Multi-Perspective Modeling and Performance Analysis of Software Product Lines

Matthias Kowal
TU Braunschweig, Germany
Email: m.kowal@tu-braunschweig.de

***Abstract*—Software system are typically available in a rich set of variants nowadays to deal with differing customer or environmental requirements and application contexts. Managing such a software product line, gets even more difficult considering multiple involved engineering disciplines and long lifetimes, as typical for industrial systems of the automation domain. The thesis tackles this system diversity by modeling interdisciplinary system variability in both problem and solution space. Based on these models, we analyze the impact on performance properties during design time giving early feedback about the system behavior. The solution space is based on a model-driven approach with UML models, using notions of delta modeling to manage system variability and evolution enriched with information needed to automatically derive and analyze a performance model. Motivated by its widespread use in software engineering, we consider feature models for the problem space that are ultimately connected to the UML models. Our evaluation is based on a real-world automation system representing a long-living and variant-rich software system.**

I. Introduction

Long-living software systems, as in the automation domain, exist in many different variants at one point in time in order to satisfy varying user requirements (anticipated variability in space). Additionally, they evolve over time in order to adapt to changing environmental conditions, such as functional, technical or legal requirements (unanticipated variability in time) [1]. Furthermore, several disciplines are involved during their development with mechanical, electric and software engineering. As a result, product-line engineering for such systems is extremely complex. A product line is typically divided into problem and solution space. The problem space captures domain knowledge in an abstract way, e.g., using feature or decision models. The solution space consists of concrete implementation artifacts such as source code fragments or UML-diagrams. Hence, we need an interdisciplinary variability modeling concept on the problem space level spanning across the three mentioned disciplines. For the solution space, we need models with an appropriate level of abstraction to support the individual developers following the separation of concerns principle, so that, e.g., software architects and engineers can focus on their specific task and unnecessary information stays hidden. A fully integrated model-driven approach also calls for the connection of both spaces to support developers as much as possible and enable the generation of the configured variants in the problem space with the respective solution space models. In each part, we have to identify suitable methods to keep the complexity as low as possible to achieve a high level of usability.

We propose a twofold modeling approach managing problem and solution space as well as their mapping. Software product lines are commonly modeled using feature models in the problem space [2]. There is already a widespread number of extensions to feature models available in literature, e.g., dealing with different versions over time [3], view-based abstractions [4] or even combining multiple product lines [5]. However, they all lack appropriate adaptations to deal with interdisciplinary aspects, since they were originally designed for software engineering and not mechanics and electronics [6]. Nonetheless, we plan to use them as a foundation for our approach and extend them where it is necessary.

For the solution space, we propose a design-level modeling approach using three levels of abstraction containing a workflow describing the path of a job through the system, an architecture and the actual behavior. In addition, we provide a mapping between individual elements of each perspective. Variability and evolution is managed using delta modeling [7]. In delta modeling, we have a core model, which often is the smallest possible variant of the system. The core can be altered using the three basic operations of *add, remove*, and *modify*. Deltas encapsulate these change operations and therefore contain the required information to derive a specific variant. However, the application of one or more deltas to generate another variant of the system may lead to an ill-formed system model. Hence, we have to validate that the resulting variant is still consistent. In each space, we pursue the established principle of separation of concerns to minimize the complexity. The perspectives are realized with a subset of UML in terms of diagrams with activity, architecture and state charts.

Predicting the system behavior of software product lines during design time, in terms of performance, is a crucial yet complex task. An automation system representing a product line typically has several non-functional requirements. For instance, it must produce or transport a specific number of items in a given amount of time. Feedback concerning these requirements would be beneficial at an early stage to prevent disadvantageous design decisions. However, the analysis of each variant in isolation, which is called product-based performance analysis [8], is inefficient due to the possible large number of variants in product lines. We need appropriate methods to overcome this aspect. Family-based approaches

allow a more efficient analysis of a complete product line [9]. We propose such a family-based analysis based on our solution space models to provide early feedback giving developers the chance to identify system bottlenecks during design time. As performance properties, we consider throughput, utilization or average queue length of the system.

