# Improving Awareness during Product Derivation in Multi-User Multi Product Line Environments

Rick Rabiser Paul Grünbacher Gerald Holl Christian Doppler Laboratory for Automated Software Engineering Johannes Kepler University, Linz, Austria rabiser@ase.jku.at

Abstract-Existing product derivation approaches focus on support for single users resolving variability based on a single variability model. However, in practice multiple users perform product derivation of large-scale systems with system-ofsystems architectures in a distributed and asynchronous manner. It is infeasible to describe the variability of such multi product lines with one integrated model. Rather, several models are needed. Existing research mainly focuses on supporting modeling in multi product lines. The aim of our ongoing research is however to improve awareness for multiple users involved in product derivation in such environments, i.e., to make users configuring one subsystem aware of the relevant configuration decisions made for other subsystems. In this paper we describe an industrial scenario of a multi product line and derive requirements for awareness support. We present a preliminary approach based on the "publish/subscribe" pattern and a tool prototype that aims at improving awareness in product derivation by sharing decisions across different derivation projects.

### Keywords-multi product lines; product derivation; awareness

## I. INTRODUCTION AND MOTIVATION

The fundamental assumption of product line engineering is that the investments required to set up a product line are outweighed by accelerating product derivation. During product derivation, diverse stakeholders resolve the variability of the product line which is usually described in variability models. Product derivation is a complex process and directly affects the success of adopting product line engineering [1][2]. In product derivation users make decisions on different levels ranging from high-level decisions (e.g., leading to the inclusion of a particular subsystem) to fine-grained adjustments (e.g., leading to the setting of parameters for software components). The typical use case in existing product derivation approaches and tools [3] is a single user working on a single variability model of a product line. However, in practical settings of large and ultra-large-scale systems there is a strong trend towards multi product lines, systemof-systems architectures, and software ecosystems [4]. A multi product line represents a collection of smaller and interdependent product lines which together form a largescale system. In such environments, multiple models represent different parts of a system and multiple users are

involved in product derivation. As a result multiple product derivation projects occur concurrently to configure different subsystems.

Existing research has focused on establishing explicit links among variability models [6][7] and on composing variability models to build one integrated model [8] as the basis for subsequent derivation. This is however often unrealistic due to the size, heterogeneity, and complexity of current systems. In our collaboration with Siemens VAI Metals Technologies [9], the world's leading steel-plant building company, we have learned that for large-scale softwareintensive systems such explicit linking and merging is often very difficult. Steel plant automation software comprises different subsystems on different levels such as the machineoriented level 1 for basic automation, the process-oriented automation level 2, the enterprise resource planning level 3, as well as maintenance and setup systems.

When modeling the variability of such systems and developing support for product derivation we observed that while some decisions are relevant across several systems, it is hard to identify these decisions already during domain engineering and to establish and maintain explicit dependencies. Also, different technologies used to develop different subsystems make it infeasible to fully integrate variability models during domain engineering as there is no modeling language that could handle all these different cases uniformly under one umbrella. To support such situations better, we have thus been working on an approach for communicating key derivation decisions in multiple concurrent derivation projects without explicitly integrating the underlying variability models. In this use case users are "loosely coupled" and are only made aware about key decisions during product derivation. We intentionally do not focus on the technical implications of the configuration decisions (e.g., the actual configuration of software artifacts) as part of this research.

The approach presented in this position paper thus emphasizes the improvement of *awareness in product derivation* meaning that *users configuring one system are informed about the relevant decisions made for other systems*. In Section 2 we describe a motivating scenario based on an existing multi product line and specify requirements for awareness support. In Sections 3 and 4 we present a preliminary approach and tool support. We discuss related work in Section 5 and conclude the paper in Section 6.

# II. REQUIREMENTS FOR PRODUCT DERIVATION AWARENESS SUPPORT

Our industry partner Siemens VAI offers complete solutions for steel plants. Mini-mills for instance are multi product line systems supporting the entire steel-making process starting with melting the raw material to liquid steel, continuing with casting liquid steel to slabs, and ending with rolling cast steel slabs. Diverse software-intensive systems are part of a mini-mill such as automation software for casting, rolling, etc. on different levels of automation and granularity (e.g., machine-oriented automation, process automation, enterprise resource planning). These systems constitute a multi product line that is defined in diverse variability models representing the variability of heterogeneous subsystems.

