Journal of Universal Computer Science, vol. 20, no. 5 (2014), 666-691
submitted: 28/7/13, accepted: 15/2/14, appeared: 1/5/14 © J.UCS

Defining and Validating a Feature-Driven Requirements Engineering Approach

Raphael Pereira de Oliveira
(Federal University of Bahia (UFBA), Salvador, Brazil
raphaeloliveira@dcc.ufba.br)

David Blanes, Javier Gonzalez-Huerta, Emilio Insfran, Silvia Abrahão
(Universitat Politècnica de València (UPV), Valencia, Spain
{dblanes, jagonzalez, einsfran, sabrahao}@dsic.upv.es)

Sholom Cohen
(Software Engineering Institute (SEI), Carnegie Mellon University, Pittsburgh, USA
sgc@sei.cmu.edu)

Eduardo Santana de Almeida
(Federal University of Bahia (UFBA), Salvador, Brazil
Fraunhofer Project Center (FPC) for Software and Systems Engineering, Brazil
esa@dcc.ufba.br)

Abstract: The specification of requirements is a key activity for achieving the goals of any software project and it has long been established and recognized by researchers and practitioners. Within Software Product Lines (SPL), this activity is even more critical owing to the need to deal with common, variable, and product-specific requirements, not only for a single product but for the whole set of products. In this paper, we present a Feature-Driven Requirements Engineering approach (FeDRE) that provides support to the requirements specification of SPL. The approach realizes features into functional requirements by considering the variability captured in a feature model. It also provides detailed guidelines on how to associate chunks of features from a feature model and to consider them as the context for the Use Case specification. The evaluation of the approach is illustrated in a case study for developing an SPL of mobile applications for emergency notifications. This case study was applied within 14 subjects, 8 subjects from Universitat Politècnica de València and 6 subjects from Federal University of Bahia. Evaluations concerning the perceived ease of use, perceived usefulness, effectiveness and efficiency as regards requirements analysts using the approach are also presented. The results show that FeDRE was perceived as easy to learn and useful by the participants.

Keywords: Software Product Lines, Requirements Specification, Reuse
Categories: D.2.1, D.2.13

1 Introduction

Defining requirements to determine what is to be developed is generally accepted as a vital but difficult part of software development. Establishing the driving architectural requirements not only simplifies the design and implementation phases but also

reduces the number of errors detected in later stages of the development process, thus reducing the risk, duration and budget of the project [Jones, 96].

The specification of requirements in Software Product Lines (SPL) [Clements, 07] development is even more critical. In this context, it is necessary to deal with common, variable, and product-specific requirements, not only for a single product but also for the whole set of products in the family. One fundamental aspect of engineering SPLs is that of applying Requirements Engineering (RE) practices to deal with the scoping and specification of the SPL in both the Domain Engineering and Application Engineering processes.

In the Domain Engineering process, the RE activities are intended to define the extent of the SPL in order to determine its products (scoping), and also to identify common, variable, and product-specific features throughout the SPL. The specification of the requirements needed to deploy features must also be specified in a systematic manner by establishing explicit traceability between features and requirements. In the Application Engineering process, the RE activities are intended to specify the requirements for a particular product in the product family. It is therefore important to determine which requirements from the SPL are relevant to the product to be developed (common and variant feature selection), and also to refine or to add new specific requirements, not present in the SPL (delta requirements).

Most of the approaches that deal with RE in SPL development tend to include variability information in traditional requirements models (e.g., use case diagrams) [Moon, 05] or to extract feature models [Kang, 90] from requirements specifications by following a bottom-up strategy [Asadi, 11] [Mussbacher, 11]. Some limitations of these approaches arise from the possibility of a large number of requirements and features making the specification of requirements hard to understand, maintain and prone to inconsistencies. The contribution of our approach is that we circumscribe the requirements specifications in order to deal with complexity in a more effective way. Effectiveness is achieved by chunking the requirement activity based on areas of the feature model. This constrains the extent of the requirements specification at any one time to a more specific area of the SPL. The feature model is used as a basis principally because in the SPL community, features are first-class citizens, which are easily identifiable, well-understood, and easy for SPL developers and domain experts to communicate. There is thus a strong need to define traceability links between these features and requirements and, whenever possible, to maintain the model and specification synchronized and consistent [Alférez, 11] [Anquetil, 10] [Heidenreich, 10].

In this paper we improved and validated the Feature-Driven Requirements Engineering (FeDRE) approach [Oliveira, 13] to help developers in the RE activity for SPL development. This paper focuses on the specification of requirements at early stages, taking as input the scoping artifacts. Thus, the approach proposes a set of artifacts, activities, roles, and guidelines on the basis of the features to be developed. The improved approach has new steps in the guidelines and a tool support for the specification. We focus on the requirements specification of the Domain Engineering activity. We further focus our description of FeDRE by starting once a feature model has been defined (in the scoping activity). However, we do not deal with Quality Attributes (QAs) in the feature model. The next FeDRE activity consists of the systematic realization of features in terms of use cases. This activity specifies

requirements but also establishes traceability between the features and the requirements. This allows us to provide variability mechanisms at the requirements level (by using use cases and alternative scenarios) according to the chunk of the feature model that these requirements specify. The main contributions of FeDRE is an RE approach that 1) systematically realizes features into requirements by considering the variability captured in the feature model and 2) breaks the top-down driven paradigm through use of the feature model in order to prioritize features according to architecturally significant areas of the SPL. A first evaluation of FeDRE was performed through a case study in a real SPL context, where the perceived ease of use, perceived usefulness, effectiveness and efficiency of the approach were evaluated.

The remainder of the paper is structured as follows. Section 2 discusses related work on SPL-specific RE approaches. Section 3 presents our feature-driven requirements engineering approach. Section 4 illustrates the feasibility of the approach through a case study conducted to develop an SPL of mobile applications for emergency notifications. Finally, Section 5 presents our conclusions and future work.

2 Related Work

Several models and techniques that deal with the specification of requirements in SPL development have been proposed over the last few years. We analyze some of these proposals by using a comparison criteria in order to discover how the current approaches cover the RE activity to model SPL requirements. The comparison criteria were formed of four main criteria. The first analyses the **SPL activities** supported in the development (Scoping, Domain Engineering, and Application Engineering). The second criterion encompasses the **RE activities** that were used in the RE approaches according to the disciplines (elicitation, specification, analysis, verification and management) that guide an RE process [Clements, 07]. In the third criterion we analyze which **artifacts** where employed to model the requirements. Finally, we analyzed **how the process was defined**. The analysis of "how the process was defined" was performed analyzing three sub-criteria: whether the approach provides guidelines, whether the approach defines roles, and whether the approach has well defined inputs and outputs.

Some approaches combine feature models with more traditional RE techniques such as use cases [Griss, 98] [Eriksson, 05]. FeatuRSEB [Griss, 98] proposes simultaneously building a use case model and a feature model, and then performing the commonality and variability analysis, first over the use case models and then over the feature model. PLUSS [Eriksson, 05] improves the FeatuRSEB approach by adding more variability mechanisms: i) at the use case level; ii) at the alternative scenario level; iii) at the flow of events from an included alternative scenario; and, iv) with cross-cutting aspects that affect several use cases. Neither FeatuRSEB nor PLUSS propose roles in their methods, and merely provide partial guidelines to help in the RE activity. In addition, the input and output artifacts are only partially defined. In FeDRE, the feature model is the main artifact used to model variability, and a use case model is built for chunks of this feature model in a systematic manner. This improves our ability to deal with complexity by narrowing the context of the use case

specification. With regard to the variability mechanisms in requirements, FeDRE borrows the first two types of variability from PLUSS (use case level, and alternative scenario).

The idea of combining a feature model with use cases was also used by the Variability Modeling Language for Requirements (VML4RE) approach in the context of the AMPLE project [Alférez, 11]. This approach presents two main contributions. First, the VML4RE language is a Domain Specific Language that is able to compose requirements with feature models; and second, a requirements process that uses: i) a feature model to perform the variability identification; ii) use cases and activity diagrams to describe the SPL domain requirements; and a iii) a VML4RE model to relate the feature model with the requirement models. These three models, along with a feature configuration, are taken as input by an interpreter to obtain the application requirements. They also provide consistency checking between features and use case scenarios. Similarly, Modeling Scenario Variability as Crosscutting Mechanisms (MSVCM) [Bonifácio, 09] is focused on obtaining the application requirements by using the following artifacts: use case model, feature model, product configuration, and configuration knowledge. These artifacts are taken as input in the weaving process, which crosscut each other according to the resulting product specific use case model. Oppositely, we present FeDRE to obtain the domain requirements from the Scoping activity following guidelines in a systematic way. Thus, our focus is on how to specify the domain requirements by using a guided process rather than obtain the application requirements, which is the focus of the VML4RE and MSVCM proposals. Moreover, these proposals do not explicitly mention guidelines.

