DOI: http://dx.doi.org/10.1145/2745802.2745814

Development of Flexible Software Process Lines with Variability Operations: A Longitudinal Case Study

Joachim Schramm, Patrick Dohrmann
Technische Universität Clausthal
Department of Informatics
Clausthal-Zellerfeld, Germany
{jschr,pdo}@tu-clausthal.de

Marco Kuhrmann
University of Southern Denmark
The Mærsk Mc-Kinney Møller Institute
Odense, Denmark
kuhrmann@mmmi.sdu.dk

ABSTRACT

Context: Software processes evolve over time and several approaches were proposed to support the required flexibility. Yet, little is known whether these approaches sufficiently support the development of large software processes. A software process line helps to systematically develop and manage families of processes and, as part of this, variability operations provide means to modify and reuse pre-defined process assets. **Objective:** Our goal is to evaluate the feasibility of variability operations to support the development of flexible software process lines. **Method:** We conducted a longitudinal study in which we studied 5 variants of the V-Modell XT process line for 2 years. **Results:** Our results show the variability operation instrument feasible in practice. We analyzed 616 operation exemplars addressing various customization scenarios, and we found 87 different operation types contributed by 3 metamodel variants developed by different teams in different contexts. **Conclusions:** Although variability operations are only one instrument among others, our results suggest this instrument useful to implement variability in real-life software process lines.

Categories and Subject Descriptors

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

General Terms

Experimentation

Keywords

software process, software process lines, variability operations, longitudinal study

1. INTRODUCTION

Software development is diverse, which makes it impossible to find the one and only software process [7]. Different studies (e.g., [4, 7, 10, 25]) show plenty of processes used in practice, and companies usually combine multiple processes and adopt these to company- or project-specific requirements. Yet, defining adequate processes is hard; it is a complex activity requiring deep knowledge of the actual domain in particular and software engineering in general. To overcome these challenges, several approaches were proposed to improve flexibility and reusability of software processes, and to define methods to rapidly compose custom processes. For instance, in Method Engineering [6], processes are composed of reusable assets. For a particular situation, so-called method chunks are selected from a method base and assembled into a situation-specific method. In [18], Rombach votes for adopting well-known concepts from software product lines [1] for developing *Software Process Lines* (SPrL), which define 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 modifying process assets [27]. Yet, these approaches lack in evidence of their feasibility in practice (e.g., [21] analyzed Method Engineering scarcely finding reports on the feasibility, and [5, 3] come to a similar conclusion for SPrLs).

In Germany, the V-Modell XT is the standard software process for IT development projects in Germany's public sector. It has a long history beginning in 1992, and was improved in several iterations. Starting with its release in 2005, the number of process variants using the so-called *reference model* increased [8]. Organizations customized the process by modifying a local copy of the reference model, which caused serious problems when the reference model evolved. Much effort was spent to analyze changes and to develop strategies to update the respective variants (e.g., [14, 15]). However, only the changes could be analyzed, while integrating evolved and customized process assets remained a critical and unresolved task. In response, it was decided to adopt concepts from SPrLs to the V-Modell XT for the purpose of improving support for an efficient management of the reference model *and* its variants. Process engineers are enabled to systematically develop a process variant grounded in the reference model, and benefit from, e.g., automatic updates. The selected approach allows for, notably, using *variability operations* as a declarative instrument to systematically customize the reference model while ensuring, e.g., consistency and compliance with the reference model.

Problem Statement. While defining the variability operations, the major problem was to find a set of suitable and actionable variability operations. Different studies, e.g.,

DOI: http://dx.doi.org/10.1145/2745802.2745814

Carvalho et al. [5] and Martínez-Ruiz et al. [12] found a gap regarding standardized and actionable variability constructors, which is also reflected by available approaches that are either conceptual [13] or generic [17] and require refinement. Missing is a set of practically proven variability operations to support the development of a variant from an SPrL. Often, process engineers develop their own portfolio of variability instruments negatively impacting the SPrL and its variants [26, 27], for example, due to competing or incompatible sets of variability instruments, causing in potential loss of compliance. Furthermore, we still lack in long-term studies analyzing the feasibility of SPrL approaches [5, 3].

Context & Previously Published Material. Facing the challenge to provide actionable variability instruments and to provide evidence regarding the feasibility thereof, our previously contributed study [9] initially analyzed the practical application of variability operations [23]. In this exploratory study, we identified 69 unique variability operation types, which were defined in two V-Modell XT metamodel variants. In total, we analyzed 340 operation exemplars defined in 5 process variants. We concluded that the concept of variability operations is a meaningful instrument to declaratively modify a given software process and to compensate metamodel evolution. Apart from the variability operations, we also found settings in which variability operations were not used, e.g., for technical reasons.

Objectives. To better understand software process variability and to direct further research on the application and the improvement of SPrL approaches in practice, we aim to analyze the feasibility of the variability operation instrument and its improvement to support the systematic development of flexible software process lines.

Contribution. In this paper, we contribute results from a longitudinal case study. For two years, we analyzed two baselines of the V-Modell XT SPrL (cf. Figure 1) of which the baselines each contained the reference model and 5 of its variants using the built-in SPrL features of the V-Modell XT. For each baseline, we analyzed the metamodels to develop a catalog of variability operation types, and we analyzed the process variants to investigate the use of the operation types, i.e., we analyzed which operations are defined and to which extend those are used in practice.

The present paper uses the findings of our previously published exploratory study [9] and—by analyzing a second data set—improves the understanding of variability operations by analyzing the use of this instrument over time and in different process releases. We thus close a gap in literature by providing insights into a practically implemented SPrL.

Outline. The remainder of the paper is organized as follows: Section 2 introduces relevant terminology and Section 3 summarizes related work. In Section 4, we describe the research design, and discuss the results in Section 5. We conclude the paper in Section 6.

