Multicard : An open hypermedia System

Antoine RIZK
EUROCLID,12 Ave des Prés
78180 Montigny le Bretonneux, France
email : rizk@nuri.inria.fr, Tel: ++.33.1.30.40.14.56, Fax: ++33.1.30.57.18.63

Louis SAUTER
Bull LPAM,7 rue Ampère
Massy 91343, France
email:L.Sauter@frmy.bull.fr,Tel: ++33.1.69.93.89.64, Fax: ++33.1.69.93.76.69

Abstract

This paper describes the Multicard hypermedia system which has been developed following an open systems approach . Multicard provides a hypermedia toolkit that allows programmers to create and manipulate distributed basic hypermedia structures; an interactive authoring/navigation tool which is itself based on the toollkit; an advanced scripting language; a multimedia composition editor, as well as a communication protocol that allows the integration of various editors and applications into a single hypermedia network.

One of Multicard's features is that it does not itself handle the contents of the nodes. Instead, it communicates with different editors, running as separate processes, using a set of messages called the M2000 protocol. Multicard has so far been connected in this way to around five different M2000 compliant editors and applications ranging from a basic text editor and data sheet to sophisticated desktop publishing and multimedia composition systems. M2000 compliant editors automatically benefit of the Multicard linking facilities and composite structures. Using the Multicard scripting language, M2000 compliant editors can also annotate their contents with scripts and communicate with each other using event and message transmission.

Keywords : Multicard, M2000, hypermedia toolkit

1 Introduction

Since the appearance of the very first hypertext systems, hypertext developers and users have called for the necessity of developing open hypertext systems which are simple to integrate with existing applications. Meyrowitz[10] and Brown[4] are amongst the first to observe this need. Meyrowitz[10] called for a link service that all applications can use. Brown[4] observes "if hypertext systems are to realise their full potential, they must aim for a seamless interface with other tools, thus making the whole greater than the sum of the parts". Malcolm[9] calls for interoperability and integration as the first requirement for industrial strength hypermedia : " Systems should be designed that enable engineers to link data created with their own tools rather than by having to use special hypermedia editors" .

If we look at current hypertext development, we observe two major trends neither of which meets the above requirements :
On the one hand, we observe the development of hypertext *systems* that focus largely on the management of hypertext objects and provide complete Runtime and Presentation layers (see Dexter[8]). These systems pay relatively little attention, given the additional cost involved, to the Within component layer. They usually offer basic document processing functionalities with no regards to standards or sophistication.

On the other hand we observe the development of hypertext *extensions* to a multitude of existing content-based applications such as word processing packages, CAD applications, graphics tools etc. These extensions, on the contrary, usually provide an elementary linking facility between two endpoints in the application data. There are no concept of nodes or composites, neither of link types etc. It is this trend that seems to be gaining support in the industrial arena judging by the number of existing products with such extensions, and by the emergence of hypermedia standards such as Hytime[6] and MHEG[5].

What lacks in order for these two trends to meet, is a clear specification of the interface between the content oriented applications and the hypermedia management system. In other words, a specification and implementation of the anchoring layer of the Dexter model both as a linkable API as well as a communication protocol.

Multicard is a complete hypermedia system in that it offers the basic set of hypermedia objects, an authoring/navigation tool, a scripting language and a multimedia composition editor. The M2000 communication protocol of Multicard provides a means for stand-alone applications, running as separate processes to be considered as an integral part of the Multicard system. Applications that respect this protocol automatically benefit of the entire Multicard features, including scripts, linking facilities and composite structures. Multicard and M2000 also provide for different such applications, a medium for communication and cooperation through cross linking and message/event transmission.

2 Related work

Existing open system/toolkit approaches to hypermedia include the Sun Link Service[7], the Hypertext Abstract Machine[12], the Andrew Toolkit[14], the PROXHY system[15], Intermedia [16] and the Eggs/HOT[13] approach.

The Sun's Link Service[7] defines a protocol for an open hypertext system. The M2000 protocol adopts a very similar approach to the Sun Link Service which is probably the only system that compares so closely to it. The Sun's Link Service however, provides an extremely loose coupling of applications and stretches openness to its limits. It provides only for a distributed linking mechanism and a means for representing and storing the source and destination of the link . As the author of Sun's Link Service observes, there is a tension between support for heterogeneity which is a goal of open systems and the notion of integration. The Sun's Link Service modulates this tension by defining as simple and unrestrictive a protocol as possible.