The essence of this thesis is to provide a scalable modeling approach covering variability and evolution in both problem and solution space across several disciplines while providing efficient performance analyses techniques for product lines.
Research Questions. In detail, the thesis is supposed to answer the following research questions:

RQ1: How can we reuse and adapt existing feature model extensions to deal with an interdisciplinary product line?

RQ2: Can we sufficiently capture variability and evolution of a real-world automation system in our models while maintaining low complexity and high consistency?

RQ3: To what extent is our family-based performance analysis superior compared to existing techniques in terms of runtime?

II. Related Work

Most feature modeling approaches consider one large model specifying the variability of a complete product line [10]. This is not feasible considering multiple disciplines, since each domain solely needs to focus on their specific features and everything else would be unnecessary information. Additionally, one large model is hard to visualize, maintain, and analyze taking evolution into account. Different views onto the feature model or the combination of two or more features models, one for each domain, as in multi software product lines possible, may solve some of the existing problems [4], [5]. There are numerous more feature model extensions and similar approaches available. As for now, it is an open question which methods can be applied to an interdisciplinary product line.

Considering the solution space, we face the same problems with multiple disciplines such as software architects or engineers. Hence, it is a common approach to follow the principle of separation of concerns during development. For instance, this is done in the SPES project in which an embedded system is developed with model-driven engineering and multiple viewpoints [11] or as in [12]. Our approach is quite similar, but also includes a variability modeling concept with delta modeling. Delta modeling is part of the transformational variability modeling concepts as is the Common Variability Language [13]. Comparing both approaches yields no significant benefit for either one, e.g. both are language independent. We argue that both emerged in parallel and we decided to use delta modeling. Other concepts are of annotative or compositional nature. The former tend to have unmanageable large models [14], while the latter can only add functionality making removals impossible [15]. The removal of functionality can be compensated by selecting an appropriate variant as starting point which is incrementally extended through addition afterwards. In delta modeling this starting point can be freely selected across all variants providing more flexibility.

Consistency checking for UML models with variability is not yet largely considered. A brief survey of consistency checking techniques can be found in [16]. Some of the existing techniques have to be further investigated in terms of applicability to product lines [17]. Other approaches already detect inconsistencies with variability [18], yet a mapping to our general delta modeling concept has to be done. Additionally, we aim at fixing the detected inconsistencies. A few of the available techniques in literature already include methods for fixing UML models in single software systems [19]. However, approaches for repairing product lines are still in the early stages. A first research result in this direction can be found in [20]. We plan to build upon this foundational work.

The proposed family-based performance analysis is related to a model checking approach of software product lines in which annotated UML sequence diagrams are used [21]. This ultimately leads to a symbolic expression similar to ours. However, we consider performance properties such as throughput and do not focus energy related parameters. Another approach translates UML-annotated software product line designs to layered queueing networks and analyzes them in a product-based way [22]. Although this network-type is more expressive than ours, an efficient family-based analysis is not available.

III. Integrated modeling and efficient performance analysis

This section is divided into three parts and each part represents a different contribution of the thesis. First, we explain the interdisciplinary modeling of a product line in the problem space. Second, we extend the approach to solution space models. Third, a family-based performance analysis technique is introduced. The overall approach is exemplarily shown in Fig. 1.

The first contribution: Interdisciplinary Modeling of Variability in the Problem Space

First contribution considers the interdisciplinary modeling approach (cf. top-left corner of Fig. 1). We devise problem space variability modeling concepts for long-living software product lines involving multiple disciplines, as typical in the automation domain. Classical feature models [23], [2] lack expressiveness to deal with interdisciplinary systems. Hence, we develop a problem space variability modeling concept capable to capture interdisciplinary variability dealing with the following limitations of existing approaches (as recently identified in [6]): (i) complex dependencies and relations between discipline-specific entities, (ii) several levels of granularity (from actuators to complete software components), and (iii) inclusion of multiple views for the individual domains (mechanics, electronics and software).

