Elicitation of Requirements from User Documentation

Isabel John, Jörg Dörr

Fraunhofer Institute for Experimental Software Engineering (IESE) – Kaiserslautern - Germany
{john; doerrj}@iese.fraunhofer.de

Abstract

This paper describes an approach for elicitation of requirements based on existing user documentation. The approach we describe in this paper supports capturing of the information found in user documentation of legacy systems, e.g., user manuals, and the specification of this information in requirements specifications, using, e.g., Use Cases. We propose a conceptual model describing the transition from user documentation to requirements artifacts describing common and variable elements of a product line model or requirements specification. We present heuristics that allow an easy identification of text elements in user documents that are then used to create a significant part of the requirements specification and product line model, respectively.

1. Introduction

The development of industrial software systems may often benefit from the adoption of a development cycle based on the so-called *system-families* or *product lines* approach [19] [6]. 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. Especially when developing more than one product, requirements elicitation is a complex task, in depth knowledge of the problem domain often is a prerequisite for a successful product family. Normally, domain experts with knowledge in the problem or application domain, have to elicit and model the requirements in an highly interactive and time consuming process. But when a company wants to build a new product, or decides to start a product line, often systems already exist that can be used as a knowledge base for the new product line [15].

Therefore, if user documentation is present, it is the first choice to start the elicitation process for the information needed in product line modeling as well as in single system development. User documentation that is useful as input for product line modeling can be found in the cases of 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) and leveraged product line engineering (the company sets up a product line based on a product line that is already in place) [25]. Furthermore, also in case of creating the requirements specification for a new single system in the product family, user documentation of recent and current products can be available.

In this paper we describe an approach for the elicitation of requirements, described in Use Cases or as textual requirements specification for product lines and single systems from existing user documentation. With the proposed approach, requirements expressed and modeled within Use Cases, features and textual requirements specifications can be elicited and specified based on existing user documentation. For describing commonalities and variabilities that are elicited from the legacy documents we use Use Case extensions such as [11] or [16]. The primary information source used for elicitation is the user documentation of systems coming from the same application domain (and often built by the same organisation) as the product line under development. By reusing the information from the user documentation of the recent and existing systems, one can produce a traceable requirements specification that is consistent and complete. Furthermore, this kind of approach ensures a systematic connection between the requirements specification and the recent and current systems.

The paper is structured as follows: in Section 2 we describe product line modeling and the benefits of using user documentation for elicitation and specification of requirements. In Section 3, we describe the conceptual elicitation model, which is the foundation for the elicitation approach we describe in section 4. As part of this approach, we present heuristics that are used to map textual elements in the user documentation to

requirements artifacts that are used to built up a significant part of the requirements specification and product line model, respectively. Finally, we conclude the paper in Section 6.

2. Motivation

Using legacy system description as input for the requirements engineering phase is on the one hand motivated by product line engineering and on the other hand by reuse principles. In this section we will describe general product line modeling concepts and the influence of legacy documentation on modeling requirements for single systems and product lines.

2.1. Product Line Modeling

Product line engineering [6] can be described as a technology providing methods to plan, control, and improve a reuse infrastructure for developing a family of similar products instead of developing single products separately. This reuse infrastructure manages commonality and controls the variability of the different products. Examples for product line approaches are PuLSE [3], FAST [30] and the SEI Product Line Practice Initiative [6].

The goal of product line engineering is to achieve planned domain-specific reuse by building a family of applications. Distinct from single system software development there are two life cycles, domain engineering and application engineering. In domain engineering the reusable asset base is built and in application engineering this asset base is used to build up the planned products. The requirements engineering phase of product line engineering is generally called domain analysis or product line modeling. Domain analysis methods provide processes for eliciting and structuring the requirements of a domain, or product line. The results are captured in a domain model. A domain model must capture both, the common characteristics of the products and their variations. The domain model is the basis for creating other reusable assets like a domain specific language or a component-based architecture. For a domain analysis method to be applicable it must be appropriate to the specific context of the organisation and the application domain and it must provide enough guidance so that it can be carried out. As in other areas of software development, the context for each domain analysis application varies, and methods that are appropriate in one context will not be in others. This fact is especially important for domain analysis because of the compound effects of inappropriate models over multiple products and over the whole lifecycle. Therefore, a generally applicable domain analysis method should be customisable to the context of the application.

