# Elicitation of Use Cases for Product Lines

A. Fantechi*, S. Gnesi^, I. John+, G. Lami^, J.Dörr+

* Dip. di Sistemi e Informatica - Università di Firenze - Italy
^ CNR - Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" - Pisa, Italy
+Fraunhofer Institute for Experimental Software Engineering (IESE) - Kaiserslautern - Germany

**Abstract.** Use Cases can be employed in system requirements engineering to capture requirements from an external point of view. In product line modeling, commonalities and variabilities of a family of systems have to be described. In order to support variability modeling for product lines with Use Cases, extensions and modifications of Use Cases have to be provided. Capturing the variations characterizing the different products is a key issue for product line requirements engineering. This paper describes an approach to derive product line requirements in the form of Use Cases, starting from the analysis of user documentations of existing systems. We provide a disciplined approach to integrate legacy information found in existing documentation into product line Use Cases and illustrate this with an example.

## 1 Introduction

The development of industrial software systems may often benefits from the adoption of a development cycle based on the so-called system-families or product lines approach [18] [7]. This approach aims at lowering production costs by sharing an overall reference architecture and concepts of the products, but allows them to differ with respect to particular product characteristics in order to e.g. serve different markets. The production process in product lines is therefore organized with the purpose of maximizing the commonalities of the product family and minimizing the cost of variations [13].
In the first stage of a software project, usually called requirements elicitation [12], the information and knowledge of the system under construction is acquired. When eliciting and modeling requirements on a product line two different problems have to be addressed. On one side there is the problem of capturing requirements common to all members of the product line and requirements valid only for parts of the line members. On the other side there is the problem of specializing and instantiating the generic product line requirements into application requirements for a single product.
To deal with these problems, the relations between line and product requirements have to be represented in the modeling approach, and the concepts of parameterization, specialization and generalization need to be supported by the modeling concepts.

When building a new product line, the approach to do so can either be independent (a company starts a product line without any predecessor products), project-integrating (existing systems under development will be integrated into the product line), reengineering driven (legacy systems have to be reengineered into the product line) or leveraged (the company sets up a product line based on a product line that is already in place) [23]. User documentation that is useful as input for product line modeling can be found in the cases of project-integrating, reengineering-driven and leveraged product line engineering. Therefore, user documentation is the first choice to start the elicitation process for the information needed in product line modeling.
As developing a product line is a complex task, in depth knowledge of the problem domain often is a prerequisite for a successful product line.  So, when a company starts to do product line engineering often systems already exist that can be used as a knowledge base for the new product line. Figure 1 describes this situation.

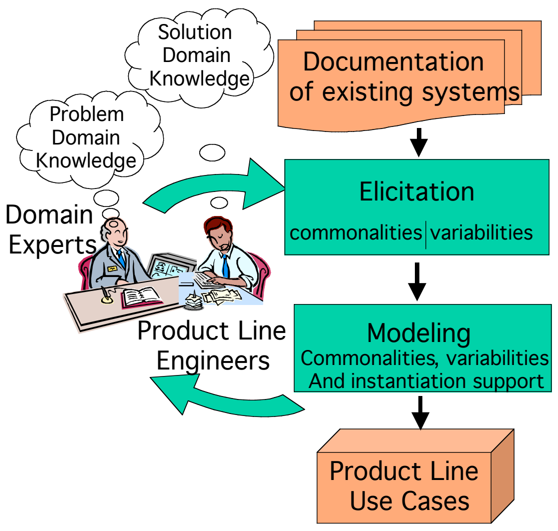

Solution Domain Knowledge
Documentation of existing systems
Problem Domain Knowledge
Elicitation
commonalities | variabilities
Domain Experts
Product Line Engineers
Modeling
Commonalities, variabilities And instantiation support
Product Line Use Cases

**Figure 1** Capturing Product Line Use Cases

Domain experts with knowledge in the problem or application domain, together with product line engineers with knowledge in the solution domain (the processes and products in product line engineering) have to elicit and model commonalities and variabilities in a highly interactive and time consuming process.
In this paper we describe an approach for elicitation and modeling Use Cases of product lines from existing user documentation. Use Cases are a powerful tool to capture functional requirements for software systems. They allow structuring requirements documents with use goals and provide a means to specify the interaction between a certain software system and its environment [8]. A Use Case defines a goal-oriented set of interactions between external actors and the system under consideration.

With the proposed approach commonalities and variabilities can be expressed and managed within Use Cases. Use Cases are able to describe both the common characteristics of all the products belonging to a product line and the variations that differentiate products among them. Use Cases describing only one line member are then obtained by an instantiation process. The primary information source used for elicitation is the user documentation of systems coming from the same application domain as the product line under development.
The paper is structured as follows: in Section 2 we describe textual Use Case, in Section 3 we describe how the notation for Use Cases can be extended in order to represent all types of variability needed to model a product line (and to support instantiation). In Section 4 we describe the elicitation of information needed for the Use Cases. We illustrate elicitation and modeling on a practical example in Section 5 and conclude the paper in section 6.

