Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 339 (2018) 5–20

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Framework for Managing Requirements of Software Product Lines

Maximiliano Arias, Agustina Buccella and Alejandra Cechich*

GIISCo Research Group
Faculty of Informatics, UNComa University
Nuequen, Argentina

Abstract

An emerging problem in the Software Product Line Engineering (SPLE) is the need for integral management of planned reuse. In SPLE there are two instances where managing requirements gains relevance. The first one arises during the construction of SPLs based on legacy software or previously developed SPLs. The second one appears when instantiating products from the SPL platform, where instantiating variability meets the custom requirements of each product. The objective of this paper is to define a framework that allows management of requirements using Natural Language Processing and Information Retrieval techniques, to structure, clean, index and find reusable functionalities according to those requirements. This framework is built in a way that allows the combination of such techniques to evaluate the best combinations for finding the correct functionalities in each SPL domain.

Keywords: Software Product Lines, Query Expansion, Requirements Engineering, Software Reuse.

1 Introduction

Reuse based paradigms are centered around the idea of controlled reuse, occasionally inside a single application domain [1].This requires a great initial effort, performing domain analysis and gathering domain requirements that will serve as building blocks of reusable artifacts. Here the importance of structuring and organizing such requirements and their derived artifacts. Structured and organized information can be explored with less effort, in order to find reusable components when new requirements appear.

When developing software, Requirements Engineering (RE) is the computer science field in charge of defining, documenting and maintaining software requirements [2], in most cases, described in natural language [3]. This fact motivated several proposals [4, 5, 6] to use Natural Language Processing (NLP) to reduce ambiguity, detect missing information, and even improve traceability with later stages of the development process.

* Email: {maximiliano.arias,agustina.buccella,alejandra.cechich}@fi.uncoma.edu.ar

https://doi.org/10.1016/j.entcs.2018.06.002

In previous works [7, 8], we have studied several aspects of SPL development, focusing on the geographic domain. In the context of this paradigm, we defined a development methodology centered around software artifacts, which guide the entire process. These artifacts, based on correctly identifying software requirements, are used to favor the effective reuse of the SPL and to manage its products throughout the whole development. Particularly, in SPLE it is necessary to identify previously developed features, based on specific requirements. This task aims towards simplifying the software engineers work in terms of effective reuse. Failing to identify already existing features could generate duplicated ones, which evolve in different directions, implementing the same requirements several times and even in different ways.

Taking into account the previously mentioned context, our proposal looks to define a framework for analyzing and managing natural language requirements in order to simplify retrieval of existing features, simplifying future reuse. The framework uses Natural Language Processing (NLP) and Information Retrieval (IR) techniques, providing the means for choosing between and evaluating different variants of such techniques. Thus, a modular approach of interchangeable techniques helps reduce the effort taken in trying new strategies. This way, the main contributions of this paper can be grouped into two main aspects, 1) to offer the means for structuring natural language requirements providing an effective method for retrieving the features that implement those requirements, and 2) to provide a suitable environment for interchanging NLP and IR techniques in order to evaluate different combination's behavior for a given domain. At the same time, the modular flexibility allows the framework to be extended in order to increase the set of available techniques and evaluate new combinations.

This paper is organized in the following way. First, Section 2 presents an analysis of existing literature, related to our proposal. Section 3 shows the motivation and previous works that inspired the present proposal. Following, Section 4 presents the *framework for managing requirements of software product lines*. Section 5 presents experimental evaluation and result analysis. Finally, Section 6 shows conclusions derived from this work.