2. CONTEXT & TERMINOLOGY

In this section, we briefly set the context and introduce the terminology used. We analyze the V-Modell XT SPrL, which is built on a comprehensive software process framework [8]. In particular, we focus on the concept of variability operations, which is realized as part of the V-Modell XT metamodel [24] and its variability instruments. In subsequent paragraphs, we provide a brief description, whereas a more comprehensive introduction can be depicted from our previously published paper [9].

Process Variants. The V-Modell XT allows for developing hierarchically organized process lines consisting of different *process variants*. In the V-Modell XT, the root variant is called the *reference model*. Extension models, which are built on the same process metamodel (or variants thereof), comprise new contents and modifications of the reference model. As all model elements from the reference model are accessible from an extension model, an extension model can refer to and thus integrate and modify any reference model element. A merge tool computes the *company-specific process* from the reference- and the extension models. New assets introduced by the extension model will then be integrated with the reference model, exclusions will be deleted, and variability operations will be processed.

In the present paper, the term *variant* is used to refer to a particular process variant from the studied SPrL. For example, the *variant Bund* refers to the V-Modell XT Bund (Figure 2, Table 2) and in particular to the respective metamodel and the extension model, as these define the variability operation types and exemplars.

Variability Operations. As part of the V-Modell XT variability instruments, a variability *operation type* is a metamodel element describing a particular change, e.g., renaming of elements, adding description text, or restructuring dependencies. The metamodel of reference model provides a basic set of variability operation types. Extension models can use metamodel variants, which can provide extra operation types. An instance of a variability operation (an *exemplar*) is a model element defined in an extension model. An operation exemplar refers to an element from the reference model to be modified, i.e., the source files of the reference model are not touched, as all change declarations are contained in the extension model. Variability operations are interpreted and executed during the merge procedure in which the company-specific variant is computed (cf. [22, 23]).

In this paper, we use the term *operation type* to denote operations defined in a metamodel that are available for developing SPrLs (Section 5.2), i.e., which process language elements are available. We use the term *(operation) exemplar* to denote the instances of the operation types (Section 5.3), i.e., which types are really used and to which extent.

3. RELATED WORK

Since Rombach's proposal [18] to organize software processes as software process lines, much research has been published discussing this topic, e.g., [13, 22, 23, 27]. Yet only few publications deal with self-contained approaches providing support for process engineers. For instance, in [2], Alegría and Bastarrica present the CASPER approach, which they define as meta-process to support the construction of project-specific processes. Martínez-Ruiz et al. [11] propose an extension of SPEM [17] that allows for improving the definition of variable process elements. Oliveira et al. [16] propose SmartySPEM, which also aims to extend the variability instruments provided by SPEM. Both contributions address the problem that SPEM, basically, provides generic variability operations and modularization concepts, but does not provide explicit and context-specific variability constructors [9]. Furthermore, in line with Carvalho et al. [5] and

DOI: http://dx.doi.org/10.1145/2745802.2745814

Chen et al. [3], both exemplarily cited SPEM-related contributions propose solutions, but lack in empirical evidence of their feasibility.

Although of particular interest, SPrL concepts are still considered immature [5, 3, 12]. For example, Martínez-Ruiz et al. [12] mention the absence of a meaningful notation for variable processes critical. Evidence of the feasibility of proposed concepts is missing. For instance, in [5], authors identified various process elements modified by variability operations. However, they neither provide details of the mentioned operations nor evidence of the feasibility.

In [23], Ternité introduces *typed variability operations* to support software process variability. This instrument is used in the V-Modell XT to realize a complex software process line (Figure 2). A typed variability operation is a change of a reference model element that is declared by a derived process variant, and that is computed during a tool-supported merge process in which a reference model and an extension model are integrated into a company-specific software process. For example, a reference process defines a role *A*, but in a particular company, the same role has the name *B*. The process variant then declares an operation *ChangeRoleName(A, B)* and a merge tool consistently changes all occurrences while computing the integrated process model (cf. [9] for an example, and [22, 23] for comprehensive description). In [9], we initially analyzed the feasibility of this instrument finding it appropriate to implement flexible SPrLs.

With this paper, we extend our initial investigation by analyzing a second baseline of the V-Modell XT process line to improve the knowledge base, to provide extra evidence, and to allow for future research. We address the call for more research on the applicability of SPrL concepts in practice [5, 3] by providing a long-term observation of one particular practically implemented software process line.

4. RESEARCH DESIGN

In this section, we present the study design. After describing the research questions, we describe the overall research design, and the case study instrument including case selection, data collection, and analysis procedures.

4.1 Research Questions

Our overall objective is the investigation of the feasibility and the practical application of variability operations to support the development of flexible software process lines. Therefore, we define the following research questions:

RQ 1: *Which variability operations are defined to realize the process line?* This research question aims to create a catalog of variability operation types. In stage 1 of the study, we developed an initial catalog of operations. In this stage, we analyzed an evolved set of process variants, and analyzed if, e.g., new operation types were defined, for the purpose to extend the original catalog.

RQ 2: *Which variability operations are practically used to which extent?* Based on the found operation types, we investigated which operations were practically used to realize process variants. We quantitatively analyzed the variability operation exemplars to investigate their practical use. We aimed to analyze the general use as well as trends.

RQ 3: *In which settings are variability operations not used and why?* In the first stage of the study, we found two settings in which variability operations were not used and analyzed the reasons. In stage 2, we studied the evolved process variants for such settings, and looked for the already found and new settings in which the use of variability operations is potentially inappropriate.

4.2 Research Method

To investigate the research questions, we conducted a longitudinal case study [28] in which we analyzed two process repository baselines over a 2-year period. In Figure 1, we illustrate the overall approach.