## 2 Use Cases

A Use Case [8] describes the interaction (triggered by an external actor in order to achieve a goal) between a system and its environment. A Use Case defines a goal-oriented set of interactions between external actors and the system under consideration. The term actor is used to describe the person or system that has a goal against the system under discussion. A primary actor triggers the system behavior in order to achieve a certain goal. A secondary actor interacts with the system but does not trigger the Use Case.
A Use Case is completed successfully when its goal is satisfied. Use Case descriptions also include possible extensions to this sequence, e.g., alternative sequences that may also satisfy the goal, as well as sequences that may lead to failure in completing the service in case of exceptional behavior, error handling, etc. The system is treated as a "black box"; thus, Use Cases capture who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals. A complete set of Use Cases specifies all the different ways to use the system, and therefore defines the whole required behavior of the system. Generally, Use Case steps are written in an easy-to-understand, structured narrative using the vocabulary of the domain. A scenario is an instance of a Use Case, and represents a single path through the Use Case. Thus, there exists a scenario for the main flow through the Use Case, and as many other scenarios as the possible variations of flow through the Use Case (e.g., triggered by options, error conditions, security breaches, etc.). Scenarios may also be depicted in a graphical form using UML Sequence Diagrams.

Figure 2 shows the template of the Cockburn's Use Case taken from [8]. In this textual notation, the main flow is expressed, in the "Description" section, by an indexed sequence of natural language sentences, describing a sequence of actions of the system. Variations are expressed (in the "Extensions" section) as alternatives to the main flow, linked by their index to the point of the main flow from which they branch as a variation. This natural language form of Use Cases has been widely used in industrial practice to specify use cases , e.g at Nokia [11].

| | | |
|---|---|---|
| **USE CASE #** | < the name is the goal as a short active verb phrase> | |
| **Goal in Context** | <a longer statement of the goal in context if needed> | |
| **Scope & Level** | <what system is being considered black box under design><br><one of: Summary, Primary Task, Sub-function> | |
| **Preconditions** | <what we expect is already the state of the world> | |
| **Success End Condition** | <the state of the world upon successful completion> | |
| **Failed End Condition** | <the state of the world if goal abandoned> | |
| **Primary,**<br>**Secondary Actors** | <a role name or description for the primary actor>.<br><other systems relied upon to accomplish Use Case> | |
| **Trigger** | <the action upon the system that starts the Use Case> | |
| **Description** | **Step** | **Action** |
| | 1 | <put here the steps of the scenario from trigger to goal delivery, and any cleanup after> |
| | 2 | <...> |
| | 3 | |
| **Extensions** | **Step** | **Branching Action** |
| | 1a | <condition causing branching> :<br><action or name of sub-Use Case> |
| **Sub-Variations** | | **Branching Action** |
| | 1 | <list of variations> |

**Figure 2** Use Cases template

## 3 Product Lines Use Cases (PLUCs)

Following the Product Line Engineering Process Reference Model defined in the CAFÉ project [18], and shown in Figure 3, product line development is characterized by two processes: domain engineering and application engineering. Domain engineering is the process aiming at developing the general concept of a product line together with all the assets which are common to the whole product line, whereas application engineering is the process aiming at designing a specific product.
During application engineering a customer specific application will be defined. However, differently from the usual single product development, the definition process of the customer specific application is not only influenced by the requirements of the customer but also by the capabilities of the product line.
This diagram shows that it is possible to move from the product line level (by means of the system line engineering activity) to the product level and vice versa (by means of the system line reverse engineering activity).

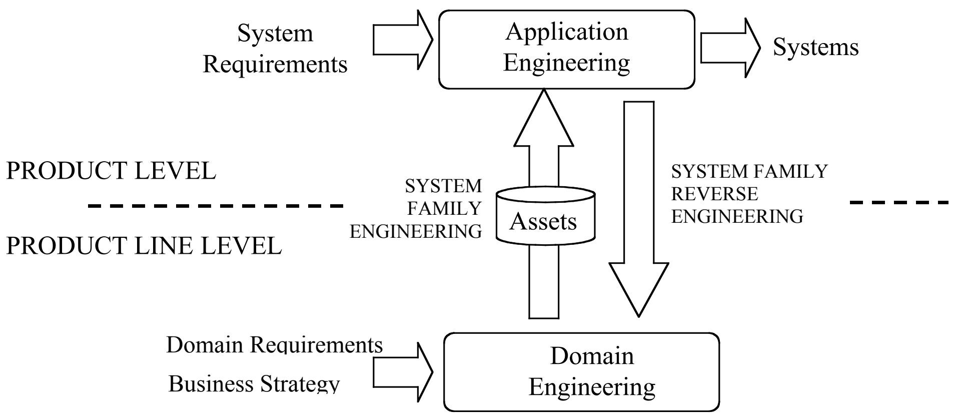


**Figure 3** The CAFÉ-Product Line reference framework

Going upwards, applications are developed considering the capabilities of the product line specializing, extending and adding line requirements. Consequently, software product lines need more sophisticated requirement processing and requirements should deal with variability.

In particular, product line requirements can be considered in general as composed of a constant and a variable part. The constant part includes all those requirements dealing with features or functionalities common to all the products belonging to the line and that, for this reason, do not need to be modified. The variable part represents those functionalities that can be changed to differentiate a product from another.

