Techniques For Building Open Hypermedia Systems

Serge Demeyer - Patrick Steyaert - Koen De Hondt

Presented at the first workshop on open hypermedia systems (Edingburg, UK — 1994)

Published in Wiil, U. K. / Østerbye, K. (Ed) "Proceedings of the ECHT'94 Workshop on Open Hypermedia Systems"; Technical report R-94-2038 / Institute for Electronic Systems Department of Mathematics and Computer Science - Fredrik Bajers Vej 7 - DK 9220 Aalborg - Denmark.

Techreport vub-prog-tr-94-11

Programming Technology Lab
PROG(WE)
VUB
Pleinlaan 2
1050 Brussel
BELGIUM

Fax: (+32) 2-629-3525
Tel: (+32) 2-629-3308
Anon. FTP: progftp.vub.ac.be
WWW: progwww.vub.ac.be

Techniques For Building Open Hypermedia Systems

Serge Demeyer - Patrick Steyaert - Koen De Hondt
Programming Technology Lab
Brussels Free University (BELGIUM)
{prsteyae, kdehondt, sademeye}@vnet3.vub.ac.be

ABSTRACT

This paper describes a methodology the authors found very useful in the development of open systems for object-oriented languages, user-interface builders and hypermedia. We promote the idea of "open designs" as being a key factor for success and discuss software engineering techniques useful in implementing such designs.

INTRODUCTION

In today's computer science community, we observe a general trend towards open systems. Important arguments for this trend are the "Buy versus Build" and "Incremental development" approaches as solutions for the inherent difficulties in constructing large software applications [BROO'87]. We perceive this open-systems-trend in areas like operating systems (OLE [MICR'94]), databases (CORBA [CATE'94]), programming languages (CLOS [KI/RI/BO'91]) and many others, including the hypermedia community. We believe it is important to learn from other research on open systems.

There are two important tendencies in open systems research: interoperability and extensibility. The first seeks to define techniques for interchanging information between different systems, the second attempts to build systems able to incorporate new kinds of information. An example for the first tendency is the CORBA-architecture [CATE'94] for interchanging data between different object oriented (database)systems. Typical for the second tendency is the CLOS Meta-object protocol [KI/RI/BO'91] that can be used to install new functionality into the Common Lisp Object System. Within this context, it is interesting to note that most of the experiments in the second tendency use object-oriented techniques.

In the Hypermedia community we encounter proponents of both tendencies as well. Within the interoperability tendency we find HyTime [NE/KI/NE'91], MHEG [CCIT'92] (information exchange standards) Proxhy [KA/LE'91], Multicard [RI/SA'92], MicroCOSM [DAVal'92] (extensible architectures providing inter application hypertext services). In the extensibility tendency we encounter De Bra et al. [DE/HO/KO'92], Hydesign [MA/SC'92] and Dexter [HA/SC'94].

Some people will argue against classifying the Dexter model [HA/SC'94] as an open system[1]. We believe it is. Not only is it defined as a set of important abstractions commonly found in a wide range of hypertext systems, but —more importantly— several people build extensible implementations of the model (e.g. DeVise HyperMedia [GR/TR'94, GROal'94], Amsterdam Hypermedia Model [HA/BU/VA'94]). Nevertheless, it is interesting to observe that the Dexter model is quite different from the others, in the way it is defined "after the facts".

In the remainder of this paper we will report on our experiences during the development of several open systems belonging to the extensibility tendency. Afterwards, we will propose some techniques we have found very useful during this process. Finally, we will return to open hypermedia systems and draw some conclusions.

EXPERIMENTS

As already stated, we are conducting several experiments implementing open systems situated in the extensibility tendency. This sections gives a brief overview of our work and will lead to the concept of "open designs".

