An Exploratory Study of Information Retrieval Techniques in Domain Analysis

Vander Alves, Christa Schwanninger, Luciano Barbosa, Awais Rashid
Peter Sawyer, Paul Rayson, Christoph Pohl, Andreas Rummler

Fraunhofer IESE
vander.alves@iese.fraunhofer.de

Siemens AG
christa.schwanninger@siemens.com

Lancaster University
{marash,p.sawyer,paul}@comp.lancs.ac.uk

University of Utah
lbarbosa@cs.utah.edu

SAP Research
{christoph.pohl,andreas.rummler}@sap.com

Abstract

Domain analysis involves not only looking at standard requirements documents (e.g., use case specifications) but also at customer information packs, market analyses, etc. Looking across all these documents and deriving, in a practical and scalable way, a feature model that is comprised of coherent abstractions is a fundamental and non-trivial challenge. We conduct an exploratory study to investigate the suitability of Information Retrieval (IR) techniques for scalable identification of commonalities and variabilities in requirement specifications for software product lines. Accordingly, based on observations derived from industrial experience and on state-of-the-art research and practice, we also propose an initial framework, leveraging IR to systematically abstract requirements from existing specifications of a given domain into a feature model. We evaluate this framework, present a roadmap for its further extension, and formulate hypotheses to guide future work in exploring IR techniques for domain analysis.

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 identifying a set of reusable functionalities for the systems in the domain [5]. This set, in turn, is needed to formulate a domain model, such as a feature model [5].

A survey and analysis of Requirements Engineering (RE) techniques for SPL that we recently conducted [12] highlighted that, although Domain Analysis is supported in most RE techniques, most of the SPL approaches for RE assume that the documents used in the requirements elicitation are structured uniformly. However, in industrial SPL contexts, this assumption often does not hold: requirements documents are often textual and heterogeneous, having different levels of abstractions, different formats and structures. Further, domain analysis involves not only looking at standard requirements documents (e.g., use case specifications) but also at customer information packs, market analyses, etc. Looking across all these documents and, in a scalable way, deriving a feature model that comprises of coherent abstractions is a fundamental and non-trivial challenge.

Another important point identified by our survey and analysis [12] is the lack of appropriate tool support for domain analysis tasks in such approaches, especially given the nature of requirements, as reported by industrial partners from the AMPLE project[1] [1]. 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, making a manual analysis of these documents time consuming or even prohibitive.

In this context, this paper addresses the following research questions:

- How to perform scalable domain analysis?
  - What are the major challenges?
  - How existing tools and techniques can help?

[1]AMPLE is an EC funded project exploring the synergies of Aspect Oriented Software Development and Model-Driven Development in addressing key SPL issues.

– What extensions to these tools and techniques are necessary to perform this task?

Accordingly, we conduct an exploratory study to investigate the suitability of Information Retrieval (IR) techniques for scalable and systematic identification of commonalities and variabilities in requirement specifications for SPLs. Further, based on observations derived from industrial experience and on state-of-the-art research and practice, we also propose an initial framework, leveraging IR to systematically abstract requirements from existing specifications of a given domain into a feature model. The framework has been proposed in the context of industrial applications, where requirements documents are potentially unstructured and textual.

The remainder of this paper is structured as follows. First, Section 2 presents an analysis investigating challenges, potential tools, and techniques related to the research question. Next, based on this analysis, Section 3 proposes a framework leveraging those techniques and tools to address the identified challenges. Section 4 then evaluates the framework, leading to hypotheses and roadmap for its further refinement in Section 5. Related work is discussed in Section 6. Finally, Section 7 offers concluding remarks.

2 Background

Our research question requires an analysis of the underlying challenges as well as of existing techniques and tools that can potentially address them. First, Section 2.1 presents assumptions specifying the scope of the research question and contributing to understanding the major challenges; next, based on the nature of the identified challenges, Section 2.2 investigates techniques that potentially help to address them; Section 2.3 then reviews an existing approach offering limited support in addressing those challenges.

2.1 Assumptions

In this subsection, we present the assumptions underpinning the research questions.

A1) Feature model is the output of Domain Analysis. Although domain analysis potentially generates different models such as domain definition, domain lexicon, concept models, and feature models [5], in this work we consider only feature models as a means for describing the domain from an existing set of requirements.

A2) Documents are heterogeneous. The existing requirements documents have a different structure and level of abstraction. For example, this occurs in both industrial case studies we have encountered in the AMPLE project [1]: Siemens' Smart Home and SAP's Sales Scenario. The former is a SPL for houses with devices and sensors interacting with its inhabitants in order to achieve comfort, safety, and security goals; its requirements specification consists of four documents totalling 97 pages. The latter is a SPL in the Customer Relationship Management domain with a requirements specification consisting of four documents totalling 40 pages. In terms of structure, some documents already contain use cases, whereas other contain 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 manifest within one document, like in the requirement specification of the Smart Home case study.