The PROXHY system[15] illustrates a very advanced architecture in the open hypermedia systems direction, in which anchors and links as well as applications are considered as separate actors communicating through message passing. Considering anchors and links as processes helps in parallelising interaction with these objects but could be rather heavy on performance. As with the Sun's Link Service, the communication protocol is minimal and unrestrictive.

With the M2000 protocol, the compliant application benefits of the entire Multicard features and components including the scripting language, object hierarchies and composites, event propagation and the authoring/navigation tool. In this sense the M2000 protocol is more developed but not as simple and unrestrictive. Editors, however, may choose to comply with M2000 at any depth, depending on the level of integration required.

The Intermedia system also provides a set of hypermedia services for applications to use. Intermedia goes into great depth as regards policy to which applications should adhere as well as in managing consistency of the hypermedia webs. As we mention later, the existence of the scripting language in Multicard imposes new functionalities on the communication protocol and renders the enforcement of presentation policies extremely difficult.

The HAM is perhaps the first system to offer a toolkit approach to hypertext. Whilst the HAM implements many of the classical hypertext features including distributed access of nodes/anchors, attribute/value pairs, and configuration management, the HAM does not define a communication protocol with independent applications. Such applications have to be tightly linked to the HAM Toolkit. A similar argument applies to the Eggs/HOT system, although the integration of existing tools should be easier given "the high degree of generality and encapsulation" in the elements of the toolkit. The authors of Eggs/HOT have however observed the possibilty of using process communication for their system.

The Andrew Toolkit defines a set of linking facilities as well as a scripting language. Whilst the ATK is a very flexible tool for the development of multimedia and hypermedia applications, the *integration* of a new editor/application is not possible. Such an editor has to be programmed as an inset using the ATK Classes language.

The Multicard system is in adequacy with respect to the Dexter model[8] in that it provides the decomposition and the functionalities of all of the Runtime Layer, the Storage Layer and the Within component layer. The M2000 protocol goes in greater depth into the specification *and* implementation of the Anchoring interface between the Storage and the Within component layers.

3 The Multicard architecture

The hypermedia system architecture is illustrated in figure 1, represented as a set of components and interfaces. The architecture is based on the general model of front-end and back-end subsystems. It has the following distinct layers : A set of Hypermedia Basic Classes that constitute the toolkit; hypermedia distributed persistent object storage; an authoring/navigation tool; a communication protocol and a series of compliant editors.

3.1 Hypermedia objects

The heart of the hypermedia toolkit is the representation of hypermedia objects (Nodes, Groups, Anchors and Links, Hypergraph etc...) together with the associated interfaces to applications, the scripting language and the editable objects. These hypermedia objects are implemented in C++. They can ce accessed either from C++ or through a C binding. We recall here briefly the specific features of these objects :

Nodes : In Multicard, there is a difference between node structure, which manages links scripts and the content of the node : Multicard manages the node structure, whereas the contents of the node may be handled by different editors. The M2000 protocol between Multicard and the editors allows to open and close documents, retrieve content portions etc...

Anchors : An anchor *represents* a sensitive portion of the content of a node. The associated anchor is the hypermedia object that carries the links, scripts, and other hypermedia properties. The sensitive portion is editor dependent.

Groups: Groups represent logical collections of nodes and other groups. Group hierarchy can be of illimited depth.

Links: Contrary to the usual usage of links in hypertext systems. Links in Multicard are viewed as event/message communication channels between two end points. Various messages can be sent through a link, including of course, the activation message which will typically open and map the destination object. Link endpoints can be anchors, nodes or groups. This way of viewing links has a fundamental advantage in hypertext systems that offer scripting languages in that it drastically improves the reconfigurability of hypertext applications.

