SEVEN WAYS TO MAKE A HYPERTEXT PROJECT FAIL

Robert J. Glushko
Hypertext Engineering

SUMMARY

Hypertext is an exciting concept, but designing and developing hypertext applications of practical scale is hard. "Hypertext engineers" must overcome seven problems to make a project feasible and successful:

1. **Developing realistic expectations in the face of hypertext hype**
2. **Assembling a multidisciplinary project team**
3. **Establishing and following design guidelines**
4. **Dealing with installed base constraints**
5. **Obtaining usable source files**
6. **Finding appropriate software technology and methods**
7. **Overcoming legal uncertainties about intellectual property concerns.**

The excitement in the technical community and the popular press is rapidly inspiring others to pursue the vision of hypertext as a means to combine text, graphics, voice, and other media to make information more accessible, usable, and entertaining. Nevertheless, the novelty and the immaturity of "hypertext engineering" as a discipline cause many projects to fall short of these goals. Elsewhere *[1]* I have emphasized some of the challenging design issues that must be overcome "to make the hypertext vision happen." In this paper I take a broader view to include management and nontechnical factors and expand the list to seven ways in which hypertext projects fail. Not all of these problems are specific to hypertext projects, but together, these seven issues combine to make hypertext applications hard to design, develop, and deploy successfully.

This article has been peer reviewed.

A COMPOSITE CASE STUDY

Attracted by the excitement about hypertext, Company C decides that links and navigation features will bring enhanced usability to a problem that has traditionally been handled in a database or document archive. Since it is the first hypertext project in Company C, all of the most talented and ambitious researchers and developers find their way onto the project. No one keeps track of how much time and effort go into it, but after a few months a carefully hand-crafted demonstration system or prototype emerges using "Hyper-X." Hyper-X is a highly touted program that everyone is talking and reading about as a revolutionary software advance.

The prototype system is flashy and compelling on the 19-inch workstation screen. It contains only a tiny amount of the information contained in the application it is intended to replace, but even a casual observer is impressed by how usable and appealing it looks. The colors, the graphics scanned from documents and books, the animation, and the sound effects come together seamlessly to create a multimedia proof-of-concept.

The vice president of Company C compares the prototype with the original text-based system and declares the Hyper-X version a smashing success. All that remains is to "scale it up" by converting the remainder of the information to hypertext.

SEVEN WAYS TO FAIL

How you interpret this case study so far might depend on where you are in your project, or whether you are part of the prototype team or a member of the team that must scale it up. But the stage is now set for disaster. In the sections that follow I analyze the seven ways in which the project can fail.

Unrealistic Expectations About Scale and Readiness

When the full-scale development project begins, the organization required to carry it out usually has unrealistic expectations about how hard it is and about the capabilities and resources needed to do it. Demonstration projects often bring together the best people, who have ample resources and whose efforts benefit from ad hoc interorganizational cooperation because of the novelty of hypertext. Many demonstration projects "succeed" by using methods and tools that are impossible to scale up *[2]*.

Often the demonstration project uses an off-the-shelf software package that provides neither the capacity nor the performance to deliver the bulk of information now managed by the traditional database or file system. Worse, the information examples and links in the demonstration project may have been carefully hand-crafted, an unworkable approach for a system several orders of magnitude larger. An obsessive focus on "how it looks" often drives out serious consideration of "how it has to work."

If a demonstration project takes three months to convert five articles from an encyclopedia into an interactive hypermedia form, how long will it take to convert a thousand articles using the same techniques? Automatic, semiautomatic, or template-based techniques are the only realistic option for large conversion projects, but these are almost never applied during the initial prototyping effort, which usually focuses on user interface concerns.

Over time, organizations will acquire an experience base that allows them to make realistic estimates for large-scale projects that involve text and hypertext. But in the meantime it is better to be conservative.

Missing Skills on Design and Development Team

Organizations considering a hypertext project usually have available plenty of software design and development skills. However, hypertext projects may require (or at least benefit from) a broader mix of skills. Appropriate skills in a hypertext project team include—

- Software designers
- User-interface designers
- Usability testers and potential users
- Technical writers and editors
- Indexers
- Database designers.

Hypermedia projects can require still other talents, including graphic artists and people who understand how to integrate multiple media, sometimes called "art directors." For conversion projects, the participation of the author or editor of the original information can be invaluable in explaining the presentation conventions of the existing format. If neither is available, the project should allow additional time to develop an understanding of the design and the design rationale of the existing information. Likewise, the people who run the plant, repair the airplanes, or otherwise are expert in the application area must be involved, or the project runs the risk of building a carefully designed user interface to the wrong information.

Few organizations or individuals have expertise in more than a few of these areas. Even if the needed mix of skills for a project organization can be assembled by matrix management or by bringing in appropriate consultants, making such an interdisciplinary team work effectively together is no small challenge.