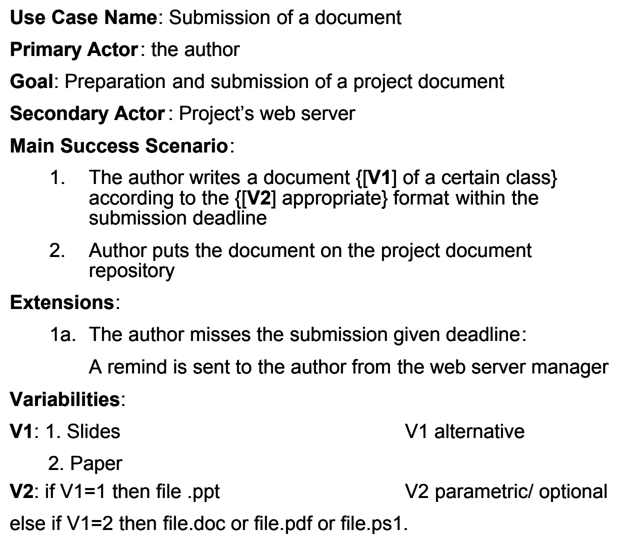
Variability can be seen from two different perspectives: the first is the product perspective where each variability has to be considered as an aspect to be instantiated. From the product line perspective a variability can be seen as a goal to be reached by abstracting all the instances related to the existing products belonging to a product line.

It is possible to move from the product line level to the product level by an instantiation process and on the contrary from the product level to the product line level by an abstraction process. In these two different processes the main objects to pay attention on are variations. A possible extension of Use Cases to express variability during requirements engineering of product lines is an extension based on structuring the Use Cases as having two levels: the product line level and the product level [5]. In this way product-related Use Cases should be derived from the product line-related Use Cases by an instantiation process.

This approach considers the variations implicitly enclosed into the components of the Use Cases. The variations are then represented by tags that indicate those parts of the product line requirements needing to be instantiated for a specific product in a product-related document. For doing that, tags are included into the Use Case scenarios (both main scenario and extensions) in order to identify and specify variations. The tags represent three kinds of variability: Alternative, Parametric, and Optional.

1. Alternative components: they express the possibility to instantiate the requirement by selecting an instance among a predefined set of possible choices, each of them depending on the occurrence of a condition;
2. Parametric components: their instantiation is connected to the actual value of a parameter in the requirements for the specific product;
3. Optional components: their instantiation can be done by selecting indifferently among a set of values, which are optional features for a derived product.

The instantiation of these types of variabilities will lead to a set of different product-related Use Cases. As an example, a Use Case in the PLUC notation is presented in Figure 4.



**Use Case Name**: Submission of a document

**Primary Actor**: the author

**Goal**: Preparation and submission of a project document

**Secondary Actor**: Project's web server

**Main Success Scenario**:

1. The author writes a document {[**V1**] of a certain class} according to the {[**V2**] appropriate} format within the submission deadline
2. Author puts the document on the project document repository

**Extensions**:

1a. The author misses the submission given deadline:

A remind is sent to the author from the web server manager

**Variabilities**:

|  |  |
|---|---|
| **V1**: 1. Slides | V1 alternative |
| 2. Paper |  |
| **V2**: if V1=1 then file .ppt | V2 parametric/ optional |

else if V1=2 then file.doc or file.pdf or file.ps1.

**Figure 4** Example of a Use Case in the PLUC notation

This Use Case describes the activities related to the submission of a project document. Let's suppose that it can be possible to submit different two types of documents: either slides (in the .ppt format) or papers (in .doc, pdf. or .ps format). Curly brackets are introduced into the Use Case elements, variables (here V1 and V2) describe the variation points within the use case. The possible instantiations and the type of the variations is given within the use case and the possible values are described with logical expressions

## 4 Elicitation of Information for Product Line Use Cases

Product Line Engineering includes the construction of a reusable set of assets. Constructing such a reusable asset base for specific products in a domain is a more sophisticated task than the development of assets for a single system because several products with their commonalities and variabilities have to be considered. This implies the

planning, elicitation, analysis, modeling and realization of the commonalities and variabilities between the planned products.
Usually, the development of a product line is not a green field task. Legacy systems exist that shall be integrated into a product line. The information from those systems is a valuable source for building the reusable assets. This information from existing systems can be found in the code, in architecture descriptions and in requirements specifications [14]. All this information can be found in documents produced during the lifecycle of the existing systems.

## 4.1 Benefits of Using Legacy Information

Until now, the information needed to build a product line model is elicited interactively with high expert involvement. As domain experts have a high workload and are often unavailable, high expert involvement is a risk for the successful introduction of a product line engineering approach in an organization.

There is a lack of guidance on how to integrate textual information found in legacy documents into product line models.

