# Modeling Variability in Software Process with EPF Composer and SMartySPEM: An Empirical Qualitative Study

Jaime W. Dias and Edson OliveiraJr
*Informatics Department, State University of Maringá, Maringá-PR, Brazil*

Keywords: Annotative Approach, Compositional Approach, Eclipse Process Framework, Empirical Qualitative Study, SMartySPEM, Variability.

Abstract: Nowadays, organizations are increasingly seeking to customize their software processes according to the market needs and projects experiences. Therefore, a systematic way to achieve such an objective is the Software Process Line (SPrL) technique, in which each member is a customized software process derived from a set of similarities and variabilities of process elements. Compositional and annotative approaches are most referenced for variability management. In this sense, the objective of this paper is to present a comparison between compositional and annotative approaches targeting variability representation capabilities from the point of view of practitioners and academic experts. The Eclipse Process Framework by means of the EPF Composer tool represents compositional approach, whereas SMartySPEM represents the annotative approach. The obtained results provided initial evidence that the annotative approach (SMartySPEM) takes advantage over the compositional approach (EPF).

## 1 INTRODUCTION

Software development companies need to quickly develop software with quality, which makes them adopt software reuse techniques (Aleixo et al., 2013). Such a reuse aims at improving productivity, reducing coding effort and increasing quality as the software has been already verified and validated.

The current scenario of competitiveness has led software companies to seek solutions beyond software reuse in order to reduce time-to-market and increase return on investment (ROI). Thus, software development projects need to be tailored according to the needs of a company and its development domain (Garcia-Borgonon et al., 2014). Therefore, the Software Process Line (SPrL) technique (Rombach, 2005) is an alternative for process customization based on similar and variable process elements. Such elements can be defined based on the concept of variability, from the Software Product Line (SPL) technique (Linden et al., 2007).

Currently, there are several tools and languages for software process modeling (Garcia-Borgonon et al., 2014), as well as approaches that guide such modeling, as for instance, compositional, annotative, transformational and model-driven (Kästner, 2010; Kästner et al., 2008; Kästner and Apel, 2008). Each approach is used in specific ways to represent variability among process elements (Aleixo et al., 2013). Therefore, this work presents a qualitative empirical study comparing the compositional approach, represented by the Eclipse Process Framework (EPF) and its Composer tool, and the annotative approach, represented by the SMartySPEM (OliveiraJr et al., 2013) approach for variability representation in software process elements.

## 2 BACKGROUND

This section presents essential concepts of software process lines, variability and the OpenUP-based SPrL, used in our qualitative study.

### 2.1 Software Processes and Process Lines

A software process can be defined as a set of techniques and technologies to support, evaluate and improve software development activities. The need to specifying software processes arises from the fact that the products quality can be directly influenced by the process adopted for their productions (Chemuturi and

Cagley, 2010). The ISO/IEC 15504 standard (ISO, 2012) defines a process as a set of activities that are interrelated or interacting to transforming inputs into outputs. This set of activities serves as a guide for those who will be responsible for the process execution and monitoring.

A software development process has four basic steps (Sommerville, 2015): specification of the software features and premises for its development; design for constructing the software according to its specifications; validation to ensure that the software meets the users needs; and evolution in order to accommodate prospective necessary modifications.

Modeling of such steps are essential for a complete understanding of the process (Garcia-Borgonon et al., 2014). Basic elements and concepts are essential for software process modeling: Role, which describes how people act in the process and their responsibilities; Task, which is an action performed by a role for executing or monitoring a project; Activity, which is a set of tasks that lead to produce/consume one or more controlled quality artifacts; Artifact, which represents the result of a task; and Process, which is an organized collection of activities.

Basic elements of a software process are essential to enable process tailoring and customization, which is currently an important research topic from the academic and industrial point of view (Martínez-Ruiz et al., 2012; Kalus and Kuhrmann, 2013; Carvalho et al., 2014). Therefore, the term Software Process Line (SPrL), proposed by Rombach (Rombach, 2005), has been considered in the last years, suggesting the adoption of important concepts from software product lines, such as similarities and variabilities. An SPrL provides techniques and mechanisms for modeling existing similarities and variabilities in a family of software processes, as well as the derivation of customized software processes that meet the specific needs of a given software development project (Rombach, 2005; Aleixo et al., 2011).

SPrLs may contain variation points, which are process elements that can be instantiated in different ways. For each variation point there are variant elements, which can be selected to resolve a specific variation point (OliveiraJr et al., 2013). Figure 1 illustrates an excerpt of an SPrL, in which the similar (mandatory) part is composed of the role `Developer` and two tasks: `Design the solution` and `Implement solution`. Such a figure also illustrates two variabilities: (i) the inclusion of developer tests practices (`Implement developer tests` and `Run developer tests`), and (ii) the inclusion of integration and creating of a build (`Integrate and create build`). This excerpt can generate four distinct process instances: (i) with common elements only by discarding the variabilities, keeping mandatory process elements, (ii) only resolving the variability concerned with `Developer test`, (iii) only resolving the variability concerned with `Integration continues`, and (iv) resolving both variabilities considering all the elements.

## 2.2 The OpenUP-based SPrL

The OpenUP process complies with the principles of the Agile Software Development Manifest, thus it can be taken as an agile version of the Unified Process (UP), meeting UP good practices. OpenUP is an iterative and incremental approach, with no specific tools.

Several activities of the OpenUP are optional, however, it does not define which elements of the processes vary from the SPrL point of view. Thus, in the work of (Aleixo et al., 2011), Aleixo et al. presents excerpts of the OpenUP modeled as an SPrL, defining similar features and variabilities. To do so, three real research and development projects based on OpenUP in addition to experienced practitioners were involved in the definition of such an SPrL. The first project dealt with the development of a software system to audit telephone networks, the second project involved the development of a module of a distributed system and the third project involved the implementation of an integrated academic and administrative management. As a result of this analysis it was identified 586 features, from which: 273 mandatory features, 239 optional features and 74 alternative features.

