Editor: Forrest Shull ■ Fraunhofer Center for Experimental Software Engineering, Maryland ■ fshull@fc-md.umd.edu

Managing Variability in Software Product Lines

Muhammad Ali Babar, Lianping Chen, and Forrest Shull

A software product line (SPL) is a set of software-intensive systems that share a common set of features for satisfying a particular market segment's needs. SPLs can reduce development costs, shorten time-to-market, and improve product quality by reusing core assets for project-specific customizations.[1,2] To enable reuse on a large scale, SPL engineering (SPLE) identifies and manages commonalities and variations across a set of system artifacts such as requirements, architectures, code components, and test cases. Many companies have adopted this development approach: Nokia, Philips, Bosch, Toshiba, Ericsson, Boeing, Hewlett-Packard, and Cummins are among the companies recognized in the Product Line Hall of Fame (http://splc.net/fame.html) for their success with SPLE.

Variability management (VM) is a fundamental SPLE activity that explicitly represents software artifact variations for managing dependencies among variants and supporting their instantiations throughout the SPL life cycle.[3] Managing variability involves extremely complex and challenging tasks, which must be supported by effective methods, techniques, and tools.[4,5] Researchers have studied these challenges and proposed solutions to them for nearly 20 years.

We recently undertook a study to systematically review this research. Our purpose was to synthesize and assess the evidence regarding the effectiveness of proposed solutions.

Challenges and Solutions

In 2001, researchers and industry representatives met to identify SPLE's main issues and problem areas.[4] To update these findings for our study, we organized group interviews with practitioners in 2008, asking them to identify the issues they experience on a daily basis. As a third data source, we reviewed the research literature published through January 2008. This review initially identified 261 papers that reported an approach to some aspect of VM in SPLE or an evaluation of an existing VM approach. The study assessed 97 papers that either claimed or provided some kind of evaluation of a VM approach, technique, or tool. (Details are available in a Web appendix to this article at www.computer.org/software/webextra.html.)

We grouped the issues from all three sources into 12 categories. Table 1 on p. 60 lists these issues, along with our subjective rating of how important each source ranked them. The table shows some changes in practitioner-identified issues from 2001 to 2008.

Although practitioners reported several challenges related to nontechnical issues, such as social, organizational, and human factors, we found no proposed solutions to these issues in our research. In our discussions of the approaches we found for the other 11 issues, we cite one review paper to represent each approach.

Commonality and Variability Identification

To identify SPL commonalities and variabilities, stakeholders must analyze and negotiate the meanings of domain concepts. This activity often proceeds informally, relying on personal domain experience. However, our study revealed four main approaches: feature-oriented domain analysis (FODA),[6] feature-oriented reuse method (FORM),[7] an approach we call *domain requirements commonality and variability analysis*,[8] and another we call *domain requirements modeling*.[9]

Table 1

Comparison of variability management issues

VM issues	Sources and ratings*		
	2001 study	2008 study	2009 literature review
Commonality and variability identification	+ + +	+	+ + +
Binding decisions	+ + +	+ +	+ + +
Variability modeling	+ + +	+ +	+ + +
Architectural design	+ + +	+ +	+ +
Product derivation	+	+	++
Variability evolution	+ + +	+ + +	+
Tool support	+ + +	+ + +	+ + +
Process support		+ + +	+
Scalability		+ + +	+ +
Quality assurance techniques		+ + +	+
Shared knowledge and rationales		+ + +	
Nontechnical issues		+ + +	+

*Degree of importance ranges from "+" for topics that a source merely acknowledged "+++" to for topics that a source deemed very important.

All these approaches claim to help domain engineers systematically identify the common and variable features in a product family. FODA and FORM represent early work in applying feature-modeling techniques.

Binding Decisions

Early decisions to bind variation points reduce the flexibility of product lines, and late binding can be expensive. When practitioners lack a tool for investigating trade-offs, they tend to make ad hoc and experience-based decisions. Our data analysis from the group interviews and literature review suggested that this approach is usually suboptimal.

Despite the importance of binding issues, only one study proposed a solution addressing them explicitly.[10] Specifically, it proposes an infrastructure—that is, a representation mechanism and tools—for specifying variability at design time and resolving it anytime.

Variability Modeling

We found more research addressing variability modeling than any of the other issues, not only in VM but also in SPLE overall.

