A Generic Dynamic-Mapping Wrapper for Open Hypertext System Support of Analytical Applications

Chao-Min Chiu
The New Jersey Center for Multimedia Research
Rutgers University
University Heights—Newark, NJ 07102 USA
chchiu@pegasus.rutgers.edu

Michael Bieber
The New Jersey Center for Multimedia Research
New Jersey Institute of Technology
University Heights—Newark, NJ 07102 USA
bieber@cis.njit.edu — http://megahertz.njit.edu/~bieber

ABSTRACT
Hypertext should augment everyday analytical applications with supplemental navigation, structuring and annotative features. Because analytical applications generate their screen contents in real time, hypertext constructs and navigation paths must be generated and mapped in real time. We are developing a hypertext engine that provides dynamic mapping automatically for any analytical application. We propose a standard, generic open hypertext system (OHS) wrapper for back-end or storage-level components that dynamically generate their contents (e.g., the analytical applications). The wrapper automatically maps hypertext constructs—nodes, links and anchors—to application contents. (The storage-level wrapper itself creates hypertext constructs instead of the users.) The hypertext engine delivers supplemental hypertext functionality based on these mappings. Furthermore, by providing a standard format and a set of guidelines, we are providing a standard protocol or systematic approach for exchanging information between an OHS and any analytical application. This adds to the work in the OHS community on developing a standard protocol for passing information among OHS and integrated applications.

KEYWORDS: Hypertext, hypermedia, open hypermedia systems, information systems, World Wide Web

1 A GENERIC DYNAMIC-MAPPING WRAPPER...

Despite over 30 years of active hypertext research, and the recent vast interest in the World Wide Web, hypertext has yet to make its greatest impact on the everyday analytical applications in the scientific and business world (accounting and finance systems, CAD, CASE, CBT, DBMS, decision support systems, executive information systems, geographic information systems, spreadsheets, etc.). Hypertext should augment these applications with supplemental navigation, structuring and annotative features. Hypertext functionality will have only secondary importance for these applications; users will employ them for their primary analytical functionality. Hypertext will give more direct access to this primary functionality, give access to metainformation about objects that appear on the application screen, provide alternate navigation paths through the application space, and enable annotation and ad hoc links. Because these analytical applications generate their screen contents in real time (as opposed to just retrieving existing documents and display contents), hypertext constructs and navigation paths must be generated and mapped for these screens in real time. Our research goal is to provide a hypertext engine that can handle this real-time or *dynamic mapping* automatically for any analytical application. (We call these applications dynamic-mapping information systems or DMISs.) Furthermore, as we cannot expect to change these applications in any major way, we need to leave them "hypertext-unaware". The hypertext engine will perform all hypertext functions, and maintain all hypertext mappings and constructs on their behalf. Indeed, for applications with APIs where we can provide our own user interface, we need make *no changes* to the application at all, while providing with complete hypertext functionality.

We propose a standard, generic open hypertext system (OHS) wrapper for back-end or storage-level components (DMISs) that dynamically generate their contents. The wrapper automatically maps hypertext constructs (nodes, links, anchors) to DMIS contents. The hypertext engine delivers supplemental hypertext functionality based on these mappings. In other words, from the OHS viewpoint, the storage-level wrapper itself creates many of the nodes, links and anchors instead of the users at the user interface level. Furthermore, by providing a standard format and a set of guidelines, we are providing a standard protocol or systematic approach for exchanging information between an OHS and any DMIS. This adds to the work in the OHS community on developing a standard protocol for passing information among OHS and integrated applications.