The most mature of these experiments is the AGORA framework [STEal'93, STEY'94, CODal'94] used for the exploration of object-oriented languages. In its basic version, AGORA is a reflective, prototype-based language that features a general mixin-based approach to (multiple) inheritance. One of the major innovations of AGORA is that many important features of object-orientation including inheritance, slot access, reification, cloning and inline objects, are introduced by means of message passing rather than by ordinary programming structures. This makes AGORA an extensible programming system that allows the exploration of different object oriented programming paradigms.

1 "It is our hope that the workshop will give the area of open hypermedia systems the same boost as the Dexter model gave the area of monolithic hypermedia system architectures and data models." [Call for participation: Workshop on Open Hypermedia Systems in connection with ECHT'94]

A second experiment investigates what is needed to make user interface builders incrementally refinable [STEal'94]. We start from the assumption that user interface builders are essential tools for the development of modern applications, and argue that the state of the art of the field lacks a way of incorporating new user interface paradigms. Indeed, in analogy to programming paradigms (e.g. functional and imperative programming), it can be observed that different user interface paradigms exist (e.g. direct manipulation, menu driven, navigational). We propose a framework for a higher level user interface builder and show how reflection can be used to install new interface paradigms.

We also investigated the possibilities of object oriented software engineering techniques to build an extensible hypertext system [DEME'94]. The introduction of a path-concept allowed us to "end the tyranny of the link" [HALA'91] and to explore different navigational paradigms (e.g. hard links, virtual links, search and query). Moreover, the notion of "virtual structures" [HAL'91] allows us to manipulate information outside the hypertext (this positions our work in both the interoperability and extensibility tendencies of open systems). All of this is demonstrated in prototype applications (e.g. a Smalltalk browser and an electronic agenda manager).

We find it very important that open systems have the means to incorporate different paradigms. It is not enough that new features can be installed: a higher level kind of extensibility is needed. Skilled users should be able to extend the system in such a way that important functionality can be absorbed in the system and applied by occasional users. On the other hand, it is important to enforce certain design constraints to avoid diverging implementations.

Systems able to incorporate different paradigms and enforce design constraints are systems with an open design. In the following section we will discuss techniques for building such systems.

TECHNIQUES FOR OPEN DESIGNS

One of the lessons learned from the above experiments is that incremental refinement is well suited for adapting open designs to different paradigms. *Refinement* enables code reuse which is important for the skilled user extending the system. *Incremental* refinement promotes design reuse, as the different steps of the refinement process are separated from each other.

Isolated from each other, the following techniques are unable to produce this notion of incremental refinement. It is only when used in conjunction that they will grow to full potential.

Object Oriented Frameworks

According to [WIRF'90], an object-oriented framework is a skeleton implementation of an application or application subsystem in a particular problem domain. It is composed of concrete and abstract classes and provides a model of interaction or collaboration among the instances of classes defined by the framework. As frameworks themselves are extended by subclassing [JO/FO'88, HE/HO/GA'90], they typically impose an incremental structure on the refinement process. Important in this context is that a framework provides two interfaces: 1) the framework internal interfaces that specify how the framework can be reused and 2) the framework external interface to clients of applications or application subsystems generated with the framework.

Object-oriented frameworks alone are not sufficient to express open systems. The emphasis of object-oriented frameworks is on code-reuse, not on design-reuse. The consequence is that while incrementally modifying a framework one typically can adapt it to problem domains outside the initial intention of the framework such that the contract, or the external interface, with the clients of the initial framework is broken. Plain reuse of a framework is too much of an uncontrolled mechanism. Therefore other mechanisms such as open implementations need to be studied.

Open Implementations

Open implementations distinguish themselves from plain 'black-box' or fixed implementations by giving a structured, 'open-ended' access to their implementation.