2 Related works

Requirements management in reuse oriented paradigms is supported by NLP in numerous tasks. There is an important set of well known techniques, used throughout the entire computer science field for these purposes, from which we can name Part Of Speech tagging [9] (POS), Name entity Recognition (NER) [10] and semantic expansion [11]. Particularly, in the semantic expansion field many approaches use custom techniques based on lexical databases such as WordNet [12], domain taxonomies and ontologies or another type of tool capable of increasing the semantic value of a given word. In the current literature we can find several proposals that try to combine different RE tasks with NLP tools in order to simplify and/or automatize them. Some papers aim towards automatic or assisted generation of models [13, 14, 15], some others try to extract requirements from free text documents [16, 17, 18], in other cases authors look to improve and extend requirements documents [19, 20] and some even try to use NLP to manage reusable software artifacts and requirements documents [21, 22, 23].

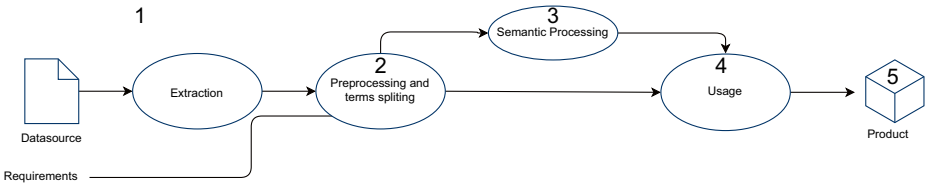

Figure 1. Requirements management processes across a0ll proposals.

In general, despite using different names, the bast majority of the studied proposals follow a fairly similar process, as seen in Figure 1. First, requirements (inputs) are presented in a given format, sometimes extracted from a particular datasource (1.a) and others from a requirements specification (1.b). Obtained requirements are preprocessed (2), going trough different processes of tagging and relevant information extraction. Later, some proposals [14, 22, 19, 23, 15, 20] perform some type of semantic expansion (3) before using the extracted information. There is an ample variety of usages (4) for preprocessed and expanded requirements depending on the RE process focused by the proposal. In general, for each type of usage a product is finally created (5). These products vary from every proposal including auto-generated models [13, 14, 15], requirements specifications [16, 18, 20], candidate software artifacts [21, 22], etc.

Particularly, in SPLE, some proposals [18, 17, 24] present a process that is similar to the one showed in Figure 1, focusing on specific development activities. In these cases, each requirements source vary from proposal to proposal, using as input software requirements in requirements specifications or extracted, for example, from public software project descriptions and even user opinions from download sites. Following, these requirements are pre-analized and semantically pre-processed with the objective of finding common features between them. Found features are later analyzed to detect which one of them can be represented as a feature model. This way, the authors propose a two-way street between software requirements and feature models, reducing the effort of identifying which requirements are implemented by which functionalities.

Table 1 shows the classification of analyzed proposals, comparing the NLP and/or IR techniques used to achieve the desired objective. We analyzed a subset of all the existing techniques, which includes Part of Speech Tagging (POS tagging), tokenization and semantical expansion. Particularly for the last one, we defined an subdivision taking into account those that use lexical databases and those that use taxonomies or ontologies. We finally included another comparison category called *Other* to identify the proposals that perform semantical expansion using other methods.

In order to clarify our analysis, Table 1 also includes our proposal, where we can observe the use of POS tagging, NER, tokenization and semantical expansion trough several methods in order to retrieve reusable software artifacts from software requirements. In contrast with other proposals, we defined an integral solution, using all presented techniques with different purposes inside the framework, as shown in Figure 1. Thus, the current work presents a solution for all the previously mentioned steps, excepting the requirements extraction process (1.a). In our context, requirements are obtained using traditional elicitation with domain experts.

