On the Evolution of Software and Systems Product Line Standards

Sridhar Chimalakonda
Software Engineering Research Center
IIIT-Hyderabad, India
sridhar_ch@research.iiit.ac.in

Dan Hyung Lee
Korea Software Technology Association
Gyeonggi-do, Korea
danlee@kaist.ac.kr

DOI: 10.1145/2934240.2934248

ABSTRACT http://doi.acm.org/10.1145/2934240.2934248
Software product lines (SPL) have evolved from being a niche research topic to a well-established approach towards development of a family of software-intensive systems, promising better quality in shorter time and less cost. The hall of fame initiative of a 16 year running conference on SPL recognizes exemplary examples of successful application of SPL in practice. However, despite these advances, SPL are seen as quite effort intensive requiring huge initial investments and are not considered as mainstream software development. One of the primary reasons for this is the *lack of standardized tools and methods* to support development, maintenance and management of SPL. Specifically, every software product line is considered as a stand-alone project with custom tools and methods developed in most of the cases. This leads to a number of diversified, inconsistent and incompatible tools making it difficult to develop new tools or to choose from existing set of tools. It is here that the working group 4 (WG4) of the seventh subcommittee (SC7) titled "Software and Systems Engineering" of the Joint ISO/IEC Technical Committee (JTC1) of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) has proposed a set of interrelated standards in the area of software and systems product lines (SSPL). In this report, we briefly outline the state-of-the-art in software product line tools and motivate the need for standardized tools and methods. We then provide an overview of SSPL standards based on our experience in the last several years. We finally conclude by inviting the entire software engineering community and practitioners to get involved in the critical analysis and creation of SSPL standards and propel their use in both academia and industry.

Keywords
Software Product Lines, Standards, Tools, Methods, Patterns.

1. INTRODUCTION
The idea of building a family of systems that share a common set of properties was discussed by Dijkstra in 1970 **[**1**]** and by Parnas in 1976 **[**2**]**, which evolved into the notion of SPL **[**3**]**. Over the last two decades, there has been an extensive research on SPL as evidenced through a series of focused conferences like *Software Product Lines Conference* (SPLC), workshops like *Variability Modelling of Software-intensive Systems* (VaMoS), *Product Line Approaches in Software Engineering* (PLEASE), and each of these venues have dedicated tracks for practitioners of SPL in industry. In the recent years, work on SPL has also made into the foray of mainstream software engineering conferences like ICSE, FSE, ASE and so on. An analysis of the literature on SPL reveals that are two major terminologies to discuss the idea of developing a family of software-intensive systems. Software Engineering Institute (SEI) has steered the research and development on software product lines (sometimes called as software product family) and has published several technical reports and case studies **[**4**]**. On the other hand, several researchers, organizations from the other side of the world used the term *"software product line engineering"* for their work in the area of SPL **[**5**]**. Specifically, several European companies, universities and research institutes performed collaborative research on SPL, which supported the systematic building of a community of software product line engineering research and practice. Some of those projects include ARES (1995-1998), PRAISE (1998-2001), ESAPS (1999-2001), CAFÉ (2001-2003), and FAMILIES (2003-2005). Recently, Metzger and Pohl have done an extensive study of 600 articles published in the area of SPL and noted that there has been impressive quantitative and qualitative progress in the field with key challenges for industrial adoption **[**6**]**. Despite such a significant progress, SPL is still not a mainstream practice in many industries owing to lack of strong business case, commitment from management, adequate organizational structures and processes **[**6**]**. But two key challenges that are relevant in the context of this report are (i) lack of adequate support for SSPL tools and methods (ii) diversified, inconsistent and incompatible tools and methods. In Section 2, we briefly provide an overview of the diversified range of tools and methods in SPL. We discuss key foundations of SSPL standards in Section 3, provide an overview of SSPL standards in Section 4 followed by conclusions and some future directions.

