Web-based System Evolution in Model Driven Architecture

Feng Chen*, Hongji Yang, Hong Zhou, Bing Qiao
Software Technology Research Laboratory
De Montfort University, Leicester, England
Fengchen, hyang, hongzhou, bqiao@dmu.ac.uk

Huifang Deng
School of Computer Science and Enginnering
South China University of Technology, China
hfdeng@scut.edu.cn

Abstract

The complexity and size of commercial Web-based systems present a grand challenge to the traditional methodology of software evolution. The booming development of Web related technologies complicates the situation. This research presents a unified solution to Web-based system evolution, which consists of three components: Web-based systems understanding, Web-based systems representation and evolvable Web Application Framework. A comprehensive case study will be given to evaluate the proposed solution in different aspects. Conclusion is drawn based on analysis, which verifies the feasibility of the proposed solution. Further research areas are also discussed.

Index Terms*: Software engineering, Software maintenance, Modelling, Architecture*

1. Introduction

With the explosively exponential growth of Web application, Web technologies have evolved from loosely connected sets of HTML pages to highly organised Web sites and to dynamic Web applications. Hence, more and more existing Web sites turn to be legacy systems. The legacy Web Site is always very complicated in the real application projects. The research is hence based on some fairly innocuous assumptions [12]: Web-based systems are complex software-intensive systems; one efficient way to manage complexity is to abstract and model them; multiple models represent different viewpoints at different levels of abstraction; and UML is the de facto modelling language.

In this paper, a unified approach to Web-based system evolution is proposed in the context of Model Driven Architecture (MDA). The rest of this paper is organised as follows: Section 2 gives an overview of model driven re-engineering. Section 3 presents the infrastructure, framework and process of developing/evolving Web-based systems. Section 4 presents a case study with tool support. Section 5 discusses the related work and Section 6 concludes the paper and makes a discussion on possible future research.

2. Model Driven Re-engineering

Generally, a model can mean an abstraction and representation of the important factors of a complex reality, which is different from the thing it models, yet has the same main characteristics as the original. Experienced developers often invest more time in building models than in writing code. Well-constructed models make it easier to deliver large, complex systems on time and within budget.

Model Driven Architecture (MDA) is a standard produced by the Object Management Group (OMG). The MDA specification puts emphasis on different level of models, including Computational Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM). These models should be machine-readable that can be accessed repeatedly and automatically transformed by tools into schemas, code skeletons, test harnesses, integration code, and deployment scripts for various platforms [9].

The concept of Model Driven Engineering (MDE) is emerged as a generalisation of the MDA approach for software development. In [8], MDE is defined on the base of MDA by adding the notion of software development process and modelling space for organising models. Model-driven engineering is a subset of system engineering in which the process heavily relies on the use of models and model engineering [5].

Nowadays, the MDE research community and the reverse engineering community have a lot to share because models are cornerstones of both disciplines [4]. In fact Modelling and Reverse Engineering both refer to the activity of creating descriptive models. Models can be used either to specify a system to be built, or to describe an existing system. This leads to the distinction between specification models and descriptive models [14]. New systems are produced from specification models, while descriptive models are produced from existing systems.

3. The Proposed Approach

This research is focused on establishing a general framework and methodology to facilitate the evolution of Web-based systems. Figure 1 shows an overall picture of the research methodology [12].

