Information Engineering on the World-Wide Web: Drawing Analogies with Software Engineering

Rob Walker

Imager Laboratory
Department of Computer Science
University of British Columbia

Abstract

The advent of the World-Wide Web as a distributed hypermedia database is forcing a departure from the traditional views of static, published information. The *perception* that the information here is easily alterable is creating, and will continue to create, the demand for continual change to this information. However, many issues such as maintaining visible structure for navigability and revision, versioning, and restructuring are causing this perception to be unfounded, particularly as the amount of information necessary to maintain increases. We draw on the experiences and observations gained in the field of software engineering to show us some solutions and limitations to these problems which are being encountered now, and which we believe must get worse.

1 Introduction

Traditionally, (non-verbally) providing information has taken the form of the publication of books, newspapers and magazines. Incremental updates could not be easily accommodated; entire new editions of a publication would be required to deliver the most trivial change.

The advent of the World-Wide Web (WWW) has enabled a marked departure from this static view of information distribution. Although at present, typical WWW pages are static entities corresponding to physically-present files at their source server, there is widespread recognition of the limitations inherent in such a scheme. The Hypertext Transfer Protocol (HTTP) [bern96], Hypertext Markup Language (HTML), and Common Gateway Interface (CGI) now provide an *interface* which may allow the *reuse* of existing information en-masse by *revising* only the portions of it which change. The treatment of HTTP requests for information as database queries will greatly alter the production and consumption of information. Examples of such "on-the-fly" generation of WWW pages already abound; the most common and most easily accessible are the WWW pages generated by commercial on-line search engines such as Lycos. The goal of creating non-static generation mechanisms seems simple: minimize the cost and impact of change while maximizing the relevancy of the delivered information — deceptively simple.

Even with the use of databases, we claim that as the size and number of pages and documents in *local web regions*[1] increases, maintenance tasks will become increasingly difficult and the frequency

[1] This term is chosen to represent a set of WWW pages and/or CGI scripts which may be edited, maintained, etc. by an individual or organization, as opposed to external WWW pages to which this local web region may contain *links*. This term is seemingly synonymous with *web locality* of Pirolli *et al.* [piro96a]; however, we feel the latter is too easily confused with the concept of a Uniform Resource Locator (URL) or WWW page.

of incorrect, misleading, out-of-date, or contradictory information will likewise increase. Drawing analogies with software engineering (SE) will yield some insight on solutions and the complexity of these problems.

2 Problems in Using the World-Wide Web

2.1 Navigation

We have observed first-hand, the truth behind this quote:

> The core of hypermedia's power lies in the complex networks of links that can be created within and between documents. However, these networks frequently overwhelm the user and become a source of confusion. [utti89]

Of course, knowledge about hypermedia systems has increased markedly since 1989, but this quote remains valid nonetheless. Other authors have also noted this "lost-in-hyperspace" problem (cf. [bota92]). In navigating through large, commercial web regions, we have often become lost because of two main factors:

1. a lack of apparent order, which is not to say that there is not one present just that it is not *visible*, and
2. the necessity of following links through the region, be they embedded in hypertext prose, or be they via hierarchical indices reflecting the file structure in which they are stored; in turn, this yields two other problems:
   (a) the presence of cycles can be very disturbing when one believes one is moving away from a starting point, and
   (b) sometimes "you can't get there from here" because no path leads to the page from the current one.

There are two forms of navigating the WWW: *searching* and *browsing*. The difference between the two has already been recognized in the literature (cf. [piro96b]). Searching consists of seeking out specific information often based on keywords. Keyword search is a heavily-used form of searching with many commercial search engines available for the WWW as a whole, and other types of *agents* such as *spiders* which are suitable for use on a local web region (cf. [cheo96]); people tend to feel that this problem is solved. However, as is often the case with natural language tasks, search engines tend to fail if there is only a semantic relationship between a WWW page and the precise keyword(s) being searched for. This can be partially mitigated by the use of thesauri and additional, secondary information repositories; however, at some point, *understanding* of both the task at hand and of the information being processed will be required. Until the field of artificial intelligence fulfills its goals, this will force human intervention at some point.