2. TOOLS IN SOTWARE PRODUCT LINES
In the last decade or so, SPL community has witnessed a voluminous number of tools from academia as well as industry to support the entire software product line development life cycle [**7**] **[**8**]**. These tools are mostly developed to support specific processes and activities in SPL like requirements engineering, feature modeling, configuration management and so on. *AHEAD* is one of the early tool suites to support feature-oriented programming based on step-wise refinement **[**9**]**. *FeaturePlugin* is another tool that was proposed to support modeling feature diagrams **[**10**]**. *Requiline* is a tool that supports management of requirements engineering for SPL **[**11**]**. A survey done in 2007 on tools in product line requirements identifies twelve tools but only four out of them are alive at that point of time **[**8**]**. A plethora of tools and methods like FODA, FaMa Tool suite, FAMILIAR, FLAME, XFeature, SPLOT to support feature modeling have emerged in the literature **[**12**] [**13**]**. Researchers have also focused on tools to support formal verification of feature models **[**14**]**. There is an abundance of tools and methods to support variability management in SPL [**7**] **[**15**] [**16**]**. Most of the software product line approaches like COPA, FAST, FORM, KobrA and QADA include custom tools to support product line activities as part of their methods and approaches **[**17**]**. In recent times, *FeatureIDE* was proposed to support the entire life cycle of software product line development **[**18**]** as an open source alternative to commercial tools like *pure:variants* **[**19**]** and *Gears* **[**20**]**. Developing tools and methods for SPL is an effort intensive activity and the large number of connected processes and activities making it an uphill task.

While we do not detail the tools in this area, most of them are developed as research prototypes to support processes and methods of SPL in the areas of scoping, feature modeling, variability management and so on. This diversified range of tools and methods is one of the primary hindrance for adoption of SPL in the industry. It is here that standards in the area of SSPL could provide a standard approach to development of tools and methods. In the next section, we summarize the key foundations towards development of SSPL standards.

3. FOUNDATIONS OF SSPL STANDARDS
With hundreds of tools and methods available to support software product line development life cycle, how do researchers, practitioners and organizations choose the relevant ones? Do artifacts developed using one tool suite work with another? What if an organization identifies unique capabilities in two tools which are required for their SPL but the tools are not compatible? How to develop new tools and methods such

that they are consistent with existing ones? To address some of these challenges, a set of interrelated software and systems product line standards have been proposed by ISO/JTC1/SC7 **[**21**]**. The history of these standards initiatives on SPL was summarized in an earlier article **[**22**]**. In this report, we succinctly summarize the basic foundations of SSPL standards and outline the current standards and future directions.

- *Systems not just software:* The scope of the SSPL standards has been broadened to include *systems* along with SPL. This essentially means that the standards deal with development of key tasks, methods, techniques, mechanisms, tools and CASE environments for the planning, development, maintenance and management (both technical and organizational) of family of systems (e.g., family of software products, family of software services, family of system products, and family of system services) or SSPL (software and systems product line) at lower cost, reduced time to market, and with better quality.
- *Market Requirements:* To cope with today's market characteristics and business challenges such as mass customization, variability, complexity, time to market and cost competitiveness, automated and/or semi-automated supports of methods/tools/environments are crucial. The SSPL standards are typically based on extensive studies involving a large number of researchers and practitioners to figure out the needs of tools, find patterns among several tools to identify the expected capabilities of methods and tools towards eventually incorporating them into standards.
- *Patterns:* Technically, the SSPL standards follow a set of patterns for the entire software product line development life cycle based on a reference model. The standardized software product line reference model consists of product line scoping, requirements engineering, domain engineering, application engineering, verification and validation, asset management, organizational management, and technical management. Domain engineering and application engineering further consist of requirements engineering, design and realization. Each of these processes and sub-processes can be considered as *high-level patterns* (akin to software architecture patterns) that emerged from recurring and standardized practice from SPL communities, researchers and practitioners. These processes are further refined to explicitly model subprocesses, activities and further detailed as tasks. Each subprocess is typically specified using a seven tuple pattern *{Title, Purpose, Outcomes, Inputs, Tasks, Methods, Tools}*, where a set of inputs are explicitly specified to produce a set of explicit outcomes achieved using a set of tasks followed by explicitly specifying the capabilities of methods and tools required for performing the tasks effectively and efficiently. Figure 1 shows an excerpt of this pattern from a standard on software product line requirements engineering **[**23**]**. This pattern occurs repeatedly across the set of all interrelated standards in SSPL and changing the tuples facilitates variability in the creation of standards as well. Based on some research work, most of these standards also differentiate between domain patterns and software patterns. But discussing the pattern-oriented way applied to SSPL standards is a research endeavor beyond the scope of this report.
- *Core Activities:* Several processes and activities have been identified as core to SSPL that would recur in most of the standards. This facilitates the creation and maintenance of tools and methods in a consistent manner across the entire family of SSPL standards. A few examples include binding of variabilties, texture (structures, rules and semantics), configuration management, traceability throughout the SSPL life cycle, platforms and so on.

