Evolution Support for Custom Variability Artifacts using Feature Models: A Study in the Cyber-Physical Production Systems Domain – Online Appendix

Kevin Feichtinger[1][0000−0003−1182−5377], Kristof Meixner[2,3][0000−0001−7286−1393], Stefan Biffl[2,4], and Rick Rabiser[1,5][0000−0003−3862−1112]

[1] LIT CPS Lab, Johannes Kepler University Linz, Austria
{kevin.feichtinger,rick.rabiser}@jku.at
[2] Institute of Information Systems Engineering, TU Vienna, Vienna
[3] Christian Doppler Laboratory SQI, TU Vienna, Vienna
kristof.meixner@tuwien.ac.at
[4] Austrian Competence Center for Digital Production, Vienna
stefan.biffl@tuwien.ac.at
[5] Christian Doppler Laboratory VaSiCS, Johannes Kepler University Linz, Austria
rick.rabiser@jku.at

1 Introduction

Cyber-Physical Production Systems (CPPSs) are highly configurable production systems with real-time control and self-adaptive behaviour [26]. CPPSs are often tailored to customer needs or environmental requirements [7], which creates a highly variable, multidisciplinary environment. A sound documentation of their variability is required to foster component reuse [11]. For this purpose, the Software Product Line (SPL) community proposed many different variability modeling approaches [29], which are used to explicitly model common and variable characteristics of a set of (software-intensive) systems [4,22]. Unfortunately, industry is mostly unaware of the plethora of existing variability modeling approaches from academia and frequently develops their own custom solutions, e.g., spreadsheet-based representations or Domain-Specific Languages (DSLs) [4].

This document is the online appendix of the paper *Evolution Support for Custom Variability Artifacts using Feature Models: A Study in the Cyber-Physical Production Systems Domain*[6]. The paper investigates the product line evolution impact on PPR–DSL artifacts [25] compared to feature models. The aim is to better understand the system evolution impact differences and work towards *enabling industrial practitioners to evolve their custom variability artifacts supported by a variability model.* Therefore, the paper uses two case study systems, i.e., the *Water filter* and the *Rocker switch* cases, from the CPPS domain [22,23].

[6] https://doi.org/xx.yyyy/zzzzzzz

2 Evolution Scenarios

Product lines manufactured on CPPSs are constantly evolving, e.g., due to new technologies or market changes [7]. We start from a reasonable subset of product variants for the case studies and change them based on the described product lines' context [23]. We specify 5 typical evolution scenarios [34] motivated by three main product line evolution triggers [21,34]: (i) user requests, (ii) technology changes/extensions, and (iii) changes in the product line environment.

S1 – Add new, optional feature: Customers frequently request new features, and their impact depends heavily on the necessary changes. In *S1*, we add a new, optional feature without additional constraints. In the *Water filter*, we add an optional water flow sensor, which informs users to refill the waste water tank. In the *Rocker Switch*, we introduce an optional light to indicate if the switch is currently active.

S2 – Add new, dependent feature: In industry, future user requests are often anticipated. Therefore, in *S2*, we restructure the product line and add new features along with new constraints. In the *Water filter*, we add an abstract `Waste Water Tank` feature and a new waste water tank (with a different size) together with constraints. In the *Rocker Switch*, we extend the product line with additional variants of the key components `Rocker`, `Off` and `Pole` with constraints to model relationships between them.

S3 – Add multiple features and constraints within product line scope: A product line is often extended by a set of features and constraints to meet new technology or user requirements. In *S3*, we add multiple features and constraints. In the *Water filter*, we extend the product line by a new filter type (`Distillation`) along with necessary components (sub-features) and constraints. In the *Rocker Switch*, we extend the rocker switches to distinguish between serial and crossover switches, together with necessary constraints.

S4 – Add multiple new features and constraints outside the scope: Sometimes product lines undergo fundamental changes, where several features are added, removed, or replaced, which also changes/extends the scope of the product line. In *S4*, we extend the product line to support additional technologies. In the *Water filter*, we extend the product line by (five different) pumps, an additional type of heater used during distillation, a new mount type, electricity support, and country-specific implementations. We also add necessary constraints (e.g., pumps require electricity or electricity prohibits using the iron frame as a mount). In the *Rocker Switch*, we extend the rocker switches with a home automation interface supporting two protocols (Zigbee [15] and Zwave [27]) along with necessary (transformer) features and constraints.