A3) Configurations versus partially instantiated feature models. In the Smart Home and the Sales Scenario SPLs, we have observed that one requirement document can comprise not only requirements from one application, thus corresponding to a configuration, but also requirements from a set of applications, thus corresponding to a partially instantiated feature model.

A4) Feature is a high level abstraction of requirements. In the Smart Home and the Sales Scenario SPLs, we have observed 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 fine-grained entities that form coarse-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 [2, 4, 9, 14] 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 our work.

A5) 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.

2.2 Information Retrieval Techniques and Tools

Addressing the research question requires considering tool support. Given the following assumptions from Subsection 2.1, A2 (documents are heterogeneous) and A5 (there is similarity within different requirements documents), tools analyzing text should be considered as well as tools to extract latent similarity structure (e.g., similarity graphs/clusters, hierarchical relationship) from such data. Accordingly, we have seen the use of IR techniques partially addressing these issues but in other contexts, e.g., identification of similar requirements in Market-driven domain [13] and of similar requirements when tracing them to source of requirements [17]. Despite the fact that these techniques offer automation and, thus potential scalability, in comparing requirements, their use in the SPL context has not been explored. Accordingly, in this study we focus on two of these techniques, Latent Semantic Analysis and Vector Space Model, since they are suitable for dealing with textual information and are widely used in practice.

Latent Semantic Analysis (LSA) [6] is a technique for uncovering common patterns of usage across a large number of documents. The idea behind LSA is reduction of dimensionality: it tries to approximate a data set by reducing its dimensionality to k fewer degrees of freedom. By doing so, it merges the most meaningful patterns, i.e., words with similar occurrence patterns are projected onto the same dimension (possible synonyms), and removes some possible noise in the data. The value of k must be chosen carefully since a too small value can result in loss of relevant patterns and a too high one in non discovery of relevant patterns.

Vector Space Model (VSM) [15] is a model that represents documents as a vector in which each element (dimension) is a word. The weight of a word can be computed in many different ways. In this paper, the weight is represented by the frequency of the word in the document. The similarity between vectors is calculated using the cosine of the angle between the vectors. In comparison to LSA, its disadvantage is that words with similar meaning (e.g., car and truck) are not merged, but it has the benefit that it does not require any kind of parameter tuning.

2.3 Clustering & Merging

Chen et al proposed a semi-automatic approach to constructing feature models based on hierarchical requirements clustering [4]. It works as follows. Initially, for each requirements document, a requirements relationship graph is created by manually assigning values of requirement similarity based on the concept of *resource* relationship. In the second step, for each such graph, an initial feature tree is constructed by assigning, at each level of the tree, requirements with similarity higher than a variable threshold. The hierarchy is then assembled by building a refinement relationship between a lower-level feature to a higher-level feature iff the requirements in the former is a subset of requirements in the latter. As a result, requirements are abstracted into configurations. Lastly, the configurations corresponding to the requirements documents are merged into a fully-fledged feature model.

This approach satisfies the following assumptions from Subsection 2.1: A1 (feature model is the output of Domain Analysis) due to the output of the merge step, A4 (feature is a high level abstraction of requirements) due to the clustering step, and A5 (there is similarity within different requirements documents), which is explored in the determination of the requirements relationship graph. However, it does not meet A2 (documents are heterogeneous), since it assumes the existence of resource relationship between requirements, and A3 (configurations versus partially instantiated feature models), which is explicitly excluded in the hypothesis of the approach. However, both A2 and A3 are relevant from industrial case studies we have observed [12]. Furthermore, the applicability of such approach is limited in practice, since it relies on a manual determination of similarity between requirements and its time complexity is O(n^3), which can be prohibitive in a large-scale context.

3 Initial Framework

Relying on the analysis from the previous section, an initial framework for identifying commonalities and variations in requirements is described in this section. Its rationale and overview are described in Section 3.1, followed by each of the method steps, in Sections 3.2-3.4.

3.1 Rationale and Overview

In order to comply with the following assumptions from Section 2.1, A1 (feature model is the output of Domain Analysis) and A4 (feature is a high level abstraction of requirements), the framework ultimately needs to provide a higher level view of requirements. Once this higher level view is obtained for each requirements document, the view is then combined in a merge step, which then explores the latent similarity within the documents, according to A5 (there is similarity within different requirements documents). The framework extends the approach mentioned in Section 2.3 and, given the research question, incorporates the use of IR techniques described in Section 2.2 to automate as much as possible the process of domain analysis.

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). Then, for each such document, the framework performs a *requirement similarity determination* step, whereby IR techniques are used to automatically determine a similarity relationship between requirements within these documents. Next, based on this relationship, clusters of requirements are identified and these are abstracted further into a configuration during the *clustering* step. Finally, the configurations corresponding to all requirement documents are merged into a fully-fledged feature model in the *merging* step. The following subsections detail each of these steps.