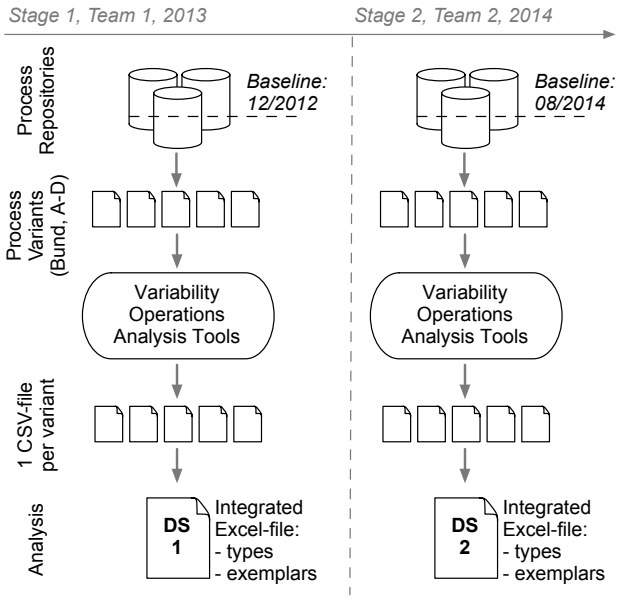

Figure 1: Overview of the research design.

In the explorative stage 1 of the study, we developed the overall research design, created a baseline of the process repositories, and selected those process variants that use the SPrL features provided by the V-Modell XT (Sect. 4.3.1). For each selected variant, we collected variability operation types and exemplars using a tool, which generates CSV files, which were integrated into a spreadsheet providing the data structure shown in Table 1 (Sect. 4.3.2). The initial findings of stage 1 were reported in [9]. In stage 2, we repeated the study. We created another baseline, and we selected the evolved versions of the initially studied variants as cases. In order to ensure the reliability of the data collection outcomes, the same researcher as in the first stage executed the data collection procedure using the same tool. Furthermore, to address validity considerations, a new team of researchers performed the second stage. Both stages produced an integrated data set each (Figure 1: DS1 in 2013 [9], and DS2 in 2014), which are the inputs for the investigation.

4.3 Case Study Instrument

We describe the instrument applied to this study in detail. In stage 2, we repeated the investigation by strictly following the procedures of stage 1 (described in [9]). In general, the case study is organized according to the guidelines proposed by Runeson et al. [19]. In the following, we describe the case selection, data collection, and analysis procedures.

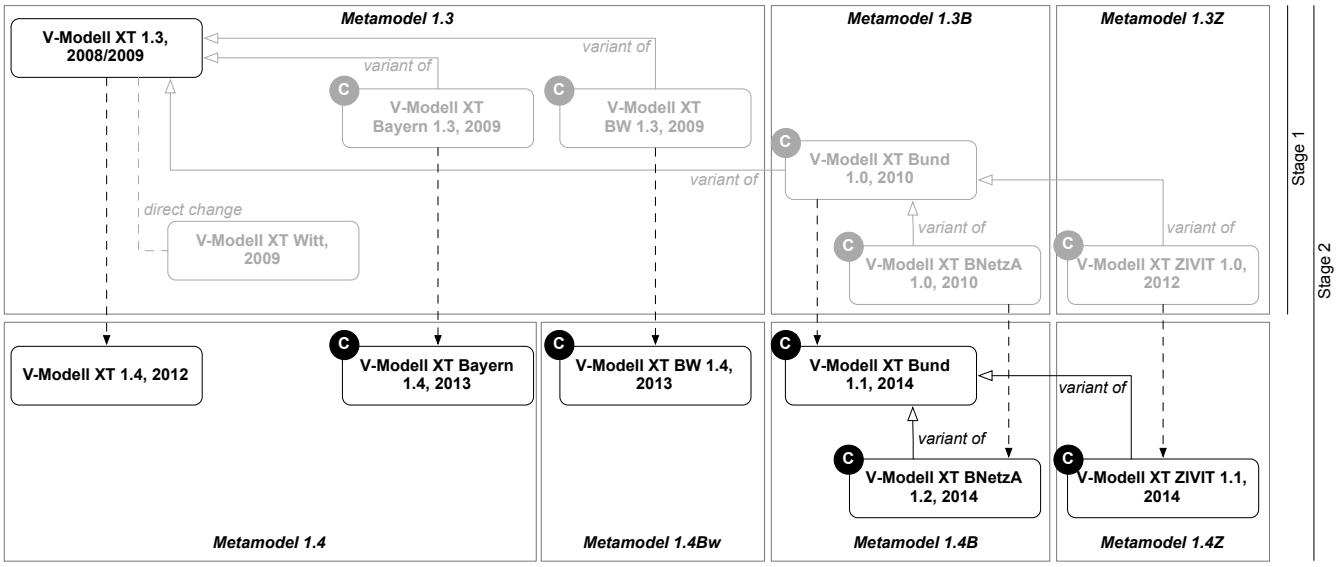

Figure 2: Snapshot of the V-Modell XT software process line (variants marked with "C" are the cases).

4.3.1 Case Selection

In the first stage, we opted for those V-Modell XT variants that were built using the SPrL features provided by the V-Modell XT. In the second stage, we opted for a straightforward case selection and looked for process variants fulfilling the same requirements, especially for evolved versions of the originally studied variants. Variants that are built by copying and directly modifying the reference model were out of scope in stage 2 of the study as well.

4.3.2 Data Collection Procedure

The data collection procedure was performed the same way as in the stage 1: To answer *RQ1* and *RQ2*, we used a tool to export lists of the variability operations defined and applied it to each considered process variant.

Table 1: 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*
Affected elements	List of model elements that are affected by the operation exemplar
Compliance criticality	Rating whether an operation potentially violates compliance requirements/constraints

All information was collected by parsing the models' XML source files, and storing the data in a spreadsheet, which has the structure shown in Table 1, to (1) create a consolidated list of operation types across all versions of the metamodel, (2) to create process-variant-specific lists of operation exemplars, and (3) an aggregated list of all variability operations, their type, number of exemplars, and so forth.

To answer *RQ3*, we had to inspect the considered process variants. We compared the process definition with its sources for added, modified and/or removed process assets that are not defined using variability operations.