S5 – Add constraints and remove features: Industrial products have to adapt to new technology and regulations regularly. The necessary specializations can be implemented by adding/removing features and/or adding constraints. In *S5*, we remove features and add constraints. In the *Water filter*, we add new constraints to the country codes for electricity to prohibit certain pump types in some countries. We also constrain mounts and remove unused pump sizes and

Table 1. Metrics we collect for PPR–DSL artifacts and feature models.

Type	Metric	Description
PPR–DSL artifact	#P	Number of products
	$\#P_{comp}$	Number of products specifying the attribute component
	$\#P_{abst}$	Number of abstract products
	$\#P_{impl}$	Number of products, which implement at least one other product
	$\#P_{child}$	Number of products, which specify children
	$\#C_{req}$	Number of products, which require at least one other product
	$\#C_{excl}$	Number of products, which exclude at least one other product
	$\#C_{complex}$	Number of constraints including more than two products
Feature Model	#F	Number of features
	$\#F_{abst}$	Number of abstract features
	$\#F_{man}$	Number of mandatory features
	$\#C_{xorgroup}$	Number of Xor group constraints
	$\#C_{total}$	Number of constraints
	$\#C_{complex}$	Number of constraints, which involve more than two features
	Tree height	Feature model tree height
	#Configs	Number of valid configurations found using sampling

country codes. In the *Rocker Switch*, we add new constraints between the rocker and changeover features and remove the light feature and its constraints.

3 Metrics to measure Evolution impact

Building on earlier work [13], we want to measure the evolution impact on the PPR–DSL artifacts and feature models using using four general characteristics of variability modeling approaches [13], namely (a) *unit of variability*, (b) *composites*, (c) *hierarchy*, and (d) *dependencies*. We use existing metrics for feature models [10] to assess these characteristics, their size and complexity. Building on these metrics, we identified similar metrics to assess these aspects for PPR–DSL artifacts. We assess the evolution impact on the PPR–DSL artifacts and feature models using metric changes after each evolution step. Table 1 summarizes the investigated metrics for PPR–DSL artifacts (upper part) and feature models (lower part).

For the *unit of variability*, we count in the PPR–DSL the number of products (#P) and the number of products specifying the attribute component ($\#P_{comp}$). We further count how many of the specified products are abstract ($\#P_{abstract}$). For feature models, we count the number of features (#F) and the number of abstract ($\#F_{abstract}$) and mandatory ($\#F_{man}$) features. The PPR–DSL and feature models have different types to capture *composite*s. We count them for the PPR–DSL (products specifying the attribute children – $\#P_{children}$) and feature models (number of XOR groups – $\#C_{xorgroup}$). *Hierarchy* is differently defined in PPR–DSL and feature models. While hierarchy in the PPR–DSL is specified using inheritance ($\#P_{impl}$), in feature models, it is defined via the feature tree

(quantifiable via the Tree height). Both artifact types allow to define *dependencies* between their respective *unit of variability*. In the PPR–DSL we count the number of products which specify requires ($\#C_{req}$) and excludes ($\#C_{excl}$) constraints. We also count the number of constraints ($\#C_{complex}$) specified in the PPR–DSL artifacts, which include more than two products. For feature models, we count the total number of constraints ($\#C_{total}$) and again complex constraints ($\#C_{complex}$), as in the case studies most constraints are defined between two features but complex constraints are particularly interesting. We also count the number of valid configurations in a feature model (#Configs) to assess the complexity of the feature model, i.e., underlying product line, and configuration relevance [32]. This metric is particularly important in CPPS engineering, as the products that can be manufactured contribute to the overall cost of the CPPS.

4 Analysis Results

4.1 Evolution Impact

In both case studies, in the scenarios *S1–S4* the PPR–DSL artifacts and feature models grow in size and complexity. Only in *S5* products in the PPR–DSL and features in the feature model are removed and constraints are added to both artifacts. As a result, the number of valid configurations in the feature model (#Configs) shrunk significantly.