Product line modeling extends requirements engineering for product lines. Apart from general requirements engineering principles, product line modeling methods have to emphasise further principles:

- Commonality and Variability
  When doing domain analysis the properties of several products have to be modelled at once. As the planned products that are analysed during domain analysis differ in their features and in their requirements, the commonalities and variabilities between those products have to be captured and adequately modelled.
- Instantiation Support
  As several products are modelled in one domain model it must be clear, which part of the model or which requirement belongs to which product. In order to have an application specific view on the product, the instantiation of the generic and variable model for several products has to be supported.
- Decision Modeling
  To get this instantiation support, the decisions that have be made also have to be captured in a separate model. This model collects and abstracts the information on which requirement is instantiated in which product and it supports the instantiation.
- Traceability
  Providing traceability from the requirements to the product and from the requirements to architecture, implementation and tests is very important in product line engineering. As a product line spans over several products and several releases of the products it has to be ensured that those two dimensions of traceability (traceability through products and through lifecycles) is provided.
- Evolution
  Product Lines are a means to cope with evolution. With product lines evolution in space (the space of the planned products) is controlled. When doing domain analysis on a portfolio of planned products evolutionary aspects are integrated and the evolution within the product portfolio is captured through commonality and variability modeling.

There are several approaches for domain analysis or product line modeling. In most product line modeling approaches, the integration of legacy systems into the domain analysis phase is not described in depth. An overview on domain analysis methods like FODA [18], ODM [27] or Commonality Analysis within FAST [29] can be found in several surveys like [8] or [2]. An approach that is often used is feature modeling [18], where features are seen as common and variable characteristics of a system that have some value to the user. Our elicitation approach supports feature modeling as with the approach, features can be identified in user documentation.

The PuLSE-CaVE (Commonality and Variability Elicitation) approach for elicitation that we describe here is integrated into the PuLSE-CDA [4] approach that builds the domain analysis component of the PuLSE™ (Product Line Software Engineering) [1] framework [3]. CDA is customisable to the project context where it will be applied. This ensures that the method and workproducts used for modeling are appropriate for the specific needs. A common basis consisting of a decision model and primitives is used and the mapping of the primitives to the modeling elements is made explicit [26]. With the help of the approach described here, information on legacy systems can be systematically integrated into a product line model developed with CDA or any other approach.

2.2. Reusing Documentation in Requirements Engineering

The information needed to build a requirements specification for a single system or a product line model is normally elicited interactively with high expert involvement (c.f. Figure 1). As domain experts have a high workload and are often unavailable, high expert involvement is a risk for the successful introduction of requirements engineering processes and methods like a product line engineering approach in an organization. Systematically using existing documentation of former or current products like user manuals to support the elicitation process reduces the expert load and makes the requirements more trustable. So, systematically integrating legacy documentation into the requirements phase has many benefits:

- Benefit 1 – Integration and reuse of textual information:
  This is achieved by integrating existing systems textual information (e.g., user manuals) into product line and requirements specifications. By integrating textual information, not only code can be reused but all assets built during the previous lifecycles.