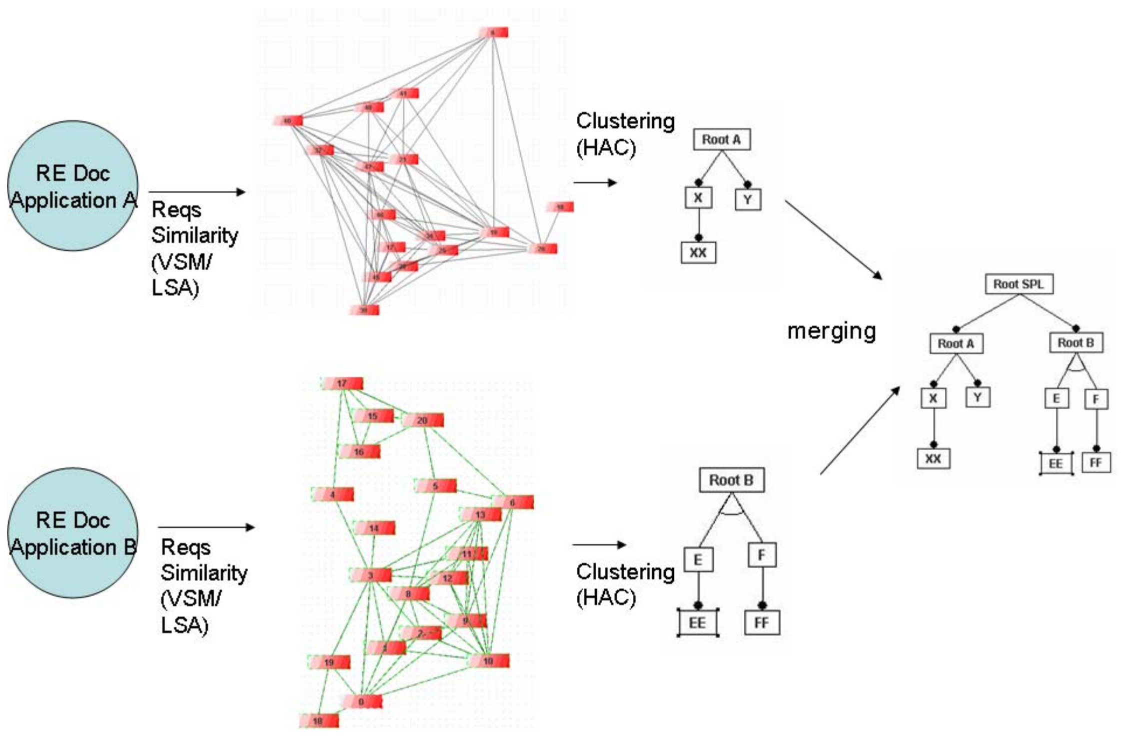


Figure 1. Framework for identifying commonalities and variations in requirements.

3.2 Requirement Similarity Determination

In the process of abstracting the requirements in order to obtain the SPL feature model, the goal of this step is to determine a weighted similarity relationship graph among the requirements of each requirements specification document. Given A2 (documents are heterogeneous) and the overall goal of scalable domain analysis, we employ automatic IR techniques, either VSM or LSA, the choice of which is subject to further evaluation (Section 4).

To illustrate the output these techniques, we show similarity values from requirements extracted from our dataset (see Section 4). For instance, requirements related to "notification" feature such as:

- "These notifications are presented to the user through the Subscriber Portal. The operator can see them in the operator UI".
- "Subscriber users and Operator users will be able to consult notifications through the subscriber portal and through operator portal, respectively. These portals will have to make pooling operations to keep updated with the eventual new notifications that arrive at the SEP. Only the administrator user has the capabilities to delete the notifications but all users associated to a subscription will see them. Users and Operator will also be able to associate the reception of a certain type of notifications with the triggering of a dynamic action".

They have a VSM similarity of 0.69 and a LSA similarity of 0.95. It is clear these requirements are in fact related and these techniques were able to give a high weight to the connection between them. Nevertheless, there are cases in which the IR technique is unable to identify any similarity. For instance, the requirement "get accounting tickets" does not share any word with other requirements in the same document; therefore, its similarity to all other requirements is 0 according to VSM.

3.3 Clustering

After requirement similarity determination for each document, and according to A4 (feature is a high level abstraction of requirements), this step further abstracts the requirements into configurations. The result of the previous step is a similarity matrix, which is a basis for establishing the initial relationship among requirements of each document. As we are interested in creating a tree comprising many levels of abstraction of the requirements, we employ the Hierarchical Agglomerative Clustering (HAC) [18] in our context

Algorithm 1 Clustering algorithm.

```
1: Input: $requirements$
2: $similarityMatrix = calculateSimilarity(requiremens)$
   {Calculate the similarities between requirements (initial clusters)}
3: repeat
4:     $closest = findTwoClosestClusters(similarityMatrix)$
       {Pick the two closest clusters}
5:     $newCluster = mergeClusters(closest)$
       {Merge the two closest clusters}
6:     $similarityMatrix = updateMatrix(newCluster)$
       {Update the similarity matrix after the merging of two clusters}
7: until number of clusters equals to 1
8: return $newCluster$
```

since it is widely used in practice and simple to implement.

