A Model and Software Architecture for Search Results Visualization on the WWW+

Omar Alonso
Oracle Corp., 500 Oracle Parkway
Redwood Shores, CA 94065 (oalonso@us.oracle.com)

Ricardo Baeza-Yates
Department of Computer Science, Universidad de Chile
Santiago, Chile (rbaeza@dcc.uchile.cl)+

Abstract

In this paper we analyze the dependency problem of the user interface with the information retrieval software. Our approach allows the separation of the user interface from the retrieval component. This is useful when the user wants to select an interface or visualization metaphor that could not always be available for different information retrieval systems. We present a model for visualizing large collections of documents in World Wide Web retrieval, independently of the retrieval system. We describe a software architecture that could be used it to implement a solution in an Intranet or Internet environment. Our proposal allows to ease the use of visualization tools which partially solve the problem of data overload on Internet.

Keywords: Software architectures for WWW search, information visualization, integration with database systems, architectures for IR systems

Introduction

The process of information resource discovery on the World Wide Web (web from now on) is a primary task that requires essential tools. Lately there has been a growth in the number of tools and applications in the information retrieval area that suppose to make the process easier for users. However searching the Web is still, sometimes, frustrating. Not only because of the underlying search engines technology but also because of the current user interfaces [8].

The typical generic scenario for searching, retrieving, and displaying information on the Web is the following (Figure 1). A user has an information need about a certain topic. With a user interface he/she formulates a query to the system (1). The query starts an action in the system (search engine, information retrieval system, digital library, or other software component) (2). The system will retrieve (or not) objects and will display them with appropriate messages and layouts in the same graphical user interface (GUI) where the user entered the query (3). Finally, the user decides if the documents are relevant or not. He/she can either exit the system because the information was found or refine the query and start again (4).

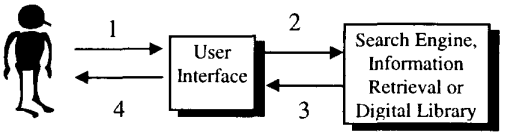

Figure 1. The information seeking process

The process of searching information on the Web can be mainly performed in the following two ways:

- The user types keywords or phrases in search engines (e.g. AltaVista, Hotbot, Northern Light, etc.) or browse and also search over predefined classifications (e.g. Yahoo!). The technology used here belongs to the domain of information retrieval. The underlying technology of a search engine, as an

+ We aknowledge the support of Fondef Grant 96-1064 and Fondecyt Grant 99-0627.

0-7695-0746-8/00 $10.00 © 2000 IEEE

example, can be simple inverted files, vector space models, or Boolean systems [2].

- Using an autonomous software agent to perform searches and inform the user of the results. This type of agents belongs to the area of artificial intelligence or agent-oriented programming [5]. A very useful form of software agents is as an information retrieval agent, whose main function is to find and gather relevant information

An interesting observation to make here is that the retrieval software has a user interface for expressing the query and for displaying the answers. The interface is tightly coupled with the retrieval software and it varies from system to system. Given this scenario the user is not allowed to choose a user interface and a search engine independently. In other words, searching and visualizing information are not independent components.

It is important that the user has an option to select an interface or visualization metaphor that he/she prefers over other ones, even the default retrieval user interface for his/her information seeking process. On the other hand, if a user has a preference on a retrieval system he/she should be able to use its services without hardware or user interface limitations.

In this paper we present a model and a software architecture to solve the problem of interfacing different search tools with different visualization tools. In Section 2 we review related work. In Section 3 we present our model, based in an intermediate XML language. In Section 4 we describe possibles software architectures for our proposal, while in Section 5 we describe two specific prototypes that use this architecture. We end with some concluding remarks and future work.

Related Work

In this section we review related work in academia and industry in the areas of information retrieval, information visualization, and markup languages.

With the boom of the Web and the Internet a lot of information retrieval technology was re-visited and new problems and solutions are available. From the academic perspective there are some important research projects that influenced trends and even become products. Examples are Harvest, the Search Broker, USE, Google, and Clever. There are several public products and services like AltaVista, NorthernLight, and Yahoo! just to mention a few of them that people are very familiar with. From the industry perspective there are products like *inter*Media Text (Oracle), Ultraseek (InfoSeek), LiveLink (OpenText), and Inktomi among others that provides APIs, environments, or services for information retrieval based solutions.

