Softw Syst Model (2017) 16:97–123
DOI 10.1007/s10270-015-0471-3

THEME SECTION PAPER

# Empirical assessment of two approaches for specifying software product line use case scenarios

**Rodrigo Bonifácio**[1] · **Paulo Borba**[2] · **Cristiano Ferraz**[3] · **Paola Accioly**[2]

Received: 10 October 2014 / Revised: 6 March 2015 / Accepted: 25 April 2015 / Published online: 26 May 2015


**Abstract** Modularity benefits, including the independent maintenance and comprehension of individual modules, have been widely advocated. However, empirical assessments to investigate those benefits have mostly focused on source code, and thus, the relevance of modularity to earlier artifacts is still not so clear (such as requirements and design models). In this paper, we use a multimethod technique, including designed experiments, to empirically evaluate the benefits of modularity in the context of two approaches for specifying product line use case scenarios: PLUSS and MSVCM. The first uses an annotative approach for specifying variability, whereas the second relies on aspect-oriented constructs for separating common and variant scenario specifications. After evaluating these approaches through the specifications of several systems, we find out that MSVCM reduces feature scattering and improves scenario cohesion. These results suggest that evolving a product line specification using MSVCM requires only localized changes. On the other hand, the results of six experiments reveal that MSVCM requires more time to derive the product line specifications and, contrasting with the modularity results, reduces the time to evolve a product line specification only when the subjects have been well trained and are used to the task of evolving product line specifications.

**Keywords** Usage scenarios · Requirements engineering · Software modularity · Software product lines · Experimentation in software engineering

Communicated by Andrzej Wąsowski and Thorsten Weyer.

✉ Rodrigo Bonifácio
rbonifacio@cic.unb.br

Paulo Borba
phmb@cin.ufpe.br

Cristiano Ferraz
cferraz@de.ufpe.br

Paola Accioly
prga@cin.ufpe.br

[1] Computer Science Department, University of Brasília, Brasília, Brazil

[2] Informatics Center, Federal University of Pernambuco, Recife, Brazil

[3] Statistics Department, Federal University of Pernambuco, Recife, Brazil

## 1 Introduction

The software product line (SPL) development approach aims to reduce costs and time to market of systems from a common domain [16,54,56]. Based on this approach, systems belonging to an SPL are generated by means of a product derivation process, which *selects and configures* reusable assets. These assets might consist of different types of artifacts, such as requirements, design models, source code, and tests that are available to all SPL members. Additionally, besides reducing the costs to launch new products, the SPL approach has the potential to improve product maintainability. For instance, Linden et al. [54] suggest that "*the amount of code and documentation that must be maintained (using an SPL approach) is dramatically reduced*," which emphasizes the relevance of *documentation reuse* and thus that SPL development is not just a matter of code reuse. In fact, SPL is a systematic approach for mass customization of software products, which involves not only binaries, but also requirements, user manuals, and so on.

Apart from considering technical perspectives, an SPL is also strongly characterized by its domain (or market segment) [16], whose relevant concepts are often documented

as features—a "*user-visible aspect or characteristic of the domain*" [40]. This way, the goal of feature modeling, a widely used technique in SPL development, is to detail the relationships between the concepts of a domain. These relationships precisely define the scope, the variability space, and the set of SPL members. As a consequence, specific feature configurations are used as input for the product derivation activity.

To support that product derivation activity, developers represent *variation points* throughout the different assets that comprise an SPL. In addition, they also have to describe how to solve these variation points, given a configuration of features. Both tasks are related to the variability management concern, which is inherently crosscutting, since the contribution of a feature is frequently scattered throughout a number of SPL assets. In addition, depending on different factors (such as modularization technique and feature granularity), the realization of a feature by the SPL assets might be based on either annotations or compositions [41]. The annotative style entangles the representation of common and variant behavior, which compromises the modular feature design, whereas the compositional style supports a better separation of variant behavior into dedicated modules.

This polarity between annotative and compositional styles does not occur only at source code, but also in other SPL assets, such as requirements specifications. As detailed in Sect. 2, recent approaches for representing variability in use case scenarios are either invasive [7,26] or compositional [3,8].

To better understand the implications of each approach, in this paper, we

- Evaluate the use of two techniques (MSVCM and PLUSS) for representing use case scenario variabilities. These techniques, discussed in Sect. 2, were selected because they are quite similar according to the input specifications and well represent the compositional (MSVCM) and invasive (PLUSS) styles for variability management.
- Report on the results of an empirical study (Sect. 3) that investigates the modularity of SPL specifications (Sect. 4) and on the results of controlled experiments proposed to evaluate the effort to derive and evolve SPL specifications (Sect. 5). Previous works do not extensively investigate the benefits and costs of using compositional approaches for representing SPL variabilities in use case scenarios.

Our main hypothesis is that using a compositional approach for specifying SPL use case scenarios improves the modularity of feature specifications. As a consequence, we expect that this benefit of feature modularity would come together with a reduction in the effort necessary to evolve the SPL specifications. We also investigate what are the payoffs related to the use of compositional approaches in the context of the extractive method for SPL development, investigating the effort to derive an SPL specification from the specification of existing products. We present some threats to the validity of our work in Sect. 6, relate our research with existing works in Sect. 7, and present our final considerations in Sect. 8.

## 2 Evaluated techniques

Optional and configurable assets support generative approaches for SPL development [49], where a given feature configuration triggers the automatic selection or customization of assets. This includes use case scenarios, which not only describe the behavior of a system but also assist different activities such as system design and the specification of test cases and user manuals.

For that reason, several notations for representing SPL variability in use case scenarios have been proposed, including product line use cases (PLUC) [7] and product line use case modeling for systems and software engineering (PLUSS) [26,28]. However, in spite of the benefits of variability representation, these approaches do not modularize the specification of crosscutting features, which scatter through several use case scenarios and do not separate variability management and scenario specifications. These limitations are typically observed in approaches that use annotations throughout the SPL assets, in order to represent how they are related to features.

To overcome these limitations, compositional approaches such as MSVCM and VML4RE [3] have been proposed to modularize features. To better understand the differences between compositional- and annotative-based product line approaches for use case scenarios, we describe two alternative specifications of a product line adaptation for the *E-Finance* system [51]. We show alternative specifications based on one annotative approach (PLUSS) and a compositional approach (MSVCM).

Figure 1 presents part of the E-Finance feature model [18,34], which depicts the common and variable features of our example. We represent the feature model using a feature diagram, where relationships between a parent feature and its children are categorized as *optional features*, which might not be selected in a specific product; *mandatory features*, which must be selected in all products if the parent feature is also selected; *inclusive features*, which, when selected, implies one or more sub-features; and *exclusive features*, which, when selected, implies that only one sub-feature must be selected. Besides these relationships, feature models allow the specification of constraints among features. For instance, the constraint ($Iris \Rightarrow IrisRecognizer$) states that the feature Iris requires the selection of the external device iris recognizer.

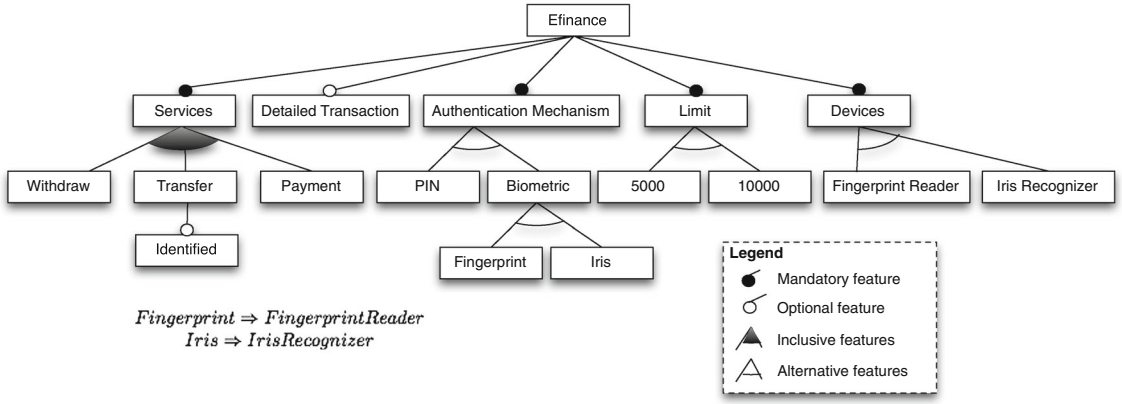



**Fig. 1** E-Finance feature model

| Code | User Action | System Response |
|---|---|---|
| SC01-1 | The customer selects a withdrawal. | The system creates a new withdrawal and asks for the amount to withdraw. |
| SC01-2 | The customer enters the amount to withdraw. | The system retrieves the current balance of the selected account. |
| SC01-3 | - | The system verifies that the requested amount is not grater than the current balance plus $LIMIT$ |
| SC01-4 (a) | - | The system asks the customer to enter her PIN (Personal Identification Number). |
| SC01-4 (b) | - | The system asks the customer to put her finger in the fingerprint reader. |
| SC01-5 (a) | The customer fills in the PIN. | The system authenticates the customer's PIN. |
| SC01-5 (b) | The customer puts her finger in the fingerprint reader device. | The system authenticates the customer, according to the comparison of the captured fingerprint date. |
| SC01-6 | - | The transaction handler starts the processing of a transaction. |
| SC01-7 | - | The system withdraws the amount from the account. |
| SC01-8 | - | The cash money is provided to the customer. |
| SC01-9 | - | An entry with the transaction information is logged to the overview of the completed transactions. |
| SC01-10 | - | The transaction is removed from the transaction queue. |

Legend: PIN | Fingerprint | Detailed Transaction

**Fig. 2** Withdraw scenario specified using an annotative approach

Figure 2 depicts the *Withdraw* service specification in PLUSS. Notice that a single artifact represents all valid configurations that are related to this scenario, mixing common behavior (non-colored steps), variant behavior (colored steps), and configuration information (required features represented by distinct colors). For example, steps SC01-4(a) and SC01-4(b), and SC01-5(a) and SC01-5(b) are never performed in a sequence. They are alternative steps: Steps SC01-4(a) and SC01-5(a) will be present in a particular product use case scenario only if the *PIN* feature is selected, whereas steps SC01-4(b) and SC01-5(b) will be present only if the feature *Fingerprint* is selected. In a similar way, steps SC01-6, SC01-9, and SC01-10 are optional and will be present only if the feature *DetailedTransaction* is selected. Representing all possible variants in the same asset prevents analysts from modularly reasoning about feature specifications.

In addition, the behavior related to the authentication mechanism is required not only by the *Withdraw* service, but also by all other *E-Finance* services such as *Transfer* and *Payment*. As a consequence, it is not possible to evolve the specification of the authentication mechanism in a modular way. Indeed, we have to change all scenarios in which the authentication concern is relevant. For instance, notice that we do not consider the effect of the Iris authentication mechanism in the specification of the *Withdraw* scenario shown in Fig. 2. We intentionally postponed the introduction of this feature in order to make clear the non-modular style of evolving product line specifications based on annotative approaches. In fact, introducing a new authentication mechanism changes the specification of all scenarios that requires authentication.

In contrast, the compositional specification of the Withdraw service and related features using MSVCM [8] modularizes optional behavior by means of specialized scenarios

**Id: SC01**

| Code | User Action | System Response |
|---|---|---|
| SC01-1 | The customer selects a withdrawal. | The system creates a new withdrawal and asks for the amount to withdraw. |
| SC01-2 | The customer enters the amount to withdraw. | The system retrieves the current balance of the selected account. |
| SC01-3 | - | The system verifies that the requested amount is not grater than the current balance plus $LIMIT$ **@authentication** |
| SC01-4 | - | The system withdraws the amount from the account. |
| SC01-5 | - | The cash money is provided to the customer. |

**Fig. 3** Withdraw scenario specified using MSVCM

named *advice*. Using MSVCM, one might specify the common steps of the Withdraw service as depicted in Fig. 3. Note that this scenario comprises only the specific steps of the Withdraw service—there is no step related to the authentication mechanism or detailed transaction within the scenario SC01 of Fig. 3. Besides that, step SC01-3 is marked with the `@authentication` tag, exposing it as a *join point* where the pieces of advice related to the authentication mechanism should be applied to. In MSVCM, the *step id* construct could also be used as a join point, as we explain later, although it is beyond the scope of this article to discuss the join point model of MSVCM.

As discussed, optional steps in MSVCM are modularized using aspect-oriented constructs, particularly advice and pointcuts [42,43,55]. Regarding the initial set of features that extend the common behavior of the Withdraw service, we specify three advice: two for modularizing the initial set of authentication mechanisms (PIN and Fingerprint) and one for modularizing the *DetailedTransaction* feature.

Figure 4 shows the advice for the PIN and Fingerprint mechanisms. Both advice should be applied after the steps marked with the `@authentication` annotation. Thus, introducing a new authentication mechanism, for instance, the Iris authentication mechanism, would require the definition of a new *after* advice whose pointcut should refer to the `@authentication` annotation—differently from the PLUSS specification, this type of change does not modify any other scenario that already exposes the mentioned join point.

The behavior of the *DetailedTransaction* feature is also modularized using advice (Fig. 5). However, since this behavior must be applied around some steps of the scenarios, we preferred to modularize it using an around advice, instead of a composition of two advice of types *before* and *after*. Around advice requires pointcuts that specify a range of steps that are overridden by the sequence of steps of the advice

**Id: ADV1**

**Pointcut: after @authentication**

| Code | User Action | System Response |
|---|---|---|
| ADV1-1 | - | The system asks the customer to enter her PIN (Personal Identification Number). |
| ADV1-2 | The customer fills in the PIN. | The system authenticates the customer's PIN. |

**(a)**

**Id: ADV2**

**Pointcut: after @authentication**

| Code | User Action | System Response |
|---|---|---|
| ADV2-1 | - | The system asks the customer to put her finger in the fingerprint reader. |
| ADV2-1 | The customer puts her finger in the fingerprint reader device. | The system authenticates the customer, according to the comparison of the captured fingerprint date. |

**(b)**

**Fig. 4** Initial set of advice for authentication. **a** Advice for the PIN feature. **b** Advice for the Fingerprint feature

**Id: ADV3**

**Pointcut: around range (SC01-4, SC01-5), ...**

| Code | User Action | System Response |
|---|---|---|
| ADV3-1 | - | The transaction handler starts the processing of a transaction. |
| **PROCEED** | | |
| ADV3-2 | - | An entry with the transaction information is logged to the overview of the completed transactions. |
| ADV3-3 | - | The transaction is removed from the transaction queue. |

**Fig. 5** Advice for the *DetailedTransaction* feature

operates over, although it is possible to use the `PROCEED` reserved word to introduce the behavior that follows the join point.

