Table of Contents

The Zypher Experiment.....1

Open Hypermedia.....6

Object-Oriented Software Engineering.....43

The Zypher Design 69

The Zypher Software Artefact.....70

Data Structures For An Interoperable Hypermedia Framework.....76

A Design Space For A Hypermedia Framework.....93

A Layered Hypermedia Framework.....106

Tailorability In An Open Hypermedia Framework.....115

Protocols in an Open Hypermedia Framework.....138

Conclusion & Appendices.....145

Conclusion 146

Appendices 154

Abstract

The dissertation concerns a study of state of the art object-oriented software engineering applied within the domain of open hypermedia systems. The results of this study are discussed within the context of a software artefact named Zypher.

The scientific contribution of this work is situated in the domain of object-oriented software engineering. The contribution is a proper combination of frameworks and meta-object protocols, which are two promising techniques in object-oriented software engineering. We show that, when combining both approaches, explicit representations of framework contracts are part of a meta-object protocol. This insight is valuable in the design of meta-object protocols.

The design of meta-object protocols is crucial in the development of large scale software systems, because it leads to greater flexibility. Flexibility is a key requirement, since large scale software systems are bound to satisfy ever changing demands, hence must be adapted continuously. This flexibility requirement is very prominent in open hypermedia, the research domain where the results are validated experimentally.

The archetype for the kind of software envisioned is an open hypermedia system supporting a group of engineers in a design and manufacturing process. Such a system must handle the management of several tens of gigabytes of data stored in different devices, contained in millions of individual design elements maintained in various formats, organised in densely interconnected graphs representing numerous relationships and —probably the most difficult requirement— constantly evolving while the design and manufacturing process proceeds. Without a doubt, flexibility is a key requirement in these engineering support systems.

We propose tailorability as a means to cope with the flexibility requirement, and we identify at least three levels of tailorability to handle different kinds of flexibility. Domain level tailorability is needed to adapt a software system to changing needs, such as the incorporation of new data formats. System level tailorability copes with requests about the system performance, data security or information integrity. Configuration level tailorability handles demands such as cross-platform portability and run-time reconfiguration.

We show that object-oriented software engineering techniques are valuable for the development of tailorable systems. More precisely, domain level tailorability can be achieved by building an object-oriented framework. Extending the framework with a meta-object protocol attains system level tailorability and yet another meta-level implementsconfiguration level tailorability.

Acknowledgements

I thank my promotor Prof. Theo D'Hondt. He pushed me to define this meta-object protocol in my Zypher framework and I confess that I doubted whether there would be one. Like all good promotors, he was right and I was wrong. I am forever in depth with Patrick Steyaert. He was my mentor during the painful days and tempered my assaults until they fitted the thesis format. Patrick also advised me to form a circle of people that could help me in filtering ideas: Koen De Hondt, Roel Wuyts, Wim Codenie and Wolfgang De Meuter, thanks for helping me out. Valuable, among others in proof-reading were Carine Lucas, Kim Mens and Marc Van Limbergen. Karel Driesen wasn't around when I finished the text, but I still owe him a lot for all the lively discussions after a pint of beer and a borrowed cigarette. To all the other people at the Programming Technology Lab, the Computer Sciences Department and the University that I worked with during these years, thanks for being nice colleagues. I would like to mention Viviane Jonckers for being an outsider with valuable comments, and Eddie Vandijck for being a believer.

Using internet I was able to manage some international contacts as well. Thanks to Uffe Kock Wiil for inviting me to organise the second open hypermedia workshop; it was a great occasion to enter the open hypermedia community. Thanks to Kasper Østerbye (and Uffe) for sending me drafts of what was to become the "Flag Taxonomy on Open Hypermedia Systems" and the helpful discussions over e-mail afterwards. Helen Ashman provided me with a draft of a paper discussing requirements lists for open hypermedia systems, which was a source of inspiration. It was good to know that I was not the only one combining design patterns with hypermedia, Alejandra Garrido does parallel work. Eric Steegmans forced me to reconsider the white-box versus black-box debate, which provided me with some valuable new insights.

I got into the hypermedia field from a computer supported co-operative work angle. I am especially grateful to Michel Tilman, for pointing me in the Hypermedia domain. I would like to thank Dirk Kenis for running the DSide project with all its ups and downs and for helping me with the apposition of this thesis. I am grateful to Wim Lybaert for working on a combination of Smalltalk and HTML/HTTP to make a version of DSide that was accessible via the web. Niels Boyen and Thierry Marchant helped during the development of other DSide prototypes.

The ARGO was an environment that confronted me with the problem how object-oriented ideas survive in real projects; to all the people at the AIV (especially Martine Devos), thanks for keeping me out of the ivory tower. Some people at OOPartners where helpful for both material and spiritual support, thanks to Wilfried "Wiffel" Verachtert, Arlette Vercammen and Peter Oyserman.