The *unit of variability* for both artifact types (#P, $\#P_{comp}$ and $\#P_{abstract}$ for PPR-DSL artifacts and #F, $\#F_{abstract}$ and $\#F_{man}$ for feature models) increased for the scenarios *S1–S4*. Only for *S5* they decreased. This development is a result of the scenarios, as in *S1–S4* features are added to the product lines. However, despite only adding one feature (#F) to the *Water filter* in *S1* (resulting in a new component product – $\#P_{comp}$), the number of overall products increased by 8. This is a result of how the PPR–DSL enables engineers to design products. Engineers must specifically define which products should be produced with the (designed) production system. In *S1*, we specified that the originally valid products are all extended by a new component, resulting in 8 new products. In a similar way, we extended the *Rocker Switch*, where the component products and features evolve at a similar rate. In *S2*, we simulated restructuring the product lines by adding newly features and constraints. Such restructuring can influence the relationships between the products (an increase of abstract products $\#P_{abstract}$ in both case studies) and the set of designed products. For instance, in the *Water filter* it did not influence the set of designed products, as we only performed the restructuring but did not create new designed products (#P remains equal). In contrast, for the *Rocker Switch* we introduced additional constraints also addressing existing components (as there were no constraints specified in *S0*), resulting in the decrease of designed products (#P). Nevertheless, in the PPR–DSL artifacts and the feature models the number of component products ($\#P_{comp}$) and features (#F) again increased equally. A trend that continues through all scenarios (including *S5* where they similarly decreased).

In both case studies, *composites* (#P$_{child}$) were added during the evolution scenarios. For the *Water filter* during *S3*, when a new filter type (`Distillation`) was added, and in the *Rocker Switch* during *S4*, when the house automation extended the scope of the product line. In the respective feature models (#C$_{xorgroup}$), the XOR groups were introduced earlier. This is because, although both metrics cover the same concept, the attribute `children` captures groups of products belonging together, whereas in the XOR groups additionally constrain the feature model. As a result, feature groups are more frequently used in feature modeling than the attribute in the PPR–DSL. In *S5*, *composites* in both artifact types do not change much.

In both artifact types, *hierarchy* (#P$_{impl}$ in the PPR–DSL and Tree height in feature models) is captured differently. Whereas in the PPR–DSL *hierarchy* is modeled via inheritance, in feature modeling, it is a key concept using the visualization in a feature tree. This difference also showed in the evolution of the artifacts. The PPR–DSL artifacts evolved similarly to the *unit of variability*, because the implements is a product attribute. During the evolution of the feature models, the Tree height hardly changed. The newly added features and constraints only increased the tree height in *S4* (*Water filter* and *Rocker Switch* respectively), confirming findings by Lotufo et al. [20], that feature models in an industrial context often evolve into a broader tree, instead of a deeper tree.

Dependencies (#C$_{req}$, #C$_{excl}$ and #C$_{complex}$ in the PPR–DSL and #C$_{total}$ and #C$_{complex}$ in feature modeling) are a key part of both artifact types. In both case studies, the dependencies between the respective *unit of variability* increase over time. Overall, the PPR–DSL artifacts appear more complex than their respective feature model counterparts. This is because in the PPR–DSL artifact the requries and excludes attributes capture direct relations between products, which in feature modeling are captured in either *composites* (#C$_{xorgroup}$) or the feature tree (Tree height). One has to add several excludes relationships between the effected products to capture a XOR group from the feature model in the PPR–DSL. Complex constraints (involving more than 3 products or features – #C$_{complex}$ in both artifacts) evolve similarly in both artifacts.

4.2 Differences to automatically derived artifacts

The automatically derived PPR–DSL artifacts are more different to the manually created PPR–DSL artifacts than the automatically obtained feature models have to their manually created counterparts.

