Personalized Information Structures II: Hyperstructure Hotlists

Scott R. Tilley Walter M. Lamia

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
E-mail: {stilley, wml}@sei.cmu.edu

Abstract

This paper describes ongoing research into the use of a domain-retargetable reverse engineering environment to aid the structural understanding of large information spaces. In particular, it presents follow-on work on the use of the environment in the documentation and hypertext domain. This programmable environment has been integrated with a popular World Wide Web browser to support hyperstructure hotlists*: an approach to managing link complexity, organizing conceptual themes, and aiding Internet navigation through the use of multiple virtual webs.*

Keywords: hyperstructure, reverse engineering, World Wide Web.

1 Introduction

The explosive growth of information sources on the Internet, particularly those accessible via the World Wide Web (WWW), has re-awakened interest in a classic hypertext problem: How to avoid the "lost in hyperspace" syndrome due to disorientation caused by a tangle of referential links in a hypertext web. However, the WWW presents several unique cognitive problems—problems that affect all classes of users, from the novice to the seasoned expert.

For the novice, the amount of information available on the WWW can be overwhelming. With little guidance, "surfing the web" is analogous to turning someone loose in the U.S. Library of Congress without any kind of organized cataloging system. Novices need a starting point to categorize what is available in some kind of taxonomic hierarchy, and to begin exploration of the info-terrain. A large number of these starting points are available on the WWW, which is itself a problem, because each one presents the view favored by the index's author. These views will certainly be influenced by the authors' purposes or interests, which may be academic, industrial, commercial, political, social, recreational, or almost anything else. Furthermore, there is almost no way to know what the biases might be, so the user is subject to a great deal of unintentional or deliberate indirect influence.

Expert users develop their own personal research techniques, usually employing one or more favorite indexing and directory servers, but this quickly leads to overloaded indexing. The problem is, what makes a page interesting is highly dependent on the context in which the user was working at the time, and this context continually evolves. Preserving the semantics of that context, which makes the information meaningful to the user, becomes more and more important as a variety of different, unrelated information is accumulated.

The use of *hyperstructure hotlists* is proposed as one solution to this problem. Hyperstructure hotlists are graphical representations of virtual webs. They are supported through the integration of two existing tools: Rigi,[1] a domain-retargetable reverse engineering environment used to aid the structural understanding of large information spaces; and Netscape,[2] a popular WWW browser.

The goal of the integration effort is to provide users with structuring mechanisms for conceptually clustering web artifacts beyond that provided by simple hotlist and bookmark mechanisms. The graphical paradigm provides a complementary navigation mechanism that, when used in conjunction with existing URL hotlinks, directly addresses one of the major problems with the WWW: finding and organizing information of interest to each individual user.

The next section discusses the theoretical foundations of hyperstructure hotlists. Section 3 outlines the support provided for hyperstructure hotlists through the integration of Rigi and Netscape. Section 4 illustrates the use of hyperstructure hotlists. Section 5 summarizes the paper and discusses future work.

2 Hyperstructure hotlists

When one attempts to understand a large body of information, the overall structure of the information space is just as important as the inner structure of any single artifact—if not more so. This is especially true when the number of artifacts in the domain is much larger than the size of each artifact. This process can be termed *hyperstructure understanding* (HSU): identifying artifacts and understanding their structural relationships in complex information webs. HSU is an objective, not a well-defined process [2]. The prefix *hyper* is used to distinguish HSU from the in-the-small activity of understanding the internal structure of any single artifact; the focus is the analysis of overall system structure.

Existing WWW browsers provide limited functionality to address the HSU problem. A *hotlist* (bookmark) mechanism is typically used to store a semi-ordered collection of links to WWW pages of interest. This mechanism is useful for a small set of links, but it provides no other structuring mechanism. Morevover, few browsers enable users to associate much context with the entries in the hotlists. As the hotlist grows, it quickly becomes unwieldy. Users forget why a certain page is referenced from their hotlist.

More importantly, there is no way of conceptually clustering links to represent common themes (other than authoring HTML documents). Some browsers provide a hierarchical tree-structured hotlist mechanism which allows users to collect entries into meaningful taxonomic groups. However, unless the user puts duplicate entries in the file to associate the same location with two or more concepts, there is no way to build multiple semantic threads through the hotlists. In addition to the complexity this introduces, duplicate entries are highly undesirable because of the configuration management problem.