Modeling issues concern the ability to satisfactorily capture, organize, and represent variability. FODA and several dozens of its derivates were the main approaches. A FODA approach that we call *cardinality-based feature modeling* combines feature modeling with staged configuration to achieve model specialization through a sequence of steps.[11] Other approaches included COVAMOF, a framework for variability modeling, orthogonal variability modeling, and decision modeling.[12]

Architectural Design

Software architects must select mechanisms for modeling variation points or choosing the best instantiation. The main proposed VM approaches for architectural design and evolution are integrated product- and component-based approaches such as SPL integrated technology (SPLIT)[13] and Kobra.[14]

Our study also revealed a decision-model approach[15] and an approach to integrate variability into IEEE Standard 1471 recommended practices for architectural description.[16]

Product Derivation

Product-derivation issues attracted the second-most research efforts, after variability modeling. The main issues relate to methodological and tool support for building a system based on existing assets.

To derive different products, an SPL system must implement appropriate variation mechanisms at different points in its processes. The main proposed approaches are COVAMOF; Koalish,[17] an architecture-centric product derivation approach; and a tool-based approach for product configuration at the file-system level.[18]

Variability Evolution

Variability evolves as a result of adding, deleting, or updating variation points and variants. However, we found little support for systematically and sufficiently supporting evolution in variability models and other related artifacts.

A few studies claim to address the issues explicitly. For example, Feature Description Language (FDL) claims to support evolution of variability models.[19] Another study proposes using two views to model feature variability and dependencies,[20] and another describes a method for detecting and removing obsolete variabilities.[21]

Tool Support

SPL systems have far too many variation points, associated variants, and interdependencies for engineers to manage manually. That's why SPL researchers have invested huge efforts developing several dozen VM tools—too many to list here. Some of the tools have entered the commercial arena. However, the practitioners in our study still reported the lack of integrated, standardized, and end-to-end tool support.

Process Support

VM tasks, inputs, and outputs require process support throughout an SPL life cycle. FAST (family-oriented abstraction, specification, and translation) and SPLIT are two prominent approaches to explicitly address VM process-related issues, but nothing yet provides full end-to-end process support. Our data indicated that practitioners recognize this challenge and are keen to find a complete solution.

Scalability

A VM approach should not only handle extremely large numbers of variabilities without compromising intuitiveness and comprehensiveness but also support small systems without incurring a huge overhead. Scalability must be a vital property of any VM approach, but the published literature hasn't emphasized it. We found only 19 of 261 papers concerned with it.

A few approaches claim to address one or more scalability dimensions. For example, two studies address separation of concerns,[12,21] another looks at organizing information into a hierarchical structure,[22] and orthogonal variability modeling offers a way to model variability separately from other artifacts.[23] However, no comprehensive solution yet exists.

Quality Assurance Techniques

Researchers have given little attention to QA techniques such as testing, inspections, and reviews of variability models and artifacts. FAST addresses some questions related to product line testing. Inspections and reviews have been completely ignored, although QA techniques for single systems don't help determine defects specific to SPL variants.

Shared Knowledge and Rationales

Failing to capture VM decisions and rationales makes subsequent product derivations and evolutionary tasks more difficult and risk prone. Our literature review found nothing specifically focused on sharing knowledge about VM decisions. However, practitioners reported severe problems arising from the lack of such knowledge. Our analysis of the interview data indicated a vital need for an approach that explicitly captures and sufficiently represents VM decision rationales.

Assessing the Evidence

More than just listing possible solutions to the technical issues, we also assessed the claimed or reported evaluations of VM approaches in 97 papers selected from 20 years of research. Figure 1 provides an overview of the temporal distribution of evidence types reported in these studies. An overall drop in the studies for which there was no evidence indicates that empirical evaluation improved slightly over the last decade. Still, in 2007, a majority (58 percent) of the studies provided no evidential support.

Looking at the kinds of evidence, we found that two-thirds of the studies were conducted in a laboratory environment with toy systems or a simple example from literature. We characterized these evidence types mostly as example applications, lab experiments-software, lab experiments-human subjects, or simulations, although the mapping isn't exact. VM is primarily intended to solve the issues of large numbers of variations in commercial-scale applications and their complex dependencies, so approaches relying on these kinds of evidence might be quite challenging to transition into real project use.

