

Variability Management for Product Lines with XVCL

Stan Jarzabek

Department of Computer Science, School of Computing

National University of Singapore

stan@comp.nus.edu.sg

Abstract

Managing variability is the essence of software product line (PL) practice. With many variant features and complex dependencies among them, it also becomes a major challenge for effective reuse. Without an adequate technique for managing variability, we face problems such as explosion of similar component versions, or difficulty to select and then adapt component configurations during reuse-based product development. These problems often hinder development productivity and may diminish the economic benefit of reuse via PL approach. A variability management technique should also allow us to evolve products, PL members, as required by their customers, without disconnecting them from also evolving reusable assets forming PL architecture. XVCL method [7] directly addresses software reuse and evolution problems that are difficult to solve using conventional OO, architectural, component approaches, and modern platform mechanisms such as .NET, JEE or Ruby on Rails. It does that on top of and in full synergy with any programming language, design technique or platform.

1. Motivation

Component-based approaches and platforms provide effective reuse solutions at the middleware level. However, in application domain-specific areas such as business logic or user interface, despite many similarities, software is still developed very much from scratch. One of the reasons why this happens is that variability affecting middleware components is lesser and can be easier localized than variability affecting upper system layers. Variant PL features tend to have complex and system-wide impact on business logic and user interface system layers. To reuse, one must understand the total impact of variant features on affected components. Unfortunately, this knowledge can't be explicitly expressed within a component paradigm. It can be expressed externally to software, but then we face problems of keeping external documentation in sync with evolving software components.

Reuse takes a long-term perspective. Therefore, an effective reuse technique must also solve problems of software evolution. In software product line (PL)

situation, products released to various customers differ in variant features. Products must evolve according to customers needs. At the same time, reusable assets, a PL architecture, also evolve. It is daunting problem to handle evolution of products and reusable assets without breaking the connection between them. Most often, products are evolved manually and cannot easily benefit from upgrades of reusable assets. Keeping products back in sync with reusable assets requires much manual rework.

Due to component inter-dependencies and non-local, system-wide impact of variant PL features, customization for reuse must be conceived at the level of component configurations, rather than at the level of individual components. While individual components can be designed to be more generic to accommodate variant features, handling customizations of component configurations is not a strong point of programming languages and component-based software technologies. Rather than reusing components, we create similar components for various reuse contexts which leads to explosion of component versions. Much of the useful functionality might have been already implemented, but it is not readily reusable.

2. The XVCL method

XVCL [7] provides simple yet effective mechanisms to address the above problems. XVCL method works in synergy with contemporary programming techniques, enhancing their ability to define generic, adaptable and easy to evolve software representations.

XVCL is a construction method. We use conventional programming languages and analysis/design techniques of our choice to define system functionality in terms of user interfaces, business logic and databases, or to tune systems for performance and to satisfy specific deployment situations. Then, we wrap conventional software into XVCL representation. Not only does XVCL representation contain full knowledge of executable programs, but also design information helpful in effective reuse and evolution. For example, in XVCL we can represent component configurations that recur in PL members in variant forms, and show which variant features affect these component configurations and how.

The ability to formally represent code along with the design information is one a unique features of XVCL. So

is the ability to represent similar program structures of any kind or granularity in a generic form, independently of how they are similar and different.

Executable programs are automatically derived from XVCL representation by the XVCL Processor. As software is always analyzed, reused and maintained at the level of an XVCL representation, we avoid problems of managing component versions: A required component version is derived from the XVCL representation based on specification of variant features affecting a component.

XVCL mechanisms were designed with evolution in mind. We can evolve products already released to customers and reusable (XVCL-based) assets independently of each other, but without breaking the connection between them.

Defining and stabilizing a runtime architecture is the first step in setting up a PL. Runtime PL architecture is a component architecture characterized by interfaces, patterns of system organization (e.g., implied by underlying platform such as .NET™ or J2EE™, or patterns such as MVC), and any other important mechanisms that ensure that systems properly execute. PL members usually share most of the runtime PL architecture assumptions. Building a runtime PL architecture is also the first step in XVCL method. By applying XVCL on top of a runtime PL architecture, we achieve extra levels of genericity, adaptability and reuse productivity.