We present a simple scenario (cf. Figure 1) to analyze and illustrate required awareness support in multi-user multi product line derivation using the mini-mill example. We assume that variability models of the subsystems already exist. However, as these variability models describe heterogeneous reusable assets they can be based on different metamodels which makes their integration hard if not impossible. The scenario has been developed based on discussions with software architects, project managers, and researchers from our industry partners Siemens VAI and Siemens CT. It is also based on experiences with applying our DOPLER tool suite to different industrial use cases.

1) Each individual product derivation project starts when the responsible stakeholder "instantiates" a variability model describing a subsystem by preparing it for derivation [10]. This can happen in parallel or sequentially. For example, one user could start deriving the level 2 automation software for continuous casting (user 1) while another user starts deriving the level 1 software for rolling (user 2).

2) The involved users begin making decisions to resolve variability. For instance, at some point user 2 realizes that when deciding about the type of rolling required she needs to know about pending decisions to be made by another user concerning the number of strands in mini-mill caster.

3) User 1 publishes decisions she regards as relevant for others. For instance, user 1 might set the number of strands of the caster (decision c in Figure 1) to two. This directly affects the second user's decisions as now rolling of steel coming from two strands has to be supported. User 1 might be unaware of the current derivation tasks of user 2 but she knows the number of strands might be relevant to others.

4) User 2 and all other users participating can query all publicly shared decisions (like the number of strands) before making the decisions for their own product line accordingly.

5) If a shared decision is important for one user then the change of its value by another user might have consequences. Therefore, the user might be interested in "subscribing" to certain decisions to be informed after changes. For example, changes to the decision on the number of strands (cf. decision c in Figure 1) need to be communicated to user 2 as this affects rolling (cf. decision i in Figure 1).

From the described scenario we derive several initial requirements for awareness support in product derivation:

Distributed and asynchronous derivation support. In large-scale systems users and work groups need to be able to derive and configure their subsystems largely independently and in a distributed and asynchronous manner. It is however important that there are well-defined communication interfaces between the groups responsible for different subsystems. In industrial settings work groups are often distributed and teams collaborate on a world-wide scale (e.g., there are regional sales offices that need to collaborate with technical staff in the configuration process). Support for distributed work is thus essential.

Mechanisms for sharing decisions with other users. Users need to be aware of decisions made by others that possibly affect their own configuration tasks. They also need support to inform others about their own decisions and to unobtrusively share this information.

*Establishing mappings between shared and local decisions.* It should be possible for users to define traceability between subscribed shared decisions and local decisions. If then a shared decision is changed the trace link can be used to notify the user about potentially affected local decisions. This requirement is particularly important to incrementally develop an understanding of the configuration dependencies between different subsystems which are typically not completely known in advance and subject to continuous change.

Integration with existing derivation tools. Existing support for product derivation shall not be replaced. Instead, support for awareness in product derivation should be seamlessly integrated in existing tools.

# III. APPROACH

We have been developing such awareness support in product derivation based on DOPLER, a decision-oriented product line approach [8] which has successfully been applied in different domains such as industrial automation [9][10] or enterprise resource planning [11].

# A. Background: The DOPLER Approach

DOPLER variability models contain Assets and Decisions. Assets represent the reusable product line artifacts (e.g., components, subsystems). Variation points are defined and presented as decisions. Important attributes of decisions are a unique id, a question that is asked to a user during product derivation, and a decision type (Boolean, enumeration, string, or number). Decisions can depend on each other hierarchically (if a decision needs to be made before another decision) or logically (if making a decision changes the answer to another decision). The decision type describes the range of possible answers which can be further constrained with validity conditions. In DOPLER, assets are linked to decisions via inclusion conditions defining when a particular asset is included in a derived product. Asset attributes can also depend on decisions to enable customization of assets.

