

Building Domain Models from Legacy Documentation Assets

Isabel John

Fraunhofer Institute for Experimental Software Engineering (IESE)
Sauerwiesen 6
D-67661 Kaiserslautern, Germany
+49 (0) 6301 707 - 250
Isabel.John@iese.fhg.de

Keywords: Product Line Engineering, Domain Analysis, Generation of Non-Code Artifacts

Classification: PhD, 1st Year's Work

1 Introduction

Product Line Engineering is an approach for planned, structured and product-centred reuse of code and non-code artifacts. Having a broad expertise and a certain domain understanding in the domain of the product line is one of the key factors and practice areas in Product Line Engineering [CN99]. Basis of domain understanding are usually legacy systems with many existing legacy assets in different forms like code, documentation etc. which should not be ignored from an economical and quality perspective, when starting to develop a product line. Those assets are a rich source of knowledge and offer easily available information, which already exists and is specific to the domain.

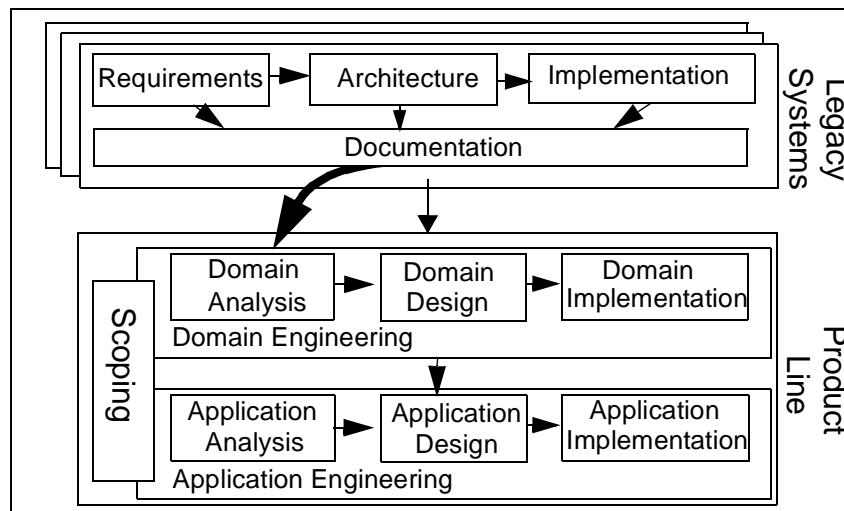


Figure 1. Legacy integration into Product Lines

Figure 1 shows a product line lifecycle consisting of domain engineering and application engineering extended with some past lifecycles of legacy systems in the product line domain. Some of the information contained in these lifecycles can be accessed through the produced documents (and through experts if they are still available).

In my PhD I will develop techniques for the structured integration of legacy documentation assets into a product line domain model. I will develop a process for the elicita-

tion of domain model elements (or product line requirements) from existing documents. The process will be supported with a tool, which uses Information Retrieval and Document Analysis techniques to capture commonalities and variabilities from documents and makes suggestions for model elements.

2 Legacy Systems, Documents and Domain Modeling

When starting a product line in an organization there are often expected to be legacy systems which already have been built in the product line domain. The integration of those existing systems into a planned and to be built product line can happen on different levels like analysis and integration of code, of existing modules or components, integration of the architecture or of requirements and general reuse and integration of knowledge and expertise in the problem domain. Each one of these levels has meaningful information for the product line under construction.

In my PhD I will focus on knowledge which is useful for the domain analysis phase in Product Line engineering. There are several approaches for domain analysis like FODA [KCH+90], ODM [ODM96], Commonality Analysis within FAST [WL99] or CDA within PuLSE [BFK+99]. But in most of these approaches, the integration of legacy systems into the domain analysis phase is not described in sufficient depth. ODM [ODM96] is one example where software systems are described as a source of legacy knowledge which can be integrated into the product line by reverse engineering. But there is lack of systematic support for integration of legacy documentation assets into the early lifecycle phases of a product line.