In the automatically derived PPR–DSL artifacts from the manually created feature models, hardly any difference can be found in the scenarios *S0–S1*. From scenario *S3* onward many differences across the characteristics, *unit of variability*, *hierarchy* and *dependencies* can be found. This can mainly be explained by the difference of how variability is modeled in the PPR–DSL and feature models, as explained when discussing the evolution impact on the *unit of variability*. In the scenarios (*S0–S1*), we listed all possible products in the PPR–DSL. As a result there is no difference between the two representations and the relationship $\#P + \#\mathrm{P}_{comp} = \#F + \#\mathrm{Configs}$ holds. From scenario *S3* onward, we

did not cover all possible products, which results in the expected increase of possible products (#P) in the automatically derived PPR–DSL artifacts. This increase also reflects in *hierarchy* ($\#P_{impl}$) and *dependencies* ($\#C_{req}$, $\#C_{excl}$ and $\#C_{complex}$) of the PPR–DSL artifacts.

Even though the structure of the automatically created artifact is the same as the one from the manually created one, we lose the *composites* ($\#P_{child}$) in the transformed PPR–DSL artifacts. Manual inspection of the derived PPR–DSL artifacts showed that the children attribute of the products was not set. We conclude that this is a result of the provided transformations by TRAVART, rather than from the evolution steps.

In the automatically derived feature models from the manually created PPR–DSL artifacts, all areas *unit of variability*, *composites*, *hierarchy* and *dependencies* evolve similarly. The only difference in the *Water filter* is the total number of constraints, which results from the way how *composites* ($\#C_{xorgroup}$) are defined in the PPR–DSL. For each XOR group, excludes relationships have to be defined using the excludes attribute. These relationships are transformed into excludes constraints, which are redundant with the XOR group. Not requiring to model these excludes constraints explicitly, but modeling them via hierarchy and feature group constraints is a clear benefit of the feature model compared to the PPR-DSL. In the *Rocker Switch* case, the only difference is the total set of valid configurations (#Configs). Again, a manual inspection of the manually created feature models and automatically derived feature models showed no difference between the models. We concluded that the difference originates from the sampler stability [33,35] of the provided sampler of TRAVART.

5 Related Work

Feature Extraction: Retrieving variability models from existing systems can be a tedious task. Several feature extraction approaches have been proposed to automatically reveal and analyze variability in various artifacts [3,8,18,24,36]. In a mapping study, [2] discuss further works on re-engineering of systems into product lines with a focus on transforming various artifacts into reusable components. Most approaches typically focus on specific types of artifacts (e.g., source code) and can only generate one kind of variability model (e.g., a feature model). This specialization makes it hard to apply these approaches in industry across a heterogeneous set of variability artifacts (e.g., spreadsheets or DSLs). Also, extraction approaches only focus on extracting the information from one version (one snapshot) of the variability artifact at a single point in time, neglecting the evolution of the artifacts. Our work aims to address the need for an evolution process for custom variability artifacts in industry, using the advantages of a variability model without constantly extracting features.

Variability Modeling in Industry: Several works investigate the use of variability models in industry [4] or propose the use of a variability model in industry [9,16,30,31]. Most works focus on a variability model as a first-class citizen of the development process. Some have already tried to use transformations

between custom representations [1,12] including Feichtinger et al. [14], whose transformation approach TRAVART we use in our work. However, in our work, we aim to still use custom representations from industry as the main artifact and support their evolution using variability models.

Product Line Evolution: Product line evolution research mostly focuses on one type of variability modeling approach [21,28,19] or on a certain aspect of evolution (e.g., evolution planning) [17]. The evolution impact on feature models has been investigated by Bezerra et al. [6]. They performed an exploratory study on the evolution impact on feature models using a maintainability catalog [5]. While these results have influenced our work, this paper investigates the product line evolution impact on custom variability artifacts in industry.

6 Conclusion

In this appendix, we described the evolution scenarios we performed on two case study systems from the Cyber-Physical Production System (CPPS) domain to investigate the evolution impact on Product-Process-Resource DSL (PPR–DSL) artifacts feature models. We also described the results of our evolution analysis in detail, which are the basis for our key takeaways described in the paper[6].

Acknowledgements

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) via “Austrian Competence Center for Digital Production” (CDP) under contract nr. 881843.

