Un Modelo para Visualización de Resultados en WWW

Omar Alonso[1] y Ricardo Baeza-Yates

DCC, Universidad de Chile

oralonso@yahoo.com, rbaeza@dcc.uchile.cl

Resumen. En este trabajo analizamos el problema de la dependencia de la interfaz gráfica con el software de recuperación de información. Nuestro modelo separa la parte de recuperación de la interfaz. Esto es particularmente útil cuando el usuario quiere seleccionar una metáfora de visualización que puede no estar disponible para todos los sistemas de recuperación de información o cuando el hardware disponible imposibilita su uso. Presentamos un modelo cuya parte central es la definición de una representación intermedia que puede ser usada como lenguaje de intercambio de datos entre el software de visualización de información y el software de recuperación de información. Dada la representación intermedia, un diseñador de metáforas de visualuzación de información puede proveer a los usuarios diferentes opciones para especificar búsquedas y observar resultados.

1. Introducción

El uso de la tecnología de World Wide Web (WWW) ha permitido que muchos sistemas en distintas plataformas sean accedidos usando una interfaz común, un navegador, desde diferentes sitios geográficos.

De las aplicaciones actuales sobre WWW hay dos tipos que predominan claramente: acceso a bases de datos y búsqueda y recuperación de información. En la primera, la tendencia es de migrar aplicaciones previas en modo cliente-servidor a WWW como parte de la solución de comercio electrónico. Nuestro interés se centra en la segunda opción: búsqueda y recuperación de información [4,11].

El proceso de descubrimiento y acceso a la información disponible en WWW es una actividad primaria que requiere el uso de herramientas esenciales. A medida que aumenta la información en WWW, el número de herramientas y aplicaciones disponibles para facilitar el proceso se ha incrementado notablemente.

Las herramientas y tecnologías disponibles hoy en día no resuelven el tema de recuperación y visualización de información en WWW [14]. Existen varios problemas abiertos cuyas soluciones pueden generar interesantes trabajos de investigación y productos comerciales.

Recuperación y visualización de información son componentes claves de sistemas más complejos como bibliotecas digitales [1,20] y administración de conocimiento [27].

El proceso de acceso a información se puede dividir en dos componentes principales: 1) búsqueda y recuperación de información y 2) visualización (análisis y síntesis) de los resultados.

1.1 Búsqueda y Recuperación de Información

El proceso de búsqueda consiste en que el usuario especifica de alguna manera qué es lo que desea encontrar y el sistema responde con una representación generalmente tabular de la respuesta. Dado que las respuestas no siempre son lo que se espera, la etapa que sigue es un filtrado o reformulación de la consulta hasta encontrar la información deseada.

El escenario típico para buscar, recuperar y desplegar información es el siguiente. Un usuario necesita información acerca de un determinado tópico. Con la ayuda de una interfaz, formula una consulta al sistema

[1] Afiliación actual: Oracle Corp., 500 Oracle Parkway, Redwood Shores, CA 94065.

de búsqueda (1). La consulta activa una acción en el sistema (motor de búsqueda, sistema de recuperación de información, biblioteca digital, etc.) (2). El sistema recuperará (o no) items y los desplegará con información adicional en la misma interfaz donde el usuario ingresó la consulta (3). Finalmente, el usuario evalúa los resultados y decide si son relevantes o no. Puede abandonar del sistema porque encontró la información deseada o puede refinar la consulta y formular una nueva (4).

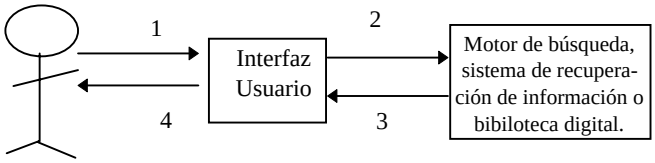

Figura 1. Escenario para buscar, recuperar y visualizar información

El proceso de búsqueda de información en WWW puede ser realizado de las siguientes dos formas:

- El usuario ingresa palabras claves o frases usando la interfaz de motores de búsqueda (AltaVista, NorthernLight, Lycos, etc.) o navega y busca sobre clasificaciones predefinidas (Yahoo!). La tecnología usada pertenece al dominio de recuperación de información. La tecnología subyacente de un motor de búsqueda varía. Algunos ejemplos son archivos invertidos, PAT, modelo booleano, etc.
- Usando un agente de software autónomo que realiza búsquedas e informa al usuario de los resultados. La tecnología usada pertenece al dominio de inteligencia artificial o programación orientada a agentes [7]. Una instancia particular de un agente de software es un agente con capacidades de recuperación de información, cuyas funciones principales es buscar y recuperar información relevante dada una especificación [19,22].

1.2 Visualización de Resultados

Las interfaces de los motores de búsquedas disponibles hoy en día son intuitivas y en algunos casos restringidas por la naturaleza misma de WWW. Hay uso limitado de color, la interacción es mínima, y muchas características disponibles en otros sistemas convencionales practicamente no existen. La interfaz tipo es un campo de texto de una sola línea donde el usuario ingresa los términos a buscar y dos botones: uno para remitir la consulta y otro para cancelar su ingreso. Hay interfaces que sólo incluyen el botón para remitir la consulta.

El proceso de visualización de respuestas puede ser netamente textual o más rico y complejo con el uso de metáforas de visualización gráficas. En el enfoque textual, el usuario recibe una lista de los primeros diez o veinte mejores documentos que contienen la información buscada. Cada item de la lista contiene la URL del documento, su título, tamaño, día de modificación y un resumen del mismo. Típicamente, el resumen en sí son los primeros ochenta caracteres (a veces menos) del documento. Toda esta información se condensa en unas cuatro líneas en promedio. Haciendo uso de la URL el usuario puede acceder al documento de interés.