978-1-4244-2790-1/08/$25.00 ©2008 IEEE

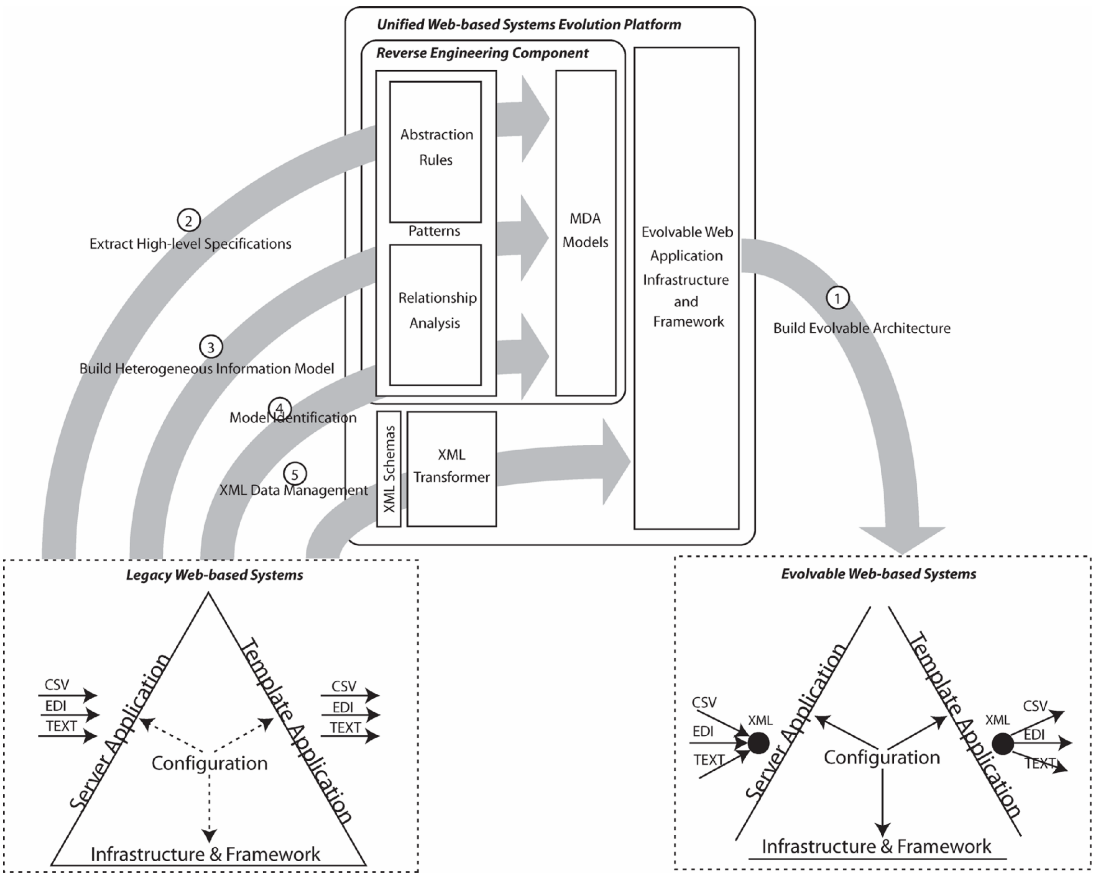

Figure 1**. Methodology of Building Evolvable Web-based Systems**

The Left of Figure 1 shows the architecture of current Web-based systems which can be divided into such components as: server application, template application, configuration, communication and infrastructure/framework.

The right of Figure 1 shows the architecture of Web-based systems built on proposed Web Application Framework (WAF) and supporting infrastructure, where the three components of server application, template application and infrastructure/framework are only connected via configurations. The configuration files of those systems not only establish relationships between different components, but provide information for instantiating classes to be used at runtime.

In the middle of Figure 1 are the techniques used for transforming existing Web-based systems to the proposed WAF and supporting infrastructure. Abstraction, Relationship Analysis and XML transformation are three techniques to target different parts of a Web-based system. Abstraction is applied on application functions/methods to extract specifications of higher level abstraction. Relationship Analysis is effective in analysing heterogeneous information to produce a set of closely-related nodes. The matching between domain knowledge base and source program can only be carried out with a thorough understanding of relationships between components in both code and domain levels. XML transformation is the centre of data communication with external systems.

