AMPLE
Aspect-Oriented, Model-Driven, Product Line Engineering
Specific Target Research Project: IST-33710

Framework for identifying commonalities and variations in requirements

ABSTRACT
This framework describes systematic identification of commonalities and variabilities in software product line specifications. It leverages Information Retrieval and Natural Language processing techniques and tools that help to abstract requirements from existing specifications of a given domain into a feature model. The framework also provides support for horizontal traceability from requirements to features and has been evaluated in the context of industrial applications, where requirements documents are potentially unstructured and textual.

Document ID:	AMPLE D1.2
Deliverable/Milestone No:	D1.2
Workpackage No:	WP1
Type:	Deliverable
Dissemination:	PU
Status:	final
Version:	V1.1
Date:	2008-01-07
Author(s):	Vander Alves, Awais Rashid (ULANC), Christa Schwanninger (Siemens), Christoph Pohl, Andreas Rummler (SAP)

Project Start Date: 01 October 2006, Duration: 3 years

History of Changes

Version	Date	Changes
0.1	2007-12-12	Initial Version
0.2	2007-12-19	After Siemens' and SAP's feedback
1.0	2007-12-28	Introduction, conclusion, formatting
1.1	2008-01-07	After ULANC's review

Contents

List of Figures

List of Tables

1 Introduction

In Software Product Line (SPL) development, Domain Engineering is an essential activity consisting of the development of reusable artifacts to be used during Application Engineering for the derivation of products. In particular, within Domain Engineering, Domain Analysis involves defining a set of reusable requirements for the systems in the domain [CE00]. Such set is organized in a domain model, such as a feature model [CE00].

According to the survey and comparative study of Requirements Engineering (RE) techniques for SPL conducted in D1.1 [KAK+07], although Domain Analysis is supported in most RE techniques, the survey has also identified that most of the SPL approaches for RE assume that the documents used in the requirements elicitation are structured uniformly. However, according to industrial partners, this assumption often does not hold in practice: requirements documents are often textual and heterogeneous, having different levels of abstractions, different formats and structure.

Moreover, the survey also revealed a lack of appropriate tool support for domain analysis tasks in such approaches, especially given the nature of requirements, as reported by industrial partners. Such support is essential for both feasibility and scalability of approaches, since many potentially large requirements documents may be given as input to domain analysis, which otherwise could be considerably time consuming. Finally, according to that survey, traceability is supported in most cases only in the vertical case and horizontal traceability could be improved by appropriate integration of Model-Driven Development approaches, which would define and map the meta-models for different elements in the problem space and from the problem space to solution space artifacts.

In this context and as foreseen by D1.1 [KAK+07], the framework presented in this document describes how developers can systematically identify commonalities and variabilities in SPL requirements specifications. It relies on a novel combination of Information Retrieval (IR) and Natural Language Processing (NLP) techniques and tools that together help to abstract individual requirements from existing requirement specifications of a given domain into a fully fledged feature model. The framework has been proposed in the context of industrial applications, where requirements documents are potentially unstructured and textual. The framework also provides support for horizontal traceability from requirements to features; the framework is compared to state-of-the-art approaches and evaluated in an industrial context.

The remainder of this document is structured as follows. First, Section 2 briefly reviews essential background for the framework, whose rationale and steps are next described in Section 3. Section 4 reports evaluation of the framework, as well as directions for its further refinement and required tool support. Finally, Section 5 offers concluding remarks and directions for future research.

2 Background on NLP and IR techniques

The framework presented in this document leverages existing state-of-the-art NLP and IR techniques. Accordingly, this section briefly reviews essential related background for the framework. NLP techniques are described in Sections 2.1 and 2.2, whereas IR techniques in Sections 2.3 and 2.4.

2.1 Wmatrix

Wmatrix [Ray07, Ray03] is a web-based information extraction environment locating various NLP tools on a web server and providing interface to those tools. A user of Wmatrix begins by uploading his/her texts, also called corpus, to the web server via a web browser. The first corpus annotation tool applied to the text is the hybrid part-of-speech (POS) tagger, CLAWS [GS97] which assigns a part-of-speech tag to every word in running text with about 97% accuracy. A second layer of annotation is applied by SEMTAG, a semantic tagger. This tool assigns a semantic field tag to every word in the text with about 92% accuracy. The resulting annotated files are presented to the user in a workarea and Wmatrix prepares word, POS and semantic tag frequency lists. These can be downloaded but can also be browsed using the web browser application. The user can select a word or tag from the lists and see a standard key word in context concordance for that item. This is prepared on the fly from the corpus on the web server. Users are guided towards interesting words or tags to investigate further by comparing frequency lists from their corpora to standard textual norms provided by frequency lists produced from the British National Corpus for example.

2.2 EA-Miner

The EA-Miner tool [SCRR05] uses the Wmatrix natural language processor to pre-process the input documents and get relevant information. Wmatrix uses part-of-speech and semantic tagging, frequency analysis and concordances to identify concepts of potential significance in the domain. Part-of-speech analysis automates the extraction of syntactic categories from the text (e.g., nouns, verbs). EA-Miner is an Eclipse plug-in using NLP features from Wmatrix to identify base concerns, early aspects and crosscutting relationships between them. For the purpose of the tool base concerns are any matter of interest at the requirements level that can be decomposed in a unit of the chosen decomposition model. For example, if the chosen decomposition model is viewpoint-oriented, a base concern is a viewpoint while if the technique used is UML-based, a base concern can be a use case or a scenario (at a finer-grained level). An early aspect is a requirements-level aspect; an entity that semantically crosscuts a concern. The early aspect modularises responsibilities that would otherwise be scattered across several concerns or tangled inside one concern if the aspectual model was not applied. Therefore, early aspects provide a means for mod-

ularising the crosscutting behaviour at the requirements level and applying it to the base concerns using a composition mechanism. EA-Miner helps to achieve separation of concerns at the requirements level by modularising crosscutting concerns as early aspects. This helps to improve requirements maintainability by facilitating change management.

