DOI: http://dx.doi.org/10.1145/2600821.2600833

Realizing Software Process Lines: Insights and Experiences

Marco Kuhrmann, Daniel Méndez Fernández
Technische Universität München
Faculty of Informatics
Munich, Germany
{kuhrmann, mendezfe}@in.tum.de

Thomas Ternité
Technische Universität Clausthal
Department of Informatics
Clausthal-Zellerfeld, Germany
thomas.ternite@tu-clausthal.de

ABSTRACT

Software process lines provide a systematic approach to construct and manage software processes. A process line defines a reference process containing general process assets, whereas a well-defined customization approach allows process engineers to create new process variants by, e.g., extending or altering process assets. Variability operations are a powerful instrument to realize a process line. However, little is known about which variability operations are suitable in practice. In this paper, we present a study on the feasibility of variability operations to support process lines in the context of the German V-Modell XT. We analyze which variability operations were defined and used to which extent, and we provide a catalog of variability operations as an improvement proposal for other process models. Our findings show 69 variability operations defined across several metamodel versions of which 25 remain unused. Furthermore, we also find that variability operations can help process engineers to compensate process metamodel evolution.

Categories and Subject Descriptors

D.2.9 [**Software Engineering Management**]: Software process models

General Terms

Management, Experimentation

Keywords

software process lines, software process metamodel, metamodel evolution, variability operations

1. INTRODUCTION

The V-Modell XT is the standard software process for IT development projects in Germany's government agencies. It has a long history beginning with the first release in 1992. In 2002, a SPI-project was initiated to refresh and modernize the entire process at its complementing ecosystem. The first release of the new V-Modell XT was published in 2005 as part of a comprehensive *software process framework*, which became subject to maintenance and continuous improvement [8]. Over the years, a number of V-Modell XT variants were built using a "copy & change" procedure in which company-specific process assets were realized using a local copy of the reference process. As the reference process evolved, this approach caused serious problems, e.g., how to integrate updated contents, how to figure out what particular customizations were affected by newer reference contents, or how to migrate existing content to a new process metamodel. Much effort has been spent to analyze the evolved variants (see, e.g., [17, 16, 15]). However, only the changes could be analyzed and documented. Efficiently integrating evolved model contents with customized ones remained a critical and unresolved task. In response, we developed a new approach to maintain the reference process *and* its variants to allow for evolution and (automatic) updates. The new approach implements concepts proposed by *software process lines* [19, 14]. The basic idea is to apply concepts of product lines [20] to the domain of software process models (see Sect. 2). Special attention was paid to the customization approach that supports process engineers, inter alia, in using *variability operations* as a declarative instrument to systematically adapt the reference model while ensuring consistency and compliance with the reference model.

Problem Statement. While defining the variability operations, we faced the problem to name a set of meaningful and actionable variability operations. Available approaches are either conceptual [14] or focus on general concepts [18] that need further refinement. However, still missing is a proven basic set of variability operations to support constructing a process variant from a software process line. That is, process engineers need to develop their own portfolio of variability operations which negatively impacts the process line and its variants, e.g., due to incompatible sets of variability operations, or potential loss of compliance. Furthermore, we still lack in long-term studies analyzing the feasibility of process line approaches from the perspective of the process engineer.

Research Objectives. To direct further research on the application of software process line approaches in practice, we aim at analyzing the feasibility of variability operations and at their improvement to support the development and maintenance of comprehensive software process lines.

ICSSP '14, May 26–28, 2014, Nanjing, China

DOI: http://dx.doi.org/10.1145/2600821.2600833

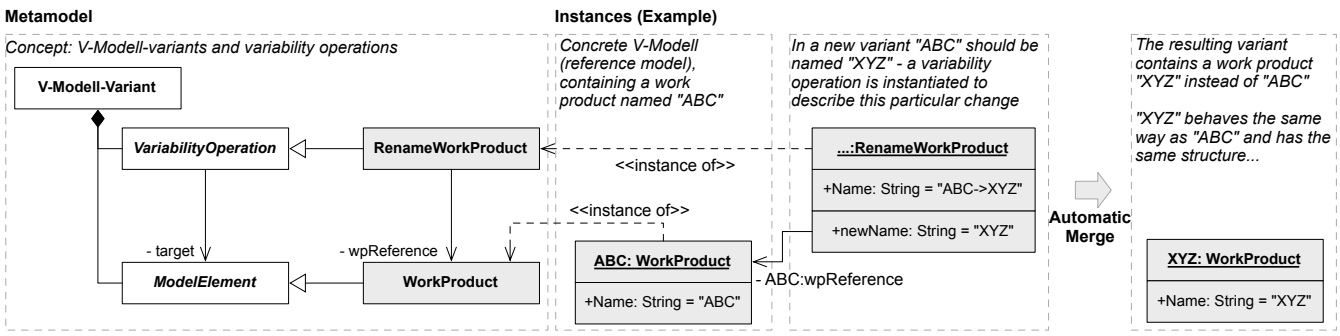

Figure 1: Variability operations (concept and example).

Contribution. In this paper, we contribute a study on the application of variability operations to realize comprehensive software process lines. Our investigation is based on a snapshot from 2013 of the V-Modell XT family [8] in which we investigated the V-Modell XT reference model and 5 of its variants using the process line features. We contribute a catalog of variability operations as implemented in the German V-Modell XT. Furthermore, beyond the catalog, we analyze the feasibility of this instrument. We investigate which variability operations were defined and to which extent those were used to identify their suitability. Thus, we close a gap in literature as—to the best of our knowledge—yet missing is an understanding about which variability operations are practically relevant at all. Finally, we also analyze settings in which variability operations were not used and provide a rationale.

Outline. The remainder of the paper is organized as follows: Section 2 describes the setting of our study and introduces the basic concepts and terminology used. Section 3 summarizes the related work. In Sect. 4, we present the study design including our research questions, the case selection, and the data collection and analysis procedures. In Sect. 5, we finally present our findings, before giving a conclusion in Sect. 6.