Other related works include approaches that extend different requirements models such as use cases and scenarios with variability concepts without explicitly using feature modeling [Bayer, 00] [Moon, 05]. The Pulse-CDA approach [Bayer, 00] takes the information from the economic scope (a range of system characteristics and a scope definition) and then outputs a domain model (composed of a set of work products that capture different domain views) and a decision model. In [Muthing, 04], the use-case technique is used as a work product to represent the variability in the use cases. Any element from the use case diagram or in the textual scenario can be a variant (e.g., an actor or a use case). The variant elements are enclosed with XML-style tags to explicitly mark variability. This solution provides the user with high flexibility. However, from our point of view, considering any element in the use case diagram or in the textual use case specification to be potentially variable could lead to a high number of different requirements from the same problem which may consequently result in the production of ambiguous use cases. In FeDRE we allow variability at use case level (a use case can or cannot be) and at scenario level (adding alternative scenarios). The FeDRE solution is an agreement between: providing sufficient expressiveness and producing unambiguous requirements specifications that mitigate this problem. DREAM [Moon, 05] is a different technique that extends traditional use cases to support variability, in which the starting point is a set of legacy systems that are analyzed to extract the requirements. DREAM uses two stereotypes that are defined to represent variability in use case diagrams: «common» when the use case is always present in every product configuration and «optional» when the use case is present in some product configurations. In Pulse-CDA the decision model is traced to the variable elements in the use case and scenario

description in order to instantiate the models. In FeDRE, this variability from use cases and scenarios is traced to the feature model through a traceability matrix. Neither PuLSE-CDA nor DREAM proposes roles in theirs RE process to extend the requirements models to support variability. However, both approaches define the input and output artifacts in their processes. PulSE-CDA does not provide guidelines, but DREAM proposes a set of guidelines to obtain the domain requirements specification from legacy systems.

Another traditional RE technique is that of goal modeling. In [Soltani, 12], the stakeholder's intentions, which are represented as goals, are mapped onto the software features in order to express their variability in an annotated feature model. In Aspect-oriented User Requirements Notation (AoURN) [Mussbacher, 11] four main domain engineering activities are proposed: i) build a stakeholder goal model; ii) build a feature model, in which features are represented as goal-tasks with the «feature» stereotype; iii) build the feature impact model to establish the impact of features on the stakeholder's goals; and iv) create the feature scenario model, in which non-leaf features are described in more detail with the Aspect-oriented Use Case Maps (AoUCM). In AoURN the traceability from features to requirements is done by using links from the stereotyped tasks in the feature model to the AoURN scenario model. These approaches do not define the roles in their processes and only provide partial guidelines for their use. Additionally, the input and output artifacts are only partially defined. Both proposals allow the RE expert to obtain a feature model from a previous goal model. In FeDRE, the starting point is a feature model, which is based on concepts that the domain expert works directly, rather than using unfamiliar goal models to guide the creation of the feature model.

Another alternative is to extend traditional UML notations with variability information. Shaker propose the Feature-Oriented Requirements Modeling Language (FORML) [Shaker, 12] based on feature modeling and UML state-machines. FORML decomposes the requirements into the world model and the behavior model. In the world model, a feature model describes the features that compose the problem. One feature in the world model is decomposed into several feature modules in the behavior model. A feature module is represented with an UML-like finite state machine. This decomposition permits feature modularity, which is one of the main contributions of the work. FORML does not define roles and guidelines in the process in order to obtain the requirements specification. Upon comparing FORML and FeDRE it will be noted that both approaches support modularity. FORML decomposes a feature model from the world model into several feature modules in the behavior model; FeDRE similarly allows sets of features to be decomposing into functional requirements by using use cases, scenarios, and traceability links.

Finally, we analyzed several RE approaches for SPL development, and we found a distinct set of approaches and techniques (Table 1). Summarizing, in many cases the scoping and requirements specification activities are considered as independent activities. According to John and Eisenbarth [John, 09], well-defined relationships and interfaces between scoping and requirements artifacts should be defined in order to reduce rework. To alleviate this problem, FeDRE considers the scoping artifacts as the starting point and defines guidelines to conduct the SPL requirements specification driven by the scoping artifacts. Another important factor is the strategy followed to specify the requirements. Several approaches, such as use cases (i.e.,

[Eriksson, 05] [Griss, 98]) or goal models adapted to the SPL domain (i.e., [Asadi, 11] [Mussbacher, 11]), extend RE models and extract feature models from these RE models. In our view, SPL developers and domain experts are more familiar with the concept of feature and variability modeling. As a means to deal with complexity, we therefore restrict the requirements specification in accordance with chunks of the feature model. Moreover, guidelines to specify functional requirements related to features are provided, resulting in an explicit traceability framework built in a systematic manner.

Approach	SPL processes	RE disciplines	Artifacts	Process definition
FeatuRSEB	Domain engineering	Elicitation, modeling, analysis	Use case model, feature model	Partially (guidelines, inputs and outputs)
PLUSS	Domain engineering, application engineering	Elicitation, modeling, analysis, management	Feature model, use case, change case	Partially (guidelines, inputs and outputs)
VML4RE	Domain engineering, application engineering	Elicitation modeling, analysis, management	Feature model, use cases, activity diagrams	Partially (inputs and outputs)
MSVCM	Domain engineering, Application engineering	Modeling, analysis, management	Use case model, feature model, product configuration, configuration knowledge	Partially (inputs and outputs)
Pulse-CDA	Scoping, domain engineering	Elicitation, modeling, analysis	Domain analysis model, use cases	Partially (inputs and outputs)
DREAM	Domain engineering	Elicitation, modeling, analysis	PR-Context matrix, use cases	Partially (guidelines, inputs and outputs)
AoURN	Domain engineering, application engineering	Elicitation, modeling, analysis	Stakeholder goal model, feature model, feature impact model, feature scenario model	Partially (inputs and outputs)
FORML	Domain engineering	Modeling	Feature model, behavior model	Partially (inputs and outputs)
FeDRE	Scoping, domain engineering, application engineering	Elicitation, modeling, management	Feature model, feature specification, product map, glossary, traceability matrix, use cases	Complete (guidelines, inputs and outputs)

Table 1: Comparative among current RE proposals from SPL

3 Feature-Driven Requirements Engineering Approach For SPL

The *Feature-Driven Requirements Engineering (FeDRE)* approach for SPLs has been defined by considering the feature model as the main artifact for specifying SPL requirements. The aim of the approach is to perform the requirements specification by systematically utilizing the features identified in the SPL domain through the use of guidelines that establish traceability links between features and requirements. By domain, we mean the context in which the family of products or functional areas across the products exhibits common, variable or specific functionalities.

The main activities of the FeDRE approach are: Scoping, Requirements Specification for Domain Engineering, and Requirements Specification for Application Engineering. Figure 1 shows the first two activities in FeDRE, which are detailed in this paper. The following roles are involved in these activities: Domain Analyst, Domain Expert, Market Expert and the Domain Requirements Analyst.

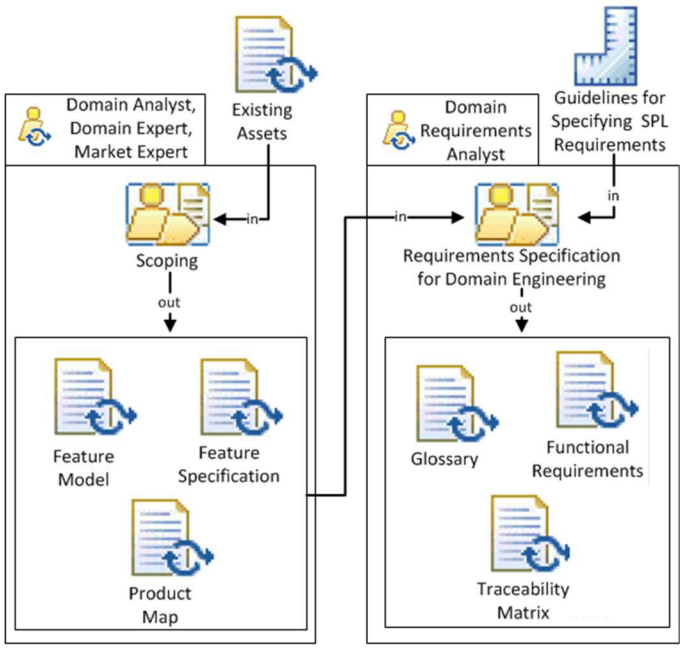

Figure 1: Overview of the FeDRE approach

3.1 Scoping

The first activity performed in FeDRE is the Scoping. This determines not only what products to include in an SPL but also whether or not an organization should launch the SPL. According to Bosch [Bosh, 00], the Scoping activity consists of three levels: product portfolio Scoping, domain Scoping, and asset Scoping. Product portfolio Scoping determines which products and product features should be included in an

SPL. Domain Scoping defines the functional areas and subareas of the SPL domain, while Asset Scoping identifies assets with costs and benefits estimated for them.

