Evaluating scenario-based SPL requirements approaches: the case for modularity, stability and expressiveness

Mauricio Alférez, Rodrigo Bonifácio, Leopoldo Teixeira, Paola Accioly, Uirá Kulesza, Ana Moreira, Joao Araujo, Paulo Borba

Evaluating scenario-based SPL requirements approaches: the case for modularity, stability and expressiveness

Mauricio Alférez · Rodrigo Bonifácio · Leopoldo Teixeira · Paola Accioly · Uirá Kulesza · Ana Moreira · João Araújo · Paulo Borba

Abstract Software product lines (SPL) provide support for productivity gains through systematic reuse. Among the various quality attributes supporting these goals, modularity, stability and expressiveness of feature specifications, their composition and configuration knowledge emerge as strategic values in modern software development paradigms. This paper presents a metric-based evaluation aiming at assessing how well the chosen qualities are supported by scenario-based SPL requirements approaches. The selected approaches for this study span from type of notation (textual or graphical based), style to support variability (annotation or composition based), and specification expressiveness. They are compared using the metrics developed in a set of releases from an exemplar case study. Our major findings indicate that composition-based approaches have greater potential to support modularity and stability, and that quantification mechanisms simplify and increase expressiveness of configuration knowledge and composition specifications.

Keywords Software product lines · Variability modeling · Use scenarios · Requirements specification

1 Introduction

The continuous globalization trend is pressuring software intensive organizations to explore efficient ways to provide high-quality products of increasing size and complexity, customized to the particular needs of individual customers of market segments. Software product lines (SPL) [10, 26, 28] aim at addressing this challenge based on the observation that (1) most products in a market segment or application domain are not new, (2) products usually share many common features and (3) most organizations build software systems in a particular domain, repeatedly releasing product variants by removing features or adding new features.

SPL takes into account these three insights to provide an approach to support evolution of its features, and to

M. Alférez (✉)
Diverse team, Inria, Rennes, France
e-mail: mauricio.alferez@inria.fr

M. Alférez · A. Moreira · J. Araújo
Computer Science Department / CITI,
Universidade Nova de Lisboa, Caparica, Portugal
e-mail: amm@fct.unl.pt

J. Araújo
e-mail: joao.araujo@fct.unl.pt

R. Bonifácio
Computer Science Department, University of Brasília,
Brasília, Brazil
e-mail: rbonifacio@cic.unb.br

L. Teixeira · P. Accioly · P. Borba
Informatics Center, Federal University of Pernambuco,
Recife, Brazil
e-mail: lmt@cin.ufpe.br

P. Accioly
e-mail: prga@cin.ufpe.br

P. Borba
e-mail: phmb@cin.ufpe.br

U. Kulesza
Department of Informatics and Applied Mathematics (DIMAp),
Federal University of Rio Grande do Norte, Natal, Brazil
e-mail: uira@dimap.ufrn.br

increase software productivity through systematic reuse of its core assets that have been proactively planned with respect to expected future requirements [28]. Productivity reduces the effort and cost required to develop, deploy and maintain a collection of similar software products. Thus, SPL manage variability and commonality by supporting the overall process of identifying common and variable features, and managing the set of products' configurations. Although most work on variability focuses on the design and code levels, variability management for requirements engineering is essential to support both requirements reuse [5] and a seamless end-to-end development approach.

It is therefore noteworthy to understand how well SPL requirements approaches support reuse and evolution. Among the various quality attributes that promote these goals, modularity, stability and expressiveness of feature specifications, their composition and configuration knowledge emerge as key in modern development. Modularity is concerned with guaranteeing that features are encapsulated in separate modules. Stability, on the other hand, is the ability of the software to minimize unexpected effects from modifications of the software, as well as the management of the changes required to evolve it from one configuration to another. Finally, expressiveness is related to how easy a specification is written and read, or understood.

The main contribution of this paper is to present a metric-based comparative study of existing scenario-based SPL requirements approaches in order to understand how they address quality attributes, specifically modularity, stability and expressiveness, when modeling and evolving SPL requirements specifications. Although our metrics suite is partially adapted from existing metrics [13, 9], it also defines new metrics for quantifying expressiveness and stability of the configuration knowledge defined as the mapping relationships between SPL features and other artifacts, e.g., scenarios. Modularity will be measured by the degree of scattering, tangling and focus of specifications. Stability will be measured by the changes required to evolve an SPL (the specifications, compositions and configuration knowledge) from one configuration to another. Expressiveness will be measured by quantifying how verbose are the specifications, compositions and configuration knowledge, as well as the changes on these.

We have chosen four representative scenario-based SPL approaches for our evaluation because use scenarios are a recurrent technique in both requirements engineering and SPL requirements approaches. Use scenarios [1] are important to understand SPL features. A "use scenario", "usage scenario", or just "scenario" describes a real-world example of how one or more actors (e.g., users, developers, domain experts, organizations) interact with a system, describing the steps (i.e., events or actions) that occur during the interaction [1]. They provide examples of the system usage to both design and subsequent usability testing [1]. Scenarios can be composed according to specific combinations of features, and therefore, they are also useful to specify the intended behavior of target products of an SPL [14].

The chosen approaches, PLUSS [14], Model Templates [11], MSVCM [7] and VML4RE [3, 29], address different notations (textual or graphical based) and different variability representation mechanisms (compositional or annotative). They are compared using the metrics suite applied to a set of releases from an exemplar case study: the car crash crisis management system SPL [21] (Sect. 2). Although some of these approaches have been evaluated before, the existing empirical studies only consider a small number of approaches [15] or do not take into consideration an extensive number of quality attributes such as modularity, stability and expressiveness [7].

The remaining of this paper is structured as follows. Section 2 gives an overview of the SPL case study that we use to illustrate our evaluation. Section 3 introduces the four SPL requirements approaches chosen. Section 4 presents the study settings and a metrics suite we developed to compare the SPL requirements approaches. Sections 5–7 discuss the design issues of the evaluated approaches for specifying variabilities in SPLs. The result of this study concludes that aspect-oriented approaches reduce scattering of features and tangling of scenarios (Sect. 5), and improve expressiveness (Sect. 6) and stability (Sect. 7) of the specifications. Section 8 presents a summary of our findings, and Sect. 9 discusses potential threats to the validity of our study. Finally, Sect. 10 relates our work with other research topics, and Sect. 11 presents our concluding remarks.

2 Overview of the car crash crisis management system

The car crash crisis management system (CCCMS) was proposed as a benchmark to evaluate aspect-oriented modeling approaches in 2010 [20] and has been used by the modeling community for the evaluation and comparison of modeling approaches (e.g., the CMA workshop series[1]). The requirements used to create this exemplar were based on the real requirements document for crisis management systems created by Optimal Security.[2]

[1] http://cserg0.site.uottawa.ca/cma2013re/, or http://cserg0.site.uottawa.ca/cma2013models/.

[2] http://www.cs.colostate.edu/remodd/v1/sites/default/files/cms_case_study.pdf.

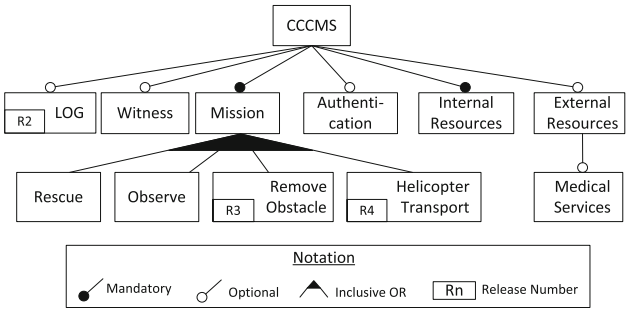

Fig. 1 Partial feature model for the CCCMS SPL

A crisis is an unpredictable situation that can lead to severe consequences if not dealt with quickly. A car crash crisis management system supports the process of identifying, assessing and handling the crisis situation by orchestrating the communication between all parties involved in handling the crisis, allocating resources and providing information to determined users. Any car crash crisis management system has a common set of responsibilities and functionalities. It is therefore natural to build an SPL of car crash crisis management systems, which can be specialized to create a particular kind of crisis to a particular context. The CCCMS involves single or multiple vehicles, and is limited to the management of human victims.

The models of the CCCMS case study are public and described in detail in [20, 25] (although many have been collected by the CMA workshop organizers and kept publicly at ReMoDD[3]). The models that served as basis for our study are the feature model, the use scenario textual descriptions, the textual requirements and a domain model. Feature models depict a hierarchical decomposition of features with mandatory (must have if all its ancestors are selected), inclusive (selection of one or more), exclusive (selection of only one) and optional (may or may not have) relationships between features. A domain model consists of class diagrams that show the relationships between the main concepts of a domain.

The feature model is the model more directly related to the SPL configurability. It identifies and relates common and variable features between the products in the domain of the SPL. Figure 1 shows the part of the feature model[4] of the CCCMS SPL that we considered in our evaluation, where 240 possible different combinations of features can be chosen. The small boxes on the bottom left of some of the features in this figure represent the identification of the release (R2, R3 and R4) that introduced the corresponding feature in the system.

To resolve a crisis, the coordinator requests the employees and external resources to execute appropriate missions. A crisis is triggered when a witness places a call to the crisis center and is answered by a coordinator. The coordinator captures the witness report in the system (Witness feature), which recommends to the coordinator the missions that have to be executed based on the current information about the crisis and resources (Mission feature). The coordinator selects one or more missions recommended by the system—for example, rescue (Rescue feature), observe (Observe feature), order a helicopter transport (Helicopter Transport feature), or remove obstacles (Remove Obstacle feature); these missions can be executed more than once and in parallel, if necessary. Depending on the mission, the coordinator assigns internal resources (Internal Resources feature) or requests external ones (External Resources feature) to fulfill the mission. After communication between the resources and the coordinator, other information and new missions or resources can be called. Finally, all resources submit a final mission report so that the coordinator can finalize the crisis resolution process.

During missions, if medical services (Medical Services feature) are available in the system, CCCMS employees (such as first-aid workers) can ask the system for the victim's medical history information relevant to his injury from all connected hospital resource systems. Also, if available in the CCCMS, it will be possible to log all processes and decisions taken (Log feature). Similarly, when assigning internal resources or taking important decisions, authentication is necessary in the products that make it available (Authentication feature).

More details on the specification and evolution of the CCCMS will be presented in Sect. 4.

3 Overview of the evaluated approaches

The four approaches we have selected for this study, PLUSS [14], Model Templates [11], MSVCM [7] and VML4RE [3, 29], range from textual to graphical, and compositional to annotative techniques. They are representative from recently proposed SPL requirements approaches, some being the evolution of several others. For example, approaches such as PLUC [6] and PLUS [18] have inspired other works, but they were not considered in this study because more recent techniques, such as PLUSS and Model Templates, build on them and provide more details for their application. The chosen approaches adopt different composition and annotation mechanisms, what makes their comparison more interesting in the context of the three quality attributes (modularity, stability and expressiveness) we plan to evaluate.

[3] http://www.cs.colostate.edu/remodd/v1/.

[4] An editable feature model and statistics are available at http://www.splot-research.org/.