Notice that the *DetailedTransaction* advice, depicted in Fig. 5, also quantifies over different scenarios, which is indicated by the set of pointcuts of the *range* type. In this way, one can evolve the specification of transaction without breaking the original scenarios, although changing a step id of a scenario could demand a review of the pointcut clause of this advice. The semantics of *PROCEED* defines the right places where the steps delimited from a *range* pointcut will appear after evaluating the advice. Therefore, if a product is configured with the features *Withdraw*, *PIN*, and *Detailed-*

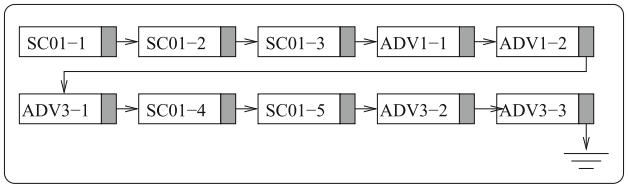



**Fig. 6** Sequence of steps expected to be found in products configured with *Withdraw*, *PIN*, and *DetailedTransaction* features

**Table 1** Configuration knowledge for the running example

| Feature expression | Transformations |
|---|---|
| Withdraw | selectScenario SC01 |
| PIN | evaluateAdvice ADV1 |
| Fingerprint | evaluateAdvice ADV2 |
| DetailedTransaction | evaluateAdvice ADV3 |
| **not** IdentifiedTransfer | evaluateAdvice ADV04 |
| LIMIT | bindParameter PL to LIMIT |

*Transaction*, a scenario with the sequence of steps in Fig. 6 will be configured.

The reader should have noticed that MSVCM scenarios and advice do not make explicit references to the *E-Finance* features. Indeed, an independent artifact, named configuration knowledge, is used to map feature expressions (which represent the presence conditions [10,19]) to transformations that generate a product-specific use case model. Three transformations were initially proposed by MSVCM [8]:

- **selectScenario scId** adds the scenario identified by *scId* to the set of scenarios of the resulting product.
- **evaluateAdvice advId** evaluates the advice identified by *advId*. All selected scenarios that match the pointcut clause of the advice are modified.
- **bindPartameter p to f** resolves the parameter *p*, replacing each occurrence of *p* in the scenarios with a string representing the children selection of feature *f*. In the example, a parameter ($LIMIT$) is declared in the third step of scenario SC01 (see Fig. 3). This parameter abstracts over the possible options of the *LIMIT* feature. A similar notation is also supported by the PLUSS approach.

Using these transformations, Table 1 shows part of the configuration knowledge for the running example. The process of interpreting the configuration knowledge leads to the generation of the product-specific use case model. This process takes as arguments the SPL feature model, the SPL use case model, a product configuration (a valid configuration of features according to the feature model semantics), and the configuration knowledge.

Previously, we have discussed that evolving product line scenarios using annotative approaches is hard, mainly because introducing new variants of a feature (such as a new authentication mechanism) might require a review of all related scenarios [1]. This happens because an annotative approach for scenario specification does not properly modularize features. In the remaining of this paper, we first proceed with a systematic evaluation of this issue using different systems; and then, we investigate the costs and benefits of a good modularization with respect to the effort for performing two activities: deriving an SPL specification from the specification of individual products and evolving existing SPL specifications.

## 3 Study settings

In this section, we present an overview of our research design, which aims to better understand the benefits, drawbacks, and limitations of representing scenario variability using the annotative (PLUSS) and compositional approaches (MSVCM) discussed in the previous section. In order to answer our research questions (Sect. 3.1), we base our evaluation on different empirical methods. First, we analyzed the specifications of six product lines (Sect. 3.2) to assess quality attributes of SPL use case scenarios. Later, we designed and executed six controlled experiments to compare the effort to derive and evolve product lines specifications using PLUSS and MSVCM. After evaluating the results of these empirical studies, we found that (a) evolving the specifications of an SPL using MSVCM requires localized changes, (b) it is more expensive to derive SPL specifications using MSVCM, although (c) MSVCM reduces the time to evolve SPL specifications, and (d) this later benefit is only observed when the subjects have been well trained in the specific task of evolving SPL specifications.

### 3.1 Goal, questions, and metrics

The following sections overview our evaluation, discussing its goal, questions, and metrics.

#### *3.1.1 General goal and scope*

This evaluation aims to compare the compositional and annotative approaches for representing scenario variability. It focuses on comparing MSVCM [8] with PLUSS [26,27], mainly because both approaches deal with variability in textual descriptions of use cases, whereas Model Templates [17,19], VML4RE [3], and MATA [64,65] represent scenario variability using graphical notations. We relate these techniques to our research in Sect. 7.

To achieve our goal, the following criteria are considered:

- Modularity of feature specifications.
- Time required to derive SPL specifications (from the specification of individual products).
- Time required to evolve existing SPL specifications, according to sets of changing requests.

We target the point of view of SPL requirements analysts, involving graduate and undergraduate students, attending to an introductory course in SPL development, and we observe the processes of bootstrapping and evolving product line specifications. In summary, the following questions are investigated:

Q1. When compared to PLUSS, does MSVCM reduce scattering of SPL specifications?
Q2. When compared to PLUSS, does MSVCM reduce tangling of SPL specifications?
Q3. When compared to PLUSS, does MSVCM increase the time to derive SPL specifications?
Q4. When compared to PLUSS, does MSVCM reduce the time to evolve SPL specifications?

The investigation of Q1 and Q2 relies on two metrics that we have adapted [8] from [24]: *degree of scattering of features* (DoS) and *degree of tangling of scenarios* (DoT). According to Eqs. (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 $f$, the greater the chance of changing different scenarios when the specification of $f$ evolves. Note in Eq. (1) that $| S |$ denotes the cardinality of the set $S$.

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

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

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 (highest degree of cohesion) and 1 (lower degree of cohesion). The greater the DoT of a scenario $s$, the greater the change of changing $s$ when the requirements of a related feature change.

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

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

**Table 2** Result of the feature assignment for the PLUSS specification of the Withdraw Scenario on Fig. 2

| | Feature | Withdraw Scenario |
|---|---|---|
| 1 | Withdraw | 12 |
| 2 | PIN | 2 |
| 3 | Fingerprint | 2 |
| 4 | DetailedTransaction | 3 |

Note in Eqs. (2) and (4) that, in order 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 [8], assigning a step *st* to a feature $f$ if, and only if, the selection of $f$ triggers the configuration of *st*. Therefore, we assign a PLUSS step *st* to a feature $f$ when:

(a) The step *st* refers to the feature $f$ as a parameter
(b) The step *st* is directly related to the feature $f$
(c) The step *st* is surrounded by a scenario related to the feature $f$.

For instance, let us review the PLUSS specification of the *Withdraw Scenario* (Fig. 2). This scenario is related to the *Withdraw* feature. Therefore, according to (c), all steps of this scenario are assigned to this feature. In addition, based on the step colors, steps SC01-4(a) and SC01-5(a) are also assigned to the *PIN* feature, since they are directly related to it. For the same reason, steps SC01-4(b) and SC01-5(b) are assigned to the *Fingerprint* feature, and Steps SC01-6, SC01-9, and SC01-10 are assigned to the *DetailedTransaction* feature.

Therefore, to compute DoS and DoT of PLUSS specifications for each scenario, we tabulate the number of steps that are related to a specific feature, considering the mentioned heuristics. For instance, Table 2 shows the resulting data for the particular case of the PLUSS specification of the *Withdraw Scenario* on Fig. 2.

Then, based on the assignment of all scenarios' steps to features, we are able to compute both DoS and DoT. To automate this calculation, we implemented several R scripts [58] that calculate these metrics from such tabular data.

Regarding MSVCM, we have to consider the configuration knowledge (*ck*) to assign a MSVCM step (*st*) to a feature ($f$). First, there must exist a feature expression ($e$) in *ck* that refers to $f$, and the assignment occurs when

(a) a transformation associated with $e$ binds a parameter $p$, and the step *st* refers to $p$
(b) a transformation associated with $e$ selects a scenario $s$, and the step *st* is included in the steps of $s$

$$refer\ (Or(e_1,\ e_2))\ f = (refer\ e_1\ f) \vee (refer\ e_2\ f)$$
$$refer\ (And(e_1,\ e_2))\ f = (refer\ e_1\ f) \vee (refer\ e_2\ f)$$
$$refer\ (Not(e_1))\ f = refer\ e_1\ f$$
$$refer\ (FeatureRef\ f')\ f = f == f'$$

**Fig. 7** Semantics of the *refer e f* predicate

(c) a transformation associated with $e$ evaluates an advice $a$, and the step $st$ is included in the steps of $a$.

In MSVCM, feature expressions are formulae in propositional logic, having feature names as atoms. So, a feature expression (`Exp`) might be just a feature reference (`FeatureRef String`) or a formula involving logical operators such as `Or(Exp, Exp)`, `And(Exp, Exp)`, and `Not(Exp)`. We also consider an two-arity predicate, named `refer e f` (see equations in Fig. 7), which holds when the expression `e` refers to the feature `f`. This predicate is necessary to assign steps to a feature.

Applying these heuristics to the configuration knowledge of Table 1 leads to the following: All steps of scenario SC01 (Fig. 3) are assigned to the *Withdraw* feature, all steps of advice ADV01 (Fig. 4a) are assigned to the *PIN* feature, all steps of advice ADV02 (Fig. 4b) are assigned to the *Fingerprint* feature, and all steps in the specifications that refer to the *LIMIT* parameter are related to the *LIMIT* feature. In this way, we could also tabulate, for each scenario and advice, the number of steps that are related to a specific feature, and then compute both DoS and DoT using the same scripts that compute these metrics for PLUSS.

Considering the effort analysis, to investigate questions Q3 and Q4, we conduct several controlled experiments, as detailed in Sect. 5. We consider the average time in minutes as the metric to verify whether MSVCM increases the time to derive SPL specifications from a set of individual product specifications (question Q3). In a similar way, we consider the average time in minutes as the metric to investigate whether MSVCM reduces the time to evolve SPL specifications accordingly to a set of change scenarios (question Q4).

All experiments that investigate questions Q3 and Q4 follow a *Latin Square Design* [12,50], in order to simultaneously block both the behavior effects of analysts (such as experience) and the characteristics (complexity level, kinds of features, etc.) of different product lines. In such a design, the number of treatments being compared, the number of row factor's levels, and the number of column factor's levels are the same, and the experiment runs resemble matrix arrangements where rows and columns accommodate the block factors (analysts and product lines), and the Latin letters component represent the treatments. Figure 8 illustrates a $3 \times 3$ Latin Square. It can be noted that the treatments appear only once in a given row or column.

| | C1 | C2 | C3 |
|---|---|---|---|
| R1 | A | B | C |
| R2 | B | C | A |
| R3 | C | A | B |

**Fig. 8** Latin square arrangement of order 3

### 3.2 Target product lines

We investigate the research questions using SPLs from different domains and available from distinct sources (books, technical reports, papers, calls for special issues, thesis, and so on). In this section, we briefly describe them. Notice that we have specified the SPLs in different ways. For instance, most product lines were specified by students. Nevertheless, the resulting specifications were revised, so as to assure their conformance to the original specifications and to the adequate practices of each technique.

#### *3.2.1 The security module of a smart home (SmartHome)*

This product line deals with different types of issues (such as intrusion and fire detection) in smart home environments. The original specification of this product line, which has been used in different studies, is from [56] and [2].

#### *3.2.2 Conference management product line (EasyChair)*

Product line of systems that help program chairs to manage the process of receiving submitted papers, assigning papers to reviewers, resolving review conflicts, and preparing the conference proceedings. The original specification of this product line is based on existing conference management systems [13,37].

#### *3.2.3 Mobile media product line (MobileMedia)*

SPL for applications that render photo, music, and video on mobile devices [30]. It has been used in several studies, mainly because it is well documented and several of its assets are easily available.

#### *3.2.4 Car crisis product line (CarCrisis)*

Product line that aims to facilitate coordination of activities and information flow between stakeholders that work together to handle a crisis. The original specification was proposed as a common case study for aspect-oriented modeling techniques [44].

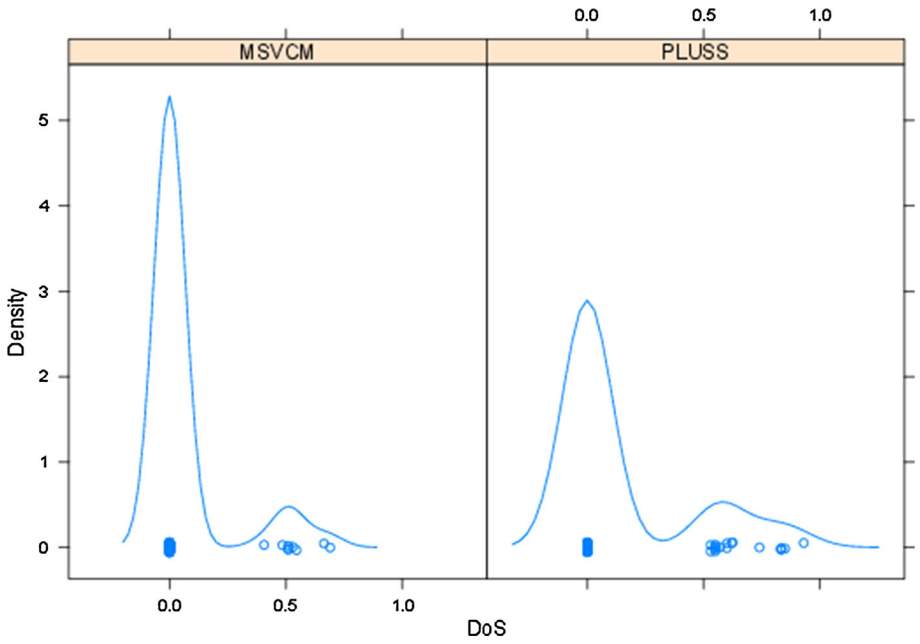



**Fig. 9** Density plot of DoS for the six SPL specifications

*3.2.5 The eShop product line (eShop)*

This product line, initially detailed by Pohl and Metzger [57], allows customers to buy goods through a Web site. The original specification was used as the motivating study of a panel about testing product lines.[1]

*3.2.6 E-Finance product line (HomeBanking)*

Product line inspired by the *E-Finance Case Study* [51]. The scope of this product line comprises banking services, such as transactions that allow a customer to withdraw and transfer money, as well as to manage stocks.

In what follows, we present the main findings of our investigation as well as more details about the research settings—in particular the experience of the participants and the instruments we used for data collection. Since these settings are not exactly the same for the different phases of our research, to avoid confusion we prefer not to present them in a single section.