In FeDRE, the Domain Expert and the Market Expert perform the product portfolio Scoping. The Domain Expert and Domain Analyst perform the Domain Scoping. Finally, all the roles in the Scoping activity perform the Asset Scoping.

Three main artifacts are produced as a result of the Scoping activity: the Feature Model, the Feature Specification, and the Product Map, using the Existing Assets (if any) as the input artifact. These three artifacts will drive the SPL requirements specification for domain engineering. Details of the Scoping activity are shown in Figure 2. Each of these artifacts (input and outputs) is detailed below.

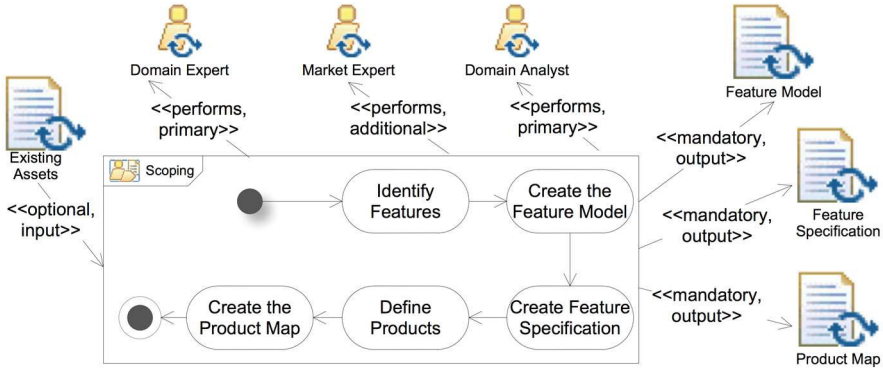

Figure 2: Detailed Scoping Activity

3.1.1 Existing Assets

When performing an extractive or reactive SPL adoption [Krueger, 01], existing assets (e.g., user manual or existing systems) help the Domain Analyst and the Domain Expert identify the features and products in the SPL. Otherwise, a proactive approach can be followed to build the SPL from scratch.

3.1.2 Feature Model

Feature modeling is a technique that is used to model common and variable properties, and can be used to capture, organize and visualize features in the SPL. The Domain Analyst and the Domain Expert identify features using existing assets as input or by eliciting information from experts and from the Market Expert. A Feature Model diagram [Kang, 90] will identify features, SPL variations, and constraints among the features in the SPL.

3.1.3 Feature Specification

The Domain Analyst is responsible for specifying the features using a feature specification template. This template captures the detailed information of the features and maintains traceability with all the artifacts involved. According to the template, each feature must have a unique identifier *Feat id* and a *Name*. The values for the

Variability field can be Mandatory, Optional, or Alternative, according to the feature specified (Table 2). The *Priority* of the feature should be High, Medium or Low. If the feature requires or excludes another feature(s), the *Feat id*(s) from the required or excluded feature(s) must be specified. If the feature has a *Parent* feature, the *Feat id* from the parent feature must be specified. The *Binding Time* can be compile time or run time, according to the time that the feature will be included in a concrete product [Czarnecki, 00]. The *Feature Type* can be concrete or abstract, and the *Description* is a brief explanation of the feature.

●	Mandatory Feature
○	Optional Feature
▲	Alternative Feature (OR) (one or more feature(s) can be selected)
△	Alternative Feature (XOR) (only one feature can be selected)

Table 2: Features Variability

3.1.4 Product Map

Each of the identified features is assigned to the corresponding products in the SPL. The set of relationships among features and products produces the Product Map artifact, which describes all the features that are required to build a specific product in the SPL. It is usually represented as a matrix in which columns represent the products and rows represent the features. The Market Analyst, the Domain Analyst and the Domain Expert produce this artifact.

All these artifacts are the input for the Requirements Specification for Domain Engineering activity, which is described below.

3.2 Requirements Specification for Domain Engineering

This activity specifies the SPL requirements for domain engineering. These requirements allow realization of the features and desired products identified in the Scoping activity. The steps required to perform this activity are described in the Guidelines for Specifying SPL Requirements, Sub-Section 3.3 below.

The FeDRE approach was defined using and extending the PLUSS approach [Eriksson, 05], which represents requirements specifications as use case scenarios. The use case scenarios "*force requirements analysts to always think about interfaces since separate fields exist for describing actor and system actions*". Our approach supports the relationship between features and use cases. The feature variability is expressed within the use cases. FeDRE differs from PLUSS as regards our approach toward two types of variability: i) use case variability, considering the whole use case as a variant; and ii) scenario variability, in which the variants are alternative scenarios of a use case. In our approach these two types of variability are sufficient to capture the variations within SPL requirements. We have experienced that, in the general-purpose SPLs the variability does not go beyond use case variability and scenario variability. We have also performed the case study to empirically validate this fact. We also analyzed the Software Product Line Conference (SPLC), and in the majority of the approaches, the variability of the examples and industry projects could be solved with these two levels of requirements variability. So far, FeDRE is responsible

for specifying requirements with a high level of abstraction, nevertheless, in the future, if we need more expressive variability mechanisms (i.e., fine-grained variability) we will consider to incorporate them.

When a requirement is identified or refined, it is necessary to determine whether it is a shared requirement for different products in the SPL, or whether it is a specific requirement of a single product. Shared requirements must also be classified into common and variable requirements. Common requirements are used throughout the SPL and variable requirements must be configured or parameterized in the specification of different variants of the SPL. In addition, some requirements may require or exclude other requirements, or may restrict possible configurations of other requirements. Feature models may help in handling the different types of dependencies among requirements, which can be complex and must be properly addressed.

The Requirements Specification for Domain Engineering activity is usually performed in an iterative and incremental manner. Sets of selected features from the Feature Model can therefore be defined as units of increments for the specification (different criteria may be used to choose features in a unit of increment, e.g., priority of implementation, cost, QAs). This activity (Figure 3) uses the Feature Model, Feature Specification and Product Map as input artifacts and produces the Glossary, Functional Requirements and Traceability Matrix as output artifacts. Each of these output artifacts is detailed below.

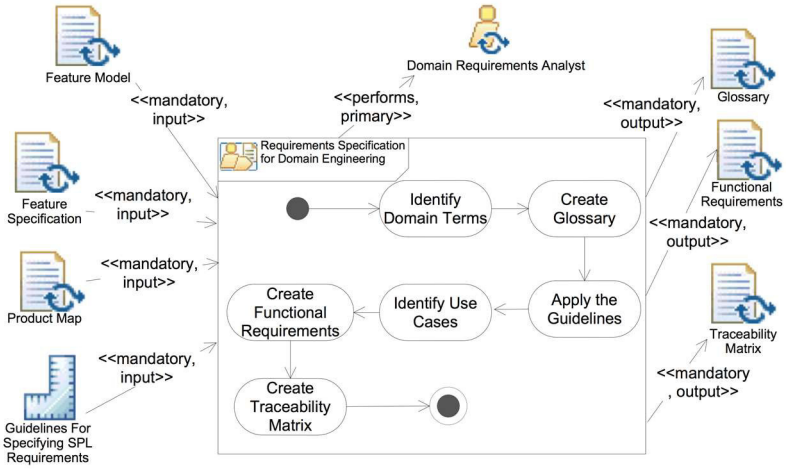

Figure 3: Detailed Requirements Specification Activity

3.2.1 Glossary

One important characteristic of SPL is the presence of multiple stakeholders, domain experts, and developers. It is therefore necessary to have a common vocabulary for describing the relevant concepts of the domain. The Glossary describes and explains the main terms in the domain in order to provide the stakeholders with a common

vocabulary and to avoid misconceptions. It is represented as a two-column table containing the term to be defined and its description (see Table 4 in Section 4).