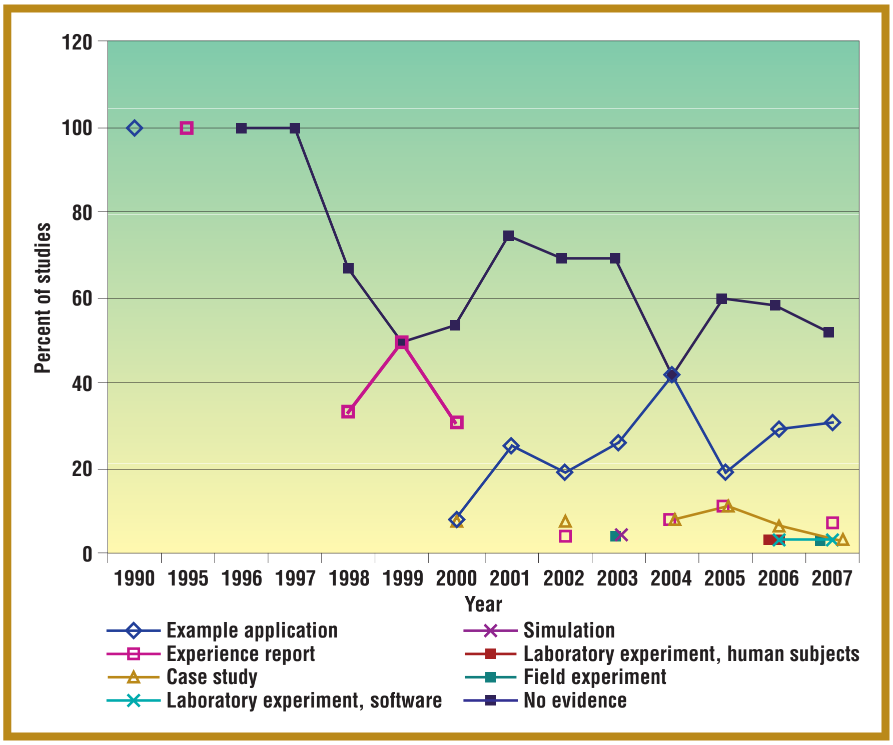

Figure 1. Distribution over time of evidence types in variability management studies. The percentage of studies for which there was no evidence dropped from 100 percent in 1996 to 58 percent in 2007, indicating some improvement over the decade.

We'd like to suggest that SPLE research focus not only on developing effective methods, techniques, and tools but also on rigorously and systematically evaluating them in industrial settings. In the meantime, because domains often have specific VM requirements, practitioners might want to evaluate a VM approach for domain suitability before selecting it for their project.

References

1. J. Bosch, *Design & Use of Software Architectures: Adopting and Evolving a Product-Line Approach*, Addison-Wesley, 2000.
2. P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison-Wesley, 2002.
3. K. Schmid and I. John, "A Customizable Approach to Full Lifecycle Variability Management," *Science of Computer Programming*, vol. 53, no. 3, 2004, pp. 259–284.
4. J. Bosch et al., "Variability Issues in Software Product Lines," *Proc. 4th Int'l Workshop on Software Product-Family Eng.*, Springer, 2002, pp. 13–21.
5. M. Sinnema and S. Deelstra, "Classifying Variability Modeling Techniques," *Information and Software Technology*, vol. 49, no. 7, 2007, pp. 717–739.
6. K.C. Kang et al., *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, tech. report CMU/SEI-90-TR-21, Carnegie Mellon Software Eng. Inst., 1990.
7. K.C. Kang et al., "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," *Annals of Software Eng.*, vol. 5, no. 1, 1998, pp. 143–168.
8. M. Moon, K. Yeom, and H.S. Chae, "An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line," *IEEE Trans. Software Eng.*, vol. 31, no. 7, 2005, pp. 551–569.
9. S. Park, M. Kim, and V. Sugumaran, "A Scenario, Goal and Feature-Oriented Domain Analysis Approach for Developing Software Product Lines," *Industrial Management + Data Systems*, vol. 104, no. 4, 2004, pp. 296–308.
10. A. van der Hoek, "Design-Time Product Line Architectures for Any-Time Variability," *Science of Computing Programming*, vol. 53, no. 3, 2004, pp. 285–304.

Continued on p. 94

organization hidden behind a complex acronym and rather purely commercial practices. Also, maximizing the number of obscure certifications on your resume may simply backfire. But even with a big name, if it offers an enormous palette of certifications driven mostly by commercial interests, the prestige might vanish. So, if you know how to reinstall Windows, you're not alone.

My final bits of advice: Be clear with yourself why you want a given certificate, for what purpose. And an objective of just personal growth is perfectly laudable. Prefer the certifications that set the bar high rather than the easy ones, or the cheap ones. Look at the longer-term rewards, not only the immediate low-hanging fruit.

Acknowledgments

This is *IEEE Software*, a publication of the IEEE Computer Society, which sponsors the Certified Software Development Professional and Certified Software Development Associate programs (www.computer.org/portal/web/certification).