## 3 VARIABILITY MODELING IN SPRL WITH COMPOSITIONAL AND ANNOTATIVE APPROACHES

The success of an SPrL depends on the accuracy of its variability management activity (Rombach, 2005). Thus, such a management is a key requirement in the development of SPrLs to provide support to specification, implementation, variability resolution and customized processes generation. Variability management defines how common and variable artifacts are represented and treated in order to generate process instances from an SPrL.

There are different approaches and techniques for variability management in the literature. They can be classified as (Galster et al., 2013): compositional, annotative, transformational, and model-driven. This study concerns on compositional and annotative ap-

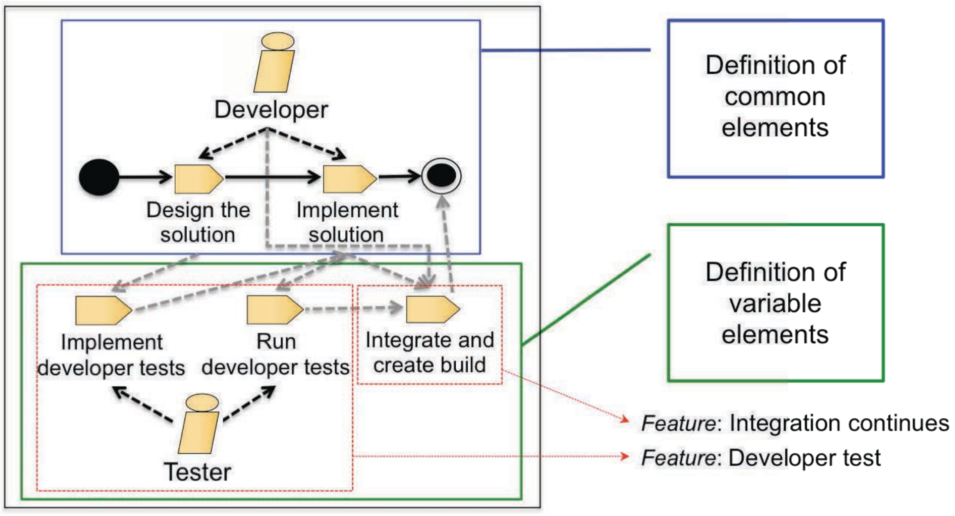

Developer
Design the solution
Implement solution
Implement developer tests
Run developer tests
Integrate and create build
Tester
Definition of common elements
Definition of variable elements
*Feature*: Integration continues
*Feature*: Developer test

Figure 1: Excerpt of an SPrL (Aleixo et al., 2013).

proaches as they are highly referenced in the literature.

## 3.1 Compositional Approach

The compositional approach supports modularity of physical features, thus generation of products occurs by means of selecting and composing modules that implement features of desired products (Galster et al., 2013). Therefore, development techniques are highlighted: Feature-Oriented Programming (FOP) (Lee and Kang, 2013) and Aspect-Oriented Programming (AOP) (Kiczales et al., 1997). FOP provides support to the similarities and variabilities be modularized and each feature implemented in a distinct module. Such a module is an increment in the functionality of a base system (step-wise refinement).

An example of software process modeling using the compositional approach is the Eclipse Process Framework (EPF), which allows editing, configuring and publishing of software processes. EPF persists process information according to the Unified Method Architecture (UMA) meta-model, developed based on the SPEM 1.0. Subsequently, UMA inspired the creation of the SPEM 2.0.

Figure 2 presents the EPF framework main parts using the EPF Composer, a process modeling tool based on the EPF framework, as follows:

- Method Content: standardizes representation and manages reusable component libraries. It defines roles, tasks, work products and their relationships;
- Process: determines the sequence of phases, iterations and activities, and defines when tasks are performed;

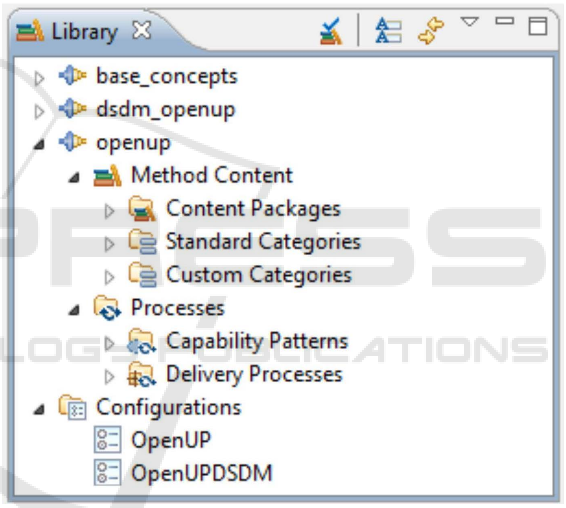


Figure 2: EPF Excerpt Illustrating Method Content, Process, Plug-ins and Configurations using the EPF Composer Tool.

- Plug-ins: represent a set of Method Content and process packages, allowing process customization;
- Configurations: selects a subset of Method Contents to form a specific process by publishing it in HTML or exporting it to MS Project or XML.

The EPF framework allows representing artifacts variability in order to control the evolution and reuse of software processes. There are four possible types of variability in EPF, which are:

- Contributes - contributing (variability) element adds to a base element;
- Replaces - replacing (variability) element replaces parts of the base element;

- Extends - extending (variability) element inherits characteristics of the base element. The base element is unchanged; and
- Extends-Replaces - combines the effects of extends and replaces variability, allowing one to selectively replace specific attributes and relationships of the base element. Extending-replacing (variability) element replaces values in the base element that have been redefined in the extending-replacing element.

### 3.2 Annotative Approach

The annotative approach provides the use of preprocessor directives to annotate code snippets associated with a particular feature (Kästner et al., 2008). The C and C++ languages already support preprocessor directives. Products generation occurs by defining the value of the symbolic constant of pre-processing directives associated with selected features, before pre-building, in order to define the presence of the features snippets selected in the generated product. Just as in Java annotations, which provide the option of using metadata over code that can be later interpreted by a compiler or pre-compiler that performs predefined tasks. Another way of annotation is the UML stereotypes that add semantics to existing elements with no changes in their meta-model.

Stereotype-based Management of Variability for the SPEM meta-model (SMartySPEM) (OliveiraJr et al., 2013) is an approach that provides the separation of elements and their management by using a visual annotation that associates notes and stereotypes to each type of process elements variability.