3 The Role of Legacy Documents

Existing requirements and existing knowledge can normally be found in documents. The information about the legacy system which exists in requirements and domain knowledge has been recorded in documents of several forms and purposes, like requirements specifications, user documentation or general domain descriptions. These documents and the information which can be elicited from them should be considered when performing a domain analysis for a product line.

In the context of the introduction of a product line approach in a software development organization my PhD will lead to the integration of the documents produced in former and ongoing software development projects in the domain into the product line engineering process. Through analysing the documents and integrating their information into the product line, documented domain knowledge and requirements can be reused for domain modeling like code can be reused by transformation through reengineering. Documents or documentation assets are representations of information concerning the legacy system and the domain, and so a valuable source for domain understanding. Typically during the lifecycle of a software product many documents with different purposes and different target audience are produced. Those documentation assets can be requirements specifications, design documents, comments in code, user manuals, contracts, or other material concerning the domain. The assets can serve as knowledge sources for the incremental introduction of product lines into the organization which produced those legacy assets. Especially in smaller companies, the only documenta-

tion assets that can be found often and are rather up to date are user manuals. In larger organizations, this situation is different. Due to defined processes, larger organizational units and more distribution of work, more documents can be found in larger organization. Companies following a defined development process have to develop certain documents. So there are many documents concerning legacy systems in the domain which can serve as a valuable knowledge source.

Requirements specifications and user manuals describe the legacy systems from a perspective which is relevant for domain modeling. It is reasonable to concentrate on the analysis of requirements specifications and user manuals of all the systems the development organization has built in the domain of the product line. For document processing there are already approaches from single system development (like [MB89] or [AG97]) to integrate certain aspects of legacy documents into the early phases of software development.

Different domain modeling approaches use different element types or primitives for modeling. Examples for those primitives are Features from FODA [KCH+90], textual commonalities and variabilities from FAST [WL99] or classes, objects, relationships or use-cases which are used in domain modeling extensions of object-oriented modeling and UML like FeaturRSEB [GFd98]. Those model elements are concepts of the application domain and can be found as words, in statements or phrases, as sentences or as chapter headings in legacy documents. Not every phrase or chapter heading is a concept useful for domain analysis, so they have to be carefully selected and proposed to the domain experts for further and more concrete modeling.

4 The Document Processing Approach

As motivated above, we will focus on requirements specifications and user manuals for document processing because they are expected to cover a large amount of information which is useful for domain modeling.

Processing the documents will help experts to come to a complete, correct, and unambiguous domain model. The information in the documentation of different systems will help to identify commonalities and variabilities among different systems. The elements found in the old systems are of course not exactly the same as those of new systems in the product line. But eliciting variabilities by comparing legacy documents can help product line engineers and domain experts to look for variations within the model in the right places. Legacy documentation alone of course cannot provide the correct and complete model but it can just give hypotheses for model elements. The described techniques (which are described in more detail in [Joh01]) provide product line engineers with basic concepts and so with the knowledge for further and deeper analysis within the domain, together with the domain experts. The steps to be performed when analysing the documents are:

- Preparing the Documents

For comparing and analysing the documents one should collect documents of one type. It is more useful to compare the user manual of one system with the user manual of other systems and requirements specifications with other requirements specifications. This ensures that terms used in the documents are used with the same or at least a sim-

ilar meaning. Each document should be reworked in order to have a table of contents for a first shallow comparison and it is also useful to have single chapters extracted to be able to compare manageable packages.

- Analyzing the commonalities and variabilities:

When analysing the documents for a further product line, commonalities and variabilities must be elicited. The way of analysing the documents depends on the primitives the domain modeling approach has and which should be extracted from the documents. Depending on the Domain Analysis approach used, features, objects and relations or use-cases should be identified and elicited from the documents.