4.3.3 Analysis Procedure

Due to the low number of cases, we present the results as data tables, and qualitatively analyze and interpret the results. In order to allow for comparable results, we applied the same analysis procedures as in the first stage, but extended the analysis by a comparative analysis to investigate evolution-related aspects and trends.

5. STUDY RESULTS

We provide a description of the cases and the subjects, before presenting and discussing the results of the study.

5.1 Case Description

We selected the V-Modell XT and the set of 5 variants that use the SPrL features of the V-Modell XT as case. Figure 2 illustrates the snapshots used in the study (stage 1 and stage 2). The emphasized variants were subject to the study. While the first stage was based on the metamodel version 1.3, the second stage is based on those variants using the V-Modell XT 1.4, which we refer to as the *reference model* on which all variants are built[1]. A tool was used in both study stages to compute all variability operation types and exemplars, and creates a dataset, which contains the information listed in Table 1.

5.2 RQ 1: Variability Operation Types

In stage 1 of the study, we found the different metamodel variants defining individual sets of variability operations. In

[1]For confidentiality, we are not allowed to relate the findings to variants from Figure 2. Therefore, we anonymized the relations by referring to variants A to D. A collection of publicly available material is provided by the Weit e.V.: `www.weit-ev.de/varianten.html` (in German).

DOI: http://dx.doi.org/10.1145/2745802.2745814

summary, in stage 1, we found 69 variability operations of which 34 operations were defined by the reference metamodel "MM 1.3", and variant "MM 1.3B" added 35 more operation types. In stage 2, we repeated the study and found a more diverse picture. Again, we first analyzed (1) which variability operation types were defined, and (2) which metamodel variant defines a particular operation type. Finally (3), we analyzed the result set for new operation types and compared the findings to the first stage[2].

Table 2: Overview of the available variability operation types (per stage and studied variant).

Study	**Ref**	**Bund**	**A**	**B**	**C**	**D**
Base		*Ref*	*Ref*	*Bund*		*Ref*
Stage 1	34	69	34	69	69	34
new		35				
Stage 2	34	72	34	72	72	56
new		38				22

In Table 2, we summarize the overall numbers of the variability operation types identified across the different metamodel versions. The process variants *Bund*, *B*, and *D* use extended variants of the reference metamodel (Figure 2). The reference metamodel "MM 1.4" still defines 34 variability operation types. The evolved version "MM 1.4B" of the process variant *Bund* added 38 types ("MM 1.3B": 35), and the variant "MM 1.4Bw" added 22 types to the reference metamodel. Still, the metamodel variant "MM 1.4Z" did not add further variability operation types. The analyzed process variants contain 16 duplicated operation types defined by the metamodels "MM 1.4B" and "MM 1.4Bw", which can be characterized a *direct copy* (7 types) or a *re-defined* operation type (9 types). To conduct the characterization, the operation definition files were inspected, and the operation types, their syntax, and semantics were compared. A copied operation type is identically present in both metamodels (e.g., the operation type *DeleteWorkProduct*), while a re-defined type differs in name, syntax, or compliance relevance, but serves the same purpose, i.e., has the same semantics (e.g., *RemovePart* in "MM 1.4B" and *DeletePart* in "MM 1.4Bw").

In summary, the analysis resulted in 87 different variability operation types. The reference metamodel "MM 1.4" defines 34 operations, "MM 1.4B" adds 38, and "MM 1.4Bw" adds 22 (Table 2). Furthermore, the metamodels "MM 1.4B" and "MM 1.4Bw" contain duplicated operations. Compared to the results from the first stage (69 operation types), in the evolved process line, 18 new variability operation types were defined.

Interpretation

We found variability operation types defined in three of the four analyzed metamodels. Furthermore, compared to the reference metamodel, we found the number of operation types more than doubled, and we also found effects of different process engineering teams working in parallel. In particular, we found 87 different variability operation types across all investigated metamodels. However, in a set of duplicated operation types, we also found 9 types that we classified as *re-defined* operation types developed by different groups of process engineers, which should be harmonized with the rest of the process line metamodels. That is, actually, the whole SPrL provides process engineers with 78 "unique" variability operation types (69 types in stage 1).

Investigating this situation in more detail, we asked the metamodel developers responsible for the "MM 1.4Bw" operation set for explanation and learned: Similar to the setting of stage 1 of the study, extending the operation set was driven by customer requirements. Furthermore, since the respective variant should be grounded in the reference model instead of the V-Modell XT Bund, required variability operations had to be defined in a new metamodel variant. That is, the development approach of the V-Modell XT Bund was replicated by another team in another context. The process engineers copied several operations directly, and—while conducting the customization project—added further operations on demand. This also explains the set of duplicated variability operation types (7 exact copies and 9 re-defined operation types). Eventually, we found two major branches of the metamodel utilizing the variability operation instrument in the respective context.

In summary, in stage 1, we concluded that the variability operation instrument supports flexibility of an SPrL to address customer requirements. The observed evolution in the metamodel variants in stage 2 as well as in the found set of variability operation types support the initial conclusion.

5.3 RQ 2: Variability Operation Use

The second research question aims at investigating which of the defined operation types were actually used. In stage 1, among 15 operation groups, we found one operation group (group: *Task Variations*) of which none of the contained operation was used. Refined to the operation type level, 25 out of 69 (36.2%) operation types were not used. To investigate the operation type use in stage 2, we quantitatively analyzed the use applying the same criteria: an operation type is used if there is at least one exemplar in any of the investigated variants.

In the following, we (1) analyze which operations were used in general, (2) which operation types remain unused, and (3) which operations are the most frequently used. Furthermore, we relate all findings to the results from stage 1 and discuss the development over time.