To the extent that other OHS support dynamic-mapping applications, they do so on a case-by-case basis instead of in a standard way (e.g., integrating Chimera with Framemaker [1]), they only support limited domains (e.g., Garrido and Rossi's support for DMIS applications written in

Hypertext 97, Southampton UK
© 1997 ACM 0-89791-866-5...$3.50

```
[User] -- [UI]  -- [UI Wrapper] =========== [UI Knowledge Base]
                              |
        [Dynamic Mapping Hypertext Engine]
                              |
         [DMIS] -- [DMIS Wrapper] ========= [DMIS Knowledge Base]
            :
            :
 ...............................................
 :                          :                           :
[DMIS Instance 1]      [DMIS Instance i]      [DMIS Instance n]
```

Figure 1: Basic Architecture for a Dynamic-Mapping Hypertext Engine. Solid lines indicate bidirectional flows of messages among the subsystems. Dotted lines indicate the various instances of (written within) the DMIS (e.g., a particular database within a DBMS or a map within a geographic information system). Double lines indicate that a wrapper stores and retrieves information with a knowledge base.

VisualWorks Smalltalk [5]), or they only map hypertext to display values as opposed to the objects underlying these values (e.g., with Microcosm's Universal Viewer [4]).

We map hypertext links to DMIS elements based on an element's internal identity, not to its display value. A stock's price, for example, can change, but the stock's identity never does. We provide mapping rules (bridge laws [2]) describing which components of the DMIS's internal structure correspond to which hypertext constructs (nodes, links, and anchors).

We model DMIS as having two logical parts: a computational portion and a user interface portion [3]. DMIS application content is produced (generated or retrieved) in the computational portion and displayed in the interface portion. In this research we concentrate on the computational portion. In the architecture below we assume that the user interface displays the information that the hypertext engine sends it and that it provides a means for selecting link markers. (This interface may be separate from the DMIS application, though in future research we wish to access DMISs through their native interfaces while not changing their computational portions at all. We initially intend to implement this architecture for DMISs with APIs, so we can utilize a WWW browser as a separate user interface.)

Figure 1 presents our basic architecture. The final architecture will serve multiple DMISs and user interfaces (UI), including WWW browsers. All components of the architecture may be distributed. The architecture's Dynamic-Mapping Hypertext Engine (DHTE) will serve any DMIS and UI that has an appropriate wrapper and wrapper knowledge base. To integrate a new DMIS or UI, one has to specify just a wrapper and knowledge base (which could very well prove a complex task). We currently are developing guidelines for doing this.

The wrappers serve three important functions. First they translate messages from the DHTE's standard format to something the UI or DMIS can process, and *vice versa*. Second, they pass messages between the two systems. Third, they buffer the UI or DMIS, implementing any functionality the engine requires of the them, which they do not provide.

The UI knowledge base contains the information the UI wrapper needs to communicate with its UIS. In it, the UIS handler maintains communication formats, current parameter settings and any routines necessary to maintain the level of coordination the DMIS requires of the UI. The DMIS knowledge base contains all mapping rules for converting DMIS constructs to hypertext constructs. It also contains network access information for each DMIS, as well as routines necessary to build messages for it and parse its responses.

ACKNOWLEDGMENTS

Ken Anderson and Uffe Wiil made invaluable comments which helped frame this research — thank you both! This research has been funded generously by the New Jersey Center for Multimedia Research, the New Jersey Commission of Science and Technology, the NASA JOVE faculty fellowship program, the AT&T Foundation, and the New Jersey Institute of Technology under Grant 991967.

REFERENCES

1. K. Anderson, R. Taylor and E. J. Whitehead, "Chimera: Hypertext for Heterogeneous Software Environments," ECHT'94 Proceedings, 94-107.
2. M. Bieber, "On Integrating Hypermedia into Decision Support and Other Information Systems," *Decision Support Systems* 14, 1995, 251-267.
3. M. Bieber and C. Kacmar, "Designing Hypertext Support for Computational Applications," Communications of the ACM 38(8), 1995, 99-107.
4. H. Davis, S. Knight and W. Hall, "Light Hypermedia Link Services: A Study of Third Party Application Integration," ECHT'94 Proceedings, 158-166.
5. A. Garrido and G. Rossi, "A Framework for Extending Object-Oriented Applications with Hypermedia Functionality," forthcoming in the *Hypermedia Journal*, Taylor Graham.