Table 1 PLUSS specification of execute rescue mission

Scenario: Execute rescue mission (SC07)

Description: The intention of the first-aid worker is to accept and then execute a rescue mission that involves transporting a victim to the most appropriate hospital

Related feature: Rescue mission

Flow of events:

Code	Related feature	User actions	System responses
1	–		System updates crisis record with the sent injury information
2	Medical services	First-aid worker determines victim's identity and communicates it to system	System requests victim's medical history from all connected Hospital resource systems
3	Medical services	Hospital resource system transmits victim's medical information to system	System notifies first-aid worker of medical history of the victim, which is relevant for his injury
4	–	–	System instructs first-aid worker to bring the victim to the most appropriate hospital
5	–	First-aid worker notifies system that he has dropped off the victim at the hospital	–
6	–	First-aid worker informs system that he has completed his mission	–

Thus, each approach represents distinguishable breeds of work, according to the variability representation style and the specification notation. For example, MSVCM and VML4RE are compositional-based approaches (in SPL terminology [19]), applying aspect-oriented techniques to model scenario variability. These approaches use independent models to express the "configuration knowledge" [12], which is the relationships between SPL features and specific fragments of the scenarios.

The other two approaches, Model Templates and PLUSS, are annotation based [19] and differ from the previous in determining which and how specific parts of the scenarios are composed according to specific selections of features. Variability can be defined through annotations, often specified using either a mapping table that relates specific parts of the scenarios to SPL features, or UML stereotypes, or notes with the feature name inserted on specific elements of the scenarios specifications. These annotations, placed throughout the specifications, determine which fragments of the scenario specifications are related to features of the SPL. Hence, annotation-based approaches do not separate common and variable scenario specifications nor do they use a dedicated configuration knowledge model to indicate how to compose scenarios according to specific feature selections.

There are different ways to express scenarios. For example, black-box textual notations describe and relate actor inputs and system responses into two columns of tables, and UML activity diagrams provide a different concrete visual syntax. PLUSS and MSVCM specify variabilities in textual scenarios, whereas Model Templates and VML4RE specify variabilities in scenarios using a graphical representation of the requirements, in particular, activity diagrams.

The four approaches are described systematically, according to three characteristics: variability representation, composition process (how to derive a product of an SPL) and the following different types of requirements variability [4]:

- Variability in function: occurs when a particular function (detailed as scenarios) might exist in some products but not in others.
- Variability in control flow: occurs when a pattern of user–system interaction within a scenario varies from one product to another.
- Variability in data: corresponds to fine-grained variations and occurs when two or more scenarios share the same behavior and differ with respect to the values of a specific concept.

3.1 PLUSS

PLUSS is a domain approach to manage variant behavior in use case models[5] [14]. PLUSS consists of a customization for feature modeling and a particular notation for specifying variant behavior in textual scenarios. These scenarios detail the whole use case model of an SPL. Its characteristics are discussed next.

3.1.1 Variability representation mechanisms

Choosing a feature for a product of an SPL might trigger the selection of a complete scenario or some steps in a scenario, or even a use case that encompasses several scenarios. Therefore, to allow the representation of variabilities, existing use cases, scenarios and individual steps in the scenario requirements specification must be related manually to features in the feature model. This kind of

[5] Use scenarios describe a single path of logic, whereas use cases typically describe several paths (usually the basic flow plus alternate paths).

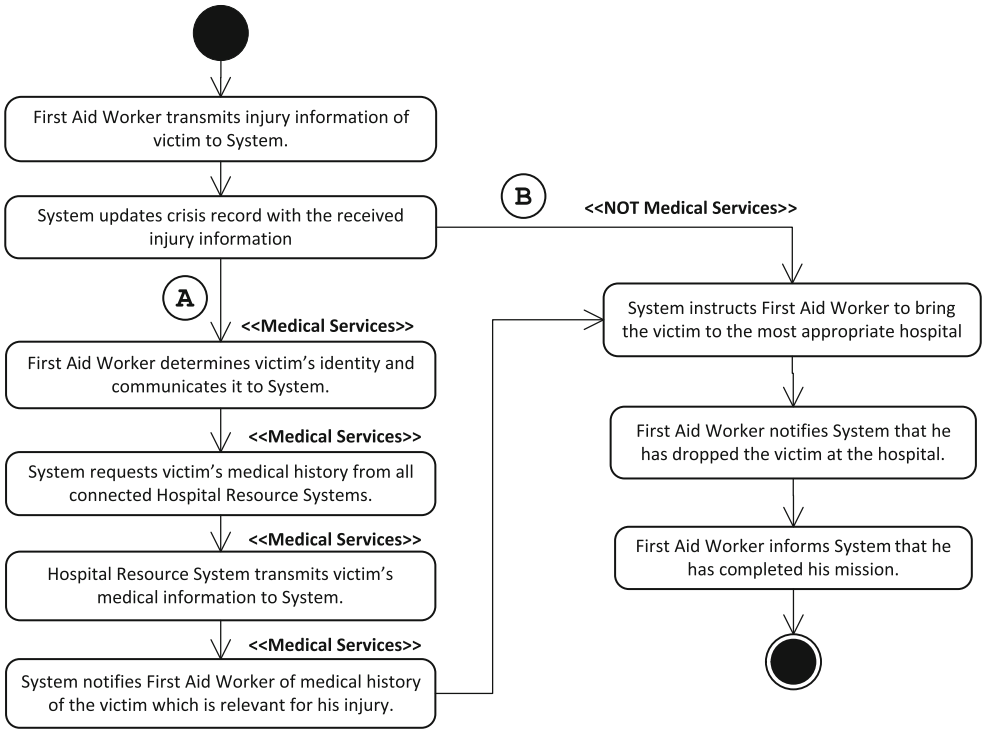

Fig. 2 Model template specification of execute rescue mission

relationship between feature and parts of the requirements specification is named "require". Table 1 shows an example of these annotations to represent variability in steps; here, the "execute rescue mission" scenario is related to the Rescue Mission feature. Also, steps 2 and 3 are related with the Medical Services feature. The remaining steps are commonalities. As such, we do not associate them with any particular feature. Data variability, or variability in values such as numbers or names used along the specification, is supported using parameters. Each parameter is related to an alternative or optional feature in the feature model. This kind of relationship is named "instantiate". When deriving an SPL member specification, the value assigned to a parameter corresponds to the selected subfeature(s) that the parameter refers to.

3.1.2 Composition process

The process of deriving products (1) filters optional use cases, scenarios and steps related to features not selected in a specific product and (2) assigns the selected features to the related parameters. It is important to emphasize that domain engineers have to annotate the specification detailing which features are related to which use cases, scenarios, steps and parameters. Thus, there is no independent model relating the use case model to features.

In our comparative study, we annotate optional scenarios and steps to indicate their dependencies with specific features. For instance, Table 1 shows that the "execute rescue mission" scenario requires the Rescue Mission feature. In other words, this scenario is only present in products configured with the Rescue Mission feature. Moreover, steps 2 and 3 in Table 1 are also optional. They are only present in products that are configured with the Medical Services feature.

3.1.3 Supported variability

PLUSS supports the three types of variability: function, control flow and data. Variability in function and control flow is made possible by relating entire scenarios and specific scenario steps, respectively, to features in the feature model. However, it is limited to only one feature by scenario or step. Variability in data is possible by adding parameters in the specification that are then instantiated according to the feature selection.

3.2 Model templates

Model Templates (MTs) use activity diagrams to specify scenarios. A model template is an annotated model expressed in the target notation defined by a metamodel [11]. Thus, a model template could be specified either

Table 2 MSVCM specification of execute rescue mission
Scenario: Execute rescue mission (SC07)
Description: The intention of the first-aid worker is to accept and then execute a rescue mission that involves transporting a victim to the most appropriate hospital
Flow of events:

Code	User actions	System responses
SC07.1	First-aid worker transmits injury information of victims to system	System updates crisis record with the sent injury information `@InjuryData`
SC07.2	–	System instructs first-aid worker to bring the victim to the most appropriate hospital
SC07.3	First-aid worker notifies system that he has dropped off the victim at the hospital	–
SC07.4	First-aid worker informs system that he has completed his mission	–

Table 3 MSVCM specification of the advice for medical services ADV01
Advice: Medical service advising execute rescue mission
Description: Transmits injury information of victim to system
Pointcut: `@InjuryData`
Flow of events:

Code	User action	System response
ADV01.1	First-aid worker determines victim's identity and communicates it to system	System requests victim's medical history information from all connected hospital resource systems
ADV01.2	Hospital resource system transmits victim's medical history information to system	System notifies first-aid worker of medical history of the victim relevant to his injury

using UML diagrams or any other domain-specific notation defined using Meta Object Facility (MOF).

3.2.1 Variability representation mechanisms

Domain engineers have to relate model elements to features, in order to compose specific scenarios for an SPL product. In case of activity diagrams, for example, the model elements that can be related to features are actions, transitions flows, start and final nodes. However, differently from PLUSS, which relates each individual asset to one specific feature, this relationship is more expressive in Model Templates, since model elements can be related to feature expressions, represented as propositional formulas involving features. For example, Fig. 2 shows two examples of propositional formulas: `<<medical services>>` and `<<not medical services>>`. The use of feature expressions increases expressiveness because it avoids the need of polluting feature models with the introduction of artificial features, such as *not medical services*.

Table 4 MSVCM configuration knowledge for release 1 of the CCCMS SPL

Expression	Transformations
CCCMS	select scenario SC01, SC03, SC04
Authentication system	select scenario SC10
Rescue mission	selectscenario SC07, SC08
Witness	select scenario SC02
Remove obstacle mission	select scenario SC09
Medical services	evaluate advice ADV01
Observe mission	select scenario SC06

3.2.2 Composition process

Composition of scenarios is based on implicit or explicit mechanisms to keep or remove model elements in the activity diagram. The explicit mechanism occurs when parts of the model are included in a product specification because they are related to a feature expression that is satisfied by the product configuration. For example, in Fig. 2, the activity "system requests victim's medical history from all connected hospital resource systems" will be kept in all the SPL products that contain the Medical Services feature. Differently, the implicit mechanism occurs when dependencies among model elements are not satisfied. For instance, if one transition points to one activity that is not selected for a specific product, it will be implicitly removed from the product specification.