Single system elicitation methods cannot be taken as they are, because multiple documentations have to be compared, commonalities and variabilities have to be elicited and additional concepts (e.g. abstractions, decisions) are needed. Systematically integrating legacy documentation into product lines models supports:

- *Integration and reuse of textual information.*
- *Feasibility of product line modeling* by decreasing the effort the domain experts have to spend with interviews and meetings.
- *Increased acceptance of the product line in the development organization* because there is confidence in the legacy products and reusing the legacy information instead of developing everything from scratch reduces the effort to build the product line. *Better traceability from the product line to the existing system.*

There are different kinds of legacy documentation (requirements specs, user manuals, design documents...). As Use Cases are more and more used in domain modeling (e.g. [6], [13], [15]), it is important to have a closer look on how to elicit information for product line Use Cases. Use Cases describe the system from a users point of view. Therefore, user manuals are the most important source of information as input for Use Cases. In this paper, we adopt an approach for controlled elicitation, which guides product line engineers and domain stakeholders in how to elicit knowledge from existing documents and how to transform documentation into product line models. This approach is called the PuLSE[1] - CaVE-approach (Commonality and Variability Elicitation) that is integrated into the PuLSE-Framework for product line engineering [4] that is a customizable and flexible framework for all phases of product line engineering. CaVE is an approach for structured and controlled integration of user documentation of existing systems into the product line [16].

[1] PuLSE is a registered trademark of Fraunhofer IESE

With the elicitation approach common and variable features [17], Use Case elements, tasks [21] and requirements can be elicited. We focus on the elicitation of Use Case elements here. As existing systems are the basis for this approach, it can be seen as a reengineering method for documents transferring user documentation into basic elements of information for product line Use Cases. The approach was validated in two case studies [16], further case studies, as a deeper empirical validation will follow. The approach consists of the following phases (see Figure 5):

- Preparation
- Search
- Selection, change and modification

The first two steps of the approach can be performed by persons who just have a slight domain understanding, they do not have to be domain experts. The third step requires involvement of domain experts as there the valid documentation entities have to be selected. We will now describe the three steps in more detail.

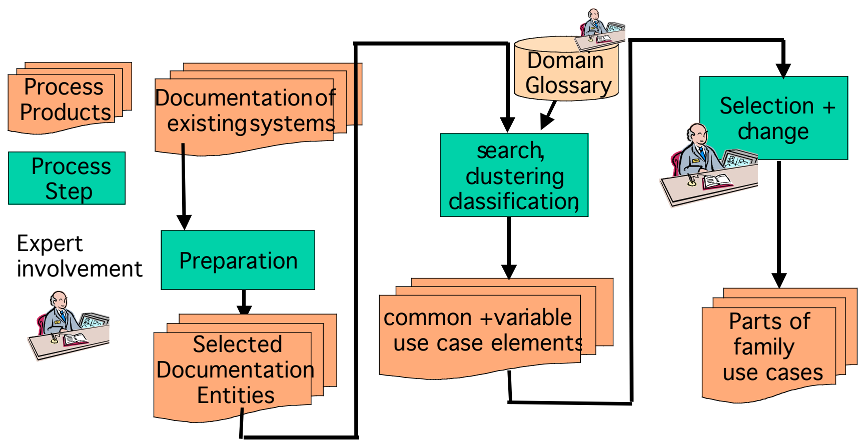

Process Products
Process Step
Expert involvement
Documentation of existing systems
Preparation
Selected Documentation Entities
Domain Glossary
search, clustering classification
common +variable use case elements
Selection + change
Parts of family use cases

**Figure 5** An outline of the elicitation approach

## 4.2 Preparation

Preparation consists of the four sub steps collection, selection, division and browsing. During collection, user documentation for the systems that should be integrated into the product line and of systems that are related should be collected to have all needed information available. In the case of a project-integrating product line adoption (c.f. section 1) these are all user-documentations of the systems currently under development (as far as they already exist), in the case of a reengineering-driven or leveraged product line adoption all user documentations of existing systems in the domain have to be considered. As parallel reading of more than one document requires divided and increased attention and leads to lower performance [28], the number of documents to be read in parallel should be reduced to a minimum. So, if there are more than 3 systems, select two or three documents that cover the variety of systems (e.g., one documentation of a low-end system, one of a high end system and one typical system) to compare for a first search in the documents. The other documents can be used to complete the elicited information after completing the search phase.

After selecting the three typical documentations, divide them into manageable and comparable parts of 3 to 10 pages (e.g., comparable subchapters). In browsing, for each of those manageable parts (or for a subset of those parts that includes typical sub domains) browse through them in order to decide the amount of variability in them. There are two alternatives:
For those document parts that differ in less than 30% of the text compare the documents in parallel in the following phases.
For those document parts that differ in more than 30% of the text, process them one after another in the following phases. Start the analysis with the biggest document.

## 4.3 Search

In the search step the identified document parts are analyzed and Use Case elements are searched. The elements to be identified in the documents, which should be sized from one word to at most 5-6 lines, are marked and tagged in the source documents. Common and variable Use Case elements that can be identified are (see. Figure ) names, actors, goals, preconditions, steps of descriptions, success conditions, extensions.

Common and variable use case elements can be identified and marked in the text with the following heuristics (specific rules-of-thumb or arguments derived from experience[27]):

- Headings of sections or subsections typically contain names of Use Cases.
- Phrases like “only by”, “by using”, “in the case of” can be markers for Use Case preconditions.
- Use Case preconditions and goals can typically be found in the beginning of a chapter.
- Use Case preconditions can be found before or within the description of a Use Case.
- Phrases like “normally” “with the exception”, “except” can mark Use Case extensions.
- Numbered lists or bulleted lists are markers for an ordered processing of sequential steps and describe Use Case descriptions.
- Sentences that describe interactions with the system in the form of “to do this…do that…” are Use Case descriptions.
- Passive voice is typically a marker for system activity (e.g. “The volume of the radio is muted” = the system mutes the volume of the radio). These sentences can be used in the Use Case description.