The SMartySPEM approach aims at supporting the identification and representation of variability in processes elements modeled with SPEM. To do so, SMartySPEM introduces the profiling mechanism based on the SMarty approach (OliveiraJr et al., 2010) for representing variability in SPEM modeled elements with specific stereotypes. SMartySPEM is composed of a UML 2.0 compliant profile, the SMartySPEMProfile, with the following stereotypes (OliveiraJr et al., 2013):

- ≪**variability**≫ - represents the concept of variability (UML note);
- ≪**variationPoint**≫ - represents the concept of variation point (VP icon), in which a variable process element provides a set of choices for customizing a software process;
- ≪**mandatory**≫ - represents compulsory process elements (MDT icon), present in every customized software process;
- ≪**optional**≫ - represents an optional variant;
- ≪**alternative_OR**≫ - represents inclusive variants (OR icon) to resolve a variation point;
- ≪**alternative_XOR**≫ - represents mutually exclusive variants (XOR icon) to resolve a variation point;
- ≪**mutex**≫ - represents mutual exclusion constraint (dependency relationship) among variants; and
- ≪**requires**≫ - represents that a given variant requires the presence of another variant (dependency relationship).

Figure 3 presents an excerpt of an `Architectural Analysis` activity modeled according to SMartySPEM. It contains the variation point `Analysis Class` with three related inclusive variants: `Control Class`, `Entity Class` and `Boundary Class`. Another variation point is `Architectural Analysis` with three variants: `Identifying Common and Special Requirements`, `Identifying Obvious Entity Classes` and `Develop Business Type Model`. `Architect` is a mandatory element for performing the `Architectural Analysis` activity. There are optional variants, such as, `Use-Case Model` and `Architecture Description`, which may or may not be present in derived processes (OliveiraJr et al., 2013).

## 4 EPF COMPOSER VS. SMartySPEM: A QUALITATIVE STUDY

This study aims at comparing compositional and annotative approaches represented, respectively, by EPF Composer and SMartySPEM. The comparison criteria to be adopted were proposed in the works (Kästner, 2010; Kästner et al., 2008; Kästner and Apel, 2008), namely: modularity, traceability, error detection, granularity, adoption and systematic variability management. These criteria were also used in the qualitative study of Aleixo et al. (Aleixo et al., 2012).

The criterion of `modularity` aims at analyzing the modularization degree of processes elements associated with specific features, enabling a better understanding and facilitating the maintenance and evolution of SPrLs. `Traceability` allows one to analyze how difficult is viewing and mapping of all process elements along with their associated features. The `error detection` criterion analyzes how efficient is