2.3 Latent Semantic Analysis

Latent Semantic Analysis (LSA) [DDF+90] is an information retrieval technique addressing the problem that search words input to search engines are not necessarily the ones used to index the documents. This problem is twofold. First, polysemy–words having multiple meanings–tends to decrease precision of such engines; for example, when prompted for the search word "performance", the search engine could return documents related to how well a computing task is performed as well as documents related to acting. Second, synonymy–a concept being denoted by different words–tend to decrease recall. For example, a concept might have been indexed by one word and might be looked up by one of its synonyms.

The relevance of the problem addressed by LSA belongs not only to the information retrieval domain, for improving the flexibility of search engines, but also to software engineering, whereby a method is provided for comparing textual requirements, which has been explored recently for relating requirements to sources of requirements [SS06] and to provide a clustering mechanism for requirements [KMB06].

LSA addresses this problem by assuming the indexed documents to have inherent noise due to polysemy and synonymy. Despite such noise, it assumes there is still a latent structure within the document, which it then tries to capture and use for comparison of documents and terms, which is useful for search engines as described previously and also for comparing software requirements documents. In order to accomplish this, it relies on a dimensionality reduction technique using Single Value Decomposition (SVD) [DDF+90]. The intuition behind such strategy is that in such a reduced dimension space there is a flexible notion in comparing documents, considering not only words that match, but also how often they occur together in the rest of the documents: words of documents occurring together in the rest of the documents suggest that these documents are similar.

First, it models the documents by a frequency matrix $X_{t\times d}$, where *d* is the number of documents and *t* the number of words within them; this matrix is decomposed into a product of matrices using SVD ($X_{t\times d} = T_{t\times m} \times S_{m\times m} \times D_{m\times d}$), where:

- $m \leq min(t, d)$;
- S is the diagonal matrix of singular values;
- T and D have orthogonal and unit-length columns.

SVD guarantees that there is only one such decomposition; next, dimension reduction is performed in this product by reducing m to k ($k \leq m$). The reduced dimension matrix is proven to be the best fit of dimension k and is assumed to provide a better representation of the original matrix, being free from the original noise. The dimension *k* is empirically determined. Based on this product, a computation can be performed for determining the similarity matrix between the input documents.

Tool support for LSA is essential, since its value lies in addressing a large number of documents containing many terms. Accordingly, Prospect [SS06] supports LSA by parsing documents as input and performing stop word removal, stemming, LSA computation, and graphical display of the similarity matrix using different layouts and according to different thresholds. It has been used to relate requirements to their sources [SS06].

2.4 Hierarchical Agglomerative Clustering

Clustering groups data objects based only on information found in the data that describes the objects and their relationships [Eve93]. The goal is that the objects within a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups. The greater the similarity within a group and the greater the difference between groups, the better the clustering. Examples for the use of clustering are as follows:

- Software Product Lines: a significant body of knowledge in SPL [Bos00, GBS01, CZZM05, RBSP02] advocates that a feature is a higher level abstraction than requirements; recent work models features as cluster of requirements;
- Information Retrieval: thousands of documents matching a query may be grouped into categories reflecting different aspects or categories of the query; such categories can be further decomposed hierarchically;
- Business. Businesses collect large amount of information on current and potential customers. Clustering can be used to segment customers into a small number of groups for additional analysis and marketing activities;
- Biology. Clustering has been used recently to group genes having similar functions.

An important category of clustering algorithms, in the scope of our work, is hierarchical, where elements are nested during the clustering process. In particular, in Agglomerative Hierarchical Clustering (AHC) the process starts with points as individual clusters, which are successively merged according to their similarity. The corresponding algorithm can be described as follows:

```
for each k from maximum proximity to minimum
  determine nodes close by proximity k
  add such nodes to level k of the cluster tree

for each pair of clusters such at k and k-1 levels,
  if the latter is a superset of the former,
      then the former refines the latter.
```

The output of this algorithm is a tree, representing the hierarchical relationship among the clusters. The intuition behind the first loop is to determine the features bottom-up in the tree. Features deeper in the tree have higher similarity than ones near the root. In addition, as similarity decreases, it may be that some features contain others. The second loop is used to determine the sub-feature relationship among the features, which is done on the basis of the set inclusion relationship. A variation of this algorithm has been used by Chen et al to cluster requirements into features [CZZM05], being represented in a configuration (instantiated feature model).

3 Framework

The framework for identifying commonalities and variations in requirements is described in this section. First, its underlying hypotheses are explained (Section 3.1); then its rationale and overview are described (Section 3.2), followed by each of the method steps (Sections 3.3 to 3.6).

3.1 Hypotheses

In this subsection, we present the hypotheses underlying the definition of the framework:

H1) The goal is to generate feature models, as a means for describing the domain, from an existing set of requirements. The features should be related to requirements.

H2) Heterogeneity of documents. The existing requirements documents have different structure and level of abstraction. This occurs in both industrial case studies of the AMPLE project. In terms of structure, some documents already contain use cases, whereas other textual description of scenarios. In terms of abstraction, documents can be at different levels, such as SAP's requirements documents: a Market Requirements Definition (MRD) is written by Solution Management or Product Definition, capturing the demand view; in turn, a Software Requirements Specification (SRS) is the agreed response of Development (architects) on what to produce, i.e., the supply view; use cases in SRS refine scenarios from MRD. We also note that both facets of heterogeneity can also occur within one document, like in Siemens' requirement specification of the Smart Home case study.

H3) Configurations versus partially instantiated feature models. One requirement document can denote either one configuration or a partially instantiated feature model. According to the industrial partners, both cases occur in practice and thus the requirement document could have both intra-application and inter-application variability.

H4) Different granularity of variability. Variability can be either inter application or intra application. Achieving the goal of the task requires identifying inter-application variability rather than intra-application variability.