General Use

Table 3 summarizes the number of operation exemplars in the respective operation groups. In summary, in stage 1, we found 340 operation exemplars across the five analyzed process variants. In the second stage, we found 616 exemplars. Furthermore, in stage 2, no operation group remained unused, i.e., the number of operation exemplars and used operation types increased over time. We found 4 out of 5 variants with increased numbers of operation exemplars. Especially, the V-Modell XT Bund and variant "D" show a significantly increased use. Table 3 shows that variations in roles, work products and topics (as parts of work products), and description texts (of the electronic process guide) contribute the majority of the operation exemplars in both study stages (cf. most frequently used operation types).

[2]For space limitations, the catalog and the raw data is not part of this paper, but is available for download: `http://goo.gl/nHClEm`

DOI: http://dx.doi.org/10.1145/2745802.2745814

Table 3: Overview of operation exemplars per operation group (by study stage).

Operation Group	Bund		A		B		C		D		
	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	
Discipline Variations	13	14	0	0	3	2	0	0	0	0	
Work Product Variations	12	26	3	3	1	0	5	6	0	37	
Topic Variations	21	35	5	5	1	1	9	9	0	32	
Activity Variations	1	20	1	1	0	0	0	0	0	14	*defines new operations*
Task Variations	0	6	0	0	0	0	0	0	0	0	
Role Variations	51	56	0	0	32	42	41	42	0	34	*defines new operations*
Tailoring Variations	4	6	1	1	0	0	1	1	0	2	*defines new operations*
Decision Gate Variations	10	10	3	3	0	1	1	2	0	6	*defines new operations*
Description Replacements	25	25	1	1	24	21	4	5	0	11	
Description Add-ons	0	0	2	2	1	1	4	5	0	0	
Description Re-Arrangements	2	4	1	1	5	5	6	6	0	0	*defines new operations*
Description Removements	3	4	0	0	5	4	1	1	0	6	
Tool/Method Reference Variations	0	0	0	0	11	11	0	0	0	0	
Mapping Variations	4	5	0	0	1	1	0	0	0	0	
Appendix Variations	21	39	0	0	0	0	0	14	0	0	
Text Module Variations		3		0		0		0		3	*new operation group*
Process Decision Gate Variations		0		0		0		0		15	*new operation group*
Product Dependencies Variations		0		0		0		0		5	*new operation group*
Process Module Variations		0		0		0		0		1	*new operation group*
	167		17		84		72		0		Stage 1 $\sum$: 340 exemplars
		253		17		89		91		166	Stage 2 $\sum$: 616 exemplars

Unused Operations

We analyze if there are variability operations defined but not used. Table 5 summarizes the unused operations, illustrates which metamodel defines the operation type, and provides a comparison of both stages.

In stage 1, we found 25 operation types defined but unused. In stage 2, we found 20 unused operation types. Table 5 shows that some of the originally unused operations were used in the evolved process variants. An exception is the operation type *ReplaceTaskDescription*: The original operation (defined in the metamodel "MM 1.4B") remains unused, but the copied operation (defined in the metamodel "MM 1.4Bw") was used. That is, the original operation remains in the set of unused operations, even though an exemplar is present in the analyzed process variants. Table 5 shows three new operations introduced by the metamodel "MM 1.4Bw", which, however, remain unused.

Table 4: Unused operation types per metamodel.

	MM 1.4		MM 1.4B		MM 1.4Bw	
All	34		72		56	
Extension			38		22	
Unused Ext.			6	15.8%	4	18.2%
Unused All	10	29.4%	16	22.2%	14	25.0%

Table 4 summarizes available operations per metamodel variant, the share of un-/used operation types and provides two perspectives: The first perspective analyzes the number of metamodel-specific operation types, i.e., are all new operation types defined in a metamodel variant used? The second perspective analyzes the whole set of operation types that a metamodel variant provides, i.e., the operation types originally defined by the reference metamodel "MM 1.4" plus the extensions. In stage 1, 25 out of 69 (36.2%) operation types were not used (metamodel "MM 1.3" defined 34 variability operation types of which 12 remain unused (35.3%), and "MM 1.3B" added 35 new variability operation types of which 13 remain unused (37.1%)). In the second stage, "MM 1.4" still defines 34 operation types of which 10 remain unused (29.4%), "MM 1.4B" added 38 operation types to the reference metamodel of which 6 were not used (15.8%), and "MM 1.4Bw" added 22 operation types to the reference metamodel of which 4 remain unused (18.2%). Since we have three different metamodel variants, we also analyze the overall use of the provided integrated operation sets. Across all variants using the reference metamodel "MM 1.4", 70.6% of all available operation types are used. From the operations available for "MM 1.4B" variants, 77.8% of the operations were used, and for the "MM 1.4Bw" variants, 75% of the available operations are used.

Most Frequently Used Operations

Finally, we analyze the process variants for the most frequently used variability operation types. In Table 6, we provide a summary and comparison of the most frequently used operation types, which we refer to as the "Top 10" operations. Table 6 lists the Top 10 of stage 1, relates the Top 10 of stage 2, and shows the differences. Four variability operation types from stage 1 are no longer in the Top 10. Furthermore, Table 6 shows changing ranks. For example, the operation type *RemoveSupportingRole* moved to the 3rd place (stage 1: 8) and *ChangeRoleClass* moved to the 5th place (stage 1: 2). Six operation types are still in the Top 10, and four new Top 10 operations suggest new requirements

DOI: http://dx.doi.org/10.1145/2745802.2745814

Table 5: Overview of unused operation types (compared, by stage).