Browsing consists more of looking at WWW pages and following hypertext links to see where they lead, sometimes because one is reading about a particular path of information, and other times out of shear curiosity (the latter being colloquially known as *surfing*). It seems that this sort of navigation should really be simpler, somehow, but the above-mentioned difficulties belies this. In observing students navigate course web regions, we have noted that they often remain unaware of the structure which has been provided for them in these regions; even after a couple of months of consistently having to use a course web region, they often do not know that particular, large amounts of information is present. This seems to be due to the lack of a *mental model* of the global structure of these regions, because after that structure is explained to them, they then are able to locate the information that they require. (The *understanding* of the structure and the efficiency with which it is *used* should not be confused, however.)

Some of this difficulty can be obviated with the use of *visualization* techniques (and the external application of structure), but there are limitations to this, as will be explained in Section 3.5. Botafago *et al.* [bota92] state that one solution to the lost-in-hyperspace problem is "improved authoring to create more comprehensible structures"; we address this in the following Subsection.

2.2 Authoring and Maintenance

Currently, most *authoring* on the WWW consists of the creation of relatively static, semi-monolithic pages with fragile connectivity to other WWW pages. Many books in the popular literature and sites on the WWW are devoted to this task — especially notorious are the ones encouraging the expenditure of vast amounts of effort into particular pages, often declaiming the shortcomings of HTML as inhibiting artistry (cf. David Siegel's pages `http://www.dsiegel.com/`). While we do not deny the desirability of such pages, we believe that they will be limited in applicability to such areas as homepages. We may liken the production of such pages to programming efforts in past years: people spent vast amounts of time getting their code to run as fast as possible, not being concerned about maintainability since "once written, they would be correct for all time" (see [parn94] for a refutation) and the authors would be the only ones making any alterations anyways...

Reality is not so cozy. Even in 1990, the necessity for relational database queries, reuse, and on-the-fly construction were observed [watt90]:

> The hyperedges and nodes are manipulated through a set of operators to compose new nodes and to instantiate new links dynamically, resulting in transient hypergraphs. These transient hypergraphs are virtual structures created in response to user queries, and lasting only as long as the query session.

Also, the problem of maintaining computer documentation, for example, is a well-known and unsolved one (cf. [same94]); we argue that the same is true for prose in general.

In authoring and maintaining a moderately-sized local web region (100+ pages), we have encountered many problems:

1. few pages are really static in their contents and many are very non-static,

2. the links between pages are not static: pages move around a lot,

3. small alterations can lead to disproportionately large amounts of work, and

4. the provision of externally visible structure is difficult.

Some of these problems are due to a lack of good tools in part. Building pages on-the-fly by the use of relational databases will eliminate some of the difficulties: pages will have *variables* embedded in them which will be instantiated from database queries[2] when provided to the user; the "virtual structures for dealing with changing information" of Halasz [hala88] is one such example. As long as all the pages have the necessary variables embedded in them, that these variables are altered as their semantics changes, and as long as the database is updated correctly, everything

[2] The question of computing cost might require that pages only be updated from the database periodically, and there remains the question of server-side vs. client-side instantiation.

will work fine — but these are rather large provisoes. This is highly reminiscent of "accidental vs. essential" problems encountered in SE [broo87]. These problems have also been recognized in the context of hypermedia [tomp89]:

> For example, a database designer might intend to place an edge labelled help from every node in some set to a particular help page: As new nodes are added to the set, such an edge might be accidentally omitted; uniformly changing the label to note requires many separate updates. This lack of one place for every fact manifests itself in this model by three related shortcomings. While some implemented hypertext systems overcome one or more of these shortcomings, it is the *model* that is inadequate.

These problems are also recognized in industry today: Intellecte/Business Web from Interleaf[3] ties "directly to the underlying information at its source — in the document management environment", i.e. a relational database management tool, and "by creating a dynamic, virtual web, link maintenance is dramatically reduced."