There has been a lot of work with protocols like the Z39.50 standard and the STARTS proposed protocol. At the same time of our work another approach defined as a search middleware was proposed [23]. We will discuss this issue in more detail in our integration section.

Markup languages like SGML were rediscovered although it has been in use for several years in industry. Lately other markup languages are available for different types of tasks that HTML cannot handle. Examples are XML (eCommerce mainly) [7, 18], RDF (metadata) [19], HDML (handheld devices) [20], NVML (navigation) [21], and WAP/WML (wireless) [25].

The area of information visualization has been also quite active lately. There has been a lot of research on different visualization metaphors [6] with a wide range of applications. Examples of information retrieval visualization metaphors and spaces are Scatter/Gather, TileBars, InfoCrystal, Hyperbolic Tree, JAIR, etc. An excellent state of the art of visualization metaphors related to information retrieval is available at [4]. There has been few studies of visualization tools in different dimensions and comparisons between them in terms of performance and results [14]. Green et al. proposed preview and overviews as information representations (textual or graphical) that are extracted from primary objects [9]. Apple's Sherlock provides a plug-in approach for a search interface [22]. There has also been an active interest from industry to provide Internet access (solutions and products) for wireless devices like cell phones, hand held organizers, pagers and wireless appliances (e.g. Oracle's Portal-to-go).

The Model

We define a "searcher" as a software component that performs search and retrieval tasks over the Web. Examples of a searcher are: a search engine (e.g. AltaVista), a software agent with information retrieval capabilities, etc. We define a "visualizer" as the software implementation of an information visualization metaphor. Examples of visualizers are TileBars and HyperbolicTree.

Each visualization metaphor uses a data table to map the data into a visual structure. A visualization can be well designed statistic graph or table [3, 16] or other form of information visualization [17]. Bertin and Tufte's books

provides a theoretical approach to information visualization that can be used for implementing metaphors in software by developers and designers.

Our model consists of a set of searchers, a set of visualizers, and an intermediate representation. The intermediate representation allows searchers and visualizers to interchange data. We are interested in the following data: score, rank, query, URL, Title, Abstract, Author, Content length, Content type, and last date modified.

We defined a language called IVL (Information Visualization Language) that can be used as the intermediate representation and therefore allows searchers and visualizers to interchange data. We describe IVL in EBNF notation as follows:

<IVL> → <Header>, <Query>, <Value>, {<Value>}, <Tail>

<Header> → **ivl**

<Tail> → **end-ivl**

<Value> → <Score> | <Rank>, <URL>, <Title>, <Abstract>, <Author>,
<Content-length>, <Content-type>, <Last-modified>, {<Value>}

<Score> → **score** <Numeric>

<Rank> → **rank** <Numeric>

<Query> → **query** <String>

<URL → **url** <String>

<Title> → **title** <String>

<Abstract> → **abstact** <String>

<Author> → **author** <String>

<Content-length> → **cl** <Numeric>

<Content-type> → **ct** <String>

<Last-modified> → **lm** <String>

<String> → **A..Z** | **a..z** | **0..9** {<String>}

<Numeric> → **0..9** {<String>}

There are at least two interesting alternatives to implement IVL in a Web environment: XML and RDF. We favored the XML implementation because of the following reasons:

- Extensibility: We can extend IVL in the future.
- Structure: Hierarchical data can be modeled to any level of complexity.
- Validation: Data can be checked for structural correctness.
- Media independence: The same content can be published in multiple media.
- Internationalization: XML is based on Unicode so it is possible to mix languages in content and tag names.
- Tools: There are several tools available for any kind of task that involves XML. Major vendors and open source communities are supporting more and more XML.
- eCommerce: XML is part of the electronic commerce infrastructure.

The RDF alternative looks promising in the near future but right now few search applications supports this kind of sophisticated metadata. Inktomi has a similar tags-based approach for its portal services [11].

The XML implementation of IVL consists of a set of tags that corresponds to the data that searchers and visualizers will exchange. An example of the results of the query "information visualization" in IVL is shown in Figure 2.

