Towards a Generic Framework for Model-Driven Engineering of Software Process Lines

H. Agh, R. Ramsin
Department of Computer Engineering, Sharif University of Technology
P.O. Box 11365-11155
Tehran, Iran
agh@ce.sharif.edu, ramsin@sharif.edu

ABSTRACT

Situational Method Engineering (SME) approaches help construct bespoke software development processes according to the specifications of the project at hand, but they are time-consuming and costly. A Software Process Line (SPrL) tackles this problem by allowing software processes to be constructed for specific project situations through reusing core process assets. Model-Driven Development (MDD) has been used for automating SPrL Engineering (SPrLE); however, existing model-driven SPrLE methods are deficient as to their coverage of key MDD features. We propose a novel model-driven SPrLE approach that aims to address these shortcomings; it can be regarded as a framework that specifies the model chain and the core model-driven SPrLE activities that should be applied. The approach is yet to be refined and evolved through application to a real-world project; however, a preliminary criteria-based evaluation has shown that the shortcomings of existing SPrLE methods have indeed been addressed by the proposed approach.

CCS CONCEPTS

• **Software and its engineering → Software development process management**; Model-driven software engineering;

KEYWORDS

Situational Method Engineering; Software Process Line; Model-Driven Development; Software Process Improvement

ACM Reference format:

H. Agh, R. Ramsin. 2017. Towards a Generic Framework for Model-Driven Engineering of Software Process Lines. In *Proceedings of European Conference on the Engineering of Computer Based Systems, Larnaca, Cyprus, August-September 2017 (ECBS'17)*, 4 pages. DOI: 10.1145/3123779.3123810

ECBS '17, August 31-September 1, 2017, Larnaca, Cyprus

ACM ISBN 978-1-4503-4843-0/17/08□$15.00
https://doi.org/10.1145/3123779.3123810

1 INTRODUCTION

Software Product Line Engineering (SPLE) [1] has proven itself as an effective approach for fast and cost-effective development of high-quality software products and software-intensive systems [2]. Software Process Line Engineering (SPrLE) is somewhat similar to SPLE, but instead of software production, it is aimed at producing software development processes.

Software processes have been recognized as essential means for developing quality software systems [3]. Situational Method Engineering (SME) [4] is the discipline concerned with constructing bespoke software processes, tailored to fit the project situation at hand. A Software Process Line (SPrL) is a specialized Software Product Line (SPL) in the context of SME [5]. According to [6], a SPrL is "a set of software development processes that share a common, managed set of features satisfying the specific needs of an organization and that are developed from a common set of core assets in a prescribed way." Tailoring a SPrL for constructing project-specific processes is slow and error-prone if done manually. Therefore, Model-Driven Development (MDD) has been applied to SPrLE; thus, tailoring the SPrL to fit a specific project context can be (semi)automatically performed if: 1) a method base of reusable core assets is created, and 2) implicit tailoring knowledge is explicitly specified in the form of tailoring transformations [7]. Existing methods for model-driven SPrLE are afflicted with major deficiencies, namely: lack of a precise method for defining core processes, inadequate attention to the quality of software process practices selected in process tailoring, lack of multi-level modeling, and nonexistence of a method base for mapping project context attributes to development practices.

We propose a novel MDD approach/framework for SPrLE with the specific aim of addressing the above issues. Similar to SPLE, our approach is performed in two phases: Domain Engineering (DE) and Application Engineering (AE). During DE, common and variable elements of the targeted software processes are identified to create a core process which forms the architecture of the SPrL; Software Process Improvement (SPI) methods are used for improving the quality of the core process. During AE, members of the SPrL (specific processes) are built based on the core process and through applying tailoring transformations.

The rest of this paper is structured as follows: Section 2 provides a survey and evaluation of existing SPrLE approaches; Section 3 introduces our proposed approach; Section 4 examines the strengths of the proposed approach in comparison to existing approaches; and Section 5 presents the concluding remarks.