Commonalities and variabilities in those elements can be found with the following heuristics:

- Arbitrary elements occurring only in one user manual probably are optional elements.

- Headings or subheadings that only occur in one of the documentations can be Use Cases that are optional as a whole.
- Headings or subheadings that have slightly different names or headings or subheadings that have different names but are at the same place in the table of contents can be hints for alternative Use Cases.
- Phrases that differ in only one or a few words can be evidence for alternatives.
- If numerical values in the document differ they can be parametrical variabilities.
- Menu items that are described only in some of the documents can be hints for optional or alternative functionality (Use Cases or parts of them).

With the support of these heuristics, which help in finding a relevant part of the Use Case elements and variabilities, the user documents should be marked (e.g. with different colors for different Use Case elements and for variabilities) and integrated into an intermediate document. The list of heuristics is expected to grow as further case studies are performed. Further Use Case elements that are identified without the help of the heuristics are added. The identified elements should be extracted from the document and tagged with attributes containing the information needed for selecting appropriate elements for modeling the product lines requirements in terms of PLUCs. Table 1 shows the elements of such a notation.

## 4.4 Selection

**Table 1** Attributes for the elicited text items

| **Attribute** | **Values** | **Description** |
|---|---|---|
| ID | e.g. 1…n or doc.number | A unique identifier for the element |
| Value | text | The text of the element that was found in the document |
| Document | Identifiers | The identifiers of the documents this element was found in |
| Use Case Type | Use Case elements | The elements of Use Cases (description, precondition..) the text matches to |
| Var Type | Comm., opt, alt, param. | The hypothesis for the variability type of the element (default is commonality) |
| Parent | ID | The element, this element is part of |
| Use Case relations | Use Case Name | A possible Use Case this element is related to |
| Var relations | List of IDs | The IDs of other elements that contain alternatives or different parameters for this element |

In the last step, selection, the extracted and tagged elements have to be checked and possibly adjusted by a domain expert. The domain expert will change the elements regarding the following aspects:

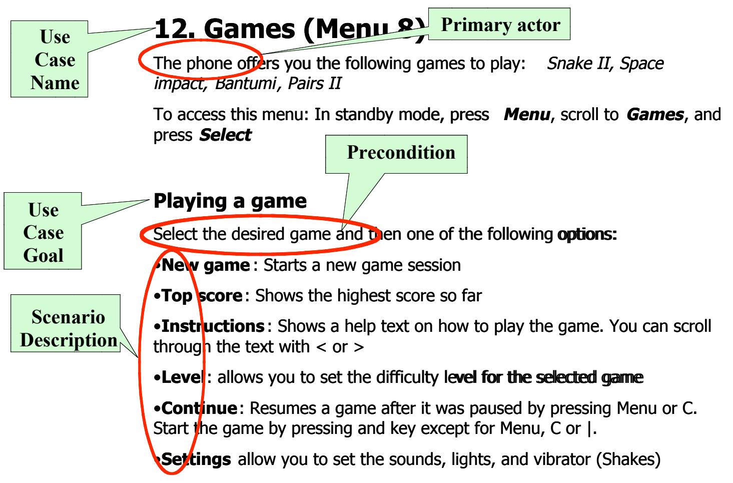
- Is a text element that was marked as a possible Use Case element, a Use Case element in the new product line?

- Is an element marked as optional/alternative really an optional/alternative element in the new product line?
- Are the Use Cases to be built out of the elements the right Use Cases to describe the systems of the product line?

The relations (see last lines of Table 1) are used to make comparisons between the documents easier, to establish traceability to the source documents and, with tool based selection, to support navigation in the elements and between the sets of documents. With these elements the domain expert and the requirements engineer can built use cases using the information about the elements collected in the tags. With the help of the tags and by putting together the elements from the document and adding parts of the use cases that are not mentioned in the document use cases in the notation described in Section 3 can be easily built.

## 5 Application on a Case Study

In this section the approach described in Section 4 to derive Use Cases using the PLUC notation is applied on an industrial case study to show its applicability:
existing user manuals of two mobile phones from the same product line that, from here in after, we indicate as P1 and P2, have been considered.
When the elicitation process is applied to a single product, the effort can be concentrated on the identification of the correspondences between the parts of the manual and the Use Case elements. Starting from a fragment of a user manual of a mobile phone, represented in Figure 6, a possible correspondence with standard Use Case elements can be shown.
When product families requirements have to be derived, starting from several documents, each related to a particular product of the product line, the process shown before is not in general sufficient. In this case all the commonalities and variabilities should be derived as the outcome of the elicitation phase. With reference to the approach described in Section 4, after the collection, selection and division phases the function descriptions of the user manuals have been identified as being the basic documentation entities for the elicitation process. In this case study the description of the GAMES functionality of the two phones P1 and P2 has been taken as example.
The browsing phase determines the amount of commonalities and variabilities between the two entities under analysis. It is then evident that the two products differ at least for the set of games provided to the user and for the presence of the WAP connection. Figure 7 shows how the parts of the user manuals of the mobile phones P1 and P2 related to the GAMES functionality can be put in relation after the preparation phase.