## 4 Modularity analysis

To investigate Q1 and Q2, we evaluated the degree of scattering and degree of tangling of each aligned specification of the product lines mentioned in the previous section. Here, aligned specifications mean that, for each target system, we are able to generate the same set of products from both PLUSS and MSVCM specifications. The subjects responsible for specifying the SPL using both techniques were undergraduate students with basic SPL knowledge and who have received introductory classes discussing both techniques. Next we selected, revised, and aligned the most suitable specifications for each target system (considering both techniques). Finally, we computed the metrics DoS and DoT for these specifications as detailed in Sect. 3.1.

**Table 3** DoS of features specified using MSVCM and PLUSS

| Target system | Feature | MSVCM | PLUSS |
|---|---|---|---|
| HomeBanking | Operation Limit | 0.55 | 0.62 |
| HomeBanking | DetailedTransaction | 0.48 | 0.83 |
| MobileMedia | UserDefinedFolders | 0.66 | 0.85 |
| MobileMedia | Multiple Recipients | 0.41 | 0.60 |
| SmartHome | Register Inhabitant | 0.69 | 0 |
| SmartHome | Request Access to Home | 0.51 | 0 |
| SmartHome | Password | 0.51 | 0.53 |
| SmartHome | Fingerprint | 0.51 | 0.53 |
| SmartHome | Home State | 0.53 | 0.55 |

Note that this table presents only the features with positive values of DoS in MSVCM

The density plot of Fig. 9 shows the DoS of features obtained from the target systems specifications. Note that the specifications of most features are well localized in both PLUSS and MSVCM, though the frequency of features with DoS greater than zero is higher in PLUSS than in MSVCM. Table 3 presents further details, showing all cases where DoS is positive in MSVCM. Note that MSVCM eliminates feature scattering in three SPL specifications (*CarCrisis*, *EasyChair*, and *eShop*); in the other ones, it reduces the DoS of several features.

[1] This panel occurred at the 10th Software Product Line Conference.

It was not possible to completely eliminate feature scattering in MSVCM (see Table 3), for reasons related to parameterization, design choices, and the crosscutting nature of certain features. For example, the variation points required by the *Operation Limit* feature are specified as parameters and appear in two different scenarios of the *HomeBanking* system. A reference to this parameter is shown in the third step of the *Withdraw* scenario in Fig. 3. A similar reference also appears in the *Transfer Amount Between Accounts* scenario specification, in such a way that the *Operation Limit* feature is assigned to one step in each of the mentioned scenarios (totalizing two steps assigned to this feature). Since the MSVCM specification of *HomeBanking* comprises five scenarios and seven advice, we have $|S| = 12$ in Eq. (1) and $DoS(OperationLimit) = 0.545$. For a similar reason (parameterization), we could not eliminate the scattering of the *Home State* feature in the MSVCM specification of *SmartHome*.

As there exist different ways to specify the *DetailedTransaction* feature using MSVCM advice, to avoid bias in favor of MSVCM, differently from what was illustrated before, the evaluation presented here specifies the *DetailedTransaction* feature using two advice: one for starting transactions (a single-step advice) and another for committing and finishing transactions (a two-step advice).

Therefore, the specification of the *DetailedTransaction* feature comprises a total of three steps distributed in two advice (from a total of 12 scenarios and advice of the whole specification). This design scatters the specification of DetailedTransaction, leading to a DoS of 0.48 in MSVCM (see Table 3). Differently, both *UserDefinedFolders* and *Multiple Recipients* features of *MobileMedia* require two alternative advice, which should be evaluated when those features are selected or not. That is, the following entries are present in the *MobileMedia* configuration knowledge:

| Feature expression | Transformations |
|---|---|
| UserDefinedFolders | selectScenario SC06 |
| UserDefinedFolders | evaluateAdvice ADV05 |
| **Not**(UserDefinedFolders) | evaluateAdvice ADV06 |
| Multiple Recipients | evaluateAdvice ADV03 |
| **Not**(Multiple Recipients) | evaluateAdvice ADV04 |

As a consequence, the *UserDefinedFolders* feature is assigned to scenario SC06 (a three-step scenario), advice ADV05 (a three-step advice), and ADV06 (a single-step advice); while the *Multiple Recipients* feature is assigned to advice ADV03 (a three-step advice) and ADV04 (a single-step advice). The *MobileMedia* specification comprises 13 scenarios, so that these multiple assignments yield $DoS(UserDefinedFolders) = 0.66$ and $DoS(Multiple\ Recipients) = 0.41$ in MSVCM.

| Code | User Action | System Response |
|---|---|---|
| SC01-1 | The home owner selects the register inhabitant option in the security configuration menu. | The system shows the inhabitant personal form. |
| SC01-2 (a) | The home owner fills in the inhabitant personal form and selects the proceed option. | The system requests the inhabitant password (and password confirmation) for getting access to the home. |
| SC01-2 (b) | The home owner fills in the inhabitant personal form and selects the proceed option. | The system requires the inhabitant fingerprint to capture it. |
| SC01-3 (a) | The new inhabitant fills in the password (and password confirmation) for home access. | The system requests the home owner confirmation password. |
| SC01-3 (b) | The new inhabitant puts her finger in the fingerprint capture device. | The system gets the new inhabitant fingerprint information and asks the home owner to proceed. |
| SC01-4 * | The home owner selects the proceed option. | The system requests the home owner confirmation password. |
| SC01-5 | The home owner fills in the configuration password and selects the proceed option | The system registers the new inhabitant, allowing she to access the smart home. |

Legend: Password | Fingerprint

**Fig. 10** PLUSS specifications of Register Inhabitant

Finally, the remaining feature scattering of *SmartHome* is a consequence of interactions between the *Register Inhabitant* and *Request Access to Home* features with the Password and Fingerprint features. It is interesting to point out that modularizing these features as advice reduces tangling but increases the degree of scattering. For a better clarification, consider the PLUSS scenarios shown in Figs. 10 and 11. The DoS of the *Register Inhabitant* and *Request Access to Home* in PLUSS is zero. These features are completely localized in the PLUSS specification of *SmartHome*, even though they are entangled with the *Password* and *Fingerprint* features.

Using MSVCM advice, we are able to remove the aforementioned tangling, separating common behavior and variant behavior. This is depicted in Figs. 12, 13, and 14, which show the MSVCM specification for registering inhabitants. We use a two-step scenario (SC01) for the common steps and two advice for representing each variation related to the *Password* (the two-step advice ADV01) and *Fingerprint* (the three-step advice ADV02) features. Nonetheless, this separation increases the DoS of the *Register Inhabitant* feature, since its specification is scattered throughout SC01 (two steps), ADV01 (two steps), and ADV02 (two steps). See the configuration knowledge in Table 4. The whole MSVCM specification of *SmartHome* comprises 17 scenarios and advice. Altogether, the DoS of the *Register Inhabitant* features equals 0.69.

| Code | User Action | System Response |
|---|---|---|
| SC05-1 (a) | The inhabitant requests access to the home at the main entrance. | The system requests the user to pass her smart card. |
| SC05-1 (b) | The inhabitant requests access to the home at the main entrance. | The system requests the user to put her finger in the fingerprint reader. |
| SC05-2 (*) | The inhabitant pass the smart card in the card reader. | The system requests the user to inform her password to access the smart home. |
| SC05-3 (a) | The inhabitant fills in her password. | The system verifies whether the password is valid or not. |
| SC05-3 (b) | The inhabitant puts her fingerprint in the fingerprint reader. | The system verifies whether the fingerprint is valid or not. |
| SC05-4 | - | The system displays the message of success authentication. |
| SC05-5 | - | The system opens the main entrance and register the occurrence. |
| SC05-6 | - | After $TimeToClose$ seconds, the main entrance is closed. |

Legend: Password | Fingerprint

**Fig. 11** PLUSS specifications of Request Access to Home

**Id: SC01**

| Code | User Action | System Response |
|---|---|---|
| SC01-1 | The home owner selects the register inhabitant option in the security configuration menu. | The system shows the inhabitant personal form. **@registerInhabitant** |
| SC01-2 | The home owner fills in the configuration password and selects the proceed option | The system registers the new inhabitant, allowing he to access the smart home. |

**Fig. 12** MSVCM scenario of Register Inhabitants

**Id: ADV1**

**Pointcut: after @registerInhabitant**

| Code | User Action | System Response |
|---|---|---|
| ADV1-1 | The home owner fills in the inhabitant personal form and selects the proceed option. | The system requests the inhabitant password (and password confirmation) for getting access to the home. |
| ADV1-2 | The new inhabitant fills in the password (and password confirmation) for home access. | The system requests the home owner confirmation password. |

**Fig. 13** MSVCM advice of Register Inhabitants with Password

The MSVCM specification of the *Request Access to Home* feature is scattered by the same reason, leading to a $DoS(RequestAccesstoHome) = 0.51$. Furthermore, we could not completely localize the specifications of the *Password* and *Fingerprint* features, since they need different

**Id: ADV2**

**Pointcut: after @registerInhabitant**

| Code | User Action | System Response |
|---|---|---|
| ADV2-1 | The home owner fills in the inhabitant personal form and selects the proceed option. | The system requires the inhabitant fingerprint to capture it. |
| ADV2-2 | The new inhabitant puts her finger in the fingerprint capture device. | The system gets the new inhabitant fingerprint information and asks the home owner to proceed. |
| ADV2-3 | The home owner selects the proceed option. | The system requests the home owner confirmation password. |

**Fig. 14** MSVCM advice of Register Inhabitants with Fingerprint

**Table 4** Some items of the *SmartHome* configuration knowledge

| Feature expression | Transformations |
|---|---|
| Register Inhabitant | selectScenario SC01 |
| **And**(Register Inhabitant, Password) | evaluateAdvice ADV1 |
| **And**(Register Inhabitant, Fingerprint) | evaluateAdvice ADV2 |
| … | … |

**Table 5** Basic statistics for the DoT metric considering all projects

| Technique | VS | Mean DoT | Median DoT | Standard deviation DoT |
|---|---|---|---|---|
| PLUSS | 37 | 0.44 | 0.57 | 0.25 |
| MSVCM | 75 | 0.08 | 0.00 | 0.18 |

advice for each join point in the specification (one for registering inhabitants and another for requesting access to home). Therefore, these features have an *heterogeneous crosscutting behavior* [4], which prevent us from removing their scattering. Finally, ADV01 and ADV02 present some degree of tangling, as they specify the interactions involving the *Register Inhabitant* feature with the *Password* and *Fingerprint* features. However, it is important to note that this tangling is not motivated by mixing common and variant behavior; instead, it mainly occurs due to the interactions between mandatory and optional features.

Further, also regarding the tangling of scenarios, Table 5 presents basic statistics of the DoT metric. Note that we could observe a lower DoT when considering all target systems. This occurs because most scenarios of the target systems require at least one variation point, which in PLUSS are directly related to a feature. For instance, considering the *eShop* target system, only the PLUSS version of the *Create Wish List scenario* is related to just one feature. Nevertheless, the MSVCM specification of the *SmartHome* target system presents several scenarios and advice with values of DoT

higher than zero. This occurs due to the feature interactions discussed before. In addition, fine-grained variability, represented as parameterized steps, also causes tangling in MSVCM scenarios and advice. It is also important to discuss the higher number of scenarios (and advice) when using MSVCM. Considering all target systems, the vocabulary size (VS), that is, the number of scenarios and advice, in MSVCM has almost doubled, when compared to the same metric in PLUSS.

These results suggest that the *greater* the number of *homogeneous crosscutting features* and the number of variants for a scenario, the greater the benefits of applying MSVCM. In those cases, the MSVCM improvements in both feature localization and scenario cohesion become more evident. In fact, the expected benefit of MSVCM is to reduce the effort for evolving an SPL. The metrics suite employed here quantifies such an impact, since (a) if a scenario has a greater DoT, introducing a new alternative to its variant behavior requires more changes in the base specification, and (b) evolving features with high DoS requires changes in different scenarios. Although the observed improvements vary, in most of the cases, MSVCM increases scenario cohesion and reduces feature scattering. The next section presents a detailed investigation about the impact of these modular benefits with respect to the effort to derive and evolve SPL specifications.

## 5 Effort analysis

The previous empirical studies suggest that MSVCM leads to SPL specifications that are more modular than PLUSS. However, they do not provide evidence about the consequences of the improved modularity, in particular the costs to derive and evolve SPL specifications using both techniques. For that reason, we conducted controlled experiments comparing PLUSS with MSVCM to assess the effort required by these approaches to derive and evolve SPL specifications. In this way, we attempt to answer questions Q3 and Q4. More precisely, the following null hypotheses are investigated:

H1.0 There is no significant difference in the time required to **derive** product line specifications, by reason of the specification technique (PLUSS or MSVCM).

H2.0 There is no significant difference in the time required to **evolve** product line specifications, by reason of the specification technique (PLUSS or MSVCM).

With the first hypothesis, we want to investigate whether modularity imposes extra costs to the process of deriving an SPL specification from the specification of individual products. Note that, to conceive a modular specification, it is necessary to experiment with different design alternatives—until a proper decomposition strategy is found. Moreover, a modular design often leads to a large number of small modules (in MSVCM, scenarios and advice) and requires an extra effort for specifying the composition rules that would be necessary during the product derivation. With the second hypothesis, we want to investigate whether modularity delivers one of its intended benefits, that is, minimizing the extra costs necessary to derive an SPL specification during the maintenance tasks.

**Table 6** Some figures about the product specifications used in the first experiments round

| | EasyChair | eShop |
|---|---|---|
| Number of features | 18 | 19 |
| Number of scenarios | 9 | 6 |
| Number of steps | 48 | 41 |

We use the *convenience sampling* method [67], where students taking a first course in SPL engineering participated as the subjects of the experiments detailed here—in order to achieve part of the requirements necessary to succeed in the course. Initially, we had planned just the two experiments detailed in Sect. 5.1. However, after analyzing the results, we decided to mitigate two threats for our initial conclusions: the reduced number of preparatory classes and the *ad hoc* procedure to collect the response variables. In order to do that, we executed two new rounds of experiments (Sects. 5.2, 5.3). In each round, we performed two experiments: one for evaluating the effort required to derive an SPL specification and another for investigating the effort required to evolve SPL specifications.

In fact, besides mitigating the mentioned threats, we also extended the second hypothesis, varying the kinds of change and the familiarity of the subjects with the task of evolving SPL specifications. Although the first round could not be conclusive due to the mentioned threats, it is still presented here in detail to explain the rationale for the second and third rounds of experiments; it also illustrates the execution procedure and the data analysis method that we used in all experiments.

### 5.1 First round

This round considered two product lines: one for the electronic commerce domain (*eShop*) and another for the conference management systems domain (*EasyChair*). Table 6 presents some figures of the input specifications, which were briefly mentioned in previous sections. We consider only that part of the specifications, to reduce the effort required to complete the planned activities. Thus, the design of the experiment controls the SPL factor, even though *eShop* and *EasyChair* belong to different domains.