The model template that specifies the "execute rescue mission" use case is shown in Fig. 2. This model template is based on annotated activity diagrams, as discussed in [11]. Note that, some activities and transitions between activities are annotated with the Medical Services stereotype, as well as a transition labeled with "A" in the diagram of Fig. 2. These elements only appear in products configured with the Medical Services feature. Similarly, the transition labeled with "B" has the "NOT Medical Services" stereotype, stating that it will be present only if a product is configured without the Medical Services feature. Therefore, similarly to PLUSS, Model Templates scatter the configuration knowledge concern, represented by annotated feature expressions throughout the requirements models, and also presents tangling of different concerns (i.e., features) inside some models. Furthermore, because of the complexity of maintaining a generally large model

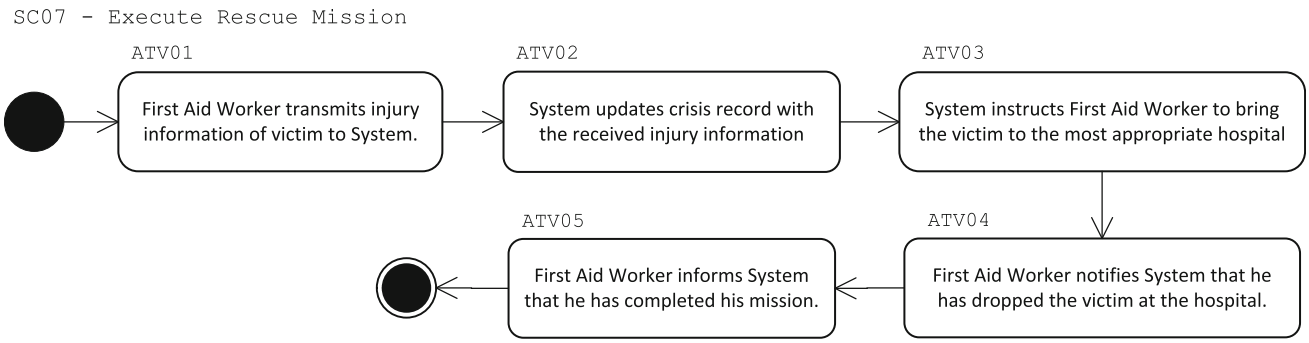

Fig. 3 VML4RE specification of execute rescue mission

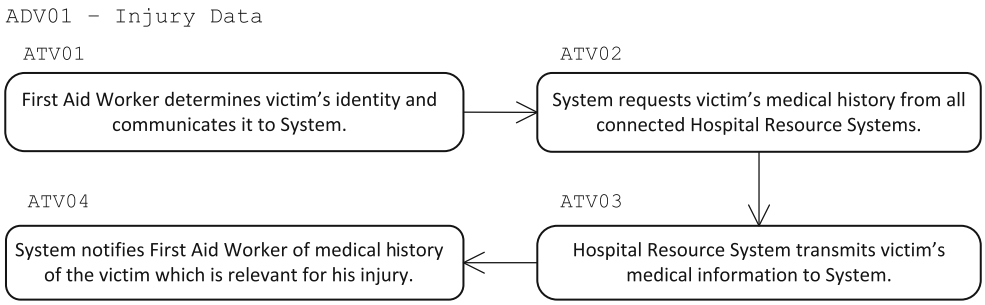

Fig. 4 VML4RE specification of the injury data advice

of the overall system, it is difficult to distinguish and maintain possible alternative flows and configure appropriately their related feature expressions. In our example of Fig. 2, if we add only one optional feature, we would almost duplicate the number of activities and would also use the same stereotypes repetitively inside the specification. We will discuss these problems further in Sects. 5 and 6.

3.2.3 Supported variability

Model Templates support variability in function and variability in control flow. Variability in function occurs when a whole scenario is annotated with a feature expression, whereas variability in control flow occurs when a specific activity is annotated with a feature expression. As discussed, this is not limited to mapping a scenario (or activity) to one single feature; instead, it allows a "many-to-many" mapping between model elements and features. It does not support variability in data.

3.3 MSVCM

Similar to PLUSS, modeling scenario variability as crosscutting mechanisms (MSVCM) is an approach to manage variant behavior using textual scenarios [7]. However, it has explicit and separated mechanisms to define variability and express configuration knowledge.

```
1.  Variant for (Medical Services) {
2.          merge ( "SC07" , "ADV01" );
3.          connect ( "SC07:: ATV02" , "SC07::ADV01_ATV01" );
4.          connect ( "SC07:: ADV01_ATV04" , "SC07:: ATV03" );
5.          removeControlFlow ( "SC07::ATV02" , "SC07::ATV03" );
6.  }
```

Fig. 5 VML4RE code snippet to insert injury data advice

3.3.1 Variability representation mechanisms

To deal with variabilities between instances of a same scenario, MSVCM proposes new constructs to describe use cases: aspectual, or crosscutting, use cases and parameters. Using aspectual use cases allows changing the behavior (represented as a sequence of steps) of an existing scenario. Scenario parameterization allows the configuration of scenarios that differ according to values in a specific domain.

3.3.2 Composition process

MSVCM aims at separating common from variant behavior. For instance, the scenario in Table 2 details the behavior required by the rescue mission feature. There is no step in this scenario related to the medical services feature, differently from the PLUSS specification depicted in Table 1. In MSVCM, the specification of the interaction between the aforementioned features can be modularized

Table 5 Summary of the main characteristics of the techniques

Technique	Dominant Notation	Var. Mechanism	Supported Var.
MSVCM	Textual	Compositional-Annotative	F/CF/D
PLUSS	Textual	Annotative	F/CF/D
VML4RE	Graphical	Compositional	F/CF/D
MT	Graphical	Annotative	F/CF

(F) Function, (CF) control flow, and (D) data

as an advice (see Table 3). Note that, in MSVCM, scenarios and advices do not make explicit references to features. Actually, an independent model, named configuration knowledge, is responsible for relating scenarios and advices to features. More precisely, a configuration model relates feature expressions to transformations that translate SPL assets into product-specific artifacts. If a feature expression evaluates to True for a given product, the related transformations are applied.

Three different kinds of transformations are supported: (1) select scenario (includes a given scenario in the final product), (2) evaluate advice (composes an advice through matched join points) and (3) bind parameter (replaces parameterized textual sentences by feature data). Therefore, the configuration knowledge of Table 4 covers the configurability of the first release of the CCCMS SPL. Note that, aspectual use cases in MSVCM support quantification. For instance, the advice ADV01 quantifies overall steps assigned to `@InjuryData` (see the *pointcut* clause of ADV01), which, differently from Model Templates, does not require any transformation related to the *not medical services* feature expression.

3.3.3 Supported variability

MSVCM transformations support the three types of variability previously discussed (variability in function, data and control flow). For instance, the *select scenario* transformation supports variability in function, allowing us to select specific scenarios for a given product configuration. Differently, the *evaluate advice* transformation deals with variability in control flow (as required by the medical services feature that changes the behavior of the execute rescue mission use case). Finally, the *bind parameter* transformation deals with variability in data, mapping parameters within the specifications to specific data obtained from the feature configurations.

3.4 VML4RE

The Variability Modeling Language for Requirements (VML4RE) [3, 29] presents a solution for the composition of model fragments for requirements models of an SPL, which includes use case diagrams and their related scenarios represented by activity diagrams. This approach aims at specifying the composition of requirements models for specific products of an SPL using a separate composition model that contains transformations (named *actions*) specially tailored for scenarios.

3.4.1 Variability representation mechanisms

VML4RE provides a set of specialized operators for referencing and composing parts of the scenarios and use case model. It does not add annotations to the scenarios as it happens with Model Templates, and it does not use free-format textual descriptions that can be ambiguous, due to the interpretation of the natural language [26]. Similarly to MSVCM, it employs a separate model, called composition model, to specify configuration knowledge. The composition model allows the specification of transformations (actions), which are linked to feature expressions.

3.4.2 Composition process

VML4RE composes specific use case models and scenarios according to the actions expressed in the composition model. Actions are responsible for different kinds of modifications in the models, such as insert, connect, merge, remove and replace use cases, actors, packages, activity diagrams, steps and their relationships.

Figure 3 shows the scenario "execute rescue mission" (SC07) that details the behavior required by the Rescue Mission feature, and Fig. 4 shows the steps that will be merged to the "execute rescue mission" activity diagram. The VML4RE code snippet in Fig. 5 merges the "injury data" advice with the "execute rescue mission" scenario. This code snippet is part of the composition model that represents configuration knowledge, as it relates the Medical Services feature to some parts of the scenarios. The merge action (line 2) copies the elements referenced by the expression given in the second argument, the advice, to the model referenced by the expression given in the first argument, the base model. To avoid duplicated model elements identifiers, this operator adds a prefix (the name of the advice scenario) to the identifier of the new model elements. After copying the model elements, it is necessary to redirect the control flow to the steps included in the advice. This is done by adding new control flows (lines 3 and 4) and removing an unnecessary control flow (line 5).

3.4.3 Supported variability

In VML4RE, variability in function and control flow is possible using a separate configuration knowledge model,

Table 6 Metrics suite used in our study

Attribute	Metric
Modularity	Degree of scattering of features [13]
	Degree of focus of scenarios [13]
Stability of the specifications	Number of steps introduced or changed
	between two releases [9]
Stability of the Compositions	Number of compositions items introduced
	or changed between two releases [9]
Stability of the CK	Numberof configuration items introduced
	or changed between two releases
Expressiveness of the composition	The ratio between the number of composition
	items and number of matched join points [9]
Expressiveness of the CK	Number of tokens required to specify
	the configuration knowledge

which may include entire scenarios or specific scenarios steps, respectively, according to a feature expression. Variability in data is also possible replacing generic activities or steps by more specific activities.

3.5 Summary

Table 5 summarizes the main characteristics of the techniques explained. The second column refers to the notation used to model scenarios. PLUSS and MSVCM use textual scenarios descriptions following a blackbox format. On the other hand, Model Templates (MT) and VML4RE use UML activity diagrams, which employ a graphical notation.

The mechanisms to represent variability can be divided into two types: annotations and compositions. Annotation-based techniques introduce annotations on the scenarios to indicate variable parts. PLUSS and Model Templates keep or remove parts of the scenarios depending on the evaluation of their annotations according to specific product configurations.

Composition-based techniques model the variations as distinct modules and so, to generate the scenarios for an SPL member, there must be a composition of variable and common modules. VML4RE is a compositional approach because activity diagrams are composed to add, replace or remove parts of the initial base scenarios. MSVCM is considered to be both compositional and annotative. MSVCM is compositional because it uses different modules to represent commonality and variability (advices), but it also uses annotations in the base scenarios to show where the advices should be applied.

All the approaches support variability in function, control flow and data, except Model Templates that does not have specific mechanisms to instantiate data inside the scenarios based on specific feature configurations.

4 Study settings

This section presents detailed information about our comparative study by first discussing the phases and assessment procedures of the study, and then describing the metrics suite to quantify modularity, stability and expressiveness.

4.1 Study phases and assessment procedures

Our study was organized in three major phases:

1. Specification of the car crash crisis management system SPL (CCCMS) using the four chosen requirements approaches.
2. Evolution of the different specifications, to address change scenarios.
3. Quantitative assessments of the different specifications and releases of the CCCMS SPL.

In the first phase, the CCCMS SPL was specified using the different modularization and composition mechanisms available in the investigated requirements approaches. From the models detailed in [25], we have developed a set of incremental releases for the CCCMS (they are available online[6]). Considering the feature model shown in Fig. 1, we have defined the base release (R1), consisting of the features CCCMS, Authentication System, Rescue Mission, Witness, Medical Services, Internal and External Resources and Observe Mission. Then, in the second phase, all specifications were evolved to address the change scenarios corresponding to the releases R2–R4, which appear in Fig. 1. The features inserted in the releases required the introduction of new scenarios and changes to existing ones. For example, R2 introduced the Log feature, which affects two existing scenarios: "execute super observer mission" (SC06) and "execute rescue mission" (SC07). However, in the subsequent releases (R3 and R4), the new use cases introduced ("remove obstacle mission" and "helicopter transport mission") also had to deal with Log. These change scenarios allowed us to exercise the different modularization and composition mechanisms provided by each approach, to observe their modularity, stability and expressiveness. Finally, we applied our metrics suite (see Sect. 4.2), to analyze and compare the obtained results for the different specifications.