Una observación interesante es que, sin hacer un estudio formal, se puede apreciar que más del 50% del contenido de una página Web no tiene que ver con los resultados de la consulta. Por ejemplo: avisos comerciales, información corporativa, etc.

Como se puede apreciar, el usuario no puede manipular la disposición de los resultados en la pantalla y está sujeto a usar una interfaz estática que no se puede personalizar y mucho menos reemplazar.

Este enfoque no es incorrecto si el documento a ubicar se encuentra entre los primeros veinte o en la primera página de respuestas. Resulta un problema cuando la respuesta de una consulta es una lista de cientos de documentos. Una metáfora de visualización gráfica resuelve este problema con una interfaz rica en la cual el usuario puede navegar, filtrar, procesar y reformular una consulta.

La idea central en el uso de visualización de información es aumentar la cognición. Esto significa que es útil apreciar el rol del mundo externo en el proceso de pensamiento y razonamiento. Mediante esta observación se pueden diseñar representaciones externas que amplifican la cognición.

La visualización de información es una herramienta efectiva para resolver parcialmente problemas de sobrecarga de datos en recuperación de información en WWW. También se ha comenzado a explorar otros tipos de interfaces para WWW [14]. De todas formas, las interfaces corrientes son fijas y no permiten herramientas de visualización como *plugs in.*

Desde un punto de vista de hardware, el proceso de búsqueda y recuperación de información en WWW está limitado a pantallas de alta resolución ignorando otros dispositivos de conectividad a la red. Si bien, existen navegadores para un asistente personal (por ejemplo Newton y PalmPilot), una metáfora de visualización sofisticada no es práctica en esta situación. Nuestro enfoque también aporta una solución en esta área. El usuario puede especificar, sobre la base de su hardware disponible, que tipo de visualización desea utilizar.

2. Estado del Arte

En esta sección describimos los trabajos relacionados en el área al igual que ciertos productos y tecnología que consideramos relevante en nuestro estudio.

2.1 Recuperación de Información

Desde el punto de vista de recuperación de información o descubrimiento de recursos, uno de los proyectos más importantes es Harvest [8] y su arquitectura distribuída. No sólo tecnológicamente, sino por su influencia en productos comerciales (Netscape) y otros proyectos de investigación como Search Broker y USE. Una taxonomía sobre herramientas para buscar información en WWW está presentada en [24].

El re-descubrimiento de ciertas tecnologías y la creación de una nueva área como bibliotecas digitales [1,20], ha posibilitado el desarrollo e incorporación de ciertos protocolos como Z39.40 o la propuesta de nuevos, como STARTS. En paralelo con el desarrollo de nuestro trabajo otro enfoque fue propuesto [40].

Desde el punto de vista industrial, existen una gran cantidad productos de recuperación de información en el mercado. Muchos de estos productos son de uso tanto en Internet como en Intranet. Algunos de ellos: AltaVista, NorthernLight y Yahoo! por ser pioneros en el uso público. *inter*Media Text (Oracle) por tener un enfoque hacia bases de datos, OpenText por su tecnología PAT, Inktomi por el concepto de servicio e InfoSeek por ser un producto netamente basado en WWW. Finalmente Google y DirectHit son ejemplos de alternativas a los enfoques tradicionales de búsqueda y recuperación en WWW.

2.2 Visualización de Información

El área de visualización ha estado muy activa en los últimos años con la baja en los precios de memorias y hardware. Definimos visualización de información al uso de representaciones visuales e interactivas de abstracciones de datos para amplificar el uso o adquisición de conocimiento. Existen taxonomías para clasificar metáforas de visualización de información y visualización científica que no describiremos en detalle [9,18].

Es importante notar que no existe una metáfora de visualización general que sea aceptable para todos los dominios. Es interesante observar que sobre la cantidad de metáforas de visualización de información que se encuentran publicadas en la literatura, muy pocas han sido exitosas de tal forma que sean usadas a menudo o formen parte de productos comerciales.

Las siguientes son algunas metáforas interesantes. Scatter/Gather realiza agrupación de documentos en grupos temáticos coherentes y presenta resúmenes textuales al usuario [10]. TileBars [17] representa efectivamente una lista de items con información contextual. InfoCrystal es una herramienta para visualizar y tambíen es un lenguaje de consulta visual [32]. Hyperbolic Tree es un navegador que implementa una

técnica de foco+contexto que está basada en una geometría hiperbólica [25]. JAIR es un espacio de información en el cual se representan objetos e información [23]. Sherlock provee un mecanismo para incorporar un *plug-in* en la interfaz [39].

2.3 Representaciones Visuales

Desde el punto de vista cognitivo se ha realizado investigación referente a las distintas representaciones visuales de ciertos objetos. Por ejemplo, bajo qué circunstancias es más fácil entender un gráfico que una tabla con números.

Lohse *et al*. desarrollaron experimentos sobre sesenta representaciones visuales donde identificaron once agrupaciones principales: gráficos, tablas, tablas gráficas, diagramas temporales, redes, diagramas de estructura, diagramas de procesos, mapas, cartogramas, íconos y figuras [26]. Una de las cosas importantes del estudio fue cómo ciertos tipos de visualizaciones comunican conocimiento. La relación con nuestro estudio se centra en que diferentes usuarios pueden preferir cierto tipo de visualización en favor de otras. Los sistemas actuales no permiten este tipo de elecciones.

Green *et al*. proponen *previews* y *overviews* como representaciones de información (textuales o gráficas) que pueden ser extraídas de objetos primarios [15]. Un *preview* se define como un avance de uno o varios objetos. Un *overview* es una colección de objetos. La idea central es que un usuario puede usar *previews* y *overviews* para discriminar objetos de interés de aquellos que no presentan mayor atractivo. El modelo que presentamos en este trabajo se puede usar para construir *previews* y *overviews*.

2.4. Lenguajes de Marcas

El auge de la tecnología Web ha hecho que se re-descubran cosas como SGML [12] y se propongan mejoras para ciertas aplicaciones como XML [13] y RDF [36].