[1] PuLSE is a registered trademark of Fraunhofer IESE

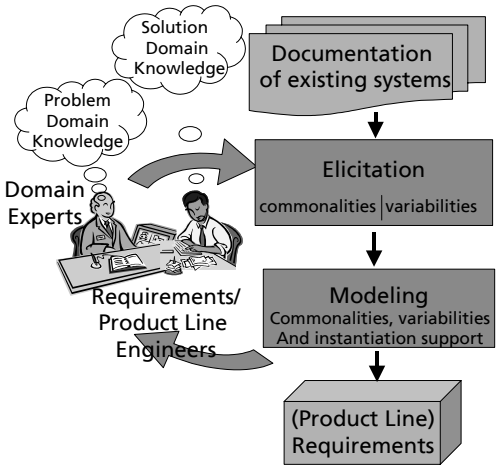

Figure 1 A requirements elicitation process

- Benefit 2 – Feasibility of requirements engineering:
  The feasibility of requirements engineering approaches and of product line modeling will be supported through these document-based techniques. A document-based technique can decrease the effort the domain experts have to spend with interviews and meetings and leads to a significant reduction of the expert load. The basic information can be elicited from documents and the experts can concentrate on planned innovative functionality.
- Benefit 3 – Increased acceptance of the product line in the development organization:
  The acceptance of the product line within the organization can be increased by reusing the legacy information, which was produced within the organization. There are two reasons for this. First, the acceptance of the product line is increased because there is confidence in the quality of the legacy products. Second, reusing the legacy information instead of developing everything from scratch reduces the effort to built the product line.
- Benefit 4 – Better traceability from the product line to the existing systems:
  Traceability to the existing system can be established only with a systematic approach which supports linking of legacy assets to the product line model built during domain analysis. Therefore, it is important to document the traces from the legacy documents to the new documents during elicitation and modeling.

There are some methods from single system requirements elicitation that describe how to elicit information from existing documents. Alexander and Kiedaisch [1], Biddle [5] , von Knethen [29] and the REVERE Project [24] focus on reusing natural language requirements in different forms. The QuARS approach [9], the KARAT approach [28] and Maarek [20] 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.

For product line modeling, 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. MRAM [21] 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. In ODM [27], the primary goal is the systematic transformation 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 but doesn't clearly state how to elicit requirements from documents.

With the approach that we present here we overcome the shortcomings of the existing approaches for product line modeling (no explicit elicitation, no systematic integration of existing documents) and for reusing requirements from single systems engineering (no consideration of variability, no use of user documentation).

3. Conceptual Elicitation Model

In this section we describe a conceptual elicitation model that is the basis for our elicitation approach described in section 0. The elicitation model consists of four parts (see Figure 2):

- A user documentation model describing the elements that are typically found in user documentations, manuals and technical specifications (e.g., sections, glossaries, and lists).
- A requirements concept model describing concepts that are typically used in requirements specifications (e.g., roles, activities, functions) independent of the notation used.
- A variability concept model describing the principle commonality and variability concepts that can be found by comparing different documents and that are used for modeling.
- A requirements artifact model describing elements of typical single system requirements specifications and product line models. These elements form a notation that is used to capture requirements (like Use Case elements, features or textual requirements). Those requirements can have, but do not have to have an explicit representation of variability.

The transition from one stage of the model to another stage is described by heuristics (specific rules-of-thumb or arguments derived from experience). These heuristics describe, e.g., which element of user documentation can be typically transformed into which requirements concept. It is also possible to directly transform requirements concepts into requirements artefacts without searching for variabilities (see arrow "single system elicitation" in Figure 2).