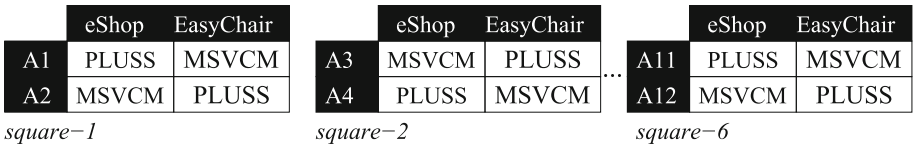



**Fig. 15** Layout of the experimental design

Here we investigate hypotheses H1 and H2 by means of two experiments. In the first one, students followed the extractive approach [48] for SPL development: They derive SPL specifications from scenarios of specific products. Differently, in the second experiment, students followed the reactive [48] approach for SPL evolution, where they had to evolve existing SPL specifications according to a set of change requests. In what follows, we detail the design, execution, and analysis of these experiments.

### *5.1.1 Preparatory classes, subjects, and tool support*

In the first round, twelve students playing the role of analysts participated in the experiments. They were undergraduate and graduate students with minimal industrial experience and basic knowledge about SPLs. Therefore, we did not use the experience of the subjects to stratify our sample—the students were not separated in distinct groups according to their experience. Nevertheless, as we discussed in Sect. 3.1 (and provide more details in Sects. 5.1.2, 5.1.3), the Latin square design we use in our experiments allowed us to control the participant experience by means of randomization and replication. In addition, to avoid the learning factor, we also assured that the participants did not have any previous experience with the target systems.

After taking introductory classes about SPLs in general, we started the experiment preparation by means of a two-hour class, where the subjects were first introduced to the concepts of each technique and to basic references about PLUSS and MSVCM. Then, during a two-hour practical class, a warm-up session detailed some of the activities the students should develop during the execution of the experiment, but focusing only on the extractive approach, since the idea was to familiarize the subjects with the techniques. In this first round of experiments, the students would derive and evolve the SPLs specifications using MS-Word templates. Therefore, at this stage, no specific tool was available to the execution of the experiments.

### *5.1.2 Assessment of SPL derivation*

Since twelve analysts were available for the experiment and the aim was to compare two treatments (PLUSS and MSVCM), we used a Latin square design of order two, as suggested by Fig. 15. Each row of a square represents an analyst, and the columns correspond to the product lines *eShop* and *EasyChair*. Randomization was independently applied to each square, following the procedures described in [12].

The execution was carried out during two laboratory sessions. In the first session, students derived the *eShop* SPL specification from existing product specifications, using either MSVCM or PLUSS. In the second session, they derived the specifications of the *EasyChair* SPL using a different technique from the first class, as prescribed by the Latin squares.

For each product line, we provided the following material to the students:

- feature model;
- two feature configurations; and
- two sets of scenario specifications, one for each product associated with the provided configurations.

During the experiment execution, several questions arose, and most of them related to the correct assignment of features to the elements of the input specifications. Answers to those questions were provided to all students. In addition, besides deriving the product line specifications, the subjects were also responsible for recording (using an MS-Word template) the time required to specify each SPL scenario. Based on this data, we could compute the time required to derive the complete product line input specification.

The box plot in Fig. 16a shows the effort (total time in minutes) to derive SPL specifications in PLUSS and MSVCM. Note that this box plot reveals an outlier in MSVCM, which corresponds to a student who did not complete the activity during the laboratory sessions. Since such a result does not correspond to an observation with the same experimental condition as the others, we replaced this outlier by the average time required to derive MSVCM specifications. Although being a naive imputation procedure, it is sufficient to avoid eliminating the remaining information for the respective Latin square, without compromising conclusions. Figure 16b shows the resulting box plot.

After removing the outlier, we proceeded with the data assessment, which leads to the analysis of variance (ANOVA) shown in Table 7. The data assumptions concerning additivity and homoscedasticity were satisfied [12]. Based on the ANOVA results and considering a $p$ value of 0.05 as a reference for declaring statistical significance, we conclude that there is not enough evidence to reject the null hypothesis that

> $H1_0$. *There is no significant difference in the time required to* **derive** *a product line, by reason of the specification technique (PLUSS or MSVCM)*

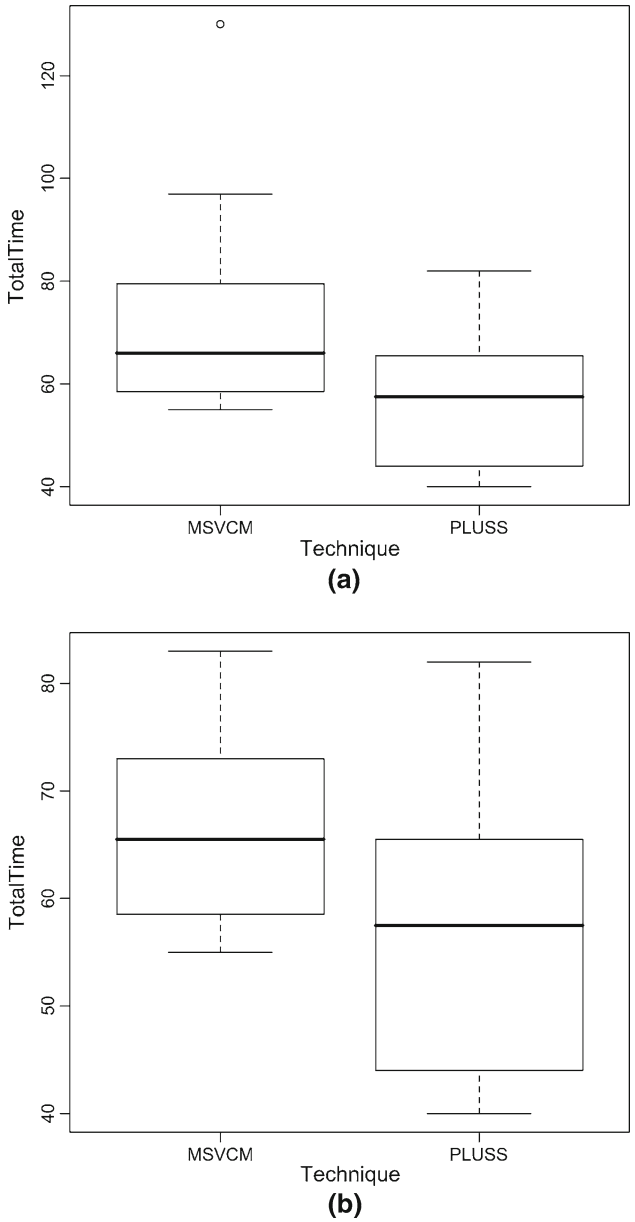



**Fig. 16** Box plots showing the effort to derive product lines in the first experiments round **a** Box plot with outliers. **b** Box plot after removing outliers

since the technique $p$ value was 0.06 (see Table 7). In summary, this experiment was not able to detect any significant difference in the average time required to derive SPL specifications using PLUSS and MSVCM.

### *5.1.3 Assessment of SPL evolution*

Similar to the previous investigation, the experimental design for this study was based on Latin squares of order two. In this case, ten analysts (a subset of the students who participated in the previous experiment) were randomly assigned to the rows of five squares.

**Table 7** Results of the ANOVA, regarding the time to derive a product line in the first experiments round

| | Df | Sum Sq | Mean Sq | $F$ value | $p$ value |
|---|---|---|---|---|---|
| Replica | 5 | 1358.21 | 271.64 | 2.34 | 0.1185 |
| Replica:Student | 6 | 640.25 | 106.71 | 0.92 | 0.5201 |
| SPL | 1 | 198.38 | 198.38 | 1.71 | 0.2206 |
| Technique | 1 | 513.37 | 513.37 | 4.42 | **0.0619** |
| Residuals | 10 | 1161.75 | 116.17 | | |

Here, the second controlled factor was two sets of change requests, which had been proposed to evolve a base *conference management* product line (*EasyChair*) specification. These change requests correspond to different types of evolution, such as adding new features to the product line, removing existing features, or introducing new options to alternative features. The two sets of change requests are available as supplemental material.[2] We request the analysts to evolve two aligned, reference specifications of *EasyChair*. One written in MSVCM and another in PLUSS. Since the subjects had already been introduced to the techniques under investigation, we considered seminar and warm-up sessions unnecessary.

Students carried on the activities during sessions of at most two hours each. Analysts had to evolve the aligned product line specifications according to the first set of change requests in the first class and according to the second set of change requests in the second class. Notice that, if a student applied the first set of CRs (SCR01) to the MSVCM specification, she had to apply the second set of CRs (SCR02) to the PLUSS specification (and vice versa). Therefore, the Latin squares were organized as shown in Fig. 17.

For each product line, we provided the following material to the students

- feature model;
- reference specifications written in MSVCM and PLUSS; and
- two sets of change requests. Each CR was specified in such a way that the students should be able to realize the impact on the base specifications.

In this activity, students were responsible for (a) figuring out the impact of the CRs, (b) evolving the reference specifications according to each CR, (c) evolving the configuration knowledge, when necessary, and (d) recording the time to evolve the specifications according to each CR.

The box plot in Fig. 18 shows the effort (total time in minutes) to evolve SPL specifications in PLUSS and MSVCM. Although we found two outliers in this response

[2] http://bit.ly/cQ7ZmU.

**Fig. 17** Layout of the experimental design

| | SCR01 | SCR02 |
|---|---|---|
| A1 | PLUSS | MSVCM |
| A2 | MSVCM | PLUSS |

*square−1*

| | SCR01 | SCR02 |
|---|---|---|
| A3 | MSVCM | PLUSS |
| A4 | PLUSS | MSVCM |

*square−2*

...

| | SCR01 | SCR02 |
|---|---|---|
| A09 | PLUSS | MSVCM |
| A10 | MSVCM | PLUSS |

*square−5*

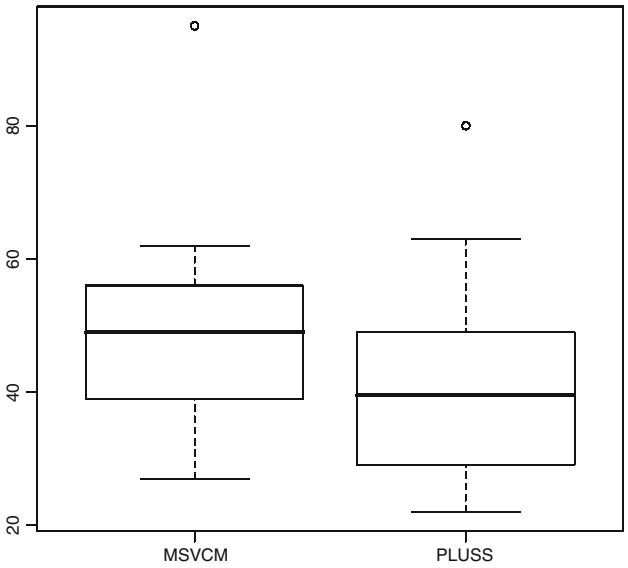



**Fig. 18** Box plot showing the effort required to evolve a base specification of *EasyChair*, using either PLUSS or MSVCM in the first experiments round

data, they were not excluded from our analysis—mainly because the corresponding values do not violate the two-hour constraint.

The analysis followed in this experiment was similar to the previous one. We first checked whether the response data satisfied all constraints of the regression model, but considering sets of change requests as treatments (instead of the SPL treatment of the previous experiment). Based on the ANOVA results and also considering a $p$ value of 0.05 as a reference, we conclude that there is not enough evidence for rejecting the null hypothesis that

> *$H2_0$. There is no significant difference in the time required to* **evolve** *a product line, by reason of the specification technique (PLUSS or MSVCM).*

since the technique $p$ value was 0.2071. So, this experiment of the first round was not able to detect any significant difference in the average time required to evolve SPL specifications using PLUSS or MSVCM (Table 8).

#### *5.1.4 Discussion*

The results of this first round contradicted our initial expectation. Previous results suggest that advanced techniques for modularizing features require more time to derive variant behavior (which should enforce the first hypothesis) [41],

**Table 8** Results of the ANOVA regarding the time to evolve a product line in the first experiments phase

| | Df | Sum Sq | Mean Sq | $F$ value | $p$ value |
|---|---|---|---|---|---|
| Replica | 4 | 691.70 | 172.93 | 1.45 | 0.3024 |
| Replica:Student | 5 | 763.75 | 152.75 | 1.28 | 0.3586 |
| ChangeSet | 1 | 4.05 | 4.05 | 0.03 | 0.8583 |
| **Technique** | 1 | 224.45 | 224.45 | 1.88 | **0.2071** |
| Residuals | 8 | 953.00 | 119.12 | | |

since one has to carefully think about how to structure variations. Additionally, since we have evidence that changes in MSVCM are more localized, we were expecting that this technique would reduce the time to evolve the specifications of an existing SPL. So, we decided to carry out a second round of experiments, which aims to generalize our results, identify the relevant factors, and mitigate some of the threats that could lead to the unexpected results of the first round. Particularly, we attempted to avoid the following potential threats:

- **Lack of training.** We had only two sessions for preparing the subjects to the mentioned activities. In the first, we introduced the techniques, whereas in the second class, we performed a warm-up. The lack of expertise of the analysts with the techniques could have influenced the results.
- **Non-representative CRs.** In the first round, just a few CRs required changes to the different parts of the specifications. Perhaps for that reason, we could not realize a significant difference in the effort to evolve SPLs regarding the *Technique* factor.
- **Imprecise data collection.** In the first round, analysts were asked to record the time to derive and evolve product lines using a MS-Word template. This could lead to imprecise data collection, so we decided to implement a tool that helps the students to record the effort of these activities.

### 5.2 Second round

The second round of experiments involved new subjects—students with similar characteristics to the participants of the first round (see Sect. 5.1.1). The goals of this round were analogous to the previous one: Two experiments were carried out in order to compare the performance of PLUSS and MSVCM, with respect to the derivative and evolutive situa-

**Table 9** Some figures about the product specifications used in the second experiments round

| | CarCrisis | MobileMedia |
|---|---|---|
| Number of features | 14 | 17 |
| Number of scenarios | 7 | 6 |
| Number of steps | 47 | 50 |

tions. In addition, the design of these experiments was similar to the ones in the first round. We highlight the main differences in what follows.

### *5.2.1 Preparatory classes, subjects, and tool support*

In this round, we allocated seven (2-h) sessions to prepare the students for executing the activities. In the first class, we explained the techniques. Then, we followed four warm-up sessions that focused on the derivative situations and aimed at familiarizing the subjects with the techniques as well as with an editing and time recording tool they should use during the activities. Indeed, during the warm-up sessions, the subjects proposed several improvements to this tool. After that, we presented the results to the participants (sixth session) and concluded the preparatory period with one warm-up session focusing on the tasks for evolving scenarios.