H5) We should define some relationship between features and requirements. Industrial partners report that, regardless of what an organization chooses–a feature as a group of requirements or vice-versa–it is a fact that there is a grouping of finer-grained entities that form coarser-grained ones that together are an interesting increment in functionality for a customer of the system. In particular, a considerable body of work in SPL [Bos00, GBS01, CZZM05, RBSP02] tends to see a feature as a higher level abstraction than requirements, e.g., a feature as a cluster of requirements. We assume this view in the framework.

H6) Merging multiple requirements documents may entail considering the requirements at a higher level of granularity. For example, the merge could operate on the configurations corresponding to each application. However, we still need to validate this and check whether merging before and clustering the requirements afterwards would be better.

H7) There is similarity within different requirements documents. Since the requirements documents specify applications in the same domain, it is assumed that there will be overlapping of the requirements of these applications. Such overlapping may not be apparent at first, due to different formats and abstraction levels of the different documents. Therefore, finding the similarity by taking information inside the documents and from the domain into account is the specific challenge here.

3.2 Rationale and Overview

The rationale behind the framework is the following. Given that the goal of the framework is to derive feature models from requirements (H1) and that there is a difference of abstraction level between features and requirements (H2-H5), the framework then needs to provide a higher level view of requirements. Once this higher level view is obtained for each requirements document, which is accomplished in pre-clustering, requirement similarity determination, and clustering steps, such view is then combined in a merge step (H6), which will then explore the latent similarity within the documents (H7).

Figure 1 illustrates the framework. The input to the framework consists of a set of documents, where each document comprises requirements specifications of different application(s). The method then, for each such document, performs a **pre-clustering step**, whereby latent variability in the requirements specification is detected. Next, a **requirement similarity determination** step is employed, which organizes the requirements into a graph, according to their

similarity. Clusters of requirements are then identified and these are abstracted further into a configuration or into a partially instantiated feature model during the **clustering** step. Finally, considering the configurations or partially instantiated feature models corresponding to all requirement documents, these are merged into a full fledged feature model in the **merging** step. The following subsections detail each of these steps.

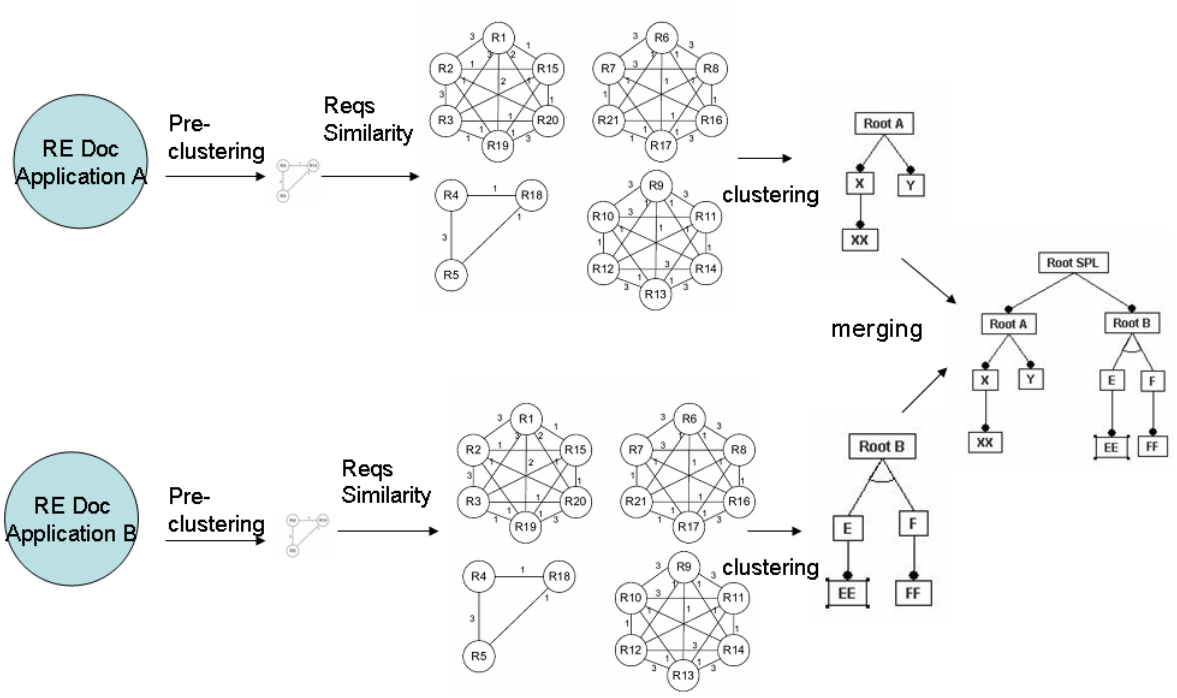

Figure 1: Framework for identifying commonalities and variations in requirements.

3.3 Pre-Clustering

According to Hypotheses 2 and 3, requirements may already have some structure, possibly hierarchical relationship among requirements and variability, for instance. In the fragment below, for example, extracted from the requirement specification document of the Smart Home case study, we see a use case description and its variants:

```
2.1.1.2 First contact
The end user connects the first time his new gateway with the SGP
from any device that has a web browser (e.g. a PC). The gateway
registers in the network (at the SEP) and gets the latest firmware
and all local services he subscribed for (initial package). When
he accesses the internet, he gets redirected to the subscriber
portal, showing the status of the first contact progress after login
with initial credentials.
```

```
Variations:

Reconnect: The end user disconnects the gateway and reconnects
later in time (also case of power failure). The gateway connects
to the SEP and checks for updates after a random delay (to avoid
network storms after wide area power failure)

Moving: The end user moves to a new apartment and connects the
gateway again.
```

Such structure should be captured and then used at the end of the clustering step. Therefore, in the requirement similarity step and clustering, such requirements would be already abstracted by a single one identified in the pre-clustering phase, e.g. the use case name. In order to identify variability already present in these documents, we determine words whose semantic category denotes variability. This is accomplished by using the capabilities of Wmatrix (Section 2.1), since it also determines the semantic categories of words in a document. For example, for the fragment below, extracted from the Smart Home case study requirement specification, the output of Wmatrix would be

```
first contact progress after login with initial credentials.
Variations:  Reconnect: The end user disconnects the gateway and
reconnects later
```

We then look at the context of such words (e.g., Variations) and identify the enclosing requirement and variants. These are then stored and considered as a single requirement for the following step, which then will not interfere with this pre-existing relationship.