References

1. Andersen, N., Czarnecki, K., She, S., Wasowski, A.: Efficient synthesis of feature models. In: Proc. 16th International Software Product Line Conference - Volume 1. pp. 106–115. ACM (2012)
2. Assunção, W.K.G., Lopez-Herrejon, R.E., Linsbauer, L., Vergilio, S.R., Egyed, A.: Reengineering legacy applications into software product lines: a systematic mapping. Empirical Software Engineering **22**(6), 2972–3016 (Dec 2017)
3. Bakar, N.H., Kasirun, Z.M., Salleh, N.: Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. Journal of Systems and Software **106**, 132–149 (2015)
4. Berger, T., Rublack, R., Nair, D., Atlee, J.M., Becker, M., Czarnecki, K., Wasowski, A.: A survey of variability modeling in industrial practice. In: Proc. 7th International Workshop on Variability Modelling of Software-intensive Systems. pp. 7–14. ACM (2013)

5. Bezerra, C.I.M., Andrade, R.M.C., Monteiro, J.M.S.: Measures for quality evaluation of feature models. In: Schaefer, I., Stamelos, I. (eds.) Software Reuse for Dynamic Systems in the Cloud and Beyond. pp. 282–297. Springer International Publishing, Cham (2014)
6. Bezerra, C.I.M., Monteiro, J.M., Andrade, R.M.C., Rocha, L.S.: Analyzing the feature models maintainability over their evolution process: An exploratory study. In: Proc. Tenth International Workshop on Variability Modelling of Software-Intensive Systems. p. 17–24. VaMoS '16, ACM (2016)
7. Biffl, S., Gerhard, D., Lüder, A.: Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems, pp. 1–24. Springer International Publishing, Cham (2017)
8. Cruz, D., Figueiredo, E., Martinez, J.: A Literature Review and Comparison of Three Feature Location Techniques Using ArgoUML-SPL. In: Proc. 13th International Workshop on Variability Modelling of Software-Intensive Systems. pp. 16:1–16:10. VAMOS '19, ACM, New York, NY, USA (2019)
9. Dhungana, D., Grünbacher, P., Rabiser, R.: The DOPLER Meta-Tool for Decision-Oriented Variability Modeling: A Multiple Case Study. Automated Software Engineering **18**(1), 77–114 (2011)
10. El-Sharkawy, S., Yamagishi-Eichler, N., Schmid, K.: Metrics for analyzing variability and its implementation in software product lines: A systematic literature review. Information and Software Technology **106**, 1–30 (2019)
11. Fadhlillah, H.S., Feichtinger, K., Sonnleithner, L., Rabiser, R., Zoitl, A.: Towards heterogeneous multi-dimensional variability modeling in cyber-physical production systems. In: Proc. 25th ACM International Systems and Software Product Line Conference. p. 123–129. SPLC '21, ACM (2021)
12. Feichtinger, K., Meixner, K., Rabiser, R., Biffl, S.: Variability transformation from industrial engineering artifacts: An example in the cyber-physical production systems domain. In: Proc. 3rd International Workshop on Variability and Evolution of Software-Intensive Systems (VariVolution), SPLC 2020. ACM (2020)
13. Feichtinger, K., Rabiser, R.: Variability model transformations: Towards unifying variability modeling. In: Proc. 46th Euromicro Conference on Software Engineering and Advanced Applications. IEEE, Portoroz, Slovenia (2020)
14. Feichtinger, K., Stöbich, J., Romano, D., Rabiser, R.: Travart: An approach for transforming variability models. In: 15th International Working Conference on Variability Modelling of Software-Intensive Systems. ACM (2021)
15. Gislason, D.: Zigbee Wireless Networking. Newnes, Burlington, USA (2008)
16. Hinterreiter, D., Prähofer, H., Linsbauer, L., Grünbacher, P., Reisinger, F., Egyed, A.: Feature-oriented evolution of automation software systems in industrial software ecosystems. In: 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA). pp. 107–114 (2018)
17. Hoff, A., Nieke, M., Seidl, C., Sæther, E.H., Motzfeldt, I.S., Din, C.C., Yu, I.C., Schaefer, I.: Consistency-preserving evolution planning on feature models. In: Proc. 24th ACM Conference on Systems and Software Product Line. SPLC, ACM (2020)
18. Linsbauer, L., Lopez-Herrejon, R.E., Egyed, A.: Feature Model Synthesis with Genetic Programming. In: Le Goues, C., Yoo, S. (eds.) Search-Based Software Engineering. pp. 153–167. Springer International Publishing, Cham (2014)
19. Lity, S., Nahrendorf, S., Thüm, T., Seidl, C., Schaefer, I.: 175% modeling for product-line evolution of domain artifacts. In: Proc. 12th International Workshop on Variability Modelling of Software-Intensive Systems. VAMOS 2018, ACM (2018)