An example of a multiple-threaded subject index is the Yahoo directory at http://www.yahoo.com. It presents a main index hierarchy, but the index has many cross-references embedded in it. For example, the main category "Computers" has an entry "Computer Science", which actually is a cross-reference to another branch, "Science: Computer Science." While Yahoo represents a good example of the semantic indexing function that is desired, it achieves this only by extraordinary manual effort and is still based on the judgment of the proprietors, similar to what professional librarians do to create catalog entries for books and other publications. Additionally, Yahoo presents a purely textual view of

[1] In this paper, "Rigi" refers to version V of the Rigi reverse engineering environment [1].

[2] Netscape Navigator version 1.1N.

the data base to users; it does not capitalize on the capabilities of graphical user interfaces that can present a richer view of the spatial interconnection topology of the underlying information space.

Hyperstructure hotlists provide two valuable enhancements to conventional WWW indexing tools. The first is a powerful and easy-to-use method for constructing arbitrary indexing schemes which capture the semantics of the users' interest in any particular set of Web locations. These semantic networks can be arbitrarily large and arbitrarily complex, permitting the user to build as many threads through the information space as desired. The second is a graphical interface that can display threaded structures in two-dimensional space with functions for expanding and collapsing sub-nets, and for creating and maintaining cross-reference links among leaf nodes and sub-nets. This interface employs shape and color to convey details about semantic concepts which would be cumbersome in a purely textual presentation.

3 Supporting hyperstructure hotlists

This section discusses the support of hyperstructure hotlists. The Rigi reverse engineering environment is discussed. The modeling of the HTML domain is described. The integration of Rigi with the Netscape WWW browser is outlined.

3.1 Rigi

Hyperstructure hotlists as described in Section 2 are supported through a prototype integration of the Rigi reverse engineering environment and the Netscape WWW browser. The current version of Rigi is the result of a three-step reengineering process that has been underway since 1992. The first step was to make the central component of Rigi (`rigiedit`: a graphical, hypertext-oriented, multi-window hyperstructure editor) programmable through the addition of a Tcl-based [3] scripting language [4]. The second step was to make the user interface customizable [5]. The third step was to integrate a layered modeling paradigm [6] into the environment.

Rigi is a prototype realization of the PHSE:[3] an architecture for a meta reverse engineering environment [7]. It provides a basis upon which users construct domain-specific reverse engineering environments. It is instantiated for a particular application domain by *specializing* its conceptual model, by *extending* its core functionality, and by *personalizing* its user interface.

In addition to its use in aiding program understanding for software maintenance, Rigi has also been retargeted to support hypertext through the creation, representation, and structuring of online documentation [8]. This earlier work presented the natural evolution of document structure, from linear text to hypertext to structured hypertext to personalized information structures. The use of personalized information structures was illustrated by creating personalized hypertext versions of LaTeX documents. In that case, the document's internal structure was of importance. For hyperstructure hotlists, it is the inter-document structure that we are interested in.

3.2 Modeling HTML

One part of retargeting Rigi is to provide a model of the new application domain. This is done using the layered modeling facilities provided by the PHSE framework. The three-level model consists of a low-level data model representing the atomic elements gathered from the implementation domain (such as HTML anchors and protocols); a high-level knowledge model representing concepts from the application domain (such as the structural relationships in and between documents); and an graphical information model used for hyperstructure navigation, analysis, and presentation. It should be

[3]The acronym PHSE, pronounced "fuzzy," stands for *P*rogrammable *H*yper*S*tructure *E*ditor.

noted that the HTML domain model used is not a complete representation of the HTML language; only the structural aspects of HTML required to support HSU are provided.

The relational data model is the foundation upon which the conceptual model is constructed. The conceptual model represents domain knowledge; it acquires its semantics when the abstract conceptual model is further refined to reflect a particular application domain. A semantic network information model is used to represent selected artifacts of the subject system; this layer forms the core of the graphical interface to the underlying information space.