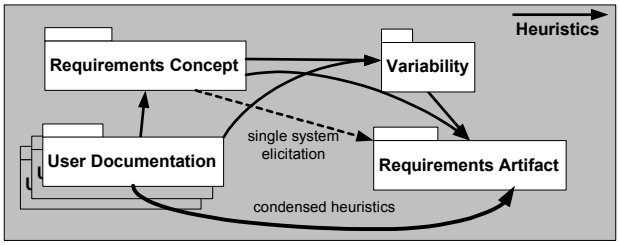

Figure 2 Overview of the model

3.1. User Documentation Model

Our user documentation model (see Figure 3) describes the principal constituents of user documents. The document types that we analyze are user documentations or user manuals that describe the functions and usage of a system and product descriptions that describe the features and technical details of a product. A document normally has a title, it often has a table of contents and a glossary and it consists of several sections. A TOC entry normally corresponds to a heading in a section. A glossary consists of a list of terms that are described in paragraphs. A paragraph consists of sentences; it can also contain figures, tables and formulas. A sentence is composed of phrases (language constructs consisting of a few words) and/or words. A phrase can also be a link (describing a reference to something inside or outside the document). Most elements of the user documentation model have attributes describing characteristics of this element (like highlighted for paragraphs and words, or numbered for lists), the attributes are not shown in the figure. This model describes the elements of a document on an adequate level for eliciting requirements concepts. Requirements concepts can be found in all parts and subparts of a document with the help of heuristics based on these elements and their attributes.

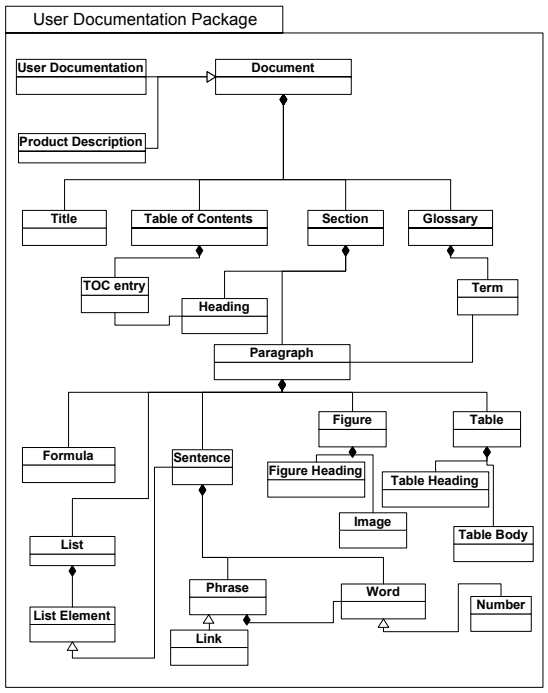

Figure 3 Model of User Documentation

3.2. Requirements Concept Model

The requirements concept model (see Figure 4) describes concepts that can be elicited from user documentation and that are normally realized or described by requirements artifacts in requirements specifications. The model describes the elements independent of a specific notation (like textual or Use Case representation). The most general requirements concept is a requirements element. A requirements element can be everything that is of value for a requirements specification. A requirements element can be a user task, a role, data, a naming convention, a constraint or a relation to something in the environment of the system to be described. Data can either be I/O data or internal data, constraints can either be usage or design constraints. A user task, that describes the high level task the user wants to perform with the help of the system can be decomposed into activities, activities consist of navigation elements, system functions and a mapping of the activities to functions.

Based on this requirements concept model and the model of user documentation described in section 3.1 we can define heuristics for the transition of elements from one model to another. Example heuristics for transitioning from a user documentation element to a requirements element are: "A heading that contains a verb often is an activity" or "a highlighted sentence containing the phrase "normally" or "with the exception" can describe constraints".

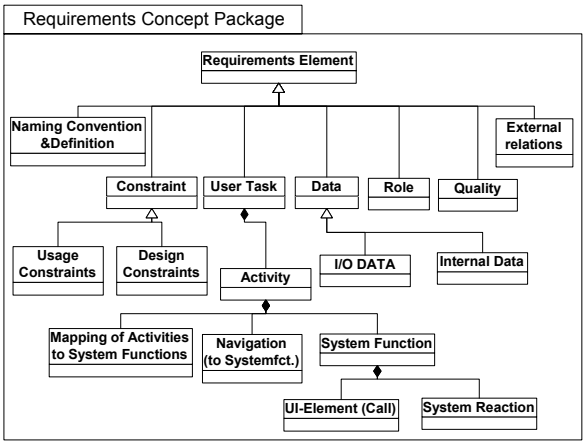

Figure 4 Requirements Concept Model

3.3. Variability Model

In the variability model, the variation aspects are described. In order to find different variability elements, the requirements elements (from the requirements concept model) found in different user documentations are compared. We decided to support the following variability elements and kinds of variation:

- **Commonality**
  No variation exists in the requirements element, the same requirements element can be found in all documentations.
- **Optionality**
  A requirements element exists in some of the products, but does not exist in some others.
- **Alternative**
  The requirements element exists in two or more different characteristics in the existing products (e.g. one product supports one database one product supports a different one).
- **Range**
  There is a range of values that is supported by the different products (e.g. the memory size can vary from 10 to 128 MB).

Based on those variability elements, heuristics can be defined that identify different variable requirements concepts by comparing the user documentations of several legacy products. These heuristics are depicted by the two arrows in
Figure 2 from user documentation and from requirements concept to variability.

Examples of such heuristics are "numbers in the document that were identified as data and belong to the same function and that have a different value can be a range variability element" or " "navigation elements that occur only in one documentation can be a hint for an optionality (an optional user interface element)".

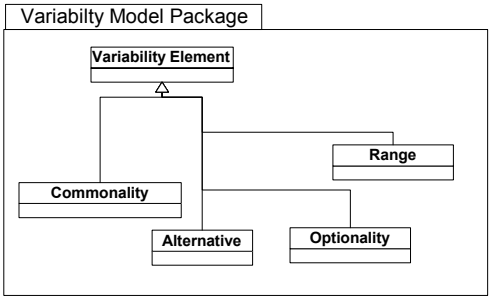

Figure 5 Variability Model

3.4. Requirements Artifact Model

The fourth package of our conceptual elicitation model is the requirements artifact model. In this model different elements of requirements specifications that can be used for single system modeling and for product line modeling are described. Different from the requirements concept model, that describes the elements on a conceptual or semantic level, the requirements artifact model describes requirements elements on a syntactic or notational level. In different kinds of requirements specifications, the same conceptual elements can be described with different notational elements, e.g. a role from the requirements concept model can be an actor in a Use Case description or a stakeholder description in a textual requirements specification.

As we also describe the application of our approach for product line modeling, we have an integrated model of variability here. The variability model we use here is the model described in the PhD thesis of Muthig [22]. In product line engineering, variability has to be made explicit in the requirements artifacts. Different extension (e.g. to UML-Use Case diagrams [16][13], or to textual Use Cases [16]) exist that make the variability explicit and give support for instantiation of requirements for application engineering. Some of these extensions use stereotypes or tags to describe variability, some extensions use extra elements to make variability explicit.

As variability is encapsulated outside the requirements artifact model in the product line artifact and the product line artifact element (see Figure 6), the model can also be used for specifying single systems requirements. At the moment we have specified two different kinds of requirements notations: Use Cases and textual requirements specifications. We have also specified artifacts that are more specific to product line modeling like feature models [18] but we will not describe them in this paper. Further requirements artefacts will be integrated into the requirements artifact model. We added different representations here, as our general approach to product line modeling [4] is customisable and highly depends on the requirements elements found in the organization that wants to do product line engineering. For doing product line engineering, we put variability elements on top of the existing notation and so can keep the notation similar to the one used in the organization before [26].

Concerning the elements in Figure 6, a Use Case diagram consists of Use Cases, actors and different relationships between the Use Cases and the actors. A textual Use Case (according to Cockburn [7]) consists of different elements like Use Case goal, precondition post condition, Use Case exceptions and the actual description of the Use Case consisting of steps. The form of requirements specification we describe here follows the IEEE Standard 830 [14]. A requirements specification is a textual document consisting of functional, non-functional and data requirements including project issues and rationales for the different requirements.