Stage 1	Stage 2	Defined by MM	Used in...
AddDisciplineDescriptionPrefix	AddDisciplineDescriptionPrefix	MM 1.4	
AddDisciplineDescriptionPostfix	AddDisciplineDescriptionPostfix	MM 1.4	
DeleteWorkProduct		MM 1.4B/MM 1.4Bw	Bund, D
ChangeWorkProduktDiscipline	ChangeWorkProduktDiscipline	MM 1.4B	
RenameCreatingDependency	RenameCreatingDependency	MM 1.4B	
RenameTailoringDependency	RenameTailoringDependency	MM 1.4B	
ReplaceTailoringDependencyDescription		MM 1.4B	Bund
ArrangeSubTopic		MM 1.4	Bund
AddActivityDescriptionPrefix	AddActivityDescriptionPrefix	MM 1.4	
AddActivityDescriptionPostfix	AddActivityDescriptionPostfix	MM 1.4	
RemoveTask		MM 1.4B	Bund
RenameTask	RenameTask	MM 1.4B	
ReplaceTaskDescription	*ReplaceTaskDescription*	MM 1.4B	D
RemoveResponsibility	RemoveResponsibility	MM 1.4	
AddRoleDescriptionPrefix		MM 1.4	D
RefineRole	RefineRole	MM 1.4	
AddProcessModule	AddProcessModule	MM 1.4	
AddDecisionGateDescriptionPrefix	AddDecisionGateDescriptionPrefix	MM 1.4	
AddChapterTextPrefix	AddChapterTextPrefix	MM 1.4	
AddSectionTextPrefix	AddSectionTextPrefix	MM 1.4	
ChangeSectionNumber	ChangeSectionNumber	MM 1.4B	
RemoveChapter		MM 1.4B	Bund
RemoveGlossaryItem		MM 1.4B	C
ReplaceGlossaryItemDescription		MM 1.4B	C
RemoveAbbreviation		MM 1.4B	Bund, C
	ChangeSupportingRole	MM 1.4Bw	
	RenameSection	MM 1.4Bw	
	RenameChapter	MM 1.4Bw	
	DeleteChapter	MM 1.4Bw	

regarding process variant construction. Still, the majority of the Top 10 operations addresses modifications in the role model and in the (electronic) process guide.

Furthermore, Table 6 shows the number of exemplars of the operation types, and shows the overall share (QTY shows (1) the Top 10 numbers per stage, and (2) the numbers for those operation types that are part of the Top 10 list in both stages). In first stage, the Top 10 operations contribute 224 of 340 exemplars (65.9%), and in the second stage, the Top 10 operations contribute 327 of 616 exemplars (53.1%). Considering those operation types that are listed in both Top 10 lists we found six operation types contributing 166 of 340 exemplars (48.8%) in stage 1, and 223 of 616 exemplars (36.2%) in stage 2. In other words: Six operation types contribute about the half of all operation exemplars in stage 1 and about one third of all exemplars in stage 2.

Interpretation

In stage 1, we found 25 out of 69 operation types unused, which could have suggested that about one third of the defined operation types are dispensable. In stage 2, we found (1) a significant increase of the operation exemplars in general, and (2) 20 operation types remaining unused. We provide a tentative discussion of these key findings:

We interpret the increased number of operation exemplars (stage 1: 340, stage 2: 616) as consequence of the dissemination of the SPrL concept. The subject of stage 1 was the first generation of process variants using the SPrL features, and as we figured out variant "D" did not use any variability operation type at all. Stage 2 used a baseline that was drawn 18 months after stage 1, and every analyzed process variant ran through at least one improvement cycle. Now, variant "D" also uses the variability operation instrument and contributed 166 exemplars. Except for variant "A", all variants had an increased number of operation exemplars (Table 3). We interpret this as indication for growing acceptance and the practical feasibility of this instrument.

In stage 1, we found 25 out of 69 operation types unused and discussed the rationale for this operations' existence. The major reason was for process language completeness[3]. In the discussion, we opted not to suggest removing unused operations, since we then had no knowledge about future variability requirements. The second data set showed several of the previously unused operations are now in use. We interpret this as follows: over time, variability requirements change and new requirements can occur that demand for operation types that were, so far, unused. Therefore, even the operations that are still unused should not be considered dispensable without continuing the long-term observation.

Finally, in the result set, we found a set of 6 variability operation types contributing a significant share of the overall operation exemplars. Together with the 16 duplicated vari-

[3]If an operation *Rename** was needed, for completeness, an operation *Replace** was defined as well, cf. [9].

DOI: http://dx.doi.org/10.1145/2745802.2745814

Table 6: Overview of the Top 10 operation types (per stage).

Rank	Stage 1	QTY		Stage 2		QTY	
1	ReplaceSectionText	46	46	ReplaceSectionText	→	53	53
2	ChangeRoleClass	36	36	ReplaceRoleDescription	↑	46	46
3	ReplaceRoleDescription	34	34	RemoveSupportingRole	↑	37	37
4	RenameRole	22	22	RenameRole	→	34	34
5	RemoveLiteratureReference	19		ChangeRoleClass	↓	31	31
6	RemoveTopicAssignment	16	16	ReplaceActivityDescription		28	
7	ChangeResponsibility	16		AddWorkProductDescriptionPostfix		27	
8	RemoveSupportingRole	12	12	AddTopicDescriptionPostfix		27	
9	ArrangeSection	12		RemoveTopicAssignment	↓	22	22
10	ChangeDisciplineNumber	11		ReplaceTopicDescription		22	
	overall exemplars: 340	224	166	*overall exemplars: 616*		327	223
		65.9%	48.8%			53.1%	36.2%

ability operations (Sect. 5.2), we interpret this to be a first indication toward standard operation (candidates). However, since this is the first comprehensive study, we cannot yet generalize this finding.

5.4 RQ 3: Variability Operations—No, Why?

In stage 1, we investigated situations in which variability is required, but was not implemented using the variability operations. We could identify two strategies:

- *Introducing new sub-processes* does not call for variability operations at all. In this strategy, the reference model is taken as is and a new sub-process comprising the customized assets is added without adopting the reference process, i.e., this strategy realizes "true" extensions.
- *Masking* serves the compensation of a technical gap in the V-Modell XT metamodel. Masking mimics a variability operation by providing a modified copy of the sub-process that requires customization, and by using the so-called *pre-tailoring* instrument to remove the original (sub-)process. These instruments can also be combined with variability operations to provide a richer set of variability instruments serving different situations and variability requirements.

In the second stage, we still found implementations of these strategies. However, since all investigated variants utilize variability operations, we did not find the aforementioned strategies in their "pure" form, but always in combination with variability operations, e.g., in the V-Modell XT Bund.