2 Related Research

In general, there are two categories of SPrLE approaches:

- **Non-model-driven approaches**: In [8], an approach is proposed for developing the architecture of a SPrL and then deriving a specific process from the architecture. In [9], an approach for creating a business SPrL is proposed. The Map Indicator-based Guidance (MIG) approach [10] extends the MAP formalism [11] to facilitate process customization. In [12], an approach is proposed for defining reusable processes in Software Process Consulting Organizations (SPCOs).
- **Model-driven approaches**: CASPER [5] is a meta-process and a set of process practices for creating adaptable process models. The method proposed in [13] supports variability management in software processes, automatic derivation of software processes, and automatic transformation of the derived processes into workflow specifications. In the approach proposed in [7], variabilities of process models are represented as feature models, the software process is modeled in eSPEM, and a MDD strategy is used for supporting automatic execution of transformation rules. In [14], a mega-model consisting of models and transformations is proposed for modeling and evolution of process lines in small software organizations.

We have evaluated the existing model-driven SPrLE approaches based on specially defined criteria. Since a comprehensive set of criteria for evaluating SPrLE approaches is not available, we have defined the criteria based on the features deemed desirable in SME and MDD frameworks, as specified in [15]. The evaluation results, shown in Table 1, indicate that these approaches are deficient in several aspects:

- Although the core process is a key part of a SPrL, many of the existing approaches lack a precise method for defining it.
- None of the approaches provides the features that are considered essential in MDD, such as multi-level modeling.
- None of the approaches provides a method base for mapping project context attributes to development practices; they thus fail to support (semi)automatic generation of custom processes.
- Although software process quality is of utmost importance [16], existing approaches fail to give proper attention to the quality of the practices selected during process tailoring.

3 Proposed Approach for Model-Driven SPrLE

Our proposed approach for model-driven SPrLE is shown in Figure 1. Each of its two phases, DE and AE (Sections 3.1 and 3.2), includes three sub-phases: Analysis, Design, and Implementation. Returning from each sub-phase to the previous one(s) is possible, as shown in Figure 1 by feedback loops. Returns from AE to DE are done in order to update the models.

Table 1: Results of criteria-based evaluation of existing model-driven SPrLE approaches

Criterion \ Model-Driven SPrLE Approach	[5]	[13]	[7]	[14]
Definition of core process(1)	Low	Low	Low	Low
Definition of modeling levels(2)	Low	Low	Low	Low
Attention to quality of process practices(3)	Low	Low	Low	Low
Potential for process automation(4)	Medium	Medium	Medium	Medium
Provision of knowledge repository(5)	Low	Low	Low	Low
Complexity management(6)	Low	High	Medium	Low
Transformation type(7)	Horizontal	Horizontal & Vertical	Horizontal	Horizontal
Maintainability(8)	Medium	High	Medium	Medium

Explanation of possible values for the criteria:

(1) - Low: The need for definition of the core process is mentioned, but a precise method for defining it is not provided.
- High: A precise method is provided for defining the core process.

(2) - Low: Some models are created throughout the process, but different levels of abstraction (prescribed by MDD) are not supported.
- High: Modeling levels are properly defined, and model transformation rules are specified.

(3) - Low: The core process is created by using the method fragments defined in organizational processes, and mechanisms such as SPI methods are not used for enhancing the quality of the core process.
- High: Mechanisms such as SPI methods are used for enhancing the quality of the core process.

(4) - Low: Most of the process is performed manually.
- Medium: A part of the process (pertaining to the application of transformation rules) is performed automatically, but complete identification of method fragments needs manual intervention by the method engineer.
- High: Most of the process is performed automatically by providing method bases of method fragments and automatic execution of transformation rules.

(5) - Low: No method base is defined for storing the method fragments and the relationships among the models.
- High: A method base is defined for storing the method fragments and the model relationships, and it is used throughout the process.

(6) - Low: Variation points and variants are included into the core process. Therefore, extending the core process will increase its complexity.
- Medium: The variability model and the core process model are defined separately, but the traceability links between them are not specified.
- High: The variability model and the core process model are defined separately, and the traceability links between them are specified.