In addition to variability, pre-existing hierarchical relationships between requirements could also be captured during pre-clustering and kept orthogonal from the requirement similarity determination step. We could apply some heuristics as proposed by John et al [JDS03], such as heading-subheading identification in order to find such hierarchical relationship. For example, in the fragment below, extracted from a requirement specification document of the Smart Home case study, the heading-subheading structure reflects such hierarchical relationships between requirements:

```
2.1.1 Remote Service Usage
Two sub cases need to be considered:

2.1.1 Web client uses local service: The web client logs into
the subscriber portal and gets the UI of all available services.
For local services, the subscriber portal request the RAM
for a connection key pass and then connects to the SGP to generate
a user interface with the actual status of the services. Actions
to the services are routed to the service gateway.
```

```
2.1.2 Web client uses networked service: After login to the
subscriber portal, the portal asks the RAM to generate service
session IDs. These IDs are included into the URLs of the portal
site referring to external network services. When accessing a
network services, the service evaluates the session key at the
RAM to authenticate the user. After this, the request is served
with a web page.
```

Similarly to the variability identified in this step, the hierarchical relationship can be stored and used later in the clustering step. In the requirement similarity determination step, only the top level requirement is used.

3.4 Requirement Similarity Determination

The goal of this step is to determine a similarity relationship among the requirements of each requirement specification document. This is used in the process of abstracting them in order to obtain the SPL feature model. In order to accomplish this, the technique employed is LSA (Section 2.3). It is flexible in comparing documents, not only taking into account words that match, but also taking into account how often they occur together in the rest of the documents: words of requirements occurring together in the rest of the documents suggest that these requirements are similar.

This flexibility is important, since according to Hypothesis 2, documents input to the framework are heterogenous, thus suggesting the inherent noise assumed by LSA. It has been used previously in other scenarios for relating requirements [KMB06, SS06], but, to the best of our knowledge, not yet in the SPL context. The additional benefit of using the technique is that it provides automation support, in comparison to previous work on relating requirements.

In order to use this technique, we employ the Prospect tool (Section 2.3). It receives as input a set of requirements and performs the LSA computation. The LSA documents correspond to requirements. Results can be visualized for different thresholds. Also, the parameter of the method, i.e., number of dimensions is determined empirically. Figure 2 illustrates the process behind this step, and Figure 3 shows the output of the tool for a set of requirements. The similarity graph is represented internally in the tool as a triangular matrix, which is used in the following step.

3.5 Clustering

After requirement similarity determination for each document, and according to Hypothesis H5, this step further abstracts the requirements into configurations or partially instantiated feature models. The result of the previous step is a similarity relationship graph, which according to previous experience [KMB06] and to our own experience, is generally a single connected component. We need to set a threshold in order to help cluster determination, grouping more meaningful requirements. After choosing a threshold, which can be done visually or

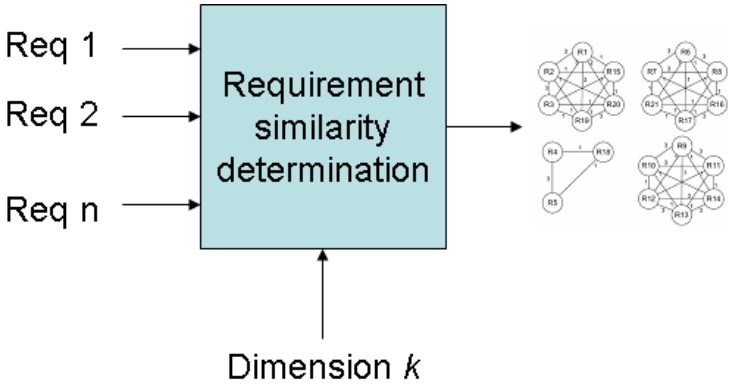

Figure 2: Requirement similarity step.

using heuristics, we apply AHC (Section 2.4) in order to derive the configuration or partially instantiated feature model. Figure 4 illustrates this step of the framework.

The use of the AHC algorithm is similar to the one employed by Chen et al [CZZM05], the difference being that: 1) in addition to a configuration, a partially instantiated feature model can also be generated in this step, which is in accordance with Hypothesis 3 (Section 3.1); 2) we apply a final step to recover requirements and variants identified previously in the pre-clustering step (Section 3.3); 3) we have identified the need to employ a strategy to find names of features in the configurations/partially instantiated feature models. The strategy can take into account part-of-speech information (e.g., nouns) as well as frequency of occurrence and structure of the requirements composing the feature, such as whether it is a heading or top level use case. In the work by Chen et al [CZZM05], no such strategy has been defined, as feature names are suggested in an ad hoc manner, which has limited scalability for the requirements satisfying the hypothesis in Section 3.1, which we assume are realistic in the SPL domain.

3.6 Merging

The previous steps are applied for each requirements specification document, thereby resulting in a series of configurations and partially instantiated feature models. Each of these represents a high level view of an application or small set of applications. In order to have an unified view of the domain of the software product line, these models are then merged into a full-fledged feature model, as illustrated by Figure 5.