> ***Open Implementations*** *[RAO91] : A system with an open implementation provides (at least) two linked interfaces to its clients, a base-level interface to the system's functionality similar to the interface of other such systems, and a meta-level interface that reveals aspects of how the base-level interface is implemented.*

The idea is that a user of an open implemented system can, by means of the meta-level interface, have a substantial influence on the implementation, and accordingly, the behaviour of the system. In an open implementation the meta-level interface specifies points where the user can provide alternative implementations. Such an alternative implementation can differ from the default implementation of the system in performance characteristics, or it can alter the behaviour of the system, or it can extend the system with new behaviour. The extent to which the behaviour of the system can be altered, or extended, depends on the meta-level interface and its link to the object-level interface. In the most literal sense the meta-level interface only gives access to implementational issues of the system. In a more liberal sense it also allows extending the behaviour of the system. Thus, rather than defining a single system, an open implementation defines a entire design space of systems.

The notion of open implementations was introduced by Rao in [RAO'91], where an open implementation is given for a windowing system that allows the exploration of different window system behaviours and implementations. The base-level interface of the windowing system is, obviously, an interface that allows the opening and closing of windows, dragging, generating pictures in windows, etc. The meta-level interface allows, for example, for the definition of new windowing relationships (such as window, sub-window relations). The so obtained architecture describes an entire design space of related windowing systems. As the windowing system itself was described in the object-oriented paradigm, the open implementation of the windowing system took the form of a meta-object protocol.

Meta-Object Protocols

Meta-object protocols are a particular form of open implementations that make use of object-oriented techniques to open up their implementation. With a meta-object protocol, a software system's implementation architecture is made explicit and open in terms of objects and their interactions. The objects that constitute the architecture (e.g. objects that represent windows, windowing relationships, etc. in a windowing system) are called meta-objects. The relation between meta-objects is formalised by means of protocols. A protocol describes the responsibilities of each meta-object in the architecture.

A meta-object protocol is an interface to a system providing users of that system the ability to incrementally modify the system's implementation and behaviour. For each meta-object a default class is given that lays down the default behaviour of the system. These default behaviours can be incrementally modified making it possible to adjust the system to a different point in the design space.

Meta-object protocols for windowing systems [RAO'91], programming languages [KI/RI/BO'91], compilers [RODR'92], and operating systems [YO/TE/TO'89] have been documented in the literature.

So, meta-object protocols rely on the same object-oriented techniques as object-oriented frameworks. They differ from 'plain' object-orientation in their emphasis on protocols. The documentation of protocols is typically done in the form of a natural language description, but can also be more formally expressed with for example contracts [HE/HO/GA'90] or class invariants [MEYE'88]. Ensuring the fact that the protocols of the meta-objects are respected is a research topic [STAal'92]. Protocols limit the applicability of an open implementation to a particular design space. This is an important issue in expressing open systems since it defines the degree to which clients of the open system can rely on the existence of some fixed interface to the open system.

WHAT ABOUT OPEN HYPERTEXT SYSTEMS ?

In the sections before, we have promoted state-of-the-art software engineering techniques (namely the combination of object oriented frameworks and open implementations) to build open hypermedia systems. Applying ideas from the software engineering world (especially object orientation) is not new to the hypertext community: Intermedia advocated the use of object oriented languages [MEYR'86] and later object oriented databases [SM/ZD'87] in building the system; Lange [LANG'90], Schütz & Streitz [SC/ST'90], WEBS [MO/PA'92], ABC [SC/SM/SM'93] moulded their model in a class hierarchy; DeVise HyperMedia [GR/TR'94, GROal'94] and Hydesign [MA/SC'92] implemented their systems on top of an object oriented database. We expect the hypermedia researchers will continue to have an open mind for new ideas.

However, prior to the construction of open hypermedia systems, careful analysis of the problem domain is required. We propose a requirement analysis in two phases: paradigm identification and meta-object definition.

Paradigm Identification

As stated before, we find it very important for open systems to have the means to absorb different paradigms. This way, users are able to choose the paradigm best suited to their needs. Small variations can be incorporated by modifying (extending) the internal implementation.