Tompa goes on to describe a model which he feels does not suffer from such shortcomings; however, we have seen the same phenomenon in SE too: using variables will solve our problems, structured programming will solve our problems, object-oriented programming will solve our problems, ... Our problems are still with us in SE, though; what remains here is to demonstrate more fully the connection between SE and *information engineering on the WWW* (IEW).

3 Parallels with Software Engineering

The use of terms such as "reuse" and "revision" alludes to a potentially powerful and deep connection with SE. Through drawing analogies between IEW and SE, we hope to gain a deeper understanding of the problems which face us in IEW, and take a huge leap forward by using the experiences already encountered in SE. But conversely, SE might also benefit from a new perspective which might be gleaned from IEW — this remains to be seen though.

3.1 Evolution

Parnas [parn94] has explained that software systems evolve: change is unavoidable for a software system unless that system is obsolescent. Likewise, computer documentation is known to evolve as the systems they describe themselves change [same94]. We claim and have shown some evidence in the previous Sections that WWW pages also evolve (or devolve), at least in certain circumstances; we extend this concept to claim that these circumstances are the rule in general, and not the exception. We have from Coney [cone96] as a first step:

> The domain of literary theory, inhabited by scholars of poetry, fiction, and belles lettres, may seem far afield from the world of computer documentation, peopled by technical writers, usability experts, and computer programmers. But from a rhetorical point of view, these two fields of study and practice appear more similar than not: both are concerned with producing manuals that convey information intended for specific purposes to an audience who is largely unknown to the author and who will use a manual under sometimes unpredictable circumstances and with unforeseen consequences.

The realm of more artistic prose is not immune to evolution. The author being a devotee of the classic fantasy novelist J. R. R. Tolkien, we are familiar with the many *years* of alterations

[3] `http://www.ileaf.com/ibwds.html`

(cf. [tolk88]) on even the most fundamental aspects of his opus *The Lord of the Rings* [tolk66]. This creation is widely held as a work of literary art (unlike many of its popular compatriots in the same genre) and yet it was not immune to the necessity for change. Nor was it immune to the problems stemming from such change: there are recognizable inconsistencies in the finished work, although minor ones, to which even Tolkien himself admitted.

One might claim that *The Lord of the Rings* is an exception and not the rule, since it was a partial view into an entire world constructed by Tolkien and that the shear volume of detail does not commonly occur in other works. While the latter is true, even works with a paucity of "factual" information can undergo a lot of change. People wish to write more works in the same context as the original at times, and in the future, people are quite likely to want to change such works as much as copyright laws will allow.

The point is: evolution in prose is already with us, and will only become more prevalent as people perceive that it is easier.

3.2 Forward Engineering

Forward engineering is generally described in terms of three major tasks:

1. *requirements*,
2. *design*, and
3. *implementation*.

A traditional, structured view of prose authoring possesses similar, parallel tasks:

1. figuring out what topics you want (or need) to discuss,
2. structuring the discussion, possibly in order of increasing complexity or temporality, and
3. filling in the details.

Just as the SE tasks are not necessarily purely sequential, neither are those of IEW.

Although these tasks are well-accepted now as being necessary for the proper development of maintainable software, this was not always the case; many complained that these sort of differentiable steps inhibited the "creative process." It became apparent that these steps were necessary as the size and complexity of the sotfware systems grew, and as the need for altering them grew too.

Much the same complaints are liable to be lodged against the need for these steps in IEW, that they are too bureaucratic, that they will inhibit the creative process. However, as the need for altering existing information resources grows, and as the size of those resources grows too, we will find that these steps (and their documentation) become more and more necessary. Botafago *et al.* have already recognized the importance of the structuring task [bota92]; they discuss the use of an iterative process of structural creation, drift from this structure as more details are filled in, structural redetermination, and structural refinement.