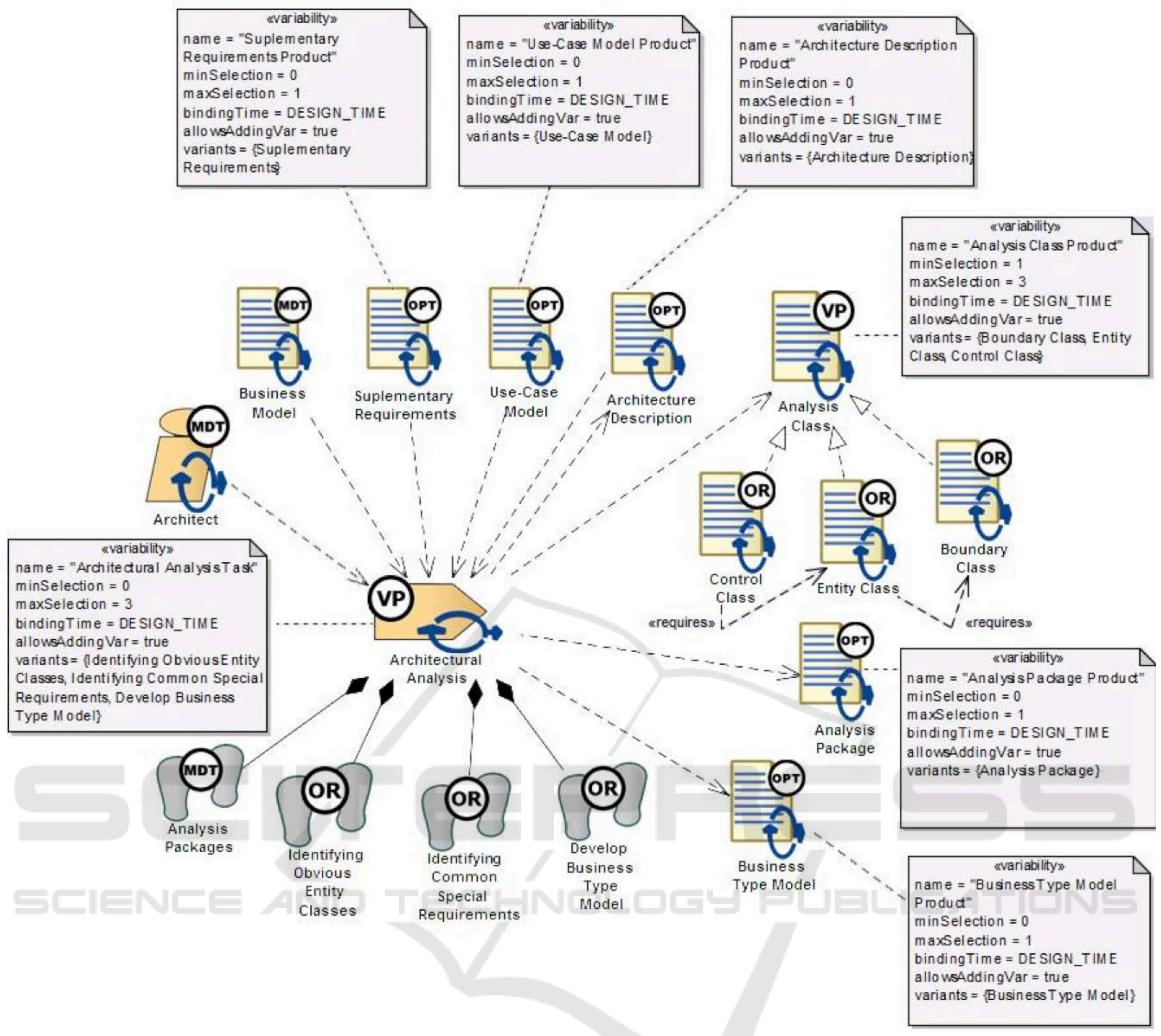

«variability»
name = "Suplementary Requirements Product"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowsAddingVar = true
variants = {Suplementary Requirements}
«variability»
name = "Use-Case Model Product"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowsAddingVar = true
variants = {Use-Case Model}
«variability»
name = "Architecture Description Product"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowsAddingVar = true
variants = {Architecture Description}
«variability»
name = "Analysis Class Product"
minSelection = 1
maxSelection = 3
bindingTime = DESIGN_TIME
allowsAddingVar = true
variants = {Boundary Class, Entity Class, Control Class}
MDT
Business Model
OPT
Suplementary Requirements
OPT
Use-Case Model
OPT
Architecture Description
VP
Analysis Class
MDT
Architect
OR
Control Class
OR
Entity Class
OR
Boundary Class
«requires»
«requires»
«variability»
name = "Architectural Analysis Task"
minSelection = 0
maxSelection = 3
bindingTime = DESIGN_TIME
allowsAddingVar = true
variants = {Identifying Obvious Entity Classes, Identifying Common Special Requirements, Develop Business Type Model}
VP
Architectural Analysis
OPT
Analysis Package
«variability»
name = "Analysis Package Product"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowsAddingVar = true
variants = {Analysis Package}
MDT
Analysis Packages
OR
Identifying Obvious Entity Classes
OR
Identifying Common Special Requirements
OR
Develop Business Type Model
OPT
Business Type Model
«variability»
name = "Business Type Model Product"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowsAddingVar = true
variants = {Business Type Model}

Figure 3: An Excerpt of Architectural Analysis Modeled with SMartySPEM (OliveiraJr et al., 2013).

an approach for identifying cohesion errors in the definition of an SPrL and its elements, as well as its derived processes. `Granularity` evaluates the approach support for representing variability in coarse and fine granularity, considering the division of the process in small or large parties. The `adoption` criterion discusses the difficulty of adopting an approach, analyzing necessary pre-knowledge to be taken to applying such an approach. `Systematic variability management` analyzes the mechanisms provided by an approach for specifying variability.

## 4.1 Objective

The aim of this study is to obtain experts' feedback with regard to the criteria for each approach. Then, compare such approaches in order to be able to draw initial evidence based on each criterion.

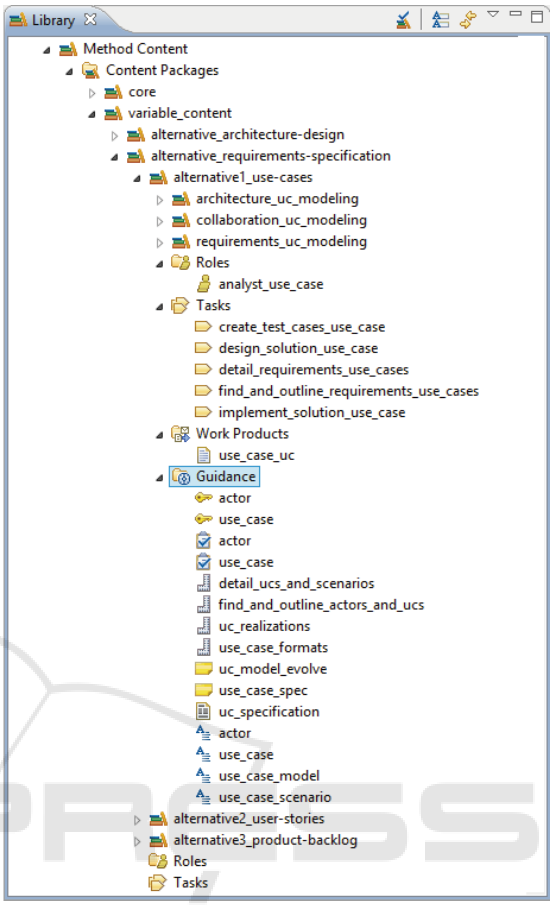
## 4.2 Planning

As study object we used the OpenUP-based SPrL from Section 2.2. As such an SPrL is too large with hundreds of features, we chose only the `Requirements Specification` feature.

Participants were given eight documents: a study consent stating the confidentiality of the responses; a characterization questionnaire to measure the participant's experience; a document with main concepts of SPrL; a document about the compositional approach using the EPF framework; a document about the annotative approach using SMartySPEM; a document with the modeling of the `Requirements Specification` feature in EPF (Figure 4); and a document with the modeling of the `Requirements Specification` feature in SMartySPEM (Figure 5). After the trainning session, each participant was given two questionnaires containing six questions. Each question with regard to each criterion for each approach. Then, participants should answer the following questions, replacing TYPE_OF_APPROACH with "Compositional" or "Annotative":

1. The `modularity` criterion measures the quantity of modules (groups of process elements) necessary for representing an SPrL, thus is it possible to measure the modularity of the TYPE_OF_APPROACH approach?
2. The `traceability` criterion allows analyzing the visualization and mapping difficulty of all process elements along with their features, thus is it possible to visualize the traceability of the TYPE_OF_APPROACH approach?
3. The `error detection` criterion analyzes how efficient is an approach to identify cohesion errors in the definition of an SPrL and its elements, as well as the derived processes from the SPrL. Is it possible to detect cohesion errors in the TYPE_OF_APPROACH approach?
4. The `granularity` criterion aims at evaluating the approach support for representing variability in coarse and fine granularities (level of abstraction), thus considering the process division in small or large parts, is it possible to evaluate the granularity at the TYPE_OF_APPROACH approach?
5. The `adoption` criterion discusses the difficulty of adopting an approach, analyzing the amount of previous knowledge for applying an approach, thus have you experienced any difficulties for understanding the TYPE_OF_APPROACH approach?
6. The `systematic variability management` analyzes the provided mechanisms of an approach for specifying variability. Do you consider sufficient the variability mechanisms of the TYPE_OF_APPROACH approach?