SGML (Standard Generalized Markup Language) es un lenguaje de marcas que es ampliamente usado en ambientes industriales de publicación de documentos y administración (y manejo) de grandes volúmenes de documentos. SGML era conocido en aplicaciones muy específicas hasta que con la llegada de WWW, se empezó a trabajar con HTML (Hyper-Text Markup Language) que originalmente fue diseñado en base a SGML. XML (eXtensible Markup Language), un subconjunto simplificado de SGML que mantiene extensibilidad, estructura y validación [13]. RDF (Resource Description Framework) es una recomendación del W3 Consortium para definir la arquitectura necesaria para soportar el intercambio de metadatos en Web [35,36]. Finalmente con la llegada de los dispositivos de conexión móviles han surgido algunas propuestas para acceder a páginas HTML desde artefactos como teléfonos celulares y asistentes personales. Eejemplos de este tipo de propuestas son WDM, (Wireless), HDML (Handheld Device Markup Language [37]) y NVML (NaVigation Markup Language [38]).

3. Modelo y Arquitectura de Software

Queremos hacer que el proceso de visualización sea independiente del proceso de búsqueda de una manera estándar. Por estándar entendemos que diferentes metáforas de visualización puedan interactuar con un software de recuperación de información y que diferente software de recuperación de información puedan producir datos para la misma metáfora de visualización. Es decir, la interfaz al usuario es independiente del sistema de recuperación y en algún sentido podemos decir que es definida por el usuario de un conjunto de interfaces disponibles [2].

Conceptualmente nuestro modelo consiste de tres partes: un conjunto de buscadores, un conjunto de visualizadores y un lenguaje de marca como representación intermedia que permite que las dos primeras partes se comuniquen.

La figura 2 muestra el punto de vista del usuario con el del sistema en acción. Un buscador está representado por un círculo. En el ejemplo, B_1 es un agente de software con características de recuperación de información. B_2 es una aplicación específica que contiene características de recuperación (asistido por

un *crawler*), el sistema de archivos del sistema operativo y una base de datos. Un visualizador está representado por círculo de doble trazo. En el ejemplo, V_1 es una implementación de HBP [3] y V_2 es un árbol hiperbólico.

Los buscadores y visualizadores intercambian información usando una representación intermedia común que será introducida más adelante. La manera de operar los datos en ésta representación intermedia depende de la implementación. Se pueden guardar los contenidos en un archivo para despues interpretarlos o se puede procesar el contenido en memoria directamente. A efectos de notación solamente, la figura sugiere que los datos se almacenan en un archivo de tipo RI (representación intermedia).

Un usuario tiene una necesidad de buscar algo y define su buscador (B_1) y visualizador (V_1). El buscador B_1 define una búsqueda sobre WWW, es decir es un motor de búsqueda. El visualizador V_1 es una metáfora similar a TileBars. B_1 escribe en un archivo lo que ha encontado como respuesta en formato de representación intermedia (RI). V_1 luego lee los contenidos de ese archivo y despliega al usuario los resultados usando su metáfora de visualización.

Por otro lado un usuario distinto define un buscador (B_2) y visualizador (V_2). El buscador B_2, realiza otro tipo de búsqueda usando un sistema de recuperación un poco más sofisticado que usa información local en el sistema de archivos y su propia base de datos, más WWW. B_2 escribe como respuesta a lo que ha encontrado en formato de representación intermedia (RI) en un archivo. V_1, que es una implementación de la metáfora de pila de libros, interpreta el contenido de ese archivo y presenta al usuario los resultados.

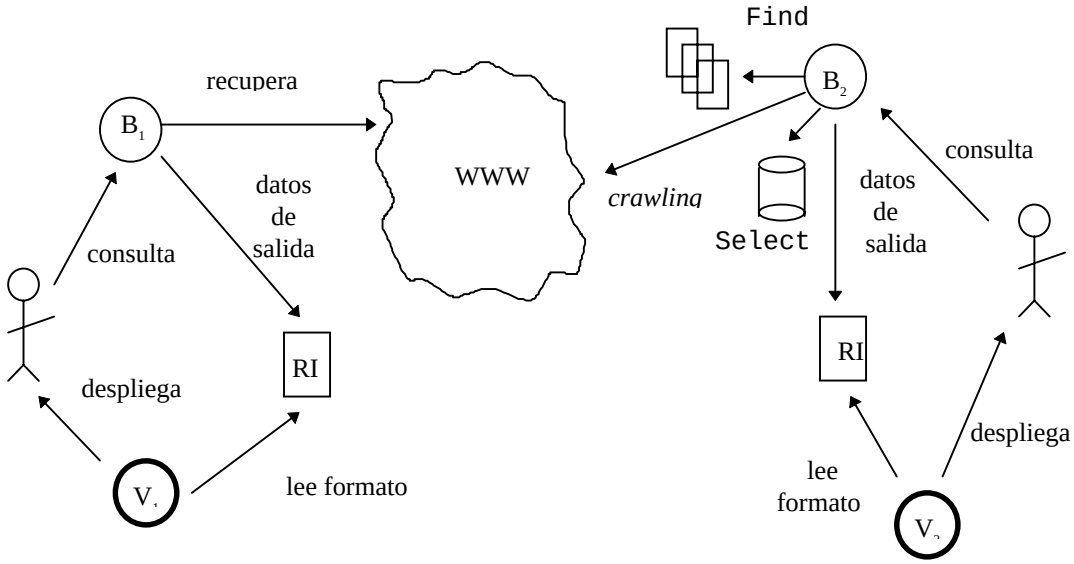

Figura 2. Una descripción general del modelo y sus componentes.