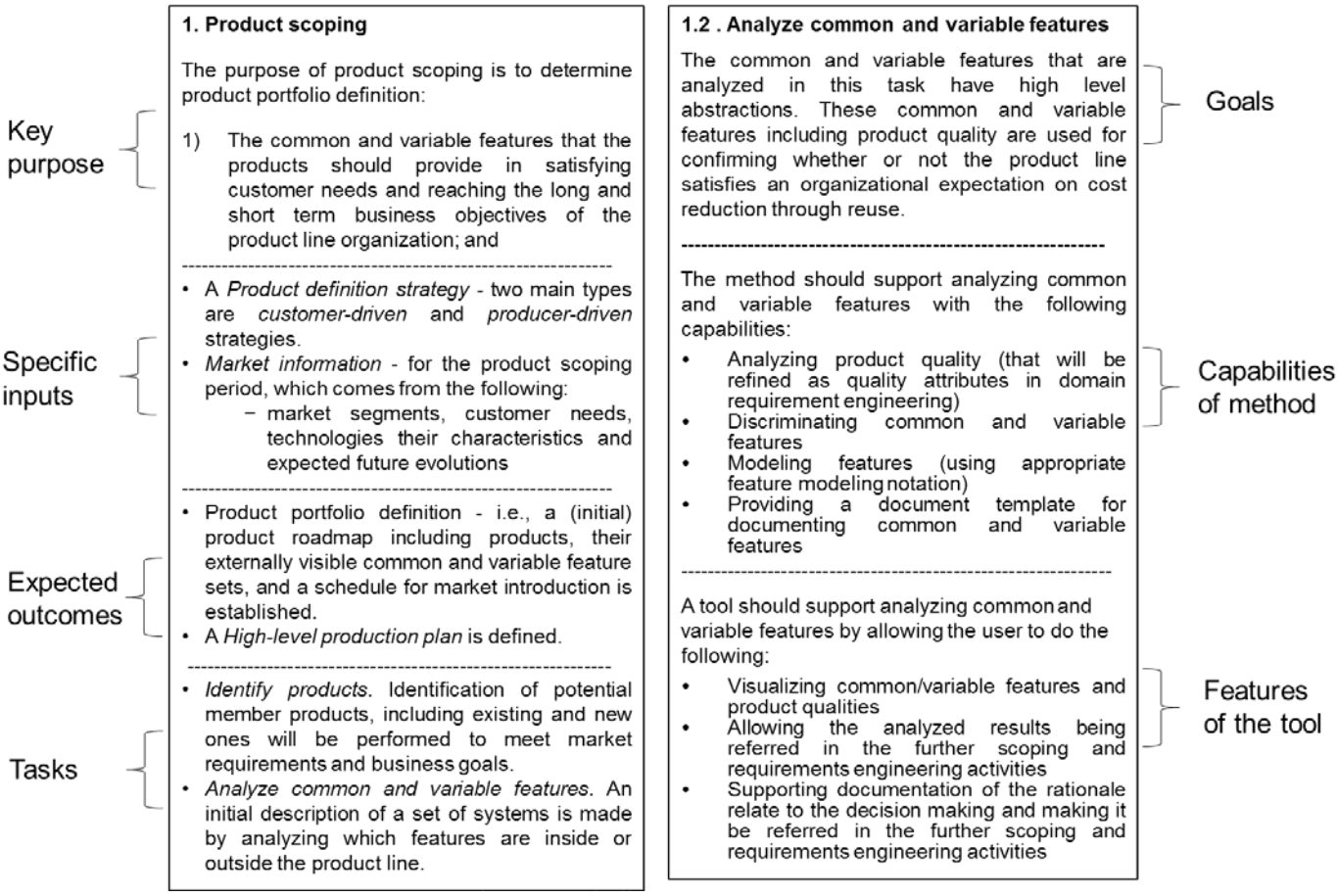

Figure 1. Excerpts of a structural pattern for software product line requirements engineering adapted from [**23**]

4. OVERVIEW OF SSPL STANDARDS

The reference model for SSPL defines software and systems product line engineering and management with two life cycles (domain engineering and application engineering) and two process groups (organizational management and technical management) **[**21**].** Figure 2 provides the reference model for all interrelated standards in SSPL. The reference model also provides interrelationships between the domain and application engineering life cycles and provides guidelines to adapt these lifecycles to a variety of organizational and technical environments for different quality and business goals. Organizational and technical management process groups focus on helping organizations to establish and improve capabilities for nurturing their product lines from conception to retirement and for establishing and managing relationships with customers, providers, and other key stakeholders. The SSPL standards can be used:

- by organizations intended to implement product lines – to understand, adopt, and enact the processes, tools, and methods for [*product line requirements engineering, architecture design, realization, verification and validation, technical management, organizational management*]
- by tool vendors to develop tools with required capabilities for supporting the entire life cycle of software product line engineering