2. CONTEXT & USED TERMINOLOGY

We analyzed the V-Modell XT process family [8]. Since the V-Modell XT is a comprehensive process framework with (over the time) built-in process line features, we first need to introduce the basic concepts and underlying terminology.

Model, Metamodel and Modules. The V-Modell XT is a modular, metamodel-based framework to define software processes. The metamodel [23] defines the *process language* to create single processes or whole variant trees. According to [7], a *V-Modell-variant* logically consist of two models: (1) a structure model contains all (atomic) model elements, and (2) an overlaying dependency model connects all model elements. Hence, if dependencies are contained in *process modules*, the configuration of such modules directly influences the dependency model, which allows for a comprehensive tailoring. During customization, all elements of these packages can be extended, altered, and so on.

Process Variants. The metamodel is designed to support hierarchically organized process variants—even the reference model itself is regarded as a variant. Creating a new variant requires to refer to a *reference model* on which the variant is based. A variant can be regarded as an *extension* applied to a reference model. Since all V-Modell-variants are (or should be) based on the same metamodel, each variant may contain a complete process. As all model elements from the reference model are accessible from a variant, a variant can refer to and thus integrate and modify any reference model element. A merge tool creates an integrated process from the variants. New process assets introduced by the variant will then be integrated with the reference model, exclusions will be deleted, and variability operations will be executed.

Variability Operations. Variability operations allow a process variant to modify contents of the reference model [21]. A variability operation is a model element that declares a change, e.g., renaming of elements, adding description text, or restructuring dependencies (see Fig. 1). During the merge procedure of a reference model and an extension model, the descriptive information provided by variability operations is operationalized by the merge tool. For example, the declaration of renaming a work product has to be evaluated and executed during that merge. If an extension model contains an instance of "RenameWorkProduct", this has to be interpreted as a rename operation on the referred instance of "WorkProduct" during the merge.

The V-Modell XT Process Line. The V-Modell XT family consists of a reference process and a number of derived variants (cf. [8] and complementing research). The snapshot, which is the case for our study (Fig. 2), shows the reference model and two kinds of variants. On the one hand, we find variants that were created as a direct modification of a local copy ("old" scheme). On the other hand, we find 5 variants that use the built-in process line features of the framework. Variants using the process line features can reuse content from the reference model and support automatic updates. If a new version of the reference process is released, in the simple case, the merge tool automatically updates a variant, e.g., by computing the variability operations again. Further information regarding the selected sample is given in Sect. 5.1.

3. RELATED WORK

In [19], Rombach votes for organizing comprehensive software processes similar to product lines. To this end, a software process line consists of a stable core (commonalities) and variable parts (variabilities) [14, 20, 4, 3]. This ap-

DOI: http://dx.doi.org/10.1145/2600821.2600833

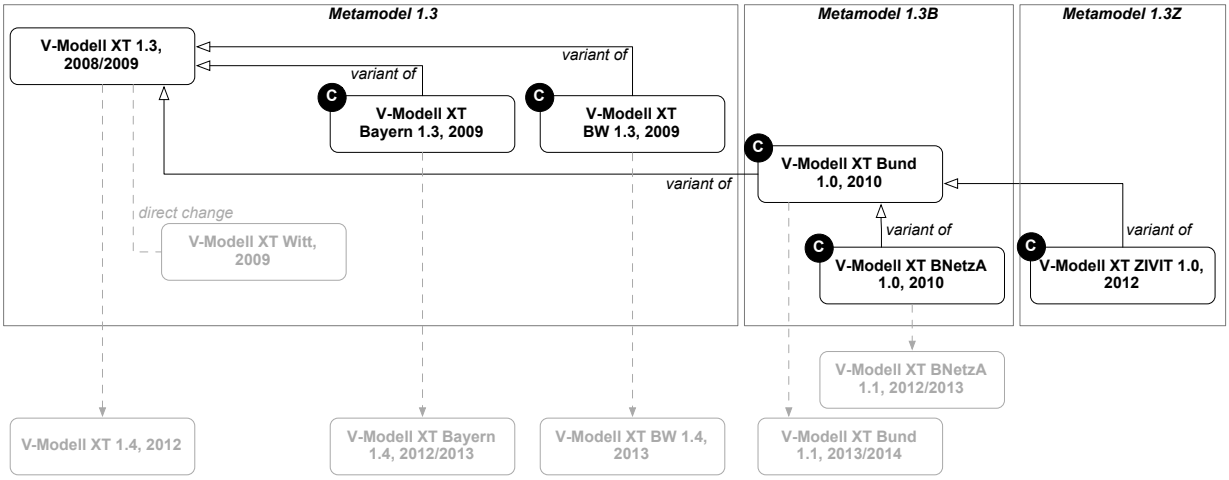

Figure 2: Snapshot of the V-Modell XT software process line (variants marked with "C" are the cases).

proach proposes advantages regarding the organization and management of process knowledge and the systematic creation of reusable (domain-specific) process assets to ease the development of process variants. A software process line is a framework for a directed and proactive process construction and management.

Since comprehensive software processes may consist of several thousands of different process assets, management is critical and, thus, a sophisticated technical basis, e.g., a process framework, is necessary. A process metamodel is required to provide process engineers with tools to create, edit, and manage the structure and the content of a process, and strong tool support is required to support management, development, and deployment tasks. Popular metamodels are, for example, the *Software & Systems Process Engineering Metamodel* (SPEM; [18]), the *Software Engineering Metamodel for Development Methodologies* (SEMDM; [5]), or the *V-Modell XT Metamodel* [23]. SPEM and the V-Modell XT explicitly define variability operations. Process assets that are built on these metamodels can extend or modify other process assets, and they can be configured from certain (process) modules. Other than SPEM, the V-Modell XT explicitly defines a process variant concept [7, 22] and provides an extensive set of *typed* variability operations for fine-grained model manipulations. Table 1 summarizes the practically applied approaches to implement *constructive* variability for process models.