3.2.2 Functional Requirements

This artifact contains all the functional requirements identified (common or variable), for the family of products that constitute the SPL. Use cases are used to specify the SPL functional requirements (each functional requirement is represented as a use case), and the variations required can be related to the use case as a whole or to alternative scenarios inside a use case. FeDRE adapts the template used in [Eriksson, 05] in order to specify functional requirements as use cases, thus supporting both types of variability. The specification of functional requirements follows the functional requirements template shown in Table 6 (Section 4). Each functional requirement has a unique *Use case id*, a Name, a Description, Associated Feature(s), Pre and Post-Conditions, and the Main Success Scenario. A functional requirement can also be related to an Actor and may have Include and/or Extend relationships with other use case(s). Extends relationships should describe a condition for the extension.

The Main Success Scenario and the Alternative Scenarios have Steps (represented by numbers), Actor Actions (representing an action from the actor) and Blackbox System Responses (representing a response from the system). The Alternative scenarios have a Name, a Condition and (optionally) relations to affected features through the Associated Feature field.

3.2.3 Traceability Matrix

The Traceability Matrix is a matrix that contains the links among features and the functional requirements. The rows in the matrix show the features and the columns show the functional requirements, as shown in Table 5 (Section 4). This matrix is also useful as regards helping in the evolution of the requirements since each change in the feature model will be traced up to the requirements through the traceability matrix (and vice versa).

3.3 Guidelines for Specifying SPL Functional Requirements

The purpose of the guidelines is to guide the Requirements Analyst in the specification of SPL functional requirements for domain engineering. The guidelines are based on a meta-model (see Figure 4) that represents the concepts involved when specifying use cases with alternative scenarios and the relationships among them.

The meta-model is used to maintain the traceability among all the elements and to facilitate understanding. The meta-model comprises the following elements:

- *RequirementsSpecification*: Is the container of all the elements in the specification
- *Feature*: This represents a feature from a variability model. Although it is not defined in this model, it is related to zero or many requirements
- *Requirement*: It is an abstract metaclass used to represent functional requirements
- *UseCase*: Represents a functional requirement. A *UseCase* is associated with a *Feature*, other *UseCases* through the *include, extend or inheritance relationships*, or with *Actors*. It contains a *Main Scenario* and zero or many *Alternative Scenarios*
- *UseCasePackage*: This is the container for a *UseCaseDiagram*
- *UseCaseDiagram*: This is a view for *Actors*, *UseCases* and *Relationships*

• *Actor*: Is an actor and can be related to other *Actors* or associated with *UseCases*
• *Relationship*: Represents the different types of relationships among *UseCases*, which are *Include, Extend and Inheritance*
• *Scenario*: This is an abstract metaclass used to represent the two types of scenarios for the *UseCase*, which are *MainScenario* and *AlternativeScenario*
• *MainScenario*: Represents the "normal flow" of steps for a *UseCase*
• *AlternativeScenario*: Represents an alternative set of steps for a *UseCase*. It can be associated with a *Feature* to represent the variability in the scenario
• *Step*: Represents a step in the *MainScenario* or *AlternativeScenario*.

The guidelines have been structured to specify functional requirements by addressing the following questions: i) **Which** features or set of features will be grouped to be specified by use cases? (In our future work, we intend to group features according to QAs) ii) **What** are the specific use cases for the feature or set of features? iii) **Where** should the use cases be specified? (when there is a set of features in a hierarchy, do we specify the use cases for each individual feature or only for the parent features?) iv) **How** is the use case specified in terms of steps?

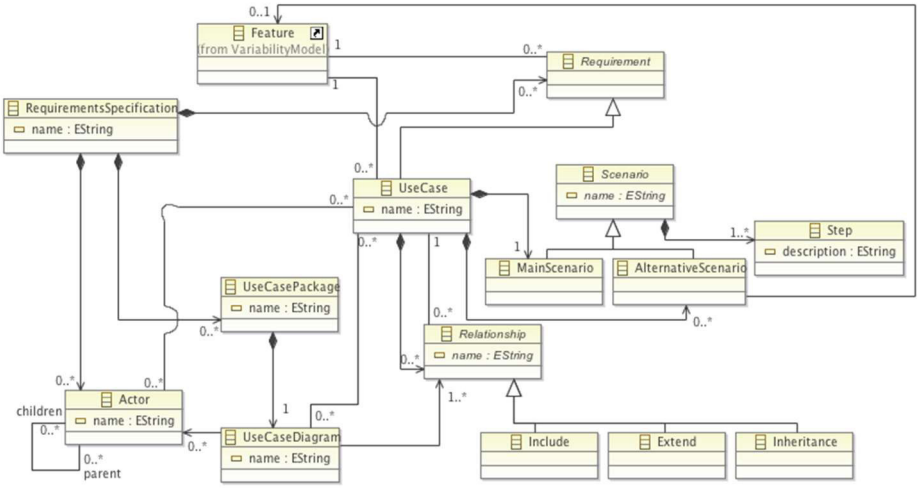

Figure 4: Meta-Model for SPL Requirements

The guidelines consider four types of feature variability that may be present in the feature model, as shown in Table 2. Activities, tasks and steps are used in the process of specifying requirements for SPL as is shown in Figure 5. The first activity, *Identify Use Cases*, uses the Feature model as an input and generates two artifacts as an output, *Traceability Matrix* and *Use Case Diagram*. The second activity, *Specify Use Cases*, uses the two outputs from the previous activity plus a *Use Case Template* to generate the *Use Case Specification.* Figure 6 shows the guidelines with the detailed steps of each task for specifying SPL functional requirements.

To easy the specification of functional SPL requirements keeping the traceability among features and requirements, we are improving a tool for managing SPL

artifacts, called *Software Product Line Integrated Construction Environment (SPLICE)*[1]. The tool is a web-based initiative to support most of the SPL process activities such as scoping, requirements, architecture, testing, version control, evolution, management and some agile practices. Since the tool considers several SPL activities and artifacts, SPLICE is responsible to keep the traceability among the SPL artifacts. For example, the tool allows the specification of features and functional requirements, and mainly the relationship (traceability) between them. Figure 7 presents a screen shot from the specification of an SPL functional requirement (use case) in the tool. There are some mandatory fields to be filled including the associated feature to this use case. The tool intends to help the requirement analysts in their activities and also help managers through reports, showing for example the traceability among the SPL artifacts.

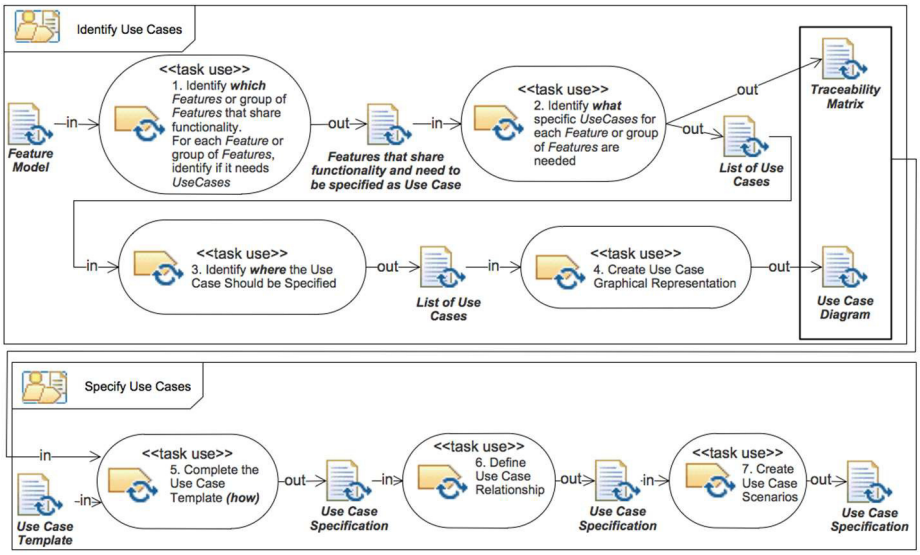

Figure 5: Overview of Activities, Tasks and Artifacts from the Guidelines

4 Case Study

An exploratory case study to assess the usefulness of FeDRE was performed by following the guidelines presented in [Wohlin, 12]. Besides this is a first evaluation of FeDRE for Domain Engineering, the obtained results make us to appreciate FeDRE as a promising approach. The stages of the case study development are: design, preparation, collection of data, and analysis of data. We additionally include a subsection for the threats to validity.