3.1 Representación Intermedia

Se necesita una representación intermedia para que los buscadores y los visualizadores intercambien datos entre ellos. Estamos interesados en los siguientes datos de los resultados de las búsquedas para su posterior despliegue por parte de un visualizador:

- Query (consulta).
- URL (Uniform Resource Locator).
- Title (Título del documento).
- Abstract (Un sumario o síntesis de su contenido, generalmente los primeros 80 caracteres del documento).
- Author (Autor del documento).
- Content-length (Longitud del contenido del documento).

- Content-type (Tipo de contendio: texto, imagen, etc.).
- Last-modified (Fecha de ultima modificación).
- Score o rank (Métrica sobre la pertinencia y relevancia del documento).

Esta representación intermedia debe ser manipulada por todos los buscadores y todos los visualizadores. Para aquellos que usan un formato interno especial, se debe transformar la representación intermedia a su propio formato interno. Por ejemplo, en una consulta sobre "information visualization", la salida de un buscador debería ser algo similar a una lista ordenada con items como: *rank*, URL, resumen, fecha de última modificación y tamaño de la página (entre otros).

```
1
http://www.cs.panam.edu/info_vis/home-info_vis.html
Information Visualization at U. Tx. - Pan American
Information Visualization. at University of Texas - Pan American. The purpose of
computing is insight, not numbers. - Richard Hamming, 1962. The goal of...
Last modified 18-Oct-96
page size 4K

2
http://www.elastictech.com/
Elastic Technology - Information visualization java tools
Java technology to visualize, navigate, and manage large information...
Last modified 27-Apr-99
page size 4K

3
http://graphics.stanford.edu/courses/cs348c-96-fall/resources.html
Some Information Visualization Resources on the Web
Information Visualization Resources on the Web. This page is a partial collection of
online InfoVis resources. See the accompanying bibliography for...
Last modified 29-May-99
page size 14K
```

La siguiente tabla (tabla 1) muestra una tabla de datos con la misma información.

URL	URL_1	URL_2	URL_3	…
Rank	1	2	3	…
Size	4	4	14	…
Last	18-oct-96	27-Apr-99	27-May-99	…
Content type	text	text	text	…
Title	…	…	…	…
Author	…	…	…	…
Abstract	…	…	…	…

Tabla 1. Tabla de datos.

Donde URL_1 = http://www.cs.panam.edu/info_vis/home-info_vis.html, URL_2 = http://www.elastictech.com/, URL_3 = http://graphics.stanford.edu/courses/cs348c-96-fall/resources.html. Los "…" indican texto.

Podemos ver que esta información podría ser estructural y semánticamente más rica si correspondiera a un lenguaje.

El visualizador toma el archivo de resultados y despliega la información de acuerdo a su metáfora. Una ventaja de este formato intermedio es que un buscador puede incluir diferentes atributos de cada documento que pueden ser usados por algunos visualizadores y por otro lado, algunos visualizadores pueden hacer uso opcional de buscadores que proveean más información.

Separando estructura y contendio de presentación, el mismo documento de representación intermedia puede ser escrito una sóla vez y luego desplegado en diferentes formas (un monitor de computadora, un teléfono celular, un asistente personal, etc.).

3.1.1 IVL

IVL son las siglas de Lenguaje de Visualización de Información (Information Visualization Language) que es la implementación de la representación intermedia descrita en la sección previa.

Existen otros formatos de representaciones intermedias para diferentes aplicaciones. Quizás uno de los más conocidos en el área de representación de conocimiento es KIF (Knowledge Interchange Format) que se ha usado en varios proyectos de inteligencia artificial. Otra representación intermedia de interés es MCF (Meta Content Framework, [16]) introducida algunos años atrás por Apple. La aplicación más interesante de MCF es HotSouce que corresponde a una visualización en tres dimensiones para navegar sitios Web.

Por otro lado ha habido gran interés en la extracción de datos semiestructurados de páginas Web. El ejemplo más notable es VDBMS (Virtual Data Base Management System) de Junglee [28] el cual hace que WWW y otras formas de datos externos se comporten como una única base de datos relacional. Desde el punto de vista de investigación un enfoque basado en extracción de datos basados en ejemplos se presenta en [29].

Nuestro énfasis en un lenguaje se debe a que nos interesa independizarnos de cualquier formato propietario. Además se puede automatizar la generación de empaquetadores (*wrappers*) como es el caso de varios de los proyectos que hacen extracción de datos semiestructurados.

Estamos interesados en usar XML y RDF como opciones para implementar IVL en un ambiente basado en WWW. ¿Por qué XML o RDF? Nuestro interés en XML y RDF se debe a que con el auge de soluciones de comercio electrónico, muchos formatos comunes entre sistemas existentes (EDI por ejemplo) están siendo re-evaluados en el entorno de XML. Incluso existen propuestas de MCF en XML [77]. Siguiendo la misma línea de descripción de metadatos, MDF (Multimedia Description Format) es una propuesta para descripción de contenidos de documentos con rico contenido multimedial. Inktomi expone parte de sus datos usando un esquema parecido de marcas.

Además esperamos que tanto XML como RDF evolucionen a medida que la tecnología WWW evoluciona. Hoy en día hay más productos y lenguajes que proveen características para el manejo de XML sobre RDF.

3.2 Arquitectura del Modelo

En el resto de esta sección describiremos la arquitectura de nuestro modelo siguiendo la notación introducida por Shaw y Garlan para definir estilos arquitectónicos en el marco del diseño de una arquitectura de software [30]. La figura 3 describe el esquema general de nuestra arquitectura.

Dada una consulta (especificada por el usuario), el Manejador de Buscadores selecciona el Buscador que es de preferencia del usuario el cual ejecuta esa consulta. Los resultados de esa consulta pueden estar ya en formato IVL o no. Si los datos están en IVL y el visualizador seleccionado entiende IVL, se muestran los resultados usando la metáfora de visualización correspondiente en base a la preferencia del usuario.

