HyperSource:
A Hypermedia Program Development And Documentation System

Jan Oliver Borchers
Institute for Operating And Dialog Systems, University of Karlsruhe, Germany
Postal Address: Gottesauer Str. 21, 76131 Karlsruhe, Germany
Phone: ++49-721-60 65 04, Scall: 01681-113 59 58
`Email: job@ira.uka.de`
`URL: http://i31www.ira.uka.de/~job`

Keywords:
Programming, Development, Documentation, Hypertext, Hypermedia, Multimedia, Authoring Systems, Emacs, HTML

Abstract

HyperSource extends hypermedia to the area of program development: Source code and documentation are developed as structured multimedia documents. Marginal annotations and images comment the source code. Links help the reader navigate through it, and find the documentation for a certain piece of code. This makes programs easier to develop, read, understand, and thus reuse. The concept can also be applied easily to the area of computer-based learning, especially to computer science practicals. The project includes the implementation of a real-world development system, based on widespread existing tools and the HTML standard for structured documents.

Background: How Programmers Document Their Code

The way most programmers write their source code to this day still seems to follow the old Internet saying, "If it was hard to write, it should be hard to read." Source code is usually saved in **ASCII** format, with some humble formatting attempts mostly consisting of lines or boxes made of little asterisks to mark comments. Images would often be very useful to explain what a piece of code actually does, but if they are included at all, then as "ASCII Artworks", pictures of a rather questionable value, and often produced arduously with hardly any tool support. A single, fixed-width font has to be used for printing and display - in short, source code is usually typographically dreadful, and takes hardly any advantage of the display capabilities of today's graphical workstations.

Moreover, those documents are essentially **linear** with no inherent support for navigation, e.g., to go from a function call to the module where it is defined, or to jump to that paragraph in the documentation where the algorithm behind a function is explained. Although external tools are available for some of these tasks (e.g., the `ctags/etags` system), they cannot deal with arbitrary links or multimedia documents.

HyperSource: The Concept

To overcome this obsolete situation, we developed the concept of **HyperSource**: Source code and documentation are created as structured hypermedia documents. This approach bears several advantages:

- *Development* becomes more natural. To design a graphics software package, for example, the developer can start from an initial "Top-level Project Index" page, and design the system in a top-down fashion, creating links from this index to other pages that represent the different modules. Those pages can contain as anchors the names of functions to be written, pointing in turn to the actual function definitions (initially empty document frames).

  This technique also supports the "Jo-Jo" style typically encountered in real-life software design, as it makes changing between different levels of abstraction easy and intuitive.

- *Implementation* becomes more efficient because navigation between modules, from function calls to their definition, etc., is facilitated. A HyperSource authoring system can insert some links automatically, others can be added by the programmer as he likes. This structural information is intrinsic to the document, not merely computed syntactically by some external tool. To insert a comment, the programmer types it in, and the system formats it automatically, e.g., as a "marginal annotation" next to the source code. Comment headers within the code can also be created easily, without having to cut-and-paste empty "comment frames" by hand (see Figure 1).

- Finally, *documentation* quality is improved through the possibility to insert multimedia comments into the source code, and to create links between it and the documentation, both being HyperSource documents. If the programmer wants to show, for example, which geometrical case is handled by a certain block of his line intersection function, he can embed a sketch directly next to that part of the code. Movies or other media types can be attached as well and displayed or edited via external modules.

This leads to the main advantage of HyperSource program documents: They are much easier to read by others than ordinary source code. This facilitates understanding the software, which in turn helps people to "trust" that package. Thus, **reusing existing software**, instead of reinventing the wheel again and again, is promoted. Figure 1 shows what HyperSource code may look like.

A Sample HyperSource Editing Tool

To see if this new programming paradigm can be applied successfully in the real world, a HyperSource editing tool is currently being implemented, mainly for use in Unix/X environments. A questionnaire showed, among other results, a strong aversion to "new" editors, so the system is implemented in LISP as an **extension of XEmacs** [Tho94], an X-aware version of the standard program editor, GNU Emacs [Sta94]. Source code documents are shown in a WYSIWYG manner, with embedded pictures and comments as marginal annotations, and including links to the documentation. Function calls and other identifiers are anchors pointing to their definition, and comment headers are displayed in a different style to structure the source code visually.

To enable many people to benefit from the formatted appearance of HyperSource documents, they are **saved in HTML format**, thus being readable by anyone who has access to a HTML browser. To preserve the "two-column" layout (source code body + annotations), the `table` feature of the new HTML3 standard [Rag95] is used.

The system is tailored to help software developers design, implement and document their projects with a minimum of additional learning or operating effort. The source code can be extracted automatically in a compiler-readable form as with the "Save as ASCII" option of standard browsers. Compiler errors may be mapped back into the HyperSource document. The editor supports the development of C programs but can be easily adapted to deal with other languages.

Evaluation of HyperSource will take place in a programming practical where students are supposed to write their own modules to complete a given program frame. Problem sheets will be HyperSource documents describing the problem and containing the existing "framework" parts of the program. We intend to release the HyperSource implementation on the Internet for evaluation afterwards.

References

[Big88]
Bigelow, James: "Hypertext and CASE", in: IEEE Software, March 1988

[Rag95]
Raggett, Dave: "HTML 3.0 Document Type Definition", Geneva 1995

[Sta94]
Stallman, Richard M. et al.: "GNU Emacs Lisp Reference Manual", Second Edition, Free Software Foundation, Cambridge,MA 1994

[Tho94]
Thompson, Chuck: "XEmacs", Urbana-Champaign 1994