The data model is implemented as a store of binary relations, similar to the mechanism used by Beynon-Davies *et al* [9]. The binary relations that represent the database are stored in one or more files as a series of RSF (*R*igi *S*tandard *F*ormat) triples. Each triple is of the form *type subject object*. The interpretation of this relation is straightforward: a directed relation of the type 'type' is asserted between the *subject* and the *object*. Since every n-ary relation can be expressed as a conjunction of $n+1$ binary relations [10], the RSF mechanism is sufficient to store the data required by the reverse engineering environment. For example, the RSF triple

```
ftp   file:/home.html   ftp.sei.cmu.edu
```

might indicate the existence of an HTML relation using the *ftp* protocol between the local file */home.html* and the host *ftp.sei.cmu.edu*.

The language Telos [11] is used in the knowledge modeling layer. Telos was selected over other conceptual modeling languages because it is more expressive with respect to attributes, it is extensible through its treatment of metaclasses, and it has already proven successful in other application domains. For example, it has been used to provide a structural framework for an authoring-in-the-large hypertext system [12]. The primitive units of Telos, individuals and attributes, have a direct mapping to the primitive units of hypertext, namely nodes and links. Moreover, an RSF representation of the knowledge base can be used to represent all Telos propositions, from metaclasses to tokens, in a uniform manner.

The information model used is a special-purpose semantic network, represented as an attributed graph. In the model, both artifacts (represented as nodes) and relations (represented as arcs) are specializations of the same parent class. Modifications to the repository occur through the insertion or deletion of artifacts and the manipulation of relations.

The information model consists of four objects: webs, nodes, links, and attributes. A web is a subset of the entire knowledge base that is related in some fashion. It is composed of typed nodes representing artifacts and typed arcs representing relations. Each node has a set of incoming arcs and a set of outgoing arcs. A node represents an artifact in the target domain. Links between nodes represent relations between artifacts. Attribute/value pairs can be attached to nodes or links, permitting the organization of nodes and links into subgraphs and webs. For example, subsets of objects may be extracted from large graphs using filtering mechanisms based on attribute predicates.

Interrelated webs of objects form the cornerstone of the information model. They are manipulated using `rigiedit`. Portions (or all) of a web are viewed by the user as a *neighborhood*. A neighborhood is simply a collection of artifacts that are immediately accessible from the current perspective. It is graphically represented in the editor as a single window containing the artifacts. Artifacts can exist in any number of neighborhoods simultaneously, since neighborhoods are simply dynamically computed perspectives of the underlying knowledge base. This permits multiple, coexisting views of the information space.

3.3 Integrating Netscape

There are several WWW browsers currently on the market, with more becoming available every day. Although it would be possible to provide WWW browser functionality by extending Rigi through the construction of the appropriate widgets, it was thought that integrating with one of the more successful existing WWW browsers would be better. The current generation of WWW browsers do not yet lend themselves well to integration, although recent additions such as Sun's Hot Java are changing that situation. In the end, it was decided to use Netscape.

The Netscape Navigator is one of the most popular WWW browsers currently available. The most recent version provides a rudimentary application program interface (API) that enables it to be remotely controlled by another application. The Netscape API is different for each of its supported platforms. For Microsoft Windows, DDE and OLE 2.0 are used; for the Macintosh, Apple Events are used; for X-Windows, Xt actions are used. The prototype integration of Rigi with Netscape is under UNIX, so the X-Windows API is used.

Like many X-Window programs, Netscape provides an application default file (`Netscape.ad`) that may be used to customize the user interface. To remotely control Netscape under X-Windows, the following command is issued:

netscape -remote *Xt-action*

where *Xt-action* represents the desired action for Netscape to perform, and is the same name as the resource name given in `Netscape.ad`.

It is also necessary for Rigito be able to gather data from the HTML documents. In particular, tags such as <A> representing anchors are essential: they represent the source or destination of a hypertext link and must be captured in the information model. This is done through the use of `html2rsf`, a Tcl procedure that translates HTML to RSF. `Html2rsf` is really two separate programs. The first is a Perl script that parses an HTML document and extracts tags of interest. These are then processed by a Tcl script that converts these HTML tags into their RSF counterparts. In this way, the data representing outgoing arcs from an HTML document, as indicated by <A HREF= ...>, are gathered and modeled using RSF and the appropriate protocol (as defined in the conceptual model). This process is illustrated in Figure 1.