Table 1: Approaches to implement variability.

Approach	SPEM	V-Modell XT
Variability ops. (general)	[18]	
Variability ops. (typed)		[7, 22]
Tailoring		[9]
Modularity	[18]	[7]

The group around Münch and Armbrust follows a distinctive approach to create process model variants. Instead of constructively defining process variants, they focus on the evolution of a process model [22]. The evolutionary approach comprises: (1) scoping processes to identify the locations where variability is needed [1, 2]; (2) providing rationale during process evolution [17, 16, 15]; and (3) analyzing differences of evolved model variants. The first aspect aims at determining the properties an actual process has and at identifying commonalities to infer needed variability. Corresponding approaches lay their focus on the analysis of an existing model and the possibilities to create a pattern, which can be used to create variants. The latter two aspects are both focused on an *a posteriori* observation of the evolved subject. They do not explicitly support the variability of a process model, but deal with evolving models in general.

Although there exists a number of approaches directly or indirectly supporting the management of process variants, a deeper understanding of the variability operations is yet missing. With the study at hands, we thus close this gap in literature.

4. STUDY DESIGN

In this section, we present the study design. After defining the goal and the research questions, we describe how we selected the case. Finally, we describe how we collected and analyzed the data, before concluding with a discussion on the validity procedures.

4.1 Research Questions

Our overall objective is the investigation of the feasibility and the practical application of variability operations to support the (long-term) development and the maintenance of software process lines. For this, we investigate which variability operations are implemented in general and to which extend these variability operations are used.

As a second step, we investigate settings in which variability operations were not used and why. To this end, we define the following research questions:

RQ 1: *Which variability operations are defined to realize the process line?* Since most related work discusses, if at all, variability operations in a generic manner (e.g., SPEM defining *extends* or *replaces*), our first research question aims to identify a set of variability operations to create a catalog.

DOI: http://dx.doi.org/10.1145/2600821.2600833

RQ 2: *Which variability operations are practically used to which extend?* The second research question aims to investigate the feasibility of the found variability operations. We analyzed to which extent the found variability operations (of a certain type) were actually used during the development of particular process variants.

RQ 3: *In which settings are variability operations not used and why?* We aim to investigate settings that are potentially inappropriate for variability operations. Thus, we analyzed settings in which variability operations were not used, and we investigated the respective settings, analyzed the instruments used instead of variability operations, and provide a rationale.

4.2 Case Selection

We opted for the V-Modell XT to collect and analyze variability operations. As we are interested in the variability operations and their use, we only consider such V-Modell XT variants that use the process line features provided by the framework. Variants that are built by copying and directly modifying the reference process are out of scope.

4.3 Data Collection Procedure

The data collection was done two-fold: To answer *RQ1* and *RQ2*, we used a tool to export lists of the variability operations defined and used. All information was collected by parsing the models' XML files, and storing the data in a spreadsheet. Therefore, we first analyzed the respective metamodel on which a process variant is based, and gathered all defined *variability operation types*. In the second step, we exported the *variability operation exemplars* as defined in the process models (in this step, we also analyzed which version of the underlying metamodel defines an instantiated operation type to track the metamodel evolution). We repeated the export process for every considered process variant to (1) create a consolidated list of operation types across all versions of the metamodel, (2) to create process-variant-specific lists of variability operation exemplars, and (3) an aggregated list of all variability operations, their type, number of exemplars, and so forth.

In order to answer *RQ3*, we had to (manually) inspect the considered process variants. We compared the merged (Sect. 2) process definition with its sources (reference and extension model) for added, altered and/or removed process assets that are not defined using variability operations. The outcome of this investigation was also stored in a spreadsheet.

4.4 Analysis Procedure

Due to the low number of cases, we present the results as data tables and simple charts, and qualitatively analyze and interpret the results.

5. STUDY RESULTS

We first give a description of the case and the subjects, before summarizing the results.

5.1 Case Description

As case, we opted for the V-Modell XT and the set of 5 variants that use the process line features of the V-Modell XT framework. Fig. 2 shows a snapshot; the highlighted variants are subject to the investigation—the other variants do not use the process line features and, thus, are out of scope. Although the V-Modell XT 1.4 was released in 2012, no variants using this version as reference model were available when we conducted the study. Therefore, our study is based on those variants using the V-Modell XT 1.3 and we refer to this version as the *reference model* on which, finally, all variants[1] are built. Each variant, except for the V-Modell XT 1.3, points to its parent (Fig. 2 also shows that the process line builds a "family tree" in which a derived variant can be a reference model for further variants). Variability operations, which are instantiated in a variant refer to process assets that are defined in the respective reference model. A tool computes all variability operation types and exemplars, and creates a dataset, which contains the information listed in Table 2. The datasets are created as CSV-files (one per variant).

Table 2: Generated data structure for the analysis.

Field	Description
Type	Variability operation type, e.g., *RenameRole*, *ReplaceSectionText*
Name	Name of the operation exemplar
Basic operations	Basic model transformation operations used to implement the variability operation, e.g., *RenameElement*, *AddText*, *ChangeReferences*
Affected elements	List of model elements that are affected by the operation exemplar
Compliance criticality	Rating whether an operation potentially violates compliance (conformance) requirements or constraints

5.2 RQ 1: Variability Operation Types

The V-Modell XT metamodel defines the set of variability operation types. Since the variants under consideration use different versions/variants of the metamodel, we first analyze (1) which variability operation types are defined and (2) which metamodel defines a particular operation type.