Table 1
Comparison between used NLP and IR techniques

Technique/ Proposal	Arora etal. [16]	Bhowmik etal. [13]	Boutkova etal. [17]	Capobianco etal. [21]	Ferrari etal. [18]	Gulia etal. [14]	Hahm et al. [22]	Körner etal. [19]	Ma etal. [23]	Mu etal. [15]	Wang etal. [20]	Bakar eta.[24]	Our Proposal
POS tagging	●	●	●	●	●	●		●		●	●	●	●
NER	●										●		●
Tokenization	●		●	●		●			●	●		●	●
Semantical Expansion													
Lexical databases		●				●	●	●	●	●	●	●	
Taxonomies/Ontologies					●	●	●	●				●	
Other						●							

3 Motivation and previous works

In previous works we have defined a methodology for developing SPL [7, 8] that uses different software artifacts to specify and implement common and variant functionalities for a particular domain. By applying this methodology, we detected the need of analyzing the way new requirements must be implemented, managed and found within developed components. In this way, we identified two development stages in which the requirement management is vital. These stages are presented in Figure 2 – *SPL Construction*, and *SPL Usage*.

On one hand, the first stage is involved in the *Domain Engineering* phase (on top of the Figure, marked as 1), in which based on user expert's requirements, domain standards, and variability information (obtained from the two previous inputs), a set of software artifacts is created to be part of the SPL platform. As we can observe in the Figure, these artifacts are the domain service taxonomy, containing common and variant services of a particular domain; and the functional datasheets, containing the variability models for each functionality. These datasheets are created by following the rules imposed by a reference architecture and by selecting services of the aforementioned domain taxonomy. Then, these datasheets must be implemented as software components.

Following, the *Application Engineering* phase represents the second stage in which requirement management is also vital, involving instantiations of the SPL developed previously (at the bottom of Figure 2).

In this phase the instantiation process includes the selection of the particular variability that must be part of the product to be generated. In this way, the specific requirements

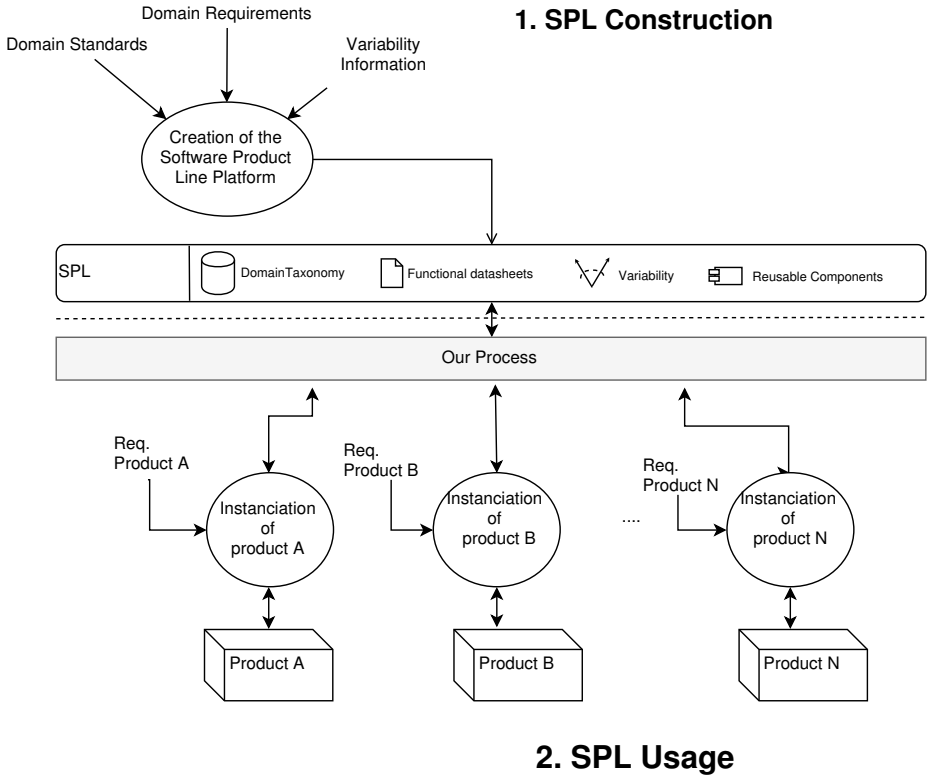

Figure 2. Process for developing and using a SPL.

of the expert users are essential for determining the final functionality of each product. The software platform, with its artifacts previously developed in the domain engineering phase, is used to configure and select the set of variant services that will be part of the final product. At the same time, as new requirements can be added, the SPL must be analyzed to find functionalities to be implemented as well as variability that must be configurated.

