Reactive and Iterative Evolution of Model-based Product Lines

Wolfgang Heider
Christian Doppler Laboratory for Automated Software Engineering
Johannes Kepler University
Linz, Austria
heider@ase.jku.at

***Abstract*—Industrial challenges in product line evolution suggest a flexible and iterative approach that integrates domain and application engineering activities. We present our ongoing and planned research addressing the evolution of model-based product lines. In particular, we propose an iterative process and initial tool support for product line evolution. Our work is based on an analysis of industrial challenges in product line evolution. The paper outlines the research method and summarizes the planned contributions.**

I. INTRODUCTION

Product lines (PL) are typically highly complex systems that are maintained and evolved over many years. Model-based approaches are frequently used to define the reusable assets and the variability of a PL. Due to continuous evolution the reusable assets and variability models need to be adapted to reflect changes to the domain requirements [1]. Pohl [2] states that "managing the evolution of software product line artifacts over time and ensuring the consistent integration of the changes in all affected product line applications are [...] key research challenges." An underlying reason for these challenges seems to lie in the inflexibility of existing product line engineering (PLE) methods. In particular, existing approaches address domain engineering (the definition of the PL and its variability) and application engineering (the actual use of the PL to create products) [2]. However, engineers often need to evolve a PL in a more iterative way by intertwining domain and application engineering activities [3]. This includes rapid feedback from application engineering projects [4] which currently takes too long in many environments. As in other areas of software engineering where sequential processes are increasingly replaced with iterative or agile approaches we believe that such changes can also be beneficial for PLE [5]. More specifically, in the area of model-based product lines there is still a lack of methods and effective tools that *treat evolution as the normal case and not as the exception*. The idea of PLE is to exploit anticipated variability, which historically has lead product line methods to overlook support for unanticipated evolution.

Authors in PLE have distinguished between proactive and reactive evolution [6]: Proactive evolution deals with preparing the PL for future needs by considering market and technology trends and forecasting general business needs to derive new domain requirements. Reactive evolution means evolving a PL by analyzing specific customer wishes regarding their potential as new domain requirements. Our research focuses on the reactive evolution scenario which is highly relevant in industry. For instance, in our collaboration with Siemens VAI [7][8][9] we have learned that new customer requests from multiple concurrent application engineering projects need to be systematically analyzed to plan the evolution. However, despite its importance reactive evolution is hardly supported by existing methods and tools in PLE.

The left part of Fig. 1 shows the traditional PLE approach with the clear separation into domain and application engineering. The right part gives a high-level view of our envisioned iterative approach for evolving model-based product lines. It covers activities for eliciting new customer application requirements during product derivation, analyzing and defining new domain requirements, and continuously evolving the PL. Our model-based approach relies on understanding changes to the key artifacts and dependencies between them. In particular, our research questions deal with: (1) understanding the relationships from new application requirements to affected PL model elements and assets that need to be customized; (2) analyzing the dependencies among application requirements in multiple projects to identify similarities or to negotiate conflicts; (3) investigation of the relationships from application requirements to emerging domain requirements to understand which PL features have been requested for which specific products; and (4) managing trace links from new or changed domain requirements to affected model elements for deploying the updated PL assets to the proper products.

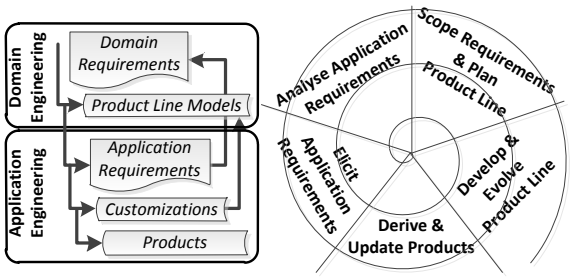

Figure 1. Sequential (left) vs Iterative (right) evolution in model-based product line engineering.

Our earlier publications focus on low-level mechanisms and tool support for change tracking and evolution in model-based environments [4][9]. This paper provides a top-down and process-oriented perspective of our approach and a discussion of the research method.