So what are the paradigms in the hypermedia domain ? We feel that the navigational access to information (commonly called "browsing") is the most essential manifestation of a hypermedia system, so we focus on this aspect. In his keynote address [HALA'91], Halasz proposes a number of dimensions of the hypermedia space we can adopt as navigational paradigms: the navigators vs. the architects, the literalists vs. the virtualists, the card sharks vs. the holy scrollers. Navigators jump from node to node through the information space, while architects manipulate a global overview (a map) of the overall structure. Literalists navigate by following explicit structural connections (hard links), while the virtualists use implicit structures (search and query). Card sharks partition the information in fixed-size information chunks (cards), while holy scrollers navigate by "flying" within and between lengthy documents.

But we should not limit our thoughts to navigation alone. The Amsterdam Hypermedia Model [HA/BU/VA'94] states that introducing the notion of time —needed to synchronise multimedia presentations— opens a wide range of possibilities. Streitz [STRE'94] discusses the effects of tele-cooperation using the well-known time-space taxonomy. Probably, there will be other areas suggesting other paradigms as well. Closely related is the work of Wiil and Leggett [WI/LE'93].

Meta-object Definition

Once the paradigms are identified, the conception of the necessary abstractions becomes feasible. We illustrate this phase with two examples. The first is the well-known anchor abstraction of the Dexter model, the second is a pathabstraction we use in our open hypermedia system.

One of the goals of the Dexter model [HA/SC'94] was to permit nodes ("components") to have arbitrary contents, stored in the within-component-layer. The storage-layer is then responsible for managing the structure of the hypertext (the links). To accomplish this, the storage-layer needs specifying substructures of components. The anchor-abstraction was introduced to ensure that the storage-layer remains independent of the within-component layer. Adding a new data-type to a Dexter hypermedia system requires the definition of a new anchor type, as is described in [GR/TR'94].

With others [HALA'91] we feel that the importance of links is overestimated in the hypertext community. In [DEME'94] we report on an experiment were an open hypermedia model is used to exploit both the possibilities of hard links and search and query navigation (e.g. the literalists vs. the virtualists). This is accomplished through the notion of paths. Every anchor operation will eventually be passed to the path, which is responsible to resolve the anchor, which means that paths can (and will) determine the navigation potential of the hypertext network. We have specialised the path mechanism to support both explicit (links) and implicit (queries) structure.

FUTURE WORK

Currently, we are building an open hypermedia system following these principles. Virtual structures allowed us to store and retrieve information external to the hypertext and we introduced the path concept to exploit different navigational paradigms. We refer to [DEME'94] for a detailed discussion of the matter. Future work includes the incorporation of other paradigms, especially in the area of Computer Supported Cooperative Work. We are also working on use of reflection facilities in an open hypermedia system, as this direction has proven its other experiments [STEal'94, STEY'94].

CONCLUSION

From our background we claim that the construction of open hypermedia systems should be preceeded by the identification of paradigms and the definition of meta-objects. These phases will lead to the necessary delineation of a complete space of systems, all of which should be covered by the open design. To help in the implementation of the open design, software engineering techniques like object oriented frameworks and open implementations are required.

REFERENCES

[BROO'87] Brooks, F. P. "No Silver Bullet. Essence and Accidents of Software Engineering"; IEEE Computer, April '87

[CATE'94] Catell, R. G. G. (Editor) "The Object Database Standard: ODMG'93 - Release 1.1"; Morgan Kaufman Publishers '94

[CCIT'92] CCITT Press Release on MHEG, December 1992.