Our requirement analysis is focused on the functional datasheets due to they describe complete functionalities with common and variable services (of the taxonomy) interacting to each other. These interactions of services are implemented by applying the orthogonal variability model, proposed by [25], and considering optional, alternative, and mandatory variability as well as require and exclusion constraints.

In this way, the instantiation of a new product requires analyzing the SPL platform by finding functionalities that implement the new requirements proposed by expert users. This retrieval step of reusable requirements must provide the best possible results by reducing the efforts for finding them and avoiding the possibility of duplicate functionalities. The functionalities of the SPL platform have been created by the domain analysis and are described, in general, in natural language. In this way, the advances of NLP techniques and information retrieval can be really useful during the process of instantiating products.

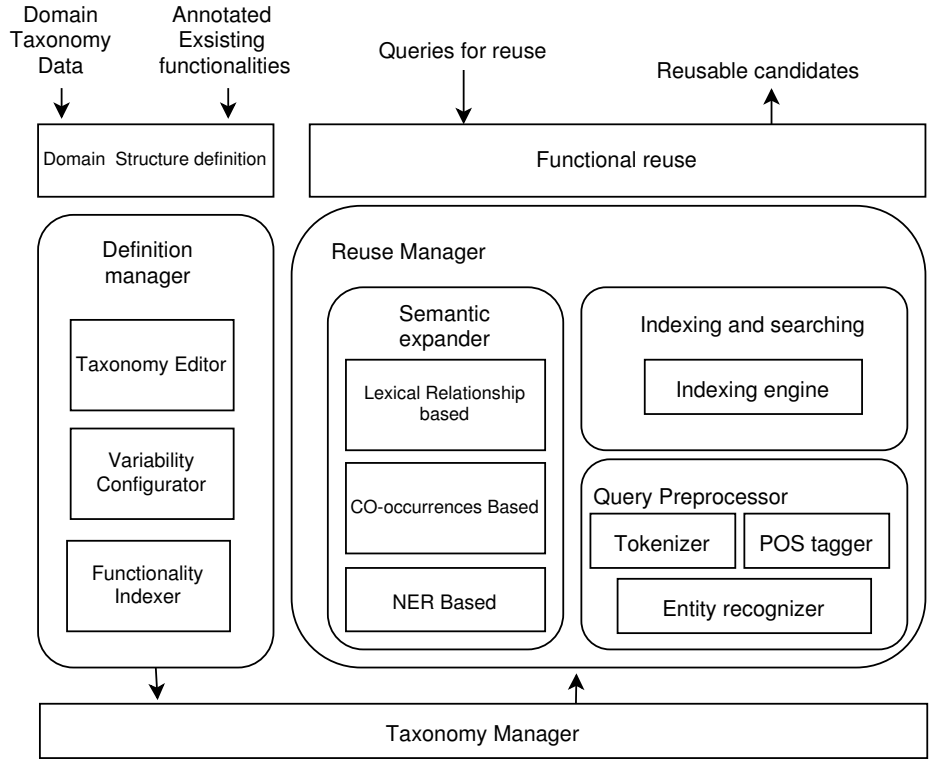

Figure 3. Working Framework's Architecture

4 A working framework for managing requirements in SPLs

Our proposal is focused on defining a working framework that allows structuring reusable artifacts of a software product line and retrieving them when new requirements arise. As Figure 1 shows, the proposal is composed of five steps, starting from elicited requirements to be pre-processed and divided into elemental pieces. Then, pre-processed requirements are semantically expanded and used to retrieve software artifacts (step 4). Then, the final product is a list of reusable software artifacts.