Participants of this study were carefully selected based on their experience with software process. In average, each participant had ten years of experience working with software processes. Thus, twelve participants were invited for this study, from which one participant was invited for a pilot study for evaluating our instrumentation, thus his/her results were discarded. Amongst the participants are researchers and practitioners from the State University of Maringá (UEM), Federal University of São Carlos (UFSCar), Federal Technological University of Paraná (UTFPR)



Library

- Method Content
  - Content Packages
    - core
    - variable_content
      - alternative_architecture-design
      - alternative_requirements-specification
        - alternative1_use-cases
          - architecture_uc_modeling
          - collaboration_uc_modeling
          - requirements_uc_modeling
          - Roles
            - analyst_use_case
          - Tasks
            - create_test_cases_use_case
            - design_solution_use_case
            - detail_requirements_use_cases
            - find_and_outline_requirements_use_cases
            - implement_solution_use_case
          - Work Products
            - use_case_uc
          - Guidance
            - actor
            - use_case
            - actor
            - use_case
            - detail_ucs_and_scenarios
            - find_and_outline_actors_and_ucs
            - uc_realizations
            - use_case_formats
            - uc_model_evolve
            - use_case_spec
            - uc_specification
            - actor
            - use_case
            - use_case_model
            - use_case_scenario
        - alternative2_user-stories
        - alternative3_product-backlog
        - Roles
        - Tasks

Figure 4: The OpenUP-based SPrL Requirements Specification Using the EPF Composer.

and University of York, all of them with masters or Ph.D.

The realization of a pilot study led us to changes in the study planning as, for instance, in the study duration time. At first, we distributed all the materials and asked the pilot study participant to answer the two sets of six questions. This whole process took about two hours and forty minutes, making the participant tired and bored. Thus, we decided two divide it into two parts, each part in different days: the first one involving only one of the approaches and the second one the other approach. Therefore, each part of the study took no longer than 50 minutes.

Another necessary change based on the pilot project results was removing the uniformity criterion that aimed at assessing the technology or independent meta-model. As the two approaches depend on specific tools (EPF Composer and a UML Tool), we understand that such a criterion could be a potential threat to our study as the participants do not have enough experience with these tools, especially EPF Composer.

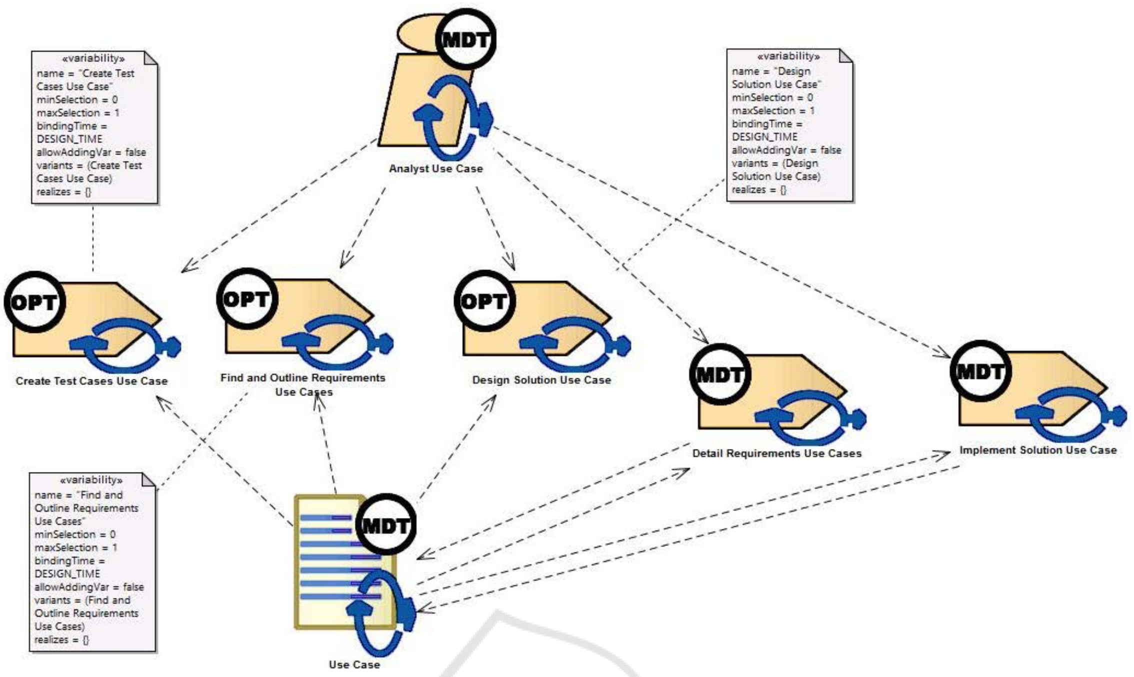

«variability»
name = "Create Test Cases Use Case"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowAddingVar = false
variants = (Create Test Cases Use Case)
realizes = {}
MDT
Analyst Use Case
«variability»
name = "Design Solution Use Case"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowAddingVar = false
variants = (Design Solution Use Case)
realizes = {}
OPT
Create Test Cases Use Case
OPT
Find and Outline Requirements Use Cases
OPT
Design Solution Use Case
MDT
Detail Requirements Use Cases
MDT
Implement Solution Use Case
«variability»
name = "Find and Outline Requirements Use Cases"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowAddingVar = false
variants = (Find and Outline Requirements Use Cases)
realizes = {}
MDT
Use Case

Figure 5: The OpenUP-based SPrL Requirements Specification According to SMartySPEM.

For reducing the threats to validity of the study, for each participant we changed the order of the evaluated approach. Then, five participants firstly received the annotative approach (SMartySPEM), whereas six participants firstly received the compositional approach (EPF Composer).