There is a general consensus in the need for properly structuring the discussion, and obviously the other two steps cannot be done without [maye85, maye95]. What does not seem yet to be recognized is the need for more formalization and documentation of these steps. It is not clear that the same level of formalization or even the same kinds of formalization as are found in SE will be necessary or sufficient in IEW; since the "code" really will be "self-documenting", it is possible that there will be some distinction. However, people can misinterpret the structure, include greater (or lesser) ambiguities, and introduce inconsistencies through a lack of understanding — and as the size of the text increases, these possibilities increase.

3.3 Reuse and Versioning

It might seem that the concept of version control is foreign to WWW authoring; Halasz [hala88] notes:

> A good versioning mechanism will allow users to maintain and manipulate a history of changes to their network. It will also allow users to simultaneously explore several alternate configurations for a single network.

The concept of version control and configuration management is already present in industry. Intra.*doc!*(TM) Management System from IntraNet Solutions, Inc.[4] is a commercial WWW configuration management tool aimed at the corporate market.

"Reuse" is what the WWW is really all about: by having a link to another WWW page, you are reusing that information resource. We can think of that link as a procedure call, and if the link is to an external web region, that procedure call effectively becomes one to a library routine; we can now see a similar problem between IEW and SE: the need to maintain library interfaces, and the semantics of those interfaces.

But reuse has other guises in IEW as well; manipulating the presentation of information based on the particular intended audience is a major goal; Lucarella and Zanzi [luca96] state:

> a strong separation of the structure from the content would allow many structures to be superimposed over the same set of information units or a unit to be shared among many nodes within a single structure.

Object-oriented documentation (distinct from documentation whose subject matter contains object-oriented concepts) is a concept which has been introduced [same94] in part because of the reuse potential for that documentation [rave94]. The isomorphism between object-oriented programming and the needs of computer documentation have also been noted by Weiss [weis94].

3.4 Reverse Engineering

Chilkofsky and Cross [chil90] define reverse engineering as "the process of analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction."

The reverse engineering parallels with IEW are probably the most prevalent today. All searching tasks can be viewed in this light. Keyword search is simply a lexical analysis tool; other forms of search attempt to analyze the semantic content of WWW pages based upon heuristics. For example, scatter/gather [piro95, piro96b] is a recently proposed technique for grouping WWW pages, and interactively, iteratively refining this structure; this is very reminiscent of cluster analysis of Hutchens and Basili [hutc85]. Pirolli *et al.* are also interested in extracting structures from the WWW [piro96a], which is a large part of the above definition of reverse engineering, and Hearst's TileBars [hear95] "partitions expository texts into multi-paragraph segments that reflect their subtopic structure."

We also have attempts at representing the semantic contents in a codified manner [cele95]; in conjunction with assertions about WWW page interrelationships, we could derive an analogue to reflexion models [murp95].

[4] `http://www.intranetsol.com/integration/intradoc/`

3.5 Visualization

Most WWW visualization "techniques" are (not-so-) fancy interfaces built on top of graph visualization techniques much as in software visualization; the WebBook and WebForager [card96] are the latest in this class, combining strict graph visualization with grouping and visualization of the actual WWW pages.

There is recognition in the WWW visualization area of the information overload problem inherent in graph visualization. Of course, this immediately led to the knee-jerk reaction that a third spatial dimension would help a lot [koik93], actually based upon Fairchild's pioneering work in software 3D visualization [fair88].

Hierarchical views have been proposed [mukh95] which are reminiscent of the pictures Rigi gives of software visualization (cf. [müll93]). There has also been an attempt at using the idea of context and focus to filter out extraneous information: the further away a piece of information is from the current point of focus the less important it is. Lamping *et al.* [lamp95] used a hyperbolic projection to produce this effect, while Koike [koik95] used fractal views. Koike's work was also used directly on viewing source code. Lucarella and Zanzi [luca96] have proposed a visual retrieval environment which they claim would allow the visualization of the "conceptual schema" by enabling the user to filter the amount of information to be displayed. Robertson *et al.* [robe93] promote their Information Visualizer which uses different visualization techniques for different underlying structures.