Our interest is in connecting a given user interface or information visualization metaphor with a different search engine. This approach could be useful for corporate intranets or knowledge management solutions where there are different types of documents and data in different formats that should be accessible from different interfaces and/or devices [12].

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

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

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

Figure 2. IVL example

Software Architecture

This section describes our proposed software architecture using a hybrid architectural style of pipes & filters with repositories [15]. Figure 3 shows the architecture for our proposed solution.

The user has the choice of selecting a searcher and a visualizer as part of his/her profile. Then he/she issues a query to the system. The Search Manager then invokes the specified searcher that will perform the query. Most of searchers uses some kind of data storage that we represented in the architecture as a data base but it could be a file system or some other structure. The results from the query can be in IVL format or not depending if the searcher supports IVL. If the searcher does not support IVL a transformer will map the internal format of the searcher to IVL.

The process of displaying the results using an information visualization metaphor or a different user interface is the following. The Visualizer Manager then invokes the specified visualizer that will display the results for the user. If the visualizer supports IVL then it just displays the results to the user. Otherwise there has to be another transformation from IVL to the internal format of the visualizer. The transformer also uses data storage (in this case a data base) for storing the transformed format

It is important to note that the architecture describes components without specifying a particular vendor or a particular technology. These components work in a Web environment and our interest is to build software using them. The Open Source community is starting to identify this way of development [26]. We will discuss technological issues in the prototypes section.

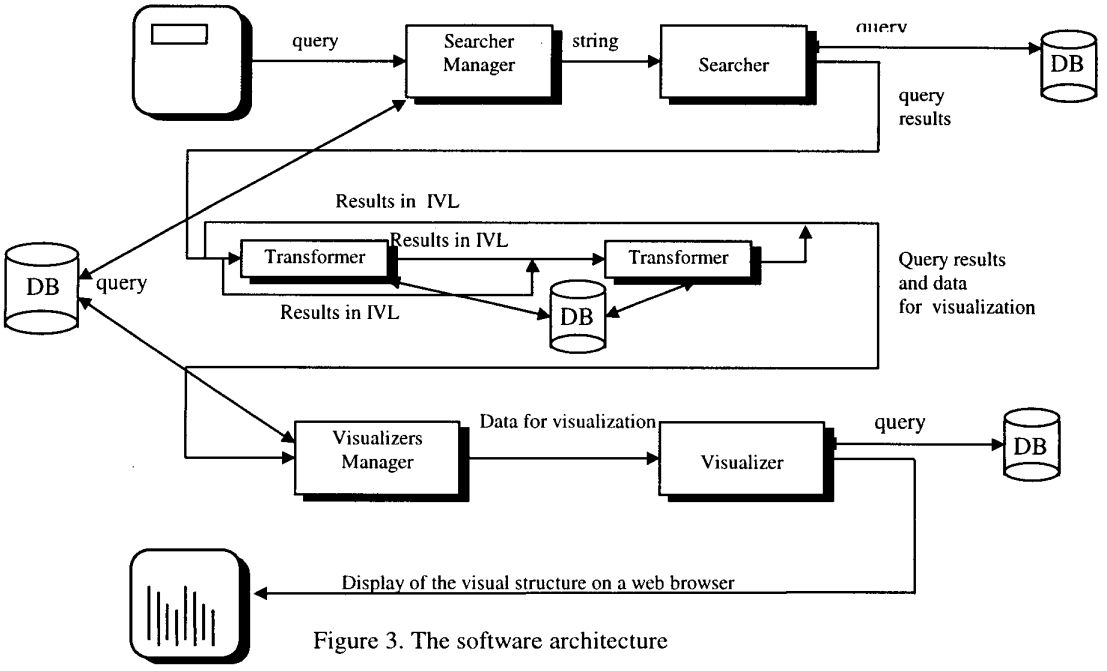

Figure 3. The software architecture

The usage of an intermediate representation like IVL implies that at some point some old formats have to be converted. In other words there has a to be an integration between different components (searchers and visualizers). This is a core piece of our architecture because it is responsible of making sure that searchers and visualizers can exchange data in case they do not support IVL.