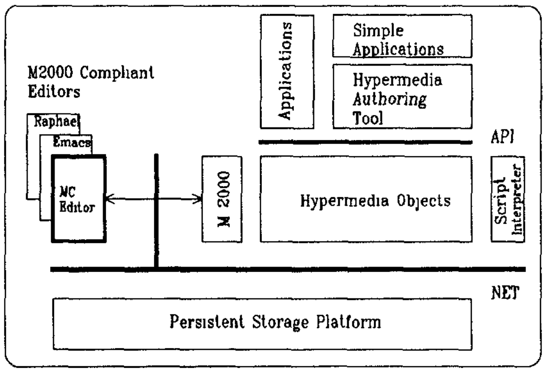

Figure 1 : Multicard Architecture

The link in this sense acts as a handle or as a port to the destination object. Scripts communicate with links given their attributes and properties that give an indication of the type of destination object. If the destination object is changed, the source object script does not have to be aware of the change.

This feature has proved extremely effective in hypertext applications that evolve constantly, and in design of applications that require a number of iterations before final release. Of course, this notion provides no advantage with systems that do not offer a scripting language.

The Application programming Interface (API) provides facilities to specific hypermedia editors, tools, and general applications for the creation, manipulation and deletion of hypermedia objects. The M2000 interface supports interaction between hypermedia objects and the content-based editors.

3.2 Script interpreter

Nodes, Groups and Anchors may have scripts attached to them. In this sense scripts are used as a way of extending the behaviour of instances of these objects. Scripts are event driven and can communicate throughout the hypermedia application using event/message passing. The scripting language contains over 150 instructions that bring the Application Interface closer to the end-user and include special instructions for manipulating editor contents, synchronising with the editor, definition of internal/external functions etc.

The existence of a scripting language adds a fundamental difference to the M2000 protocol design :

i) The protocol has to meet the requirements for dynamic hypertext support[2] ;

ii)It should allow for dynamic content manipulation ;

iii) It should be extensible through scripts ;

iv) It has to provide for transmission of system events and user messages ;

v) With scripts documents are active and *become* part of the user interface [3]. The authoring tool of Multicard could for example be entirely rewritten using an editor, the M2000 protocol and the scripting language. The provision for a specific homogeneous user interface for protocol manipulation across applications is therefore unnecessary.

3.3 Hypermedia persistent storage

This provides distributed persistent storage for hypermedia basic objects and consists of two parts. The front-end supports access by the hypermedia basic class objects and guarantees consistent behaviour independently of the actual storage management implemented by the back-end. This approach enables the storage mechanism to be implemented using relational database or object-oriented database without affecting the toolkit interface.

3.4 Hypermedia authoring tool

The hypermedia authoring tool is an X/Motif application written using the Hypermedia objects API. It allows to browse through the group hierarchy, to interactively create hypermedia objects (using the group editor) and to edit scripts (using the script editor). The hypermedia authoring tool can be used to create simple hypermedia applications. The author creates groups and nodes, edits the contents of the nodes using an M2000 compliant editor, creates anchors and links and optionally writes scripts to determine how objects react to certain events.

More complex applications can be implemented:

- by first writing external procedures (e.g. in C) that can be linked with the hypermedia authoring tool, and can be called from the scripts, and then proceeding as above.

- by writing a new application, independent of the hypermedia authoring tool. Typical applications of this kind include program driven on-line help systems, expert system driven maintenance tools etc....

4 The M2000 protocol

4.1 M2000 compliant editors

As mentioned above, M2000 compliant editors are used to manage the content portion of nodes. When an application creates a new node, it must indicate which editor will be used to manage the node content. From then on, the appropriate editor will be activated whenever the node is to be displayed. Many editors have so far been rendered M2000 compliant. These include the Multicard specific multimedia composition editor MCEditor, emacs, the GODraw graphics editor and the Raphael/Balzac desktop publishing system of Bull.

Editors can provide various levels of support for the M2000 protocol; the minimum M2000 support consists in handling two requests (Open and Close Node) and sending an error message for all other requests. Each hypermedia node structure (handled by the hypermedia toolkit) contains the name of the editor that is to be used to edit the corresponding document.

Multicard does not define the user interface for the editors: editors are free to use any style of presentation. The format in which the contents are stored is also editor dependent. In order to manage node deletion more consistently, Multicard suggests the path of a directory in which the document may be stored. This path is passed to the editor every time the document is opened.