II. State of the Art

Researchers in the areas of product line evolution, model evolution, domain requirements engineering, and software processes have proposed approaches that provide important insights for our planned research.

A. Product Line Evolution

Several authors have stressed the importance of approaches for PL evolution and describe scenarios for PL adoption. Bosch [10] discusses different approaches of adopting a PL and defines maturity levels for PL artifacts. Knauber [11] discusses differences between proactive and reactive evolution of a PL. His paper offers guidelines for dealing with changing requirements and outlines the challenges for the management of reactive evolution. Dhungana *et al.* [8] present an approach of organizing product lines as a set of interrelated model fragments. They support to semi-automatically merge fragments into complete PL models for co-evolution of models and their respective meta-models. While these approaches provide partial solutions they however do not support the full cycle of reactive and iterative evolution of model-based product lines as required e.g., in the domain of our industry partner.

B. Model Evolution

Deng *et al.* [12] describe a model-driven PL approach that focuses on the problem of domain evolution with regard to PL architectures. They discuss several challenges for the evolution of model-driven software PL architectures and present an approach based on automated domain model transformations. Sprinkle [13] discusses the model migration problem for evolving meta-models, i.e., that models become invalid when meta-models evolve. Their solution is to automatically migrate existing models such that they conform to the new meta-model while preserving the available information as much as possible. Salinesi *et al.* [14] introduce an approach for analyzing and modeling the difference of two situations – before and after a change. Their gap modeling approach helps to better express evolution requirements with meta-modeling and operators. While these contributions provide important technical solutions for implementing support in model-based development they currently do not support the complete evolution cycle proposed in our approach.

C. Domain Requirements Engineering

Requirements engineering and management approaches provide an understanding of the key models and types of requirements as well as stakeholders and core evolution activities. Moon *et al.* [15] describe a process for developing domain requirements explicitly considering commonality and variability. They also describe an environment for managing commonality and variability analysis of domain requirements. Thurimella *et al.* [16] propose a rationale-based approach to support PL evolution and to handle PL requirements. It is based on the Questions, Options and Criteria model and uses a modified version of EasyWinWin. This work provides basic foundations for requirements negotiation in the evolution cycle that are also relevant in our approach. Etien and Salinesi [17] present a framework that defines challenges for RE caused by concurrently evolving components of a system. They show an approach addressing co-evolution in RE based on five defined dimensions of management issues.

D. Software Processes

Existing process models and frameworks provide useful definitions of key activities for our reactive approach to PL evolution. April *et al.* [18] propose a maturity model covering important activities of software maintenance and evolution. Deelstra *et al.* [6] describe a product derivation process including discussions of different types of PL adaptations. According to scoping and evolution of PL assets they identified several problems that provide important insights for developing our tool-supported approach. Clements *et al.* [19] discribe applications and adaptations of known project management practices in the context of PLE. Regarding scoping Noor *et al.* [20] propose a collaborative approach for PL planning showing the need to maintain a balance between agility and more disciplined processes in PLE. Ghanam and Maurer [21] present a test-driven approach for agile PLE. They enable organizations with agile practices to manage variability and its evolution based on established agile concepts and test artifacts.

III. Proposed Solution

As the basic idea of PLE is to exploit anticipated variability, we are still missing methods with support for cycles of unanticipated evolution. Based on our existing research in PLE – i.e., the model-based DOPLER tool suite [7] − the *goal of this research is to develop a tool-supported method that supports rapid and iterative evolution in model-based product lines*. In particular, we focus on the challenge of reactive PL evolution in environments with PL models and multiple concurrent application engineering projects that use the PL models. This scenario is mission-critical in the domain of our industry partner but is hardly supported by current PLE methods and tools. In particular, we focus on the scenario of incrementally evolving a model-based product line by analyzing application requirements elicited during product derivation in customer projects.

A. Iterative Process

Our tool-supported method will address the five activities in the spiral process shown in Fig. 2.