(7) - Vertical: It is used when the source and destination models are at different levels of abstraction.
- Horizontal: It is used to convert models that are at the same abstraction level

(8) - Low: Any change in any part of the models leads to a ripple effect throughout the models.
- Medium to High: Mechanisms such as intermediate models are provided for enhancing the maintainability of the models.
- High: In addition to intermediate models, automatic propagation of maintenance changes to the models is supported.

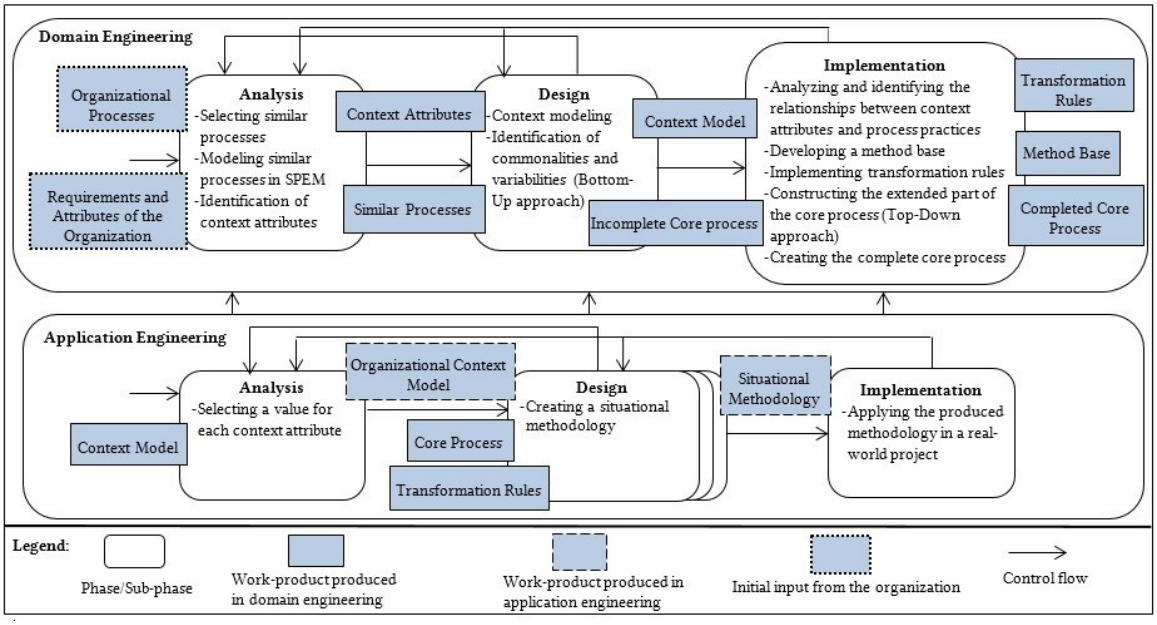

Figure 1: Proposed approach for SPrLE

3.1 Domain Engineering (DE)

The sub-phases of DE are explained in the following sections.

3.1.1 Analysis

The activities performed in this sub-phase are as follows:

- Selecting similar processes: Organizational processes are examined, and similar processes are identified.
- Modeling similar processes in SPEM: The identified processes are all remodeled in SPEM; this facilitates the identification of their commonalities and variabilities.
- Identification of context attributes: Attributes important to the organization are elicited from the tacit process knowledge of employees and published empirical knowledge.

3.1.2 Design

The activities performed in this sub-phase are as follows:

- Context modeling: Identified context attributes are modeled based on the Software Process Context Metamodel (SPCM) [7].
- Identification of commonalities and variabilities: A bottom-up approach is usually used for producing the core process, in which knowledge on existing process definitions and applications (in a well-known problem domain) is used for extracting the commonalities and variabilities. However, this approach can result in an inadequate core process, as existing processes may not be adequate. In the top-down approach, which is based on analyzing the domain, it is difficult to adequately elicit the commonalities and variabilities from scratch [8]. Therefore, we use both approaches: the bottom-up approach is applied in the Design sub-phase of DE to produce an initial core process; the top-down approach is then used in Implementation to improve the quality of the core process.