We apply existing feature model extensions to the automation system product line. A possible extension is the usage of different views onto the feature model based on individual domains. Furthermore, the principle of multi software product

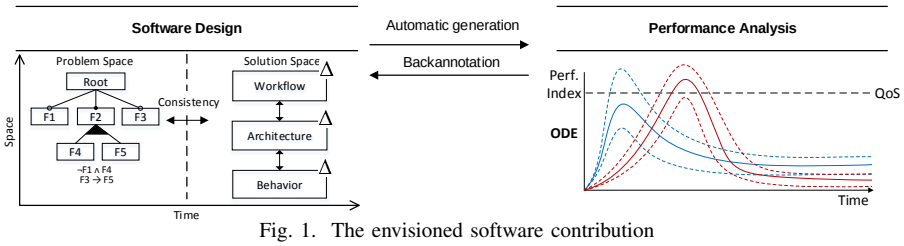

Fig. 1. The envisioned software contribution

lines can be applied to solve the granularity limitation, which basically means having separate feature models for each domain. In addition, extended cross-tree constraints are available in multi software product lines to capture the complex dependencies.

Additionally, we must as well address evolution extending the previously developed variability modeling concept with feature versioning which is required for the case study. We base our approach on Hyper Feature Models (HFMs) [3] which is devised in prior work. HFMs capture feature versions and allow reasoning about dependencies of features in different versions. As HFMs are also originally not designed to handle interdisciplinary models, we adapt HFMs suitably to integrate them with the previously mentioned interdisciplinary concepts. A closer look to traceability approaches across several models may also be beneficial here [24].

We can safely assume that during system development with the described interconnected modeling approach, errors are a steady companion for the developers. Hence, we have to devise analyses techniques for reasoning about consistency to provide a correct and reliable design [6]. We analyze the feature model for anomalies such as dead features meaning features that can never be part a any variant due to constraints. In addition, we have to ensure that the derivable set of product variants is valid, an essential prerequisite for the configuration of software systems. As we also consider the evolution of the feature model itself, all analyses have to work incrementally in order to allow for efficient re-analyses after evolutionary changes. For evaluation purposes, we strive for a close cooperation with automation engineers of the TU Munich, since they are the domain experts for such systems and can give qualitative feedback about the feasibility of our concepts.

Preliminary Results: This contribution is the main task for the remaining time of the thesis. The developed results should answer **RQ1** using an explorative study. A first result about challenges here is given in [25].

The second contribution: Multi-Perspective Modeling of Variability and Performance in the Solution Space

Second contribution is all about solution space models (cf. top-left corner of Fig. 1). The solution space is divided into three perspectives following separation of concerns and therefore minimizing the complexity for individual developers: *workflow*, *architecture*, and *behavior* each represented by a different UML model with activity, block-based, and state chart diagrams. The workflow describes the path of a workpiece throughout the system (e.g., a bottle on an assembly line), while the architecture presumes the system structure and the behavior provides the actual implementation. In addition, elements can be mapped between the different perspectives. We equipped each perspective with the delta modeling approach to capture variability and evolution by the same means. Since deltas can be applied to each perspective in isolation, we provide a meta structure which we call *interdelta* containing information about the required deltas to derive a valid system variant. In order to map features to the solution space, we define higher-order deltas operating on the previously mentioned meta structure for connecting problem and solution space models.

A system in development is under constant change where each modeling perspective may evolve independently, but consistency between different perspectives, e.g., activities mapped to suitable components in the architecture, must be maintained. For instance, an inconsistency is an architectural component without a connection to any state chart. Hence, we need to develop a concept for incremental consistency checking to support the developer when evolving a system within the multi-perspective modeling approach. Again incrementality is key here, as it minimizes the verification time as only changed parts and parts possibly affected by the changes need to be considered. As a conceptual idea for the incremental consistency checking approach, we use previous work on incrementally type checking of delta-oriented product lines in Java and adapt it to our specific needs [26]. Considering tooling the Epsilon framework, which includes the Epsilon Validation Language, may be a promising candidate for an appropriate solution.

We want to provide the developer not only with the knowledge of existing errors (consistency checking), but devise a concept for repair operations as well. Similar to IDE functionality for repair operations in programming languages, e.g. adding the *import* statement for a used function, we want to provide repair operations for our models. For instance, a component in the architecture has to be connected to a state chart in the behavior perspective. If there is no such behavior, a repair operation is to create a default state chart, if the developer complies with the suggestion. The possible cases for inconsistencies have to be explored and fixing operations for these cases have to be provided.