A more expressive number of analysts participated in this round—24 students got involved in the experiment regarding the extractive approach, that is deriving an SPL, whereas 22 students took part in the experiment about the reactive approach, evolving an SPL. All participants were computer science undergraduate students enrolled in an introductory SPL course. To avoid a learning effect, none of these participants was involved in the previous round.

As mentioned before, in order to mitigate errors during data collection, we implemented a Web-based tool that allows the subjects to derive and evolve SPLs using PLUSS and MSVCM. This tool records all SPL specifications and the time spent by the students to execute their activities. Moreover, it also computes the metrics discussed in Sect. 3.1, increasing our confidence about the acquired data.

### *5.2.2 Assessment of SPL derivation*

This round considered two different product lines: the car crisis product line (*CarCrisis*) and the mobile media product line (*MobileMedia*). Table 9 presents some figures about the sets of product specifications.

Before analyzing the results, using a process similar to the previous round, we first evaluated the students results so that we could discard incomplete and inaccurate specifications. We excluded specifications with missing scenarios as well as specifications whose scenarios were mostly wrong and that would require a significant effort to fix. Nevertheless, we considered in our analysis specifications presenting small mistakes, such as a wrong assignment of a step to a feature, mainly because we comprehend that such faults would not compromise the effort assessment. After excluding some of the specifications, our analysis considered only seven Latin squares (instead of the 12 squares that could be drawn from 24 subjects).

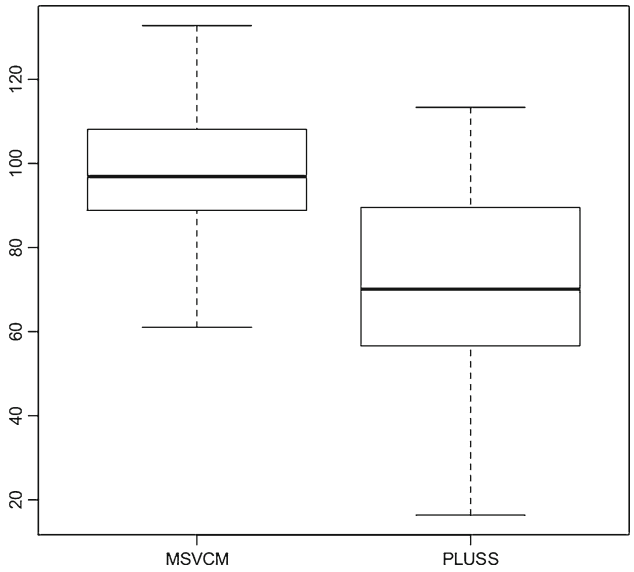



**Fig. 19** Box plot showing the effort required to derive SPLs, using either PLUSS or MSVCM in the second experiments round

**Table 10** Results of the ANOVA regarding the time to derive an SPL in the second experiments round

| | Df | Sum Sq | Mean Sq | *F* value | *p* value |
|---|---|---|---|---|---|
| Replica | 6 | 4055.07 | 675.84 | 3.04 | 0.0480 |
| Replica:Student | 7 | 6466.26 | 923.75 | 4.15 | 0.0152 |
| SPL | 1 | 85.20 | 85.20 | 0.38 | 0.5476 |
| Technique | 1 | 4597.74 | 4597.74 | 20.67 | **0.0007** |
| Residuals | 12 | 2669.69 | 222.47 | | |

Figure 19 and Table 10 show the results. Note that differently from our previous findings, here we obtained strong evidence for rejecting $H1_0$ ($p$ value $= 0.0007$). As a consequence, we could assume the *Technique* factor as being significant regarding the time to derive product line specifications from individual product specifications. Moreover, since we are evaluating just two techniques, we could infer which technique has a better performance by comparing their respective means. Since the difference between the mean time of MSVCM and PLUSS equals 26 min, we have evidence that PLUSS requires less effort than MSVCM for creating SPL specifications.

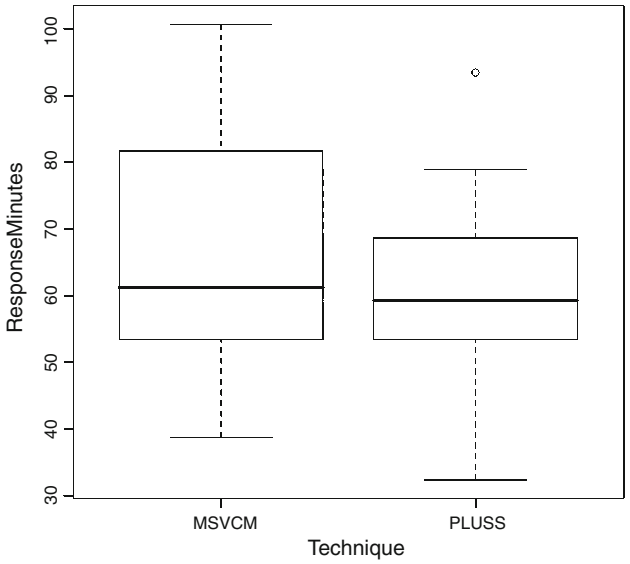



**Fig. 20** Box plot showing the effort required to evolve SPLs, using either PLUSS or MSVCM in the second experiments phase

**Table 11** Results of the ANOVA regarding the time to evolve an SPL in the second experiments round

| | Df | Sum Sq | Mean Sq | *F* value | *p* value |
|---|---|---|---|---|---|
| Replica | 7 | 2140.03 | 305.72 | 1.98 | 0.1304 |
| Replica:Student | 8 | 2447.92 | 305.99 | 1.99 | 0.1250 |
| CR | 1 | 936.90 | 936.90 | 6.08 | 0.0272 |
| **Technique** | 1 | 212.54 | 212.54 | 1.38 | **0.2598** |
| Residuals | 14 | 2157.34 | 154.10 | | |

#### *5.2.3 Assessments of SPL evolution*

In this experiment, students extended reference specifications of the Car Crisis product line (CarCrisis), according to two sets of change requests. Here, we aimed to evaluate our hypothesis *H2*: *The time to evolve SPL specifications using MSVCM is smaller than using the PLUSS approach*. In this round, some of the changes were crosscutting, having an effect on different parts of the reference specifications. Our analysis is identical to the previous round.

As discussed previously, 22 students were involved in this experiment (which should totalize 11 replicas—one replica for each Latin square). However, after discarding incomplete and wrong specifications, using the same criteria presented in the previous section, only eight replicas were considered in our analysis. Figure 20 and Table 11 show our findings. Notice in Table 11 that the *p* value for the *Technique* factor is 0.2598. Therefore, as in the first round, we could not reject $H2_0$, and differences regarding this factor could be motivated by chance. We believe this was motivated by the lack of confidence of some students regarding how to proceed to apply some types of changes in MSVCM. Particularly, some of the change requests required simple modifications only in the configuration knowledge, but subjects spent a long time to figure that out.

**Table 12** Some figures about the product specifications used in the third experiments round

| | MobileMedia | HomeBanking |
|---|---|---|
| Number of features | 17 | 16 |
| Number of scenarios | 6 | 6 |
| Number of steps | 48 | 60 |

In order to evaluate whether a compositional approach requires more training sessions to pay off for evolution tasks, we decided to conduct a new round of experiments. For this reason, in the third round, we dedicated more time to warm-up sessions for training students in the evolutive approach. After analyzing the results of the first two rounds, our new hypothesis was that modularity pays off for evaluation tasks only in situations where the analysts have been well trained in the tasks for evolving SPL specifications.

### 5.3 Third round

Similar to the previous rounds, the two experiments here aimed at evaluating both derivative and evolutive situations. However, comparing the second round, the major difference was that we had four more warm-up sessions, in order to better prepare the students to evolve SPL specifications.

#### *5.3.1 Preparatory classes, subjects, and tool support*

The settings here were similar to the second round. Particularly, we use the same tool of the second round to derive and evolve SPL specifications and record the time required to perform each activity.

Besides that, a new set of students took part on this experiment. In more detail, 16 analysts participated in this round, for both derivative and evolutive experiments. Clearly, this number of students allows only a reduced number of replicas in each experiment, when compared to the second round. Nevertheless, such a reduction did not compromise the quality of the data we obtained—the response data conform to the model constraints.

#### *5.3.2 Assessment of SPL derivation*

This round considered two different product lines: the mobile media product line (*MobileMedia*) and the home banking product line (*HomeBanking*). Table 12 shows some figures about the input set of product specifications.

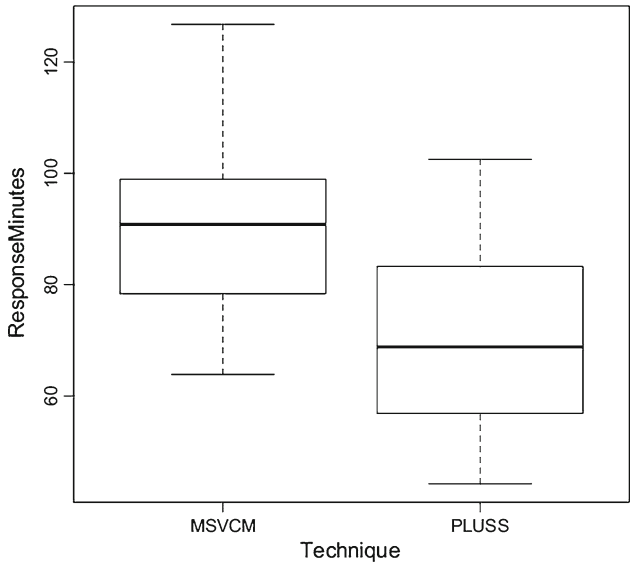



**Fig. 21** Box plot showing the effort required to derive SPLs, using either PLUSS or MSVCM in the third experiments round

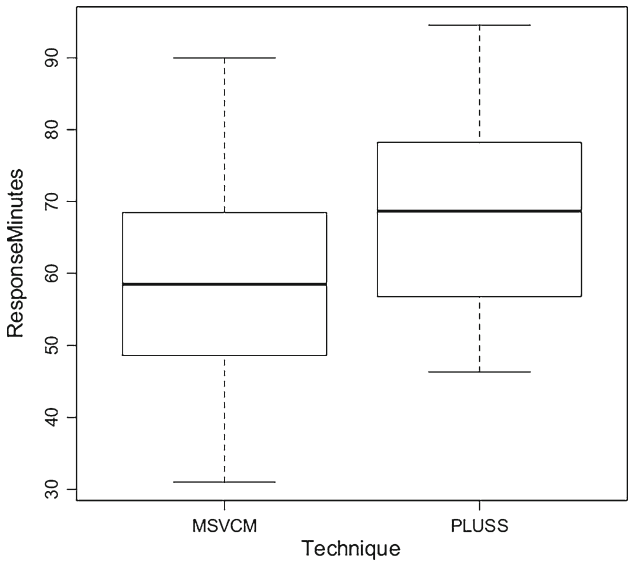



**Fig. 22** Box plot showing the effort required to evolve SPLs, using either PLUSS or MSVCM in the third experiments round

**Table 13** Results of the ANOVA regarding the time to derive an SPL in the third experiments round

| | *df* | Sum Sq | Mean Sq | *F* value | *p* value |
|---|---|---|---|---|---|
| Replica | 4 | 3243.65 | 810.91 | 7.91 | 0.0070 |
| Replica:Student | 5 | 1596.15 | 319.23 | 3.11 | 0.0747 |
| SPL | 1 | 620.14 | 620.14 | 6.05 | 0.0394 |
| **Technique** | 1 | 2573.54 | 2573.54 | 25.09 | **0.0010** |
| Residuals | 8 | 820.56 | 102.57 | | |

**Table 14** Results of the ANOVA regarding the time to evolve an SPL in the third experiments round

| | Df | Sum Sq | Mean Sq | *F* value | *p* value |
|---|---|---|---|---|---|
| Replica | 5 | 1621.09 | 324.22 | 3.21 | 0.0551 |
| Replica:Student | 6 | 3164.69 | 527.45 | 5.22 | 0.0112 |
| CR | 1 | 521.58 | 521.58 | 5.16 | 0.0465 |
| Technique | 1 | 380.14 | 380.14 | 3.76 | 0.0812 |
| Residuals | 10 | 1010.97 | 101.10 | | |

Although 16 students were involved in this experiment (which should lead to eight replicas), our analysis here considered only five replicas—mainly because we had to discard some specifications that were either incomplete or inaccurate.

We proceeded with our analysis as in previous rounds. Figure 21 and Table 13 show the results, after treating an outlier that was probably related to a problem during data collection. Similar to the second round, here we also obtained strong evidence for rejecting $H1_0$ and we could infer which technique has a better performance by comparing the average response of *MSVCM - PLUSS*, which equals 20.12 min. As a consequence, we get more evidence that PLUSS requires less effort than MSVCM for creating SPL specifications, which confirms our initial expectations.

### *5.3.3 Assessment of SPL evolution*

In this experiment, students evolved aligned specifications of the *E-Finance product line* (*HomeBanking*), according to two sets of change requests. We were expecting to confirm *H2*, since students were better prepared to this activity, and the results of the empirical studies detailed in Sect. 4 suggested that MSVCM improves modularity, when compared to PLUSS. Similar to the previous investigation, 16 students were involved in this experiment (which should lead to eight replicas). Nevertheless, after discarding some incomplete specifications, we only considered six replicas in our analysis, where each replica corresponds to a Latin square.

Our design, model, and analysis here were also similar to the previous round. Figure 22 and Table 16 show our findings. This time, the *p* value of the technique factor was 0.08, which according to Ramsey [59] suggests the technique factor as being relevant to the process of evolving SPL specifications. Thus, although we do not have enough evidence for rejecting $H2_0$, the response data of this experiment suggest to reject it. In addition, the differences in the average responses *MSVCM - PLUSS* is negative (−8.14 min), meaning that if the technique factor is really significant, we would have evidence that MSVCM has a better performance than PLUSS when considering SPL evolution with more trained subjects (Table 14).