Si los datos del buscador están en IVL pero el visualizador no entiende o los datos del buscador no están en IVL y el visualizador si entiende IVL, se procede a transformar esos formatos internos a IVL y viceversa.

El manejador de buscadores y el manejador de visualizadores consultan información en la misma base de datos (o repositorio) del sistema. Si bien, nos inclinamos por tener una solución centralizada de las bases de datos o repositorios, hemos dejado libre la elección en la arquitectura.

Los componentes más importantes de esta infraestructura son los dos manejadores (buscadores y visualizadores) y algún mecanismo para transformar formatos propietarios en IVL. Se espera que algunos de los buscadores y visualizadores sean provistos como componentes externos.

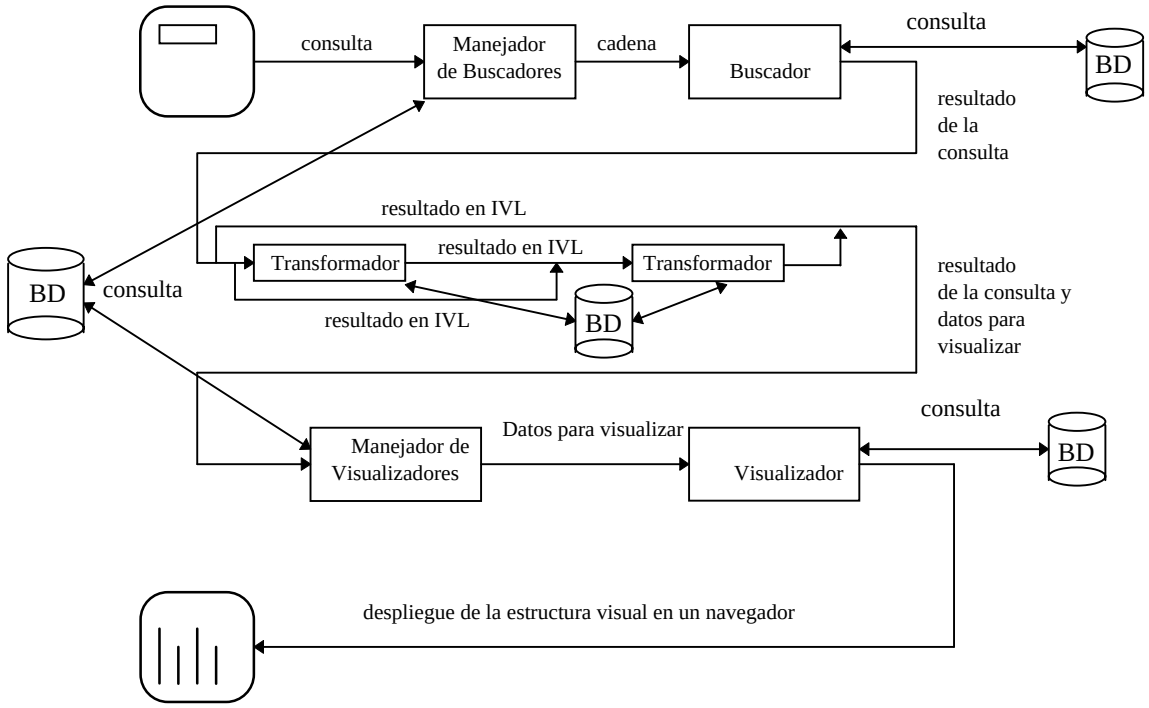

Figura 3. Descripción del modelo usando estilos de arquitectura.

3.4 Integración

A lo largo de este trabajo nos hemos referido de un modo implícito a un cierto esfuerzo de integración de ciertos componentes. Sería iluso pretender que todos los buscadores y visualizadores puedan manipular IVL sin ningún problema. Y como en casi todos los proyectos, siempre existirá el problema de integración con algún sistema propietario o antiguo pero que sigue en funcionamiento.

En esta sección describiremos una solución del punto de vista de ingeniería de software para resolver el problema. La tecnología llamada sistemas transformacionales existe desde hace varios años y ha sido usada primariamente en el área de compiladores [6]. Sin embargo con el auge de WWW y la explosión de administración de contenido en diferentes formatos y lenguajes, su uso se ha extendido notablemente.

El uso de soluciones con sistemas transformacionales require experiencia en el dominio de aplicación y a veces resulta una tarea difícil y dura. Pero las ganancias en tiempo de desarrollo, tiempo de integración y mantenimiento son muy buenas

Volviendo al tema principal, un punto de integración muy imporante es el siguiente. Supongamos que existe un visualizador disponible que posee una metáfora muy interesante pero que no entiende IVL ¿Cómo

lo integramos al sistema? Para ejemplificar el problema de integrar un buscador con un visualizador, presentamos dos sistemas B y V. Supongamos que un buscador B retorna resultados en el siguiente formato:

```
rank, URL, resumen, fecha modificación, tamaño;
```

Un ejemplo de los resultados es:

```
1,http://www.dcc.uchile.cl,12-nov-99,12Kb;
2,http://www.go.com,10-oct-98,4Kb;EOF
```

Y supongamos que un visualizador V que está implementado como una Java *applet* tiene parámetros los cuales representan su formato interno.

```
<APPLET CODE="v.class" HEIGHT="400" WIDTH="500">
<PARAM NAME="rank", VALUE="1">
<PARAM NAME="URL", VALUE="http://www.dcc.uchile.cl">
<PARAM NAME="fecha_mod", VALUE="12-nov-99">
<PARAM NAME="tamaño", VALUE="12Kb">     …
</APPLET>
```

El problema a resolver es cómo transformar un formato basado en comas (",") del buscador B en un formato para una *applet* que es basado en marcas. En este caso la proyección es directa y basta con escribir un transformador que se encarge de agregar las marcas del *applet* por cada campo separado por comas. Esa solución se complica cuando hay un cierto número de buscadores y un cierto número de visualizadores. Es altamente costoso y laborioso construir y mantener un transformador por cada posible proyección.