Beyond the already known strategies, we could not find more strategies to implement variability without the variability operation instrument.

Interpretation

In the first stage, we interpreted the findings as follows: Variability operations can be considered a meaningful tool, but there are settings in which variability operations are not necessary or not available. Settings in which variability operations would be beneficial can be considered candidates for metamodel improvements. The findings from the second stage support the initial interpretation.

5.5 Validity of the Results

In this section, we evaluate our findings and critically review our study regarding the threats to validity based on Wohlin et al. [28]. The *internal validity* could be threatened by a bias toward the variant construction process, because the authors are also the developers of the metamodels (and partially the process variants). We addressed this threat by relying on a tool, which was used in both stages and for all variants, and by strictly following the research design of stage 1. Furthermore, a new team of researchers performed stage 2 of the study, while only one of the authors of stage 1 supported the study execution to ensure the full implementation of the overall study design. The *external validity* is threatened as we have little knowledge to which extent we can generalize our results, e.g., transfer to other software process lines. As this is the first comprehensive analysis of variability in this context, a generalization of the findings is not the primary intention at this stage. We are interested in analyzing the feasibility of variability operations, and confirming initial findings and to prepare future research on SPrLs by improving the available data.

6. CONCLUSION & FUTURE WORK

The overall goal of our research is to better understand software process variability and to analyze the feasibility of the variability operation instrument, and—by developing a catalog of variability operations—to support process engineers in the systematic and flexible development of process variants within software process lines. To this end, we opted for the V-Modell XT as a practically implemented software process line, and analyzed the reference model and 5 process variants using the built-in SPrL features of the V-Modell XT. In two teams, we conducted a longitudinal case study over a 2-year period in which we analyzed two baselines of the V-Modell XT process line. In the first (explorative) stage, we collected initial data and developed an initial catalog of variability operations [9]. In the second stage, we repeated the study to improve the data set, to analyze evolutionary effects occurring in continuous process maintenance, and to draw initial conclusions.

DOI: http://dx.doi.org/10.1145/2745802.2745814

Summary of Conclusions

Overall, in the V-Modell XT ecosystem, we found four metamodel variants of which three define variability operation types: The metamodel "MM 1.4" of the reference model defines 34 variability operations, the metamodel "MM 1.4B" adds 38 more operations to the basic set, and the metamodel "MM 1.4Bw" adds 22 operations to the basic set. Analyzing the found operations, we found a set of 16 duplicated variability operations in the metamodels "MM 1.4B" and "MM 1.4Bw" of which 7 are exact copies and 9 are re-defined operations, which differ in syntax but not in semantics. In summary, we identified 87 different operation types (including the duplicates) and, harmonizing the duplicated operations, we have to consider 78 unique variability operation types (stage 1: 69 unique types). The found operation types allow process engineers to declaratively modify process content, e.g., by providing new text snippets, and also allow for modifying the structure of a process model, e.g., by changing responsibilities, removing references, and modifying the tailoring behavior.

Furthermore, we investigated which variability operations are applied in practice to determine the feasibility of this instrument. In summary, we found 616 exemplars across all analyzed variants (stage 1: 340), and all analyzed variants use this instrument. However, we found 20 operation types defined, but unused. These operations were either defined during the initial development of the instrument, or the operations were introduced during metamodel adaptations.

In the results, we also observed evolutionary effects: First, even in stage 1, we found operations serving long-term development of a process line. In particular, the operation type *ChangeRoleClass* aids the structural modification of "legacy" process assets so that they can be used in newer or modified versions of a process. Second, we could observe parallel development of variability operations (performed by two teams of process engineers) underpinning that variability operations can be utilized to systematically extend a process language to address specific customer requirements without affecting the rest of the process line.

Apart from the variability operation instrument, just in the first stage, we found further strategies to realize variability without using variability operations. We identified two strategies, which are, eventually, used in combination with variability operations.

Implications

In summary, we found the concept of variability operations sufficient to support process engineers in constructing a process variant from a software process line. However, variability operations are only one instrument among others and, thus, can (and should) be combined with other instruments. Our analysis also showed the difficulty to define variability operations, as we for instance found a number of operations defined, but unused. However, over time we observed the number of unused operation types decreasing pointing to the need for further investigation.

Practitioners can benefit from our findings: Variability operations help to declaratively define modifications of a reference process and, thus, this concept is beneficial in domains in which regularized processes must be applied, e.g., medicine, automotive, and avionics. For instance, a company-specific process declares the modifications of the reference process using variability operations, which can be easily tracked and, eventually, support audits and assessments.

In the context of the V-Modell XT, our findings provide important insights for the developers of the V-Modell XT reference metamodel—they suggest improvements grounded in practical use. Results obtained so far provide two major findings: (1) developers of the metamodel "MM 1.4Bw" can use the findings to harmonize their branch with the rest of the process line, and (2) the developers of the reference metamodel "MM 1.4" find a number of variability operations, which should analyzed for implementation in the reference metamodel to improve process variant creation.

6.1 Relation to Existing Evidence

Variability operations are a meaningful instrument to support process variability, however, as discussed in, e.g., [12], there is a gap in process frameworks regarding the capability to model flexible processes and as discussed in [5, 3] there is a gap regarding process flexibility and the adaptation of process metamodels [20]. Martínez et al. [12] found activities the most frequently modified process assets, followed by artifacts and roles. Our contribution also shows roles frequently modified, however, our results do not support the findings regarding activities and artifacts. In our study, roles and the electronic process guide are the most frequently modified process assets. However, our study is focused on variability operations thus excluding other customization approaches from the investigation.