4 Using hyperstructure hotlists

This section illustrates the use of hyperstructure hotlists through the retargeting of Rigi and the integration of Netscape into the environment. The HSU process manipulates three types of "artifacts:"[4] (1) data: the factual information used as a basis for reasoning, discussion, or calculation; (2) knowledge: the sum of what is known and represents the body of truth, information, and principles acquired; and (3) information: the communication or reception of knowledge obtained from investigation, study, or instruction. Based on these definitions, we can identify three canonical reverse engineering operations: (1) data gathering; (2) knowledge organization; and (3) information navigation, analysis, and presentation. Therefore, the retargeting consists of three steps: (1) gathering data via structural feature extraction, (2) organizing knowledge by specifying a domain model, and (3) navigating, analyzing, and presenting information by extending the editor.

The first step is the construction of the knowledge layer through the specification of a Telos conceptual model representing the HTML domain. A graphical representation of this schema is shown in Figure 2. As stated in Section 3.2, the model does not describe all of HTML, just those features needed for illustration purposes. Nodes in the HTML domain model represent artifacts (such as documents), while links represent relations (HTML protocols) between these artifacts.

[4] The definitions used here are in accordance with Webster's online dictionary.

The following HTML fragment (from the file sei.html):

```
<h1>Contacting us

<p>If you have questions about the <a href="http://www.sei.cmu.edu">SEI</a> home page,
please send send e-mail to our <a href="mailto:webmaster@sei.cmu.edu">webmaster</a>.
```

is converted to the following RSF stream using the `html2rsf` program:

http	sei.html	http://www.sei.cmu.edu
mailto	sei.html	mailto:webmaster@sei.cmu.edu

Figure 1: Converting HTML to RSF

Artifact	**Icon representation**
web	
strand	
document	
host	
user	
newsgroup	

Table 1: HTML artifacts and their icons

The second step is the construction of the data layer through the gathering of HTML artifacts. There are several ways this can be done. For example, because the RSF formalism is so simple, an RSF representation of an HTML web can be constructed by hand. However, the most common method is to use the `html2rsf` translator to dynamically capture the artifacts and relations of interest from user-selected HTML documents. Typically, the HTML documents will be those currently accessed using the Netscape WWW browser, but other tools might also be used off-line to construct hyperstructure hotlist representations of larger webs (as discussed in Section 5).

The third step is the manipulation (navigation, analysis, and presentation) of the information layer, which is a graphical representation of the gathered artifacts from the source document. The HTML artifacts are represented in the editor by their respective icons, as shown in Table 1. The icons used to represent HTML artifacts, like almost all aspects of Rigi, are completely under user control.

Traversing a web involves moving from one neighborhood to another. All traversal operations are based on the currently selected set of nodes. By default, double-clicking on a single node invokes the predefined procedure `rcl_open_node`, which causes a new neighborhood to be entered by following all outgoing arcs of the current outarc type. The routine is usually replaced by users to perform actions specific to a particular application domain, or specific to a node type. More sophisticated web traversals are possible using the widget shown in Figure 3. Webs may also be edited (adding, deleting, or altering artifacts) and/or analyzed. General-purpose widgets and routines are provided for all of these operations, but they are usually augmented by the user during retargeting.

Figure 4 contains a screen dump of the environment in use. The window at the top (with title PHSE) is the main control widget. It contains a variety of action