The approach used is similar to previous work [CZZM05], whereby configurations, previously clustered from similar requirements, are incrementally merged using a depth-first algorithm comparing individual nodes of such configurations. Our approach differs in the following:

- **Semantics of nodes**. When comparing the individual nodes, we take into account their semantic information using Wmatrix (Section 2.1), thus

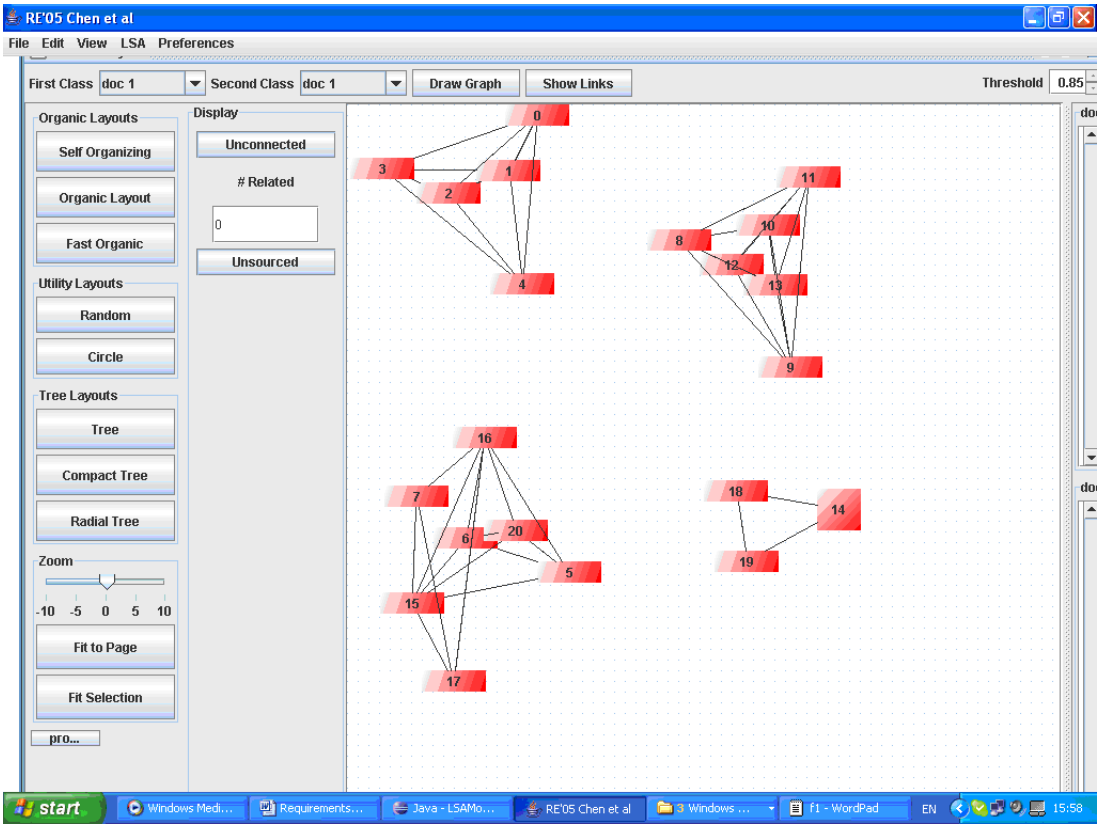

Figure 3: Visual representation of requirement similarity in Prospect.

making the comparison more flexible and resilient to underlying synonymy;

- **Merged elements**. The elements to be merged may include partially instantiated feature models and not only configurations;
- **Feature type determination**. The determination of whether a feature is optional, alternative, or or-feature is performed by analyzing the context of its requirements by searching for words whose semantic categories identified by Wmatrix match these concepts.

4 Evaluation

The framework has been partially evaluated in some case studies (Sections 4.1 and 4.2). Further evaluation is on going work. We also identify further areas for refinement of the framework (Section 4.3) and tool support (Section 4.4).

4.1 Library management system

Our framework builds on previous work ([CZZM05]), by also having requirement similarity determination, clustering, and merge steps. However, there is a striking difference in how requirement similarity determination is performed. In the

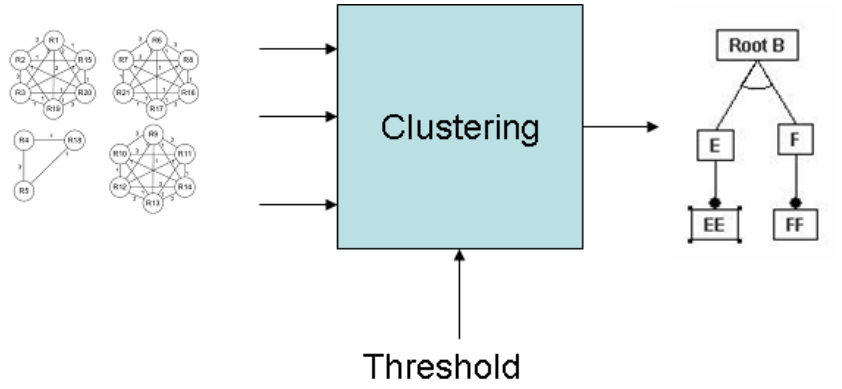

Figure 4: Clustering step of the framework.

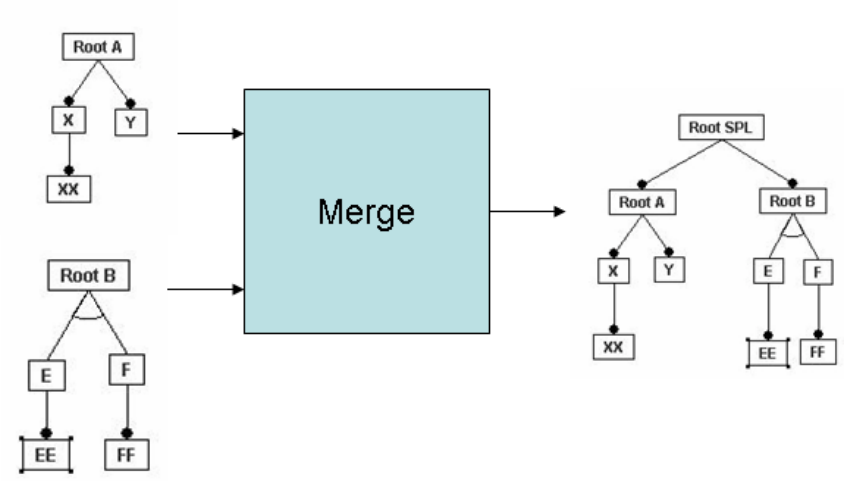

Figure 5: Merge step of the framework.

former case, the concept of *resource* is used to define similarity: requirements are similar whenever they share resources; nevertheless, how this approach works is not clear: in fact, the authors do not present an algorithm for accomplishing this task; instead, it is done manually based on the knowledge of the person who is doing the similarity analysis.