4.2 Sensitive areas and anchors

Interactive multimedia applications developed using Multicard allow, for example, to trigger actions by clicking on some part of a picture. What actually happens is this :

- The editor detects a mouse click in a predefined area in the picture;
- It sends information about the mouse click to the Multicard object representing the sensitive area;
- Multicard handles the event by executing the Multicard object's script.

The parts of documents that can trigger actions are called sensitive areas. The properties of sensitive areas (geometry, kind of events captured etc ...) are the sole responsibility of the content editor. An anchor is a hypermedia object associated to a sensitive area. Events (Mouse click, key events ...) occuring inside a sensitive area are sent to the corresponding anchor. Multicard manages the semantics (links, scripts , ...) as well as the storage of anchors.

Each anchor has a unique identifier, delivered by Multicard when the anchor is created, and which must be stored by the editor for future reference together with the properties of the sensitive areas. It is entirely up to the editor to determine what constitutes a sensitive area, this could be a hot spot or hot word, a graphical object, a push button or slider bar, a datasheet cell or any collection of these. Moreover, a sensitive area does not have to be a contiguous piece of information in its physical representation.

Editors should support two modes : *AuthorMode*in which the user can edit the contents of the document, select objects, add or remove anchors, etc. and *ReaderMode*where most user actions are sent as M2000 messages if they occur inside the region of an anchor, and are ignored otherwise.

4.3 The M2000 Library

The M2000 protocol can operate with any editor whether running under the X-Window system or not and regardless of the user interface the editor might have. At connection setup, an editor using the X-lib initializes the M2000 by providing its context, name, a call-back function for receiving the M2000 requests, and the maximum number of connections it can handle. Editors not using the X-window provide a list of various streams from which they could handle input.

Once the connection is established, the editor communicates with the hypermedia system by exchanging a set of messages. These can be grouped into two main classes :

<u>Multicard messages</u>

These are requests sent by Multicard when some action is to be performed by the editor. These in turn can be classified as :

i) *Requests concerning nodes* : Open, Close, Print Save, Get/Set Property(background color, page scroll etc .), Search for Text etc ;

ii) *Requests concerning anchors* : Create Delete, Map ... These are used only via scripts to manipulate anchors dynamically ;

iii) *Requests concerning editable objects* : Create, Cut/Paste, Select zone, Get/Set object properties, Set focus etc. Multicard supports four ways for referring to editable objects in a document : By editable object id if the editor manages persistent identifiers, by name, by position and by anchor. The latter refers to the editable object containing the anchor.

iv) *Requests concerning menu management :* These requests allow scripts to modify the menu bar for each document in Reader/Author mode. Such messages may request the editor to Add/Remove/Activate/Deactivate a given menu/menu item and to Hide/Show the menu bar.

Editors can choose to respect M2000 at any level, provided this is done in a consistent manner. Should an editor require special extensions to M2000, this could be done through a specific message that takes a simple string as argument to be processed as a series of commands. The content of the string and the manner in which it is processed are editor dependent.

Editor messages

These are messages sent by the editor when an action is to be performed by Multicard. They in turn can be classified as :

i) Replies to above Multicard requests;

ii) Messages containing the notification of events such as mouse events, focus events and menu events. Menu events are sent to Multicard when the user selects a menu item. In most cases, this is optional, but doing so will allow application authors to tailor the behaviour of the used editors.

In Multicard, the *editor* is responsible for providing the user interface for manipulating anchors (Delete, Create, Select, EditScript, ...) and optionally for the manipulation of nodes (Open, Close, Save, Delete...). Such user interface commands are sent as specific menu events to Multicard. A typical session for creating an anchor would be :

–The user marks some region in the document as sensitive area and chooses "create anchor" in the editor menu;

–The editor sends a CreateAnchor–MenuEvent to the hypermedia system ;

–The hypermedia system creates a new anchor and sends a CreateAnchor request to the editor with the new anchor id;

–If the editor receives a valid CreateAnchor request, it updates its contents to include the new anchor id. Its future messages concerning this anchor will include the anchor id as parameter.

5 Conclusions

This paper has described the Multicard system and the M2000 communication protocol. Although Multicard offers a complete set of functionalities for the development of classical hypermedia applications in a stand–alone fashion, 4 out of 5 Multicard users develop access to M2000 and use the provided API in order to integrate their specific functions and tools. Full compliance with M2000 requires on average two to four weeks of development effort. However, editors and applications can choose to comply with M2000 at any depth depending on the level of integration required.