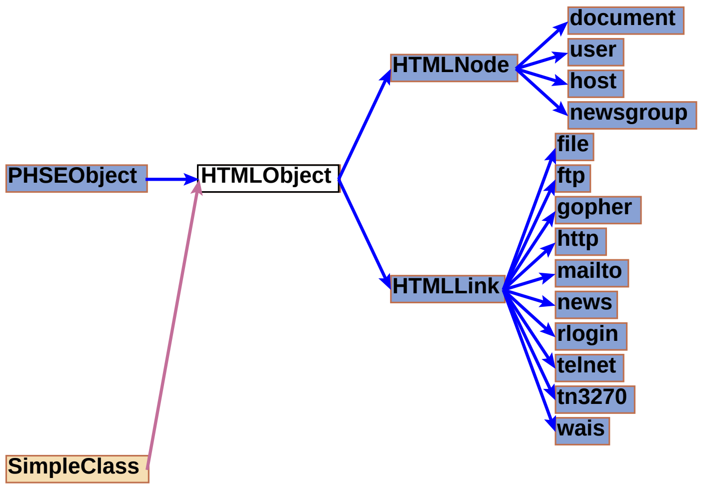

Figure 2: HTML conceptual model

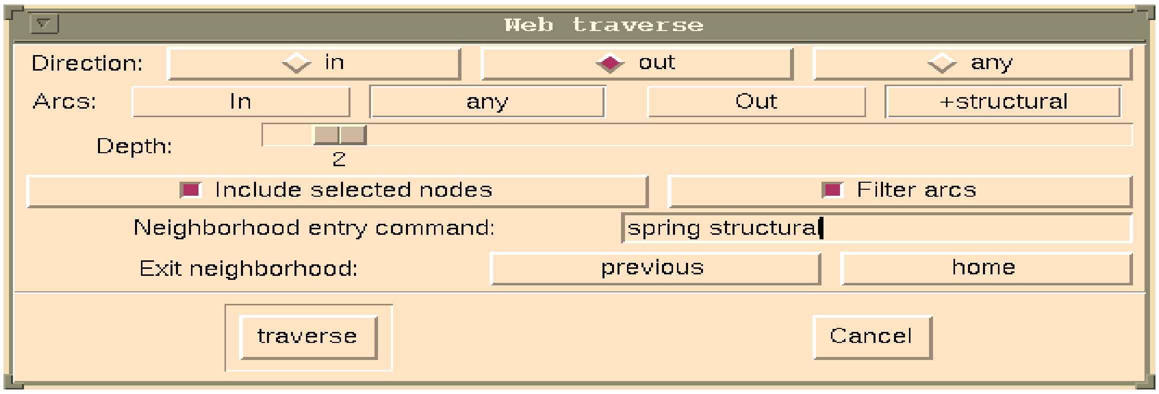

Figure 3: Web traversal widget

items, accessible through hot buttons, cascading menus (not shown), or directly through the command line. The window at left (with title General:2) is a view of the reference hyperstructure hotlist. A special layout algorithm that models the neighborhood as a physical system of springs and anchors has been used to display the hotlist in such a way as to enhance the structural relationships between artifacts. For example, one can see that two `mailto` artifacts are shared by both the *people* and *CMU* webs. Although not discernible from the black and white image, the nodes and arcs are color coded to aid understanding; they are also scaled in Figure 4 to fit in the window frame.

The rightmost window in Figure 4 is an instance of Netscape; it is displaying the contents of the HTML document http://www.sei.cmu.edu. This was accomplished by the user selecting the iconic representation of this URL in the Rigi window at left (as reported by the message at the bottom of the window) and pushing the right-most hot-button . This button is attached to a simple user-defined Tcl procedure that causes Netscape to access the selected HTML nodes. If just one artifacts is currently selected, the current instance of Netscape is used; if more than one artifact is selected, a new Netscape session is initiated for the second and subsequent artifacts.