Note that we only discarded incomplete specifications—where at least one scenario was missing, using the same

**Table 15** Measured effort (in seconds) of the evolutive experiment, third round

| Replica | Student | Replica:Student | CR | Technique | Effort | effort′ | effort″ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | CR01 | MSVCM | 4155 | 4155 | 4155 |
| 1 | 1 | 1 | CR02 | PLUSS | 3717 | 3717 | 3717 |
| 1 | 2 | 2 | CR01 | PLUSS | 4609 | **4916** | **5632** |
| 1 | 2 | 2 | CR02 | MSVCM | 3560 | 3560 | 3560 |
| 2 | 3 | 1 | CR01 | PLUSS | 4541 | **4843** | **5564** |
| 2 | 3 | 1 | CR02 | MSVCM | 2741 | 2741 | 2741 |
| 2 | 4 | 2 | CR01 | MSVCM | 5397 | 5397 | 5397 |
| 2 | 4 | 2 | CR02 | PLUSS | 4777 | **5393** | **5800** |
| 3 | 5 | 1 | CR01 | PLUSS | 3355 | **3578** | **4378** |
| 3 | 5 | 1 | CR02 | MSVCM | 3460 | 3460 | 3460 |
| 3 | 6 | 2 | CR01 | MSVCM | 3094 | 3094 | 3094 |
| 3 | 6 | 2 | CR02 | PLUSS | 4901 | 4901 | 4901 |
| 4 | 7 | 1 | CR01 | PLUSS | 5673 | 5673 | 5673 |
| 4 | 7 | 1 | CR02 | MSVCM | 3928 | 3928 | 3928 |
| 4 | 8 | 2 | CR01 | MSVCM | 4063 | 4063 | 4063 |
| 4 | 8 | 2 | CR02 | PLUSS | 3569 | 3569 | 3569 |
| 5 | 9 | 1 | CR01 | MSVCM | 3332 | 3332 | 3332 |
| 5 | 9 | 1 | CR02 | PLUSS | 3461 | 3461 | 3461 |
| 5 | 10 | 2 | CR01 | PLUSS | 2815 | 2815 | 2815 |
| 5 | 10 | 2 | CR02 | MSVCM | 2001 | 2001 | 2001 |
| 6 | 11 | 1 | CR01 | PLUSS | 2778 | **2963** | **3801** |
| 6 | 11 | 1 | CR02 | MSVCM | 1859 | 1859 | 1859 |
| 6 | 12 | 2 | CR01 | MSVCM | 5399 | 5399 | 5399 |
| 6 | 12 | 2 | CR02 | PLUSS | 4524 | 4524 | 4524 |

criteria of the previous experiments. However, here we observed that several PLUSS specifications do not satisfy all change requests, most likely because some of these change requests demanded scattered changes. Since these faults were not observed in the corresponding MSVCM specifications, we decided to carry out a (simple) sensitivity analyses on the data.

In the first sensitivity analysis, we first calculated the total number of PLUSS modifications ($m$) required by each set of change requests on the original specifications (the number of PLUSS modifications required by the first set of change requests equals 30, whereas the number of modifications required by the second set of change requests equals 35). Then, for each specification that satisfies $n$ modifications, we adjusted the *effort* (response data measured in seconds[3]) by:

$$effort' = effort + (m - n) \times \frac{effort}{n}$$

In this case, if the specification satisfies all modifications, effort′ equals *effort*. Differently, if the specification satisfies 31 of the 35 expected modifications of the second set of change requests and considering the response data equals 4777 s, the resulting effort (effort′) equals 5393 s. Table 15 presents the measured data as well the adjusted effort (in bold face) for this experiment.

**Table 16** Results of the ANOVA regarding the time to evolve an SPL in the third experiments round, but considering an adjusted response data

| | Df | Sum Sq | Mean Sq | $F$ value | $p$ value |
|---|---|---|---|---|---|
| Replica | 5 | 1936.86 | 387.37 | 4.02 | 0.0293 |
| Replica:Student | 6 | 3150.33 | 525.06 | 5.44 | 0.0096 |
| CR | 1 | 585.75 | 585.75 | 6.07 | 0.0334 |
| Technique | 1 | 627.64 | 627.64 | 6.51 | **0.0288** |
| Residuals | 10 | 964.51 | 96.45 | | |

Finally, to conclude the first sensitivity analysis, we performed an ANOVA test, but now considering the adjusted effort as response variable. In this case, we obtained a strong evidence that we should reject $H2_0$.

The second sensitivity analysis considered the standard deviation ($\sigma$) of the response data to adjust the measured effort in seconds. For this experiment, the standard deviation of the response data equals 1024 seconds. It is important to

[3] Actually, the effort was collected in milliseconds. However, to facilitate the reader to comprehend our results, we discuss our findings in this paper either in seconds or in minutes.

note that we adjusted only the measure data of the specifications that did not satisfy all change requests, according to the formula:

$$effort'' = effort + \sigma$$

Table 15 presents the results of this adjustment. Again, we conducted an ANOVA test with respect to the adjusted effort and obtained a $p$ value of 0.0046 for the technique factor.

### 5.4 Meta-analysis

In this section, we detail the results of a meta-analysis that combines the results of the three rounds of experiments. According to Horthon and Everitt, meta-analysis is a quantitative procedure for increasing the confidence of systematic reviews [29]. Such a procedure combines statistical results from different studies and has become increasingly popular within the field of evidence-based medicine. Although the experiments described in this paper were statistically designed and analyzed for the purpose of this research, we explore combining their results in a meta-analysis approach computing their effect sizes and estimating their summary effect and heterogeneity [11].

There are different models for performing a meta-analysis, as discussed in [11] and [29]. Here, we use the *fixed-effects* model, mainly because our goal is to investigate whether a treatment (MSVCM or PLUSS) had an impact on the response variable (effort to derive or evolve SPL specifications) and the number of investigated studies is small [29,47]. Indeed, a meta-analysis consists of first computing the effect size and the weight factor for each study. Then, we compute the summary effect of the treatment according to the meta-analysis model. Finally, the results of the meta-analysis are discussed in terms of the summary effect size and the heterogeneity of the studies, and the resulting statistics are typically represented using forest plots. A detailed introduction about how to conduct a meta-analysis could be find in [11] (Table 16).

We computed the effect size of each study using the raw mean difference ($D$), as detailed in [11,15]. Let $\overline{X_1}$ and $\overline{X_2}$ be the sample means of two independent groups. Then, the difference in the sample means is $D = \overline{X_1} - \overline{X_2}$. In addition, let $S_1$ and $S_2$ be the sample standard deviations of the two groups and $n_1$ and $n_2$ be the respective sample population's size. Under the assumption that the standard deviations are the same for both populations, we can calculate the variance of $D$ ($V_D$) according to Eqs. (5) and (6), and the standard error of $D$ is $SE_D = \sqrt{V_D}$.

$$V_D = \frac{n_1 + n_2}{n_1 n_2} S^2_{\text{pooled}}, \text{ where} \tag{5}$$

$$S_{\text{pooled}} = \sqrt{\frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}} \tag{6}$$

Besides, the weight assigned to each study in a *fixed-effect* meta-analysis is given by $W_i = \frac{1}{V_{Y_i}}$, where $V_{Y_i}$ is the within-study variance for the $i$th study. The weighted mean $M$, the variance of the summary effect $V_M$, and the estimated standard error of the summary effect ($SE_M$) are computed according to Eqs. (7), (8), and (9), considering that $Y_i$ is the effect size for the $i$th study.

$$M = \frac{\sum_{i=1}^{k} W_i Y_i}{\sum_{i=1}^{k} W_i} \tag{7}$$

$$V_M = \frac{1}{\sum_{i=1}^{k} W_i} \tag{8}$$

$$SE_M = \sqrt{V_M} \tag{9}$$

We are also interested in estimating the real differences in the effect size (that is, the heterogeneity). These differences could not be explained by the within-study error. There are several statistics for testing the heterogeneity among different studies, though most of them are based on $Q$—a statistic that is equivalent to the weighted sum of squares [11]. Our analysis of heterogeneity is based on the $I^2$ statistic [see Eqs. (10), (11), and (12)], since it is not affected by the number of studies [11]. In fact, the $I^2$ statistic gives *the proportion of variance that is due to the real differences in effect size*.

$$Q = \sum_{i=1}^{k} \left( \frac{Y_i - M}{S_i} \right)^2 \tag{10}$$

$$I^2 = \left( \frac{Q - df}{Q} \right) \times 100\,\% \tag{11}$$

$$df = k - 1 \tag{12}$$

Figures 23 and 24 present the forest plots showing the results of the meta-analysis, regarding the derivative and evolutive situations. This kind of plot is commonly used in meta-analyses, mainly because it graphically depicts the statistics (effect size and weight assigned to each study, summary of the effect size, variance of the summary effect, and so on) mentioned before. In our case, the first three lines of the forest plots represent a round of experiments (regarding either the derivative or evolutive situations), while the fourth line represents the summary of the meta-analysis. The first columns present information about each study—a short description, the mean, and the standard deviation for each technique. Next, there is a chart representing the effect size of each study. Indeed, the effect size of each study is represented as a square of size proportional to the weight assigned to each study and centered at the computed effect size. The

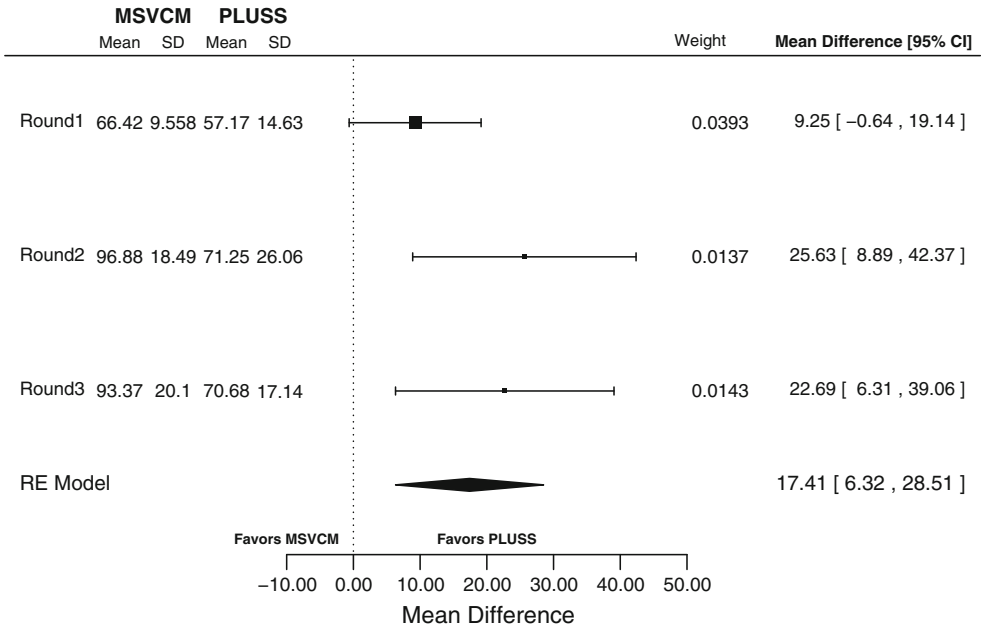



**Fig. 23** Forest plot with the results of the meta-analysis for the derivative situation

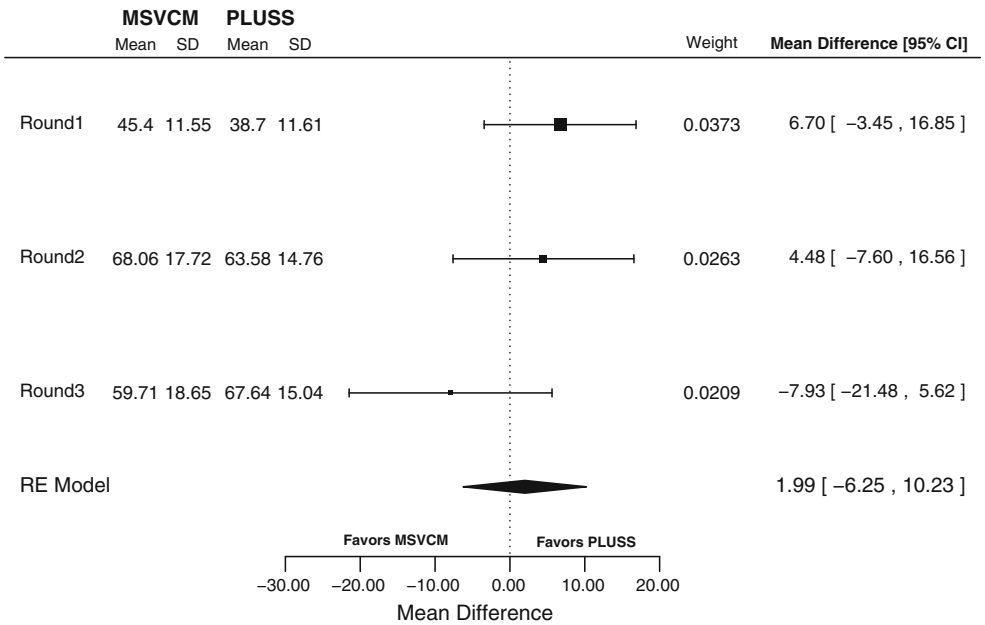



**Fig. 24** Forest plot with the results of the meta-analysis for the evolutive situation

last two columns of the forest plot show the weight, the raw mean difference, and the confidence interval (considering 95 % as reference). Finally, the diamond in the last line of the forest plot represents the overall summary effect size. The center of the diamond shows the point that estimates the summary effect size, whereas the diamond's width represents the limits, considering a 95 % confidence interval.

According to the forest plot in Fig. 23, the effect size of the three rounds of experiments are in favor of PLUSS, regarding the derivative situation—indeed, the summary effect size equals 17.41 min in favor of PLUSS. The 95% confidence interval is between 6.32 and 28.51 min. Note that comparing the effect size of the three rounds, the first leads to the lower effect size (9.25 min). We believe that this result might have been motivated by the lack of precision for data collection in the first round of experiments (as discussed in Sect. 5.1.4). This might also have increased the heterogeneity of the results ($I^2 = 46.72\,\%$). Therefore, the proportion of variance that is due to real difference is almost 50 %, with statistics $Q = 3.68$, $df = 2$, and $p$ value = 0.15.

According to the forest plot of Fig. 24, which considers the evolutive situation, we emphasize that only the third round reports an effect size in favor of MSVCM (in this case, a raw mean difference of −7.93 min or $-8'33''$). Considering the three investigations regarding the evolutive situation, the summary effect size equals 2.39 min, which is a value in favor of PLUSS. In this case, the 95 % confidence interval is in the range of −6.25 to 10.23 min. This investigation also reveals the presence of heterogeneity ($I^2 = 31.54\,\%$), with an estimation of $Q = 3.03$, two degrees of freedom, and $p$ value0.21 for the null hypothesis of homogeneity. We believe that this heterogeneity is mostly due to the preparatory classes we presented to the subjects in the third round of experiments.

### 5.5 Summary

In summary, our conclusion is that the benefits regarding better modularity provided by the compositional approach MSVCM have a payoff during evolution. However, this can only be achieved after an up-front investment in training to reduce the effort required to evolve SPL specifications. Besides that, we believe that localized changes could reduce errors when evolving specifications, although the investigation of this attribute was not considered as an hypothesis of this study. Finally, regarding the effort to derive a SPL specification, the additional costs related to the design of modular specifications in MSVCM might be amortized by the development of specific tools for helping the analysts. This is left for future work.