Algorithm 1 presents how the HAC works in our context. Initially, the similarity matrix is created according to the IR technique used (LSA or VSM) (line 2). At the beginning, each requirement represents a single cluster. The next step is to determined the two closest clusters: $c1$ and $c2$ (line 4). They are then merged, representing a higher level of abstraction (line 5). Next, the similarity matrix has to reflect this merge, i.e., to remove the entries in the matrix related to $c1$ and $c2$ and create a new entry with respect to the cluster represented by the merged clusters. The merge process continues until only a single cluster remains. The output of the algorithm is then a configuration.

3.4 Merging

The previous steps are applied for each requirements specification document, thereby resulting in a series of configurations. Each of these represents a high level view of an application. In order to have a unified view of the domain of the SPL, these models are then merged into a fully-fledged feature model.

The approach used is similar to previous work [4], whereby configurations, previously clustered from similar requirements, are incrementally merged using a depth-first algorithm comparing individual nodes of such configurations. Our framework additionally includes the following: 1) **semantics of nodes**. When comparing the individual nodes, we take into account their semantic information and not only feature names, thus making the comparison more flexible and resilient to underlying synonymy; 2) **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 denoting these concepts (e.g., "optionally", "alternatively", "at least one of").

4 Evaluation of the Framework

Our main goal in the experimental evaluation is to verify how the techniques mentioned previously work in the context of requirement documents. More specifically, we want to observe how effective these techniques are in identifying similar requirements and clustering them in groups.

4.1 Experimental Setup

Data set. The framework has been initially evaluated in the Smart Home industrial SPL (Section 2.1). It was 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.

From the Smart Home case study, we considered two requirements specification documents: "Requirements from Building Architect Point of View", which contains 28 requirements, and "Requirements Totally Integrated Home", with 59 requirements, hereafter referred to as **APV** and **TIH**, respectively. We broke each document into requirements and numbered them in the order that they appear in the document. These requirements documents had been written by stakeholders playing different roles: APV by the house architect and TIH by a requirement analyst. Some of the requirements are represented with long sentences, whereas others only with few words, e.g., "Subscriber info". According to a domain expert, these document satisfy the assumptions in Section 2.1.

Configurations. There are two main points we need to define to perform this experiment: 1) the similarity measure between requirements: for this purpose, LSA and VSM were used because they are widely used in document-based mining; 2) the clustering algorithm: there are a great variety of clustering algorithms; we opted to use HAC because it is deterministic, as opposite to K-means [18], widely used in practice and simple to implement. At the end, for each document we ran two different configurations: 1) HAC with LSA; 2) HAC with VSM.

Evaluation Metric. For evaluation purposes, first an expert from the Smart Home domain identified the expected groups of requirements in each document; we then measured the performance of each similarity metric against them. To measure the quality of the clusters derived by clustering algorithm, we use a standard measure: **entropy** [8]. For each cluster c_j, we compute the probability p_{ij} of a member of c_j belonging to class i. Using this class distribution, the entropy of each cluster is calculated using the standard formula:

	HAC+VSM	HAC+LSA	Baseline
APV	0.7	0.89	1.05
TIH	0.83	1.05	1.25

Table 1. Entropy values for the different configurations over two requirement documents: APV and TIH

$$Entropy_j = -\sum_i p_{ij} log(p_{ij}) \qquad (1)$$

The total entropy for the set of all clusters is the sum of the entropies of each cluster, weighted by the size of each cluster. Intuitively, the better the clustering solution, the more homogeneous the clusters, and consequently, the lower the entropy. An entropy of 0 means a perfect clustering output.

4.2 Results and Discussion

The first issue that we want to verify is whether these IR techniques are in fact able to capture similarity patterns between requirements generating meaningful clusters. For this, in addition to the HAC+LSA and HAC+VSM, we executed a random approach as a baseline. The results are presented in Table 1.

The numbers show that both proposed approaches obtained a better result than the baseline (0.7 to VSM and 0.89 to LSA versus 1.05 to baseline). As a result, these techniques are in some way able to identify relevant clusters in the data. As an example, consider these 3 requirements, depicted from the TIH, belonging to the "notification" cluster: "The reception of notifications (Push interface)", "The retrieval of the notifications (Pull interface)", "The removal of notifications". As one can see, the word "notifications" is shared for all requirements. As a result, both configurations (HAC+VSM and HAC+LSA) were able to capture this and place these requirements in the same final cluster.