We have defined heuristics for transitioning requirements concepts into requirements artifacts (e.g., "a role is described as actor in a Use Case diagram") and heuristics that additionally include variability (c.f. Figure 2). An example of such a heuristic also considering variability is "an optional activity can be represented as an optional Use Case in a use diagram".

For the transition between elements of these packages we have found different heuristics. For users of the approach and the conceptual model those heuristics can be integrated to condensed heuristics describing the transition from user documentation directly to requirements artifacts (c.f. arrow "condensed heuristics" from user documentation to requirements artifact in Figure 2). The elicitation approach we describe now uses the heuristics in this direct form to make elicitation easier when applying the approach.

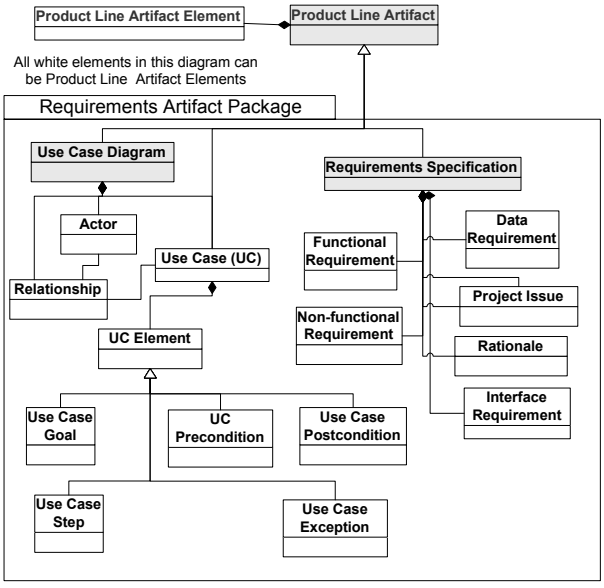

Figure 6 Requirements Artifact Model

4. An elicitation approach using user documentation

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 [15]. All this information can be found in documents produced during the lifecycle of the existing systems.

In this paper, we propose 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, PuLSE-CaVE (Commonality and Variability Elicitation) is an approach for structured and controlled integration of user documentation of existing systems into the product line [17]. The approach is compliant with the conceptual model described in Section 3 and is also very valuable for single system requirements engineering if legacy documentation is available.

With the elicitation approach common and variable features [18], Use Case elements [7], tasks describing user activities in an interactive system [23] and textual requirements can be elicited. 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 requirements specifications. The approach was applied in three case studies [17] [11], further case studies will follow. The approach consists of the following phases (c.f. 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 documentation entities have to be validated and selected. We will now describe the three steps in more detail.

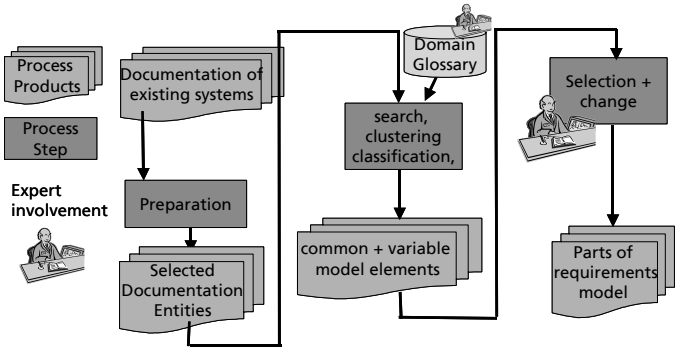

Figure 6 An outline of the elicitation approach

4.1. 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 those systems that are related should be collected to have all needed information available. In the case of a project-integrating product line adoption 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 [31], the number of documents to be read in parallel should be reduced to a minimum. So, if there are more than 3 systems, we recommend to 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.

4.2. Search

In the search step the identified user document parts containing documentation elements (c.f section 3.1) are analyzed and requirements artifacts 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 requirements artifacts that can be identified for Use Cases are for example Use Case names, actors, goals, preconditions, steps of descriptions, success conditions, and extensions. Also features and different kinds of requirements can be defined.

