Chapter VI

Software Evolution with XVCL

Weishan Zhang, Tongji University, P.R. China

Stan Jarzabek, National University of Singapore, Singapore

Hongyu Zhang, RMIT University, Australia

Neil Loughran, Lancaster University, UK

Awais Rashid, Lancaster University, UK

Abstract

This chapter introduces software evolution with XVCL (XML-based Variant Configuration Language), which is an XML-based metaprogramming technique. As the software evolves, a large number of variants may arise, especially when such kinds of evolutions are related to multiple platforms as shown in our case study. Handling variants and tracing the impact of variants across the development lifecycle is a challenge. This chapter shows how we apply XVCL to handle variants that arise during software evolution. It also shows how we can maintain different versions of software in a reuse-based way.
Introduction

Software systems evolve and there can be many kinds of changes (Mens et al., 2003), such as porting to a new platform or enhancing user requirements. During evolution, multiple versions of a system arise differing in variant requirements. For example, in the evolution to a new platform, we have to handle large numbers of platform-related variants that have a local and sometimes global impact on the system. At times, we have to refine software architecture to mitigate the architecture erosion problem (Hoek et al., 1999). To facilitate evolution, it is essential to ensure traceability from variants in high-level software models, to architecture and to code components, test cases, and so forth. Another important issue is to maximize reusability across system versions emerging during evolution in order to save costs.

Metaprogramming techniques can help in software evolution by automating some of the tedious and error-prone tasks. In this chapter, we investigate the evolution with an XML-based metaprogramming technique of XVCL (XML-based Variant Configuration Language) (Jarzabek & Zhang, 2001; Soe, Zhang, & Jarzabek, 2002; Jarzabek et al., 2003). We apply XVCL on top of programs designed using traditional design techniques for enhanced maintainability and reusability. In the remaining part of the chapter, we will first give a brief introduction to XVCL. Then, an experiment of reengineering an existing PC-based system into a product line is described. After the case study, we discuss literature review of software evolution techniques and variability mechanisms. Finally we give some concluding remarks and discuss the future work.

Overview of XVCL

A Brief Introduction to XVCL™

XVCL is a metaprogramming technique and tool that provides effective reuse mechanisms. Being a modern and versatile version of Bassett’s frames, a technology that has achieved substantial gains in industry (Bassett, 1997), the underlying principles of XVCL have been thoroughly tested in practice. The basic building block in XVCL is called a metacomponent, which is an XML file instrumented with XVCL commands for ease of change and evolution. XVCL uses composition with adaptation rules to generate custom artifacts (code, documents, models, etc.) from a compact base of generic, reusable metacomponents. Metacomponents are component building blocks, designed for ease of adaptation and reuse. XVCL can successfully manage a wide range of variants for all software assets, which can be represented as textual content. For the more detailed description of XVCL, please refer to its homepage at http://fxvcl.sourceforge.net.

While designing a frame architecture is not trivial, subsequent productivity gains and maintenance savings often repay the effort many times over. An independent analysis showed that frames can reduce large software project costs by over 84% and their times-to-market by 70%, when compared to industry norms (Bassett, 1997). By reusing skillfully
36 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the product's webpage:

www.igi-global.com/chapter/software-evolution-xvcl/29613?camid=4v1


www.igi-global.com/e-resources/library-recommendation/?id=1

Related Content

Support for Collaborative Component-Based Software Engineering
www.igi-global.com/chapter/support-collaborative-component-based-software/4931?camid=4v1a

Normalization and Translation of XQuery
www.igi-global.com/chapter/normalization-translation-xquery/41509?camid=4v1a

Rule-Based OWL Ontology Reasoning Systems: Implementations, Strengths, and Weaknesses
www.igi-global.com/chapter/rule-based-owl-ontology-reasoning/35857?camid=4v1a

On the Co-Evolution of SSADM and the UML
www.igi-global.com/chapter/evolution-ssadm-uml/29612?camid=4v1a