Another interesting result was that VSM outperformed LSA in both scenarios. A possible explanation for this is that LSA obtains better performance over a large corpus. As in many cases requirements are represented for single or even very short sentences, they provide little information to reliably infer similarity between requirements. More concretely, the dimensionality reduction (Section 2.2) can result in loss of significant information for the similarity measure. Although the performance of VSM can also be damaged by this lack of information, VSM, in contrast to LSA, uses all available information.

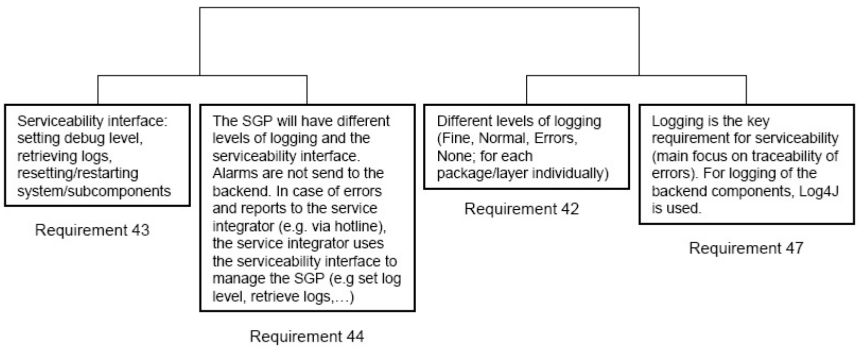
To illustrate in more details the output generated by the clustering algorithm, Figure 2 depicts part of the tree generated by HAC+VSM having the requirements of TIH as input and Figure 3 for APV. For the TIH sub-tree, the algorithm correctly clusters similar requirements related to logging functionality: requirements 42,43,44, and 47. However, the APV sub-tree shows a slightly different result. Although requirement 2 is more related to requirements 1 and 13 (devices and their integration), and requirement 19 is more related to requirement 12 (detection/notification danger), requirement 2 was first merged with requirement 19 (see Figure 3). The reason for this is the fact that requirements 2 and 19 share more common words than requirements 12 and 19 do. In a second step, cluster 2-19 is merged to requirement 12 and finally, in a third step, requirements in cluster 2-12-19 are merged with cluster 1-13. A possible solution for this is taking into account the proximity of requirements in the document. More specifically, similar requirements tend to be physically close in the requirement document. The intuition behind this is that, as requirement documents are written by humans in natural language, these latter tend to put together related requirements in order to improve the document flow. Therefore, for the previous example, since requirements 1 and 2 are close, there might exist some way of increasing their content similarity and correctly clustering them.

At some level, the clustering algorithms were able to identify this kind of patterns since these requirements share some words. This can be observed in Figure 4, which illustrates the result of clustering in TIH[2]. Requirements as 51,52,53,54,55,56,58 were grouped in a same cluster and in fact they belonged to the same group in the expected groups defined by the expert as well as 42,43,44, and 47 presented in Figure 2.

An entropy of 0 means a perfect clustering. Our framework obtained higher values than this (0.7 for APV and 0.83 for TIH), but lower entropy than a random approach (1.05 for APV and 1.25 for TIH). This means that it did not obtain a perfect output, but at least it can help users in the process of creating application configurations. To improve its performance, we suggest some extensions as for instance looking at the document structure, as further discussed in Section 5.1. When VSM is used in the *requirement similarity determination* step of the framework, we are implicitly assuming that similar requirements share similar vocabulary; therefore, the framework will not perform well when this is not the case.

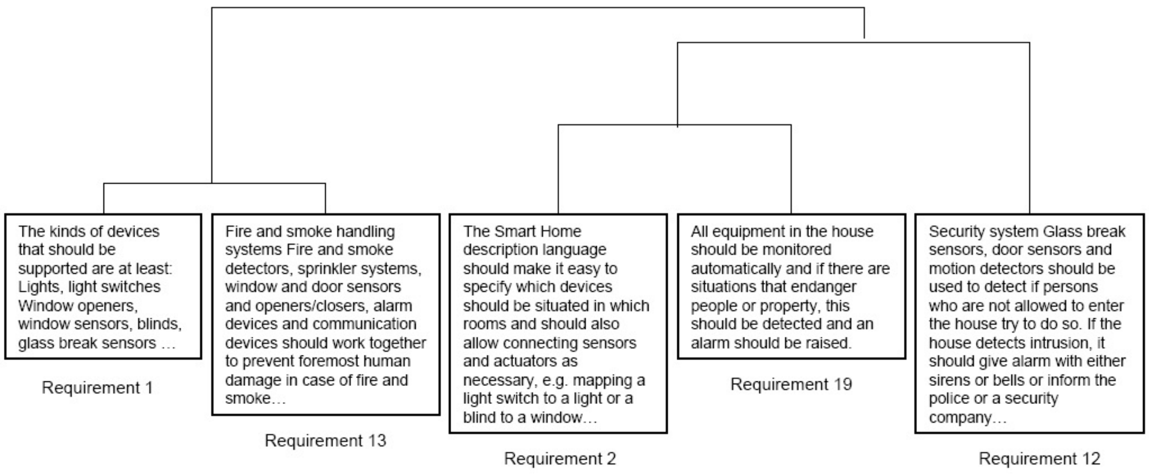
Additionally, we note that due to the agglomerative nature of the clustering algorithm, features closer to the root comprise an increasingly high number of requirements. Therefore, naming and finding a semantic definition of features that summarizes all its encompassing requirements should be addressed with a scalable approach. Although