6.2 Limitations

The major limitation is that our study is based on the V-Modell XT only, which has three major implications: First, we only analyzed one specific platform. However, to the best of our knowledge, the V-Modell XT is the only practically disseminated process framework that provides process engineers with this kind of explicit support to create process variants. Second, as figured out in [8], the V-Modell XT is a national standard. Hence, our investigation is limited to process variants customized for the use in Germany, and may be hard to transfer to other contexts. Finally, so far we observed the evolution of the process line over a 2-year period, and we also analyzed only two baselines of the process line. Therefore, transfer and generalization of the findings have to be made carefully.

6.3 Future Work

We provide a study, which is based on the V-Modell XT SPrL. Since the study includes the two baselines around the reference model versions 1.3 and 1.4, the study needs to be repeated when the reference model is released in its next iteration, and all derived variants are updated and disseminated accordingly. This is a necessary step to improve the knowledge about the use of the variability operations, and to get further insights into the evolution of the process line.

Furthermore, since the investigated concept is a specific approach of the V-Modell XT, still, it remains an open question if, for instance, the generic concept provided by SPEM could benefit from our findings. Moreover, we still lack in evidence-based and generalizable knowledge about, e.g., appropriateness of variability constructors in general and variability pattern. For instance, in the first stage, we observed a trend toward a set of "core variability operations." Only few operation types contributed a high number of opera-

DOI: http://dx.doi.org/10.1145/2745802.2745814

tion exemplars. In the second stage, we observed a similar trend (Sect. 5.3, Table 6). Thus, the paper at hand can be considered an initial step toward a generalization, which, however, requires more and independent research to which we cordially invite the reader.

Acknowledgements

This work would not have been possible without Thomas Ternité and Daniel Méndez Fernández and their contribution to the first stage of the study. Furthermore, we want to thank Jürgen Münch and Sebastian Eder for their valuable feedback on early versions of this manuscript.

7. REFERENCES

[1] Software Product Lines. Online (last visit: 2007-07-10): `http://www.sei.cmu.edu/productlines`.

[2] J. A. H. Alegría and M. C. Bastarrica. Building software process lines with casper. In *Intl. Conf. on Software and Systems Process*, pages 170–179. IEEE, 2012.

[3] L. Chen, M. A. Babar, and N. Ali. Variability management in software product lines: A systematic review. In *Intl. Software Product Line Conference*, pages 81–90. Carnegie Mellon University, 2009.

[4] M. Cusumano, A. MacCormack, C. F. Kemerer, and B. Crandall. Software development worldwide: The state of the practice. *IEEE Software*, 20(6):28–34, 2003.

[5] D. D. de Carvalho, L. F. Chagas, A. M. Lima, and C. A. L. Reis. Software process lines: A systematic literature review. In *Software Process Improvement and Capability Determination*, volume 477 of *CCIS*, pages 118–130. Springer, 2014.

[6] B. Henderson-Sellers, J. Ralyté, P. J. Ågerfalk, and M. R. (Autor). *Situational Method Engineering*. Springer, 2014.

[7] C. Jones. Variations in software development practices. *IEEE Software*, 20(6):22–27, 2003.

[8] M. Kuhrmann, D. M. Fernández, and R. Steenweg. Systematic software process development: Where do we stand today? In *Intl. Conf. on Software and System Process*, pages 166–170. ACM, 2013.

[9] M. Kuhrmann, D. M. Fernández, and T. Ternité. Realizing software process lines: Insights and experiences. In *Intl. Conf. on Software and Systems Process*, pages 110–119. ACM, 2014.

[10] M. Kuhrmann and O. Linssen. Welche Vorgehensmodelle nutzt Deutschland? In *PMV 2014*, LNI. Gesellschaft für Informatik (GI) e.V. (in German), 2014.

[11] T. Martínez-Ruiz, F. García, and M. Piattini. Towards a spem v2.0 extension to define process lines variability mechanisms. In *Software Engineering Research, Management and Applications*, volume 150 of *Studies in Computational Intelligence*, pages 115–130. Springer, 2008.

[12] T. Martínez-Ruiz, J. Münch, and M. Piattini. Requirements and constructors for tailoring software processes: A systematic literature rewview. *Software Quality Journal*, 20(1):229–260, 2010.

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

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

[15] 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*, volume 4589 of *LNCS*, pages 160–174. Springer, 2007.

[16] E. A. Oliveira, M. G. Pazin, I. M. S. Gimenes, U. Kulesza, and F. A. Aleixo. SMartySPEM: a SPEM-based approach for variability management in software process lines. In *Intl. Conf. on Product-Focused Software Process Improvement*, volume 7983 of *LNCS*, pages 169–183. Springer, 2013.

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

[18] D. Rombach. Integrated software process and product lines. In *International Software Process Workshop*, LNCS, pages 83–90. Springer, 2005.

[19] P. Runeson, M. Höst, A. Rainer, and B. Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, 2012.

[20] J. Schramm, P. Dohrmann, A. Rausch, and T. Ternité. Process model engineering lifecycle: Holistic concept proposal and systematic literature review. In *Euromicro Conference on Software Engineering and Advanced Applications*, pages 127–130. IEEE, 2014.

[21] A. H. M. ter Hofstede and T. F. Verhoef. On the feasibility of situational method engineering. *Information Systems*, 22(6-7):401–42, 1997.

[22] T. Ternité. Process lines: A product line approach designed for process model development. In *Euromicro Conference on Software Engineering and Advanced Applications*, pages 173–180. IEEE, 2009.

[23] T. Ternité. *Variability of Development Models*. PhD thesis, Technische Universität Clausthal, 2010.

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

[25] L. Vijayasarathy and C. Butler. Choice of software development methodologies - do project, team and organizational characteristics matter? *IEEE Software*, (99):1–1, 2015.

[26] H. Washizaki. Building software process line architectures from bottom up. In *Intl. Conf. on Product-Focused Software Process Improvement*, volume 4034 of *LNCS*, pages 415–421. Springer, 2006.

[27] H. Washizaki. Deriving project-specific processes from process line architecture with commonality and variability. In *Intl. Conf. on Industrial Informatics*, pages 1301–1306. IEEE, 2006.

[28] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Springer, 2012.