[6] http://people.irisa.fr/Edward-Mauricio.Alferez_Salinas/REJ/REJ-Data.htm.

All specifications were written according to alignment rules, which were necessary not only to verify that good practices were used in the approaches, but also to ensure that the comparison of the specifications was equitable and fair. Three researchers performed these alignment activities. All misalignments found were discussed between the study participants, and eventual corrections were applied to the specifications to guarantee their alignment. For example, we ensured that (1) every variability was modularized using the appropriate modularization and composition mechanisms of each approach, (2) textual and graphic-based approaches used an equal number of elements that represents the same abstraction (such as activities in VML4RE and Model Templates, and textual steps for the scenarios in MSVCM and PLUSS approaches) and (3) the specifications reflect the same functionalities/features and are consistent between them.

4.2 The metrics suite

Since we are interested in comparing different SPL requirements engineering approaches from the modularity, stability and expressiveness perspectives, we selected a metrics suite that allows the quantification of these attributes in the different specifications of the case study. Table 6 gives an overview of the metrics suite used in our study.

Our modularity investigation relies on two metrics that we have customized [7] from [13] *degree of scattering of features* (DoS) and *degree of tangling of scenarios* (DoT). According to equations (1) and (2), DoS quantifies the concentration of a feature over each scenario $s \in S$ (the set of scenario specifications). Values of DoS are normalized between 0 (completely localized) and 1 (completely scattered). The greater the DoS of a feature f is, the greater is the probability of reviewing different scenarios when the specification of f has to evolve. Note in Eq. (1) that $\mid S \mid$ denotes the cardinality of the set S.

$$DoS(f) = 1 - \frac{\mid S \mid \sum_{s \in S} \left(Conc(f,s) - \frac{1}{|S|} \right)^2}{\mid S \mid -1} \quad (1)$$

$$Conc(f,s) = \frac{number\ of\ steps\ in\ s\ assigned\ to\ f}{number\ of\ steps\ assigned\ to\ f} \quad (2)$$

Likewise, according to Eqs. (3) and (4), DoT considers how many steps of a scenario are related to each feature $f \in F$ (the set of features). Values of DoT are similarly normalized between 0 (completely focused) and 1 (completely tangled). The greater the DoT of a scenario s is, the greater the probability of reviewing s when one of the related features changes. We use the metric degree of focus (DoF) when presenting desired values in modularity. Therefore, DoF indicates the contrary of DoT and corresponds to $1 - DoT$. Thus, the lower the DoF of a scenario s is, the higher is the tangling (DoT) of features it specifies.

High values in the degree of focus and low values in the degree of scattering are usually associated to well-modularized systems [13, 23, 24].

$$DoT(s) = 1 - \frac{\mid F \mid \sum_{f \in F} \left(Dedi(s,f) - \frac{1}{|F|} \right)^2}{\mid F \mid -1} \quad (3)$$

$$Dedi(s,f) = \frac{number\ of\ steps\ in\ s\ assigned\ to\ f}{number\ of\ steps\ of\ s} \quad (4)$$

Note that in Eqs. (2) and (4) to evaluate these metrics, we have to assign features to the individual steps of a specification. We follow the *configuration dependency analysis* as a guide [7], considering that a step *st* depends on a feature f if, and only if, the selection of f triggers the

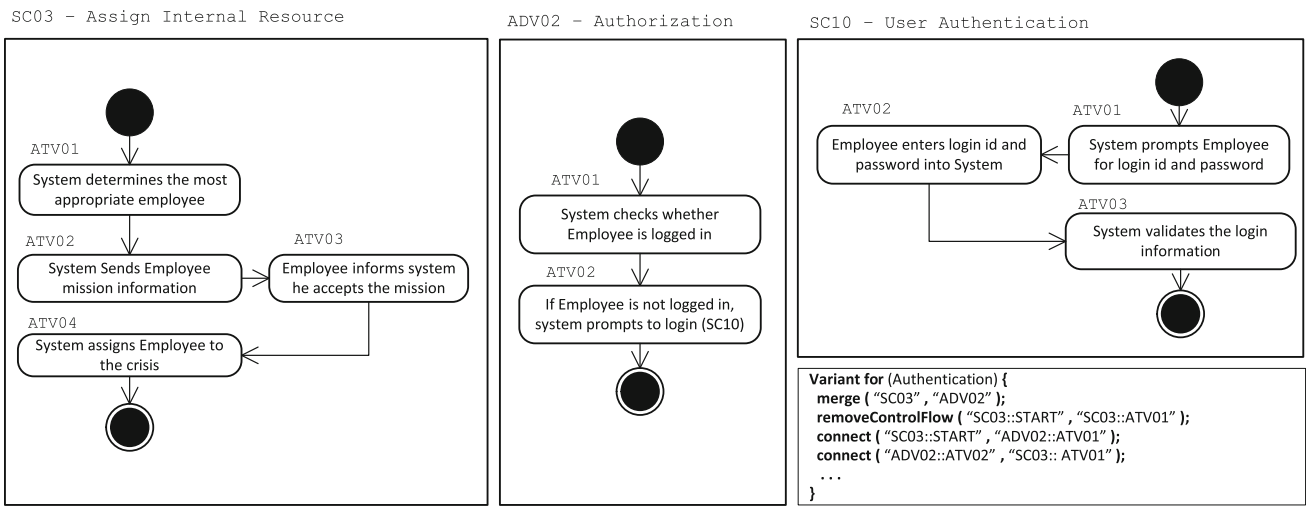

Fig. 6 Separation of the authentication feature into one authorization advice (ADV02) and one login scenario (SC10) in VML4RE. In this case, the authorization advice could be merged into different scenarios

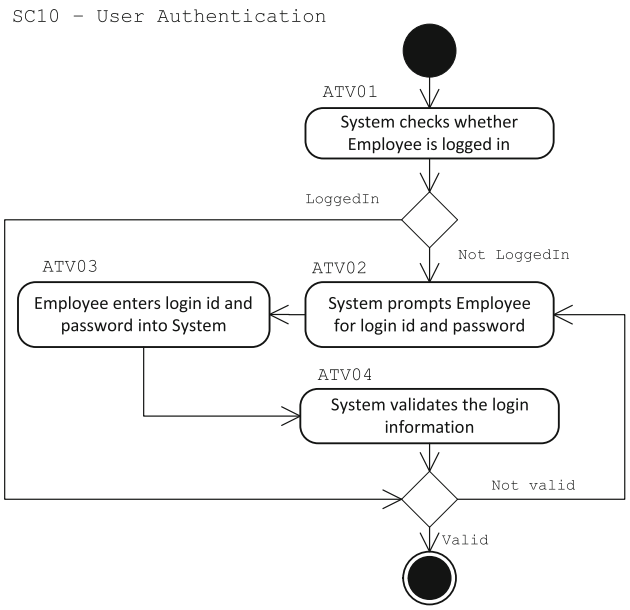

Fig. 7 In this version, both authorization and Login behavior are represented as a single scenario (SC10) in VML4RE. This version, although reducing the scattering of the authentication feature, compromises the reuse of specific steps related to authorization

configuration of scenario *s*. Similarly, we assigned the modeled activities to features in the Model Templates and VML4RE approaches.

Regarding our stability assessment, we adapted a metrics suite that has also been validated and used to compare semantic and syntactic approaches for aspect-oriented requirements engineering [9]. These metrics quantify the stability of specifications and code elements that represent a software artifact in the context of evolutionary scenarios. In our study, we used them to quantify the stability of the requirements specifications along the different releases of the CCCMS SPL. We measured the stability of the specifications and the stability of the compositions. They are quantified by the number of modified or introduced steps or scenarios, and the number of modified or introduced composition items between two subsequent releases. In addition, we have also proposed a metric that quantifies the stability of the configuration knowledge.

Finally, we investigate the expressiveness of the specifications. For measuring the expressiveness of the configuration knowledge, we count how many tokens are required to map features to other models. For example, in PLUSS, we have the "related feature" column that associates steps to features. For example, for the PLUSS specification of execute rescue mission shown in Table 1, we associate steps 2 and 3 with the medical services feature. Thus, the total count of tokens for this scenario is 4. We count tokens similarly for Model Templates, using the stereotype annotations. In the case of MSVCM and VML4RE, we count all tokens used in their respective dedicated configuration knowledge models (Sect. 6.1 provides concrete examples). Although the unit to quantify expressiveness of the configuration knowledge is rather low level, it allowed us to uniformly assess the different representations of the configuration knowledge. To measure expressiveness of the compositions, we use the notion of reachability [9], computing the ratio between the number of matched join points and the number of composition items. Finally, we only measure stability and expressiveness of compositions for the MSVCM and VML4RE compositional-based approaches. Since the other two approaches are annotation based, variant behavior is already composed into specifications.

5 Assessment of variability modularity

We chose to analyze modularity based on the concepts of scattering and tangling—scattering of features and tangling of scenario specifications. Although other characteristics typically related to modularity, particularly cohesion, could be considered to evaluate the degree of modularity of a specification, in this study we decided to address modularity from the aspect-oriented perspective. It is well known that aspect orientation offers a step-forward on modularization (with respect to classical software development approaches, such as structured or object oriented), by reducing scattering and tangling.

This section considers the different releases of the case study specified using each technique and uses metrics to quantify the degree of scattering of features (Sect. 5.1), and the degree of tangling and focus of specifications (Sect. 5.2).

5.1 Degree of scattering of features

The degree of scattering (DoS) of a feature quantifies to what extent the specification of a feature is disperse. In our study, most of the feature specifications are well localized, which led to specifications with low DoS. Actually, only two features present some scattering: Log and Authentication.