On the other hand, our approach relies on LSA (Section 2.3) to identify similar requirements. Despite the fact that the parameter *k* (number of dimensions) has to be determined empirically, LSA provides a degree of automation and it lays on a sound linear algebra foundation. In order to compare its results to those obtained by previous work, we applied LSA to the same set of requirements [CZZM05] and the result is already represented in Figure 3. The number of clusters match Chen et al's results [CZZM05] and there is overlap of most of the clusters; the differences (top left most cluster and bottom right most cluster)reflect the notion of similarity based on the concept of resources. The clustering and the merging steps of both approaches would be the same for this example, hence we have not compared these.

Table 1: Requirements of Library Management System [CZZM05].

Requirement code	Requirement text
R0	Lend a copy of a book.
R1	Return a copy of a book.
R2	Renew a copy of a book.
R3	Reserve a book.
R4	Cancel a reservation for a book.
R5	Add a user of the library.
R6	Remove a user of the library.
R7	Modify the information of a user.
R8	Add a copy of a book to the library.
R9	Remove a copy of a book from the library.
R10	Remove all copies of a book from the library.
R11	Get a list of books in the library by title.
R12	Get a list of books in the library by author.
R13	Get a list of books in the library by publisher.
R14	Find out what books are currently checked out by a user.
R15	Get a list of users by address.
R16	Get a list of users by name.
R17	Find out all the reservations made by a user.
R18	Remind a user his/her lent book is overdue.
R19	When a book is overdue, automatically remind the user who has lent the book.
R20	Find out the total number of library users.

4.2 Smart Home

The framework has been partially evaluated in the Smart Home case study. Further evaluation is on going work. Siemens' Smart Home case study has been chosen as the first industrial case to be used in assessing the framework because its requirements are mostly textual, which is supported by our current tools, and at the moment the framework is still being tuned for requirement structure detection. SAP's Sales Scenario case study will be used in due course.

The pre-clustering and the requirement similarity steps have been determined for two documents ("Requirements from Architecture point of View" and "Requirements Totally Integrated Home"). The first step basically consisted of identifying variability related words and encompassing requirements and variants as well as hierarchical structure in heading/subheading and use case/use case descriptions and variants. The pre-clustering was performed manually, because we still need to customize Wmatrix (Section 2.1) to identify variability related terms; additionally, we do not yet have tool support for identifying

structure in the documents, such as heading/subheading.

Requirement similarity determination was performed with LSA using Prospect. Figure 6 shows the result for "Requirements from Architecture point of View" and Figure 7 for "Requirements Totally Integrated Home". The threshold was adjusted visually in order to obtain more cohesive clusters. Accordingly, the number of dimensions was empirically determined. Initially, some of the requirements were very fine-grained, as a consequence of breaking the text into single sentences representing one requirement. Consequently, they tended to appear connecting different clusters. Later we used more coarse-grained requirements to avoid this. The initial results so far show that, while some detected emerging clusters are relevant to building a feature model, the accuracy of the requirement similarity determination step still needs to be improved, which suggests a need to integrate this step better with the pre-clustering step in order to capture pre-existing structure and semantics in the requirements. This is on going work, as described in Section 4.3.

Clustering has been performed in the "Requirements from Architecture point of View" document. Figure 8 illustrates the result. The strategy for determining feature names still has to be employed and refined. One finding is that similar requirements tend to occur close in the document. We still need to apply clustering to another requirements documents in the same domain and perform the merging step.