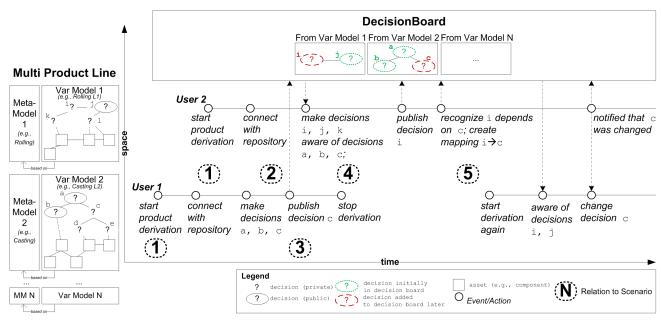


Figure 1. The DecisionBoard enables awareness in product derivation in multi product lines. In the example two users perform product derivation for two (sub-)systems defined by two variability models that are based on different meta-models. They make decisions locally and publish selected decisions to the decision board. Users can consult the decision board about decisions shared by others. For instance, user 2 creates a mapping from decision i (e.g., rolling mode in the mini-mill) to c (e.g., number of strands in the caster) to be notified after changes to the number of strands.

Decisions in DOPLER can be private (default) or public. Originally this attribute has been introduced to support integrating variability model fragments in domain engineering (merging them to one integrated model). The DOPLER model merging approach [8] allows defining model elements such as decisions or assets as placeholders by only defining their name and type. Later, the placeholders are resolved with concrete public decisions and assets from other models based on name and type matching. Merging models with this approach works well for single product lines if all models to be merged are based on the same meta-model. However, this assumption does not hold in the multi product line scenario. For the approach outlined in this paper we benefit from DOPLER's concept of public and private model elements. Our focus however is not on merging models during domain engineering. Instead, we aim at making users aware of public decisions during derivation independent of the meta-models or technical solution assets used in the different subsystems. We try to make our approach as simple as possible as it is common that domain experts with no deep technical knowledge in variability modeling are involved in derivation.

# B. DecisionBoard Approach

We thus propose a *decision board* realized as a shared repository that allows the participating users to *publish* their decisions in the repository and to *subscribe* to decisions in order to be aware about other projects. When users first connect to the repository, decisions already set as public during variability modeling are published automatically. Users can however decide to publish additional decisions during product derivation. Furthermore, users can define links from their local private decisions to shared public decisions. These links are used to notify users as soon as the value of a shared relevant decision changes.

Figure 1 depicts the basic idea of our approach in relation to the scenario described in Section 2 and shows the participating components of our approach (multi product line, multiple users, decision board, shared and local decisions).

## IV. TOOL PROTOTYPE

We have implemented the approach as an extension to the existing DOPLER product derivation tool ConfigurationWizard [9][10]. This tool enacts product line models and is used during product derivation by domain experts and engineers to interactively derive a concrete product from a product line. The tool presents decisions to responsible users in different views based on their roles and tasks. Visualization and filtering capabilities allow working with large product line models. The tool explains dependencies to ease navigation in the model. As soon as users make decisions, the ConfigurationWizard gives immediate visual feedback about the consequences. The ConfigurationWizard provides several extension points. For example, one extension point is for integrating product configuration and documentation generators. Using the Eclipse extension point mechanism, additional views can be integrated in the ConfigurationWizard tool. We have implemented our DecisionBoard by developing such an additional view (cf. Figure 2). The decisions are stored in a shared repository at an arbitrary URL. They are organized and grouped by their originating variability models of the involved subsystems.

oject <u>E</u> dit <u>S</u> earch <u>R</u> ole <u>W</u> indow <u>H</u> elp				
📓 🖳 🗁 🗟 🐗 🛛 🖓 🕎 🏠 🦂 🔸	👫 🗞 Aa 🧔	] • ↓ <mark>a</mark> • ≱ •	• 🦀 • i	
) *Decisions (C:/CW/demo_rolling.gen)				🧼 🔖 Aa '
ecision (rolling_mode)				
<ul> <li>Shall the production process be are</li> </ul>	hived conforming Is	SO900X?	ves	
<ul> <li>Init the product provided by Siemens VAI?</li> <li>What is the scope of the product to be delivered?</li> </ul>			yes	
Which rolling modes are supported	?	Trij		· 2
Which rolling modes are supported  Required Assets (64) (E) Effects of answering the	?		(Decision Board)	
Which rolling modes are supported Required Assets (64) Effects of answering the Question	current decision 🕥	Shared Decisions	le	
Which rolling modes are supported Required Assets (64) [2] Effects of answering the Question	current decision (2) Answer	Shared Decisions	(Decision Board)	
Which rolling modes are supported Required Assets (64) Effects of answering the Question Can the mold width be changed?	current decision 2	Shared Decisions Source Casting L2	(Decision Board) Mapped to	
Which rolling modes are supported Required Assets (64) (E. Effects of answering the Question Can the mold width be changed? Which casting modes are supported?	current decision 2 Answer true Triple	Shared Decisions Source Casting L2 Casting L2	CDecision Board) Mapped to rolling_mode	
Which rolling modes are supported Required Assets (64) Cuestion Cuestion Con the mold width be changed? Which casting modes are supported? Which casting modes are supported? Which casting modes are have?	current decision 2 Answer true Triple 2.0	Shared Decisions Source Casting L2 Casting L2 Casting L2 Casting L2	(Decision Board) Mapped to rolling_mode	