At an early stage, it was recognized that there was a need for context and orientation in hypermedia networks [utti89]. This led to the idea of providing visualization tools for the local connectivity to other WWW pages, pages already visited, and a path record. The latter two, at least, have already found their way into commercial WWW browsers.

Ball and Eick [ball96] use software visualization techniques for code version history, differences between releases, static properties of code, code profiling and execution hot spots, and dynamic program slices. Most of these would also be of interest in IEW visualization, where execution hot spots would become access hot spots. The linear pictures Ball and Eick use are very similar to Hearst's TileBars [hear95] although the latter is an attempt at visualizing semantic structure.

4 Structure

The central, underlying force in SE is structure: its maintenance, visibility and communication.

> An important part of designing a system is designing its structure: designers generally begin with high-level structural design and then, for each piece of the system, perform structural design before detailed design. An important part of understanding a system is understanding its structure: new readers of a system generally try to gain an appreciation of global structure before they look at any details. Global structure, in fact, provides the framework within which to examine and understand details. [ossh87]

Ossher outlines two criteria which a particular partitioning of a system must meet if it is to "promote understanding." These are:

local readability: system pieces must be understandable in near-isolation, and

global structural visibility: the way in which the pieces fit together must be explicit and clear.

In WWW pages, local readability is generally a given since their intended consumers are mostly only other human beings. Source code can suffer from a lack of local readability because programmers (and software engineers) tend to overlook a very important fact: source code is a description *intended for a human audience* of a machine's encoding of an algorithm. This statement is likely to raise cries of "Foul!" but is true nonetheless. The machine does not require structure in its code — such structure in fact often detracts from the performance of the machine, and gives rise to the necessity (or at least use) of optimizing compilers. This fact can most often be seen reflected in legacy code: when it was written, no attention whatsoever was paid to the need for human understanding of the code, and so code maintenance becomes a difficult if not impossible task. And yet, without maintenance, the machine still executes the code as (in)correctly as it always has.

It is important to notice the use of the words "generally" and "mostly" in the first sentence of the previous paragraph. This statement is not true in all situations anymore, and threatens to become increasingly false. The excessive use of gimmicks such as image maps as complete content tends to degrade this readability as is oft heard from critical users. As the use of automated *agents* [cheo96] increases, there is likely to be a trend towards catering to their needs instead of the needs of human beings, which needs are likely to remain disparate entities for some time to come.

What is currently absent from IEW is a coherent, committed view to global structural visibility. Yet, there have been several works which have used or attempted to use such structure: hypergraphs for grouping [tomp89, watt90], hierarchies [bota92, robe93, sali95], objects and classes [same94, rave94, luca96], linear [robe93] (and `latex2html`), and spatial [robe93]. What is needed is an "information architecture" approach akin to software architecture [garl93]. Since structure can provide partial context, and context can aid understanding, global structure in a local web region could aid understanding of that region. The identification of such structures and the elimination of ones which are too complicated for users to understand would be a nice first step. For example, Shipman *et al.* [ship95] "examined the kinds of spatial layouts that people produce in both computational and noncomputational media [and] developed a 'spatial parser' to recognize common structures." Familiar patterns permit both a context for discourse and for easier application of prior experience.

5 Conclusion

We have presented the case that information engineering on the World-Wide Web is intimately related to software engineering. IEW will suffer from the same problems that SE has, and may benefit from the practices learned there. The two activities are very similar: the construction, maintenance, determination, abstraction, and presentation of structure explicitly for human consumption.

6 Acknowledgements

Many of the ideas in this work have been refined, broadened and otherwise improved upon with the assistance, suggestions and criticisms of Kellogg Booth, Gail Murphy, Scott Flinn, Valerie Summers, Jim Boritz, Carolyn Wick, Makoto Kobayashi, and Chris Ellefson. Thank you one and all!

References

[ball96] Thomas Ball and Stephen G. Eick. "Software Visualization in the Large". *Computer*, Vol. 29, No. 4, pp. 33–43, April 1996.

[bern96] T. Berners-Lee, R. Fielding, and H. Frystyk. "Hypertext Transfer Protocol – HTTP/1.0". RFC 1945, IETF, May 1996.