There are two ways to solve the problem The first one is to write by hand a transformer for each searcher or visualizer that is new to the solution. This is fine is there are few components in the system but it is a brute force

approach if there are several components and even some of them change the formats often.

The second approach is to use a transformational system [4] that given a specification for a searcher S and a visualizer V will write the transformer for them. This approach is some how expensive at the beginning of the cycle but it is the better choice for long term and also the solution that scales.

A key feature of using a transformation engine along with an intermediate representation is that it is possible to automate, in some cases, wrappers or adaptors generation. As an example, Given IVL, it is very easy to generate a plug-in for Sherlock. In this case we generate a plug in called "MacSearch". MacSearch uses the Sherlock user interface for text searching. *inter*Media Text is the "engine' behind the scenes (note the action URL). This engine is the same component that was used for the BookMedia prototype that we discuss in the next section. The XML is shown below.

```
<search
    name="MacSearch"
    action="http://test-
sun.us.oracle.com/imt/ImtServer"
        type="internet"
    method="GET"
>
   <INPUT NAME="reqDirection" VALUE="First" />
   <INPUT NAME="collection"   VALUE="oa"    />
   <INPUT NAME="qmethod"      VALUE="Web"   />
   <input name="reqstring"    user          />

   <interpret
    resultListStart = "<ivl>"
    resultItemStart = "<score>"
    resultListEnd   = "</end-ivl>"
    resultItemEnd   = "<lm>"
    relevanceStart  = "<score> "
    relevanceEnd    = "</score>"
    />
</search>
```

There are several ways of generating the plug-in code shown below. One way is using XLS/XSLT for transforming XML code (IVL) into the Sherlock specification. Another similar approach is to use Jade and DSSSL. The purist way is to use a generic transformation engine that given the IVL and Sherlock specification generates the plug-in.

Our approach is different from the ones based on Z39.50 or STARTS. First of all we provide an architecture that

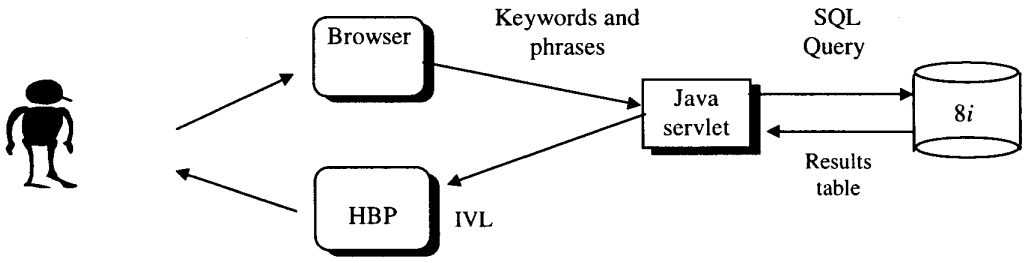

Figure 4. BookMedia Architecture

can handle different search and display components. Each component can interchange data based on an intermediate representation (IVL). Also a component can manipulate IVL to add more information in a given case.

Second, solutions based on standards that not provide some kind of data or semantic transformation don't scale. This is very important because search software evolves (e.g. new releases, new requirements, etc.) and there will always a new piece that needs to be incorporate into the protocol. In other words we have again an integration problem. Our solution incorporates a transformation component that helps to solve the integration problem. The proposed approach can be viewed as an infrastructure for searching and visualization.

The problem of integration is becoming more and more important due to the massive amount of data (structured, semi-structured, and unstructured) in heterogeneous repositories (e.g. databases, file systems, etc.) and the number of existing applications. Adam et al. describes some approaches to system integration in digital libraries [24].

Prototypes

In this section we will describe two prototypes that we wrote to test the concepts: BookMedia and SAlta. BookMedia (Bookpile with *inter*Media) integrates Oracle's *inter*Media Text [13] with a reduced version of the horizontal bookpile metaphor [1]. SAlta (Searching with AltaVista) uses AltaVista as the search engine with a home grown user interface.

There are several ways of implementing the prototypes in a Web environment using Java technology: servlets, applets, JSP (Java Server Pages), and EJB (EnterpriseJava Beans). There are some techniques about where to place the components based on the degree of separation between presentation and business logic. Our emphasis is moving most of the computation to the server side so we choose servlets as the basic model. However we believe that a similar development can be done using other server models.