**12. Games (Menu 8)**

The phone offers you the following games to play: *Snake II, Space impact, Bantumi, Pairs II*

To access this menu: In standby mode, press ***Menu***, scroll to ***Games***, and press ***Select***

**Playing a game**

Select the desired game and then one of the following **options:**

•**New game**: Starts a new game session

•**Top score**: Shows the highest score so far

•**Instructions**: Shows a help text on how to play the game. You can scroll through the text with < or >

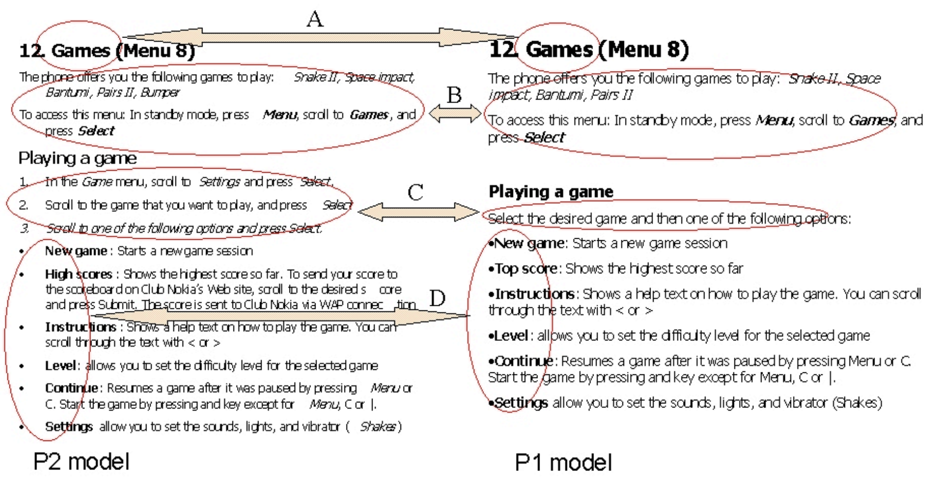
•**Level**: allows you to set the difficulty **level for the selected game**

•**Continue**: Resumes a game after it was paused by pressing Menu or C. Start the game by pressing and key except for Menu, C or |.

•**Settings** allow you to set the sounds, lights, and vibrator (Shakes)

**Figure 6** Correspondences between user manual parts and Use Case elements

In the search phase the identified parts are analysed and the Use Case elements are searched. The found commonalities and variabilities are extracted and tagged and then integrated into an intermediate document represented in Table 2 in a tabular format. Where each row of this table represents an elicited text item according to the scheme of Table 1.



**12. Games (Menu 8)**

The phone offers you the following games to play: *Snake II, Space impact, Bantumi, Pairs II, Bumper*

To access this menu: In standby mode, press ***Menu***, scroll to ***Games***, and press ***Select***

Playing a game

1. In the *Game* menu, scroll to *Settings* and press *Select*.
2. Scroll to the game that you want to play, and press *Select*
3. *Scroll to one of the following options and press Select.*

- **New game**: Starts a new game session
- **High scores**: Shows the highest score so far. To send your score to the scoreboard on Club Nokia's Web site, scroll to the desired s core and press Submit. The score is sent to Club Nokia via WAP connec tion
- **Instructions**: Shows a help text on how to play the game. You can scroll through the text with < or >
- **Level**: allows you to set the difficulty level for the selected game
- **Continue**: Resumes a game after it was paused by pressing *Menu* or C. Start the game by pressing and key except for *Menu*, C or |.
- **Settings** allow you to set the sounds, lights, and vibrator ( *Shakes*)

**12. Games (Menu 8)**

The phone offers you the following games to play: *Snake II, Space impact, Bantumi, Pairs II*

To access this menu: In standby mode, press ***Menu***, scroll to ***Games***, and press ***Select***

**Playing a game**

Select the desired game and then one of the following options:

•**New game**: Starts a new game session

•**Top score**: Shows the highest score so far

•**Instructions**: Shows a help text on how to play the game. You can scroll through the text with < or >

•**Level**: allows you to set the difficulty level for the selected game

•**Continue**: Resumes a game after it was paused by pressing Menu or C. Start the game by pressing and key except for Menu, C or |.

•**Settings** allow you to set the sounds, lights, and vibrator (Shakes)

**Figure 7 Correspondences between P1 and P2**

**Table 2** Text items elicited in the case study

| ID | Value | Doc. | Use Case Type | Parent | Var Type | Use Case relations | Var Relations |
|---|---|---|---|---|---|---|---|
| P1.1 | The phone | P1 | Actor | - | Comm. | UC Games | P2.1 |
| P1.2 | Games to play: SnakeII, Space impact, Bantumi, PairsII, | P1 | Scenario Description | - | Comm | UC Games | P2.2 |
| P1.4 | Bumper | P1 | Scenario Description | P1.2 | Opt | UC Games | |
| P1.5 | To access this menu: press menu, scroll to games, select | P1 | Precondi-tion | - | Comm | UC Games | P2.3 |
| P1.6 | In the game menu, scroll to settings and press select | P2 | Scenario Description | - | Opt | UC Games | - |
| P2.1 | The phone | P2 | Actor | | Comm. | UC Games | P1.1 |
| P2.2 | Games to play: SnakeII, Space impact, Bantumi, PairsII, | P2 | Scenario Description | - | Comm | UC Games | P1.2 |
| P2.3 | To access this menu: press menu, scroll to games, select | | Precondi-tion | - | Comm | UC Games | 1.3 |