En general hay dos soluciones posibles para el problema: escribiendo un adaptador para V específicamente o usando un esquema transformacional para todos los Vs. La primera solución es difícil de mantener y laboriosa ya que requiere escribir el adaptador cada vez que un visualizador o buscador no "entiende" IVL. La segunda es mucho más formal lo que significa que es de costo alto al principio pero que luego el servicio es automático. El uso de un esquema basado en sistemas transformacionales es de gran utilidad si se espera integrar una gran cantidad de visualizadores y buscadores entre sí. Si la cantidad de buscadores y visualizadores es mínima, el desarrollo y mantenimiento de un reducido número de transformadores puede ser una solución viable.

Un rasgo clave en el uso de un motor transformacional junto a una representación intermedia es que es posible automatizar la generación de adaptadores o empaquetadores. Nuestro enfoque se diferencia en aquellos basados en Z39.50 o STARTS ya que proponemos una arquitectura que contiene un componente que realiza transformación semántica y por ende es escalable.

Se puede implementar un sistema transformacional simple usando XLS/XSLT como tecnología base asumiendo que la mayoría de los formatos son del estilo XML o HTML.

4. Prototipos

En esta sección describimos la implementación en rasgos generales de los componentes más importantes del modelo. Para ello hemos desarrollado dos prototipos que demuestran los conceptos más importantes

4.1 BookMedia

BookMedia (Bookpile with *inter*Media) es un prototipo desarrollado en Java y Java *servlets* que consiste de *inter*Media Text (Oracle) como buscador y una versión simplificada de la metáfora de visualización de pila de libros horizontal.

Nuestra metáfora de visualización gráfica llamada "horizontal bookpile" (HBP) está basada en [3]. Usamos una metáfora de biblioteca o pila de libros para visualizar documentos dependiendo si la herramienta se usa horizontal o verticalmente.

Cada documento (que se ve como un libro) está representado como un rectángulo de color gris, altura y ancho con una posición en el conjunto. Cada uno de esos atributos gráficos, incluyendo el orden en la lista pueden ser proyectados a un atributo del documento (*score*, tamaño, etc.). Estas proyecciones seleccionadas por el usuario permiten estudiar diferentes correlaciones de atributos sobre el conjunto de documentos, ayudando al usuario a seleccionar el documento apropiado.

Nuestra implementación es un subconjunto de toda la funcionalidad introducida por la HBP original con algunos re-diseños de los componentes. La figura 4 muestra HBP.

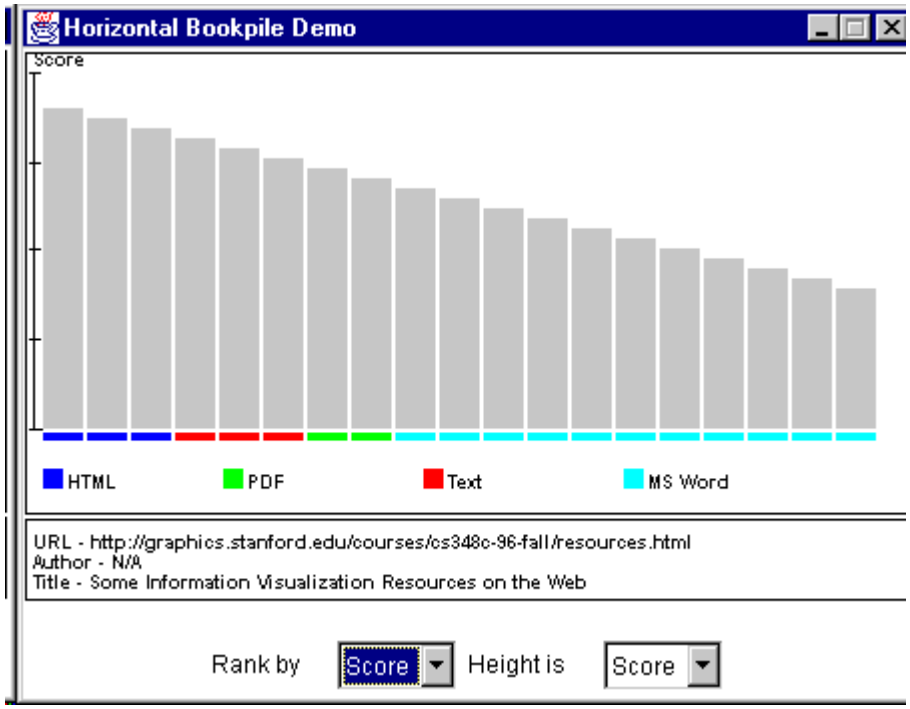

Fig. 4. La pila de libros horizontal.

El diseño de la visualización de los documentos sigue la arquitectura de pequeños múltiplos introducida por Tufte. En la opción por defecto el eje X representa el *score* y el eje Y la cantidad de elementos jerarquizados por *score*. Para que la visualización de la métrica sea más fácil, el eje X está marcado en cuatro partes, cada una representando un cuarto de un céntimo.

Los colores de los documentos son iguales pero su tipo no. Por ejemplo, los tres primeros son de tipo HTML y los tres siguientes son de tipo texto puro (ASCII plain text). Esta información está visualizada por pequeños rectángulos coloreados debajo de cada documento y a su vez como indicaciones de la cantidad de tipos representados. En las figuras se muestran siempre cuatro tipos (HTML, PDF, Text, y MS Word).

4.2 SAlta

SAlta (Searching with AltaVista) es un prototipo que consiste de AltaVista como el buscador y una versión simple de tablas como visualizador (figura 5).

Una vez que el usuario ingresa las palabras claves, el procesador de consulta transforma esas palabras claves en la sintaxis de AltaVista y le pasa la consulta al servidor. El servidor (que es parte de SAlta) se encarga de abrir una conexión a la red directamente y hace uso de AltaVista.