3.1. Process Model

The proposed approach to building evolvable Web-based systems will be elaborated as numbered elements shown in Figure 1:

1. Build Evolvable Framework. The purpose of reengineering existing Web-based systems is to add extensibility and maintainability supported by the infrastructure and Web Application Framework.
2. Extract High-level Specifications. It introduces reverse engineering methods with strong calculation/control logics.
3. Build Heterogeneous Information Model. It introduces data mining based reverse engineering of heterogeneous information with various link types to construct a representation of a Web-based system.
4. Build Multi-layer Models. Reverse engineering techniques rely on successful pattern recognitions. Domain patterns, system patterns and integration

patterns are used to identify models from existing Web-based systems.

5. Build XML Data Management. An evolvable system needs an extensible and interchangeable data representation as well as an implementation of those logics in modern programming languages.

3.2. Evolvable Web Application Framework

Framework is an important method for large granularity software reuse. A successful Web Application Framework and supporting infrastructure are built to provide evolvable target architecture for existing Web-based systems. General Management Information System Architecture (GMISA) [7] is designed for above purpose.

GMISA introduces a general framework based on the enterprise application, which extended MDA development with a domain general framework. Here, PIM is divided into domain general PIM and enterprise special PIM. Domain general PIM directly transform into code by Framework Code Generator (FCG). Enterprise special PIM can be transformed into PSM and then into code by MDA tools. The merits of GMISA are that the layers are independent with clear responsibility and they adapt to group development and team cooperation. GMISA provides standard software architecture for a family of products that can be configured, extended, and customised to customer-specific solutions.

3.3. Modelling Languages

To support the analysis of complex systems, a system description is made of numerous models. Each model represents a different level of abstraction. The hierarchical structure provides ways to organise the models, to build larger models from smaller one, which is important techniques to modularise model complexity.

The proposed approach has defined a set of models that separate concerns and partition the complexities of the total system to comprehend both the problem domain and the software systems at different levels of abstraction and from different points of view. To accord with MDA, the spectrum of modelling language consists of Common Modelling Language (CML), Architecture Description Language (ADL), and Domain Specific Modelling Language (DSML). Rules for model transformation and abstraction can be found in [2, 3, 12].

4. Case Study with Tool Support

The application used to demonstrate the proposed Web Application framework is an online pet store. The pet store currently sells pets of three categories, each of which has a default delivery plan, used for most customers, but occasionally a particular customer may vary this slightly. The delivery plan for foreign customers might need extra work in accordance with export regulations.

To appeal new visitors and retain existing customers, it must be possible to change the presentation of the site to another style without changing the basic functionality. In addition, some of site's visitors are in non-English speakers, so there might be requirement for internationalisation. Finally the pet store currently has no online ticketing system. Little of the existing order processing system, except parts of the database schema, is likely to be reusable for the web interface.

To date, this research does not intend to present a full fledged tool capable of reversing source code automatically, but to establish a reference model consisting of description languages, abstraction rules and a reengineering process. In this case study, Web tier needs be re-implemented to achieve the similar functionality of Pet Store, but a cleaner and thinner architecture.

Before abstraction, the legacy case system is reverse engineered with tool FermaT Maintainer's Environment (FME) [2]. FME provides four basic functions: a parser to present program/model in Abstract Syntax Tree (AST) structure, a WSL/WML editor, program/model transformation facilities and a command line console. Specification of legacy system is made more concise and professional by applying domain knowledge. The final recovered architecture consists of three main business logics in this application are: to obtain reference data about categories, breeds and descriptions; to obtain information about stock availability and to implement the ordering process. FCG tool in GMISA is utilised to construct multi-layer models from database based on system templates/patterns. It specifies a collection of built-in system services and configuration mechanisms and eliminates the dependency of applications by providing a clean and thin Web tier.

5. Related Work

Many formalisms and models have been introduced to specify Web-based systems. Araneus [10], AutoWeb [6], and WebML [1] present some interesting approaches in this area, where high-level models are used to design and develop each aspect of a Web-based system, ranging from requirements analysis, Web objects development to database design and presentation. MIC is a software and system development approach that advocates the use of domain-specific models to represent relevant aspects of a system. MIC can be considered as a particular manifestation of MDA, which is tailored towards system construction via domain-specific modelling languages [15]. Architecture Driven Modernisation (ADM) Task Force aims at extending MDA practices and standards to existing systems [11]. ADM specifically addresses the modernisation of legacy systems in the context of the MDA. The work presented in [13] is to investigate the feasibility of adopting MDA techniques for

their legacy systems, in order to safeguard the future maintainability of these systems. The steps have been implemented in a prototype "harvesting" tool.