[2]In Figure 4, each feature comprises a cluster of requirements from that document, which is indicated by each feature containing a set of requirement numbers from the document (requirement numbers are separated by a "-").



Serviceability interface: setting debug level, retrieving logs, resetting/restarting system/subcomponents
Requirement 43
The SGP will have different levels of logging and the serviceability interface. Alarms are not send to the backend. In case of errors and reports to the service integrator (e.g. via hotline), the service integrator uses the serviceability interface to manage the SGP (e.g set log level, retrieve logs,…)
Requirement 44
Different levels of logging (Fine, Normal, Errors, None; for each package/layer individually)
Requirement 42
Logging is the key requirement for serviceability (main focus on traceability of errors). For logging of the backend components, Log4J is used.
Requirement 47

Figure 2. Example of tree generated by the HAC+VSM for the TIH document.



The kinds of devices that should be supported are at least: Lights, light switches Window openers, window sensors, blinds, glass break sensors …
Requirement 1
Fire and smoke handling systems Fire and smoke detectors, sprinkler systems, window and door sensors and openers/closers, alarm devices and communication devices should work together to prevent foremost human damage in case of fire and smoke…
Requirement 13
The Smart Home description language should make it easy to specify which devices should be situated in which rooms and should also allow connecting sensors and actuators as necessary, e.g. mapping a light switch to a light or a blind to a window…
Requirement 2
All equipment in the house should be monitored automatically and if there are situations that endanger people or property, this should be detected and an alarm should be raised.
Requirement 19
Security system Glass break sensors, door sensors and motion detectors should be used to detect if persons who are not allowed to enter the house try to do so. If the house detects intrusion, it should give alarm with either sirens or bells or inform the police or a security company…
Requirement 12

Figure 3. Example of tree generated by the HAC+VSM for the APV document.

our framework currently does not address it, we suggest extensions to the framework in order to address this issue (Section 5). Because naming and determining a unique semantic representation of the features was not possible, the *merging* step could not be performed.

Another observation regarding the *clustering* step of the framework is that it currently does not generate a partially instantiated feature model, which could happen according to A3 (configurations versus partially instantiated feature models). Nevertheless, in Section 5, we also suggest extensions to the framework to address this issue.

5 Hypotheses and Roadmap for Further Research

The analysis conducted in Section 2 and the evaluation of our initial framework in Section 4 unveil a number of findings that contribute to defining hypotheses to guide further research. In this section, we present such hypotheses and outline a corresponding roadmap for assessing them.

Hypothesis 1: Textual requirements documents have latent structure that complements VSM/LSA. VSM and LSA determine relationship among the requirements, but assume a flat structure, i.e., also known as *bag of words*. Although the requirement documents considered in the evaluation of the framework are textual, they still have latent structure, such as hierarchy and variability relationship between requirements, and proximity structure, i.e., similar requirements are physically closer in the document, as observed in Section 4. Such latent structure could be used to improve requirement similarity determination and, complementarily, latent variability detection, according to A3 (configurations versus partially instantiated feature models), from Section2.1. Accordingly, we could enhance the