The outcome of this phase is an intermediate document, which is obtained from both the previous ones by tagging the common parts and the detected variabilities. The final outcome of the selection phase could be a first version of a Use Case expressed in the PLUC formalism.
Realistically, we do not have to expect that the variable parts to be tagged could be identified simply by a "difference" operation between the two docs. For example, the text elements put in correspondence by the C double arrow in Figure 7 have the same meaning, but, in the case of the P1 model they seem suitable for being put in the Precondition element of the resulting Use Case, while for the P2 model they seem more suitable for being considered as belonging to the Scenario.
The resulting Use Case will depend on the decision on how to mark these text elements. Such a decision cannot be made in general but should be left to a domain expert in the selection phase of the approach. This can be made either as a case-by-case decision for each element or as a general parameter to select before applying the process. In Figure 8 a possible outcome in the PLUC format of the product line requirements related to the GAME functionality is shown.

## 6 Conclusions

In this paper we described an approach for elicitation and specification of Use Cases for product lines based on existing user documentation. Use Cases, which are quite

**Primary Actor**: the user, the {[***V0***]} mobile phone (the system)
**Goal**: play a game on a {[***V0***]} mobile phone and record score
**Preconditions**: the function GAMES has been selected from the main MENU
**Main Success Scenario**:
- The system displays the list of the available games: SnakeII, Space impact, Bantumi, PairsII and {[**V1**] additional}
- The user select a game
- The system displays the logo of the selected game
- The user selects the difficulty level by following the {[***V2***] appropriate} procedure and press YES
- The system starts the game and plays it until it goes over
- The user records the score achieved and {[***V3***] possibly} send the score to Game Club via WAP
- The system displays the list of the {[***V1***] available} games
- The user presses NO

| | | |
|---|---|---|
| ***V0***: alternative | 1. P1 model<br>2. P2 model | |
| ***V1***: optional | if *V0*=2 then Bumper | |
| ***V2***: parametric | if ***V0***=1 then procedure-A: | - press Select<br>- scroll to Settings and press YES<br>- scroll to Difficulty Level and press YES<br>- select the desired difficulty level, press YES |
| | else if ***V0***=2 then procedure-B: | - press Select<br>- scroll to Level and press YES<br>- select the desired difficulty level, press YES |
| ***V3***: parametric | if ***V0***=1 then function not available<br>else if ***V0***=2 then function available | |

**Figure 8** The resulting PLUC

common in single system requirements engineering are also often used in product line engineering to model requirements on a line of systems. The approach we describe here supports capturing of the information found in user documentation of legacy systems and the specification of this information in Use Cases that are extended with a mechanism to express variabilities. Up to now, there only small case studies exist with the approach, additional case studies and applications are described in [5] and [16]. It still has to be shown that the approach scales to large product families, but as the elicitation, analysis and modeling of the use cases can be performed subdomain by subdomain in larger systems we hope that the approach will scale.

There are several approaches for domain analysis and product line modeling. An overview on domain analysis methods like FODA [17], ODM [24] or Commonality Analysis within FAST [26] can be found in several surveys like [9] or [2]. But in most of these approaches, the integration of legacy systems into the domain analysis phase is not described in depth. In ODM [24], the primary goal is the systematic transformations of artifacts (e.g., requirements, design, code, tests, and processes) from multiple existing systems into assets that can be used in multiple systems. ODM stresses the use of legacy artifacts and knowledge as a source of domain knowledge and potential resources for reengineering/reuse. MRAM [20] is a method that describes how to analyze and select appropriate textual requirements for a product line but their focus is on the transition from domain engineering rather than on the transition between existing systems and domain engineering. There are some methods from single system requirements elicitation that describe how to elicit information from existing documents. Alexander and Kiedaisch [1], Biddle [4] and the REVERE Project

[22] focus on reusing natural language requirements in different forms. The QuARS approach [10], the KARAT approach [25] and Maarek [19] apply natural language processing or information retrieval techniques to requirements specifications in order to improve their quality. The approach that we describe here overcomes the shortcomings of other approaches by explicitly considering variability and integrating user documentation into product line modeling and modeling of use cases.

With the help of an automatic tool, the selection of the text elements and the tagging with the attributes could be performed semi-automatically. The process of analyzing a user manual using information retrieval methods [3] in a semi-automated process opens up the possibility to capitalize on the wealth of domain knowledge in existing systems considered for migration to next-generation systems. Converting these existing requirements into domain models can reduce cost and risk while reducing time-to-market. Tool support can increase efficiency of processing and correctness of the results significantly for the techniques proposed and can relieve experts and product line engineers. It is planned to develop an elicitation tool which integrates document-analysis and information retrieval techniques like indexing or morphology to support document-based modeling of the commonalities and variabilities of planned products in the domain of a product. This tool can then support stakeholders in the domain in identifying, eliciting and analyzing commonalities and variabilities in the domain, which are retrieved from the existing documents. Such a tool could integrate existing tools that, by means of natural language processing techniques, are able to analyze documents and to point out particular sentences or special wordings, to give support for the search phase.