AltaVista retorna los resultados en su propio formato y el servidor se encarga de tomar esos datos. El transformador de resultados toma ese flujo (*stream*) de código HTML y lo transforma en IVL. El

componente responsable de desplegar la tabla toma IVL y despliega la tabla de resultados que ahora puede ser apreciada por el usuario.

SAlta puede ser considerado en parte como un párasito ya que depende de otro motor de búsquedas para poder satisfacer la consulta. Nuestra arquitectura no hace tanto énfasis en el componente "parásito" como otros sistemas sino que se lo incluye como un componente más.

SAlta es una solución barata desde el punto de vista de implementación y producción. Sólo se necesita un ambiente Java y un servidor de WWW para que funcione tanto en UNIX como Windows NT.

No se necesita tener conocimientos de bases de datos o de algún producto de recuperación de información para que funcione. Sólo se necesita saber cómo invocar a AltaVista. De todas formas lo que parece ser muy fácil y barato termina siendo de alto de costo de mantenimiento. Cada vez que AltaVista cambia sus formatos o la secuencia de llamada, se debe cambiar el código de SAlta y recompilar su fuente. Y esto en un sistema de producción es inaceptable. Si se desea elegir éste tipo de solución se recomienda usar la API de AltaVista en lugar de tratar de adivinar su formato.

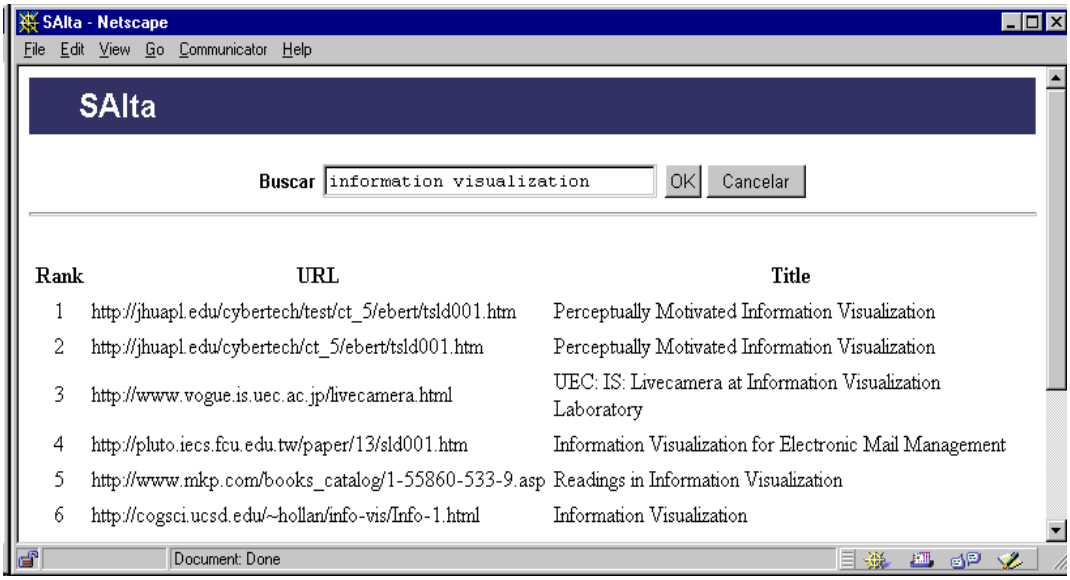

Figura 5. SAlta con un diseño simple de tablas.

5. Resultados y Conclusiones

Hemos presentado un modelo para buscar y visualizar información de documentos en WWW con énfasis en la importancia de una representación intermedia para permitir el intercambio de datos entre diferentes sistemas.

Definimos un lenguaje (IVL) para manipular la integración entre buscadores y visualizadores que a su vez puede ser usado por diseñadores como representación intermedia de diferentes metáforas de visualización. IVL puede ser usado como lenguaje de intercambio de datos entre software de visualización de información y software de recuperación de información.

La representación intermedia puede ser usada para manipular el *score* y para la construcción de *previews* y *overviews* de información de una base de datos al igual que la utilización de las respuestas.

Con tantos sistemas y herramientas para recuperar información en WWW, nuesto modelo define una forma de compartir las salidas de aquellos para que el proceso de visualización sea más independiente y por ende una opción de varias para el usuario.

Una aplicación de las ideas presentadas en este trabajo es en el área de manejo de conocimiento donde existen diferentes sistemas que proveen servicios de búsqueda pero su integración es débil y la visualización de información es muy limitada.

Hay varias cosas que se pueden realizar como trabajos futuros de este proyecto. Para empezar, IVL necesita reflejar más datos de lo que actualmente representa. Esto involucra un esfuerzo de estudio de diferentes formatos internos de un cierto número de productos más la incorporación de algunas características especiales para otros tipos de medios que no sean sólo texto.

Desde el punto de vista integrador es necesario probar IVL con simples metáforas de visualización en un cierto número de dispositivos de hardware que no sea una computadora personal. Por ejemplo implementar acceso y despliegue de información desde un teléfono celular, un asistente personal, un *beeper*, etc. Esto demanda el estudio de los mini-sistemas operativos de éstos disposotivos y el uso de tecnología sin cable (*wireless*) para luego desarrollar el software de acceso y despliegue de información.

En un enfoque del área de interacción hombre-máquina es importante realizar experimentos para probar y demostrar la utilidad de acceder la misma información desde diferentes dispositivos usando una o varias metáforas de visualización de información.

6. Referencias

[1] ACM Special Issue on Digital Libraries (1995).

[2] Omar Alonso and Ricardo Baeza-Yates, "Visualization of large answers in WWW", *Proceedings of SSSC 98*, Antofagasta, Chile (1998). Páginas 2-7.

[3] Ricardo Baeza-Yates, "Visualization of Large Answers in Text Databases", *AVI '96*, Giubbo, Italy (1996). Páginas 101-107.