Derive and Update Products. Sales people and application engineers initiate *customer projects* to derive products from the PL using the variability models. *Product releases* are created after each iteration in the customer project and after new releases of the PL. An important research challenge lies in understanding which ongoing customer projects and deployed products are affected by new releases of the PL and its models. It is also challenging to document the per-

formed customizations such that they can be reapplied to updated product releases if necessary.

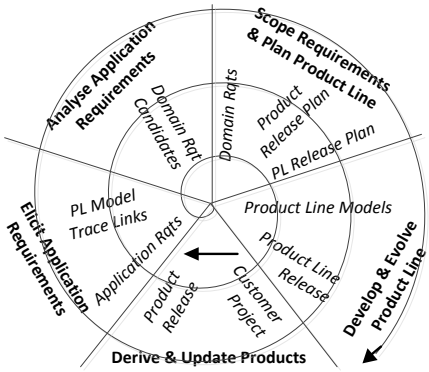

Figure 2. Iterative evolution in model-based PLE. Activities and key artifacts.

Elicit Application Requirements. Analysts document *application requirements* in concurrent customer projects and establish *trace links* to product line assets and variation points to mark artifacts that need to be updated. After each project iteration customer wishes are checked with ongoing customer-specific developments or PL updates. A research issue lies in providing tool support for maintaining trace links between application requirements and PL artifacts (i.e., the variability models) during the project iterations.

Analyze Application Requirements. Application engineers and domain experts analyze and inspect requirements to find similar or duplicate requests from different projects. A research issues is to provide tool support for searching, filtering and collaboratively discussing application requirements when elaborating the *domain requirement candidates*.

Scope Requirements and Plan Product Line. PL managers, product managers, project managers, domain experts and application engineers negotiate the domain requirement candidates and decide about the evolution path of the PL. They need to agree on *new domain requirements* and their priorities when creating a *product line release plan* which has to be consistent with the ongoing customer projects and the *product release plans.* A challenge here lies in identifying and resolving conflicts during negotiations about domain requirements. For the tasks of conflict analysis, scoping, prioritizing and release planning a negotiation model is needed that considers the specifics of PL evolution.

Develop and Evolve the Product Line. PL engineers need to locate *changes* not only in assets but also in *product line models*. They need to adapt the variability models to ensure a consistent *product line release*. A research issue lies in providing support for adapting large model-based product lines based on fine-grained trace information.

B. EvoKing Tool in the Model-based DOPLER Tool Suite

As part of the ongoing research we have been developing the extendable EvoKing tool [9] for tracking evolution and changes in Eclipse-based modeling environments. This framework extends the PLE tool suite DOPLER [7] and will support our described iterative approach with fine-grained change tracking to maintain traceability and gather information needed throughout the evolution cycle.

The evolution data and traceability information provided by EvoKing will be used to support and guide engineers in the five PL evolution activities. The EvoKing prototype can track arbitrary PL artifacts and allows collecting application requirements from multiple concurrent customer projects as a prerequisite for subsequent requirements scoping [4] (cf. Fig. 3).

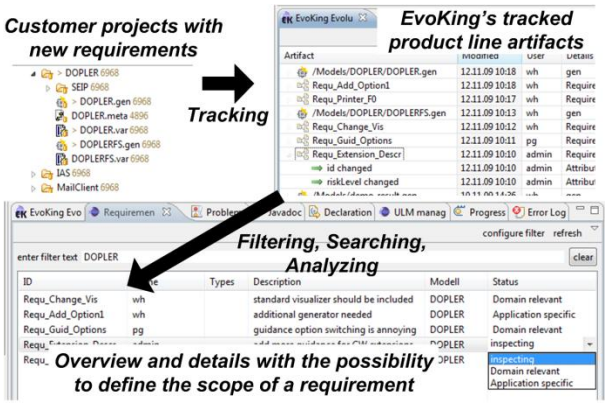

Figure 3. EvoKing tracks requirements in customer projects and provides an overview for planning subsequent product line evolution.

