



XML-based Variant Configuration Language

<http://xvcl.comp.nus.edu.sg>

A technology for enhanced reusability and maintainability based on Bassett's frames

Technology Summary

XVCL is a public domain technology for enhanced reusability and maintainability developed in the Software Engineering Lab at NUS.

- *Do you find yourself solving similar design problems and writing similar code over and over again?*
- *Do you escape to replicating code sections, files and designs, but then you end up maintaining overly complex, error-prone programs?*
- *When maintaining multiple versions of the same software system released to different customers – do you take full benefit of the high similarity among system releases? Or do you rather maintain each release as a separate product?*
- *Are you at lost when modifying programs because of complex and undocumented impact of changes?*

If your answer to some of the above is 'yes' – you may find XVCL useful. Please read on for a brief description of the XVCL solution, and then examine our web site for further details.

Common sense suggests that you should be able to express your design and code without repetitions, whenever you wish to do so. The basic idea behind software reuse is to exploit similarities within and across software systems to avoid repetitive development work. XVCL allows you to do that and more.

Structural similarities are repetition patterns in software of any type or granularity, from similar code fragments to recurring architecture-level component configuration patterns. The unique feature of XVCL is its ability to represent any repetition pattern within a system or across systems, in a generic, adaptable form. Such non-redundant software representation is simpler to understand and maintain than conventional software.

While it is always good to work with a simpler program representation than with a complex one, the concept of representing structural similarity patterns in generic form is particularly useful in reuse via Product Line (PL) approach. Conventional reuse is based on components and architectures. XVCL-powered reuse of structural similarities significantly extends the benefits of PL solutions based on conventional component reuse only.

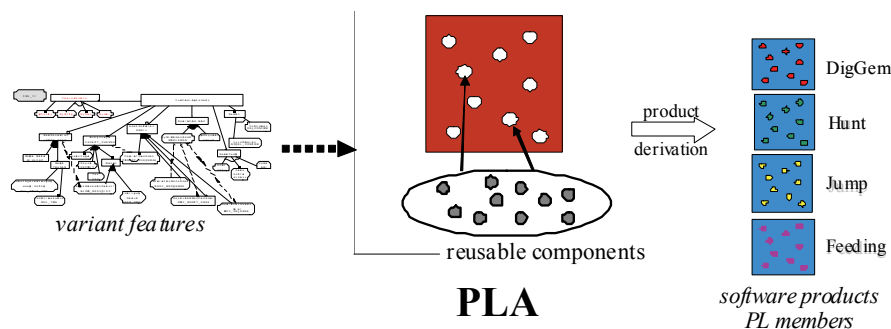


Fig. 1. Product Line Architecture (PLA) and product derivation

Fig. 1 depicts the main concepts for a PL formed by Role Playing Games (RPG) for mobile phones. All RPGs are similar, but they also differ in some functional requirements, and in characteristics of a specific mobile device on which they run. The goal of mobile device contents providers is to support possibly large number of games, on a possibly wide range of mobile devices. Games for new mobile device models should

be delivered fast. Much similarity among games makes reuse a promising approach to cut game development time and effort [13].

RPG Product Line Architecture (PLA) absorbs the impact of variant features on RPG components. PLA includes a base of reusable components organized into architectural structures that make component reuse easier. From PLA, we derive custom products, PL members.

Managing variability with suitable variations mechanisms is the essence of reuse: PL members share common features and differ in variant features. The role of a variation mechanism is to help us build a PLA, and automate reuse-based derivation of products from a PLA. A suitable variation mechanism can also extend reuse benefits to post-delivery product evolution. XVCL is a variation mechanism designed to effectively play the above roles.

XVCL Approach

XVCL is not yet another programming language – you still use conventional programming languages and platforms (e.g., JEE or .NET) to develop program logic, user interfaces, etc. You apply XVCL to tackle problems related to repetitions and frequent changes. XVCL works in synergy with any programming technology, and does not impose any restrictions on how systems are implemented and deployed.

The core concept behind XVCL is to represent each important recurring software structure in generic, adaptable form, along with full details of its implementation.

In Fig. 2, similar program structures S-i have been observed in members of the RPG PL, and then represented with S-gen in an RPG PLA built with XVCL. At the same time, we specified customizations required to derive concrete structures S-i from S-gen (circles below S-gen). XVCL Processor derives concrete structures from S-gen based on specified customizations.