Common and variable requirements artifacts can be identified and marked in the text with the following heuristics The heuristics described here are heuristics that transform user documentation into requirements artifacts, so these heuristics build a connection between user documentation and requirements artifacts by using requirements concepts and variability (c.f. Figure 2 , the heuristics described here are condensed heuristics) The heuristics we show here are just examples, the complete heuristics can be found in [17]:

Use case elements

- 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.
- 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.

Requirements

- Phrases like "press", "hold", "hold down" , "press briefly", "select" , "key in" "scroll" etc. mark a dialogue with the user interface or navigation elements
- Activities or system functions are those elements that were marked as features that contain a verb
- Non functional requirements cannot be found explicitly in user manuals, but hints to non functional requirements and to qualities can be found. Shortcuts are alternative usage scenarios and can therefore be a marker for a non functional requirement like "the system shall be used in two alternative ways…."
- Adverbs and adjectives (longer, fast, quickly….) can mark NFRs, especially if a phrase or sentence appears in the user manual once with the adverb, once without. (e.g. "to turn off the radio" and "to quickly turn off the radio")
- Technical data can give a clue to non-functional attributes of the system (e.g. size of the display, battery size etc.)
- Numbers in the identified elements can be hint for a non-functional requirement (why was exactly this number chosen?)

Features

- Headings of sections or subsections typically contain features
- Features can be found in highlighted phrases (bold or italic font) or in extra paragraphs
- Technical descriptions or short descriptions of a system often contain lists of features

Commonalities and variabilities

- 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 model elements 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 model elements.
- 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.

These heuristics form a first set of heuristics that will be extended in future when applying more case studies. With the support of these heuristics, which help in finding a significant part of the requirements artifacts (i.e., of the requirements specification or product line model) and variabilities, the user documents should be marked (e.g., with different colors for different model elements and for variabilities) and integrated into an intermediate document. 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. Table 1 shows the elements of such a notation.

4.3. Selection

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 requirements artifact really a requirements artifact

Table 1 Attributes for the elicited document parts

Attribute	Values	Description
ID	e.g. 1…n or docnumber.nr	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
Requirements Artifact Type	Requirements Artifact	The reqirements artifacts (Use Case description, precondition, feature, textual requirement..) the text matches to
Var Type	commonality, optionality, alternative, range	The hypothesis for the variability type of the element (default is commonality)
Parent	ID	The element, this element is part of
relations	Requirements Artifact Name	A possible requirements artifact this element is related to
Var relations	List of IDs	The IDs of other elements that contain alternatives or different parameters for this element

that shall be integrated into the requirements specification and product line model, respectivelly?

- Is an element marked as optional/alternative really an optional/alternative element in the new product line?
- In case we have want to build a product line: Are the product line models to be built out of the elements the right models to describe the systems of the product line?

The relations (see 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.

4.4. Results

The results of the approach are approved requirements artifacts that can easily be integrated in requirements specifications and product line model elements, respectively. Which model elements should be elicited depends on what the modeling approach used needs as primitives. 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 build Use Cases and requirements using the information about the elements collected in the tags.

We have applied the approach in three case studies until now, one of the case studies is described in [11], the others will be described in [17]. In two of the case studies we analyzed user documentation from the embedded system domain, one case study was from the information systems domain. In the case studies use case elements, functional and non-functional requirements and features were the primary elements found by comparing the documentation of three to five legacy systems.

5. Conclusions

In this paper we described an approach for elicitation and specification of requirements specifications and product line models, respectively, based on existing user documentation. Use Cases, which are quite 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 described here supports capturing of the information found in user documentation of legacy systems to use them in requirements specifications and product line models, respectively. We presented heuristics that allow an easy identification of text elements in user documents based on a conceptual model.

With the help of a supporting 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 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.