SSPL is a complex endeavor requiring a diversified range of tools and methods for industrial adoption. To address these needs, an entire set of proposed standards (ISO/IEC 26550 to 26599) are envisioned with few standards published, few under development and few yet to be developed. On the other hand, SSPL is still an emerging area witnessing a significant progress in terms of new tools and methods. This requires extensive study of evolving market requirements and will lead to continuous evolution of the standards as well. We list some of the core standards as follows:

- The product line reference model (ISO/IEC 26550) provides an overview of the consecutive International Standards (i.e., ISO/IEC 26551 through ISO/IEC 26556) as well as the structure of the model.
- Processes and capabilities of methods and tools for product line scoping, domain requirements engineering, and application requirements engineering are described by *ISO/IEC 26551, Software and systems engineering – Tools and methods for product line requirements engineering.*

The rest of the standards follow the similar pattern to describe processes and capabilities of methods and tools for the entire software product line engineering life cycle.

- *ISO/IEC 26552, Software and systems engineering – Tools and methods for product line architecture design*.
- *ISO/IEC 26553, Software and systems engineering – Tools and methods for product line realization.*
- *ISO/IEC 26554, Software and systems engineering – Tools and methods for product line verification and validation.*
- *ISO/IEC 26555, Software and systems engineering – Tools and methods for product line technical management.*
- ISO/IEC *26556, Software and systems engineering – Tools and methods for product line organizational management.*

In addition, there are several draft standards currently in progress for variability mechanisms (ISO/IEC 26557), variability modeling (ISO/IEC 26558) and variability traceability (ISO/IEC 26559).