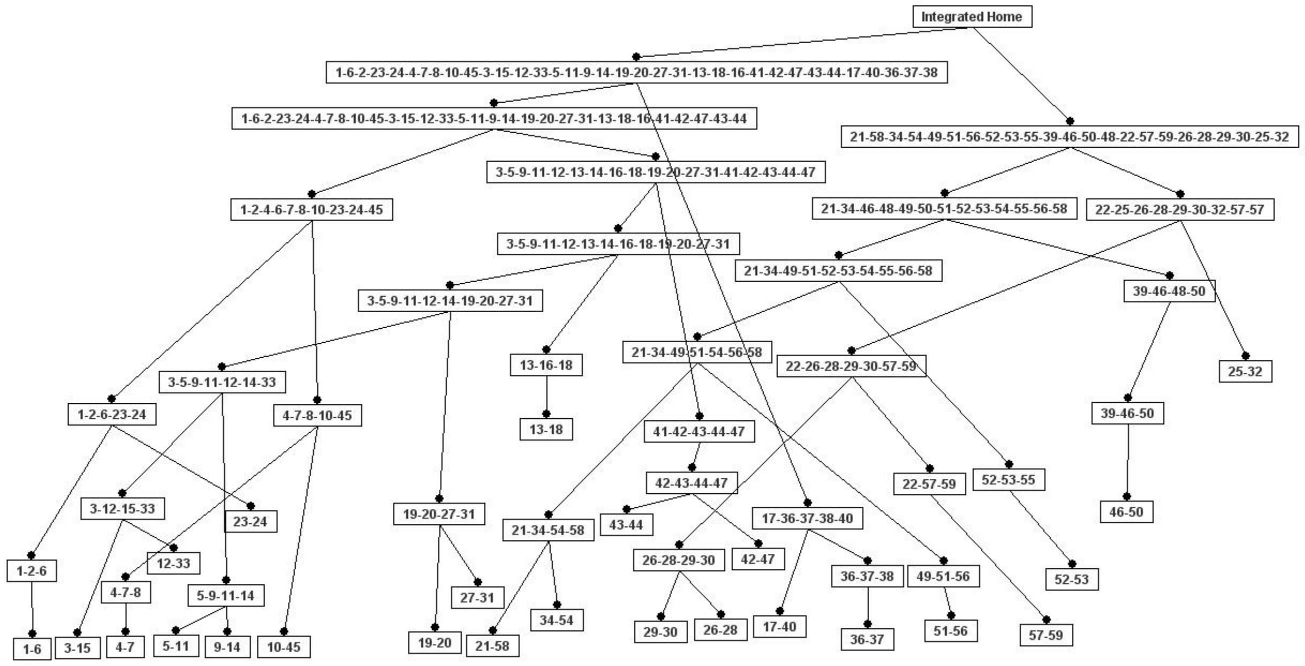


Figure 4. Clustering Requirements from the TIH document.

framework with a phase to capture such latent structure, which we further motivate and outline in Section 5.1.

Hypothesis 2: VSM scales over LSA in Domain Analysis. Regarding the flexibility intended by LSA, this is important, since according to our assumptions (Section 2.1), documents input to the framework are heterogenous, thus suggesting the inherent noise assumed by LSA. However, the evaluation so far, with better results for VSM, suggest that, since the size of individual requirements is not big enough, VSM should be the preferable comparison technique.

Correspondingly, we plan to conduct additional experiments with larger set of documents and, within each document, larger requirements in order to verify whether there will be a point where requirements will be large enough for LSA to outperform VSM.

Hypothesis 3: feature naming should be scalable. As the HAC algorithm proceeds higher in the feature hierarchy, features comprise an increasingly high number of requirements, as shown in Figure 4. Therefore, naming them is a non-trivial task.

In order to address this, our initial suggestion is to consider the following data of the requirements comprising a feature: 1) semantic information of its words, e.g., words referring to raising an alarm; 2) part-of-speech information, e.g., nouns. The relevance of such data could be weighted by the frequency of occurrence of words and the structure in the requirements composing the feature, such as whether it is a heading or top-level requirement. Given the high number of requirements and features, such extraction should be performed with tool support, which is possible using Wmatrix [16], a Natural Language Processing tool that determines semantic, part-of-speech, and frequency information of words in text.

Hypothesis 4: compare documents at same level of abstraction. Regarding A5 (there is similarity within different requirements documents), such overlapping may not be apparent at first, due to different formats and abstraction levels of the different documents like APV and TIH (Section 4). Therefore, finding the similarity by taking information inside the documents and from the domain into account is the specific challenge here.

Aiming at more flexible comparison of documents, we envision employing ontology-based tools, such as OntoLancs [7], which take into account domain information and does not simply perform a string matching of feature names; in addition, we plan to combine this with the Wmatrix, as described in Hypothesis 3. Lastly, in order to further evaluate the framework, we will also use more structured documents, such as SAP's Sales Scenario [1].

5.1 Mining Structure

According to Hypothesis 1, and A2, and A3, requirements may already have some latent structure. Correspondingly, we suggest refining the framework to encompass a new step, *mining structure*, which should come before the *requirement similarity* step, and whose purpose is to capture such latent structure, thus improving requirement similarity determination and, complementarily, latent variability detection.

Given the structural information that similar requirements tend to be physically close in the requirements document (Section 4.2), this could be used as an additional criteria in determining requirement similarity relationship. For example, in Figure 3, since requirements 1 and 2 are physically close, this could help to increase the initial similarity computed by VSM, which in fact would increase precision. A similar consideration applies for LSA.

Regarding latent variability, in the following fragment, for example, extracted from the requirement specification document of the Smart Home SPL, we see a requirement 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 browse (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).

Variations:
Reconnect: The end user disconnects the gateway and reconnects later in time (also case of power failure).

