.g. a link. Hence, we (Lennert, Pete, Sigi) decided to return a set of
        anchor-parent pairs which we define below. -->
<!ELEMENT LINKFOLLOWFROMLOCSPEC ((%locSpec;))>
<!ELEMENT LINKFOLLOWED          (ANCHORPARENTPAIRSET)>
<!ELEMENT ANCHORPARENTPAIRSET   (ANCHORPARENTPAIR*)>
<!ELEMENT ANCHORPARENTPAIR      (ANCHOR, (%hmObject;))>


<!--    LinkFollowFromEndpoint
        This is the "classic" follow link. -->
<!ELEMENT LINKFOLLOWFROMENDPOINT (ENDPOINTID)>
```

Referencias

[1] Conklin J.: "Hypertext: An introduction and Survey", IEEE Computer, 20(9) (pp. 17-41), 1987.

[2] Nelson T. H.: "A File Structure for the Complex, the Changing, and the Indeterminate". 20th National Conference, New York, Association for Computing Machinery, 1965.

[3] Bush V.: "As we may think". The Atlantic Monthly (pages 101–108), 1945.

[4] Nelson T. H.: "The Hypertext". Proceedings International Documentation Federation Annual Conference, 1965.

[5] Nelson T. H.: "Replacing the printed word". Information Processing (pages 1013– 1023), 1980.

[6] Berners Lee T., Cailliau R., Groff J. F., and Pollerman B.: "World Wide Web: The information universe". Electronic Networking: Research, Applications and Policy, 1(2), 1992.

[7] Halasz, F. G., and Schwartz, M. D.: "The Dexter Hypertext Reference". In Proc. of Hypertext Standardization Workshop (pp. 95-133), Gaithersburg, USA, 1990.

[8] Engelbart D.: "Authorship provisions in Augment". In Proceedings of the IEEE Compcon Conference, San Francisco, USA, 1984.

[9] Akscyn R. M., McCracken D. L., and Yoder E. A.: "KMS: A distributed hypermedia system for managing knowledge in organizations". Communications of the ACM, 31(7):820–835, 1988.

[10] Halasz F. G., Moran T. P., and Trigg R. H.: "NoteCards in a nutshell". In Proceedings of ACM Conference on Human Factors in Computing Systems and Graphics Interface (pages 45–52), Toronto, Canada, 1987.

[11] Halasz F. G.: "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia System". Communications of the ACM, 31(7) (pp 836-852), 1988.

[12] Yankelovich N., Haan B. J., Meyrowitz N. K., and Drucker S. M.: "Intermedia: the concept and the construction of a seamless information environment". IEEE Computer Society (pages 81–96), 1988.

[13] Meyrowitz, N.: "The missing link: Why we're all doing hypertext wrong". In Barrett, E. (ed.). The society of text: Hypertext, hypermedia and the social construction of information (pp. 107- 114) MIT Press, Cambridge, Massachusetts., USA, 1989.

[14] Gronbaek, K.: "Composites in a Dexter-Based Hypermedia Framework". In proceedings of the ACM European Conference on Hypermedia Technology (ECHT '94) (pp. 59-69), Edinburg, UK, 1994.

[15] Pearl A.: "Sun's link service: A protocol for open linking". In Proceedings of the 2nd ACM Conference on Hypertext (pages 137–146). Pittsburgh, USA, 1989.

[16] Clark W.: "Motif Applications + LinkWorks = Hyperenvironment". In Proceedings of the ACM Conference on Hypertext. Milan, Italy, 1992.

[17] Rizk, A., and Sauter, L.: "Multicard: An open hypermedia system". In ECHT '92. Proceedings of the ACM conference on Hypertext (pp. 4-10), Milan, Italy, 1992.

[18] Anderson K. M., Taylor R. N., and Whitehead E. J. Jr.: "Chimera: Hypermedia for heterogeneous software development environments". In proceedings of the ECHT´94. European conference on Hypermedia Technologies (pp. 94-107), Edinburgh, Scotland. New York: ACM Press, 1994.

[19] Campbell, B., Goodman, J.: "HAM: A general purpose hypertext abstract machine". Communications of the ACM, Vol. 31, Nº 7 (pp. 856-861), 1998.