Few Published Case Studies or Design Guidelines

Because hypertext is a relatively new design field, there are few detailed published case studies or design guidelines that designers can readily use. Published reports about hypertext are not representative, being typically biased toward small-scale demonstration or research projects. While hypertext applications of practical scale have been successfully designed and implemented, such projects are generally not documented in the literature because of resource constraints in development organizations, because of proprietary considerations, or because they are classified for security reasons.

While there is a growing body of empirical research evaluating particular hypertext systems or specific design options, this work does not usually generalize well. In addition, formal experiments have been able to establish that most of the design choices, when considered in isolation, have only small impacts on system usability *[3]*. To complicate matters, sometimes small changes in the user's task or the structure of the hypertext can lead to conflicting answers about the relative merit of design features *[4]*. Individual differences in users, especially motivational differences, and the effects of different tasks seem to have major effects on system usability. Yet the hypertext system designer usually cannot control who the users are and the tasks they want to carry out.

However, designers of hypertext systems can take steps to ensure that their systems are acceptable and effective for their users. While empirically validated design guidelines remain some way off, design methodologies for hypertext are being proposed; the most comprehensive of these is that of Perlman *[5]*. Designers should be conservative in adding features and should adopt an iterative design and evaluation strategy that can identify key usability challenges when it is still early enough to do something about them.

Installed Base Constraints

Hypertext demonstration projects are often done in research organizations that have advanced technology, including workstations and high-resolution 19-inch monitors. HyperCard on the Macintosh computer is also a very popular environment for demonstration projects.

In contrast, the users for whom full-scale versions of these demonstration systems must be targeted often work with an older or different installed base of computing equipment. This installed base may consist predominantly of IBM AT-compatible processors with small display screens having limited graphics resolution.

This situation often poses a dilemma for hypertext projects. Advanced technology may be needed to demonstrate the benefits of hypertext capabilities, but the presentation of these capabilities in the demonstration projects exceeds what the installed base will support. It is essential that the funding or marketing organization promoting the project know the costs and tradeoffs implied by various technology alternatives. Which is more successful, a project that uses less-advanced technology to create lower expectations that can be met, or a project that uses state-of-the-art technology that is not readily available for the average user? There is no right answer, but to ask the question is essential when project goals are being established.

In any case, the installed base slowly changes, so hypertext designs should be prepared to take advantage of new technology capabilities as they become available. A key consideration is separating the information storage from the user interface of the hypertext system so that the user interface can be enhanced as the installed base permits it. Temporary constraints in processing power for installed-base computers can be overcome by exploiting space vs. time tradeoffs in hypertext designs. For example, navigation maps can be precomputed and stored on a CD-ROM instead of computed in real time.

Poor Quality or Availability of Source Files

Many hypertext conversion projects are plagued by the poor quality or availability of source files. Many documents have no digital form, and even when one exists, it may not be readily usable unless a hypertext version was planned or contracted for when the documents were created.

One common limitation derives from the language with which the digital version of the document was composed or marked-up. The ideal situation is when the markup language is Standard Generalized Markup Language (SGML) or some other language that specifies the logical structure of the document rather than its presentation.

Optical character recognition (OCR) technology is rapidly improving, and new OCR devices that output text in SGML form are especially promising *[6]*. Nevertheless, error rates are not negligible, so proofreading is always required, and the nature of the residual errors in OCR documents makes manual text entry preferable unless correct recognition rates exceed 98 percent *[7]*.

Taken together, potential problems with source files make it essential that hypertext projects carefully investigate source quality and availability before committing to a project schedule. A single document sample may not be representative; often a large document or document collection (such as the complete set of manuals for a large system) is assembled from parts created by different vendors or subcontractors. Each supplier may have provided documents in a different source form. If documents are obtained in various source formats, it is generally more cost-effective to have a third-party text-conversion service transform all of them to a common format than to use project software resources to carry out the conversion.

Fortunately, as the "neutral data" specifications being developed as part of CALS *[8]* and IMIS *[9]* take hold, building hypertext applications on existing information will become significantly easier.

Hypertext is an attractive vision, but practical hypertext applications are hard to build. Disciplined approaches to analyzing information, identifying constraints in its structure and in the task environment, and using the appropriate implementation technology are required.

Hypertext projects whose application involves periodic publication of text created elsewhere should define formatting standards and quality control procedures for the organization that produces the information. These measures can lead to substantial improvement in the productivity of hypertext conversion by enabling the development of automatic conversion software.

Lack of Appropriate Software Tools

Most off-the-shelf hypertext software is oriented toward creating new hypertexts and is not well suited for converting existing documents *[1-2; 10]*. Demonstration projects often use this software to create expectations about the look and feel of a full-scale implementation, and developers often get a harsh shock when they discover fundamental limitations in the software that jeopardize the viability of a project.

It may be worth waiting for the next generation of hypertext software that directly supports conversion. The CALS initiative has prompted many vendors to enter this market, and the tools are expected to improve rapidly *[11]*. Alternatively, some database programs or expert-system shells may better support hypertext features than programs that call themselves hypertext.