## 6 Threats to validity

*Internal validity* In the first round, we did not use an adequate instrument for measuring the time required to extract and evolve product line specifications. This motivated us to implement the tool support used in the second and third rounds. Such a tool measures the effort of the subjects to carry out their activities in milliseconds. We are confident that this tool collects the time of the activities consistently. Nevertheless, we realized a sole atypical measurement, where the student spent more than 3000 min to conclude his task. A data collection error might have occurred during this measurement. So, we decided to treat this outlier as described in Sect. 5.1.2, replacing this observation by the average time required to extract product lines in the specific technique.

*Conclusion validity* We rely on two assumptions to the measurement process: (a) The measured data (effort) should be a positive value and most of the collected data should be less than 120 min—since most of the students concluded their activities on time and (b) there must exist a strong correlation between the measured data *MD* and the difference *D* between the time in which the student concluded his(her) activity and the time in which he(she) started his(her) activity. The software used in the second and third rounds collected those moments. Figure 25 shows the density plot of the second (R2) and third rounds (R3), grouped by the extractive (P1) and evolutive situations (P2). It is possible to observe that the response data concentrate on the interval between 50 and 100 min, as expected. Further, we verified that the correlation coefficient between *MD* and *D* is 0.91, supporting our expectations.

In addition, we also investigated the accuracy with which our study measures the factors or situations under investigation, regarding both modularity of SPL specifications (Sect. 4) and the effort to create and maintain SPL specifications (Sect. 5). We used two metrics (degree of scattering and degree of tangling) for modularity measurement that have been proposed and validated [23] according to the Kitchenham, Pfleeger, and Fenton framework for software measurement validation [46] and used in a number of previous studies [8,24,25]. We could have used other metrics to measure scattering and tangling, such as *Concern Diffusion over Components* and *Concern Diffusion over Lines of Code* [31], but they lead to absolute values that might hinder the comparison between two specifications. Differently, DoT and DoS lead to normalized values (between 0 and 1), according to the recommendations for software measurement [46]. Regarding our effort analysis, we measured the average time to conclude the activities. This is a practical approach to compare the effort required by different techniques to solve a similar task and has also been used in other studies [20,22,52,61]. For the reasons presented above, we consider that we measured the two factors of interest (specification's modularity and effort) properly.

*External validity* There are some factors that might prevent the generalization of our results. First, one might argue that the target systems (Sect. 3.2) are not representative and, for this reason, the results could not be generalized for larger software product lines. As explained in Sect. 3.2, the target systems have been used in the literature and there was a conservative limit on the tasks size: Each task should be concluded in at most two-hour sessions. Second, the use of students as subjects in software engineering experiments has also been criticized [35]. Nevertheless, we agree with the opinion that the use of students is "well suited to investigate certain issues that do not require high levels of industrial experience" [6,45], which is the case of our investigation. We did not conduct a pretest to determine the experience of the subjects, but based on the characteristics of the undergraduate and graduate students at Federal University at Pernambuco, it was possible to generalize a typical profile for the participants (as discussed in Sects. 5.1.2 and 5.1.3). Besides that, we claim that this threat does not reduce the relevance of our findings, since the design of the experiments deals with the experience factor of the students, as discussed in Sects. 3, 5.1.2, and 5.1.3.

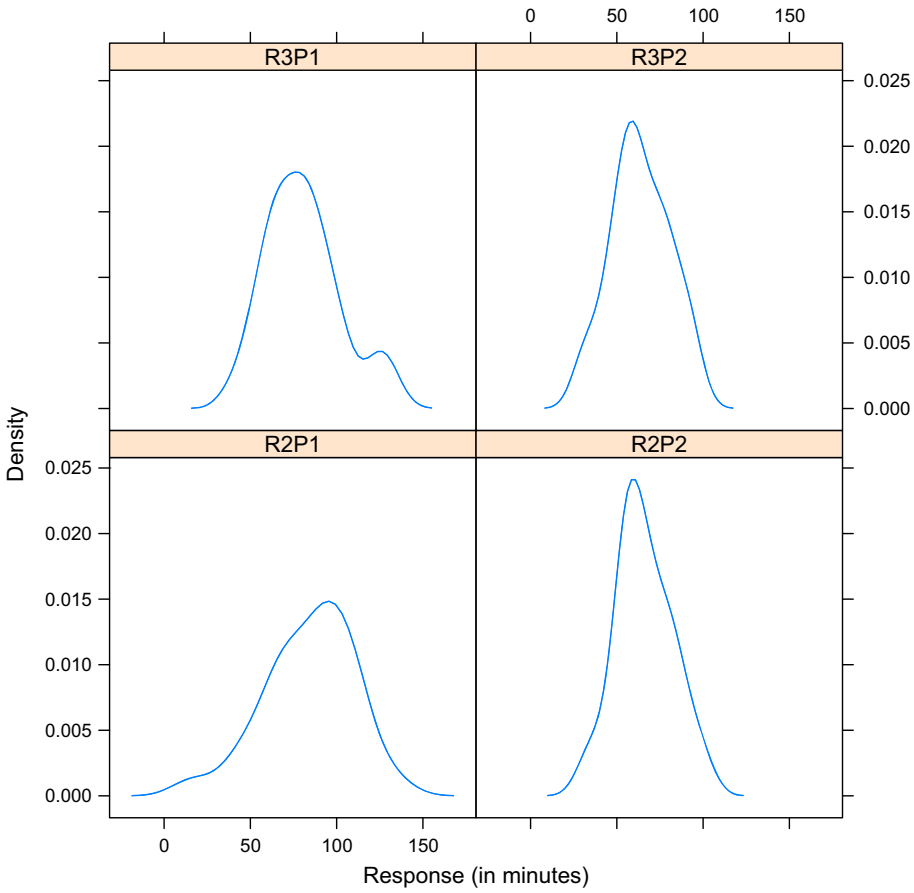



**Fig. 25** Density plot of the response data with respect to the second and third rounds of experiments

## 7 Related work

As detailed next, the investigation described in this paper is mainly related to use case scenario variability management, empirical studies in aspect orientation, and experimentation in software engineering.

### 7.1 Scenario variability management

Besides PLUSS and MSVCM, there exists other techniques for representing SPL variabilities in use case scenarios, as mentioned in Sect. 2. Here, we present an overview of those techniques—apart from MSVCM and PLUSS.

#### *7.1.1 PLUC*

PLUC (product line use cases) is an extension of use cases that allow system analysts to represent variability in requirements. In fact, PLUC was proposed as a mean to represent requirements in a suitable way to derive SPL test cases from use case scenarios.

Specifications written in PLUC represent variability by means of special tags, which are explicitly enclosed throughout the use case sections. Therefore, similar to PLUSS, one single asset describes both common and variant behavior—there is no separation between a base specification and the optional features that crosscut it.

Besides this initial similarity with PLUSS, PLUC presents more problems regarding the separation of concerns in product line development. These problems are derived from PLUC's approach for defining a SPL scope: It is defined by means of the variability tags, instead of using a selection of features to define a product. As a consequence, the scope (or the set of SPL products) has to be specified through different scenarios using particular tags, which make PLUC specifications hard to maintain. Even simple changes, such as introducing an alternative to a feature (which enlarges the SPL scope), might require changes in different scenarios. In a previous work, we also evaluated PLUC specifications empirically [9]. From that, we conjecture that running the experiments with PLUC, instead of PLUSS, would produce evidence more favorable to MSVCM.

#### *7.1.2 VML4RE*

VML4RE is a compositional approach for representing SPL variability in use cases and activity diagrams [3]. Therefore, similar to Model Templates, scenarios in VML4RE are specified using activity diagrams. Besides that, it is possible to

separate common and variant behavior of a scenario using a symmetric, compositional approach.

Therefore, regarding modularity, VML4RE and MSVCM are equivalent, since both techniques present a clear separation between common and variant specifications. However, apart from the notation used, these techniques differ with respect to several points of view.

First, MSVCM follows an asymmetric approach for composing scenarios and advice, which are proposed to represent, respectively, common and variant behavior. Differently, VML4RE was built upon a symmetric approach, in which both common and variant behavior are represented using the well-known notation of UML activity diagrams.

Another difference between these approaches is that MSVCM pointcuts are pieces of information of the advice construct, whereas pointcuts in VML4RE are specified in the configuration space. We consider the pointcut model of MSVCM more general, since it allows the composition of scenarios and advice to occur independently of the configuration process of a SPL member.

Finally, VML4RE requires one configuration item for each join point that is effected by a variant behavior—thus, the degree of quantification in VML4RE is particularly limited when compared to MSVCM.

### 7.2 Empirical studies in aspect orientation

Several studies have been proposed to assess the impact of aspect-oriented programming [24,25,30,32,33,36] and aspect-oriented requirements engineering (AORE) [8,14,60].

These studies rely on metrics for quantifying scattering and tangling of concerns. Some of them use absolute metrics, such as *Concern Diffusion over Components* and *Concern Diffusion over Lines of Code* [31], while others propose *degree of scattering* and *degree of tangling*, which are, respectively, normalized with respect to the number of concerns or components. We could have used absolute values for quantifying scattering and tangling, as in a previous work [9]. However, absolute values just reveal whether a feature is scattered or not—without any information about the degree of its scattering. In fact, this limitation hinders the comparison of modularity between different specifications, which justifies our decision to evaluate scattering and tangling using DoS and DoT [24].

As mentioned, empirical studies in AORE have also been conducted. First, Sampaio end colleagues [60] present a case study that compared four different AORE approaches in terms of time-effectiveness and the quality of their produced outcomes. Differently from our study, they were not interested in assessing the improvements regarding modularity and separation of concerns that an AORE approach could bring. In addition, their case study involved four requirements engineers, each one assigned to restructure the Health Watcher system [63] specifications using one of the evaluated AORE approaches. Since they did not block the *subject factor*, their conclusions could not be generalized and the observed variations might only be motivated by the differences between subjects or by the available tools used during the activity. In contrast, here we have replicated our experiment, blocked the subject factor, and conducted all activities related to the effort analysis in laboratory. This improves the confidence of our conclusions.

Besides that, we agree with Sampaio when he suggests that accuracy is an important property that should be investigated [60], but we postpone such a comparison in the context of our research to a future work. At first glance, investigating accuracy with the outcomes of our study seems to be straightforward. However, we should have proceeded differently during the executions of our experiments, if we had aimed to check accuracy. For instance, we should not have allowed the participants to make questions, nor answered those questions during the execution of the experiment.

Chitchyan et al. present a comparison between *Syntactic*- vs. *Semantic*-based approaches for AORE [14]. Their work focuses on comparing two properties: expressiveness and stability of the pointcut clauses of each approach. In contrast, here we compared two techniques: aspect-oriented one and non-aspect-oriented one, proposed for representing SPL variability in use case scenarios. Our comparison also has a different focus, which aims to understand the benefits and limitations of using an aspect-oriented approach for modularizing feature specifications. Regarding modularity, as explained before, we think that (a) any AORE approach investigated in [14] would perform as good as MSVCM (b) and RDL provides a more stable pointcut definition than MSVCM, as previously discussed. With respect to the expressiveness of the compositions, we have to investigate it further before taking any conclusion.

Finally, in a previous work [8], we have introduced MSVCM and attempted to compare it with the PLUSS approach. Here, we present a deeper comparison, investigating other case studies and relating modularization to the effort required to extract and evolve SPL specifications. To our knowledge, there is no other work that investigates those characteristics in the context of SPL development.

### 7.3 Experimental evaluation in software engineering

As discussed, there is a body of knowledge that brings evidence about the benefits of aspect-oriented constructs to improve modularity of source code, as well as requirements specifications. However, while conducting this research, we did not find enough evidence in the literature stating that modularizing crosscutting concerns using aspect constructs reduce the effort to evolve a software, for instance. As a con-

sequence, we could not relate our work to other studies that investigate questions similar to ours. The most likely cause is the lack of experimental evaluation of aspect-oriented software development. For that reason, we consider that our work provides a singular evaluation of the use of aspect orientation to modularize feature specifications.

Nevertheless, the response data present in [62] suggest that AOP reduces the time to implement and evolve a Web-based system. In that study, students were randomly organized in two groups. A control group using the Java programming language and another group using AspectJ to perform the assigned tasks. Indeed, this design uses one project, two techniques, and two groups of subjects—each group assigned to one technique. According to Basili [5], a design such as that is classified as *Replicated Project*—multiple teams developing activities related to a unique project. Since the subject factor was not controlled, the observed differences could have also been motivated by the subject factor.

Hanenberg and others investigate the hypothesis that AOP has a positive impact on the development time [38]. In their experiment, twenty students were assigned to execute nine tasks proposed to evolve a game in an incremental way, according to the introduction of different features (logging, synchronization, and so on). The students were divided into two groups: One group initially implemented the tasks using OOP and then developed the same tasks using AOP. The other group proceeded the execution of the experiment in the opposite order (first using AOP and then OOP). The response data revealed that (a) the technique is significant, (b) AOP reduced the effort to perform only two of the nine tasks, and (c) AOP takes significantly more time then OOP when considering the whole experiment as a single task. In addition, similar to our findings, they have also observed great differences related to the subject factor. Finally, the authors also point that a more intensive training in AOP would significantly change the results, which is one of the conclusions of our experiment.

Regarding the design of the experiment, the results present in [38] could have been influenced by the learning effect, since each subject performed the same tasks twice. Here, we attempt to minimize such an effect by asking students to perform different activities using each technique (PLUSS or MSVCM). Indeed, we followed an extension of the *Blocked Subject–Project* design [5] to evaluate the effort to extract and evolve SPL specifications, in which each subject uses both techniques in different projects. Perhaps due to the lack of experimentation in software engineering, Basili did not consider more elaborate designs in his classifications [5].

In fact, searching in the literature that surveys experiments in software engineering [21,39,66], we did not find examples of experiments using the Latin square design, even though this design is explained in [39]. To our knowledge, only [53] uses a design similar to ours. This enforces our experiment design as being an important contribution of this paper.

## 8 Summary

In this paper, we reported on the use of a multimethod approach to compare two techniques for specifying SPL usage scenarios (PLUSS and MSVCM). The goal of this comparison was to investigate whether a compositional approach (MSVCM) introduces extra costs for modularizing scenario specifications—we use PLUSS (an annotative-based technique) as a baseline. We first carried out an empirical study to quantify the modular benefits of MSVCM, and then, we conducted a number of experiments to estimate the extra costs for deriving and evolving SPL specifications.

The reason for conducting several rounds of experiments was that the first results appeared to contradict our initial expectation that a modular approach for specifying product lines would reduce the effort during maintenance tasks—one of the expected benefits of modularity. After reviewing the data collection and training procedures through the different rounds, we conclude that although MSVCM improves modularity, when compared to the non-compositional approach PLUSS, the response data obtained from the different experiments revealed that: (a) MSVCM requires more time to derive SPL specifications and (b) in order to reduce the time to evolve the SPL specifications, MSVCM requires more investments on training.