[CODal'94] Codenie, W. / De Hondt, K. / D'Hondt, T. / Steyaert, P. "Agora: Message Passing as a Foundation for Exploring OO Languages"; Submitted to SIGPLAN Notices. See also WWW & ftp {progwww, progftp}.vub.ac.be

[DAVal'94] Davis, H. / Hall, W. / Heath, I. / Hill, G, / Wilkins, R. "Towards An Integrated Information Environment With Open Hypermedia Systems"; Proceedings of the ACM ECHT'92 Conference (November 30 - December 4, Milano, Italy)

[DE/HO/KO'92] De Bra, P. / Houben, G. J. / Kornatzky, Y. "An Extensible Data Model for Hyperdocuments"; Proceedings of the ACM ECHT'92 Conference (November 30 - December 4, Milano, Italy)

[DEME'94] Demeyer, S. "Virtual Hypertext based on Paths and Warm Links" (Technical report); See WWW & ftp {progwww, progftp}.vub.ac.be

[GR/TR'94] Gronbaek, K. / Trigg, R. H. "Design Issues for a Dexter-Based Hypermedia System"; Communications of the ACM, Vol. 37(0), February '92. Also in Proceedings of the ACM ECHT'92 Conference (November 30 - December 4, Milano, Italy)

[GROal'94] Gronbaek, K. / Hern, J. E. / Madsen, O. L. / Sloth, L. "Cooperative Hypermedia Systems: A Dexter-Based Architecture"; Communications of the ACM, Vol. 37(0), February '92. Also in Proceedings of the ACM HT'93 Conference (November 14-18, Seattle, Washington)

[HA/BU/VA'94] Hardman, l. / Bulterman, D. C. A. / Van Rossum, G. "The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model"; Communications of the ACM, Vol. 37(0), February '92

[HA/SC'94] Halasz, F. / Schwartz, M. "The Dexter Hypertext reference Model"; Communications of the ACM, Vol. 37(0), February '92

[HALA'91] Halasz, F. "Seven issues revisited". Slides from the ACM Hypertext '91 Conference Keynote speech (December 15-18, San Antonio, Texas)

[HE/HO/GA'90] Helm, R. / Holland, I. M. / Gangopadhyay, D. "Contracts: Specifying Behavioral Compositions in Object-Oriented Systems"; ACM

ECOOP/OOPSLA'90 Conference Proceedings (October 21-25, Ottawa, Canada)

[JO/FO'88] Johnson, R. E. / Foote, B. "Designing Reusable Classes" in Journal of Object-Oriented Programming 1(2), February '88 p. 22 - 35

[KA/LE'91] Kacmar, C. J. / Legget, J. J. "PROXHY: A Process-Oriented Extensible Hypertext Architecture"; ACM Transactions on Information Systems, Vol. 9 (4), October '91

[KI/RI/BO'91] Kiczales, G. / Rivières, J. / Bobrow, D. G. "The Art of the Metaobject Protocol"; The MIT Press 1991

[LANG'90] Lange, D. B. "A Formal Model of Hypertext"; Proceedings of the 1990 NIST Hypertext Standardization Workshop (January 16-18, Gaithersburg, MD)

[MA/SC'92] Marmann, M. / Schlageter, G. "Towards a Better Support for Hypermedia Structuring: The HyDESIGN Model"; Proceedings of the ACM ECHT'92 Conference (November 30 - December 4, Milano, Italy)

[MEYE'88] Meyer, B. "Object Oriented Software Construction"; Prentice Hall, 1988

[MEYR'86] Meyrowitz, N. "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework" in ACM OOPSLA'86 Conference Proceedings (Sept 29 - October 2, Portland, Oregon)

[MICR'94] Microsoft "How To Apply OLE 2 Technology in Applications"; Available through anonymous ftp at ftp.microsoft.com /developer/drg/ole-info/

[MO/PA'92] Monnard, J. / Pasquier-Boltuck, J. "An Object-Oriented Scripting Environment for the WEBSs Electronic Book System"; Proceedings of the ACM ECHT'92 Conference (November 30 - December 4, Milano, Italy)

[NE/KI/NE'91] Newcomb, S. R. / Kipp, N. A. / Newcomb, V. T. "Hytime: Hypermedia / Time-based Document Structuring Language"; Communications of the ACM, Vol. 34(11), November '91