I thank my parents as well. First of all for giving me the opportunity to study, it was the first step towards this thesis. Also for all the love and support during these 30 years. My brothers, the rest of my family and family in law provided for the necessary distraction once in a while. Special thanks go to my brothers for the nice artwork in this dissertation.

And finally I would like to thank my wife, Ann. I know she doesn't believe me, but I wouldn't have finished this thing without her.

Serge Demeyer

July, 1996

PART 1

The Zypher Experiment

Introduction....2
Road Map....5
Chapter 1: Open Hypermedia 6
Hypertext & Hypermedia....7
Open Hypermedia....22
The Zypher Perspective....30
Chapter 2: Object-Oriented Software Engineering.... 43
Incremental Development....44
Object-Oriented Frameworks....46
Meta-object Protocols....56
The Zypher Contribution....63

Introduction

Nowadays, a considerable amount of effort is spent on the design of open systems. This phenomenon can be observed in operating systems (e.g. UNIX and OS/2), databases (e.g. Exodus and CORBA), inter application communication (e.g. OLE and OpenDoc), tailorable software (e.g. Emacs and AutoCAD), programming languages (Smalltalk and CLOS) and last but not least —certainly with the advent of the world-wide web— hypermedia.

A closer examination of these fields reveals that there are a number of common themes underlying all these efforts. A first one is *interoperability*, an idea that starts from the perception that software systems are getting bigger and bigger, but rarely incorporate new ideas. That is, in most occasions each major release clones features from other complementary systems. Rather than ending up with a huge system that behaves like a common denominator of many other systems, one wants users to choose their favourite and specialised systems and provide some form of co-operation facilities between them. A second theme in the design of open systems is *extensibility*, where system designers acknowledge that a single application will never be able to serve all the needs of all of its users. Rather than building static systems, system designers want to provide some form of extensibility so that end users can extend the systems to their needs. A final theme is *distribution*, where people want to share resources by connecting workstations and peripheral hardware through computer networks.

From an end users perspective the interest in these themes is justified, since they try to provide quite simple solutions for quite simple problems. However, when software designers must incorporate these solutions into their systems, they are faced with a number of complex issues and questions. Each software designer comes up with different answers and this again gives rise to the enormous pool of standards, languages and protocols we find in today's software systems. To cope with such a chaotic diversity systems get bigger and bigger, so consume more resources, contain more bugs and are more expensive to purchase. Something needs to be done to stop this evolution and this explains the motivation for the numerous efforts on designing open systems: they are widely recognised as an attempt to address this problem.

This Ph.D. research is a contribution to the same attempt. As Ph.D. research is well suited for exploring the more risky aspects of open systems, we have examined state of the art techniques in other domains, applied these techniques to the design of a particular open system situated in the hypermedia domain, identified combinations that work, clarified why these combinations work and tried to generalise the results afterwards.

The techniques we applied in this work stem from the domain of software engineering. In this domain it is recognised (see among others the influential article "No Silver Bullet: Essence and Accidents in Software Engineering" [Brooks'87]) that software systems are more difficult to build than any other kind of systems built in any other engineering discipline. This is motivated from the observation that software systems are intrinsically complex, must deal with the non-conformity of numerous human institutions and systems, is subject to a constant demand for change and is inherently invisible thus hard to conceive. One of the ways to deal with the inherent difficulties of constructing software systems is called *incremental development*. In natural systems like the human eye or brain, all

complexity is the result of evolution, so proponents of incremental development feel that complexity in software systems should be the result of evolution as well. So in the incremental development approach, a software system is regarded as a flexible construction that is constantly adapted, rather than a static device that can not be changed once it has left the design table.

The software engineering community puts forward various approaches towards incremental development. Two of them —*object-oriented frameworks* and *meta-level abstraction*— have proven their value in the development of commercially available systems. The former is a design specification technique for reusable designs in a particular problem domain. The latter is an abstraction technique that provides specific interfaces to manage system mutations.

From these short descriptions, it follows that both techniques have different goals — frameworks aim at design reuse while meta-level abstractions aim at adaptability. Yet, both techniques are applicable in an incremental development approach — frameworks because the same design is likely to be reused in future instances of a system, meta-level abstractions because they facilitate system mutations. So we may conclude that both techniques are *complementary* with respect to the incremental development approach.

Returning to the main theme of the dissertation, it seems that the incremental development approach is quite suitable when constructing open systems. Open systems address the chaotic diversity that follows from the numerous standards, languages and protocols that exist in today's software systems. Natural systems cope with chaos by evolution, so incremental development is likely to be applicable to the design of open systems. This brings us to the first research hypothesis formulated below.

Research Hypothesis 1

> *Object-oriented frameworks and meta-level abstraction are two complementary techniques in the design of open systems.*

This hypothesis embodies the main ideas that have been driving our research. On the one hand, we want to understand how complementary both techniques are and —based on this insight— propose a design methodology as a proper combination of object-oriented frameworks and meta-level abstraction. On the other hand, we want this design methodology to address problems and issues that are encountered in the design of real systems.