Table 5 shows the complete list (the catalog) of variability operations, classified according to their *operation groups*. An operation group comprises all operation types that are logically related (e.g., changes on work products). The column "MM" of Table 5 indicates which metamodel version (Fig. 2) defines the operation ("1.3" refers to the original metamodel on which the reference model is based; "1.3 B" refers to the metamodel on which the V-Modell XT Bund is based; the metamodel "1.3 Z" does not contribute new variability operations).

In total, the V-Modell XT metamodel provides process engineers with a set of up to 69 variability operation types. Figure 3 summarizes Table 5, shows the operation groups, and the number of operation types per group and per meta-

[1]**Note**: Except for the variants *V-Modell XT 1.3* and *V-Modell XT Bund 1.0*, that are publicly available, for confidentiality reasons, we are not allowed to relate the findings to a variant from Fig. 2; we provide the data, but anonymized. A collection of publicly available material is provided by the Weit e.V.: `www.weit-verein.de/varianten.html` (in German).

DOI: http://dx.doi.org/10.1145/2600821.2600833

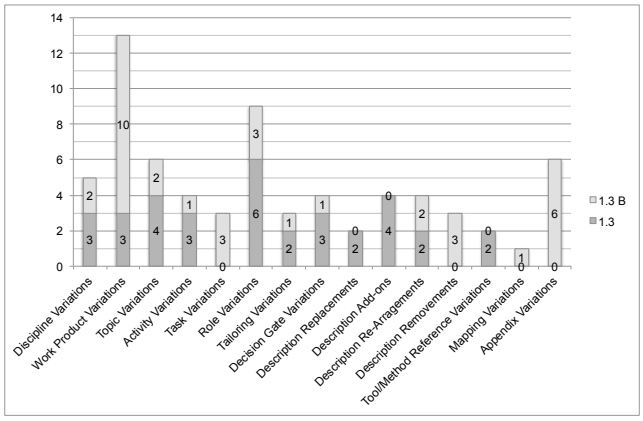

Figure 3: Operation types per metamodel version.

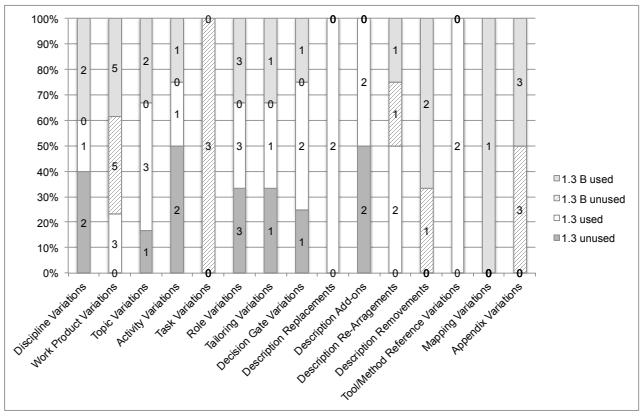

Figure 4: Used op. types per metamodel version.

model version. Furthermore, Fig. 3 also reflects the evolution of the metamodel—35 new operation types were introduced in the metamodel "1.3 B" (two years after the publication of the reference model 1.3, which defines 34 variability operation types).

Interpretation. We found variability operation types defined in two metamodel versions. Moreover, the number of operation types doubled. The explanation can be found in the metamodel's evolution. The metamodel "1.3 B" got a substantial improvement, which was based on customer requirements, whereas the initial set of operation types was derived from known improvements at this time and compliance requirements in the context of a certification program. So far, the growing number of operation types indicates that the mechanism "variability operation" can be used to foster flexibility in a process line (in response to customer requirements).

5.3 RQ 2: Variability Operation Use

The second research question aims at investigating which of the defined operation types are actually used. Fig. 4 quantifies the use within the operation groups and per metamodel version (see also Table 6). An operation type is in the set of used operations if there is at least one exemplar in any of the investigated variants. Fig. 4 shows which metamodel defines how many operation types (per operation group) and how many of them are used in the variants (overall count). Table 6 gives the more detailed perspective based on the exemplars per operation group. In the following, we investigate (1) which variability operation types remain unused, and (2) which types are the most frequently used ones.

Table 3 gives detailed information on the defined, but unused operation types. The table shows that several operation types that are defined in the V-Modell XT 1.3 were not used, e.g., operations to add description text to an existing one. Furthermore, some operations that are newly defined in the metamodel "1.3 B" were also not used. On the other hand, several operation types are frequently used across several process variants. Table 4 lists the most frequently used operation types, including their number of instances. The data shows that most of the frequently used operations modify text fragments of the process description (e.g., *ReplaceSectionText*, *ArrangeSection*) or alter the role model (e.g., *ReplaceRoleDescription*, *RenameRole*). Another interesting finding is the operation type *ChangeRoleClass*, which is on the second rank. 36 instances can be found in two process variants. This special operation type does not change any description text, but modifies the structure of role definitions in the process model in response to a metamodel evolution. That is, beside content-related variability operations, we also found variability operations modifying the structure of process assets, e.g., to enable for backward-compatibility. Finally, the V-Modell XT variant "D" (Table 5) does not contain any variability operation exemplar.

In summary, 25 out of 69 (36.2%) operation types are not used. Refined to the metamodel versions/variants, we get the following numbers: The metamodel "1.3" defines 34 variability operation types of which 12 remain unused (35.3%). The metamodel "1.3 B" introduces 35 new variability operation types; 13 thereof remain unused (37.1%).

Interpretation. As 25 operation types remain unused, one may conclude that about one third of the variability operations seems to be dispensable. For instance, Table 3 shows the operation type *RenameCreatingDependency* to be unused, while *ReplaceCreatingDependencyDescription* is used. The reason for the existence of such operations is mainly for process language completeness: the metamodel defines a pair of *Rename** and *Replace** operations for each of the process dependency types. The definition of these operations was a design decision during the development of the metamodel "1.3 B". Since the V-Modell XT framework is designed as a generic framework, we yet cannot judge the relevance of the unused operation types, as future process variants may potentially use them. Furthermore, our findings show that the most frequently used operations address the customization of description texts, e.g., *ReplaceSectionText* or *ReplaceRoleDescription*. In the context of a software process line, we interpret the frequent use as a standard use case in which a generic process description (reference model) is refined for a particular process variant. Finally, we found the concept of variability operations also applied to compensate metamodel evolution.