The specification of Log imposes an homogeneous behavior that scatters throughout all the use cases related to the Mission subfeature (see Fig. 1) in the case study. Differently, the Authentication feature requires two distinct procedures: (1) one related to the login behavior and (2) another that verifies if an employee had already been authenticated. Using the compositional approaches (VML4RE and MSVCM), we could modularize these procedures into independent assets (advices). Nevertheless, such decision leads to the scattering of the Authentication feature specification, which could also be realized in the annotative-based specifications. Figure 6 shows how we could separate these procedures using VML4RE. However,

Table 7 Assignment of features to the scenarios' steps in the first release

Feature/Scenario	SC01	SC02	SC03	SC04	SC06	SC07	SC10	ADV01	ADV02
(a) MSVCM									
CCCMS	8	–	–	–	–	–	–	–	–
Witness	–	4	–	–	–	–	–	–	–
Int.Resources	–	–	3	–	–	–	–	–	–
Authentication	–	–	–	–	–	–	2	–	2
Ext.Resources	–	–	–	2	–	–	–	–	–
Observe	–	–	–	–	5	–	–	–	–
Rescue	–	–	–	–	–	4	–	–	–
Med.Services	–	–	–	–	–	–	–	2	–

Feature/Scenario	SC01	SC02	SC03	SC04	SC06	SC07	SC10	ADV01
(b) VML4RE								
CCCMS	14	–	–	–	–	–	–	–
Witness	–	8	–	–	–	–	–	–
Int.Resources	–	–	4	–	–	–	–	–
Authentication	–	–	–	–	–	–	3	–
Ext.Resources	–	–	–	3	–	–	–	–
Observe	–	–	–	–	8	–	–	–
Rescue	–	–	–	–	–	5	–	–
Med.Services	–	–	–	–	–	–	–	4

Feature/Scenario	SC01	SC02	SC03	SC04	SC06	SC07	SC10
(c) PLUSS							
CCCMS	8	–	–	–	–	–	–
Witness	–	4	–	–	–	–	–
Int.Resources	–	–	3	–	–	–	–
Authentication	–	–	2	–	–	–	2
Ext.Resources	–	–	–	2	–	–	–
Observe	–	–	–	–	5	–	–
Rescue	–	–	–	–	–	4	–
Med.Services	–	–	–	–	–	2	–

Feature/Scenario	SC01	SC02	SC03	SC04	SC06	SC07	SC10
(d) Model templates							
CCCMS	14	–	–	–	–	–	–
Witness	–	8	–	–	–	–	–
Int.Resources	–	–	4	–	–	–	–
Authentication	–	–	2	–	–	–	3
Ext.Resources	–	–	–	3	–	–	–
Observe	–	–	–	–	8	–	–
Rescue	–	–	–	–	–	5	–
Med.Services	–	–	–	–	–	4	–

after considering other factors, such as the growth of the configuration knowledge, we decided to merge the complete specification of the Authentication feature in VML4RE (see Fig. 7). This was a controversial decision, since merging both procedures in a single asset eliminates the Authentication feature scattering, even though it increases the coupling between the mentioned procedures, which hampers the possibilities to reuse the specific steps of the authorization procedure in other scenarios. Due to the small overhead on the configuration knowledge, we

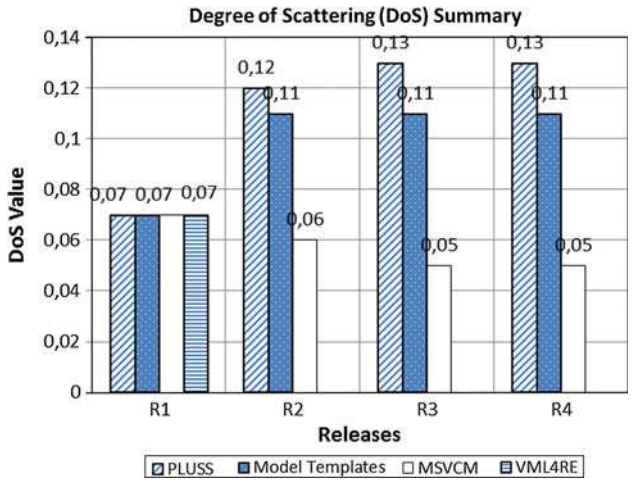

Fig. 8 Average degree of scattering

decided to specify the Authentication feature in MSVCM using a decomposition such as depicted in Fig. 6.

For the first release (R1), Table 7 details the results of the feature assignment process, which relates features to the scenarios (and advice) steps. As explained in Sect. 4, the Log feature was not detailed in the first release of the CCCMS specifications. In that case, only the authentication feature presents some scattering, leading to a DoS of 0.56 in MSVCM, Model Templates and PLUSS specifications. Considering the other features that were well modularized in the first version (leading to a DoS of zero), the resulting average DoS of those techniques was 0.07 (see Fig. 8). Since we merged the specifications of the Authentication feature in VML4RE, all features were fully modularized in VML4RE, leading to an average degree of scattering of zero.

Tables 8 and 9 summarize the assignment of features to the specifications' steps of the last release. Since release R2, we could modularize the Log specification using both MSVCM and VML4RE. For this reason, introducing new scenarios in the later releases (R3–R4) of MSVCM reduced the average degree of scattering (leading to an average DoS of 0,05). Differently, the Log specifications increased the average degree of scattering in PLUSS as well as in Model Templates (DoS of 0,13 and 0,11 respectively). Actually, the DoS was even higher in PLUSS because the complete

Table 8 Assignment of features to the scenarios' steps in the fourth release

Feature/Scenario	SC01	SC02	SC03	SC04	SC06	SC07	SC08	SC09	SC10	ADV01	ADV02	ADV03
(a) MSVCM												
CCCMS	8	–	–	–	–	–	–	–	–	–	–	–
Witness	–	4	–	–	–	–	–	–	–	–	–	–
Int.Resources	–	–	3	–	–	–	–	–	–	–	–	–
Authentication	–	–	–	–	–	–	–	–	2	–	2	–
Ext.Resources	–	–	–	2	–	–	–	–	–	–	–	–
Observe	–	–	–	–	5	–	–	–	–	–	–	–
Rescue	–	–	–	–	–	4	–	–	–	–	–	–
Med.Services	–	–	–	–	–	–	–	–	–	2	–	–
Log	–	–	–	–	–	–	–	–	–	–	–	5
Obstacle	–	–	–	–	–	–	–	6	–	–	–	–
Helicopter	–	–	–	–	–	–	6	–	–	–	–	–

Feature/Scenario	SC01	SC02	SC03	SC04	SC06	SC07	SC08	SC09	SC10	ADV01	ADV03
(b) VML4RE											
CCCMS	14	–	–	–	–	–	–	–	–	–	–
Witness	–	8	–	–	–	–	–	–	–	–	–
Int.Resources	–	–	4	–	–	–	–	–	–	–	–
Authentication	–	–	–	–	–	–	–	–	4	–	–
Ext.Resources	–	–	–	3	–	–	–	–	–	–	–
Observe	–	–	–	–	8	–	–	–	–	–	–
Rescue	–	–	–	–	–	5	–	–	–	–	–
Med.Services	–	–	–	–	–	–	–	–	–	4	–
Log	–	–	–	–	–	–	–	–	–	–	5
Obstacle	–	–	–	–	–	–	–	8	–	–	–
Helicopter	–	–	–	–	–	–	8	–	–	–	–

Table 9 Assignment of features to the specification's steps in the fourth release (cont.)

Feature/Scenario	SC01	SC02	SC03	SC04	SC06	SC07	SC08	SC09	SC10
(a) PLUSS									
CCCMS	8	–	–	–	–	–	–	–	–
Witness	–	4	–	–	–	–	–	–	–
Int.Resources	–	–	3	–	–	–	–	–	–
Authentication	–	–	2	–	–	–	–	–	2
Ext.Resources	–	–	–	4	–	–	–	–	–
Observe	–	–	–	–	5	–	–	–	–
Rescue	–	–	–	–	–	4	–	–	–
Med.Services	–	–	–	–	–	2	–	–	–
Log	–	–	–	–	5	5	5	5	–
Obstacle	–	–	–	–	–	–	–	6	–
Helicopter	–	–	–	–	–	–	6	–	–

Feature/Scenario	SC01	SC02	SC03	SC04	SC06	SC07	SC08	SC09	SC10	SCLog
(b) Model Templates										
CCCMS	14	–	–	–	–	–	–	–	–	–
Witness	–	8	–	–	–	–	–	–	–	–
Int.Resources	–	–	4	–	–	–	–	–	–	–
Authentication	–	–	2	–	–	–	–	–	3	–
Ext.Resources	–	–	–	3	–	–	–	–	–	–
Observe	–	–	–	–	8	–	–	–	–	–
Rescue	–	–	–	–	–	5	–	–	–	–
Med.Services	–	–	–	–	–	4	–	–	–	–
Log	–	–	–	–	1	1	1	1	–	5
Obstacle	–	–	–	–	–	–	–	8	–	–
Helicopter	–	–	–	–	–	–	8	–	–	–

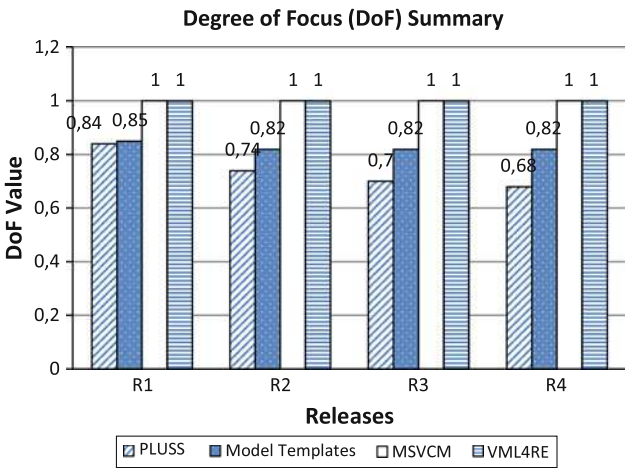

Fig. 9 Average degree of focus

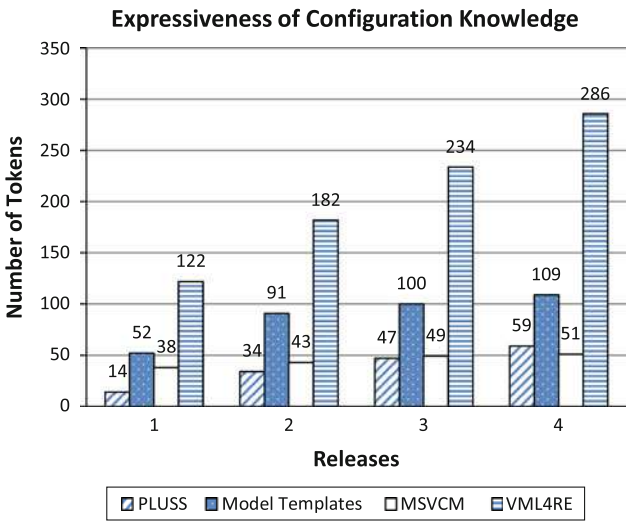

Fig. 10 Expressiveness of the configuration knowledge

behavior of the Log feature was scattered throughout the missions specified using the PLUSS notation (see values of 5 from SC06 to SC09 in Table 9). In contrast, the Model Templates specification required basically one `Log Activity` in each mission, indicating the right point in the specification where the log behavior had to start (see values of 1 from SC06 to SC09 in Table 9). For this reason,

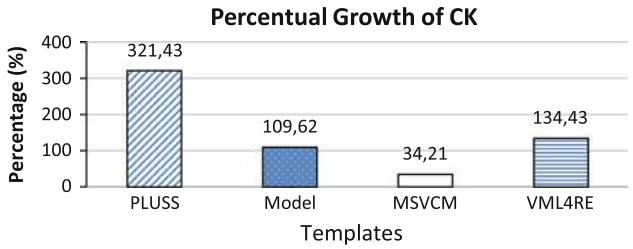

Fig. 11 Growth of the configuration knowledge

we could say that the Log behavior in Model Templates was partially extracted from the missions and specified in a particular activity diagram (SCLog—Log service).

To understand the behavior of a mission, for example, it would be necessary to compose the Log behavior with the existing mission specifications. Otherwise, we would not be able to reason about the complete specification of each mission. Therefore, without proper mechanisms for composing specifications, as supported by MSVCM and VML4RE, relating specifications by means of references could harm understandability, even though this design leads to a better modularization. Using PLUSS, we could have specified the Log behavior in a similar fashion as we have specified it using Model Templates, but the mentioned problem would also have arisen with PLUSS.