**Acknowledgements:** This work was partially supported by the Eureka Σ!2023 Programme., ITEA (ip00004, Project CAFÉ). We wish to thank Alessandro Maccari from NOKIA for interesting discussions on the topics of this paper.

## References

1. I. Alexander and F. Kiedaisch. Towards recyclable system requirements. In ECBS'02, 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems, April 2002, Lund, Sweden, 2002.
2. G. Arango. Domain analysis methods. In W. Shaefer, R. Prieto-Diaz, and M. Matsumoto, editors, Software Reusability. Ellis Horwood, 1993.
3. R. Baeza-Yates and B. Ribeiro-Neto. Modern information retrieval. Addison-Wesley, 1999.
4. J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, and J.-M. DeBaud. PuLSE: A Methodology to Develop Software Product Lines. In Proceedings of the Symposium on Software Reusability (SSR'99), Los Angeles, CA, USA, May 1999. ACM.
5. A. Bertolino, A. Fantechi, S. Gnesi, G. Lami, A. Maccari, Use Case Description of Requirements for Product Lines, REPL'02, Essen, Germany, September 2002.
6. Robert Biddle, James Noble, and Ewan Tempero. Supporting Reusable Use Cases. In Proceedings of the Seventh International Conference on Software Reuse, April 2002.
7. P. C. Clements and L. Northrop. Software Product Lines: Practices and Patterns. SEI Series in Software Engineering. Addison-Wesley, August 2001
8. A. Cockburn. Writing Effective Use Cases. Addison Wesley, 2001.

9. J.-M. DeBaud and K. Schmid. A Practical Comparison of Major Domain Analysis Approaches - Towards a Customizable Domain Analysis Framework. In Proceedings of SEKE'98,San Francisco, USA June 1998.
10. F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. The linguistic approach to the natural language requirements quality; benefit o the use of an automatic tool. In Proceedings of the 26th Annual IEEE Computer Society Nasa Goddard Space Flight Center Software Engineering Workshop, 2001.
11. A. Fantechi, S. Gnesi, G. Lami, and A. Maccari, Application of Linguistic Techniques for Use Case Analysis, RE'02, Essen, Germany, September 2002
12. J. A. Goguen, Charlotte Linde, Techniques for Requirements Elicitation, Proceedings of the 1st International Symposium on Requirements Engineering, p.152-163, 1993
13. G. Halmans, K. Pohl Communicating the Variability of a Software-Product Family to Customers Journal of Software and Systems Modeling, Springer, 2003 to appear
14. I. John. Integrating Legacy Documentation Assets into a Product Line. In: Proceedings of the Fourth International Workshop on Product Family Engineering (PFE-4), Bilbao, Spain, October 2001.
15. I. John, D. Muthig, Tailoring Use Cases for Product Line Modeling, REPL'02, Essen, Germany, September 2002
16. I. John, J. Dörr. Extracting Product Line Model Elements from User Documentation. Technical Report, Fraunhofer IESE, 2003
17. K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.
18. F. van der Linden. Software Product Families in Europe: The Esaps and Café Projects. IEEE Software, 19(4):41--49, JulyAugust 2002.
19. Y. S. Maarek, D. M. Berry, and G. E. Kaiser. GURU: Information retrieval for reuse. In P.Hall, editor, Landmark Contributions in Software Reuse and Reverse Engineering. Unicom Seminars Ltd, 1994.
20. M. Mannion, B. Keepence, H. Kaindl, and J. Wheadon. Reusing Single System Requirements for Application Family Requirements. In Proceedings of the 21st International Conference on Software Engineering (ICSE'99), May 1999.
21. B. Paech and K. Kohler. Task–driven Requirements in Object-oriented Development. In Leite, J., Doorn, J., (eds) Perspectives on Requirements Engineering, Kluver Academic Publishers, 2003, to appear
22. P. Rayson, L. Emmet, R. Garside, and P. Sawyer. The REVERE project: experiments with the application of probabilistic nlp to systems engineering. In Pro-ceedings of 5th International Conference on Applications of Natural Language to Information Systems (NLDB'2000). Versailles, France, June, LNCS 1959, 2000.
23. K. Schmid and M. Verlage. The Economic Impact of Product Line Adoption and Evolution. IEEE Software, 19(4):50--57, JulyAugust 2002.
24. Software Technology for Adaptable, Reliable Systems (STARS). Organization Domain Modeling (ODM) Guidebook, Version 2.0, June 1996.
25. B. Tschaitschian, C. Wenzel, and I. John. Tuning the quality of informal software requirements with KARAT. In Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'97), 1997.
26. D. M. Weiss and C.T.R. Lai. Software Product Line Engineering: A Family Based Software Development Process. Addison-Wesley, 1999.
27. http://www.whatis.com
28. C.D. Wickens. Processing resources in attention. In R. Parasuraman & R. Davies (eds.), Varieties of attention (pp.63-101). New York, 1984, Academic Press.