[1] Tool developed by RiSE Labs (https://wordpress.dcc.ufba.br/riselabs/)

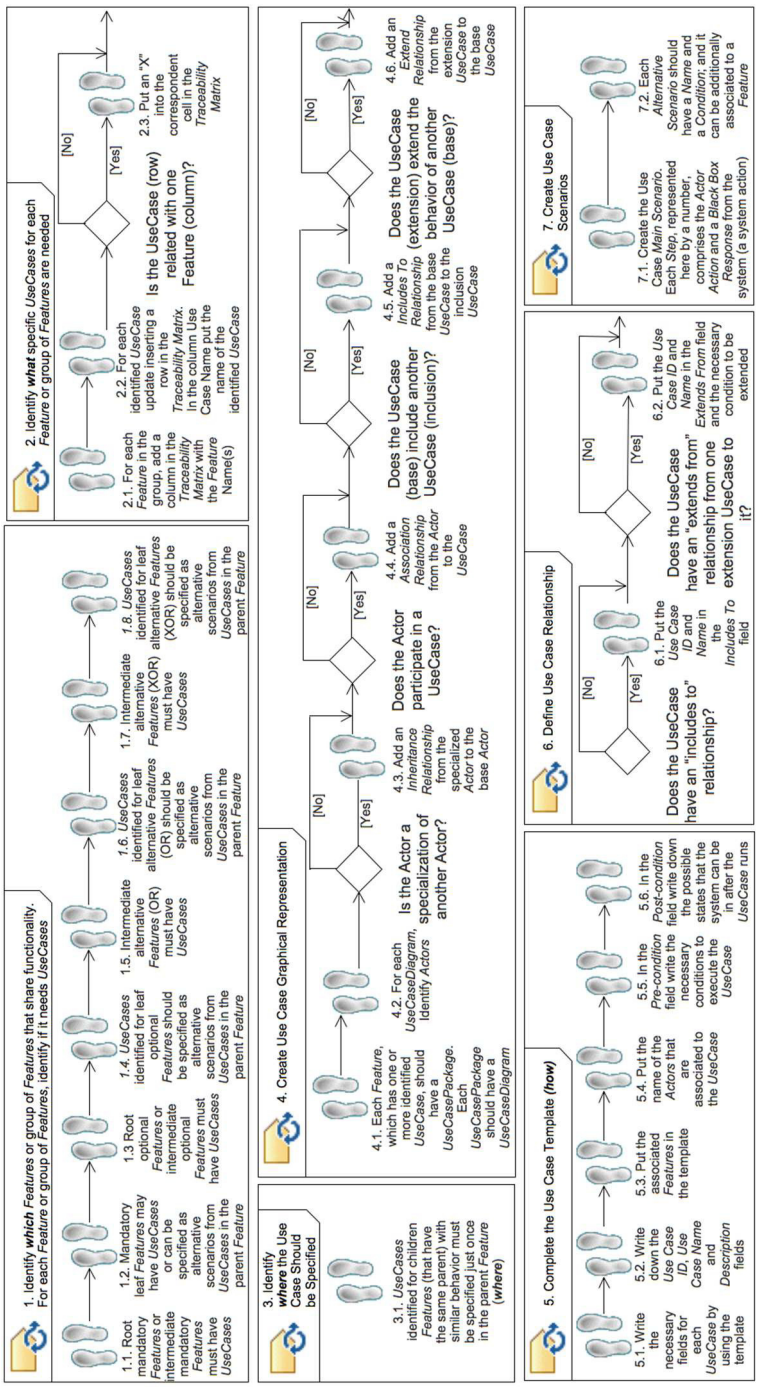

Figure 6: Guidelines for Specifying SPL Functional Requirements

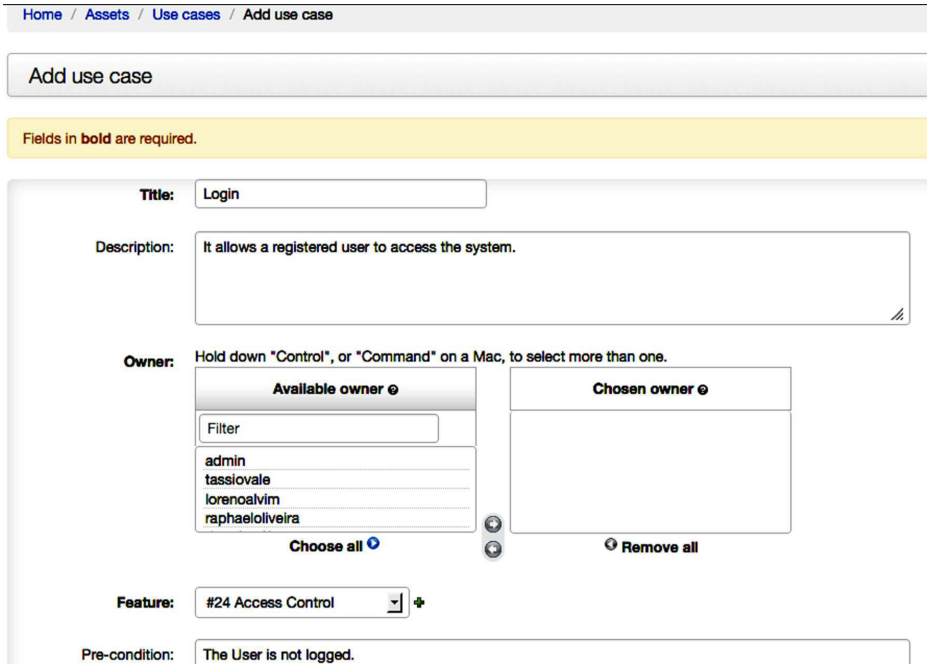

Figure 7 : Tool Support for Specifying Functional Requirements

4.1 Design of the case study

Firstly, the objectives and case study are planned. In order to define which objectives the case study would have, we applied the *Goal-Question-Metric* (GQM) [Mashiko, 97]. After applying this technique, we stated that the goal of the case study was: to **analyze** FeDRE **for the purpose** of evaluating it **with regard to** its perceived ease of use, and perceived usefulness **from the viewpoint** of a set requirements engineers.

The context of the case study is the requirements modeling of a real application in an SPL context. The SPL selected is for the mobile software called SAVi (http://goo.gl/1Q49O), which is an application that notifies and allows a mobile contact list to track a user in an emergency situation, sending a code by SMS and email to the contact list. We chose SAVi in order to apply FeDRE in a real SPL project using an extractive / reactive SPL adoption.

There are **two subjective dependent variables**: *perceived ease of use* and *perceived uselfulness*. To measure both variables after applying the FeDRE approach, we used an existing measurement instrument proposed for the evaluation of requirements modeling methods based on user perceptions [Abrahao, 11]. More specifically, we adapted two perception-based variables from the aforementioned instrument, which were based on two constructs from the Technology Acceptance Model (TAM) [Davis, 89]:

- *Perceived Ease of Use (PEOU)*: the degree to which a person believes that using FeDRE will be effort-free. This variable represents a perceptual judgment of the effort required to learn and use the FeDRE approach;
- *Perceived Usefulness (PU)*: the degree to which a person believes that FeDRE will achieve its intended objectives. This variable represents a perceptual judgment of the FeDRE approach's effectiveness.

We defined a set of items to measure these *perception-based variables*. These items were combined in a survey consisting of 7 statements. The items were formulated by using a 5-point Likert scale, using the opposing-statement question format. Various items within the same construct group were randomized to prevent systemic response bias. PEOU and PU were measured by using three and four items in the survey, respectively[2].

We formulated the following **hypotheses**:

- $H1_0$: FeDRE is perceived as difficult to use, $H1_a$: FeDRE is perceived as easy to use.
- $H2_0$: FeDRE is perceived as not useful, $H2_a$ FeDRE is perceived as useful.

In addition, before the case study session, a requirements specification considered as the correct solution was defined. This requirements specification was created by three of the authors of this paper. The aim of this agreed solution was to compare the subjects' solutions with the agreed solution in order to analyze the degree in which the subjects applied FeDRE in an effective and efficient way. ***Four objective dependent variables*** were defined with this aim in mind:

- *Effectiveness_UC*, which is calculated as the ratio between the number of right use cases that the subject identified and the total number of right use cases.
- *Effectiveness_SCEN*, which is calculated as the ratio between the number of right alternative scenarios that the subject identified and the total number of right alternative scenarios.
- *Efficiency_UC*, which is calculated as the ratio between the number of right use cases that the subject identified and the total time spent on the use cases identification.
- *Efficiency_SCEN*, which is calculated as the ratio between the number of right alternative scenarios that the subject identified and the total time spent on the alternative scenarios specification.