However, the analysis of variant "D" (no operation exemplars, Table 5) also shows that variability operations are only one instrument among others and, thus, process lines can also be created and managed using other mechanisms.

DOI: http://dx.doi.org/10.1145/2600821.2600833

Table 3: Unused variability operation types.

No.	Operation Type	MM
4	AddDisciplineDescriptionPrefix	1.3
5	AddDisciplineDescriptionPostfix	1.3
7	DeleteWorkProduct	1.3 B
12	ChangeWorkProduktDiscipline	1.3 B
13	RenameCreatingDependency	1.3 B
17	RenameTailoringDependency	1.3 B
18	ReplaceTailoringDependencyDescription	1.3 B
24	ArrangeSubTopic	1.3
27	AddActivityDescriptionPrefix	1.3
28	AddActivityDescriptionPostfix	1.3
29	RemoveTask	1.3 B
30	RenameTask	1.3 B
31	ReplaceTaskDescription	1.3 B
33	RemoveResponsibility	1.3
38	AddRoleDescriptionPrefix	1.3
40	RefineRole	1.3
42	AddProcessModule	1.3
46	AddDecisionGateDescriptionPrefix	1.3
50	AddChapterTextPrefix	1.3
52	AddSectionTextPrefix	1.3
57	ChangeSectionNumber	1.3 B
59	RemoveChapter	1.3 B
64	RemoveGlossaryItem	1.3 B
65	ReplaceGlossaryItemDescription	1.3 B
68	RemoveAbbreviation	1.3 B

5.4 RQ 3: Variability Operations – No, Why?

So far, we investigated the feasibility of variability operations. Finally, the third research question aims at investigating whether there are situations in which variability is required, but not implemented using variability operations.

Masking. In the analysis, we found a use case that is actually no variability operation, although it could be considered as such. *Pre-tailoring* is a built-in mechanism that allows for a coarse-grained modification of configuration containers (usually to remove whole sub-processes). Furthermore, to construct a process variant, software process lines also allow for adding (completely) new process assets. The combination of pre-tailoring and adding new content can be used in a strategy called "masking", which allows for *replacing* whole sub-processes. In the analysis of the considered V-Modell XT variants, we found two cases in which masking was used to realize variability (top-level process configurations—project type variants—were removed from the reference process and substituted by similar ones).

The analysis of this situation showed the following setting: In several project type variants, an existing process module should be replaced by an equivalent, but customized one. However, no variability operation was defined to perform this replacement. Moreover, it turned out that a variability operation could not be implemented, as operation exemplars refer by *id* and *name* to the process assets to be altered. The used metamodel, however, did not provide these attributes for the specific model element referring the process modules in the process configuration. That is, the missing variability operation caused by a gap in the metamodel was "faked" using the masking mechanism and, thus, is not trackable during an automatic compliance check anymore.

Table 4: Most frequently used variability operations.

No.	Operation Type	QTY	MM
49	ReplaceSectionText	46	1.3
36	ChangeRoleClass	36	1.3 B
37	ReplaceRoleDescription	34	1.3 B
35	RenameRole	22	1.3
66	RemoveLiteratureReference	19	1.3 B
19	RemoveTopicAssignment	16	1.3 B
32	ChangeResponsibility	16	1.3
55	ArrangeSection	12	1.3
20	RenameTopic	10	1.3
45	ReplaceDecisionGateDescription	10	1.3 B

New Sub-Processes. The analysis of variant "D" (Table 5) showed a setting in which a variant was derived without using variability operations. As mentioned before, software process lines also allow for deriving a process variant by adding new content. Variant "D" is an example: The reference model was just taken and *extended* by new content, e.g., new process modules, new roles, and new project type variants assembling the new process modules and such from the reference model. The new content, obviously, showed no need for the use of variability operations as, for instance, no re-naming or text replacements were necessary.

Interpretation. So far, we found two settings in which variability operations were not applied. In one setting, new sub-processes were introduced, which does not require the use of variability operations. However, both instruments can also be combined, e.g., in variant "B" and variant "C" new sub-processes were introduced, but these variants also use variability operations. In the second setting, we found the "masking" strategy, which was used to compensate a gap in the metamodel. In this setting, several other operations were used to simulate missing variability operations.

We interpret our findings as follows: Variability operations are a meaningful tool. However, there are settings in which variability operations are not necessary, and there are also settings in which variability operations would be beneficial (candidates for further metamodel improvements).

5.5 Validity of the Results

In this section, we evaluate our findings and critically review our study regarding the threats to validity. Regarding the *construct validity*, we see no major threat, because the research questions were narrowly defined. We consider this investigation to be a first step. The *internal validity* could be threatened by a bias toward the variant construction process, because two of the authors are also the developers of the metamodels (and partially the processes). We minimized this threat by relying on an analysis tool, which was applied

DOI: http://dx.doi.org/10.1145/2600821.2600833

to all variants, and by calling in a third researcher for triangulation. The *external validity* is threatened as we have little knowledge to which extend we can generalize our results, e.g., to other software process lines. As this is the first analysis of variability in this context, a generalization of the findings is not the intention at this stage. We are interested in analyzing the feasibility of variability operations, and to prepare future research on software process lines.