Figure 2. DecisionBoard extension in the ConfigurationWizard derivation tool. The view on the lower half shows shared decisions and their mappings to local decisions shown in the upper half, if any. The user is notified if a shared decision mapped with a local decision changes.

#### V. RELATED WORK

The approach presented in this paper is related to existing work on coordinating complex configuration processes during product derivation. For instance, Mendonca and Cowan [12] propose a collaborative product configuration approach that aims to support coordinating teamwork decisionmaking in the context of product derivation. Our approach in contrast follows the asynchronous concept of publish/subscribe to coordinate teams. Czarnecki et al. [13], present a staged approach based on the idea of stepwise specialization of feature models. Our approach does not specialize models over time but aims to enable sharing decisions from multiple models. In [6], Reiser et al. address the problem of feature modeling in large-scale embedded systems and propose product sets and configuration links to define dependencies between different feature models to be evaluated at the time of their selection. This approach requires to pre-define explicit links which we wanted to avoid with our approach. Hubaux et al. [14] propose feature configuration workflows (defined in a workflow language) as a new formalism for supporting configuration of large-scale systems based on feature models. We do not define such a workflow but rather let the users decide when to share/use which decision. The issue of derivation in multi product lines is also addressed in our earlier work [11] where we demonstrated how decision models can be used to support the configuration of complex systems across multiple levels of software vendor, customers, and end-users. In this earlier work, we however assumed explicit dependencies among the different levels. Elsner et al. [15] propose an approach for product configuration that works across the boundaries of multi product lines. Their framework converts meta-models of the different product lines into a common meta-model format (Eclipse EMF Ecore). It provides real-time constraint checking in product configuration based on constraints defined by domain experts for the different product lines. Our approach does not require the meta-models to be converted as it focuses on the configuration level only, i.e., sharing decisions.

Our work is also related to the field of collaborative design. For instance, Wang *et al.* [17] evaluate different approaches for collaborative design and conclude that conventional approaches for sharing design decisions are insufficient. They propose web-based or agent-based collaborative design spaces to for publishing information needed during collaboration in the design process. Pahng *et al.* [16] propose a web-based collaborative design framework where designers can model a system in a distributed way. It enables the designer to see the remote effects of a local decision.

## VI. CONCLUSIONS AND FUTURE WORK

In multi product line environments it is important to ensure knowledge transfer across multiple concurrent product derivation projects with regard to different decisions made. In this paper, we have thus presented an approach and initial tool support for awareness in product derivation based on a motivating scenario from a real-world multi product line. Although the approach is intentionally simple we are confident that it can improve awareness about important configuration decisions in multi product line scenarios without the need to explicitly integrate diverse variability models that are hard, infeasible, or even impossible to integrate.

In future work we will refine and validate the approach in user studies. An important research issues is to explore the trade-off between sharing information early and often (thus risking information overload) vs following a more restrictive policy. It is important to understand the balancing between the early propagation and likelihood of the information changing. We also expect that the user studies will contribute to a deeper understanding of the high-level requirements discussed in Section 2.

Furthermore, we plan to address several issues related to the tool prototype: it is important to avoid the deterioration of the decision board after continuous updates and we will develop filtering and search features for the decision board. Another challenge lies in detecting inconsistencies in the decision board (e.g., caused by different users publishing the same decision from two copies of the same model). Finally, we will investigate the integration of the decision board approach with our model merging approach [8] (e.g., by analyzing and exploiting mapping links set by users). We plan to evaluate and improve the approach in case studies with industrial users.

## ACKNOWLEDGMENT

This work has been conducted in cooperation with Siemens VAI Metals Technologies and Siemens Corporate Technology and has been supported by the Christian Doppler Forschungsgesellschaft, Austria.