In the domain of software engineering, it is common practice to validate techniques experimentally [NationalAcademy'94] — researchers build prototype systems for a particular application domain to investigate benefits and drawbacks of a technique, and generalise these results to a broader set of application domains afterwards. Such a prototype system is called a software artefact and in our dissertation this is role is played by the Zypher system.

In such an experimental validation, the choice of the target application domain is of prime importance: the software artefact's application domain must be *representative* for a larger set of application domains. The idea is that the prototype system should be small enough so that it can be built and adapted rapidly, yet scalable for the kind of problems encountered in real systems. When the software artefact's application domain is representative for a larger set of application domains, the results obtained in the specific software artefact can be generalised to the larger set of applications.

We have chosen our prototype system to be an open hypermedia system. Open hypermedia is about the management of large, complex and heterogeneous information structures and as such, open hypermedia covers the common themes underlying all open systems — interoperability, extensibility and distribution. So it is likely that the results of an open hypermedia software artefact can be generalised to the larger set of open systems, which brings us to the second research hypotheses formulated below.

Research Hypothesis 2

> *Open hypermedia systems are representative for the larger set of open systems.*

The generic set of open systems is quite large. Besides open hypermedia systems, it comprises domains like operating systems, database systems, inter application communication standards, tailorable software systems and programming languages. Yet we chose hypermedia as the representative element and we had a very special motivation to do so.

The archetype for the kind of information to be managed by an open hypermedia system is the engineering documentation used in the design and manufacturing process of airplanes. Such a system must handle the management of several tens of gigabytes of data stored in different devices, contained in millions of individual design elements maintained in various formats, organised in densely interconnected graphs representing numerous relationships and —probably the most difficult requirement— constantly evolving while the design and manufacturing process proceeds.

Open hypermedia is recognised as a *technological cornerstone* for systems supporting airplane manufacturing. That is, it is possible to construct such a system without incorporating open hypermedia, but neglecting the technology implies the loss of valuable concepts and strategic advantages.

If open hypermedia is recognised as an invaluable ingredient in airplane manufacturing, it may support other engineering disciplines as well. In the context sketched above, a logical candidate is the software engineering discipline, which brings us to the third research hypothesis formulated below.

Research Hypothesis 3

Open hypermedia is a technological cornerstone for supporting software engineering.

This dissertation is concerned with the examination of the three research hypotheses. We will devise a model for open hypermedia systems that can be generalised into a model for open systems; we will propose a design methodology based on a proper combination of object-oriented frameworks and meta-level abstraction and we will experiment with the notion of a framework browser as a tool that supports the management of object-oriented frameworks.

Road Map

The dissertation is divided in three parts. The first part presents an overview of the important research fields where this work found its roots. It starts with an overview of open hypermedia, including a generic model for open hypermedia systems and a scenario for a hypothetical framework browser. The former is based on the notion of a design space with three levels of tailorability. This part of the dissertation supports the second research hypothesis and introduces some arguments for the third hypothesis (see "The Zypher Perspective" - p.30). Next follows a survey of the two investigated techniques (object-oriented frameworks and meta-object protocols) followed by a discussion on how those techniques can be used to build an open system — that is a system that adheres to the principles argued in the design space with three levels of tailorability. This part of the dissertation introduces the main arguments for the first research hypothesis (see "The Zypher Contribution" -p.63).

The second part describes the software artefact used in our experiments, i.e. it provides a description of the Zypher framework in design pattern form. In this part, each element in the design and its contribution to the overall framework is explained. The design patterns are written to stand on themselves, so many of the ideas and argued in the first part will be repeated here (i.e. the design space, the meta-object protocols, …). The second part serves two purposes. Besides being a description of how we applied object-oriented frameworks and meta-object protocols in the design of the Zypher framework (i.e. the experimental validation of the first research hypothesis), it also reports on how we build a prototype of a framework browser (i.e. the experimental validation of the third research hypothesis).

Finally, the last part includes the conclusion and appendices. The conclusion reconsiders the three research hypotheses in the light of all that precedes and review the results from a broader perspective. The appendices includes the apposition —which is an necessary part of a Ph.D. dissertation at our university— and readers aids like a reference list, an index, a list of figures and a table of contents.

While the first part is quite conventional, the idea of non-linearity (hypermedia remember) is crucially important in the second part, i.e. the design pattern description of the Zypher framework. Indeed, the real power of design patterns lies in the idea of overlapping many of them in a single space. Overlapping patterns include several mutual references, hence the importance of non-linearity. *Nevertheless we want to stress that for a first tour, the sequential path is probably the best. Only for an in depth study, the referencing scheme is essential to understand how all patterns work together to achieve the overall design.*

Of course this text is available in hypertext format as well. Check the Zypher home page *http://progwww.vub.ac.be/prog/pools/Zypher/zypher.html*, to read the most recent version and other material created as part of the Zypher project.