5. CONCLUSION & FUTURE DIRECTIONS

In order to address the inconsistency, incompatibility and reusability of a diversified range of tools in SSPL and better proliferation of SSPL in practice, we strongly motivated the need for standardization of SSPL tools and methods. We succinctly presented a few foundations for SSPL standards emphasizing the critical role of patterns followed by an overview of current SSPL standards. We invite researchers, practitioners and organizations involved in SPL

- To critically analyze existing standards in SSPL, provide valuable feedback and get involved in the standards initiatives
- To expose interfaces of tools in SSPL for better tool integration and standardization across organizations

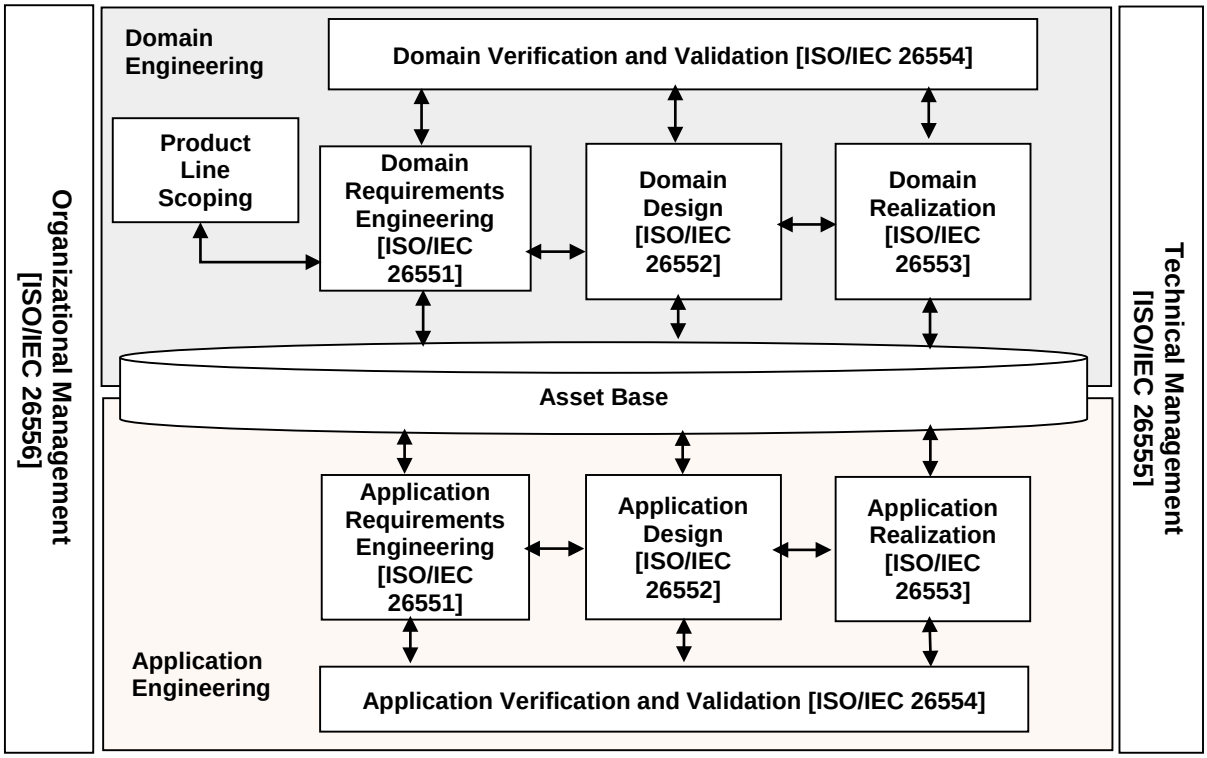

Figure 2. Reference model for software and systems product line adapted from [**21**]

- To create a repository of tools in all areas of SSPL, annotate and classify them from several dimensions
- To propose new and emerging areas/processes/activities in SSPL that have potential for new standards

Over time, as the tools and methods mature and industry moves towards adoption of particular segment of tools and methods, standardization might happen slowly even without explicit efforts. However, it comes with severe penalties like huge investments, risks, waste of effort, compromise on quality, inconsistent and incompatible tools. *To this end, this report is a call for our community to join hands in improving the standards in the area of software and systems product lines.*

6. REFERENCES

[1] Dijkstra, E. W. *Structured programming*. In J. Buxton and B. Randell, editors, Software Engineering Techniques, pages 84–87. NATO Scientific Affairs Division, 1970.