After quantifying the average DoS metric (Fig. 8), we noticed that MSVCM and VML4RE reduce, or even eliminate, the scattering of features for the releases of the CCCMS SPL that we modeled. Differently, we are not able to eliminate the Log scattering in PLUSS and Model Templates specifications, mainly because they do not support the composition of common and variant behavior.

5.2 Degree of tangling and focus of specifications

Degree of tangling (DoT) and degree of focus (DoF) (that corresponds to 1-DoT) measure how dedicated a scenario is to one or more features of the SPL. Figure 9 summarizes the corresponding average degree of focus (DoF) of the evaluated specifications. Note that, there is no tangling (DoT = 0) in the MSVCM and VML4RE specifications in the CCCMS, which leads to an average DoF equal to one in those techniques.

In contrast, we were not able to remove the tangling associated with the Authentication and Log features using either PLUSS or Model Template. This tangling occurs because:

- In both techniques (PLUSS and Model Templates), the authentication behavior was specified within the specification of the scenario that assigns tasks to internal resources of the CCCMS. Therefore, using these techniques, the specifications regarding authentication are tangled with the assignment tasks to internal resource specifications.
- Similarly, the specifications using both techniques tangle the Log behavior within the specifications of each mission—since all relevant information (such as used strategies, duration of resolution and problems encountered) have to be registered for each assigned mission.

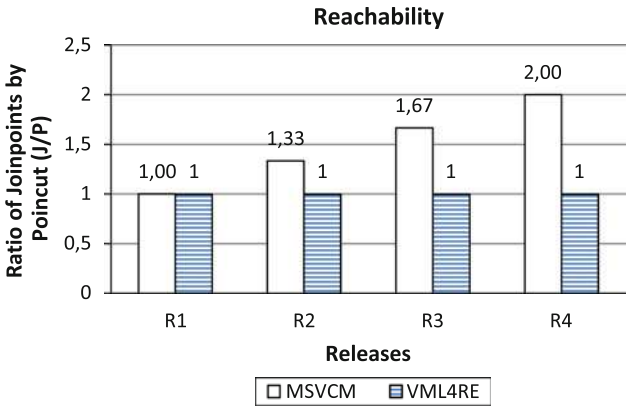

Fig. 12 Average degree of reachability

In fact, the resulting tangling was even higher in PLUSS, because the entire Log specification was scattered throughout several scenarios as it was previously explained. In a different way, we introduced just one activity related to the Log behavior in each mission specified using Model Templates.

The analysis of DoS and DoF here suggests that most of the features require localized and independent specifications. This is a different result when comparing with other studies that evaluated these metrics in source code. For instance, Marc Eaddy found that 95 % of the concerns are scattered through the modular units of a source code [13]. Identifying why those findings were so different is a matter of future work.

6 Analysis of expressiveness

This section presents our analysis of expressiveness considering the different releases of the case study for each technique. We used a metrics suite that quantifies the expressiveness of the configuration knowledge (Sect. 6.1) and the expressiveness of the compositions (Sect. 6.2).

6.1 Expressiveness of the configuration knowledge

In our study, the expressiveness of the configuration knowledge was measured in terms of tokens. In the case of PLUSS and Model Templates, we count tokens looking at the annotations on steps (PLUSS) and transitions between

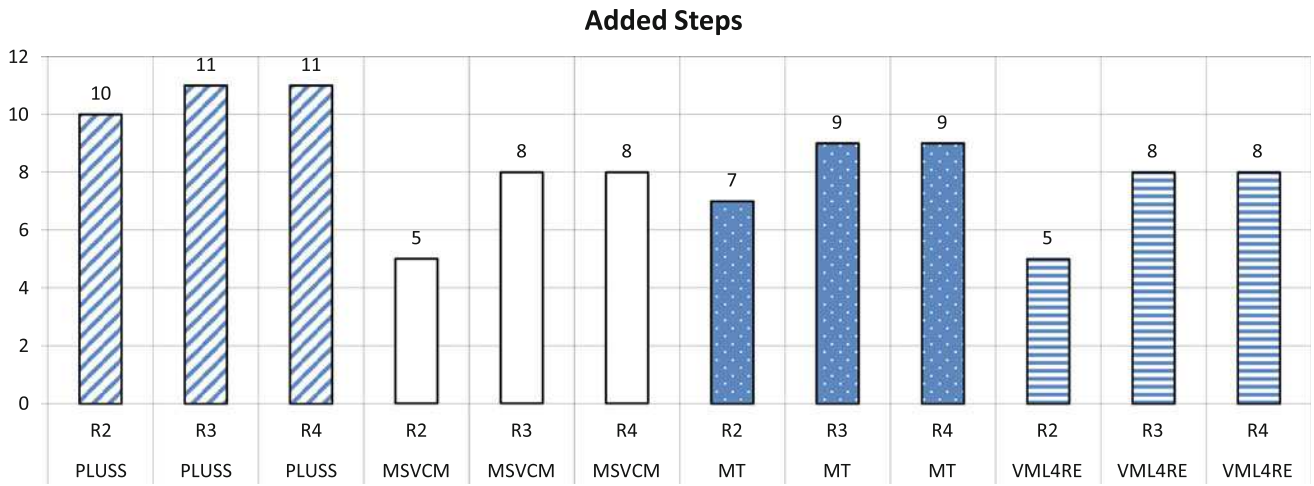

Fig. 13 Number of steps introduced in each release

activities (Model Templates). For example, the annotation "`Medical Services`" has 2 tokens in PLUSS or Model Templates. For MSVCM and VML4RE, we count tokens looking at the specific models used to describe the compositions. For example, "`Medical Services evaluate advice ADV01`" in the MSVCM configuration knowledge model in Table 4 has 5 tokens. An example for VML4RE can be found in the composition model in Fig. 5, lines 1, 2 and 6, where " `Variant for (Medical Services) {merge (''SC07'' , ''ADV01'');}`" has 19 tokens.

Looking at the absolute numbers (Fig. 10), we notice that graphics-based approaches Model Templates, and specially VML4RE, are more verbose than PLUSS and MSVCM to describe the configuration knowledge. PLUSS and Model Templates only associate features with variant behavior. MSVCM and VML4RE use dedicated models to describe all compositions, both common and variable.

If we observe the growth of these numbers, as illustrated in Fig. 11, from the first release (R1) to the last (R4), we notice that it is greater in PLUSS (321%) than in the others. The graphics-based approaches Model Templates (109.62 %) and VML4RE (134.43 %) grow in a similar rate. The growth rate for Model Templates could be a lot higher (comparable with PLUSS) if we had not used one separate diagram for the log services feature, as already discussed. Based on the collected measurements, we can argue that while composition-based approaches require a bigger effort to build a first version of a release (larger upfront investment) than annotation-based approaches (PLUSS and Model Templates), their evolution happens in smaller increments, by requiring a reduced number of new constructs. The large numbers for VML4RE indicate that its composition language could be improved using new actions and removing unnecessary syntactic sugar to reduce verbosity when specifying the configuration knowledge.

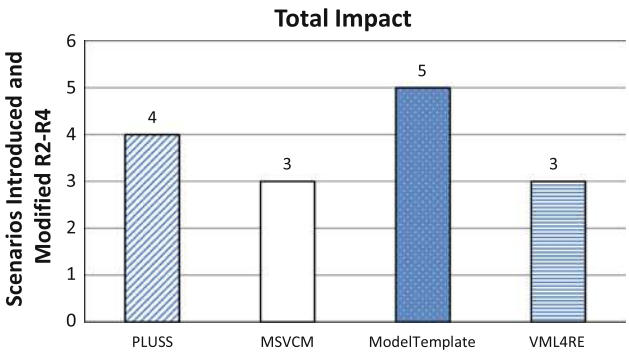

Fig. 14 Stability of the specifications

6.2 Expressiveness of compositions

From the four approaches under investigation, only MSVCM and VML4RE offer specific mechanisms to compose common and variant behavior. To quantify the expressiveness of compositions, we use the notion of *reachability* [9], computed as the ratio between the number of matched join points and the number of composition items [9]. Composition items in MSVCM correspond to the pointcut clauses of advice. Therefore, MSVCM compositions are defined within the *specification language*, whereas composition items in VML4RE correspond to actions such as the *connect* construct detailed in each variant element of the VML4RE *configuration language*.

In our study, the reachability of VML4RE compositions is one—each VML4RE composition *reaches* a particular join point.[7] In contrast, MSVCM supports different

[7] An early version of VML4RE [2] had some language constructs that simplify matching specific fragments (i.e., join points) in the scenarios reducing the verbosity of the composition model. For example, pointcut designators such as "equal," "startsBy," "finishesWith," "contains" and quantifiers such as "*", "?"

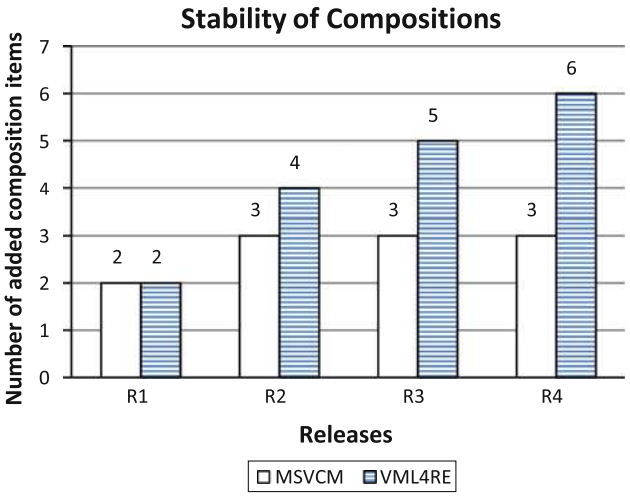

Fig. 15 Stability of compositions

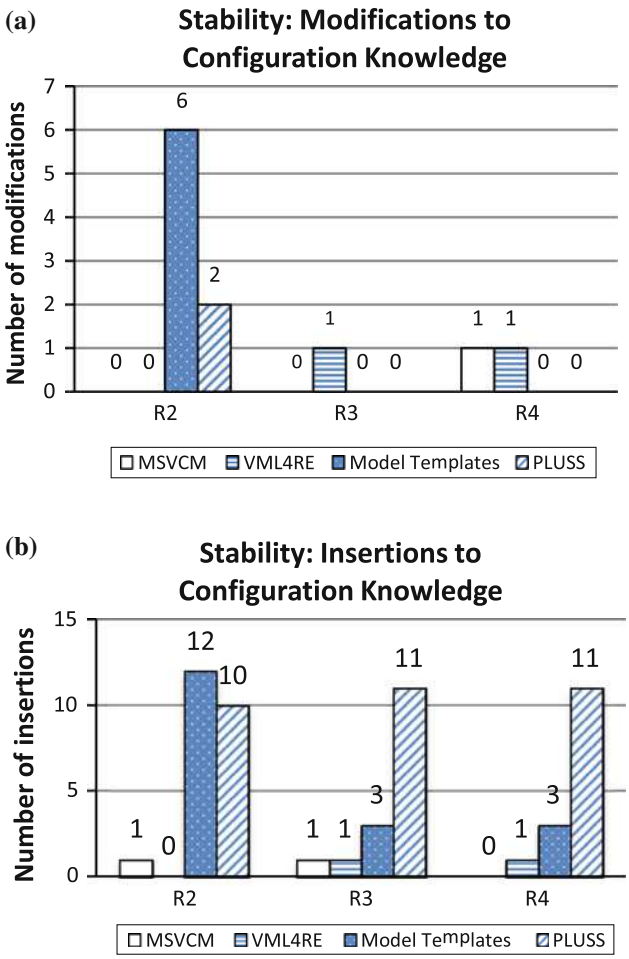

Fig. 16 Stability of the configuration knowledge (a) Modifications. (b) Insertions

mechanisms for quantification. For instance, the Log advice is applied after all scenarios named with the pattern `% mission%` . Consequently, the average expressiveness of the composition increases (see Fig. 12) in the later releases of MSVCM specifications, since new missions are introduced.