20. Lotufo, R., She, S., Berger, T., Czarnecki, K., Wasowski, A.: Evolution of the linux kernel variability model. In: Bosch, J., Lee, J. (eds.) Software Product Lines: Going Beyond. pp. 136–150. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
21. Marques, M., Simmonds, J., Rossel, P.O., Bastarrica, M.C.: Software product line evolution: A systematic literature review. Journal of Inf. Softw. Technol. **105** (2019)
22. Martinez, J., Assunção, W.K., Ziadi, T.: ESPLA: A catalog of Extractive SPL Adoption case studies. In: Proc. 21st International Systems and Software Product Line Conference. pp. 38–41. ACM (2017)
23. Meixner, K., Feichtinger, K., Rabiser, R., Biffl, S.: A reusable set of real-world product line case studies for comparing variability models in research and practice. In: Proc. 25th ACM International Systems and Software Product Line Conference. p. 105–112. SPLC '21, ACM (2021)
24. Meixner, K., Rabiser, R., Biffl, S.: Feature identification for engineering model variants in cyber-physical production systems engineering. In: Proc. 14th International Working Conference on Variability Modelling of Software-Intensive Systems. pp. 1–5. ACM (2020)
25. Meixner, K., Rinker, F., Marcher, H., Decker, J., Biffl, S.: A Domain-Specific Language for Product-Process-Resource Modeling. In: IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA). IEEE (2021)
26. Monostori, L.: Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. Procedia CIRP **17**, 9–13 (2014)
27. Paetz, C.: Z-Wave Essentials. CreateSpace Independent Publishing Platform, North Charleston, SC, USA (2018)
28. Pleuss, A., Botterweck, G., Dhungana, D., Polzer, A., Kowalewski, S.: Model-driven support for product line evolution on feature level. Journal of Systems and Software **85**(10), 2261–2274 (2012), automated Software Evolution
29. Raatikainen, M., Tiihonen, J., Männistö, T.: Software product lines and variability modeling: A tertiary study. Journal of Systems and Software **149**, 485–510 (2019)
30. Rabiser, D., Prähofer, H., Grünbacher, P., Petruzelka, M., Eder, K., Angerer, F., Kromoser, M., Grimmer, A.: Multi-purpose, multi-level feature modeling of large-scale industrial software systems. Software & Systems Modeling **17**(3) (Jul 2018)
31. She, S., Lotufo, R., Berger, T., Wasowski, A., Czarnecki, K.: The Variability Model of The Linux Kernel. In: Proc. 5th International Workshop on Variability Modelling of Software-intensive Systems. pp. 45–51. ACM (2010)
32. Sundermann, C., Nieke, M., Bittner, P.M., Heß, T., Thüm, T., Schaefer, I.: Applications of #sat solvers on feature models. In: 15th International Working Conference on Variability Modelling of Software-Intensive Systems. ACM (2021)
33. Sundermann, C., Thüm, T., Schaefer, I.: Evaluating sat solvers on industrial feature models. In: Proc. 14th International Working Conference on Variability Modelling of Software-Intensive Systems. VAMOS '20, ACM (2020)
34. Svahnberg, M., Bosch, J.: Evolution in software product lines: Two cases. Journal of Software Maintenance: Research and Practice **11**(6), 391–422 (1999)
35. Thüm, T., Apel, S., Kästner, C., Schaefer, I., Saake, G.: A classification and survey of analysis strategies for software product lines. ACM Comput. Surv. (Jun 2014)
36. Valente, M.T., Borges, V., Passos, L.: A semi-automatic approach for extracting software product lines. IEEE Trans. Softw. Eng. **38**(4), 737–754 (2012)