3. Experiences

XVCL has been applied in lab studies and industrial projects in application domains ranging from Web Portals, command and control systems, business systems, role-playing games for mobile devices and class libraries, developed with ASP, JEE, .NET, Java, or C++. Experiences from these projects are documented in publications and on xvcl.comp.nus.edu.sg.

As an example, in the ASP Web Portal (WP) Product Line project [6], our industry partner ST Electronics Pte. Ltd. applied state-of-the-art design methods to build a Team Collaboration Portal (TCP) product line. Still, a number of problem areas were observed that could be improved by applying XVCL. The benefits of an ASP/XVCL PL architecture were the following:

- Short time (less than 2 weeks) and small effort (2 persons) to transform the TCP into the first version of an ASP/XVCL PL architecture.
- High productivity in building new portals from the ASP/XVCL PL architecture. New portal modules could be built by writing as little as 10% of unique custom code, while the rest of code could be reused. This code reduction translated into an estimated eight-fold reduction of effort required to build new portals.
- Significant reduction of maintenance effort when enhancing individual portals. The overall managed code lines in the ASP/XVCL PL architecture

supporting nine portals were 22% less than the original single portal.

- Wide range of portals differing in a large number of inter-dependent features supported by the ASP/XVCL PL architecture.

In another industrial project, XVCL was applied to manage variability in a role-playing game Product Line for mobile devices [10]. This project demonstrated that reuse may go hand-in-hand with improving, rather than compromising, the performance.

References

- [1] Basit, H.A., Rajapakse, D.C., and Jarzabek, S. "Beyond Templates: a Study of Clones in the STL and Some General Implications," *Int. Conf. Software Engineering, ICSE'05*, St. Louis, USA, May 2005, pp. 451-459
- [2] Jarzabek, S. *Effective Software Maintenance and Evolution: Reuse-based Approach*, Taylor & Francis CRC Press, May, 2007
- [3] Jarzabek, S. and Li, S. "Eliminating Redundancies with a "Composition with Adaptation" Meta-programming Technique," *Proc. ESEC-FSE'03, European Soft. Eng. Conf. and ACM SIGSOFT Symposium on the Foundations of Soft. Eng.*, Sept. 2003, Helsinki, pp. 237-246; the paper received ACM SIGSOFT distinguished paper award
- [4] Jarzabek, S. and Li, S. "Unifying clones with a generative programming technique: a case study," *Journal of Software Maintenance and Evolution: Research and Practice*, John Wiley & Sons, Volume 18, Issue 4, July/August 2006, pp. 267-292
- [5] Jarzabek, S. and Pettersson, U. "Cost-Effective Engineering of Web Applications—Pragmatic Reuse: Building Web Application Product Lines," *Int. Conf. Software Engineering, ICSE'06*, Shanghai, May 2006, pp. 1053-1054 (description of the tutorial presented at ICSE)
- [6] Pettersson, U., and Jarzabek, S. "Industrial Experience with Building a Web Portal Product Line using a Lightweight, Reactive Approach," *ESEC-FSE'05, European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, ACM Press, September 2005, Lisbon, pp. 326-335
- [7] XVCL (XML-based Variant Configuration Language) method and tool for managing software changes during evolution and reuse, <http://xvcl.comp.nus.edu.sg>
- [8] Yang, J. and Jarzabek, S. "Applying a Generative Technique for Enhanced Reuse on J2EE Platform," accepted for 4th Int. Conf. on Generative Programming and Component Engineering, *GPCE'05*, Sep 29 - Oct 1, 2005, Tallinn, Estonia, pp. 237-255
- [9] Zhang, H. and Jarzabek, S. "A Mechanism for Handling Variants in Software Product Lines," *Science of Computer Programming*, Volume 53, Issue 3, Dec. 2004, pp. 255-436
- [10] Zhang, W. and Jarzabek, S. "Reuse without Compromising Performance: Experience from RPG Software Product Line for Mobile Devices," accepted for 9th Int. Software Product Line Conf., *SPLC'05*, September 2005, Rennes, France, pp. 57-69