6. Conclusion and Future Work

This research is focused on establishing a general framework and methodology to facilitate the Web-based system evolution. It can be claimed that any non-trivial evolution projects of Web-based systems would have to adopt a similar, if not identical, design and implementation strategy used in this research.

The research presented in this paper is not the end of the story. Further research on the proposed approach will elaborate on the following issues:

- ✓ For the research, the Framework is enough to demonstrate the Dependency Push concept. However, for developing full-blown Web applications, they need to be further developed.
- ✓ Compiler is the core of automation of abstraction and transformation rules. The design and implementation of a complete compiler is beyond the current work but should be a priority for future research.
- ✓ There is not enough experiment done on relationship analysis and data mining due to limited time of this research. In future work, this should be another priority.
- ✓ Although it is enough to demonstrate its powerful potential to handle the diversity of file formats in Web-based systems, the XML transformation implemented in this research is not complete enough for practical applications. This could be an improvement point in future work.

References

[1] S. Ceri, P. Fraternali and A. Bongio, "Web Modeling Language (WebML): a Modeling Language for DesigningWeb Sites", *Computer Networks*, vol. 33(1-6), Jun. 2000, pp. 37-157.

[2] F. Chen, "Model Driven Software Modernisation", PhD Thesis, De Montfort University, England, 2007.

[3] F. Chen, H. Yang, B. Qiao and C.-C. Chu, "A Formal Model Driven Approach to Dependable Software Evolution", 30th IEEE International Computer Software and Application Conference (COMPSAC'06), Chicago, USA, Sept. 2006, pp. 205-214.

[4] J. Favre, "Foundations of Model (Driven) (Reverse) Engineering: Models", Int. workshop Dagsthul, 2004.

[5] J. Favre, "Towards a Basic Theory to Model Model Driven Engineering", 3rd Workshop on Software Model Engineering (WiSME'04), Lisboa, Portugal, Oct. 2004.

[6] P. Fraternali and P. Paolini, "Model-driven Development of Web Applications", *ACM Transactions on Information Systems: the AutoWeb System*, vol. 18(4), 2000, pp. 323-382.

[7] H. Guo, F. Chen, Y. Wang and Y. Sun, "A Reusable Software Architecture Model for Manufactory Management Information System", 26th IEEE International Computer Software and Application Conference (COMPSAC'02) Oxford, England, Sep. 2002, pp. 469-471.

[8] S. Kent, "Model Driven Engineering", 3rd International Conference on Integrated Formal Methods (IFM'02), LNCS 2335, 2002, pp. 286-298.

[9] A. Kleppe, J. Warmer and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison Wesley, 2003.

[10] P. Merialdo, P. Atzeni and G. Mecca, "Design and Development of Data-intensive Web Sites: The Araneus Approach", *ACM Transactions on Internet Technology*, vol. 3(1), 2003, pp. 49-92.

[11] OMG, "Architecture Driven Modernization Roadmap", Technical Report, Draft #1, ADM Task Force, OMG, http://adm.omg.org/ADMTF Roadmap.pdf, 2006.

[12] B. Qiao, "Evolution of WEB-based Systems in Model Driven Architecture", PhD Thesis, De Montfort University, England, 2005.

[13] T. Reus, H. Geers and A. v. Deursen, "Harvesting Software Systems for MDA-Based Reengineering", in *Model Driven Architecture – Foundations and Applications*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2006, pp. 213-225.

[14] E. Seidewitz, "What Models Mean", *IEEE Software*, Sep. 2003.

[15] J. Sztipanovits and G. Karsai, "Model-Integrated Computing", *IEEE Computer*, Apr. 1997, pp. 110-112.