Responses from the experts were qualitatively analyzed using Grounded Theory (GT) procedures (Corbin and Strauss, 2008). The GT approach is based on coding concepts. Coding allows one to assign codes or labels for text snippets (Open Coding), which can be grouped and classified (Axial Coding) according to an idea expressed in order to elucidate a given phenomenon (Corbin and Strauss, 2008). As a result, such codings enabled the creation of a conceptual model. During the encoding process two categories were created "Feasible Criteria" and "Non-Feasible Criteria", thus grouping answers with respect to each criterion. An assistance tool for qualitative analysis was used, named Dedoose. Such a tool allows one to import a spreadsheet with results, then creating codings in specific excerpts of the answers. The created codings can relate to one another in a significant level. Then, after creating all codings one can produce graphical representations, such as, charts with different results visualizations.

### 4.3 Results

Figure 6 presents the results of this study as a comparison of the answers for each criterion of EPF Composer and SMatySPEM. The more centralized the line is (red or blue) in the chart the worse is the evaluation of a given criterion. As a result, the compositional approach was better evaluated for criteria `Modularity` and `Error Detection`, whereas the annotative approach was better evaluated for criteria `Traceability`, `Granularity`, `Adoption` and `Systematic Variability Management`.

**Modularity** in the annotative approach obtained nine positive evaluations, for example, expert #3 answered that "*...it is possible to measure such a modularity, as well as presenting the relationship between process elements in a module. Such a view contributes to the comprehension of what happens in each process module and, consequently, enables better understanding the tasks of a given derived process...*". However, it took two negative answers. Expert #5 stated that "*...SMartySPEM does not appear to enable an effective organization of the process elements of an SPrL. It is possible to establish the relations, but not distribute them so efficiently...*". Expert #4 answered that "*...as the process model grows, measuring modularity becomes more complex as it depends on the visualization (annotations) of all process elements in a diagram...*". In fact, for large SPrLs, visualization of the modules is jeopardized due to the amount of elements and annotations in a same diagram. On the other hand, the compositional approach took 100% of positive answers as it allows grouping the SPrL com-

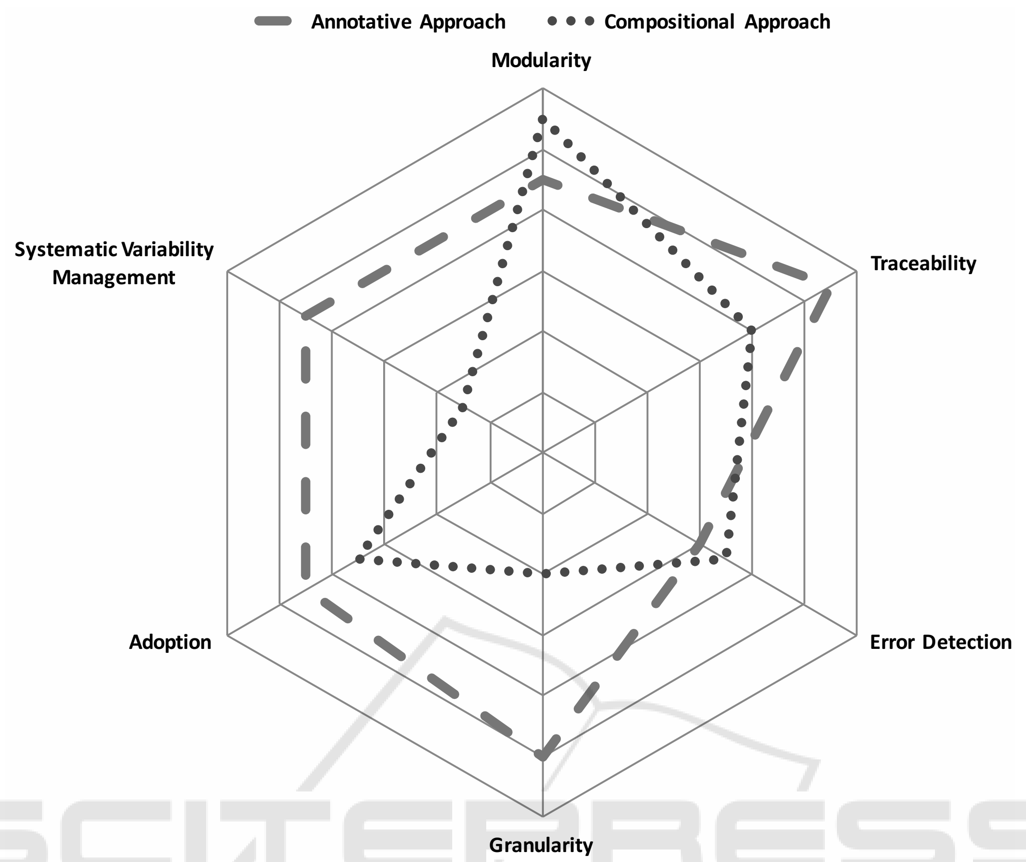


Figure 6: Spider Chart for Evaluating Compositional and Annotative Approaches based on the Defined Criteria.

ponents in hierarchical packages, with a better visualization of the modules as, for instance, reported by the expert #5 "*...EPF Composer is able to encapsulate the compositional elements in a satisfactory way. The use of plugins, method contents and configurations enables reusing such modules in an agile and intuitive way...*".

The **error detection** criterion was not satisfactory for both compositional and annotative approaches, as none of them provides verification activities. However, the compositional approach took a subtle advantage as the EPF Composer checks models at specific process publication (derivation) time as stated by expert #11 "*...it has some kind of support mechanisms for error detection. Some filters, as in the association with variabilities avoid wrong contributions/replacements/extensions of incompatible process elements...*". On the other hand, in the annotative approach, as it is an UML extension and it allows the relationship among several different elements, it is difficult to detect errors, as said by expert #8 "*...Because there is no automation of the approach, the only way to detect errors would be based on reviews and in-depth analysis comparing the process modeling with the description of the SPrLs elements...*".

Traceability obtained all the positive ratings for the annotative approach as it has a very visual appeal, facilitating the identification and traceability among process elements, as well as the `realizes` meta-attribute of the stereotypes ≪variability≫ and ≪variationPoint≫, which represents a collection of low-level models that realizes a given variability. An example is the statement from expert #8 "*...traceability is perceived by means of dependencies among process elements, as well as the* `realizes` *meta-attribute from variabilities or optional and variation point elements...*". For the compositional approach eight positive answers were given as such an approach allows relationships among process elements. However, various negative aspects were pointed out as the difficulty of graphically visualization of the relations between elements, and hiding of variability elements as stated by the expert #5 "*...The user accessing the published process, in html format, generated by the EPF Composer can not visualize inherited packages and configurations of a given element, when its variability allows such a visualization...*".