By comparing these results with the stability of composition assessments (Sect. 7.2), we could realize a strong correlation between the expressiveness and the stability of the compositions—the greater the stability, the greater the expressiveness of the compositions. Introducing new specifications that satisfy a composition item does not require changes in the composition concern. Besides that, a potential side effect regarding expressiveness is that undesired join points might be caught by a composition item. In those situations, the composition item must be refined.

7 Analysis of stability

This section presents our analysis of stability considering the different releases of the case study specified with each technique. The metrics used quantify the stability of the specifications (Sect. 7.1), the stability of the compositions (Sect. 7.2) and the stability of the configuration knowledge (Sect. 7.3).

7.1 Stability of specifications

Figure 13 shows how many steps in MSVCM and PLUSS (or activities in Model Templates and VML4RE) have been introduced to evolve the specifications from one release to another. It can be noticed that more steps were introduced to evolve the annotative techniques (PLUSS and Model Templates), since the specification for the log feature is not well modularized.

For instance, the second release introduced the log feature, whose entire specification is scattered throughout the *rescue* and *observe* missions of PLUSS specifications. Since five steps were required to specify the log feature, we had to introduce a total of ten new steps in the second release of the PLUSS specifications (five steps for each scenario that requires the Log behavior). In the Model Templates specification, one activity was introduced in each mission (to indicate where the Log behavior should start), and a new activity diagram for specifying the Log behavior was created. Using the compositional approaches (MSVCM and VML4RE), only steps for describing the Log behavior had to be created, in such a way that no additional steps were introduced in the original specifications of rescue and observe missions.

The latter releases (R3 and R4) detailed the behavior of helicopter transport and remove obstacle missions, which also require the log behavior. Besides the steps related to

Fig. 17 Summary of evaluation results

Attribute	Metric	PLUSS	MT	MSVCM	VML4RE
Modulartiy	Average DOS	0,113	0,100	0,058	0,000
	Average DOF	2,960	3,310	4,000	4,000
Stability of specifications	Added steps	32	25	21	21
	Impact	4	5	3	3
Stability of composition	Added compositions	-	-	2,75	4,25
Stability of CK	Modifications	2	6	1	2
	Insertions	32	18	2	2
Expresiv. of compositions	Reachability	-	-	1,5	1,0
Expresiveness of CK	Percentual Growth	321,43	109,62	34,21	134,43

those missions, in PLUSS we had also to detail the steps of the Log behavior within both missions. For this reason, the number of introduced steps is higher in PLUSS than in the other approaches. For instance, in Model Templates we just had to specify the behavior of the new missions, plus one specific activity to indicate the point where the log behavior starts. Moreover, we did not have to introduce additional steps when using MSVCM and VML4RE, which reduced the number of introduced steps.

Also, regarding the stability of specifications, Fig. 14 summarizes the number of scenarios that have been introduced and modified, according to changes required to evolve the specifications from the first to the last release. In Fig. 14, PLUSS and Model Templates required a higher number of introduced and modified scenarios. Indeed, the evolution of SPL specifications using the annotative style requires to take into account many places affected by the propagation of the changes.

7.2 Stability of compositions

We measured stability in terms of the number of composition items required to specify each CCCMS release. Since in PLUSS and Model Templates variability is tangled with core functionality, only MSVCM and VML4RE were evaluated. Remember that composition items in MSVCM are specified in the pointcut clause of advices, whereas compositions in VML4RE are specified in the variant construct from the textual description in that language.

In the first release, since each of the optional features (Authentication and Medical Services) changes a specific point of CCCMS, the number of composition items in both techniques is the same (Fig. 15). In the second release, the Log behavior was specified in one MSVCM advice whose composition item refers to all missions. Differently, two composition items were required to indicate that Log advice in VML4RE should be applied to the observer and execute rescue missions. For that reason, the number of composition items in MSVCM increased by one in the second release, while in VML4RE the number of composition items increased by two in the second release.

Likewise, introducing new missions that had to be advised by the Log behavior did not require new composition items in MSVCM. However, new composition items had to be introduced to *connect* the Log advice specified in VML4RE to the new missions specified in the third and fourth releases of the case study. This leads to a worse stability of the compositions, when compared to MSVCM. This highlights the benefits of using more expressive mechanisms to compose common and variant behavior.

7.3 Stability of the configuration knowledge

Stability of the configuration knowledge is quantified by measuring the modifications (changes or insertions) made to the configuration knowledge. PLUSS and Model Templates do not provide specific and separate models to represent the configuration knowledge. As a consequence, this knowledge is scattered throughout the specifications in the form of feature annotations for steps (PLUSS) and stereotypes (Model Templates). While in VML4RE, we observe changes and additions of *variants* in the VML4RE composition specification; in MSVCM, we observe changes and additions of *configuration items* in the configuration knowledge.

Figure 16 summarizes our findings (lower is better). We can notice that the composition-based approaches tend to be more stable than annotation-based approaches PLUSS and Model Templates. This may be explained by the fact that MSVCM and VML4RE have specific models to deal with the configuration knowledge, while in PLUSS and Model Templates, the knowledge is spread across multiple artifacts. For Model Templates, due to the modularization of the Log service feature in one diagram, which is refered by other scenarios, the values for insertions in R3 and R4 (Fig. 16b) do not grow as much as PLUSS that does not modularize the Log behavior. MSVCM and VML4RE require the least numbers of insertions and modifications during the evolution scenarios as it will be shown in the Impact metric in Fig. 16b. Further studies could indicate if that holds true for most cases, and in which cases these compositional techniques would not be of good use.

8 Summary of results

Figure 17 complements the analysis provided from Sects. 5–7 with a summary of the evaluation results for the four CCCMS releases. For each metric, we assigned a symbol that helps to distinguish which techniques have good, average or bad results in comparison with the others. The upward arrow means "good", the rightward arrow means "average" and the downward arrow means "bad". The assignment of each symbol was determined automatically using the conditional formatting feature of MS Excel which assigned symbols to series of values based on percentages. In our metrics, we used the following percentages limits: bad $\geq$67, average $<$67 and $\geq$33, good $<$33. For stability and expressivity of the compositions that only applied to two techniques, we used short arrows indicating which one had a better value (upward arrow) than the other (downward arrow).

In general, the lower the value obtained for a metric, the better the approach for the corresponding attribute, except for degree of focus (DoF) and reachability, which follow an inverse logic. For example, in DoS, the percentage limits to assign the symbols were as follows: good $\geq$67, average $<$67 and $\geq$33 and bad $<$33.

Modularity for each technique was measured as the means of DoS and DoF for the four releases. The compositional approaches VML4RE and MSVCM had better results in both DoS and DoF, the reason being that both approaches help specifying each feature separately in one or few scenarios. This leads to DoS values very close to 0 for MSVCM and 0 for VML4RE. Similarly, VML4RE and MSVCM specified each scenario focusing on only one or few features, which resulted in a good DOF = 4 in comparison with Model Templates (DOF = 3.31) and PLUSS (DOF = 2.96). We believe that the annotations mechanism used by Model Templates and PLUSS fail to improve modularity of scenario specifications, even with few features that are scattered through the system such as Log and Authorization.

Stability of specifications, composition and configuration knowledge (CK) was measured as the sum of all the individual values for stability metrics obtained in all the releases. Similarly, the Impact was measured as the sum of added and modified scenarios in all releases. The compositional approaches VML4RE and MSVCM obtained the same values for stability of the specifications, meaning that the most noticeable differences between the compositional approaches (apart from their notation) are found in their expressivity and not in the specification of the scenarios or modularity itself. On the other hand, PLUSS was the best technique to keep almost intact CK specifications (Modifications = 2), although it was done at the price of many insertions (Insertions = 32). A different phenomenon happen with the rest of the approaches that faced evolution of CK combining few modifications and insertions of CK.

Expressiveness of CK was taken directly from Fig. 11. It was measured as the mean of reachability for the four releases. VML4RE had a low reachability (=1) compared to MSCVM (=1.5). The percentual growth of expressiveness of CK in MSCVM (=34.21) was the best while in PLUSS, it was the worse (=321.43). The results of expressivity of VML4RE are similar to the ones of Model Templates (percentual growth of expressiveness of CK = 134.43 and CK = 109.62) that does not have any separate configuration knowledge model. We see that the lack or presence of quantification mechanisms affected expressivity, and expressivity affected stability of compositions and CK. For example, the lack of quantification mechanisms in VML4RE limited the reachability of its pointcuts and influenced negatively the stability of compositions and CK because of new required variants and compositions items to match elements introduced in new scenarios.

9 Threats to validity

We chose the CCCMS SPL because the original specifications [25, 21] as well as the scenarios for each technique are publicly available (see footnote 6), allowing other researchers to replicate and extend our study. Indeed, CCCMS SPL is a good choice for conducting our assessments, as it is becoming a benchmark for SPL development, being used by different communities, such as the AOM workshop at Bellairs[8] and Comparison Modeling Approaches (CMA) workshop series (see footnote 1). In particular, these workshops have been contributing to create a body of knowledge around this exemplar by challenging authors of modeling approaches to apply their approaches to this case study and upload the resulting models into ReMoDD (Repository for Model Driven Development) (see footnote 3). (19 approaches had already been modeled at the time this paper was written.)

The resulting feature model, although not very large, includes several *mandatory*, *optional* and *or features*; some of the optional features change the base specification in a single place, whereas others (e.g., the Log feature) change the specification throughout different scenarios.

However, the type and size of the investigated releases limit our conclusions, as we mainly concentrate our study on increments to the base specification (the first release). Other types of changes were not covered here, such as bug fixes. Nonetheless, some of our conclusions are still valid and could be generalized. For instance, evolving a localized feature specification should not reveal significant

[8] http://www.cs.mcgill.ca/~joerg/SEL/AOM_Bellairs_2012.html.

differences among the investigated techniques. On the contrary, if we had to evolve the Log specification, which is not well localized in PLUSS and Model Templates, our assessment procedure would reveal that these techniques are less stable than MSVCM and VML4RE—since several places of the specifications written in PLUSS and Model Templates are likely to change. Moreover, we do not have to change or introduce new requirements to the original SPL specifications, using the releases presented in this paper. Consequently, they have not been proposed to favor any particular approach.