The working framework has two main goals as follows. Firstly, providing a supporting tool that allows developers to create a structure as a way of describing a particular domain, and define which reusable functionalities can be used; and secondly, allowing developers to explore the structure and find reusable functionalities from new software requirements.

The working framework's architecture is shown in Figure 3, which is composed of two main elements – one in charge of defining the domain structure (left side of the Figure), and another in charge of exploiting the structure to satisfy the needs of reuse (right side).

The *Definition Manager* (left side) creates the domain structure and its functionalities (left side) by using three components: the *Taxonomy Editor*, the *Variability Configurator*, and the *Functionality Editor* (from previous work [26]). To accomplish these goals, our proposal is based on a previously defined methodology [8] that relies on the existence of domain taxonomy. We model functionalities as services, where each service is unique and indivisible, and each functionality is an structure that defines its relationships. In this

way, the functionalities are designed through a common language – the taxonomy. This fact facilitates indexing and searching software for reuse. The variability configuration is defined during the construction of the functionalities by identifying optional, alternative, and mandatory variability.

Finally, the *Reuse Manager* uses the structure to recommend existing reusable functionalities (right side of the Figure). This is a core component of the framework, since software product lines are focused on reusing previously defined functionalities (planned reuse [27])). In the lowest part of Figure 3, we can see the *Taxonomy Manager* horizontally traversing the other two managers. It handles reading/writing operations on the taxonomy. During the domain structure definition, the Taxonomy Manager is used in writing mode when the structure is being created; and in reading mode when functionalities are being generated. As another example, the Taxonomy is used in reading mode when searching candidates for reuse, becoming an intermediary between requirements and complex functionalities [28]. In this work, we focus on the Reuse Manager detailing its structure as well as its internal behavior.

4.1 Reuse Manager

The Reuse Manager is composed of three modules: the Query Preprocessor, Query Expander [29], and the Indexer. The first module, Query Preprocessor, is composed in turn of three components: Tokenizer, in charge of separating terms; POS Tagging, which detects different word types; and Entity Recognizer, which identifies named entities. The second module, Query Expander, is in charge of semantically expanding, where each of its components represents a different expansion strategy. Currently, the module is implemented with three strategies that are respectively based on the use of lexical relationships, the use of a database of word co-occurrence, and the recognition of named entities. Finally, the Indexer takes the products generated by the previous modules and uses them to explore the structure defined by the Definition Manager and using the Taxonomy Manager.

Figure 4 shows the Reuse Manager's workflow. The three columns of the Figure represent the three modules that compose the Manager; and each hexagon represents a transforming process that is performed by internal components. We can see that, from left to right, these columns correspond to the steps (as numbered circles) of the general process identified in Figure 1 of the Related Works Section. Following, we detail each of the steps of Figure 4. To facilitate explanations, we will use the following requirement:

"When used by Calin Rovinescu, the president, inside the Air Canda offices in Montreal , the system must allow him to log in with full privileges"

1) Query Preprocessor

This module is implemented through three steps, numbered from 1-3 in Figure 4. They are carried out by the *POS Tagging*, *Named Entity Recongnizer* (NER), and *Tokenizer* respectively.

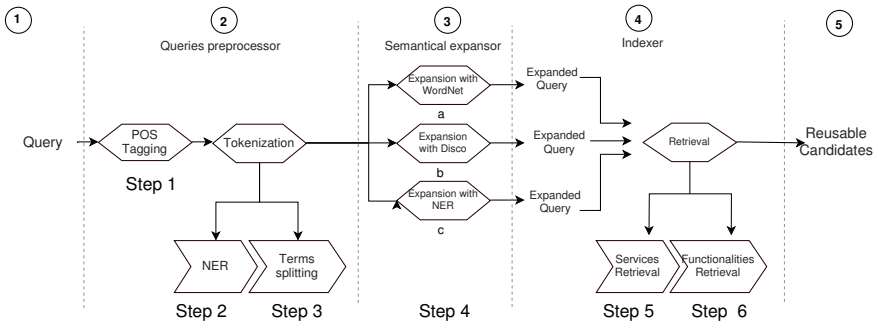

Figure 4. Reuse Manager's Workflow

a) Step 1 - Tagging word types (POS Tagging)