IV. RESEARCH METHODS AND PROGRESS

The work plan for this research is structured into three iterations in which we perform *observation, development, and evaluation* activities (cf. Table 1) contributing to our overall research goal.

TABLE I. RESEARCH ITERATIONS AND PHASES.

Activity	Iteration 1	Iteration 2	Iteration 3
Observation	Industrial PL; find tool requirements	Interviews with engineers from industry partner	Track evolution of industrial PL with tracking tool
Development	Provide initial process definition; develop EvoKing tracking tool	Extend EvoKing tool to guide engineers through evolution	Refine process and tool based on feedback from users
Evaluation	Validate tool requirements using the DOPLER PL case study	Perform experiments and structured interviews with engineers from industry partner	Perform simulation to investigate impact on scalability

During the *first iteration* we have already observed the model evolution of our industry partner's PL. We elicited tool requirements for efficiently evolving the product line models. We have been developing a prototype of the EvoKing tracking tool [9] that allows monitoring the evolution of model-based product lines. The evolution tracking tool will be evaluated during the observation phases in all re-

search iterations. Additionally, we are working on a case study in which we monitor modeling activities required during the refactoring and evolution of the DOPLER tool-suite PL [7][22].

In the *second iteration* we will conduct interviews with the engineers maintaining the DOPLER product line and observe the workflow of maintenance. This will help us to discover further necessary tool support in the product line evolution cycle. By enhancing the EvoKing tool we plan to address these gaps. The records from ongoing product line evolution will be used to create realistic maintenance tasks for experiments with engineers. Our evolution tracking tool will help gathering data allowing a quantitative evaluation of the time and effort required for the tasks. As part of the experiments we will use structured interviews and receive qualitative feedback from the product line engineers at the end of this second iteration.

The *third research iteration* will we based on observing an evolving product line of our industry partner. In particular we will investigate the usability of our tools to further improve the process definitions and tools. Additionally, we plan to evaluate the scalability of our approach using our existing simulation framework for product lines we developed in earlier work [23].

V. CONTRIBUTIONS

We aim to provide three contributions. Our first contribution is the definition of a process for the reactive and iterative evolution of model-based product lines. The approach includes definitions of activities, involved artifacts and presents the entire workflow to manage evolution. A second contribution is the EvoKing tool for tracking the evolution of a model-based product line. It is customizable to different environments for integrated tool support to guide engineers during product line evolution. Finally, we will present case studies and experiments regarding the evolution of model-based product lines in our lab and of our industry partner.

REFERENCES

[1] M. Svahnberg, and J. Bosch, "Evolution in software product lines: two cases," *J. of Software Maintenance: Research and Practice*, vol. 11 (6), pp. 391-422, 1999.

[2] K. Pohl, G. Bockle, and F. Van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*, Springer, 2005.

[3] J. D. McGregor, *The evolution of product line assets*, Technical Report CMU/SEI-2003-TR-005 ESC-TR-2003-005, CMU/SEI, 2003.

[4] W. Heider, and R. Rabiser, "Tool Support for Evolution of Product Lines through Rapid Feedback from Application Engineering," *Proc. 4th Intl. Workshop on Variability Modelling of Software-intensive Systems* (VaMoS 2010), Linz, Austria, 2010, pp. 167-170.

[5] P. Clements, and C. W. Krueger, "Point/Counterpoint," *IEEE Software*, vol. 19(4), pp. 28-31, 2002.

[6] S. Deelstra, M. Sinnema, and J. Bosch, "Product derivation in software product families: a case study, " *J. of Systems and Software*, vol. 74, no. 2, pp. 173-194, Jan., 2005.

[7] D. Dhungana, R. Rabiser, P. Grünbacher, and T. Neumayer, "Integrated Tool Support for Software Product Line Engineering, " *Proc. 22nd IEEE/ACM Intl. Conf. on Automated Software Engineering* (ASE 2007), Atlanta, USA, 2007, pp. 533-534.