The granularity criterion had much more positive evaluations for the annotative approach than the compositional due to the possibility of modeling different types of diagrams representing different abstraction levels of SPrLs. For instance, flow of activities with Activity elements for coarse-grained granularity and diagram description of a process with activities as fine-grained granularity. We can observe these examples from the expert #2 answer excerpt as follows: "*...the annotative approach clearly presents two levels of abstraction encompassing both fine and coarse-grained granularity, as the Activity process element provides such an abstraction. Abstraction levels allow and facilitate the visualization of the whole process, as well as make it easier identifying the variation of each of the process elements in lower-level abstraction levels...*". In the compositional approach this criterion obtained seven negative evaluations, because of the approach consider only one low abstraction level, not allowing a wide view of the SPrL. Examples of this characteristic are the expert #5 answer "*...the compositional approach has fine-grained granularity as the EPF Composer is aimed at configuring all aspects of the process by structuring it in detail...*" and the expert #1 answer "*...There is no higher abstraction level in order to compare variabilities of an SPrL...*".

The adoption criterion obtained nine positive evaluations for the annotative approach against seven for the compositional approach. We understand that this is straightforwardly related to the influence that the annotative approach because of the UML standard notation as we can see in the statements of expert #4 "*...the level of difficulty on applying the approach is low...*", expert #10 "*...I had no difficulty at understanding the approach as the number of elements to be learned for adopting the approach is not very large. I would adopt the approach without much effort...*" and expert #11 "*...I believe that the difficulty of adopting this approach is low, as its stereotypes, comments and elements are quite similar to UML representations...*". On the other hand, in the compositional approach, behavior complexity and use of variability types influences the adoption of such an approach, as well as difficulties in using the EPF Composer tool to manipulate the SPrL elements, as reported by the expert #2 "*...I had difficulties mainly in the elements that define variability. Also, such a tool needs extra effort on creating and maintaining process elements...*".

For **systematic variability management** the compositional approach was highly unsatisfactory, according to the experts evaluation, because of its variability mechanisms, because of the visualization of the applied variabilities to process elements. In addition, the experts judged that the four variability types are not enough. It can be noted in the expert #10 statement "*...as I have knowledge of other approaches for variability representation in software product lines, I understand that different variability mechanisms could be added to the EPF Composer. Thus, I judge such a set of mechanisms insufficient...*", and in the statement of the expert #9 "*...some new mechanisms of variability types are necessary as, once there is the extend mechanism, it must exist include, mandatory and optional mechanisms...*". In the annotative approach variability mechanisms had more positive evaluations as such an approach has specific stereotypes for representing variability of each process element type. Exemplary statements excerpts from experts are: expert #1 stated that "*...I believe the variability mechanisms are sufficient as stereotypes allow the specification of different variability types...*"; expert #3 said "*...I understand the SMartySPEM variability mechanisms are sufficient to significantly demonstrate the types of relationships between a variation point and variants...*"; and expert #10 said "*...I consider the variability mechanisms adequate for representing mandatory, alternative and optional elements. Furthermore, the mechanisms allow clearly visualize where variation points and variants take place in a non-ambiguous basis for those who use process models...*".

## 4.4 Validity Evaluation

Results validity evaluation is an essential issue of empirical studies (Wohlin et al., 2000). We discuss the main threats relevant to our study, as follows:

**Internal Validity.** Tasks performed by experts were conducted in a similar manner except the order of the application of the study objects and questionnaires, which were random. Experts were trained on the basics of SPrLS and variability in compositional and annotative approaches using EPF Composer and SMartySPEM. We reduced the fatigue effects allowing the experts to answer the questionnaires in at most fifteen days;

**External Validity.** Features related to Requirement Specification of the OpenUP-based SPrL were used during both the training sessions and the empirical study. This could jeopardize the external validity, thus we tried to use original and technical documents of the OpenUP;

**Conclusion Validity.** The major threat to conclusion is related to the sample size, eleven experts. However, prior knowledge of such experts is significant. Therefore, we understand that for a qualitative study in which grounded theory procedures, such as Coding, were established eleven experts is a satisfactory number;

**Contruct Validity.** This study was planned based on a pilot project carried out for evaluating the instrumentation and its duration time for the application of questionnaire. Although the knowledge level required on software process and variability is essential, participants presented a high level of expertise.

## 5 RELATED WORK

Compositional and annotative integration and/or comparative studies have been carried out in the literature for several different domains, such as embedded systems and software product lines (Kästner and Apel, 2008; Ferreira Filho et al., 2013; Behringer, 2014). For software process lines or process tailoring/customization there is a lack of such a study type.

The study of Aleixo et al. (Aleixo et al., 012a) is the most direct related work to ours in the literature, in which they empirically compare variability capabilities in compositional and annotative approaches for SPrL based on EPF Composer and GenArch-P. GenArch-P is a model-driven approach to managing and customizing software process variabilities proposed in (Aleixo et al., 2010). The comparison is based on the same criteria adopted in our study, except that we discarded the uniformity criterion. As in our study, Aleixo et al. come up with better results for the annotative approach.

## 6 CONCLUSION

This paper presented an empirical qualitative study comparing the representation of variability in compositional and annotative approaches. Such a study provided, although initial, evidence that the annotative approach, in this study represented by SMartySPEM, has more advantages over the compositional approach, represented by EPF Composer. Although the criteria of modularity and detection errors had lower results in the annotative approach, they might be improved by using UML packages for modularity and applying inspection activities for error detection such as in (Geraldi et al., 2015).

As future work, we intend to plan and conduct empirical quantitative studies in order to compare our annotative approach, the SMartySPEM, to other compositional and annotative approaches. In addition, we are working on the establishment of a Scrum-based SPrL by taking real projects experience from industry as well as practitioners as Scrum Masters expertise as there is no real SPrL available in the literature for carrying out empirical studies and evaluating our SPrL-related theories and tools.