[20] Schnase J. L., Leggett J. J., Hicks D. L., Nürnberg P. J., and Sánchez J. A.: "Design and implementation of the hb1 hyperbase management system". Electronic Publishing - Origination, Dissemination and Design, 6(1) (125–150), 1993.

[21] Fountain M., Hall W., Heath I., and Davis H. C.: "Microcosm: An open model for hypermedia with dynamic linking". In Rizk A., Streiz N., and Andrè J., editors, Proceedings of the European Conference on Hypertext, Cambridge University Press, U.K., 1990.

[22] Gronbaek, K., and Wiil, U. K.: "Towards a common reference architecture for open hypermedia". In JoDI, Journal of Digital Information, I(2), 1997.

[23] Wiil U. K. and Leggett J. J.: "The HyperDisco approach to open hypermedia systems". In Proceedings of the 7th ACM Hypertext Conference, (pages 140–148), Washington DC, USA, 1996.

[24] Nürnberg P. J.: "HOSS: An Environment to Support Structural Computing". PhD thesis, Department of Computer Science, Texas A&M University, College Station, Texas, USA, 1997.

[25] Karousos N., Tzagarakis M., Koumbarou N.: "Selecting Services for Web Applications: The Open Hypermedia Case". International Workshop on Web Engineering in conjunction with ACM Hypertext 2004.

[26] Haake J. M., Knopik T., and Streitz N.: "The SEPIA hypermedia system as part of the POLIKOM telecooperation scenario". In Proceedings of the 5th ACM. Hypertext Conference (pages 235–237). Seattle, USA, 1993.

[27] Wiil U. K.: "Using events as support for data sharing in collaborative work". in International Workshop on CSCW, (Berlin, Germany) (pp. 162—176). Institute of Informatics and Computing Technique, Germany, 1991.

[28] Maurer H.: "Hyperwave — The Next Generation Web Solution". Addison Wesley, 1996.

[29] Maler, E., and DeRose, S.J. (Eds.).: "XML Linking Language (XLink) Design Principles". http://www.w3.org/TR/NOTE-xlink-principles, 1998.

[30] Maler, E., and DeRose, S.J. (Eds.): "XML Pointer Language (XPointer)". http://www.w3.org/TR/WD-xptr, 1998.

[31] Millard D. E., Moreau L., Davis H. C., and Reich S.: "Fohm: a fundamental open hypertext model for investigating interoperability between hypertext domains". In Proc. of the 11th ACM Hypertext Conf. (pages 93–102) San Antonio, TX, USA, May 2000.

[32] Gronbaek K., Bouvin N. O., and Sloth L.: "Designing Dexter-based hypermedia services for the World Wide Web". In M. Bernstein, L. Carr, and K. Osterbye, editors, Proceedings of the 8th ACM Hypertext Conference (pages 146–156) Southampton, UK, 1997.

[33] Gronbaek K., Sloth L., and Bouvin O.: "Open Hypermedia as User Controlled Meta Data for the Web". In Proceedings of the Nineth International World Wide Web Conference (pages 553–566), Amsterdam, Netherlands, 2000.

[34] Gronbaek K., Sloth L., and Orbaek P.: "Webvise: browser and proxy support for open hypermedia structuring mechanisms of the World Wide Web". In Proceedings of the 8th International World Wide Web Conference (pages 253–267), Toronto, Canada, 1999.

[35] Bouvin N. O., Zellweger P. T., Gronbaek K., and Mackinlay J. D.: "Fluid annotations through open hypermedia: Using and extending emerging Web standards". In Proc. of the 11th WWW Conf. (pages 160–172), Honolulu, USA, 2002.

[36] Christensen B. G.: "Xspect: Bridging Open Hypermedia and XLink". In Proc. of the WWW Conf., Budapest, Hungary, 2003.

[37] Berners Lee T.: "The World Wide Web - Past, Present and Future". In JoDI, Journal of Digital Information, I(1), 1996.

[38] Engelbart D.: "A conceptual framework for the augmentation of man's intellect". In Howerton P., editor, Vistas in Information Handling, volume 1, (pages 1–29). Spartan Books, Washington DC, USA, 1963.

[39] Nielsen J.: "The Art of Navigating through Hypertext". Communications of the ACM, 33(3) (287-310), 1990.