These results reinforce the need for other studies on product lines, feature modularity, and specially aspect-oriented software development, since several works suggest similar benefits as provided by MSVCM, even though the costs of using compositional techniques are still unclear. In this paper, we provide one deeper investigation about that, focusing on the use of aspect-oriented constructs to modularize feature specifications.

**Acknowledgements** This work was partially supported by the National Institute of Science and Technology for Software Engineering [INES (http://www.ines.org.br], funded by CNPq and FACEPE, Grants 573964/2008-4 and APQ-1037-1.03/08. The first author is supported by the Grants CNPq CT-INFO 17/2007.

## References

1. Alférez, M., Bonifácio, R., Teixeira, L., Accioly, P., Kulesza, U., Moreira, A., Araújo, J.a., Borba, P.: Evaluating scenario-based SPL requirements approaches: the case for modularity, stability and expressiveness. Requir. Eng. pp. 1–22 (2013). doi:10.1007/s00766-013-0184-5
2. Alferez, M., et al.: A model-driven approach for software product lines requirements engineering. In: Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE'2008), pp. 779–784. San Francisco, USA (2008)

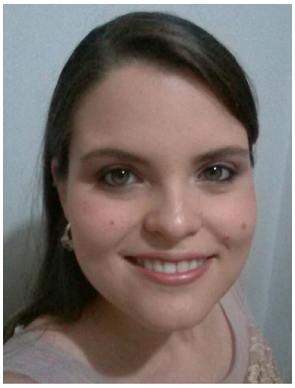
3. Alférez, M., et al.: Multi-view composition language for software product line requirements. In: 2nd International Conference on Software Language Engineering. Denver, USA (2009)
4. Apel, S., Leich, T., Saake, G.: Aspectual mixin layers: aspects and features in concert. In: 28th International conference on Software engineering, pp. 122–131. ACM, New York, NY, USA (2006). doi:10.1145/1134285.1134304
5. Basili, V.: The role of experimentation in software engineering: past, current, and future. In: Software Engineering, 1996, Proceedings of the 18th International Conference on, pp. 442–449 (1996). doi:10.1109/ICSE.1996.493439
6. Basili, V., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Trans. Softw. Eng. **25**(4), 456–473 (1999). doi:10.1109/32.799939
7. Bertolino, A., Gnesi, S.: Use case-based testing of product lines. In: 9th European Software Engineering Conference (ESEC/FSE), pp. 355–358. ACM, New York (2003). doi:10.1145/940071.940120
8. Bonifácio, R., Borba, P.: Modeling scenario variability as crosscutting mechanisms. In: AOSD '09: Proceedings of the 8th ACM International Conference on Aspect-Oriented Software Development, pp. 125–136. ACM, New York, NY (2009). doi:10.1145/1509239.1509258
9. Bonifácio, R., Borba, P., Soares, S.: On the benefits of variability management as crosscutting. In: Early Aspects Workshop at AOSD. Brussels, Belgium (2008)
10. Borba, P., Teixeira, L., Gheyi, R.: A theory of software product line refinement. Theoretical Computer Science 455(0), 2–30 (2012). doi:10.1016/j.tcs.2012.01.031. ¡ce:title? International Colloquium on Theoretical Aspects of Computing¡ce:title¿
11. Borenstein, M., Hedges, L., Higgins, J., Rothstein, H.: Introduction to Meta-Analysis. Statistics in Practice. Wiley, New York (2009)
12. Box, G., Hunter, J., Hunter, W.: Statistics for Experimenters: Design, Innovation, and Discovery, 2nd edn. Wiley-Interscience, New York (2004)
13. Chaudhuri, S.: Microsoft conference management system. on-line (2009). http://cmt.research.microsoft.com/cmt/
14. Chitchyan, R., Greenwood, P., Sampaio, A., Rashid, A., Garcia, A.F., da Silva, L.F.: Semantic vs. syntactic compositions in aspect-oriented requirements engineering: an empirical study. In: K.J. Sullivan (ed.) Proceedings of the 8th International Conference on Aspect-Oriented Software Development, AOSD 2009, Charlottesville, Virginia, USA, March 2–6, 2009, pp. 149–160. ACM (2009). doi:10.1145/1509239.1509260
15. Ciolkowski, M.: What do we know about perspective-based reading? an approach for quantitative aggregation in software engineering. In: Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on, pp. 133–144 (2009)
16. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley Professional, Reading (2001)
17. Czarnecki, K., Antkiewicz, M.: Mapping features to models: A template approach based on superimposed variants. In: Generative Programming and Component Engineering (GPCE), LNCS, vol. 3676, pp. 422–437. Springer (2005). doi:10.1007/11561347_28
18. Czarnecki, K., Eisenecker, U.: Generative Programming: Methods, Tools, and Applications. ACM Press/Addison-Wesley Publishing Co., New York (2000)
19. Czarnecki, K., Pietroszek, K.: Verifying feature-based model templates against well-formedness ocl constraints. In: 5th International Conference on Generative Programming and Component Engineering, pp. 211–220. ACM, New York, NY (2006). doi:10.1145/1173706.1173738
20. Deligiannis, I., Shepperd, M., Roumeliotis, M., Stamelos, I.: An empirical investigation of an object-oriented design heuristic for maintainability. J. Syst. Softw. **65**(2), 127–139 (2003). doi:10.1016/S0164-1212(02)00054-7
21. Deligiannis, I.S., et al.: A review of experimental investigations into object-oriented technology. Empir. Softw. Eng. **7**, 193–231 (2002)
22. Do, H., Mirarab, S., Tahvildari, L., Rothermel, G.: The effects of time constraints on test case prioritization: A series of controlled experiments. IEEE Trans. Softw. Eng. **36**(5), 593–617 (2010). doi:10.1109/TSE.2010.58
23. Eaddy, M.: An empirical assessment of the crosscutting concern problem. Ph.D. thesis, Graduate School of Arts and Sciences, Columbia University (2008)
24. Eaddy, M., Aho, A., Murphy, G.: Identifying, assigning, and quantifying crosscutting concerns. In: First Workshop on Assessment of Contemporary Modularization Techniques (ACOM). Minneapolis, USA (2007)
25. Eaddy, M., Zimmermann, T., Sherwood, K.D., Garg, V., Murphy, G.C., Nagappan, N., Aho, A.V.: Do crosscutting concerns cause defects? IEEE Trans. Softw. Eng. **34**(4), 497–515 (2008). doi:10.1109/TSE.2008.36
26. Eriksson, M., Borstler, J., Borg, K.: The pluss approach - domain modeling with features, use cases and use case realizations. In: 9th International Conference on Software Product Lines, pp. 33–44. LNCS (2005)
27. Eriksson, M., Borstler, J., Borg, K.: Software product line modeling made practical. Commun. ACM **49**(12), 49–54 (2006). doi:10.1145/1183236.1183265
28. Eriksson, M., Börstler, J., Borg, K.: Managing requirements specifications for product lines: An approach and industry case study. J. Syst. Softw. **82**, 435–447 (2009)
29. Everitt, B., Hothorn, T.: A Handbook of Statistical Analyses Using R. Chapman & Hall/CRC (2006). http://CRAN.R-project.org/package=HSAUR
30. Figueiredo, E., et al.: Evolving software product lines with aspects: an empirical study on design stability. In: 30th International Conference on Software Engineering, pp. 261–270. ACM, New York, NY (2008). doi:10.1145/1368088.1368124
31. Figueiredo, E., et al.: On the Maintainability of Aspect-Oriented Software: A Concern-Oriented Measurement Framework. Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on pp. 183–192 (2008)
32. Filho, F.C., Cacho, N., Figueiredo, E., ao, R.M., Garcia, A., Rubira, C.M.F.: Exceptions and aspects: the devil is in the details. In: 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 152–162 (2006)
33. Garcia, A., et al.: Modularizing design patterns with aspects: a quantitative study. 4th International Conference on Aspect-oriented Software Development 14(18), 3–14 (2005)
34. Gheyi, R., Massoni, T., Borba, P.: A theory for feature models in alloy. In: First Alloy Workshop, pp. 71–80. Portland, United States (2006)
35. Glass, R.L.: The software-research crisis. Softw. IEEE **11**(6), 42–47 (1994). doi:10.1109/52.329400
36. Greenwood, P., et al.: On the impact of aspectual decompositions on design stability: An empirical study. 21st European Conference on Object-Oriented Programming (2007)
37. Group, Z.: Open conference management system. on-line (2009). http://www.openconf.com/
38. Hanenberg, S., Kleinschmager, S., Josupeit-Walter, M.: Does aspect-oriented programming increase the development speed for crosscutting code? An empirical study. In: ESEM '09: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, pp. 156–167. IEEE Computer Society, Washington, DC, USA (2009). doi:10.1109/ESEM.2009.5316028
39. Juristo, N., Moreno, A.: Basics of Software Engineering Experimentation. Kluwer Academic Pub, Boston (2001)

40. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (foda) feasibility study. Tech. Rep. CMU/SEI-90-TR-21, Software Engineering Institute (1990)
41. Kästner, C., Apel, S., Kuhlemann, M.: Granularity in software product lines. In: 30th International Conference on Software Engineering, pp. 311–320. ACM, New York (2008). doi:10.1145/1368088.1368131
42. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.: Getting started with aspect. Commun. ACM **44**(10), 59–65 (2001). doi:10.1145/383845.383858
43. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.G.: An overview of AspectJ. In: ECOOP '01: Proceedings of the 15th European Conference on Object-Oriented Programming, pp. 327–353. Springer, London (2001)
44. Kienzle, J., et al.: Crisis management systems (2009). http://www.cs.mcgill.ca/joerg/taosd/TAOSD/
45. Kitchenham, B., Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D., El Emam, K., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. **28**(8), 721–734 (2002). doi:10.1109/TSE.2002.1027796
46. Kitchenham, B., Pfleeger, S.L., Fenton, N.: Towards a framework for software measurement validation. IEEE Trans. Softw. Eng. **21**(12), 929–944 (1995). doi:10.1109/32.489070
47. Konstantopoulos, S., Hedges, L.V.: Analyzing effect sizes: Fixed-effects models. The handbook of research synthesis and meta-analysis pp. 279–293 (2009)
48. Krueger, C.W.: Easing the transition to software mass customization. In: Revised Papers from the 4th International Workshop on Software Product-Family Engineering, PFE '01, pp. 282–293. Springer, London (2002). http://dl.acm.org/citation.cfm?id=648114.748909
49. Krueger, C.W.: New methods in software product line practice. Commun. ACM **49**(12), 37–40 (2006). doi:10.1145/1183236.1183262
50. Kuehl, R., Kuehl, R.: Design of experiments: statistical principles of research design and analysis. Duxbury-Thomson Learning (2000)
51. Lagaisse, B., Win, B.D., Joosen, W., Oeyen, J.V.: E-finance case study: analysis and requirements. Tech. Rep. CW438, Department of Computer Science, Katholieke Universiteit Leuven (2006). http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW438.abs.html
52. Lemos, O., Ferrari, F., Silveira, F., Garcia, A.: Development of auxiliary functions: Should you be agile? an empirical assessment of pair programming and test-first programming. In: Software Engineering (ICSE), 2012 34th International Conference on, pp. 529–539 (2012). doi:10.1109/ICSE.2012.6227163
53. Lima, L., Iyoda, J., Sampaio, A., Aranha, E.: Test case prioritization based on data reuse an experimental study. In: ESEM '09: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, pp. 279–290. IEEE Computer Society, Washington, DC (2009). doi:10.1109/ESEM.2009.5315980
54. Linden, F.J.v.d., Schmid, K., Rommes, E.: Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer, New York (2007)
55. Masuhara, H., Kiczales, G.: Modeling crosscutting in aspect-oriented mechanisms. In: European Conference on Object-Oriented Programming (ECOOP), Lecture Notes in Computer Science, pp. 2–28. Springer (2003)
56. Pohl, K., Böckle, G., van der Linden, F.J.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer, New York (2005)
57. Pohl, K., Metzger, A.: The eshop product line. Online: http://www.sei.cmu.edu/splc2006/eShop.pdf
58. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna (2011). http://www.R-project.org/. ISBN 3-900051-07-0
59. Ramsey, F., Schafer, D.: The Statistical Sleuth: A Course in Methods of Data Analysis. Duxbury Press Belmont, California (1997)
60. Sampaio, A., Greenwood, P., Garcia, A., Rashid, A.: A comparative study of aspect-oriented requirements engineering approaches. In: Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on, pp. 166–175 (2007). doi:10.1109/ESEM.2007.15
61. Sjoberg, D.I., Yamashita, A., Anda, B., Mockus, A., Dyba, T.: Quantifying the effect of code smells on maintenance effort. IEEE Transactions on Software Engineering 99(PrePrints), 1 (2012). doi:10.1109/TSE.2012.89
62. Soares, S., Borba, P., Laureano, E.: Distribution and persistence as aspects. Softw Pract Exp **36**(7), 711–759 (2006)
63. Soares, S., Laureano, E., Borba, P.: Implementing distribution and persistence aspects with aspectj. SIGPLAN Not. **37**(11), 174–190 (2002). doi:10.1145/583854.582437
64. Whittle, J., Araujo, J.: Scenario modelling with aspects. IEE Proc Softw **151**(4), 157–171 (2004). doi:10.1049/ip-sen:20040921
65. Whittle, J., Moreira, A., ao Araújo, J., Jayaraman, P., Elkhodary, A., Rabbi, R.: An expressive aspect composition language for uml state diagrams. In: International Conference on Model Driven Engineering, Languages and Systems (MODELS'2007), LNCS, vol. 4735, pp. 514–528. Springer (2007)
66. Wohlin, C., Runeson, P., Host, M., Ohlsson, C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluver, Norwell (2000)
67. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer Science & Business Media, New York (2012)


**Rodrigo Bonifácio** is an Assistant Professor at the Computer Science Department of University of Brasília. His main interests are software product lines, software modularity, and empirical software engineering.

**Paulo Borba** is a Professor of Software Development at the Informatics Center of the Federal University of Pernambuco, Brazil, where he leads the Software Productivity Group. His main research interests are in the following topics and their integration: software modularity, software product lines, and refactoring.

**Cristiano Ferraz** is a Professor of Statistics at Federal University of Pernambuco. He is an International Statistical Institute 2013 member elected. His areas of interest are survey sampling, design and analysis of experiments, and statistical methods.



**Paola Accioly** is a Computer Science PhD Student at the Informatic Center of the Federal University of Pernambuco, Brazil. Her research interests are focused in the following areas: software modularity, collaborative development, and software product lines.