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

This latent variability structure could be leveraged in accordance with A3 to help to determine partially instantiated feature models from a requirements document, which had not been performed by the clustering algorithm from the framework (Section 3.3). Such variability structure should then be captured and used at the end of the *clustering* step: in the *requirement similarity* and the *clustering* steps, such requirements would be already abstracted by a single requirement identified in the *mining structure* step, e.g., the requirement name.

In order to identify variability already present in these documents, we could look for words denoting variability. This can be accomplished in a scalable way by using the capabilities of Wmatrix. For example, for the previous fragment, 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... Moving: The end user moves...

We then look at the context of such words (e.g., variations, optionally, alternatively) and identify the enclosing requirement and variants. In the example above, the context comprises the requirement description as well as two variants, i.e., *reconnect* and *moving*. These are then stored and considered as a single requirement for the following step, which then will not interfere with this pre-existing relationship. The notion of context is flexible and can encompass not only phrases delimited by punctuation marks, but also a complete paragraph or subsection. The MyTagWizard feature in Wmatrix supports the creation of a user defined dictionary for words denoting variability.

In addition to variability, pre-existing hierarchical relationships between requirements could also be captured during the *mining structure*. We could apply some heuristics as proposed by John et al [10], 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 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.

2.1.2 Web client uses networked service: After login to the subscriber portal, the portal asks the RAM to generate service session IDs.

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.

6 Related Work

Chen et al. [4] also propose a framework which, based on a clustering algorithm, generates features models. Similar to our work, the output of their clustering step is a configuration for each requirements document. Nevertheless, in terms of requirement similarity determination, they use the concept of *resource* to define similarity: requirements are similar whenever they share resources; however, 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. We instead employ automatic IR techniques (LSA and VSM), which rely on a sound linear algebra foundation. We further identify the need for scalable and systematic naming of features in configurations and propose some heuristics. Another major difference from their approach is that we use a O(n^2) time complexity clustering algorithm (HAC [18]), instead of their O(n^3) algorithm, which compromises the applicability of the latter.

LSA has been used to determine requirement similarity and to provide a clustering mechanism for requirements [11] and relating requirements to sources of requirements [17]. VSM has also been used to compare requirements, but in the context of Market-driven domain with the purpose of identifying requirements that might look different at first, but which are ultimately very similar and thus should not be considered duplicates and not be addressed to avoid development rework [13].

CaVE (Commonality and Variability Extraction) [10] applies extraction patterns to transform documents to requirements and features. The process is divided into preparation, selection, and validation. This work shares similar goals, but the methodology does not explore the use IR techniques. Bragança et al [3] present an automatic model-driven development approach for building a domain model from use cases. In contrast, our approach currently focuses on textual requirements.

7 Conclusion

We have conducted an exploratory study on leveraging IR techniques for achieving scalable domain analysis. Accordingly, based on an industrial context, we have analyzed the underlying challenges, automatic IR techniques, and existing approaches. We also presented a framework describing how developers can systematically identify commonalities and variabilities in SPL requirements specifications. It relies on a novel application of IR techniques and tools that together help to abstract individual requirements from existing requirements specifications of a given domain into 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 is compared to state-of-the-art approaches and evaluated in an industrial context. Moreover, the evaluation has unveiled findings that allowed us to identify new hypotheses to guide further research, and to provide an enhancement roadmap for the framework.

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, which is essential for addressing the research questions; additionally, such results suggest that the framework accuracy can be improved by better integration of the *mining structure* and *requirement similarity determination* steps, improvement of the *mining structure* step to detect different kinds of structure, such as hierarchical relationship among the requirements, and refinement of strategy for naming features in clustering.

Acknowledgements

This research has been sponsored in the context of the AMPLE project. The authors also thank the anonymous reviewers for valuable feedback.

References

[1] *Aspect-Oriented, Model-Driven Product Line Engineering (AMPLE) Project.* `http://www.ample-project.net/`. Last access, May. 2008.

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

[3] A. Bragança and R. J. Machado. Automating mappings between use case diagrams and feature models for software product lines. In *Proc. of the 11th International Software Product Line Conference (SPLC'07)*, 2007.

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

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

[6] 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.

[7] R. Gacitua, P. Sawyer, and P. Rayson. A flexible framework to experiment with ontology learning techniques. In *Proc. of the Twenty-seventh SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 153–166, 2007.

[8] R. M. Gray. *Entropy and Information Theory*. Springer-Verlag, 1991.

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

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

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

[12] 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.

[13] J. N. och Dag, T. Thelin, and B. Regnell. An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development. *EMSE - Empirical Software Engineering*, 11(2):303–329, 2006.

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

[15] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1976.

[16] P. Sawyer, P. Rayson, and K. Cosh. Shallow knowledge as an aid to deep understanding in early phase requirements engineering. *IEEE Trans. Software Eng.*, 31(11):969–981, 2005.

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

[18] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.