6. CONCLUSION

Our main goal was to create a catalog of variability operations to support the realization of software process lines. To this end, we opted for the V-Modell XT, and analyzed the reference process and 5 variants that use the built-in process line features. In the V-Modell XT ecosystem, we identified two metamodel versions that define variability operations: the metamodel of the reference process defines 34 variability operation types, and the improved metamodel of the V-Modell XT Bund adds 35 more types. Summarized, we collected 69 variability operation types. The found variability operations allow process engineers to declaratively alter process content, e.g., by providing new text snippets, and the operations also allow for modifying the structure of a process variant, e.g., by changing responsibilities, removing references, and modifying the tailoring behavior. Furthermore, we investigated which variability operations were applied in practice, which allows us to rate the feasibility of the variability operations. Summarized, we found 25 operation types defined, but unused. Among these unused operations, we found two categories of operations: (1) operations that were introduced in the reference model and that are either based on past improvement projects or that are required to ensure a constructive compliance. (2) We found operations that were introduced during the improvement of the metamodel. Such operations were defined to improve the completeness of the process language, e.g., *RenameTailoringDependency* that was introduced, as for all dependency types exist corresponding operations.

Our findings also show that the concept of variability operations not only achieves the requirement to build process variants, but also serves metamodel evolution, which inherently appears in a long-term development. For instance, we found variability operations that allow for structurally modifying "legacy" process assets so that they can be used in newer versions of a process.

Finally, we found settings in which variability operations were not used. In such settings, variability operations were either unnecessary or missing. In the setting in which variability operations were missing, we could identify the "work around" used to simulate the missing operation and, hence, we also identified metamodel improvement candidates.

In summary, we found the concept of variability operations sufficient to support process engineers in constructing a (new) process variant from a process line. However, variability operations are only one instrument among others and, thus, can (and should) be combined with other instruments. We also showed the difficulty to define a set of meaningful variability operations, as we for instance found a number of variability operations defined, but unused. Nevertheless, for all these operations exists rationale why they are part of the model, however, further evaluation remains a topic for future investigation.

Practitioners can also benefit from our findings. As variability operations are means to declaratively define modifications of a reference process, this concept offers payoffs in domains in which regularized processes must be applied, e.g., medicine, automotive, and avionics. A company-specific process can declare the modifications regarding the reference process using variability operations that can be easily tracked and, thus, support audits and assessments.

6.1 Relation to Existing Evidence

In [1, 2, 15, 16, 17], first research was done in the area of (evolutionary) variability analysis. However, these contributions aim at identifying variations considering given models. Our research is focussed on a constructive approach that supports variability by design. Martínez-Ruiz et al. [11] conducted a study in which they investigated the constructors used in tailoring. The literature review revealed that current tailoring constructors do not meet industry requirements, and argue for an instrument that allows for variability and consistency at the same time. The study at hand investigates the concept of variability operations addressing this need (not for tailoring but for a whole process line). Although SPEM already defines a set of basic variability operations (e.g., extends or replaces), no case study is available presenting concrete experiences. Only in [10, 12, 13] addons to SPEM are discussed that are, however, not part of the standard. To the best of our knowledge, no comparable studies are available in the field of process engineering. The study at hand is a step toward closing this gap in literature.

6.2 Limitations

The major limitation is that our investigation is based on the V-Modell XT only. However, to the best of our knowledge, the V-Modell XT is the only process framework that provides process engineers with this kind of support to create process variants. Therefore, the transfer (e.g., to the generic concept provided by SPEM) and the generalization of the findings have to be made carefully. Furthermore, the V-Modell XT provides a rich portfolio of instruments to create process variants. In this paper, we focused on the variability operation instrument, and we barely scratched the surface regarding other instruments (e.g., Sect. 5.4).

6.3 Future Work

As our investigation is based on a snapshot of the V-Modell XT process line, which is based on the version 1.3 of the reference model and all related variants, the study at hand needs to be repeated when the version 1.4 of the reference model is sufficiently disseminated and all variants are migrated to the new reference model. This allows us to better analyze the role of variability operations to support metamodel evolution (as for instance found in Sect. 5.3). Furthermore, a repeated analysis allows for analyzing the evolution of the instrument itself, e.g., are there new variability operations (e.g., addressing the gaps discussed in Sect. 5.4), or are unused variability operations removed. As a second step, independent research is necessary to analyze the transfer options to other frameworks. Variability operations are a meaningful instrument to support process variability, however, as we already discussed in [6] and as also mentioned in [11], there is a gap in process frameworks regarding the capability to model flexible processes. This gap needs to be closed and, thus, it needs to be investigated whether variability operations can substantially contribute.

DOI: http://dx.doi.org/10.1145/2600821.2600833

7. REFERENCES

[1] O. Armbrust, M. Katahira, Y. Miyamoto, J. Münch, H. Nakao, and A. Ocampo. Scoping Software Process Models - Initial Concepts and Experience from Defining Space Standards. In *Intl. Conf. on Software Process*, 2008.

[2] O. Armbrust, M. Katahira, Y. Miyamoto, J. Münch, H. Nakao, and A. Ocampo. Scoping Software Process Lines. *Software Process: Improvement and Practice*, 14(3):181–197, 2009.

[3] G. Chastek, P. Donohoe, K. C. Kang, and S. Thiel. Product line analysis: A practical introduction. Technical report, Software Engineering Institute, 2001.

[4] S. Cohen. Guidelines for developing a product line concept of operations. Technical Report CMU/SEI-99-TR-008, SEI, 1999.

[5] ISO/IEC JTC 1, SC 7. Software engineering – metamodel for development methodologies. Technical Report ISO/IEC 24744:2007, ISO, 2007.

[6] G. Kalus and M. Kuhrmann. Criteria for Software Process Tailoring: A Systematic Review. In *Intl. Conf. on Software and Systems Process*. ACM Press, 2013.

[7] M. Kuhrmann. *Konstruktion modularer Vorgehensmodelle*. PhD thesis, TU München, 2008.

[8] M. Kuhrmann, D. M. Fernandez, and R. Steenweg. Systematic Software Process Development: Where Do We Stand Today? In *Intl. Conf. on Software and Systems Process*. ACM Press, 2013.