Acknowledgements

This work was partially supported by the Eureka Σ!2023 Programme, ITEA , Ip00004, Project CAFÉ and Ip00103, Project Empress.
We want to thank Alessandro Fantechi, Stefania Gnesi and Guiseppe Lami for performing the joint case study described in [11] that influenced the work described here.

References

[1] I. Alexander and F. Kiedaisch. Towards recyclable system requirements. In Proceedings of ECBS'02, 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] 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.

[4] J. Bayer, D. Muthig, and T. Widen. Customizable Domain Analysis. In Proceedings of GCSE '99, Erfurt, Germany, September 1999

[5] Robert Biddle, James Noble, and Ewan Tempero. Supporting Reusable Use Cases. In Proceedings of the Seventh International Conference on Software Reuse, April 2002.

[6] P. C. Clements and L. Northrop. Software Product Lines: Practices and Patterns. SEI Series in Software Engineering. Addison-Wesley, August 2001

[7] A. Cockburn. Writing Effective Use Cases. Addison Wesley, 2001.

[8] 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.

[9] 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.

[10] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari, Application of Linguistic Techniques for Use Case Analysis, RE'02, Essen, Germany, September 2002

[11] A. Fantechi, S. Gnesi, I.John, G.Lami, J. Dörr Elicitation of Use Cases for Product Lines. Submitted to RE '03, 2003

[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] IEEE-Std 830-1998 IEEE Guide to Software Requirements Specifications, The Institute of Electrical and Electronics Engineers, New York, 1998

[15] 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.

[16] I. John, D. Muthig, Tailoring Use Cases for Product Line Modeling, REPL'02, Essen, Germany, September 2002

[17] I. John, J. Dörr. Extracting Product Line Model Elements from User Documentation. Technical Report, Fraunhofer IESE, 2003, to appear

[18] 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.

[19] F. van der Linden. Software Product Families in Europe: The Esaps and Café Projects. IEEE Software, 19(4):41--49, July/August 2002.

[20] 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.

[21] M. Mannion, B. Keepence, H. Kaindl, and J. Wheadon. Reusing Single System Re-quirements for Application Family Requirements. In Proceedings of the 21st International Conference on Software Engineering (ICSE'99), May 1999.

[22] Dirk Muthig A Light-weight Approach Facilitating an Evolutionary Transition Towards Software Product Lines. PhD Theses in Experimental Software Engineering. Dissertation, Fraunhofer IRB, 2002.

[23] 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

[24] P. Rayson, L. Emmet, R. Garside, and P. Sawyer. The REVERE project: experiments with the application of probabilistic NLP to systems engineering. In Proceedings NLDB'2000. Versailles, France, June, LNCS 1959, 2000.

[25] K. Schmid and M. Verlage. The Economic Impact of Product Line Adoption and Evolution. IEEE Software, 19(4):50--57, July/August 2002.

[26] K. Schmid and I. John. Generic Variability Management and its application to Product Line Modeling in Proceedings of the First Workshop on Software Variability Management, Groningen, 2003.

[27] Software Technology for Adaptable, Reliable Systems (STARS). Organization Domain Modeling (ODM) Guidebook, Version 2.0, June 1996.

[28] B. Tschaitschian, C. Wenzel, and I. John. Tuning the quality of informal software requirements with KARAT. In Proceedings of REFSQ'97, 1997.

[29] A. v. Knethen, B. Paech, F: Kiedaisch und F. Houdek. Systematic Requirements Recycling through Abstraction and Tracability. Proceedings of RE02, Essen, 2002.

[30] D. M. Weiss and C.T.R. Lai. Software Product Line Engineering: A Family Based Software Development Process. Addison-Wesley, 1999.

[31] C.D. Wickens. Processing resources in attention. In R. Parasuraman & R. Davies (eds.), Varieties of attention (pp.63-101). New York, 1984, Academic Press.