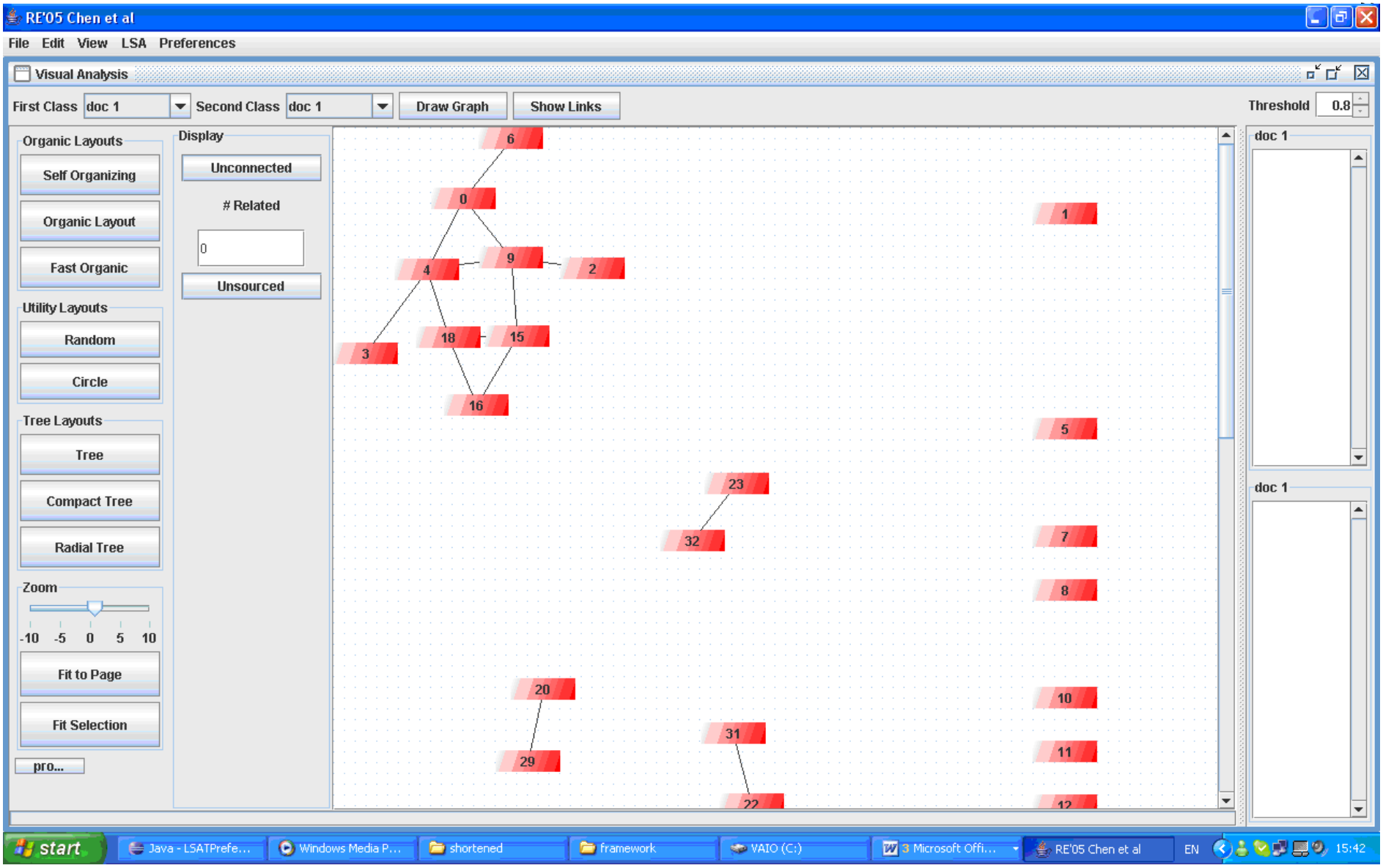

Figure 6: Requirement similarity determination in Requirement from Architecture point of view.

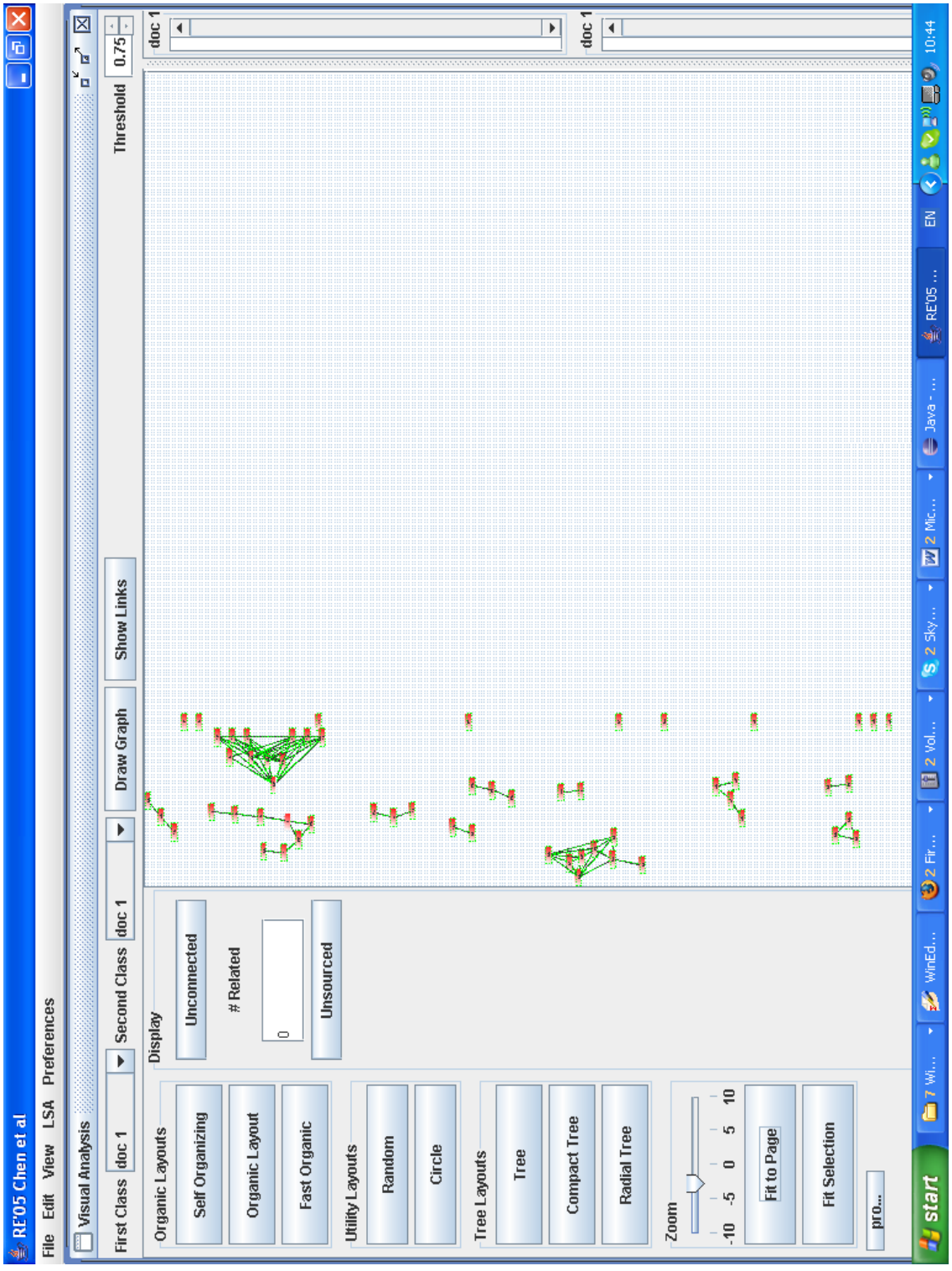

Figure 7: Requirement similarity determination in Requirements Totally Integrated Home.