[40] Gronbaek K. and Malhotra J.: "Building tailorable hypermedia systems: the embedded-interpreter approach". In Proceedings of the 9th conference on Object Oriented Programming Systems, Language, and Applications (pages 85–101), ACM, 1994.

[41] Gronbaek K., Hem J. A., Madsen O. L., and Sloth L.: "Designing Dexter-based cooperative hypermedia systems". In Proceedings of the 5th ACM Hypertext Conference (pages 25–38), Seattle, USA, 1993.

[42] Gronbaek K.: "Eurocode work package wp2 task t2.2 report: Object oriented model for the distributed hypermedia toolkit". Technical Report CODE-AU-93- 10, Aarhus University, Denmark, 1993.

[43] Gronbaek, K., and Trigg, R. H. "Design issues for a Dexter-based hypermedia system". In Communications of the ACM, 37 (pp. 40-49), 1994.

[44] Gronbaek K., Hem J. A., Madsen O. L., and Sloth L.: "Cooperative hypermedia systems: A Dexter-based architecture". Communications of the ACM, 37(2):64– 74, 1994.

[45] Gronbaek K. and Trigg. R. H.: "Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects". In Proceedings of the 7th ACM Hypertext Conference (pages 149–160), Washington DC, USA, 1996.

[46] Spivey J.M.: "The Z notation". Hertfordshire, England. Prentice-Hall, 1989.

[47] Leggett J. J. and Schnase J. L.: "Viewing Dexter with open eyes". Communications of the ACM, 37(2):77–86, 1994.

[48] Halasz F. G. and Schwartz M.: "The Dexter hypertext reference model". Communications of the ACM, 37(2):30–39, 1994.

[49] Whitehead E. J. Jr.: "Control choices and network effects in hypertext systems". In Proceedings of the 10th ACM Hypertext Conference (pages 75–82), Darmstadt, Germany, 1999.

[50] Brown, P.J. "Turning ideas into products: The Guide System". In ACM Hypertext '87. Proceedings (pp. 33-40), Chapel Hill. New York: ACM Press, 1987.

[51] Halasz, F. and Schwartz, M. (edited by Gronbaek, K. and Trigg, R.H.): "The Dexter Hypertext Reference Model". Communications of the ACM 37 (pp. 31-39), 1994.

[52] Bouvin N. O.: "Unifying strategies for Web augmentation". Proceedings of ACM Hypertext (p 91–100), 1999.

[53] Hall, W., Davis, H., and Hutchings, G.: "Rethinking Hypermedia: the Microcosm approach", Boston, 1996.

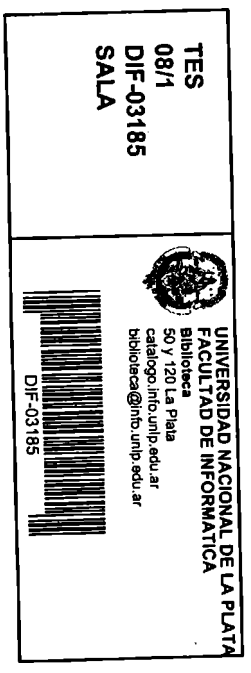
[54] Nürnberg P. J., Legget J. J., Schneider E. R., and Schnase J. L.: "Hypermedia Operating System: A New Paradigm for Computing". In Proceedings of the Seventh ACM Conference on Hypertext (pp. 194-202). Washington D.C., New York: ACM Press, 1996.

[55] Landow G. P.: "Relationally Encoded Links and the Rhetoric of Hypertext". In Hypertext '87 (pp. 331-343). New York: ACM Press, 1987.

[56] Akscyn, R. M., McCracken, D. L.: "*ZOG and the USS CARL VINSON: Lessons in System Development*", Carnegie-Mellon Technical Report CMU-CS-84-127. In Proceedings of the First IFIP Conference on Human-Computer Interaction; Interact '84; London, England, 1984.

[57] Akscyn, R. M., McCracken, D. L.: "Design of Hypermedia Script Languages: The KMS Experience". In Proceedings of ACM Hypertext '93 (pp. 268-269), Seatle. New York: ACM Press, 1993.


29 NOV 2007


TES
08/1
DIF-03185
SALA
UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMATICA
Biblioteca
50 y 120 La Plata
catalogo.info.unlp.edu.ar
biblioteca@info.unlp.edu.ar
DIF-03185