[bota92] Rodrigo A. Botafago, Ehud Rivlin, and Ben Shneiderman. "Structural Analysis of Hypertexts: Identifying Hierarchies and Useful Metrics". *ACM Transactions on Information Systems*, Vol. 10, No. 2, pp. 142–180, April 1992.

[broo87] Frederick P. Brooks, Jr. "No Silver Bullet: Essence and Accidents of Software Engineering". *Computer*, Vol. 20, No. 4, pp. 10–19, April 1987.

[card96] Stuart K. Card, George G. Robertson, and William York. "The WebBook and the WebForager: An Information Workspace for the World-Wide Web". *Conference on Human Factors in Computing Systems, CHI-96*, pp. 111–117, April 1996.

[cele95] Augusto Celentano, Maria Grazia Fugini, and Silvano Pozzi. "Knowledge-Based Document Retrieval in Office Environments". *ACM Transactions on Information Systems*, Vol. 13, No. 3, pp. 237–268, July 1995.

[cheo96] Fah-Chun Cheong. *Internet Agents: Spiders, Wanderers, Brokers, and 'Bots.* New Riders, Indianapolis, IN, 1996.

[chil90] Elliot J. Chilkofsky and James H. Cross II. "Reverse Engineering and Design Recovery: A Taxonomy". *IEEE Software*, Vol. 7, No. 1, pp. 13–17, January 1990.

[cone96] Mary B. Coney and Carl S. Chatfield. "Rethinking the Author-Reader Relationship in Computer Documentation". **The Journal of Computer Documentation*, Vol. 20, No. 2, pp. 23–29, May 1996.

[fair88] K. M. Fairchild, S. E. Poltrock, and G. W. Furnas. "SemNet: Three-dimensional graphic representation of large knowledge bases". *Cognitive Science and Its Applications.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[garl93] D. Garlan and M. Shaw. "An Introduction to Software Architecture". *Advances in Software Engineering and Knowledge Engineering.* 1993.

[hala88] Frank G. Halasz. "Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems". *Communications of the ACM*, Vol. 31, No. 7, pp. 836–852, July 1988.

[hear95] Marti A. Hearst. "TileBars: Visualization of Term Distribution Information in Full Text Information Access". *Conference on Human Factors in Computing Systems, CHI-95*, pp. 59–66, May 1995.

[hutc85] David H. Hutchens and Victor R. Basili. "System Structure Analysis: Clustering with Data Bindings". *IEEE Transactions on Software Engineering*, Vol. 11, No. 8, pp. 749–757, August 1985.

[koik93] Hideki Koike. "The Role of Another Spatial Dimension in Software Visualization". *ACM Transactions on Information Systems*, Vol. 11, No. 3, pp. 266–286, July 1993.

[koik95] Hideki Koike. "Fractal Views: A Fractal-Based Method for Controlling Information Display". *ACM Transactions on Information Systems*, Vol. 13, No. 3, pp. 305–323, July 1995.

[lamp95] John Lamping, Ramana Rao, and Peter Pirolli. "A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies". *Conference on Human Factors in Computing Systems, CHI-95*, pp. 401–408, May 1995.

[luca96] Dario Lucarella and Antonella Zanzi. "A Visual Retrieval Environment for Hypermedia Information Systems". *ACM Transactions on Information Systems*, Vol. 14, No. 1, pp. 3–29, January 1996.

[maye85] Richard E. Mayer. "Structural Analysis of Science Prose: Can We Increase Problem-Solving Performance?". *Understanding Expository Text.* chapter 3, pp. 65–87. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1985.

[maye95] Richard E. Mayer. "A Second Look at Increasing the Understandability of Scientific Text". **The Journal of Computer Documentation*, Vol. 19, No. 3, pp. 45–50, August 1995.

[mukh95] Sougata Mukherjea, James D. Foley, and Scott Hudson. "Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views". *Conference on Human Factors in Computing Systems, CHI-95*, pp. 331–337, May 1995.