BookMedia

The user issues a query from the browser. A Java servlet is responsible for opening a JDBC connection to the Oracle 8*i* database, issuing the text query, and retrieving the rows. With the XML supported features from the product, the horizontal bookpile metaphor (HBP) reads the IVL format and displays the results to the user.

The SQL statement has the following structure where the keyword `contains` enables text searching:

```
SELECT Id, Title, Descr, Date
      FROM Doc_table
      WHERE
CONTAINS(document,'information
visualization') > 0;
```

The servlet has to "massage" the results from that query and produce IVL markup code. HBP reads IVL and produce a graphical representation of the result set.

Figure 4 shows the architecture of BookMedia. The back end side is an Oracle 8i database with *inter*Media Text and a Java servlet. The front end side is a variation of the horizontal bookpile metaphor. Figure 5 shows the metaphor in action.

The main advantage of this solution is that has an industrial strength back-end that allows the prototyping of different visualization metaphors and user interfaces. On the other hand it requires a level of expertise in a number of components (database, *inter*Media Text, servlets).

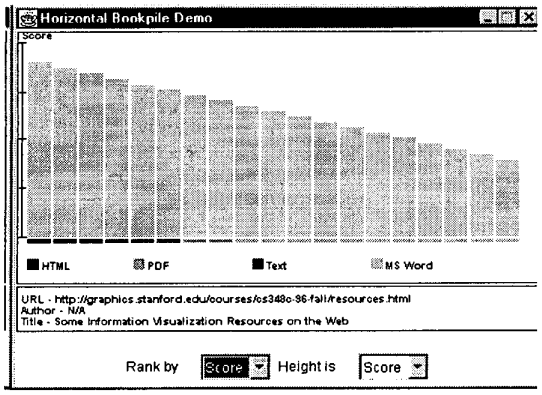

Figure 5. Horizontal Bookpile

SAlta

With SAlta the user specifies a query that the Query Processor translates to AltaVista syntax. The server opens the URL connection with that syntax and returns the results in HTML from for that query. The Results Transformer applies a transformation to that HTML format and generates IVL. Then the results are displayed to the user using an HTML layout table. Figure 6 shows the architecture of SAlta and figure 7 shows the prototype in action.

The are two transformations going on. The first one is to "guess" the real content from the HTML page that AltaVista returned. The output of this process is IVL format. For this particular application IVL is not the choice for presentation so there is another transformation to a particular table layout in HTML.

There are two reason for having a Java server that talks to the AltaVista site. The first one is to retrieve all the results at once. Usually the users clicks on each link to browse the results hits. In our case we want to retrieve all the results as one big HTML document. The second reason is to separate the AltaVista access from the rest of the application. The advantage of this approach is that Altavista performs the searchers so it is basically a free text search component. The big disadvantage in case you don't know the API, is that you have to guess the results format and then transform it to IVL.

This process of "HTML scrapping" can be tedious and very expensive to maintain because the API and the layout structure of the HTML result set can change without any notice.