Table 3 shows the case study planning. Before the case study session, there was a 30 minutes **training session** to present an introduction of RE from SPL, and the use cases main concepts and notation (i.e., use cases, actors, types of relations, etc.), the FeDRE method, and the objectives and procedures of the study. After the training session, the case study was performed. This session was composed of two tasks. When the subjects finished, they filled in a questionnaire about FeDRE.

Several **documents**[3] were designed as instrumentation for the case study: slides for the training session, an explanation of the method, a data gathering form, and a questionnaire. These documents were used by the **subjects**, which were chosen for

[2] The questions are available at: http://users.dsic.upv.es/~dblanes/JUCS2013/Questions.pdf.

[3] The material is available at: http://users.dsic.upv.es/~dblanes/JUCS2013

convenience from a group of software engineering research associates. The subjects were asked about their experience in the area, and the results showed that none of them had a previous background in this context. In consequence, we did not establish a classification of subjects based on their experience in SPL RE and we did not apply a levering questionnaire. The first session was composed of 8 Ph.D. students from the Universitat Politècnica de València, and the second was composed of 6 Ph.D. students from the Federal University of Bahia.

	Group
Training (30 min)	Introduction to FeDRE
	Exercise explanation
Session (90 min)	Task 1: Identify Use Cases
	Task 2: Specify Use Cases
	FeDRE Questionnaire

Table 3: Planning

4.2 Preparation of the case study

With regard to the Scoping activity, all the artifacts (i.e., Feature Model, Feature Specification and Product Map) were created by one domain analyst and one domain expert, who were also assisted by a scoping expert with more than 6 years of experience in SPL scoping activities. The marketing analysis was carried out on the basis of other products, with a similar purpose to that of SAVi, which are available at the *AppStore*[4]. The functionalities of these products were included in the SAVi feature model, in which 27 features were identified.

Since the FeDRE approach is flexible to support the incremental requirements specification, a set of features was selected for the case study. The selection of these features was made on the basis of which features are present in most of the products in the Product Map and are easier to be implemented. Figure 8 shows an excerpt of the Feature Model and the features selected for the case study.

Each of the 27 features from the feature model was specified according to the feature specification template. In addition, during the Scoping activity, a list of products for the mobile application for the emergency notifications domain was defined, thus allowing the creation of the Product Map artifact. With regard to the Requirements Specification for Domain Engineering activity, two requirements analysts from the team created the Glossary artifact based on the artifacts that had been created in the Scoping activity. A total of 16 relevant terms were identified for the domain. An excerpt of this artifact is shown in Table 4.

The artifacts created by the Scoping activity (Feature Model, Feature Specification and Product Map) and the Glossary artifact created by the Requirements Specification for Domain Engineering activity made it possible to create the Functional Requirements and the Traceability Matrix artifacts by applying the guidelines for specifying SPL requirements.

[4] Help.me: http://goo.gl/hSWpq | Rescue Button: http://goo.gl/asli3 | Red Panic Button: http://goo.gl/FpVsk | RescueMe Now: http://goo.gl/pDY9o

Term	Definition
Contact	It represents a person to be contacted in an emergency situation. It includes relevant information like e-mail, phone number, Facebook ID, Twitter ID.
Contact List	Collection of contacts sorted in an alphabetical order.
Twitter	Micro blogging service. It is a site on which the user can share small messages and retrieve contacts to SAVi.
User	It represents the person who uses the application.

Table 4: Excerpt from the Glossary

4.3 Collection of the data

The data for this case study was collected during the Requirements Specification for Domain Engineering activity. The SPL Functional Requirements were specified by recruiting fourteen Ph.D. students, from both universities (Spain and Brazil), who were asked to apply the guidelines for specifying SPL functional requirements (shown in Sub-Section 3.3) in order to answer the following questions: i) ***Which*** features can be grouped to be specified by Use Cases (UC)?; ii) ***What*** are the specific use cases for the feature or set of features?; iii) ***Where*** should the use case be specified?; and iv) ***How*** is each use case specified in terms of steps?

4.3.1 Which features can be grouped to be specified by UC?

This step analyzes all the features included in the increment unit for the current iteration. The subjects had to decide which of these features (see Figure 8)[5] would be specified by use cases. According to the first task of the guidelines, the most of the requirements analysts (subjects) started the iteration with the feature *Access_Control* and its children, because they are a group of features that share functionality (Task 1 from the guidelines). Since there are two ways of implementing an import contact (one optional: *Web_Access_Control*; and one mandatory: *Mobile_Access_Control*) some requirements analysts followed the guidelines (Steps 1.2 and 1.4 from the guidelines) and decided that those features would not be specified as use cases. Thus, they were specified as alternative scenarios in the use case related to the feature *Access_Control*. In a similar way, some subjects specified the features *Facebook_Import*, *Twitter_Import* as alternative scenarios from a use case of the *Import_Contact* feature, and some subjects specified the features *SMS_Destination, Twitter_Destination*, *Facebook_Destination* and *Email_Destination* as alternative scenarios in use cases related to the *Destination* feature. Following the guidelines, most of the subjects decided that the features: *Contact*, *Import_Contact*, *Add_Contact*, *Destination* and *Emergency_Numbers* would be specified as use cases. Unfortunately, there were some subjects that did not decide to specify alternative scenarios as the guidelines recommend, ignoring the variability from the feature model.

[5] An additional table with the list of features and the chosen decision is available at: http://users.dsic.upv.es/~dblanes/JUCS2013/CaseStudy.pdf

4.3.2 What are the specific UC for the feature or set of features?

After deciding which features need to be specified as use cases, the subjects had to identify which use cases should be associated to each feature. Moreover, the Traceability Matrix was incrementally filled in with traceability information between the use case and the feature. An excerpt of the traceability matrix (features X UC) is shown in Table 5.

The most common identified use cases by the subjects for the selected features are presented next[6]. The following use cases were identified for the *Access_Control* feature: *Create_User*, *Login*, *Show_Profile*, *Remember_Password* and *Send_E-mail*. The following use cases were identified for the *Web_Access_Control* feature: *Update_User,* and *Delete_User*. The following use cases were identified for the *Contact* feature: *Show_Contact*, *Delete_Contact* and, *Update Contact*. The following use case was identified for the *Add_Contact* feature: *Add_Contact*. The following use cases were identified for the *Import_Contact* feature: *Retrieve_Contacts* and *Import_Contacts*. The *Destination* feature contains the use case *Send_Notification*. The following use cases were identified for *Emergency_Numbers* feature: *Create_Emergency_Number*, *Delete_Emergency_Number* and *Update_Emergency_Number*. The corrected number of use cases to be identified by the subjects should be seventeen use cases for the selected features (Figure 8).

	UC012	**…**
Access_Control	X	
Mobile_Access_Control	X	
Web_Acces_ Control	X	

Table 5: Excerpt from the Traceability Matrix

4.3.3 Where the UC should be specified?

Since some use cases with similar behavior may be identified for different features that have the same parent, the subjects should decide where to relocate the specification for this use case (this is to avoid the redundant specification of similar behavior). When this happens, the use case was specified once only at the parent feature level. As soon as all the use cases have been identified for each feature, it is possible to start modeling the use cases. A use case package is created for each feature that will have use cases, and a use case diagram is created to include the use cases, actors and relationships among them. An example for the *Access Control* feature (use case diagram) is shown in Figure 9.

4.3.4 How each UC is specified in terms of steps?

After identifying the use cases and relating them to the features, the subjects specified each use case by taking into account the variations from the Feature Model. Table 6

[6] An additional table with the list of identified Use Cases for each Feature is available at: http://users.dsic.upv.es/~dblanes/JUCS2013/CaseStudy.pdf

shows the Functional Requirement specification for one of use case related to the *Access_Control* feature, which is the *Login* use case.