It is important to consider that a given word can have different meanings depending on the context or the way it is used. Therefore, the first task is tagging the words according to the way they were used by the query expressed in natural language. To do so, we used a variation of the tagger introduced in [9] (part-of-speech) and we tagged the words according to the 36 categories defined by the *Penn Treebank Project* [1] , which include different types of nouns, verbs, conjunctions, adverbs, etc.

For our example, after tagging we can observe that *When* corresponds to the category WRB (Adberv Wh-), *used* to the category VBN (Verb in past perfect), *by* to the category IN (preposition); and so forth.

b) Step 2 - Named Entity Recognizer (NER)

After tagging, we apply the NER technique for recognizing named entities (NER [2]) to improve tokenization. A named entity is a word or a set of words that constitutes an entity able to be classified into predefined classes. These classes include *Location, Person, Organization, Money, Percent, Date* and *Time*. Therefore, we apply NER, as it is presented in [10], to refine the initial list of words. Those entities conformed by more than one word are grouped again and tagged with its corresponding named entity class. Coming back to our example, the process would identify three entities, *Calin Rovinescu , Air Canada* and *Montreal*, tagging them as *Person*, *Organization*, and *Location* respectively. It makes that the words *Air* and *Canada* are not expanded separately, avoiding meaningless information with respect to the original query.

c) Step 3 - Tokenization

Firstly, we start tokenizing the query in such a way that each word can be treated separately. Then, the list of tokens should be filtered by eliminating symbols, letters, words, and other unwanted or useless elements from the semantic point of view. These elements are called *stopwords*. Now, following our example and considering that during the entity recognition *Calin Rovinescu* and *Air Canada* were detected as entities, the vector of terms would be [*used*, *Calin Rovinescu, president,inside, Air Canada,...*]. We can note that the words *when, by, the,* were eliminated since they were classified as *stopwords*.

[1] https://www.cis.upenn.edu/˜treebank/

[2] http://nlp.stanford.edu/software/CRF-NER.html

2) Query Expander

Query expansion means adding semantic value through some expanding process, which can be carried out by following different methods from the information recovery field. For instance, to add value, a method might explore semantic relationships among terms to expand the set of words of a query. As these semantic relations may vary according to each method, we use three expansion methods in such a way that efficacy evaluation can be measured for the process depicted in Figure 4.

a) Step 4.a – Query expansion using WordNet

Firstly, we search for the root of the terms (*stemming*) by using the algorithm provided by WordNet. In this way, we start to build an expanded vector of terms that will be used by the search engine. Having the correct meaning of a word is a key issue when using WordNet, since it offers a set of definitions organized according to the type of word. To select the definition, we used the Part of Speech tagging performed in the pre-processing step. Due this tag was done by using 36 categories and WordNet only has 4 – *(noun, verb, adjective* and *adverb*), we had to make a correspondence between the two sets of categories. For instance, the categories *NNS (Noun, plural)* and *VBN (Verb, past participle)* of PTBP were mapped to the category *noun* and *verb* of Wordnet respectively. In spite of identifying the type of a word, we should note that a word can have more than one meaning for the same type. In this case, we arbitrary select one of the meanings of the set.

b) Step 4.b – Query expansion using DISCO

This tool determines similarity relationships between words by performing a co-occurrence analysis. That is, similar words are those that appear along with the initial word inside a text corpus extracted from Wikipedia. In this way, for each word in the already cleansed vector of terms, we obtained a list of descendent ordered co-occurrences. Then, we take the k first words and we add them to the vector. So, the new query will contain the original words plus the k more frequent occurrences of them.

c) Step 4.c – Quey expansion using NER

Our proposal of expanding with NER uses a knowledge base that can be queried about the detected entity. Our current implementation uses an API of Wkipedia [3] to get information about a particular entity. The recognized and tagged entities in previous steps are converted into messages to the API, getting the associated information in JSON [4] as the answer. This information is processed by an algorithm that finds the most frequent terms not considered as *stopwords*. In this way, the k most frequent terms are added to the query. Note that the knowledge base might be changed just considering that there is a defined interface for retrieving the stored information.