Multicard is now being used for research as well as for real applications at many sites throughout Europe.

Real applications include the development of an online help system integrated to a France Télécom network control service, an estate department imagery application, a multimedia presentation server etc...

Research and development experiments include integration of Multicard with the object–oriented databases ONTOS and O2 (at Bull and INRIA–Rocquencourt respectively), design and implementation of a graph–based query language[1], experimenting with multimodal user interfaces for hypertext (at INRIA Grenoble), porting on the distributed object–oriented ANSA platform (STC Great Britain), Integration with the Knowledge Engineering System Luigi (at AEG Germany) and the development of MHEG compatible convertors (Euroclid France).

Acknowledgments

Multicard is the result of four years of team effort within the Esprit project Multiworks. The following team members have directly contributed to the design and development of Multicard : Gianlucca Pancaccini, David Price, François Thorel, Li Xiang Ru, Maria Ahedo, Philippe Dagaud, Nour Didi and Rama Krishna Reddy,

References

[1] Amman B., and Scholl M., Gram: A Graph Data Model and Query Language, in Proc. *ACM Conference on Hypertext ECHT'92,* Milan 1992.

[2] Bieber M., Issues in Modelling a Dynamic Hypertext Interface, *in ACM Hypertext'91 Proceedings, San Antonio, December 1991.* pp 203–218:

[3] Bier E., Goodisman A, Documents as User Interfaces, in *Proceedings of Electronic Publishing EP90,* Ed. Furuta R., Cambrige University Press, 1990. pp 277–290.

[4] Brown P., A Hypertext System for UNIX, *Computing Systems, Vol. 2, No. 1, Winter 1989.* pp.37-53.

[5] ISO/IEC JTC/SC2WG12,working document : "Coded Representation of Multimedia and Hypermedia Information".

[6]ISO 10744 Hypermedia/Time-based structuring Language (HyTime), ISO Standard.

[7] Pearl, A. Sun's Link Service: A Protocol for open Linking. *In Hypertext'89, ACM, Pittsburgh, Pa.,* November 5-7, 1989, pp. 137-146.

[8] Halasz, F. and Schwartz, M. The Dexter Hypertext Reference Model. *In Proceedings of the Hypertext Standardisation Workshop,* J. Moline, D. Benigni, and J. Baronas, Eds., National Institute of Standards and Technology, Gaithersburg, MD 20899, January 16-18, 1990, pp. 95-133.

[9] Malcolm K., Poltrock S., Schuler D., Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise, ACM *Hypertext'91 Proceedings, San Antonio,* December 1991. pp13-24.

[10] Meyrowitz N., The Missing Link: Why we are all doing it wrong, position paper, *Hypertext'87, Univ. of North Carolina,* 1987.

[11] Meyrowitz N., Hypertext Does It Reduce Cholesterol Too?, *Hypertext'89 invited paper, Pittsburgh, PA.,* November 1989, IRIS Technical report 89-9, Brown University, Providence, RI, 1989.

[12] Campbell, B. and Goodman, J.M. HAM: A General-purpose Hypertext Abstract Machine. *In Hypertext'87 Papers, Chapel Hill, NC,* November 13-15, 1987, pp21-32.

[13] Puttress J.J. and Guimaraes N.M., The Toolkit Approach to Hypermedia, *in European Conference on Hypertext ECHT'90,* eds. Rizk A., Streitz N. and André J., Cambridge University Press, 1990, pp. 25-37.

[14] Sherman M., Hansen W., McInerny M. and NeuenDorffer T. Building Hypertext on a Multimedia Toolkit, *in European Conference on Hypertext ECHT'90,* eds. Rizk A., Streitz N. and André J., Cambridge University Press, 1990, pp. 13-24.

[15] Kacmar C. J. and Leggett J. J., PROXHY: A Process-Oriented Extensible Hypertext Architecture, *ACM Transactions on Information Systems,* Vol. 9, No. 4, October 1991, pp. 399-419.

[16] Haan B. J., Kahn P. , Riley V. A. , Coombs J. H., Meyrowitz N. K. , IRIS Hypermedia Services, *CACM,* January 1992, Vol. 35, No.1, pp 36-51.