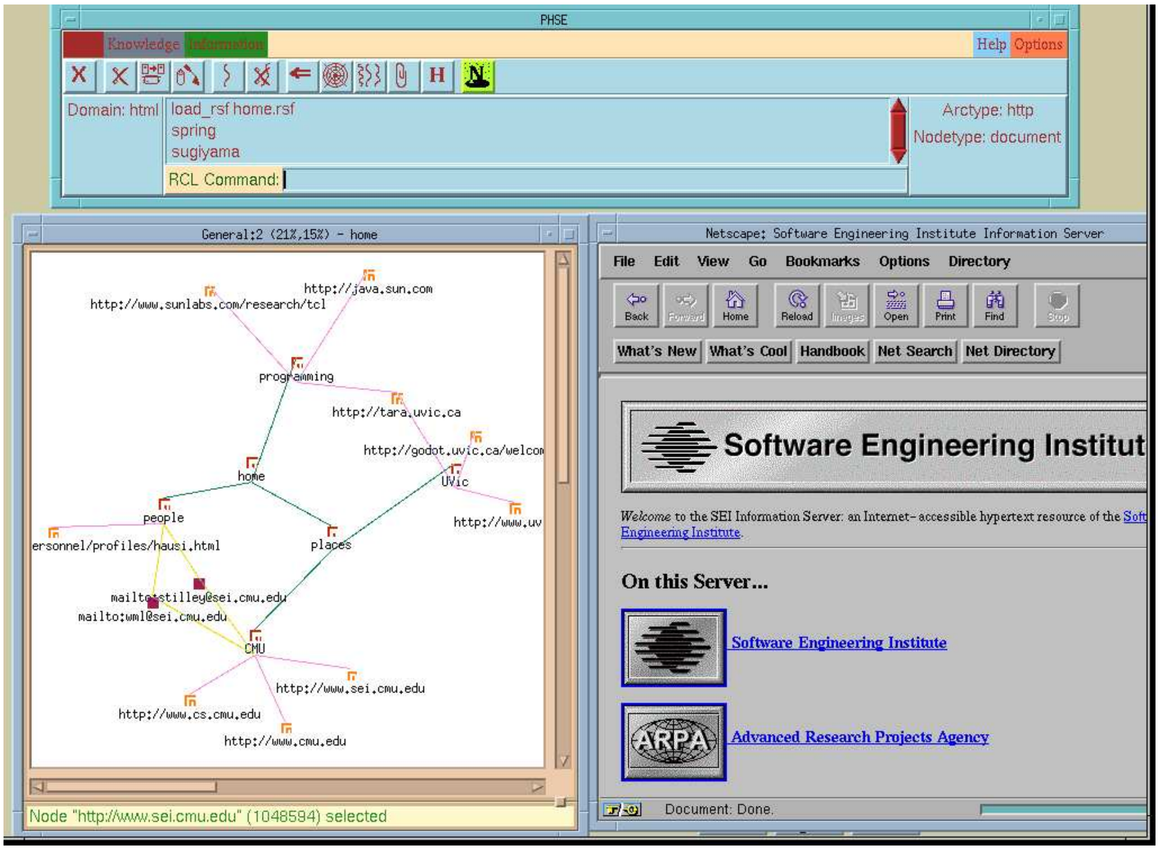

Figure 4: Using hyperstructure hotlists

5 Summary

This paper presented hyperstructure hotlists, an approach to managing link complexity, organizing conceptual themes, and aiding WWW navigation through the use of multiple virtual webs. A proof-of-concept integration of the Rigi reverse engineering environment and the Netscape WWW browser provides the support for hyperstructure hotlists. The result is an improvement over the simple hotlist mechanism provided by most browsers and an alternate perspective of the underlying information space.

Hyperstructure hotlists are constructed by retargeting Rigi to support HTML. A simple relational model is used to represent HTML at the data level, while a scripting language is used to enable the gathering of this data from HTML documents, and to communicate with Netscape. A conceptual modeling language is used to organize knowledge and reduce cognitive overhead concerning the (potentially very large) information space of the WWW. A graphical representation of the semantic network used to model the information artifacts facilitates the interactive navigation, analysis, and presentation of the information space.

In summary, the information space is constructed out of data artifacts, structured through the construction of conceptual models, and interactively explored in a hypertextual manner by representing neighborhoods as (parts of) semantic networks. The approach encourages the integration of additional tools into the reverse engineering environment, building on other research work in an evolutionary manner.

5.1 Future work

This work has opened up several areas of future research, including refining the integration between Rigi and Netscape, visualizing larger URL databases, and integrating other tools into the environment.

The integration of Rigi and Netscape is by no means complete; further work is needed if the two tools are to become truly cooperating processes. It would be interesting to integrate Rigi with other WWW browsers. A prime candidate is Hot Java, since it provides a programmable interface through the Java language.[5] As there is already work underway on a Tcl to Java interface, the integration between Rigi and a Tcl-aware browser would be much easier, and much more functionally complete, than the current Netscape interface. The use of more advanced browsers such as Hot Java will also open the door to more interactive HTML browsing, through the use of "applets."