As a potential future work, we are considering studying the granularity criterion on the specification of lower-level software process activities, such as, in business process models, allowing one to customize the steps of the activities as, for instance, in multitenancy architectures of Software as a System (SaaS).

## ACKNOWLEDGEMENTS

The authors would like to thank Masters and Ph.D. lecturers an practitioners experts for attending this study and for their valuable contribution on assessing the compositional and annotative approaches.

## REFERENCES

Aleixo, F., Freire, M., Santos, W., and Kulesza, U. (2010). A Model-driven Approach to Managing and Customizing Software Process Variabilities. In *International Conference on Enterprise Information Systems*, pages 92–100. SCITEPRESS.

Aleixo, F. A., Freire, M., Alencar, D., Campos, E., and Kulesza, U. (2012a). A Comparative Study of Compositional and Annotative Modelling Approaches for Software Process Lines. In *Brazilian Symposium on Software Engineering*, pages 51–60.

Aleixo, F. A., Freire, M. A., Santos, W. C., and Kulesza, U. (2011). Automating the Variability Management, Customization and Deployment of Software Processes: a Model-Driven Approach. In Filipe, J. and Cordeiro, J., editors, *Enterprise Information Systems*, volume 73 of *Lecture Notes in Business Information Processing*, pages 372–387. Springer Berlin Heidelberg.

Aleixo, F. A., Kulesza, U., Freire, M. A., da Costa, D. A., and Neto, E. C. (2012). Modularizing software process lines using model-driven approaches - a comparative study. In *International Conference on Enterprise Information Systems*, pages 120–125. SCITEPRESS.

Aleixo, F. A., Kulesza, U., and OliveiraJr, E. (2013). Modeling Variabilities from Software Process Lines with Compositional and Annotative Techniques: a Quantitative Study. In *International Conference on Product-Focused Software Development and Process Improvement*, pages 153–168.

Behringer, B. (2014). Integrating Approaches for Feature Implementation. In *ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 775–778, New York, NY, USA. ACM.

Carvalho, D. D., Chagas, L. F., Lima, A. M., and Reis, C. A. (2014). Software Process Lines: A Systematic Literature Review. In Mitasiunas, A., Rout, T., OConnor, R., and Dorling, A., editors, *Software Process Improvement and Capability Determination*, volume

477 of *Communications in Computer and Information Science*, pages 118–130. Springer International Publishing.

Chemuturi, M. K. and Cagley, T. M. (2010). *Mastering Software Project Management: Best Practices, Tools and Techniques*. J. Ross Publishing, Inc.

Corbin, J. M. and Strauss, A. L. (2008). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage Publications, Inc.

Ferreira Filho, J. a. B., Barais, O., Acher, M., Baudry, B., and Le Noir, J. (2013). Generating Counterexamples of Model-based Software Product Lines: An Exploratory Study. In *International Software Product Line Conference*, pages 72–81, New York, NY, USA. ACM.

Galster, M., Weyns, D., Tofan, D., Michalik, B., and Avgeriou, P. (2013). Variability in Software Systems A Systematic Literature Review,. *IEEE Transactions on Software Engineering*, pages 81–90.

Garcia-Borgonon, L., Barcelona, M. A., Garcia-Garcia, J. A., Alba, M., and Escalona, M. J. (2014). Software Process Modeling Languages: a Systematic Literature Review. *Information and Software Technology*, 56(2):103–116.

Geraldi, R. T., OliveiraJr, E., Conte, T. U., and Steinmacher, I. F. (2015). Checklist-based Inspection of SMarty Variability Models: Proposal and Empirical Feasibility Study. In *International Conference on Enterprise Information Systems*, pages 268–275. SCITEPRESS.

ISO (2012). ISO/IEC 15504-5:2012 Information technology – Process Assessment – Part 5: An Exemplar Software Life Cycle Process.

Kalus, G. and Kuhrmann, M. (2013). Criteria for Software Process Tailoring: A Systematic Review. In *International Conference on Software and System Process*, pages 171–180, New York, NY, USA. ACM.

Kästner, C. (2010). *Virtual Separation of Concerns: Toward Preprocessors 2.0*. PhD thesis, Otto von Guericke University Magdeburg.

Kästner, C. and Apel, S. (2008). Integrating Compositional and Annotative Approaches for Product Line Engineering. In *Workshop on Modularization, Composition and Generative Techniques for Product Line Engineering*, pages 35–40.

Kästner, C., Apel, S., and Kuhlemann, M. (2008). Granularity in Software Product Lines. In *International Conference on Software Engineering*, pages 311–320.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., and Irwin, J. (1997). Aspect-Oriented Programming. In *European Conference on Object-Oriented Programming*, pages 220–242. Springer Berlin Heidelberg.

Lee, H. and Kang, K. C. (2013). A Design Feature-based Approach to Deriving Program Code from Features: A Step Towards Feature-oriented Software Development. In *International Workshop on Variability Modelling of Software-intensive Systems*, pages 1–6, New York, NY, USA. ACM.

Linden, F. J. v. d., Schmid, K., and Rommes, E. (2007). *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Martínez-Ruiz, T., Münch, J., García, F., and Piattini, M. (2012). Requirements and Constructors for Tailoring Software Processes: a Systematic Literature Review. *Software Quality Control*, 20(1):229–260.

OliveiraJr, E., Gimenes, I. M. S., and Maldonado, J. C. (2010). Systematic Management of Variability in UML-based Software Product Lines. *Journal of Universal Computer Science*, 16:2374–2393.

OliveiraJr, E., Pazin, M. G., Gimenes, I. M. S., Kulesza, U., and Aleixo, F. A. (2013). SMartySPEM: a SPEM-based Approach for Variability Management in Software Process Lines. In *International Conference on Product-Focused Software Development and Process Improvement*, pages 169–183. Springer Berlin Heidelberg.

Rombach, D. (2005). Integrated Software Process and Product Lines. In *International Conference on Unifying the Software Process Spectrum*, pages 83–90. Springer-Verlag Berlin.

Sommerville, I. (2015). *Software Engineering*. Pearson, 10 edition.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in Software Engineering: an Introduction*. Kluwer Academic Publishers, Norwell, MA, USA.