[8] D. Dhungana, P. Grünbacher, R. Rabiser, and T. Neumayer, "Structuring the Modeling Space and Supporting Evolution in Software Product Line Engineering," *J. of Systems and Software*, vol. 83(7), pp. 1108-1122, 2010.

[9] W. Heider, R. Rabiser, D. Dhungana, and P. Grünbacher, "Tracking Evolution in Model-based Product Lines," 1st Intl. Workshop on Model-driven Approaches in Software Product Line Engineering (MAPLE 2009), *Proc. (vol 2) of the 13th Int. Software Product Line Conf.* (SPLC 2009), San Francisco, USA, 2009, pp. 59-63.

[10] J. Bosch, *Design and Use of Software Architectures: Adopting and Evolving a Product Line Approach*, ACM Press/Addison-Wesley Publishing Co., New York, USA, 2000.

[11] P. Knauber, "Managing the Evolution of Software Product Lines," *8th Intl. Conf. on Software Reuse* (ICSR-8), Madrid, Spain, Springer LNCS, 2004.

[12] G. Deng, J. Gray, D. Schmidt, Y. Lin, A. Gokhale, and G. Lenz, "Evolution in model-driven software product-line architectures," in *Designing Software-Intensive Systems*, P. Tiako, Ed., Idea Group Inc (IGI), pp. 1280-1312.

[13] J. M. Sprinkle, *Metamodel driven model migration*, Ph.D. thesis, Vanderbilt University.

[14] C. Salinesi, A. Etien, and I. Zoukar, "A Systematic Approach to Express IS Evolution Requirements Using Gap Modelling and Similarity Modelling Techniques," *Proc. of 16th Intl. Conf. on Advanced Information Systems Engineering* (CAiSE04), Riga, Latvia, pp. 338-352.

[15] M. Moon, K. Yeom, and H. Seok Chae, "An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line," *IEEE Transactions on Software Engineering*, vol. 31, no. 7, July, 2005, pp. 551-569.

[16] A. K. Thurimella, and B. Bruegge, "Evolution in Product Line Requirements Engineering: A Rationale Management Approach," *Proc. of 15th IEEE Intl. Requirements Engineering Conf.* (RE 2007), 2007, pp. 254-257.

[17] A. Etien, C. Salinesi, "Managing Requirements in a Co-evolution Context," *Proc. of 13th IEEE Intl. Requirements Engineering Conf.* (RE'05), 2005, pp. 125-134.

[18] A. April, J. Huffman Hayes, A. Abran, and R. Dumke, "Software Maintenance Maturity Model (SMmm): the software maintenance process model," *J. Softw. Maint. Evol.*, vol. 17, pp. 197–223, 2005.

[19] P. Clements, L. Jones, L. Northrop, "Project Management in a Software Product Line Organization," *IEEE Software*, vol. 22 (5), pp. 54-62, 2005.

[20] M. A. Noor, R. Rabiser, and P. Grünbacher, "Agile Product Line Planning: A Collaborative Approach and a Case Study," *J. Systems and Software*, vol 81 (6), June 2008, 868-882.

[21] Y. Ghanam, and F. Maurer, "Extreme Product Line Engineering: Managing Variability and Traceability via Executable Specifications," *Proc. of Agile Conference*, 2009, pp. 41-48.

[22] P. Grünbacher, R. Rabiser, and D. Dhungana, "Product Line Tools Are Product Lines Too: Lessons Learned from Developing a Tool Suite*," Proc. 23rd IEEE/ACM Intl. Conf. on Automated Software Engineering*, L'Aquila, Italy, 2008, pp. 351-354.

[23] W. Heider, R. Froschauer, P. Grünbacher, R. Rabiser, and D. Dhungana, "Simulating Evolution in Model-based Product Line Engineering," *J. Information and Software Technology*, vol. 52 (7), pp. 758-769, 2010.