[9] M. Kuhrmann, T. Ternité, and J. Friedrich. *Das V-Modell XT anpassen*. Springer, 2011.

[10] T. Martínez-Ruiz, F. García, M. Piattini, and F. De Lucas-Consuegra. Process variability management in global software development: A case study. In *Intl. Conf. on Software and Systems Process*. ACM Press, 2013.

[11] T. Martínez-Ruiz, J. Münch, F. García, and M. Piattini. Requirements and constructors for tailoring software processes: a systematic literature review. *Software Quality Journal*, 20(1):229–260, 2012.

[12] Martinez-Ruiz, T., Garcia, F., Piattini, M., and Münch, J. Applying AOSE Concepts to Model Crosscutting Variability in Variant-Rich Processes. In *EUROMICRO Conf. on Software Engineering and Advanced Applications*, 2011.

[13] Martinez-Ruiz, T., Garcia, F., Piattini, M., and Münch, J. Modeling Software Process Variability: An Empirical Study. *IET Software*, 5(2), 2011.

[14] M. Niazi and S. Zahran. *Software Process Lines: A Step towards Software Industrialization*, chapter 1, pages 1–17. IGI Global, 2012.

[15] A. Ocampo and J. Münch. Rationale modeling for software process evolution. *Software Process: Improvement and Practice*, 14(2):85–105, 2009.

[16] A. Ocampo, J. Münch, and W. Riddle. Incrementally Introducing Process Model Rationale Support in an Organization. In *Intl. Conf. on Software Process*, 2009.

[17] A. Ocampo and M. Soto. Connecting the Rationale for Changes to the Evolution of a Process. In *Intl. Conf. on Product-Focused Software Process Improvement*, 2007.

[18] OMG. Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0. Technical report, Object Management Group, 2008.

[19] D. Rombach. Integrated Software Process and Product Lines. In *Intl. Software Process Workshop (SPW)*, 2005.

[20] SEI. Software Product Lines. Online: `http://www.sei.cmu.edu/productlines`.

[21] T. Ternité. Process lines: A product line approach designed for process model development. In *35th EUROMICRO Conf. on Software Engineering and Advanced Applications*, 2009.

[22] T. Ternité. *Variability of Development Models*. PhD thesis, TU Clausthal, 2010.

[23] T. Ternité and M. Kuhrmann. Das V-Modell XT 1.3 Metamodell. Research Report TUM-I0905, Technische Universität München, 2009.

APPENDIX

A. CONCEPTS & DATA TABLES

In this appendix, we provide the data tables that were created to answer the research questions. Table 5 lists all variability operations, the metamodel in which a particular operation type is defined, and the number of exemplars across all investigated variants. Furthermore, this table serves as a "catalog" for defined and practically applied variability operations. In Table 6, we provide an aggregated view that focusses on the operation groups. For each variant, the table shows the operation instance count per operation group and metamodel version. In this table, we skipped variant "D", as it does not contain any operation exemplar (Sect. 5.3).

Variability Operation Concepts. Although most of the operations have "telling names", we give a very brief introduction to the concepts. A variability operation is a declarative instrument, which is defined in a process variant being derived from a reference model. The operation exemplar refers to a model element in the reference process, and describes how the referred element will be treated during the merge procedure in which the reference model and an extension model are computed in order to compile the company-specific process variant (Sect. 2, Fig. 1). Variability operations are composed of elementary model-transformation operations, e.g., *RenameElement*, *AddText*, *ReplaceText*, or *SwapRefences* [22].

Some variability operations are not intuitive. For space limitations, we give only two small examples. A detailed list and dataset can be depicted from `http://www4.in.tum.de/~kuhrmann/sonst/varops.xslx`.

Example. The operation *AddWorkProductDescriptionPostfix* (No. 10, Table 5) adds a text snippet b to an existing description text a of a work product p. The result in the merged variant is thus: $p.\text{descrText}_{merged} = a \circ b$.

Example. The operation *ChangeResponsibility* (No. 32, Table 5) replaces the responsible role r_1 for a work product p by a role r_2, which means for the merged process and the resulting variant: $resp(r_1, p) \dashrightarrow resp(r_2, p)$. This operation addresses the dependency *RoleIsResponsibleForProduct* [23] and replaces the identifier of r_1 by the identifier of r_2.

DOI: http://dx.doi.org/10.1145/2600821.2600833

Table 5: Overview V-Modell XT variability operation types and exemplars per variant.