Additionally, the chosen metrics suite could be seen by some a threat to the validity of our work because the metrics could be engineered to favor one approach over the others. However, some of the metrics (DoS, DoF, reachability and stability of the compositions and specifications) have been previously validated and used in related works [7, 9]. The metrics proposed to evaluate the configuration knowledge (expressiveness and stability) are new contributions of this paper, and we acknowledge their validation as a matter of future work. In particular, we use the number of tokens to count the expressiveness of the configuration knowledge, what makes our findings dependent on the concrete syntax of the current languages used to specify the configuration knowledge in each technique. As a future work, we plan to investigate the use of other ways to quantify the expressiveness of different representations of configuration knowledge.

Finally, we chose to use feature diagrams to model variability because (1) the case study provided no other alternative models, and (2) this technique is widely used in requirements engineering to specify variability, not only by researchers but also by industries (e.g., Gears, pure::variants, Linux Kconfig and eCos). Although this choice could raise a question about languages that use different variability models, we are focusing on "features" and so "feature" diagrams emerged naturally to specify features and their relationships. Additionally, alternative specification diagrams for variability are not representative of any group of techniques.

10 Related work

There are several works that are connected to our research. Here, we introduce these works and relate them to our study.

Metrics for quantifying scattering and tangling have been applied for assessing modularity in aspect-oriented programs [13, 16]. We could have adapted absolute measures for quantifying scattering and tangling, such as concern diffusion over components and concern diffusion over lines of code [17]. However, absolute measures just reveal if a feature is scattered or not, giving no information about the level of scattering. As a matter of fact, this limitation hinders the comparison of modularity between different specifications. Consequently, we customized a metrics suite proposed by Eaddy and his colleagues [13].

To improve our confidence in the results, we also measure the stability of the specifications and compositions by means of a suite of metrics that had been already validated and used in a previous work [9]. Here, we use these metrics to evaluate stability of SPL scenario specifications, whereas Chitchyan et al. proposed and used those metrics to assess stability of semantic and syntactic aspect-oriented approaches for requirements engineering (AORE). Hence, we had to extend their metrics to assess also the stability and expressiveness of the configuration knowledge, not only the stability of specifications and compositions.

In previous work, we presented a comparison of modularity involving MSVCM and PLUSS [7]. Here, we contribute with a deeper evaluation, considering other quality attributes (such as stability and expressiveness) and additional techniques (Model Templates and VML4RE). Also, Sampaio et al. compared different AORE approaches with respect to the accuracy of resulting specifications and the effort required to build *aspect compositions* [27]. We postpone a similar evaluation to future work.

There are other works specifically proposed to represent variability in requirements models. For instance, PLUC [6] extends the use case notation for SPL engineering. It follows an annotative style, but it does not present a good separation between the problem and the solution spaces. We investigated PLUC in a previous work [8], which led us to conclude that PLUC is not maintainable at all.

Finally, there is a comprehensive report about Requirements Engineering in SPLE [22]. That report includes diverse techniques including for example, scenarios, goals and viewpoints. Nevertheless, it does not employ any empirical assessment using a common case study or experiments. Finally, it does not analyze expressivity, modularity and stability in depth as our work does.

11 Conclusions

This paper presented an empirical study that compares and analyzes four existing approaches—MSVCM, PLUSS, Model Templates and VML4RE—to model and manage variabilities in SPL requirements specifications. The investigated approaches are representative of a set of new variability management approaches. They reflect different perspectives of existing approaches, in particular textual versus graphical based and annotation versus composition based. In our study, they were evaluated for their support to modularity, expressiveness and stability through the

specification and evolution of a car crash crisis management system SPL.

Our findings have shown that the composition-based approaches have greater potential to produce more stable and modular SPL requirements specifications. MSVCM and VML4RE promoted the modular specification of the scenarios and configuration knowledge, which brought more stability to SPL requirements specifications during their respective evolution. It was also observed that the aspect-oriented mechanisms of MSVCM offer great contributions to improve the modularization of variabilities in the scenario specification. In particular, scenario advices were used to modularize variabilities, which are composed with the SPL core scenarios. The absence of quantification mechanisms in the VML4RE specifications has negatively contributed to reduce their expressiveness and stability.

The following recommendations were derived from the analysis and results of our study, which might contribute to the definition of new variability management approaches for requirements specifications:

1. Support the separated and modular specification of configuration knowledge between variability and requirements models;
2. Adopt quantification mechanisms for the specification of the configuration knowledge, aiming at simplifying and increasing its expressiveness;
3. Use early aspects techniques to modularize crosscutting scenarios and promote their seamless composition with SPL core requirements, thus contributing to variability management.

The benefits that aspect-oriented techniques can bring to the variability management in the SPL requirements specifications should be further explored. We are currently using these recommendations to improve MSVCM and VML4RE.

Based on the metrics used, we observed that aspect-oriented approaches produce more modular and stable specifications, mainly when crosscutting features exist in the system. However, there may be other unexamined factors, such as learning curve and usability, where the annotation-based approaches may be more advantageous. In the near future, we plan to extend our metrics suite to include other relevant quality attributes, particularly reliability, usability and footprint.

Acknowledgments This work was partially supported by the Centro de Informática e Tecnologias de Informação (CITI) - Portugal, the European Project AMPLE - contract IST-33710, the grant SFRH/BD/46194/2008 of Fundação para a Ciência e a Tecnologia - Portugal, and the National Institute of Science and Technology for Software Engineering (INES) - CNPq under grants 573964/2008-4 and CNPQ 560256/2010-8.

References

1. Alexander IF, Maiden N (eds) (2004) Scenarios, stories, use cases: through the systems development life-cycle. Wiley, Chichester
2. Alférez M, Kulesza U, Weston N, Araújo J, Amaral V, Moreira A, Rashid A, Jaeger MC (2008) A metamodel for aspectual requirements modelling and composition. Tech. Report D 1.3 Ample-project
3. Alférez M, Santos J, Moreira A, Garcia A, Kulesza U, Araújo J, Amaral V (2010) Multi-view composition language for software product line requirements. In: SLE'09: Revised selected papers, Denver, CO, USA, pp. 103–122. doi:10.1007/978-3-642-12107-4_8
4. Bachmann F, Bass L (2001) Managing variability in software architectures. In: SIGSOFT Software Engineering Notes, vol 26. New York, NY, USA, pp 126–132. doi:10.1145/375212.375274
5. Bergmans L, Aksit M (2001) Composing crosscutting concerns using composition filters. Commun ACM 44(10):51–57. doi:10.1145/383845.383857
6. Bertolino A, Gnesi S (2003) Use case-based testing of product lines. In: ESEC/FSE'03, Helsinki, Finland, pp. 355–358. doi:10.1145/940071.940120
7. Bonifácio R, Borba P (2009) Modeling scenario variability as crosscutting mechanisms. In: AOSD'09, Charlottesville, Virginia, USA, pp 125–136. doi:10.1145/1509239.1509258
8. Bonifácio R, Borba P, Soares S (2008) On the benefits of variability management as crosscutting. In: Early Aspects Workshop at AOSD. Brussels, Belgium
9. Chitchyan R et al. (2009) Semantic vs. syntactic compositions in aspect-oriented requirements engineering: an empirical study. In: KJ Sullivan (ed) AOSD'09, Charlottesville, Virginia, USA, pp 149–160. doi:10.1145/1509239.1509260
10. Clements P, Northrop LM (2001) Software product lines: practices and patterns professional. Addison-Wesley, Boston
11. Czarnecki K, Antkiewicz M (2005) Mapping features to models: a template approach based on superimposed variants. In: GPCE'05, Tallinn, Estonia, pp 422–437. doi:10.1007/11561347_28
12. Czarnecki K, Eisenecker U (2000) Generative programming: methods, tools, and applications. ACM Press/Addison-Wesley Publishing Co. New York
13. Eaddy M, Aho A, Murphy GC (2007) Identifying, assigning, and quantifying crosscutting concerns. In: ACoM'07 held with ICSE'07, Minneapolis, MN, USA, p 2. doi:10.1109/ACOM.2007.4
14. Eriksson M, Borstler J, Borg K (2005) The pluss approach, domain modeling with features, use cases and use case realizations. In: SPLC'05, Rennes, France, pp 33–44. doi:10.1007/11554844_5
15. Eriksson M, Börstler J, Borg K (2009) Managing requirements specifications for product lines: an approach and industry case study. J Syst Softw 82(3):435–447. doi:10.1016/j.jss.2008.07.046
16. Figueiredo E et al (2008) Evolving software product lines with aspects: an empirical study on design stability. In: ICSE'08, Leipzig, Germany, pp 261–270. doi:10.1145/1368088.1368124
17. Figueiredo E et al. (2008) On the maintainability of aspect-oriented software: a concern-oriented measurement framework. In: CSMR'08, Athens, Greece, pp 183–192. doi:10.1109/CSMR.2008.4493313
18. Gomaa H (2004) Designing software product lines with UML: from use cases to pattern-based software architectures. Addison Wesley Longman Publishing Co., Inc., Redwood City
19. Kästner C, Apel S, Kuhlemann M (2008) Granularity in software product lines. In: ICSE'08, Leipzig, Germany, pp 311–320. doi:10.1145/1368088.1368131
20. Katz S, Mezini M, Kienzle J (eds) (2010) Transactions on aspect-oriented software development VII: a Common case study for aspect-oriented modeling. Lecture notes in computer science, vol 6210. Springer

21. Kienzle J, Guelfi N, Mustafiz S (2010) Crisis management systems: a case study for aspect-oriented modeling. T. Aspect-Oriented Software Development 7:1–22
22. Kovacević J, Alférez M, Kulesza U, Moreira A, Araújo J, Amaral V, Alves V, Rashid A, Chitchyan R (2007) Survey of the state-of-the-art in requirements engineering for software product line and model-driven requirements engineering. Tech. Report D 1.1, Ample-project
23. Kulesza U, Sant'Anna C, Garcia A, Coelho R, von Staa A, de Lucena CJP (2006) Quantifying the effects of aspect-oriented programming: a maintenance study. In: ICSM'06, Philadelphia, PA, USA, pp 223–233. doi:10.1109/ICSM.2006.48
24. Lippert M, Lopes CV (2000) A study on exception detection and handling using aspect-oriented programming. In: ICSE'00, Limerick, Ireland, pp 418–427. doi:10.1145/337180.337229
25. Mustafiz JKNGS (2009) Crisis management systems: a case study for aspect-oriented modeling. Tech. Report, School of Computer Science, McGill University, Montreal. http://www.cs.mcgill.ca/joerg/taosd/TAOSD/TAOSD.html
26. Pohl K, Böckle G, Linden FJvd (2005) Software product line engineering: foundations, principles and techniques. Springer, New York, Inc., Secaucus, NJ
27. Sampaio A, et al. (2007) A comparative study of aspect-oriented requirements engineering approaches. In: First international symposium on empirical software engineering and measurement, 2007. ESEM'07, Madrid, Spain, pp 166–175. doi:10.1109/ESEM.2007.15
28. Sugumaran V, Park S, Kang KC (2006) Introduction. Commun ACM 49(12):28–32
29. Zschaler S, Sánchez P, Santos J, Alférez M, Rashid A, Fuentes L, Moreira A, Araújo J, Kulesza U (2009) VML*: a family of languages for variability management in software product lines. In: SLE'09: Revised selected papers, Denver, CO, USA. doi:10.1007/978-3-642-12107-4_7