Norman Meyrowitz

Institute for Research in Information and Scholarship (IRIS)
Brown University
155 George Street
Box 1946
Providence, RI 02912

The Missing Link: Why We're All Doing Hypertext Wrong

Despite over a quarter century of history, hypertext/hypermedia has not yet caught on as a **fundamental tool** for daily knowledge work. In the 60s and 70s, Engelbart's NLS possessed extremely advanced functionality, while Brown's FRESS provided rich word processing combined with sophisticated linking capability. In the 80s, Xerox's Notecards and Brown's Intermedia , to name two, have been research successes but have not yet been extremely widely accepted outside of the lab. Even OWL's Guide, with fairly good penetration in both the Macintosh and IBM PC markets and Apple's HyperCard, bundled with all newly-sold Macintoshes, are used not for daily knowledge work, but as a special-purpose tools to create online-help or specialized corpuses for a particular problem domain.

Carefully examining the systems created to date, however, uncovers a single common thread that I contend explains why hypertext/hypermedia systems have not caught on: virtually all systems to date are **insular, monolithic packages** that demand the user disown his or her present computing environment to use the functions of hypertext and hypermedia.

NLS, though probably a more powerful, integrated, and feature-laden package than any that have since been created, still made it difficult for the user to use the other non-NLS applications upon which he or she depended. Rather than use those applications within the NLS environment, the user was forced to exit or suspend NLS to operate non-NLS programs. Likewise in FRESS, one resided in a word processing/hypertext environment. To move to a compiler or a statistics package, one needed to exit the FRESS environment. This certainly was a chore if one needed to read the statistics or source code in creating a hypertext document, and a "show-stopper" if one wanted to actually link to the statistics or code into the hypertext corpus.

Even in today's technology, the same barrier exists. In Notecards, which runs in the very rich InterLisp-D environment, Notecards "takes over" and provides a monolithic environment. If one wants to create hypertext documentation, for, let's say, an expert system written in the InterLisp-D programming environment, it would be difficult to do so using Notecards; when Notecards is operational, the other environments are largely unreachable. On the Macintosh and under Microsoft Windows, Guide affords a similarly insular environment. Rather than allowing users to link a Word document to an Excel document to a MacDraw document, Guide forces the user to "import" all of those documents into the Guide system. Once in Guide, the word processing features

of Word are no longer available for further editing, the drawing facilities of MacDraw are no longer available for further drawing, and the calculation facilities of Excel are no longer available for further refinement. With HyperCard, one can essentially suspend HyperCard and launch one of the existing Mac desktop applications like Excel or MacDraw, exit those, and return to the launching point in HyperCard, but it is impossible to link into HyperCard from any of those desktop applications.

In our own Intermedia system, we have much the same situation. We have built an architecture that allows new applications, if they implement a small linking protocol, to be full-fledged Intermedia documents that can be the source or destination of links. Yet our application base consists entirely of those that we write ourselves. Unfortunately, we do not have the market impact and persuasive power to convince third-party developers to create scores of new applications that adhere to a linking protocol. Who does?

Five years ago, with the exception of people at Xerox PARC and a few pioneers using Smalltalk and Lisp in research laboratories and academia, the paradigm of "cut, copy, and paste" was virtually unknown. Now, with the advent of the Lisa and Mac toolboxes, and more recently, of Microsoft Windows, that paradigm is a familiar one, even to five-year-olds using MacPaint. This paradigm caught on for four reasons: 1) powerful things could be done with this paradigm; 2) the paradigm was extremely easy to motivate and teach to end-users; 3) the toolbox vendors touted the copy and paste protocol as an important integrating factor that all software developers should include in their applications; and 4) most importantly, the toolbox supporters provided the framework for copy and paste deep in the system software and provided developers the protocols that enabled them to incorporate the paradigm into their software with relative ease. The paradigm is so widely-accepted that consumers regularly sneer at and ignore software that does not provide full cut, copy, and paste support.

Hypertext/hypermedia has the same potential for making fundamental improvements to people's daily work. Like "cut, copy, and paste," making and following links fulfills factors one and two — it provides a powerful integrating ability and is reasonably easy to motivate and teach to knowledge workers. I believe, however, that hypertext/hypermedia will only catch on as a fundamentally integrating paradigm when factors three and four can be fulfilled. Linking functionality must be incorporated, as a fundamental advance in application integration, into the heart of the standard computing toolboxes — the Mac desktop, Microsoft Windows, etc. — and application developers must be provided with the tools that enable applications to "link up" in a standard manner. Only when the paradigm is positioned as an integrating factor for all third-party applications, and not as a special attribute of a limited few, will knowledge workers accept and integrate hypertext/hypermedia into their daily work process.