[2] Parnas, David L. On the design and development of program families. *Software Engineering, IEEE Transactions on*, 1 (1976), 1-9.

[3] Clements, Paul and Northrop, Linda. *Software product lines: practices and patterns*. Addison-Wesley Reading, 2002.

[4] Northrop, Linda M. SEI's software product line tenets. *IEEE software*, 19, 4 (2002), 32.

[5] Pohl, Klaus, B, and van Der Linden, Frank J. *Software product line engineering: foundations, principles and techniques*. Springer Science \& Business Media, 2005.

[6] Metzger, Andreas and Pohl, Klaus. Software product line engineering and variability management: achievements and challenges. In *Proceedings of the on Future of Software Engineering* (2014), 70-84.

[7] Chen, Lianping, Ali Babar, Muhammad, and Ali, Nour. Variability management in software product lines: a systematic review. In *Proceedings of the 13th International Software Product Line Conference* (2009), 81-90.

[8] Djebbi, Olfa, Salinesi, Camille, and Fanmuy, Gauthier. Industry survey of product lines management tools: Requirements, qualities and open issues. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International* (2007), 301-306.

[9] Batory, Don. Feature-oriented programming and the AHEAD tool suite. In *Proceedings of the 26th International Conference on Software Engineering* (2004), 702-703.

[10] Antkiewicz, Michal and Czarnecki, Krzysztof. FeaturePlugin: feature modeling plug-in for Eclipse. In *Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange* (2004), 67-72.

[11] von der Maßen, Thomas and Lichter, Horst. Requiline: A requirements engineering tool for software product lines. *Lecture notes in computer science* (2004), 168-180.f

[12] Benavides, David, Segura, Sergio, and Ruiz-Cort. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35, 6 (2010), 615-636.

[13] Apel, Sven, Batory, Don, K, and Saake, Gunter. *Feature-oriented software product lines: concepts and implementation*. Springer Science & Business Media, 2013.

[14] Schobbens, Pierre-Yves, Heymans, Patrick, and Trigaux, Jean-Christophe. Feature diagrams: A survey and a formal semantics. In *Requirements Engineering, 14th IEEE international conference* (2006), 139-148.

[15] Berger, Thorsten, Rublack, Ralf, Nair, Divya, Atlee, Joanne M, Becker, Martin, Czarnecki, Krzysztof, and W. A survey of variability modeling in industrial practice. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems* (2013), 7.

[16] Czarnecki, Krzysztof, Gr, Rabiser, Rick, Schmid, Klaus, and W. Cool features and tough decisions: a comparison of variability modeling approaches. In *Proceedings of the sixth international workshop on variability modeling of software-intensive systems* (2012), 173-182.

[17] Matinlassi, Mari. Comparison of software product line architecture design methods: COPA, FAST, FORM, KobrA and QADA. In *Proceedings of the 26th International Conference on Software Engineering* (2004), 127-136.

[18] Th, K, Benduhn, Fabian, Meinicke, Jens, Saake, Gunter, and Leich, Thomas. FeatureIDE: An extensible framework for feature-oriented software development. *Science of Computer Programming*, 79 (2014), 70-85.

[19] pure:systems. *pure:variants*. Website, available online at http://www.pure-systems.com/pure_variants.49.0.html, ; visited on May 2nd, 2016.

[20] Gears. *Gears: A Software Product Line Engineering Tool*. Website, available online at http://www.biglever.com/solution/product.html, Big Lever Software Inc.; visited on May 2nd, 2016.

[21] ISO (2015), ISO/IEC 26550:2015, Software and systems engineering -- Reference model for product line engineering and management.

[22] Käkölä, Timo. Standards initiatives for software product line engineering and management within the international organization for standardization. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on* (2010), 1-10.

[23] ISO (2016), ISO/IEC 26551:2016, Software and systems engineering -- Tools and methods for product line requirements engineering.