Overall we have models for both problem and solution space with feature and UML models. Each concept must ensure a certain level of consistency in isolation as well as in between

both sides resulting in three points that require consistency concepts.

Preliminary Results: As a first result, we already have developed the design-level solution space modeling approach based on UML diagrams [27], [28]. In addition, we can manage variability and evolution with delta modeling [27], [28], [29]. Finally, we enriched the workflow perspective with non-functional performance properties such as arrival and service rates at the nodes and probabilities at the transitions. This is a prerequisite for the third contribution and allows us to do a performance analysis of the system and predict measures such as utilization, throughput and average queue lengths. As before, this also includes the full delta modeling support to vary performance properties between variants [27], [30]. The implementation is designed as an Eclipse plugin using XText and EMF for the individual domain-specific languages (DSLs) for the modeling perspectives. In total, we have three DSLs for the perspectives and an additional one for the combining meta-structure. Although, the textual editor has comfort functions such as auto-completion, syntax highlighting and basic syntactic consistency checks, we started to develop concepts for graphical delta modeling editors using the Graphical Editing Framework (GEF) for each perspective. We were able to answer **RQ2** as explored in three different publications [27], [28], [29]. Method of evaluation was the real world automation system located at the AIS chair of the TU Munich. However, the consistency concept including repair operations is still an open point in this contribution.

The third contribution: Efficient Performance Analysis.

Third part of the contribution is a scalable performance analysis of the complete product line. The analysis is based on the previously described solution space models, especially the workflow. The workflow is interpreted as a continuous-time Markov chain that underlies a Jackson-type queueing network [31]. Fig. 1 implies an automatic generation of such performance models. A product-based analysis is straightforward possible by solving the following equation for γ

$$(I - P^T)\gamma = \lambda, \tag{1}$$

where I is the identity matrix, P is the adjacency matrix and λ are the arrival rates. We can easily compute the throughput, utilization, average queue length or response time of the system based on the solution [31]. However, for systems with a large variant space such as product lines, it is not efficient to compute the solution for each variant in isolation which is necessary since we cannot reuse any numerical results [32]. We propose a family-based analysis to overcome this disadvantage and analyze the full product line at once. Again, we benefit from the delta modeling principle for our efficient analysis. We create a super-variant, also called 150%-model, based on the core and all deltas so that the full variability is concentrated in exactly one model. Each value that is changed by a delta is represented by a symbol and not a concrete value as in the product-based analysis. Again, we can create the system of equations as in (1), but solve it symbolically in our family-based approach. As a result, we get a large symbolic expression in which we just have to plug-in the values for a specific variant to calculate the performance properties. The symbolic solving has to be done once for the complete product line and not for each variant over and over again. The gathered information about possible system bottlenecks can influence modeling decisions at design time again and we achieve a round-trip engineering concept as depicted in Fig. 1. However, the underlying Jackson-type queueing networks are quite limiting in terms of service time distribution and supported model elements which is why we extended this work to Coxian-distributed networks [33].

Preliminary Results: Main contribution here is the proposed family-based performance analysis in [30]. Numerical experiments demonstrated the superiority of the approach against a product-based solution, especially in the case of large-scale networks with high degree of variability. The experiment shows speed-ups of up to 2000%. The analysis is also evaluated with the automation case study [27]. An extension to Coxian-distributed networks is also available [33]. However, the numerical results are not as good as before due to more complex formulae. A product-based performance analysis works for the workflow in our Eclipse plugin. The family-based analysis is realized in Matlab. Each part of the tooling was evaluated and applied to the automation case study. This includes an actual code generation for an automation control software [27]. As a result, **RQ3** is successfully answered.

To conclude this section, we can safely assume that the solution space concept is mostly finished except for advanced consistency checks and repair operations. It is similar for the performance analysis part. Main focus for the remaining thesis time lies on interdisciplinary concepts and the connection of problem and solution space models.

IV. Evaluation and Status

The following section reflects present and future achievements in a condensed form. In addition, we provide more details about the evaluation methods.

A. Expected Contributions

We expect the following contributions:

- Application of feature modeling to product lines containing hardware and software.
- Management of variability, evolution and consistency in the solution space.
- Performance analysis techniques for product lines.