[RAO'91] Rao, R. "Implementational Reflection in Silica"; ECOOP'91 Proceedings, Lecture Notes in Computer Science, P. America (Ed.), pp. 251-267, Springer-Verlag, 1991

[RI/SA'92] Rizk, A. / Sauter, L. "Multicard: An open Hypermedia System"; Proceedings of the ACM ECHT'92 Conference (November 30 - December 4, Milano, Italy)

[RODR'92] Rodriguez JR., L. H. "Towards a better understanding of compile-time mops for parallelizing compilers"; Proceedings of the IMSA'92 Workshop on Reflection and Meta-level Architectures (1992).

[SC/SM/SM'93] Schackelford, D. E. / Smith, J.B. / Smith, F.D, "The Architecture and Implementation of a Distributed Hypermedia Storage System"; ACM Hypertext '93 Conference Proceedings (November 14-18, Seattle, Washington USA)

[SC/ST'90] Schütt, H. / Streitz, N. "HyperBase: A hypermedia engine based on a relational data-base management system"; Rizk, A. / Streitz, N. / André, J. "Hypertext: concepts, systems and Applications - Proceedings of the European Conference on Hypertext" (November, Versailles, France)

[SM/ZD'87] Smith, K. E. / Zdonik, S. B. "Intermedia: A case study of the Differences Between Relational and Object-Oriented Database Systems" in ACM OOPSLA'87 Conference Proceedings (October 4-8, Orlando, Florida)

[STAal'92] Stacy, W. / Helm, R. / Kaiser, G. E. / Meyer, B. "Ensuring Semantic Integrity of Reusable Objects (Panel)"; ACM OOPSLA'92 Conference Proceedings (October 18-22, Vancouver, Canada)

[STEal'93] Steyaert, P. / Codenie, W. / D'Hondt, T. / De Hondt, K. / Lucas, C. / Van Limberghen, M. "Nested Mixin-Methods in Agora"; ECOOP '93 European Conference on Object-Oriented Programming, Springer-Verlag. See also WWW & ftp {progwww, progftp}.vub.ac.be

[STEal'94] Steyaert, P. / De Hondt, K. / Demeyer, S. / De Molder, M. "A Layered Approach To Dedicated Application Builders Based On Application Frameworks"; Submitted to OOIS'94 (International Conference on Object-Oriented Information System), London, UK, 19-21 December 1994. See also WWW & ftp {progwww, progftp}.vub.ac.be

[STEY'94] Steyaert, P. "Open Design of Object-Oriented Languages: A foundation for Specialisable Reflective Language Frameworks"; Phd. thesis Vrije Universiteit Brussel, 1994. See also WWW & ftp {progwww, progftp}.vub.ac.be

[STRE'94] Streitz, N. A. "Putting Objects to Work: Hypermedia as the Subject Matter and Medium for Computer-Supported Cooperative Work"; ECOOP'94 Proceedings, Lecture Notes in Computer Science (821), Tokoro, M. / Pareshi, R. (Ed.), pp. 183-193, Springer-Verlag, 1991

[WI/LE'93] Wiil, U. K. / Legget, J. J. "Concurrency Control in Collaborative Hypertext Systems"; ACM Hypertext '93 Conference Proceedings (November 14-18, Seattle, Washington USA)

[WIRF'90] Wirfs-Brock, A. "Panel Designing Reusable Designs: Experiences Designing Object-Oriented Frameworks"; Sigplan Notices Special Issue OOPSLA-ECOOP'90 Addendum to the Proceedings (Jerry L. Archibald and K.C. Burgess Yakemovic eds.), pp.19-24, 1990.

[YO/TE/TO'89] Yokote, Y. / Teraoka, F. / Tokoro, M. "A reflective architecture for an object-oriented distributed operating system"; Proceedings of European Conference on Object-Oriented Programming (ECOOP) (July 1989).