Unification of similar program structures is the core of XVCL engineering process. It is done at all levels, from similar code fragments (such as class methods), to classes, components, component configurations, and subsystems. At the end, we build a generic representation of PL members as a PLA from which custom products are derived.

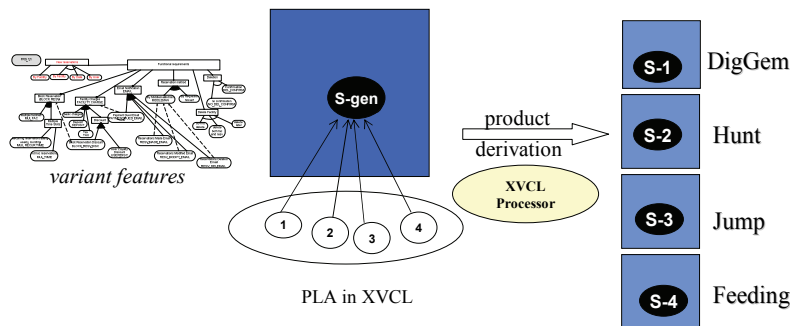


Fig. 2. A PLA for RPGs and product derivation with XVCL Processor

Salient features of XVCL

Applications of XVCL range from designing Product Line architectures, to day-to-day maintenance, and to managing multiple similar software products released to different customers over years of evolution.

The following problems often impede conventional component/architecture-based reuse. First, as a PL evolves, companies observe explosion of similar component versions in the PLA. This hinders selection and customization of components for reuse when deriving new products from a PLA. Functionality already implemented may be difficult to reuse in new products, defeating the very purpose of establishing a PL. Second problem is that product derivation is mostly done manually, with the help of complementary techniques such as wizards or configuration files. Finally, benefits of component/architecture-based reuse are mainly observed during new development, but are less evident in long-term evolution of successful products.

XVCL counters the above problems.

In the nutshell, XVCL offers these unique capabilities (listed from specific to general):

- (1) represents each important group of recurring software structures in a generic, adaptable XVCL form, along with full details of structure implementation,
- (2) shows a clear picture of differences among software structures in each group as deltas from their generic XVCL counterpart,
- (3) explicates important application domain or design concepts and shows how they are implemented,
- (4) captures traces of modifications applied to program components during maintenance,
- (5) offers a simplified, change- and reuse-oriented program representation that extends reuse rates beyond what is possible with conventional component/architecture approaches,
- (6) represents program code along with design information crucial for program understanding, maintenance and reuse, in fully integrated form, both human-readable and amenable to automatic manipulation by the XVCL Processor.

Other salient features of XVCL include:

- Synergy with any other programming language, design technique or component platform.
- Incremental, lightweight adoption: low-cost start, quick results, refinement of a solution for increased productivity.
- Synergy with both formal and agile development processes.
- Managing variability in a range of PL assets including code components, test cases, end-user documentation, UML models [8], and formal specifications [17].

An Example

In the ASP Web Portal (WP) Product Line project [9] ST Electronics (Info-Software Systems) Pte Ltd applied state-of-the-art design methods to maximize reusability of a Team Collaboration Portal (TCP) in other contexts. Still, a number of problem areas were observed that could be improved by applying XVCL. The benefits of a generic ASP/XVCL solution were the following:

- Short time (less than 2 weeks) and small effort (2 persons) to transform the TCP into the first version of a “Generic TCP” in ASP/XVCL.
- High productivity in building new portals from the Generic TCP. Based on the Generic TCP, ST Electronics could build new portal modules by writing as little as 10% of unique custom code, while the rest of the 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 Generic TCP 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 Generic TCP.

Current status of XVCL

The XVCL Web site xvcl.comp.nus.edu.sg contains XVCL specifications, the processor, case studies and other learning materials. We have implemented an XVCL Workbench with the following tools:

Basic Workbench – supports developers in organizing XVCL project information, creating, maintaining and reusing XVCL representation. Basic Workbench includes Smart Editor that hides the XML syntax of XVCL commands, and allows developers to work in terms of XVCL structures rather than the text.

Debugger – interprets XVCL representation in the interactive mode, allowing a developer to trace the processing sequences and states. The processing can be suspended at designated break points and a developer can examine/change the state of processing.

Static/dynamic analyzer of XVCL representation – answers developer’s queries about the properties of XVCL representation and processing.

Visualizer – shows graphical views of XVCL representation.