No.	Operation Type	Operation Group	MM	Variants					Sum
				A	Bund	B	C	D	
1	RenameDiscipline	Discipline Variations	1.3		1	2			3
2	ChangeDisciplineNumber	Discipline Variations	1.3 B		11				11
3	ReplaceDisciplineDescription	Discipline Variations	1.3 B		1	1			2
4	*AddDisciplineDescriptionPrefix*	Discipline Variations	1.3						0
5	*AddDisciplineDescriptionPostfix*	Discipline Variations	1.3						0
6	RenameWorkProduct	Work Product Variations	1.3	1	1		1		3
7	*DeleteWorkProduct*	Work Product Variations	1.3 B						0
8	ReplaceWorkProductDescription	Work Product Variations	1.3 B		1		2		3
9	AddWorkProductDescriptionPrefix	Work Product Variations	1.3	1					1
10	AddWorkProductDescriptionPostfix	Work Product Variations	1.3	1			2		3
11	RemoveWorkProductDecisionGateAssignment	Work Product Variations	1.3 B		1				1
12	*ChangeWorkProduktDiscipline*	Work Product Variations	1.3 B						0
13	*RenameCreatingDependency*	Work Product Variations	1.3 B						0
14	ReplaceCreatingDependencyDescription	Work Product Variations	1.3 B			1			1
15	RenameContentDependency	Work Product Variations	1.3 B		6				6
16	ReplaceContentDependencyDescription	Work Product Variations	1.3 B		3				3
17	*RenameTailoringDependency*	Work Product Variations	1.3 B						0
18	*ReplaceTailoringDependencyDescription*	Work Product Variations	1.3 B						0
19	RemoveTopicAssignment	Topic Variations	1.3 B		16				16
20	RenameTopic	Topic Variations	1.3.		2	1	7		10
21	ReplaceTopicDescription	Topic Variations	1.3 B		3				3
22	AddTopicDescriptionPrefix	Topic Variations	1.3				1		1
23	AddTopicDescriptionPostfix	Topic Variations	1.3	5			1		6
24	*ArrangeSubTopic*	Topic Variations	1.3						0
25	RenameActivity	Activity Variations	1.3	1					1
26	ReplaceActivityDescription	Activity Variations	1.3 B		1				1
27	*AddActivityDescriptionPrefix*	Activity Variations	1.3						0
28	*AddActivityDescriptionPostfix*	Activity Variations	1.3						0
29	*RemoveTask*	Task Variations	1.3 B						0
30	*RenameTask*	Task Variations	1.3 B						0
31	*ReplaceTaskDescription*	Task Variations	1.3 B						0
32	ChangeResponsibility	Role Variations	1.3		2	10	4		16
33	*RemoveResponsibility*	Role Variations	1.3						0
34	RemoveSupportingRole	Role Variations	1.3 B		2	4	6		12
35	RenameRole	Role Variations	1.3		2	4	16		22
36	ChangeRoleClass	Role Variations	1.3 B		32	4			36
37	ReplaceRoleDescription	Role Variations	1.3 B		13	10	11		34
38	*AddRoleDescriptionPrefix*	Role Variations	1.3						0
39	AddRoleDescriptionPostfix	Role Variations	1.3				4		4
40	*RefineRole*	Role Variations	1.3						0
41	ChangeStandardValue	Tailoring Variations	1.3	1			1		2
42	*AddProcessModule*	Tailoring Variations	1.3						0
43	ReplaceProcessModuleDescription	Tailoring Variations	1.3 B		4				4
44	RenameDecisionGate	Decision Gate Variations	1.3	1					1
45	ReplaceDecisionGateDescription	Decision Gate Variations	1.3 B		10				10
46	*AddDecisionGateDescriptionPrefix*	Decision Gate Variations	1.3						0
47	AddDecisionGateDescriptionPostfix	Decision Gate Variations	1.3	2			1		3

DOI: http://dx.doi.org/10.1145/2600821.2600833

No.	Operation Type	Operation Group	MM	Variants					Sum
				A	Bund	B	C	D	
48	ReplaceChapterText	Description Replacements	1.3		5	2	1		8
49	ReplaceSectionText	Description Replacements	1.3	1	20	22	3		46
50	*AddChapterTextPrefix*	Description Add-ons	1.3						0
51	AddChapterTextPostfix	Description Add-ons	1.3			1	4		5
52	*AddSectionTextPrefix*	Description Add-ons	1.3						0
53	AddSectionTextPostfix	Description Add-ons	1.3	2					2
54	ArrangeChapter	Description Re-Arrangements	1.3	1					1
55	ArrangeSection	Description Re-Arrangements	1.3		1	5	6		12
56	ChangeChapterNumber	Description Re-Arrangements	1.3 B		1				1
57	*ChangeSectionNumber*	Description Re-Arrangements	1.3 B						0
58	RemovePart	Description Removements	1.3 B		3				3
59	*RemoveChapter*	Description Removements	1.3 B						0
60	RemoveSection	Description Removements	1.3 B			5	1		6
61	RemoveMethodReference	Tool/Method Reference Variations	1.3			2			2
62	RemoveToolReference	Tool/Method Reference Variations	1.3			9			9
63	RemoveMapping	Mapping Variations	1.3 B		4	1			5
64	*RemoveGlossaryItem*	Appendix Variations	1.3 B						0
65	*ReplaceGlossaryItemDescription*	Appendix Variations	1.3 B						0
66	RemoveLiteratureReference	Appendix Variations	1.3 B		19				19
67	ReplaceLiteratureReferenceDescription	Appendix Variations	1.3 B		1				1
68	*RemoveAbbreviation*	Appendix Variations	1.3 B						0
69	ReplaceAbbreviationDescription	Appendix Variations	1.3 B		1				1
			Sum	**17**	**167**	**84**	**72**	**0**	

Table 6: Used variability operations by variant, operation type, and metamodel release.

Operation Group	Variant A			Variant Bund			Variant C			Variant B		
	1.3	1.3 B	Sum	1.3	1.3 B	Sum	1.3	1.3 B	Sum	1.3	1.3 B	Sum
Discipline Variations	0	0	0	1	12	13	0	0	0	2	1	3
Work Product Variations	3	0	3	1	11	12	3	2	5	0	1	1
Topic Variations	5	0	5	2	19	21	9	0	9	1	0	1
Activity Variations	1	0	1	0	1	1	0	0	0	0	0	0
Task Variations	0	0	0	0	0	0	0	0	0	0	0	0
Role Variations	0	0	0	4	47	51	24	17	41	14	18	32
Tailoring Variations	1	0	1	0	4	4	1	0	1	0	0	0
Decision Gate Variations	3	0	3	0	10	10	1	0	1	0	0	0
Description Replacements	1	0	1	25	0	25	4	0	4	24	0	24
Description Add-ons	2	0	2	0	0	0	4	0	4	1	0	1
Description Re-Arrangements	1	0	1	1	1	2	6	0	6	5	0	5
Description Removements	0	0	0	0	3	3	0	1	1	0	5	5
Tool/Method Reference Variations	0	0	0	0	0	0	0	0	0	11	0	11
Mapping Variations	0	0	0	0	4	4	0	0	0	0	1	1
Appendix Variations	0	0	0	0	21	21	0	0	0	0	0	0
Sum	17	0	**17**	34	133	**167**	52	20	**72**	58	26	**84**