#### REFERENCES

- S. Deelstra, M. Sinnema, and J. Bosch, "Product derivation in software product families: a case study," Journal of Systems and Software, vol. 74(2), pp. 173-194, 2005.
- [2] P. O'Leary, R. Rabiser, I. Richardson, and S. Thiel, "Important Issues and Key Activities in Product Derivation: Experiences from Two Independent Research Projects, "Proc. of the 13th International Software Product Line Conference (SPLC 2009), San Francisco, CA, USA, ACM ICPS, CarnegieMellon University, 2009, pp. 121-130.

- [3] R. Rabiser, P. Grünbacher, and D. Dhungana, "Requirements for Product Derivation Support: Results from a Systematic Literature Review and an Expert Survey," Information and Software Technology, vol. 52(3), pp. 324-346, 2010.
- [4] J. Bosch and P. Bosch-Sijtsema, "From integration to composition: On the impact of software product lines, global development and ecosystems," Journal of Systems and Software, vol. 83(1), pp. 67-76, 2010.
- [5] C. W. Krueger, "New Methods in Software Product Line Development", Proc. of the 10th International Software Product Line Conference (SPLC 2006), Baltimore, Maryland, 2006.
- [6] M.-O. Reiser, R. T. Kolagari, and M. Weber, "Compositional Variability—Concepts and Patterns, "Proc. of the 42nd Hawaii International Conference on System Sciences (HICSS-42), Waikoloa, Hawaii, USA, IEEE Computer Society, 2009.
- [7] R. van Ommering, "Software reuse in product populations," IEEE Transactions on Software Engineering, vol. 31(7), pp. 537-550, 2005.
- [8] D. Dhungana, P. Grünbacher, R. Rabiser, and T. Neumayer, "Structuring the Modeling Space and Supporting Evolution in Software Product Line Engineering," Journal of Systems and Software, vol. 83(7), pp. 1108-1122, 2010.
- [9] P. Grünbacher, R. Rabiser, D. Dhungana, and M. Lehofer, "Modelbased Customization and Deployment of Eclipse-Based Tools: Industrial Experiences, "Proc. of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE 2009), Auckland, New Zealand, IEEE/ACM, 2009, pp. 247-256.
- [10] R. Rabiser, P. Grünbacher, and D. Dhungana, "Supporting Product Derivation by Adapting and Augmenting Variability Models, "Proc. of the 11th International Software Product Line Conference (SPLC 2007), Kyoto, Japan, IEEE Computer Society, 2007, pp. 141-150.

- [11] R. Rabiser, R. Wolfinger, and P. Grünbacher, "Three-level Customization of Software Products Using a Product Line Approach, "Proc. of the 42nd Annual Hawaii International Conference on System Sciences, Waikoloa, HI, USA, IEEE CS, 2009.
- [12] M. Mendonca and D. Cowan, "Decision-making coordination and efficient reasoning techniques for feature-based configuration," Science of Computer Programming, vol. 75(5), pp. 311-332, 2009.
- [13] K. Czarnecki, S. Helson, and U. W. Eisenecker, "Staged configuration using feature models, "Proc. of the 3rd International Software Product Line Conference (SPLC 2004), Boston, MA, USA, Springer Berlin Heidelberg, 2004, pp. 266-283.
- [14] A. Hubaux, A. Classen, and P. Heymans, "Formal modelling of feature configuration workflows, "Proc. of the 13th International Software Product Line Conference (SPLC 2009), San Francisco, CA, USA, ACM International Conference Proceeding Series; Vol. 446, CarnegieMellon University, 2009, pp. 221-230.
- [15] C. Elsner, P. Ulbrich, D. Lohmann, and W. Schröder-Preikschat, "Consistent Product Line Configuration Across File Type and Product Line Boundaries., "Proc. of the 14th International Software Product Line Conference (SPLC 2010), Jeju Island, South Korea, 2010 (to appear).
- [16] G.-D. Pahng, S. Bae, D. Wallace, "Web-based collaborative design modeling and decision support", Proc. of the ASME Design Engineering Technical Conferences (DETC 1998), Atlanta, Georgia, 1998
- [17] L. Wang, W. Shen, H. Xie, J. Neelamkavil A.Pardasani, "Collaborative conceptual design—state of the art and future trends", Computer-Aided Design, vol. 34, pp. 981-996, 2002