It would be interesting to integrate existing WWW search and catalog tools, such as Lycos and Yahoo, into the current environment. This would enable the user to navigate a graphical representation of potentially very large WWW information spaces. The visual representation of the underlying webs might uncover interesting "hot" spots. Tools that verify link integrity could also be used to "clean" webs of dead links. The consolidation of separate webs through the use of the web editing and navigation mechanisms provided by Rigi would also permit one to view incoming links to selected HTML documents, something that Nelson's Xanadu [13] system aspired to.

There is a growing interest in the use of intelligent agents to aid users in finding, filtering, and filing information of interest to them on the WWW. The integration of such a tool into the environment would enhance the use of hyperstructure hotlists by facilitating their dynamic construction based on specific, user-specified, search criteria.

[5] It was recently reported that Netscape has licensed the Java language from Sun, so that Netscape will also be Java-enabled.

For example, there are now several commercial packages available that analyze HTML documents and automatically extract their semantic content. Such tools, used alone or in conjunction with the aforementioned intelligent agents, would also increase the usefulness of hyperstructure hotlists by refining the webs constructed by the user.

Natural language processing tools could also be used for automatic semantic thread generation. This is a rapidly advancing technology which is becoming fast enough and accurate enough to automatically construct useful semantic networks from a large textual information base. The synthesis of these tools with graphical tools such as Rigi shows much promise for users who are rapidly being flooded by the burgeoning overabundance of information on the WWW.

Acknowledgments

The genesis of this work began while the first author was at the University of Victoria. The pioneering work on personalized information structures was shared by the co-authors of [8]. The referees' comments are also greatly appreciated.

References

[1] K. Wong, B. D. Corrie, H. A. Müller, M.-A. D. Storey, S. R. Tilley, and M. Whitney. Rigi V user's manual, 1994. Part of the Rigi distribution package.

[2] A. O'Hare and E. Troan. RE-Analyzer: From source code to structured analysis. *IBM Systems Journal*, 33(1):110–130, 1994.

[3] J. K. Ousterhout. *An Introduction to Tcl and Tk.* Addison-Wesley, 1994.

[4] S. R. Tilley, H. A. Müller, M. J. Whitney, and K. Wong. Domain-retargetable reverse engineering. In *Proceedings of the 1993 International Conference on Software Maintenance (CSM '93),* (Montréal, Québec; September 27-30, 1993), pages 142–151. IEEE Computer Society Press (Order Number 4600-02), September 1993.

[5] S. R. Tilley. Domain-retargetable reverse engineering II: Personalized user interfaces. In *International Conference on Software Maintenance (ICSM '94),* (Victoria, BC; September 19-23, 1994), pages 336–342. IEEE Computer Society Press (Order Number 6330-02), September 1994.

[6] S. R. Tilley. Domain-retargetable reverse engineering III: Layered modeling. In *Proceedings of the 1995 International Conference on Software Maintenance (ICSM '95),* (Nice, France; October 16-20, 1995), May 1995. To appear.

[7] S. R. Tilley. *Domain-Retargetable Reverse Engineering.* PhD thesis, Department of Computer Science, University of Victoria, January 1995. Available as technical report DCS-234-IR.

[8] S. R. Tilley, M. J. Whitney, H. A. Müller, and M.-A. D. Storey. Personalized information structures. In *Proceedings of the 11th Annual International Conference on Systems Documentation (SIGDOC '93),* (Waterloo, Ontario; October 5-8, 1993), pages 325–337. ACM (Order Number 6139330), October 1993.

[9] P. Beynon-Davies, D. Tudhope, C. Taylor, and C. Jones. A semantic database approach to knowledge-based hypermedia systems. *Information and Software Technology*, 36(6):323–329, 1994.

[10] R. Kowalski. *Logic for Problem Solving.* North-Holland, 1979.

[11] J. Mylopoulos. Conceptual modelling and Telos. Technical Report DKBS-TR-91-3, Department of Computer Science, University of Toronto, November 1991.

[12] R. Sobiesiak. A hypertext authoring framework based on ceonceptual modelling. Master's thesis, University of Toronto, 1991.

[13] G. Wolf. The curse of Xanadu. *Wired*, pages 137–202, June 1995.