References

1. *ACM/IEEE Software Engineering Code of Ethics and Professional Practice*, v.5.2, ACM and IEEE, 1999, section 8.01; www.acm.org/about/se-code.
2. S.E. Dreyfus and H.L. Dreyfus, *A Five-Stage Model of the Mental Activities Involved in Directed Skill Acquisition*, Operations Research Center, UC Berkeley, tech. report ORC 80-2, Feb. 1980; http://handle.dtic.mil/100.2/ADA084551.
3. R. Fox, "Shu Ha Ri," *The Iaido Newsletter*, vol. 7, no. 2, 1995; www.aikidofaq.com/essays/tin/shuhari.html.
4. A. Cockburn, *Agile Software Development*, Addison-Wesley, 2002, p. 17.
5. B.S. Bloom et al., eds., *Taxonomy of Educational Objectives: The Classification of Educational Goals: Handbook I, Cognitive Domain*, Longmans Green, 1956.

Philippe Kruchten is an IEEE CSDP (Certified Software Development Professional), part of the inaugural batch (# 99), as well as a Professional Engineer in Canada. He's currently professor of software engineering at the University of British Columbia, Vancouver, Canada, after retiring from a 33+ year career in industry.

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1314; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscription rates: IEEE Computer Society members get the lowest rate of US$52 per year, which includes printed issues plus online access to all issues published since 1988. Go to www.computer.org/subscribe to order and for more information on other subscription prices. Back issues: $20 for members, $163 for nonmembers (plus shipping and handling).

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.

Continued from p. 91

11. K. Czarnecki, S. Helsen, and U. Eisenecker, "Staged Configuration Using Feature Models," *Proc. 3rd Int'l Conf. Software Product Lines* (SPLC 04), LNCS 3154, Springer, 2004, pp. 266–283.
12. M. Sinnema et al., "COVAMOF: A Framework for Modeling Variability in Software Product Families," *Proc. 3rd Int'l Conf. Software Product Lines* (SPLC 04), LNCS 3154, Springer, 2004, pp. 197–213.
13. M. Coriat, J. Jourdan, and F. Boisbourdin, "The SPLIT Method: Building Product Lines for Software-Intensive Systems," *Proc. 1st Int'l Software Product Line Conf.* (SPLC 00), Kluwer Academic Publishers, 2000, pp. 147–166.
14. C. Atkinson, J. Bayer, and D. Muthig, "Component-Based Product Line Development: The KobrA Approach," *Proc. 1st Int'l Software Product Line Conf.* (SPLC 00), Kluwer Academic Publishers, 2000, pp. 289–309.
15. D. Muthig and C. Atkinson, "Model-Driven Product Line Architectures," *Proc. 2nd Int'l Conf. Software Product Lines* (SPLC 02), Springer, 2002, pp. 79–90.
16. S. Thiel and A. Hein, "Systematic Integration of Variability into Product Line Architecture Design," *Proc. 2nd Int'l Conf. Software Product Lines* (SPLC 02), LNCS 2379, Springer, 2002, pp. 67–102.
17. T. Asikainen, T. Soininen, and T. Männistö, "A Koala-Based Approach for Modeling and Deploying Configurable Software Product Families," *Proc. 5th Workshop Software Product-Family Eng.* (PFE 03), LNCS 3014, Springer, 2003, pp. 225–249.
18. C. Krueger, "Variation Management for Software Production Lines," *Proc. 2nd Int'l Conf. Software Product Lines* (SPLC 02), LNCS 2379, Springer, 2002, pp. 107–108.
19. A. van Deursen, M. de Jonge, and T. Kuipers, "Feature-Based Product Line Instantiation Using Source-Level Packages," *Proc. 2nd Int'l Conf. Software Product Lines* (SPLC 02), LNCS 2379, Springer, 2002, pp. 19–30.
20. H. Ye and H. Liu, "Approach to Modeling Feature Variability and Dependencies in Software Product Lines," *IEE Proc. Software*, vol. 152, no. 3, 2005, pp. 101–109.
21. F. Loesch and E. Ploedereder, "Optimization of Variability in Software Product Lines," *Proc. 11th Int'l Conf. Software Product Lines* (SPLC 07), IEEE CS Press, 2007, pp. 151–162.
22. M.-O. Reiser and M. Weber, "Multi-Level Feature Trees: A Pragmatic Approach to Managing Highly Complex Product Families," *Requirements Eng.*, vol. 12, no. 2, 2007, pp. 57–75.
23. F. Bachmann et al., "A Meta-Model for Representing Variability in Product Family Development," *Proc. 6th Software Product-Family Eng.* (PFE 04), LNCS 3014, Springer, 2004, pp. 66–80.

Muhammad Ali Babar is an associate professor at the IT University of Copenhagen. Contact him at malibaba@itu.dk.

Lianping Chen is a doctoral student with Lero, University of Limerick. Contact him at lianping.cehn@lero.ie.

Forrest Shull is a senior scientist at the Fraunhofer Center for Experimental Software Engineering, Maryland, and director of its Measurement and Knowledge Management Division. Contact him at fshull@fc-md.umd.edu.