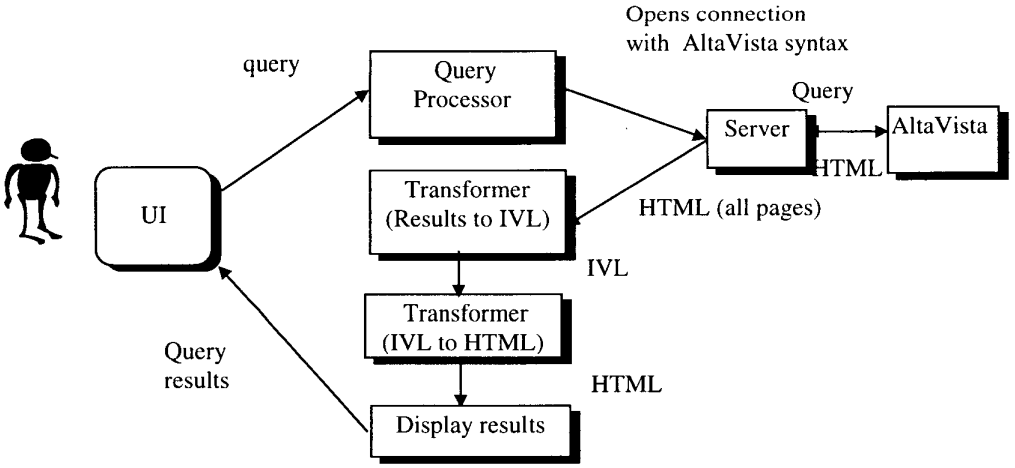

Figure 6. SAlta architecture

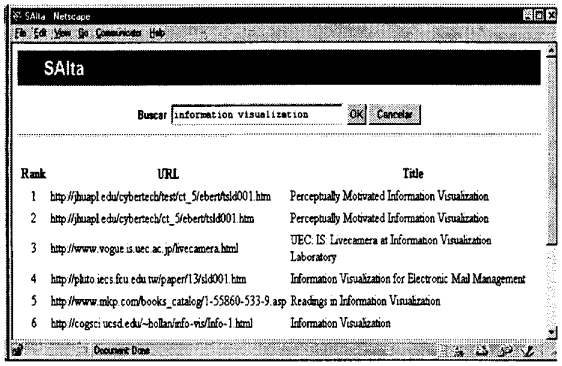

Figure 7. SAlta query results

Conclusions and Future Work

We have presented a model and a software architecture for searching and visualizing answers in web retrieval with emphasis in the visualization process. We included a markup language as part of our model. We know that IVL is not complete for industrial applications but is a starting point for research and prototype development.

With so many systems and tools for web retrieval, our model defines a way to share the output of those so that the visualization is more independent and therefore an option of several to the user.

Our software architecture could be used to implement knowledge management solutions where information is in different formats in different places. The model can also be used to implement previews and overviews interfaces for digital libraries.

Future work includes experiments with different hardware devices (e.g. cell phones, personal assistants, etc.) and integration with different information retrieval and visualization products that are available in the market.

From the usability and human interaction point of view we plan to make experiments to test theses ideas using different search engines and different visualization metaphors.

References

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

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

[3] Jacques Bertin, *Semiology of Graphics* (Traducción.), Univ. of Wisconsin Press (1978).

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

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

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

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

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

[9] 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).*

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

[11] Inktomi Portal Services, *IDP User's Guide*, Inktomi Corp. (1999).

[12] Daniel O'Leary, "Enterprise Knowledge Management", *IEEE Computer*, Vol. 31, No. 3 (March 1998).

[13] Oracle 8*i* *inter*Media Text Reference Manual, Oracle Corp., Redwood Shores, CA (1998).

[14] Marc Sebrechts, Joanna Vasilakis, Michael Miller, John Cugini, Sharon Laskowski, "Visualization of Search Results: A Comparative Evaluation of Text, 2D, and 3D Interfaces", *Proceedings of SIGIR '99*, Berkeley, CA (1999).

[15] Mary Shaw and David Garlan, *Software Architecture*, Prentice-Hall, Upper Saddle River, NJ. (1996)

[16] Edward Tufte, *The Visual Display of Quantitative Information,* Graphics Press, Cheshire CT (1982).

[17] Edward Tufte, *Envisioning Information,* Graphics Press, Cheshire CT (1990).

[18] W3C, "Extensible Markup Language (XML)" http://www.w3.org/TR/1998/REC-xml-19980210

[19] W3C, "Resource Description Format" http://www.w3.org/TR/PR-rdf-schema/ .

[20] W3C, "Handheld Device Markup Language Specification" (http://www.w3.org/TR/NOTE-Submission-HDML-spec.html).

[21] W3C, "NaVigation Markup Language (NVML)" (http://www.w3.org/TR/NVML).

[22] Sherlock (http://www.apple.com/sherlock/).

[23] 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).

[24] Nabil Adam, Vijayalakhmi Atluri, and Igg Adiwijaya, "SI in Digital Libraries", *Comm. Of the ACM*, Vol. 43, No. 6 (June 2000).

[25] Wireless Application Protocol (http://www.w3.org/Mobile/).

[26] Tim O'Reilly. Keynote Address, JavaOne Conference, San Francisco, CA (2000).