B. Case Study

The proposed contributions are evaluated using a real-world automation system called the *Pick and Place Unit* (PPU) which is provided by the AIS chair of the TU Munich and exists in 15 different variants [34]. For the first contribution, we conduct an explorative study based on the PPU. Second contribution includes modeling and analyzing the complete PPU product line with our solution space models. Since the PPU is still a lab demonstrator, we have to conduct larger experiments for the third contribution concerning the

performance analysis part to investigate the behavior in large-scale product lines. Finally, we want to empirically compare our textual DSL modeling concept against the graphical one to validate the superiority of visual delta modeling e.g. in terms of speed and understandability. A lab experiment with students is planned in the future.

C. Current Status

Currently six publications have emerged during this thesis tackling solution space models and the performance analysis. Thus, answering **RQ3** completely [30], [33], **RQ2** in many parts [27], [28], [29] except the advanced consistency and repair operation concepts and **RQ1** is still at the beginning [25].

The time line for completion is as follows:

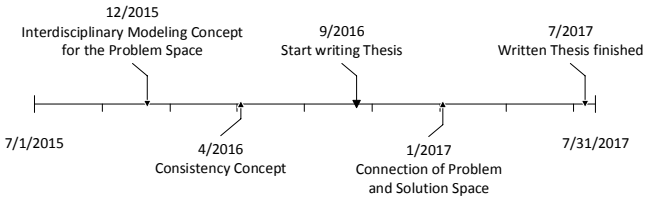

Fig. 2. Timeline for the remaining duration of the thesis

V. Conclusion

We have presented a twofold modeling approach for both problem and solution space in product lines. The abstract domain knowledge is captured with common feature models. However, the challenge of multiple involved disciplines demands several feature model extensions for which we explore the applicability to an automation system or provide necessary adaptations. The concrete system is modeled with three different perspectives. Each perspective allows variability and evolution through delta modeling. Both spaces are equipped with analyses techniques and a consistency concept. The feature modeling includes an analysis to detect anomalies and validate the set of derivable variants. The UML models are enriched with performance annotations to deduce a separate performance model. This model can be analyzed with either a product- or family-based performance analysis to provide feedback about, e.g., bottlenecks. The complete approach is evaluated using the PPU as a long-living product line.

Acknowledgment

The PhD thesis is supervised by Ina Schaefer and partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future — Managed Software Evolution. The author would like to thank Thomas Thüm and Ina Schaefer for valuable feedback.

References

[1] S. Braun, C. Bartelt, M. Obermeier, A. Rausch, and B. Vogel-Heuser, "Requirements on evolution management of product lines in automation engineering," in *Int. Conf. Math. Modelling*, Vienna, Austria, 2012.

[2] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and S. P. A., " Feature-Oriented Domain Analysis (FODA) Feasibility Study," Carnegie Mellon University, Tech. Rep., 1990.

[3] C. Seidl, I. Schaefer, and U. Aßmann, "Integrated management of variability in space and time in software families," in *SPLC '14*.

[4] A. Hubaux, T. T. Tun, and P. Heymans, "Separation of Concerns in Feature Diagram Languages: A Systematic Survey," *ACM Computing Surveys*, vol. 45, no. 4, pp. 51:1–51:23, Aug. 2013.

[5] T. Berger, R. Rublack, D. Nair, J. M. Atlee, M. Becker, K. Czarnecki, and A. Wasowski, "A Survey of Variability Modeling in Industrial Practice," in *VaMoS*. New York, NY, USA: ACM, 2013, pp. 7:1–7:8.

[6] S. Feldmann, C. Legat, and B. Vogel-Heuser, "Engineering support in the machine and plant manufacturing domain through interdisciplinary product lines: An applicability analysis," in *INCOM'15*, 2015.

[7] I. Schaefer, "Variability modelling for model-driven development of software product lines," in *VaMoS*, 2010, pp. 85–92.

[8] W. J. Stewart, *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press, 2009.

[9] A. von Rhein, S. Apel, C. Kästner, T. Thüm, and I. Schaefer, "The PLA model: on the combination of product-line analyses," in *VaMoS*, 2013.