Figure 8: Clustering Requirement from Architecture point of view document.

4.3 Further Enhancements of the framework

As explained in the previous section, the framework is being evaluated and refined. The following enhancements are necessary:

- Better integration of the pre-clustering and requirement similarity determination steps; this integration can capture latent structure in the requirements and use them for the benefit of the clustering step, which occurs after these two initial steps; this is also expected to help to improve the precision and recall of the requirement similarity determination step;
- improvement of the pre-clustering step to detect different kinds of structure, such as hierarchical relationship among the requirements; similarly to the previous enhancement, this is expected to improve accuracy of the requirement similarity determination step;
- refinement of strategy for naming features in clustering. As the AHC algorithm proceeds higher in the feature hierarchy, features refer to an increasingly high number of requirements, as shown in Figure 8. Therefore, naming them is a non-trivial task. Our initial strategy considers part-of-speech information (e.g., nouns) as well as frequency of occurrence and structure of the requirements composing the feature, such as whether it is a heading or top level use case.

4.4 Initial requirement list for further tool support

The current framework already leverages existing NLP and IR tools in order to build a feature model from existing requirements documents. Nevertheless, some enhancements of such tools and their integration are still necessary:

- Customize Wmatrix to detect variability related words. This is necessary in the pre-clustering step for detecting latent structure and in the merging step for classifying feature types (optional, mandatory, or-feature). This enhancement in the tool is expected to be a simple task, as it already offers support for customization based on a specific lexicon;
- Integrate EA-miner and Prospect. EA-Miner performs requirement elicitation and analysis using AOSD concepts; this could be used in the pre-clustering step in order to detect crosscutting relationship among requirements; such structure would then be fed into Prospect, which performs the requirement similarity determination;
- Implement a tool for assisting in detecting structure during pre-clustering; this tool could implement a number of strategies aiming to detect hierarchy (heading/subheading), variability and commonality, abstraction(use case/use case description) as well as others proposed elsewhere [JDS03].

5 Conclusion

In this document, we have presented a framework describing how developers can systematically identify commonalities and variabilities in SPL requirements specifications. It relies on a novel combination of IR and NLP techniques and tools that together help to abstract individual requirements from existing requirements specifications of a given domain into full fledged feature model. The framework has been proposed in the context of industrial applications, where requirements documents are potentially unstructured and textual. The framework also provides support for horizontal traceability from requirements to features; the framework is compared to state-of-the-art approaches and evaluated in an industrial context.

The framework is being further evaluated and refined. Partial results comparing it to related work show that it offers concrete tool support for relating and abstracting requirements into features; additionally, such results suggest that the framework accuracy can be improved by better integration of the pre-clustering and requirement similarity determination steps, improvement of the pre-clustering step to detect different kinds of structure, such as hierarchical relationship among the requirements, and refinement of strategy for naming features in clustering. Further refinement and evaluation will be addressed in future work.

Additional future work consists of extending tool support for this framework, by customizing Wmatrix to detect variability related words, integrating EA-miner and Prospect, and implementing a tool for assisting in detecting structure during pre-clustering. This will be done in the context of Task 1.7 of the AMPLE project. Another task from the project, directly related to the results of this framework is Task 1.3, which defines a meta-model for aspectual requirements modelling and composition. The framework presented here imposes constraints on such meta-model, such as traceability support from features to requirements.

References

[Bos00] J. Bosch. *Design & Use of Software Architectures - Adopting and Evolving a Product Line Approach*. Addison-Wesley, 2000.

[CE00] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.

[CZZM05] K. Chen, W. Zhang, H. Zhao, and H. Mei. An approach to constructing feature models based on requirements clustering. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 31–40, 2005.

[DDF+90] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[Eve93] B. S. Everitt. *Cluster Analysis*. Edward Arnold, 1993.

[GBS01] J. Van Gurp, J. Bosch, and M. Svahnberg. On the notion of variability in software product lines. In *Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WISCA'01)*, pages 45–54, Amsterdam, The Netherlands, August 2001.

[GS97] R. Garside and N. Smith. A hybrid grammatical tagger: Claws4. In *Corpus Annotation: Linguistic Information from Computer Text Corpora*, 1997.

[JDS03] I. John, J. Dörr, and K. Schmid. User documentation based product line modeling. Technical Report IESE-Report No. 004.04/E, Fraunhofer IESE, 2003.

[KAK+07] J. Kovacevic, M. Aferez, U. Kulesza, V. Alves, A. Moreira, J. Araujo, V. Amaral, A. Rashid, and R. Chitchyan. Survey of state-of-the-art in requirements engineering for software product lines and model-driven requirements engineering. Technical Report Deliverable D1.1, AMPLE Project, 2007.

[KMB06] L. Kit, C. Man, and E. Baniassad. Isolating and relating concerns in requirements using latent semantic analysis. In *OOPSLA'06*, pages 383–396, 2006.

[Ray03] P. Rayson. *Matrix: A statistical method and software tool for linguistic analysis through corpus comparison.* PhD thesis, Computing Department, Lancaster University., 2003.

[Ray07] P. Rayson. *Wmatrix: a web-based corpus processing environment*. World Wide Web, `http://www.comp.lancs.ac.uk/ucrel/wmatrix/`, 2007.

[RBSP02] M. Riebisch, K. Böllert, D. Streitferdt, and I. Philippow. Extending feature diagrams with UML multiplicities. In *Integrated Design and Process Technology, IDPT-2002*, pages 45–54, 2002.

[SCRR05] A. Sampaio, R. Chitchyan, A. Rashid, and P. Rayson. Ea-miner: a tool for automating aspect-oriented requirements identification. In *ASE '05: Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 352–355, New York, NY, USA, 2005. ACM.

[SS06] A. Stone and P. Sawyer. Identifying tacit knowledge-based requirements. *IEE Proceedings - Software*, 153(6):211–218, 2006.