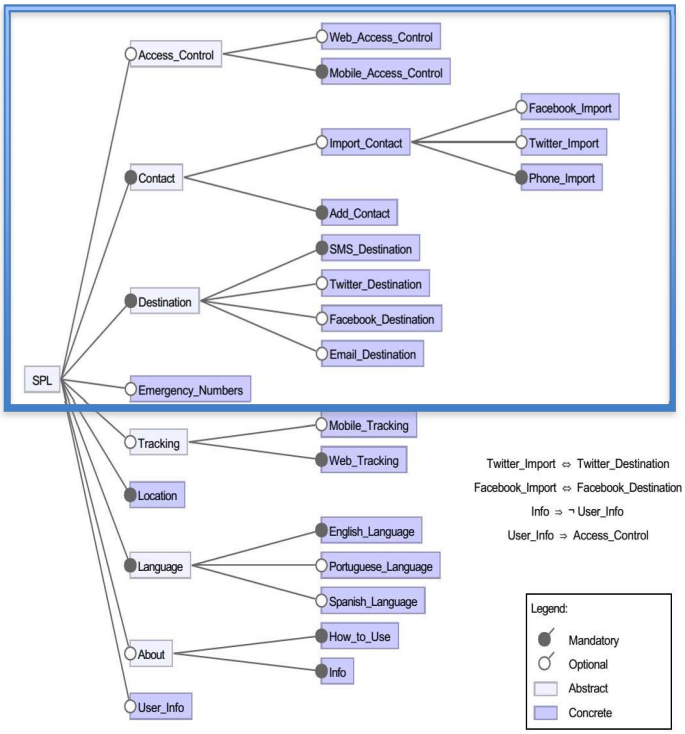

Figure 8: Selected Features from the Feature Model for the Case Study

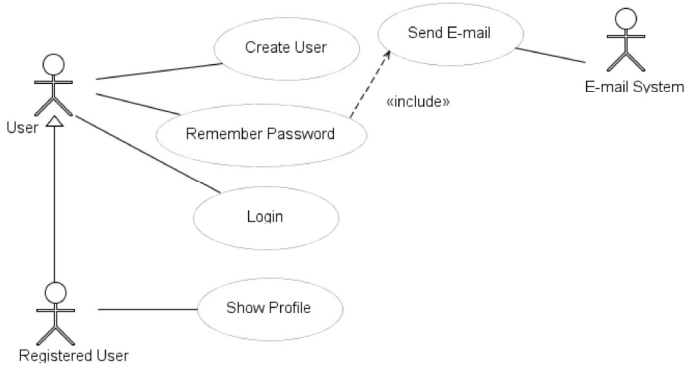

Figure 9: UC Diagram (Feature Access Control)

This use case specification has the optional feature *Web_Access_Control*, which is specified as an alternative scenario. This use case specification thus handles the

variability expressed in the Feature Model in which, depending on the selected feature, an alternative scenario can be included in the use case specification. Another advantage of using alternative scenarios to represent the variability is that of reuse. Since the alternative scenarios are specified once only within a use case, several products can be instantiated by reusing the same use case. Different behaviors may thus appear in the same use case for different products, depending on the selected features.

4.4 Data Analysis

We performed two analyses for the collected data: qualitative and quantitative. The first analysis was related to the objective variables (effectiveness, efficiency) observed during the execution of both tasks. Since we did not have a control group, this analysis was performed in order to identify deficiencies in the guidelines and improve the redaction in a qualitative manner. The quantitative analysis was performed by using closed questions, which were filled in by the subjects after the case study execution. This information was analyzed in a quantitative manner in order to check the results of the subjects' perception of ease of use and usefulness, and their statistical relevance.

***Use case id:**	UC012			
***Name:**	Login			
***Description:**	It allows a registered user to access the system			
***Associated feature:**		Access Control	**Actor(s) [0..*]:**	User
***Pre-condition:**	The User is not logged in	***Post-condition:**	The User accesses the system	
Includes To:	-	**Extends From:**	-	
***Main Success Scenario**				
Step	**Actor Action**	**Blackbox System Response**		
1	The user asks to login using the mobile	The **System** shows the username and password fields to be filled in		
2	The **User** fills in the username and password fields	The **System** validates the username and password and allows the user access		
Alternative Scenario name:		Web Access Control Login		
Condition	The user must be using a computer			
Associated feature [0..1]:		Web Access Control		
Step	**Actor Action**	**Blackbox System Response**		
2.1	Requires the login through a computer	The **System** shows a form to be filled in		
2.2	The **User** fills in the username and password and confirm	The **System** login in to Savi by using the Web Access for computers		

Table 6: Retrieve Contacts Use Case Specification

4.4.1 Qualitative analysis

Regarding effectiveness in task 1 of the case study, we measure the quotient of the right number of uses cases identified by the total number of use cases modeled in the solution (*Effectiveness_UC*). The users were able to identify correctly 62.1% of the total use cases in the task 1. In order to check the effectiveness in task 2 of the case study, we compared the specified alternative scenarios with the scenarios modeled in the solution (*Effectiveness_SCEN*). The results show that this variable has a mean of 0.523, meaning that the users were able to correctly specify 52.3% of the total use cases (see Table 7).

Additionally, for each task we measured the time used and the efficiency estimation. The results show that the subjects took around 51 minutes to complete the task 1, with an *Efficiency_UC* value of 0.216 (number right use cases / time). The subjects took around 39 minutes to complete the task 2, with an *Efficiency_SCEN* value of 0.048 (number of right alternative scenarios / time).

	Mean	**SD**[7]
Effectiveness_UC	0.621	0.182
Effectiveness_SCEN	0.523	0.447
Time task 1	51.86	13.091
Efficiency_UC	0.216	0.085
Time task 2	39	19.247
Efficiency_SCEN	0.0484	0.048

Table 7: Mean and Standard Deviation for the analyzed variables

Finally, a qualitative analysis was performed by analyzing the open questions that were included in the questionnaire. For example, some subjects suggested reformulate some guideline rules to avoid ambiguities during the use cases identification (e.g. identifying group of features that share functionality), or during the use case specification (e.g. defining alternative scenarios). Other subjects suggested including in the guidelines rules for dealing with relationships among features (includes / extends). The analysis of these quantitative data revealed several important issues that have to be considered to improve FeDRE.

4.4.2 Quantitative analysis

In this section, we discuss the results of the case study by quantitatively analyzing the data according to the hypotheses stated. All the results presented were obtained by using the SPSS v20 statistical tool with an alpha value of 0.05. The subjective variables were analyzed by comparing whether the mean of the responses to the questions related to each variable were significantly greater than the Likert neutral value[8] (equal to 3). In our case, the mean variable ranging from 1 to 5 for the

[7] SD: Standard Deviation

[8] The subjects' responses are available at:
http://users.dsic.upv.es/~dblanes/JUCS2013/Questionnarie.pdf.

measurement of both subjective variables has been considered as an interval scale [Carifio, 07]. Both variables have a mean over the neutral value 3 (see Table 8)[9].

In order to verify the hypotheses with this data, we first selected which test was most appropriate for the data. It was first necessary to check whether the data was normally distributed. Since the sample size is smaller than 50, we applied the Shapiro-Wilk test to verify whether the data is normally distributed. The results of the normality test (Table 8) show that both variables are normally distributed in this evaluation method since the results are greater than 0.05. As consequence, we check the hypotheses by performing a one-tailed t-test for independent variables with a test value of 3. The p-values obtained (Table 8) were < 0.05 ($p \leq \alpha$). As consequence we reject both null hypotheses; accepting that FeDRE is perceived as easy to use and useful.

	Mean	**SD**	**Shapiro-Wilk**	**Alpha Cronbach**	**t-**
Perceived Ease of	3.857	0.813	0.696	0.833	0.002
Perceived	3.880	0.771	0.066	0.722	0.001

Table 8: Analysis of the PEOU and PU variables

4.5 Threats to validity

The main threats to the **internal validity** of the case study are: evaluation design, subject experience, information exchange among evaluators, and the understandability of the documents. The *evaluation design* might have affected the results owing to the selection of features to be taken as input to extract the requirements when applying FeDRE. We attempted to alleviate this threat by considering a subset of features, which implied applying the complete set of the FeDRE guidelines rules. *Subject experience* was alleviated owing to the fact that none of the subjects had any experience in requirements modeling in SPL development. *Information exchange* was mitigated by monitoring the participants while they performed the tasks. We performed the experiment in two sessions but no relationships were established between Spanish and Brazilian subjects and no information was exchanged among them. We alleviated the *understandability of the material* by clearing up all the misunderstandings that appeared in each session.

The main threat to the **external validity** of the experiment is the *representativeness of the results*. To alleviate this threat and make the tasks enough representative, the complexity of the exercise was adjusted for the subjects to be able to apply every single rule of the guidelines at least once, considering that the duration of the experiment was limited to 90 minutes.