Candidates for classes are Nouns. The nouns, classes can be derived from, are the ones often used in the textual part of the document. Those nouns should be domain specific, so it is important to filter the terms from other domains. This can be reached through filtering out those nouns with the highest frequency and including a list of commonly used nouns to be ignored during the analysis. Relationships between classes can be found by analyzing the context of the nouns already identified. Common features can be found in a similar way.

When looking for variabilities in legacy documentation, the differences between the documentation of the different legacy systems have to be analysed. Potential optional model elements are those that can be found in one (or some of the) documentation but not in all of them. For the identification of alternatives in different systems the context of the model element is important. If in the context of a subdomain (which can be isolated within a chapter of the documentation) different elements are described they can be candidates for alternatives. If for example in the spelling-checker subsection of the user manuals of two different text processing products you find the phrases “the languages english, spanish, german and french are supported” and “only english language is supported”, the common and alternative languages can be collected from these sentences with “language support” as their higher level commonality.

- Representation of the results

The result of document processing are candidate features, classes, use-cases and other model elements. These candidates should be collected in lists and tables sorted by their relevance and organized and graphically represented according to already identified subdomains. The model elements can then be used as basic material and input for a further ascertainment of the product line domain model. The elicited model elements, partial models commonalities and variabilities can serve as a basis for well directed questions and so tighten the phase of modeling where expert involvement is needed.

5 Conclusions

Using document-related techniques as an initial input for domain modeling has many advantages. By integrating the legacy documentation assets into the product line completeness, systematics, traceability and reliability are increased. The amount of reuse is increased by reusing other sources than code, the acceptance of the product line within the organization can be increased by reusing the old information which was produced within the organization. A document based technique can decrease the time the domain experts have to spent in interviews and meetings to a minimum. So, expert load reduc-

tion is the main advantage I see in using document-based techniques.

My PhD consists of three parts, theory, tool and validation. In the theory part will develop a method for the integration of legacy knowledge into a product line domain model through investigation of legacy documents. I will support this method with a tool which analyses documents like requirements specifications or user manuals with the help of domain analysis and information retrieval techniques (c.f. [BYRN99]) and generates model elements and partial models for a product line. 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. With a tool, models can be generated semi-automatically and so traceability can be easily achieved. I will validate the method and the tool with case studies and controlled experiments in the context of our product line projects and within student internships. The method and tool I develop in my PhD will be integrated into PuLSE-CDA, the domain analysis method of our product line framework PuLSE [BFK+99].

6 References

- [AG97]V. Ambriola and V. Gervasi. Processing natural language requirements. In *Proceedings of International Conference on Automated Software Engineering*. IEEE Computer Society Press, 1997.
- [BFK+99]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 Symposium on Software Reusability, SSR'99*, Los Angeles, USA, May 1999.
- [BYRN99]R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, 1999.
- [CN99]P. Clements and L. Northrop. A framework for software product line practice — Version 2.0. Technical report, SEI, 1999.
- [GFd98]M. Griss, J. Favaro, and M. d'Alessandro. Integrating feature modeling with the RSEB. In *Proceedings of the Fifth International Conference on Software Reuse*, Vancouver, Canada, June 1998.
- [Joh01]I. John. Integration legacy documentation assets into a product line. *submitted to International Workshop on Product Family Engineering PFE-4*, October 2001.
- [KCH+90]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, Carnegie Mellon Software Engineering Institute, November 1990.
- [MB89]Y. S. Maarek and D. M. Berry. The use of lexical affinities in requirements extraction. In *Proceedings of the Fifth International Workshop on Software Specification and Design, Pittsburg PA, 1989*, 1989.
- [ODM96]STARS Technical Report STARS-VC-A025/001/00. *Organization Domain Modeling (ODM) Guidebook, Version 2.0*, June 1996.
- [WL99]D. M. Weiss and C.T.R. Lai. *Software Product Line Engineering: A Family Based Software Development Process*. Addison-Wesley, 1999.