[müll93] Hausi A. Müller, Mehmet A. Orgun, Scott R. Tilley, and James S. Uhl. "A Reverse Engineering Approach to Subsystem Structure Identification". *Software Maintenance: Research and Practice*, Vol. 5, No. 4, pp. 181–204, December 1993.

[murp95] Gail C. Murphy, David Notkin, and Kevin Sullivan. "Software Reflexion Models: Bridging the Gap between Source and High-Level Models". *Proceedings of the 3rd ACM Symposium on the Foundations of Software Engineering*, 1995.

[ossh87] Harold L. Ossher. "A Mechanism for Specifying the Structure of Large, Layered Systems". *Research Directions in Object-Oriented Programming*, pp. 219–252. The MIT Press, 1987.

[parn94] David Lorge Parnas. "Software Aging". *Proceedings of the 16th International Conference on Software Engineering*, pp. 279–287, 1994.

[piro95] Peter Pirolli and Stuart Card. "Information Foraging in Information Access Environments". *Conference on Human Factors in Computing Systems, CHI-95*, pp. 51–58, May 1995.

[piro96a] Peter Pirolli, James Pitkow, and Ramana Rao. "Silk from a Sow's Ear: Extracting Usable Structures from the Web". *Conference on Human Factors in Computing Systems, CHI-96*, pp. 118–125, April 1996.

[piro96b] Peter Pirolli, Patricia Schank, Marti Hearst, and Christine Diehl. "Scatter/Gather Browsing Communicates the Topic Structure of a Very Large Text Collection". *Conference on Human Factors in Computing Systems, CHI-96*, pp. 213–220, April 1996.

[rave94] Mary Elizabeth Raven and Richard Thomson. “Can Principles of Object-Oriented System Documentation Be Applied to User Documentation?”. **The Journal of Computer Documentation*, Vol. 18, No. 1, pp. 15–19, January 1994.

[robe93] George G. Robertson, Stuart K. Card, and Jock D. Mackinlay. “Information Visualization Using 3D Interactive Animation”. *Communications of the ACM*, Vol. 36, No. 4, pp. 57–71, April 1993.

[sali95] Airi Saliminen, Jean Tague-Sutcliffe, and Charles McClellan. “From Text to Hypertext by Indexing”. *ACM Transactions on Information Systems*, Vol. 13, No. 1, pp. 69–99, January 1995.

[same94] Johannes Sametinger. “Object-Oriented Documentation”. **The Journal of Computer Documentation*, Vol. 18, No. 1, pp. 3–14, January 1994.

[ship95] Frank M. Shipman III, Catherine C. Marshall, and Thomas P. Moran. “Finding and Using Implicit Structure in Human-Organized Spatial Layouts of Information”. *Conference on Human Factors in Computing Systems, CHI-95*, pp. 346–353, May 1995.

[tolk66] John Ronald Reuel Tolkien. *The Lord of the Rings*. Allen & Unwin, London, UK, 1966.

[tolk88] Christopher Tolkien, editor. *The Return of the Shadow*, Vol. 1 of *The History of the Lord of the Rings*. Unwin Hyman, London, UK, 1988.

[tomp89] Frank William Tompa. “A Data Model for Flexible Hypertext Database Systems”. *ACM Transactions on Information Systems*, Vol. 7, No. 1, pp. 85–100, January 1989.

[utti89] Kenneth Utting and Nicole Yankelovich. “Context and Orientation in Hypermedia Networks”. *ACM Transactions on Information Systems*, Vol. 7, No. 1, pp. 58–84, January 1989.

[watt90] Carolyn Watters and Michael A. Shepherd. “A Transient Hypergraph-Based Model for Data Access”. *ACM Transactions on Information Systems*, Vol. 8, No. 2, pp. 77–102, April 1990.

[weis94] Edmond H. Weiss. “Isomorphism Between OOP and Documentation: Reflections on Sametinger’s “Object-Oriented Documentation””. **The Journal of Computer Documentation*, Vol. 18, No. 2, pp. 8–12, May 1994.