The main threat to the **construct validity** of the experiment was the reliability of the questionnaire, related to the two case study hypotheses. This *reliability* was tested by applying the Cronbach's alpha test to each set of closed questions which measured

[9] The box plots for the subjective PEOU and PU variables are shown at: http://users.dsic.upv.es/~dblanes/JUCS2013/Bloxplots.pdf

[10] P-values from the one-tailed t-test

the PEOU and PU variables, obtaining a value of 0.833, and 0.722 respectively (higher than the minimum acceptance threshold α=0.70) [Maxwell 02].

The main threat to the **conclusion validity** of the experiment was the *validity of the statistical test applied*. This was alleviated by applying the most common test that is employed in the empirical software engineering field [Maxwell, 02].

5 Conclusions and Further Work

This paper introduces the FeDRE approach to support the requirements specification of SPLs. In this approach, chunks of features from a feature model are realized into functional requirements, which are then specified by use cases. The required requirements variations can be related to the use case as a whole or to alternative scenarios inside a use case. A set of guidelines was provided to help SPL developers to perform these activities and as a means to systematize the process and ensure a correct traceability between the different requirements artifacts. We believe that this approach provides a solution that is capable of dealing with the complexity involved in SPLs with a large number of requirements and features.

The feasibility of FeDRE was evaluated using a case study involving a mobile application for emergency notifications. The results show that the analysts perceived the approach as easy to use and useful for specifying the functional requirements in this particular SPL. However, the approach needs further empirical evaluation with larger and more complex SPLs. Such an evaluation is part of our future work where we are considering the execution of a second case study to strengthen our results. We also want to check the necessity of specifying other variability types, like cross-cutting parameters variability and step variability (as presented in [Eriksson, 05]). We are working in a tool support for the approach. The web-based tool SPLICE already supports the specification of features and use cases. We plan to apply FeDRE in the development of other SPLs during the domain and application engineering processes. We also intend to extend the approach to enable it to deal with non-functional requirements, QAs in the feature model and to explore the use of model-driven techniques to (partially) automate the guidelines to check the completeness and consistency of artifacts.

Acknowledgements

This research work is cofounded by the Hispano-Brazilian Interuniversity Cooperation Program (HBP-2011-0015), the MULTIPLE project (TIN2009-13838) and the FPU program (AP2009-4635) from the Spanish Ministry of Education and Science, and the ValI+D program (ACIF/2011/235) Generalitat Valenciana.

MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT. This material has been approved for public release and unlimited distribution. Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University. DM-0000867. This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[11]), funded by CAPES, CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08 and CNPq grants 305968/2010-6, 559997/2010-8, 474766/2010-1 and FAPESB. The authors also appreciate the value-adding work of all their colleagues Loreno Alvim, Larissa Rocha, Ivonei Freitas, Tassio Vale and Iuri Santos who make great contributions to the Scoping activity of FeDRE approach.

References

[Abrahao, 11] Abrahao S., Insfran E., Carsí J.A., and Genero M. 2011. Evaluating requirements modeling methods based on user perceptions: A family of experiments. Information Science. 181(16), 3356-3378.

[Alférez, 11] Alférez, M., Lopez-Herrejon, R. E., Moreira, A., Amaral, V. and Egyed, A.: Supporting consistency checking between features and software product line use scenarios. In Proc. of the 12th Int. conference on Top productivity through software reuse (ICSR), Springer-Verlag, Berlin, Heidelberg, 20-35.

[Anquetil, 10] Anquetil, N., Kulesza, U., Mitschke, R., Moreira, A., Royer, J., Rummler, A. and Sousa, A; A model-driven traceability framework for software product lines. Softw. Syst. Model. 9(4), 427-451, 2010.

[Asadi, 11] Asadi M., Bagheri E., Gašević D., Hatala M., Mohabbati, B.: Goal-driven software product line engineering. Proc. of the 2011 ACM Symposium on Applied Computing, 691-698.

[Bayer, 00] Bayer, J., Muthig, D., and Widen, T.: Customizable domain analysis. In Proceedings of the First Int. Symp. on Generative and Component-Based Software Engineering (GCSE), Springer, 178–194.

[Bonifácio, 09] Bonifácio R., and Borba, P.:Modeling scenario variability as crosscutting mechanisms. In Proceedings of the 8th ACM international conference on Aspect-oriented software development (AOSD), ACM, 125-136.

[Bosh, 00] Bosch, J. Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach (Addison-Wesley, 2000).

[Carifio, 07] Carifio J. and Perla, R.J., 2; Ten common misunderstandings, misconceptions, persistent myths and urban legends about Likert scales and Likert response formats and their antidotes. Journal of Social Sciences, 3(3), 106-116.

[Clements, 07] Clements, P. and Northrop, L.: Software Product Lines: Practices and Patterns, Addison-Wesley, Boston.

[Davis, 89] Davis F.D.: Perceived Usefulness, Perceived ease of use and user acceptance of information technology. MIS Quarterly 13(3), 319–340.

[Czarnecki, 00] Czarnecki, K., Eisenecker, U.W.: Generative Programming: Methods, Tools, and Applications. Addison-Wesley, 2000.

[11] INES - http://www.ines.org.br

[Eriksson, 05] Eriksson, M., Börstler, J., and Borg, K.: The PLUSS approach - domain modeling with features, use cases and use case realizations. In Proceedings on 9th International Conference on Software Product Lines, Springer, 33–44.

[Griss, 98] Griss, M. L., Favaro, J., and d' Alessandro, M.: Integrating feature modeling with the RSEB. Proceeding on the Fifth International Conference on Software Reuse (ICSR), (Victoria, BC, Canada), pp. 76–85.

[Heidenreich, 10] Heidenreich, F., Sánchez, P., Santos, J., Zschaler, S., Alférez, M., Araújo, J., Fuentes, L., Kulesza, U., Moreira, A. and Rashid, A.: Relating feature models to other models of a software product line: a comparative study of featuremapper and VML. In Trans. on aspect-oriented software development VII. Springer-Verlag, Berlin, Heidelberg 69-114.

[John, 09] John, I., Eisenbarth, M. 2009. A decade of scoping: a survey. Proceeding on the 13th Software Product Lines (SPLC) (San Francisco, California, USA, August 24-28). ACM, 31-40.

[Jones, 96] Jones C., Applied Software Measurement: Assuring Productivity and Quality, McGraw-Hill: New York, 1996.

[Kang, 90] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical report. Soft. Eng. Institute.

[Krueger, 01] Krueger, C.W.: Easing the transition to software mass customization. In Proceedings of the 4th International Workshop on Software Product-Family Engineering (Bilbao, Spain, October 3–5, 2001). Springer, 282–293.

[Mashiko, 97] Mashiko, Y., Basili, V. R.: Using the GQM Paradigm to Investigate Influential Factors for Soft. Process Improvement. Journal of Systems and Software 36(1), 17-32

[Maxwell, 02] Maxwell, K.: Applied Statistics for Software Managers. Software Quality Institute Series, Prentice Hall.

[Moon, 05] Moon, M., and Chae, H. S.: An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. IEEE Trans. Softw. Eng. 31(7), 551–569.

[Mussbacher, 11] Mussbacher, G., Araújo, J., Moreira, A., and Amyot, D.: AoURN-based Modeling and Analysis of Software Product Lines. Software Quality Journal 20 (3-4), 645-687.

[Muthing, 04] Muthig, D., I. John, M. Anastasopoulos, T. Forster, J. Dorr, and K. Schmid: GoPhone – A Software Product Line in the Mobile Phone Domain. IESE. Tech report: 025.04/E. 2004.

[Oliveira, 13] Oliveira, R. P., Insfrán, E., Abrahão, S., Gonzalez-Huerta, J., Blanes, D., Cohen, S., Almeida, E. S.: A Feature-Driven Requirements Engineering Approach for Software Product Lines. In Proceedings of the VII Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS) (Brasília, Brazil, 29 September – 4 October 2013). IEEE ,1-10.

[Shaker, 12] Shaker, P., Atlee, J. M., Wang, S.: A feature-oriented requirements modeling language. In Proc. on 20th IEEE Int. Requirements Engineering Conference, Chicago, USA, 151-160.

[Soltani, 12] Soltani, S., Asadi, M., Gasevic, D., Hatala, M., Bagheri, E.: Automated planning for feature model configuration based on functional and non-functional requirements. Proc. on the 16th Software Product Line Conference (SPLC). (Salvador, Brazil). ACM, Vol. 1, 56-65.

[Wohlin, 12] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B.: Experimentation in Software Engineering (XXIII), 1-236, 2012.