[4] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman, Endinburgh Gate, London (1999).

[5] Jacques Bertin, *Semiology of Graphics* (Traducción del francés), Univ. of Wisonsin Press (1978).

[6] Ted Biggerstaff, "A Perspective of Generative Reuse", *Annals of Software Engineering*, Volume 5, (1998). Páginas 169-226.

[7] J. Bradshaw (Ed.), *Software Agents*,. The MIT Press, Cambridge, MA. (1997).

[8] C. Browman, Peter Danizg, Darren Hardy, Udi Manber, and Michael Schwartz, "The Harvest information discovery access system", in *Proc. 2nd Int. WWW Conference* (October 1994). Páginas 763-771.

[9] Stuart Card, Jock Mackinlay, and Ben Shneiderman. *Readings in Information Visualization*, Morgan Kaufmann, San Francisco, CA (1999).

[10] Douglass Cutting, Jan Pedersen, David Karger, and John Tukey, "Scatter/Gather: A cluster-based approach to browsing large document collections", *Proc. of the 15th SIGIR Conference*, Denmark (1992). Páginas 318-329.

[11] William Frakes and Ricardo Baeza-Yates, *Information Retrieval. Data Structures and Algorithms*, Prentice-Hall, Englewood Cliffs, NJ. (1992).

[12] Charles Goldfarb, *The SGML Handbook*, Oxford University Press (1990).

[13] Charles Goldfarb and Paul Prescod, *The XML Handbook* Prentice-Hall, Upper Saddle River, NJ. (1998).

[14] Ilan Greenberg and Lee Garber, "Searching for New Search Technologies", *IEEE Computer*, Vol. 32, No. 8 (August 1999).

[15] Stephan Greene, Gary Marchionini, Catherine Plaisant, and Ben Shneiderman, "Previews and Overviews in Digital Libraries: Designing Surrogates to Support Visual Information Seeking", *Journal of the American Society for Information Science (to appear).*

[16] R.V. Guha, "Meta Content Framework" (http://www.xspace.net/hotsauce/mcf.html).

[17] Marti Hearst, "TileBars: Visualization of term Distribution Information in Full Text Information Access", *Proceedings of CHI '95*, Denver, CO (May 1995).

[18] Marti Hearst, "User Interfaces and Visualization" in *Modern Information Retrieval* (by R. Baeza-Yates and B. Ribeiro-Neto), Addison-Wesley, Endinburgh Gate, London. (1999).

[19] IEEE Internet Computing, Special Issue on Software Agents, (July/August 1997).

[20] IEEE Special Issue on Digital Libraries (1996).

[21] Lucas Introna and Helen Nissenbaum. "Defining the Web: The Politics of Search Engines", *IEEE Computer*, Vol. 33, No. 1, (January 2000).

[22] N. Jacobs and R. Shea, “The Role of Java in InfoSleuth: Agent-based Exploitation of Heterogeneous Information Resources”, MCC Technical Report #MCC-INSL-018-96 (1996).
[23] JAIR Information Space (http://www.ai.mit.edu/projects/infoarch/jair)
[24] Charlotte Jenkins, Mike Jackson, Peter Burden, and Jon Wallis. “Searching the world wide web: an evaluation of available tools and methodologies”. *Information and Software Technology*, Vol. 39, No. 14-15 (February 1998). Páginas 985-994.
[25] J. Lamping and R. Rao, “The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarachies”, *Journal of Visual Languages and Computing*, 7(1), (1996).
[26] Gerald Lohse, Kevin Biolsi, Neff Walker, and Henry Rueter, “A Classification of Visual Representations”, *Comm of the ACM*, Vol. 37, No. 12 (December 1994).
[27] Daniel O’Leary, “Enterprise Knowledge Management”, *IEEE Computer*, Vol. 31, No. 3 (March 1998).
[28] Anand Rajaraman, STS Prasad, and Debra Knodel, “Junglee Virtual DBMS”, in *The XML Handbook* Prentice-Hall, Upper Saddle River, NJ. (1998).
[29] Berthier Ribeiro-Neto, Alberto Laender, and Altigran da Silva, “Extracting Semi-Structured Data Through Examples”, *Eighth ACM Int. Conference on Knowledge and Information Management* (ACM CIKM'99), Kansas City, Missouri, 1999.
[30] Mary Shaw and David Garlan, *Software Architecture*, Prentice-Hall, Upper Saddle River, NJ. (1996)
[31] Ben Shneiderman, Donald Byrd, and W. Bruce Croft, “Clarifying Search: A User-Interface Framework for Text Searches”, *Dlib Magazine* (January 1997).
[32] Anselm Spoerri, “InfoCrystal: A Visual Tool for Information Retrieval”, *Proceedings of IEEE Visualization* (1993).
[33] Edward Tufte, *The Visual Display of Quantitative Information*, Graphics Press, Cheshire CT (1982).
[34] Edward Tufte, *Envisioning Information*, Graphics Press, Cheshire CT (1990).
[35] W3C, “Extensible Markup Language (XML)” http://www.w3.org/TR/1998/REC-xml-19980210
[36] W3C, “Resource Description Format” http://www.w3.org/TR/PR-rdf-schema/
[37] W3C, “Handheld Device Markup Language Specification” (http://www.w3.org/TR/NOTE-Submission-HDML-pec.html)
[38] W3C, “NaVigation Markup Language (NVML)” (http://www.w3.org/TR/NVML)
[39] Apple Computer. Sherlock Home Page (http://www.apple.com/sherlock)
[40] Andreas Paepcke, Robert Brandiff, Greg Janee, Ray Larson, Bertram Ludaescher, Sergey Melnik, and Sriram Raghavan, “Search Middleware and the Simple Digital Library Interoperability Protocol”, *D-Lib Magazine*, Vol. 6, No. 3., March (2000).