3.1.3 Implementation

The activities performed in this sub-phase are as follows:

- Analyzing and identifying the relationships between context attributes and process practices: The practices that are useful for each specific situation are identified. SPI practices can be used for improving the quality of the core process.
- Developing a method base: A method base is built for storing the core assets as well as the relationships between project context attributes and the core assets.
- Implementing transformation rules: Transformation rules are implemented using a language such as ATL. They are used for automatic derivation of a specific methodology from the SPrL.
- Constructing the extended part of the core process: The top-down approach is used for improving the quality of the core process based on the project situation. In [17], a reference framework is defined for situational factors that affect software processes; we use this framework to specify the situation. The mappings between context attributes (situational factors) and suitable development practices should also be identified so as to automatically determine the practices that fit the situation.
- Creating the complete core process: Transformations are applied to automatically merge the models created by the top-down and bottom-up approaches.

3.2 Application Engineering (AE)

The sub-phases of AE are explained in the following sections.

3.2.1 Analysis

The context model of DE is taken as input, and the values of context attributes are set by the method engineer based on the project situation, thus yielding an organizational context model.

3.2.2 Design

This is where MDD is applied. The organizational context model, core process model, and transformation rules are used for generating a situational methodology: the situational methodology is gradually generated at multiple modeling levels, from abstract to concrete, through the application of transformation rules. Two alternatives have so far been identified as the bases for defining the modeling levels: the granularity level of method fragments and the abstraction level of context attributes. In the former solution, variation points with higher granularities, such as the ones associated with phases or activities, are resolved first (at higher modeling levels); whereas variation points with lower granularities, such as those associated with tasks, roles, and work products, are resolved at lower levels. In the alternative solution, situational factors are classified based on their abstraction levels; variation points dependent on the values of higher-abstraction factors, such as organizational factors, are resolved first, whereas variation points dependent on lower-abstraction factors, such as project factors, are resolved at lower modeling levels.

3.2.3 Implementation

The methodology produced in the previous phase is enacted in the real world, and the results of its application may call for further iterations of the DE and AE phases.

4 Discussion

The problems observed in existing approaches, listed in Section 2, are addressed in our proposed approach; namely:

- In DE, a precise method is presented for creating the core process using bottom-up and top-down approaches.
- SPI methods are used in order to improve the quality of the core process and the derivable processes.
- Multiple modeling levels are defined in AE.
- A method base is devised for storing the core assets, and also for mapping project context attributes to the relevant practices.
- The target process, and parts of the core process, are generated automatically, thus providing a high degree of automation.
- The variability- and core-process models are produced separately, and a traceability matrix is used for maintaining their dependencies. Also, the target process is generated gradually during AE. Disruptive complexity is thus avoided.
- Horizontal and vertical transformations are both supported, the former for creating models such as the core process model, and the latter for generating the lower-level models in AE.
- Intermediate models are created, and transformation rules are applied for automatic propagation of changes to the models. Maintainability is thus promoted through MDD features.

5 Conclusion and Future Work

Even though the general specifications of the proposed approach are complete, the following elements have not been finalized yet: the contents of the method base, the modeling levels used in AE for model-driven development of processes, and the transformation rules. Future effort will hence focus on completing and refining the proposed approach through empirical evaluation.

REFERENCES

[1] Klaus Pohl, Günter Böckle, and Frank J. van Der Linden. 2005. Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag.

[2] Linda M. Northrop. 2002. Software Product Line Tenets. *IEEE Software* 19, 4 (2002), 32–40. DOI: http://dx.doi.org/10.1109/MS.2002.1020285

[3] Silvia T. Acuna, Angelica D. Antonio, Xavier Ferre, Marta Lopez, and Luis Mate. 2000. The Software Process: Modelling, Evaluation and Improvement. In *Handbook of Software Engineering and Knowledge Engineering*, Shi Kuo Chang (Ed.). World Scientific, Vol. 1, 193–238. DOI: http://dx.doi.org/10.1142/9789812389718_0011

[4] Brian Henderson-Sellers, Jolita Ralyté, Pär J. Agerfalk, and Matti Rossi. 2014. Situational Method Engineering. Springer-Verlag. DOI: http://dx.doi.org/ 10.1007/978-3-642-41467-1.