X-Profiler – computes XVCL metrics such as meta-component reuse statistics, the number of times variation points have been modified during customization, and other statistics that help in reusing and evolving XVCL representations.

X-Profiler – computes XVCL metrics.

Backward Propagation Tool (BPT) – allows a developer to propagate code fixes from the generated source code back to meta-structures, in a semi-automatic way.

References

- [1] XVCL is fully documented in Jarzabek, S. *Effective Software Maintenance and Evolution: Reused-based Approach*, Auerbach, CRC Press Taylor and Francis, May 2007
- [2] 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
- [3] Basit, A.H. and Jarzabek, S. “[Detecting Higher-level Similarity Patterns in Programs](#),” *ESEC-FSE’05, European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, ACM Press, September 2005, Lisbon, pp. 156-165
- [4] Basit, H. A., Jarzabek, S. “Data Mining Approach for Detecting Higher-level Clones in Software,” to appear in *IEEE Trans. on Soft. Eng.*
- [5] Bassett, P. *Framing software reuse - lessons from real world*, Yourdon Press, Prentice Hall, 1997
- [6] Clone Miner and Clone Analyzer: Technology Summary
- [7] Jarzabek, S. and Li, S. “[Eliminating Redundancies with a “Composition with Adaptation” Meta-programming Technique](#),” *Proc. ESEC-FSE’03, European Software Engineering Conf. and ACM SIGSOFT Symp. on the Foundations of Software Engineering*, ACM Press, September 2003, Helsinki, pp. 237-246; paper received ACM Distinguished Paper award; extended version: 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
- [8] Jarzabek, S. and Zhang, H. “[XML-based Method and Tool for Handling Variant Requirements in Domain Models](#)”, *Proc. 5th International Symposium on Requirements Engineering, RE’01*, August 2001, Toronto, Canada, pp. 166-173
- [9] Jarzabek, S. “Genericity - a “Missing in Action” Key to Software Simplification and Reuse,” *13th Asia-Pacific Soft. Eng. Conference, APSEC’06*, 6-8 December 2006, Bangalore, India, pp. 293-300
- [10] 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 Symp. on the Foundations of Software Engineering*, ACM Press, Sept. 2005, Lisbon, pp. 326-335
- [11] Rajapakse, D. and Jarzabek, S. “Towards generic representation of web applications: solutions and trade-offs,” to appear in *Software, Practice & Experience*
- [12] Zhang, H. and Jarzabek, S., “[An XVCL-based Approach to Software Product Line Development](#)”, *Proc. 15th International Conference on Software Engineering and Knowledge Engineering (SEKE’03)*, San Francisco, USA, 1 - 3 July, 2003.
- [13] Zhang, W. and Jarzabek, S. “[Reuse without Compromising Performance: Experience from RPG Software Product Line for Mobile Devices](#),” *9th Int. Software Product Line Conference, SPLC’05*, September 2005, Rennes, France, pp. 57-69
- [14] Zhang, H., Jarzabek, S. and Myat Swe, S. “XVCL Approach to Separating Concerns in Product Line Assets,” *Proc. 3rd International Conference on Generative and Component-Based Software Engineering*, LNCS, Springer Verlag, September 2001, Erfurt, Germany, pp. 36-47
- [15] Zhang, H. and Jarzabek, S. [A Mechanism for Handling Variants in Software Product Lines](#),” special issue on Software Variability Management of Elsevier’s journal *Science of Computer Programming*, Volume 53, Issue 3, Dec. 2004, pp. 255-436
- [16] Yang, J. and Jarzabek, S. “[Applying a Generative Technique for Enhanced Reuse on J2EE Platform](#),” *4th Int. Conf. on Generative Programming and Component Engineering, GPCE’05*, Sep 29 - Oct 1, 2005, pp. 237-255
- [17] Yuan, L., Dong, J.S. and Basit, H. :Generic Fault Tolerant Software Architecture Reasoning and Customization,” *IEEE Trans. on Reliability*, vol. 55(3), 2006, pp. 421-435

Contact: Stan Jarzabek

Department of Computer Science, School of Computing, National University of Singapore
Computing 1, #03-68 Law Link, Singapore 117590

e-mail: stan@comp.nus.edu.sg; <http://www.comp.nus.edu.sg/~stan>

fax: 65-6779-4580; tel: 65-6874-2863 (office) 65-96255863 (mobile)