[10] P.-Y. Schobbens, P. Heymans, and J.-C. Trigaux, "Feature diagrams: A survey and a formal semantics," in *RE'06*.

[11] K. Pohl, H. Hönninger, R. Achatz, and M. Broy, *Model-Based Engineering of Embedded Systems*. Springer, 2012.

[12] J. Kienzle, W. Al Abed, and J. Klein, "Aspect-oriented multi-view modeling," in *AOSD'09*.

[13] O. Haugen, B. Møller-Pedersen, J. Oldevik, G. K. Olsen, and A. Svendsen, "Adding standardized variability to domain specific languages," in *SPLC*, 2008, pp. 139–148.

[14] C. Kästner and S. Apel, "Integrating compositional and annotative approaches for product line engineering," in *McGPLE*, 2008.

[15] C. Klein, C. Prehofer, and B. Rumpe, "Feature specification and refinement with state transition diagrams," in *4th IEEE WS on Feature Interactions in Telecommunications Networks*. IOS Press, 1997.

[16] M. Usman, A. Nadeem, T. hoon Kim, and E. suk Cho, "A survey of consistency checking techniques for uml models," in *ASEA 2008*.

[17] A. Egyed, "Instant consistency checking for the uml," in *ICSE '06*.

[18] R. E. Lopez-Herrejon and A. Egyed, "Detecting inconsistencies in multi-view models with variability," in *ECMFA'10*.

[19] A. Egyed, "Fixing inconsistencies in uml design models," in *ICSE'07*.

[20] R. E. Lopez-Herrejon and A. Egyed, "Towards fixing inconsistencies in models with variability," in *VaMoS'12*.

[21] C. Ghezzi and A. M. Sharifloo, "Verifying non-functional properties of software product lines: Towards an efficient approach using parametric model checking," in *SPLC*, 2011, pp. 170–174.

[22] R. Tawhid and D. C. Petriu, "Automatic derivation of a product performance model from a software product line model," in *SPLC*, 2011.

[23] K. Czarnecki and E. Ulrich, *Generative Programming: Methods, Tools, and Applications: Methods, Techniques and Applications*. Addison-WesleyLongman, 2005.

[24] I. Galvao and A. Goknil, "Survey of traceability approaches in model-driven engineering," in *EDOC*, 2007.

[25] B. Vogel-Heuser, J. Folmer, M. Kowal, I. Schaefer, S. Lity, A. Fay, W. Lamersdorf, T. Kehrer, M. Tichy, and B. Beckert, "Selected challenges of software evolution for automated production systems," in *Industrial Informatics*, 2015.

[26] L. Bettini, F. Damiani, and I. Schaefer, "Compositional type checking of delta-oriented software product lines," *Acta Informatica*, 2013.

[27] M. Kowal, C. Prehofer, I. Schaefer, and M. Tribastone, "Model-based development and performance analysis for evolving manufacturing systems," *at - Automatisierungstechnik*, vol. 62, pp. 794–802, 2014.

[28] M. Kowal, C. Legat, D. Lorefice, C. Prehofer, I. Schaefer, and B. Vogel-Heuser, "Delta Modeling for Variant-rich and Evolving Manufacturing Systems," in *MoSEMInA*, 2014.

[29] B. Vogel-Heuser, J. Mund, M. Kowal, C. Legat, J. Folmer, S. Teufl, and I. Schaefer, "Towards interdisciplinary variability modeling for automated production systems," in *Industrial Informatics*, 2015.

[30] M. Kowal, I. Schaefer, and M. Tribastone, "Family-Based Performance Analysis of Variant-Rich Software Systems," in *FASE*, 2014.

[31] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. Wiley, 2005.

[32] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake, "A Classification and Survey of Analysis Strategies for Software Product Lines," *ACM Computing Surveys*, vol. 47, no. 1, pp. 6:1–6:45, Jun. 2014.

[33] M. Kowal, M. Tschaikowski, M. Tribastone, and I. Schaefer, "Scaling Size and Parameter Spaces in Variability-aware Software Performance Models," in *ASE*, 2015.

[34] C. Legat, J. Folmer, and B. Vogel-Heuser, "Evolution in industrial plant automation: A case study," in *IECON*, 2013.