[5] Julio Ariel Hurtado Alegría and María Cecilia Bastarrica. 2012. Building Software Process Lines with CASPER. In *Proceedings of the 12th International Conference on Software and System Process* (ICSSP'12). IEEE, 170–179. DOI: http://dx.doi.org/10.1109/ICSSP.2012.6225962

[6] Ove Armbrust, Masafumi Katahira, Yuko Miyamoto, Jürgen Münch, Haruka Nakao, and Alexis Ocampo. 2009. Scoping Software Process Lines. *Software Process: Improvement and Practice* 14, 3 (2009), 181–197. DOI: http://dx.doi.org/10.1002/spip.412

[7] Julio Ariel Hurtado Alegría, María Cecilia Bastarrica, Sergio F. Ochoa, and Jocelyn Simmonds. 2013. MDE Software Process Lines in Small Companies. *Journal of Systems and Software* 86, 5 (2013), 1153–1171. DOI: http://doi.org/10.1016/j.jss.2012.09.033

[8] Hironori Washizaki. 2006. Building Software Process Line Architectures from Bottom Up. In *Lecture Notes in Computer Science*, Vol. 4034. Springer, 415–421. DOI: http://dx.doi.org/10.1007/11767718_37

[9] Vanessa Tavares Nunes, Claudia Maria Lima Werner, and Flávia Maria Santoro. 2010. Context-Based Process Line. In *Proceedings of the* 12th *International Conference on Enterprise Information System* (ICEIS'10). 277–282.

[10] Rébecca Deneckère and Elena Kornyshova. 2010. Process Line Configuration: An Indicator-Based Guidance of the Intentional Model MAP. In *Lecture Notes in Business Information Processing*, Vol. 50. Springer, 327–339. DOI: http://dx.doi.org/10.1007/978-3-642-13051-9_27

[11] Colette Rolland, Naveen Prakash, and Adolphe Benjamen. 1999. A Multi-Model View of Process Modeling. *Requirements Engineering* 4, 4 (1999), 169–187. DOI: http://dx.doi.org/10.1007/s007660050018

[12] Ahilton Barreto, Elaine Duarte, Ana Regina Rocha, and Leonardo Murta. 2010. Supporting the Definition of Software Processes at Consulting Organizations via Software Process Lines. In *Proceedings of the 7th International Conference on the Quality of Information and Communications Technology* (QUATIC'10). IEEE, 15–24. DOI: http://dx.doi.org/10.1109/QUATIC.2010.19

[13] Fellipe Araújo Aleixo, Marília Aranha Freire, Wanderson Câmara dos Santos, and Uirá Kulesza. 2010. Automating the Variability Management, Customization and Deployment of Software Processes: A Model-Driven Approach. In *Lecture Notes in Business Information Processing*, Vol. 73. Springer, 372–387. DOI: http://dx.doi.org/10.1007/978-3-642-19802-1_26

[14] Jocelyn Simmonds, Daniel Perovich, María Cecilia Bastarrica, and Luis Silvestre. 2015. A Megamodel for Software Process Line Modeling and Evolution. In *Proceedings of the 18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems* (MODELS'15). IEEE, 406–415. DOI: http://dx.doi.org/ 10.1109/MODELS.2015.7338272

[15] Halimeh Agh and Raman Ramsin. 2016. A Pattern-Based Model-Driven Approach for Situational Method Engineering. *Information and Software Technology* 78 (2016), 95–120. DOI: http://dx.doi.org/10.1016/j.infsof.2016.05.010

[16] Noor Azura Zakaria, Suhaimi Ibrahim, and Mohd Naz'ri Mahrin. 2015. The State of the Art and Issues in Software Process Tailoring. In *Proceedings of the 4th International Conference on Software Engineering and Computer Systems* (ICSECS'15). IEEE, 130–135. DOI: http://dx.doi.org/10.1109/ICSECS.2015.7333097

[17] Paul Clarke and Rory V. O'Connor. 2012. The Situational Factors That Affect the Software Development Process: Towards a Comprehensive Reference Framework. *Information and Software Technology* 54, 5 (2012), 433–447. DOI: http://dx.doi.org/10.1016/j.infsof.2011.12.003