If off-the-shelf software must be used for a hypertext project, it is imperative that any demonstration or proof-of-concept phase carefully address the pragmatic issues of scaling up. These include both capacity concerns—can the program manage significantly larger amounts of information with acceptable performance?—and resource concerns—does the program imply or impose design methods for defining units, links, or other features that cannot be applied on a large scale *[1; 10]*?

Legal Uncertainties

In recent years there has been a rash of "look and feel" copyright infringement lawsuits and similar claims for software patents. These legal controversies have arisen because software has been defined both as a kind of "literary work," which makes it copyrightable, and as a kind of machine or method of operting one, which makes it patentable. While these legal analogies may be wrong and may someday be corrected by a new intellectual property law that recognizes the special character of software *[12]*, software designers and developers today are faced with chaos, uncertainty, and legal action.

As unclear as the situation is for software in general, the novel character of hypertext and hypermedia software raises still more complexities for intellectual property law. For example, if copyright law has different rules for "literary works," "audiovisual works," "sound recordings," and "pictorial works," into what legal category does an interactive hypermedia encyclopedia or a talking book fall? Are new links or nodes in a hypertext system considered "derivative works" under copyright law? These and other issues are not just legal curiosities; they will have considerable impact on the legal protection available and hence the economic viability of hypermedia systems.

One aspect of copyright infringement that confronts hypermedia designers and developers is clear and well-known, but new technology has made it easier to break the law. OCR, scanners, digital samplers, and video "frame grabbers" make it possible to copy almost anything and incorporate it into a hypermedia system. But having the technology does not imply the right, and a sure way to invite a lawsuit is to assume that it does. It is not a coincidence that many hypertext applications have used government documents like standards and regulations that are free of copyright restrictions.

The best defense against a copyright infringement claim is to be able to prove independent development, so keeping careful documentation of design decisions is essential. In addition, designs based on experiments or evaluations give the design a "functional" character that narrows the scope of copyright infringement claims. It is best to follow the golden rule when designing a system: Borrow from others no more than you would have them borrow from you. An alternative formulation of this principle can be found in a well-reasoned paper that presents both sides of the look-and-feel debate: Let he who has never borrowed cast the first lawsuit *[13]*.

SUMMARY

Hypertext is an attractive vision, but practical hypertext applications are hard to build. Disciplined approaches to analyzing information, identifying constraints in its structure and in the task environment, and using the appropriate implementation technology are required. Successful hypertext projects are those that take a cautious approach to problems of scale and that make the right tradeoffs along the way. Ω

REFERENCES

1. R.J. Glushko, "Visions of Grandeur?" *Unix Review* 8, no. 2 (1990): 70–80.
2. L. Alschuler, "Hand-Crafted Hypertext: Lessons from the ACM Experiment," in E. Barrett, ed., *The Society of Text: Hypertext, Hypermedia, and the Social Construction of Information* (Cambridge, MA: MIT Press, 1989), 343–361.
3. J. Nielsen, "The Matters That Really Matter for Hypertext Usability," *Hypertext '89 Proceedings* (New York: Association for Computing Machinery, 1989), 239–248.
4. P. Wright, and A. Lickorish, "An Empirical Comparison of Two Navigation Systems for Two Hypertexts," in C. Green and R. McAleese, eds., *Hypertext: Theory into Practice II* (Oxford: Intellect Press, 1990), 84–93.
5. G. Perlman, "Asynchronous Design/Evaluation Methods for Hypertext Technology Development," *Hypertext '89 Proceedings* (New York: Association for Computing Machinery, 1989), 61–81.
6. G. Grygo, "High-Volume Scanners Aid Conversion to CALS Standard," *Digital Review* 6, no. 27 (July 10, 1989): 29.
7. W. Cushman, P. Ojha, and C. Daniels, "Usable OCR: What Are the Minimum Performance Requirements? in *Proceedings of the CHI '90 Conference on Human Factors in Computing Systems* (New York: Association for Computing Machinery, 1990), 145–151.
8. Department of Defense, *Computer-Aided Acquisition and Logistic Support* (Washington, DC: Office of the Secretary of Defense CALS Office, 1985).
9. D. Thomas, and J. Clay, *Computer-Based Maintenance Aids for Technicians: Project Final Report.* Air Force Human Resources Laboratory Technical Report AFHRL-TR-87-44, Wright-Patterson Air Force Base, OH, 1988.
10. R.J. Glushko, "Using Off-the-Shelf Software to Create a Hypertext Electronic Encyclopedia," *Technical Communication* 37, no. 1 (February 1990): 28–33.
11. R. Smithmidford, "Vendors Focus on CALS Conversions for Existing Paper Documents," *Federal Computer Week* 3, no. 3 (September 4, 1989): 38.
12. P. Samuelson, "Why the Look and Feel of Software User Interfaces Should Not Be Protected by Copyright Law," *Communications of the ACM* 32, no. 5 (1989): 563–572.
13. P. Samuelson, "Protecting User Interfaces Through Copyright: The Debate," in *Proceedings of the ACM Conference on Computer-Human Interaction - CHI '89* (New York: Association for Computing Machinery, 1989), 97–103.