[3] https://www.mediawiki.org/wiki/API:Main_page

[4] http://www.json.org/

3. Indexer

According to the general process depicted in Figure1, the fourth step summarizes the use of expanded requirements and the generation of a product. In our case, the use of expanded requirements means *retrieving* functionalities as a list of candidates.

a) Step 5 – Service Retrieving

Indexer retrieves taxonomy services by using the queries generated from the requirements. Then, it produces an ordered list of candidate services that are relevant to the queries. These services are part of the designed functionalities; therefore, we take n services from the list of relevant services and convert them into queries used to search those functionalities. In this way, the taxonomy is like a pivot between requirements and complex functionalities.

b) Step 6 – Functionality Retrieving

The n queries retrieved in the previous step (generated from the taxonomy services), are used to search stored functionalities. For each query, we get a set of candidate functionalities; so we get n sets. Finally, we generate a new list of unified candidates by using the relevance value that the indexing engine calculates and by grouping those candidate functionalities that are equal. This list is handed out to the user.

4.2 Implementation

We have implemented a first version of our working framework by using JAVA and Spring Resftul [5] for the processing layer; along with a web interface developed with Angular [6]. This tool allows interchanging and adding new expanding and indexing techniques.

5 Experimental evaluation

Our framework is currently implemented as a working prototype, which uses a taxonomy in a given a format and any set of functionalities described as functional datasheets. In this context, we have built a service taxonomy [8] for the geographic domain, following the standard ISO 19119 [7]. We also built and indexed a set of functionalities that form the SPL.

In order to evaluate our work, we designed an experiment whose objective is to analyze the performance of the different combinations of the interchangeable components that are currently implemented by the framework (expansion strategies and indexers). Performance of a particular combination should measure how effective is the framework at identifying reusable functionalities from a set of software requirements. At the moment, our framework implements three semantical expansion techniques and three indexers, giving a total of nine possible combinations to compare. In addition, we performed an experiment execution that skips the semantical expansion process.

[5] https://spring.io/guides/gs/rest-service/

[6] https://angularjs.org/

[7] http://www.iso.org/iso/catalogue_detail.htm?csnumber=39890

5.1 Experiment configuration

Due to our proposal acts as a mediator between developers and the SPL, helping with existing functionality retrieval from new software requirements, our experiment aims in this direction. For this matter, it is necessary to establish a ground scenario, similar to the product instantiation process, by using an implemented SPL and a set of experimental requirements. To generate this scenario we have defined our experiment in the following way:

(i) *Definition of experimental requirements:* we must define a set of experimental requirements that will be used to instantiate the new product.

(ii) *Dataset configurations:* it is necessary to prepare the dataset on which the process is going to be applied, using as input the requirements defined in step 1. In our scenario, the dataset is conformed of all the elements that describe the functionalities of the SPL.

(iii) *Definition of retrievable functionalities:* After defining the requirements and the dataset it is necessary to establish which are the expected results in order to assess the results (gold standards or ground truth). This way, we must establish which functionalities are expected to be retrieved from which requirements.

(iv) *Execution:* After defining all the elements that conform the experiment, it must be executed.

(v) *Results analysis:* the final step is to analyze the results and determine if the results meet the gold standards defined in step 2. This way we are able to measure performance through the well known metrics *recall* and *cumulative precision at N* [30].

Following, each of the previously described steps are explained in detail.

1. Defining experimental requirements:

The first step is to establish the set of requirements that will form the input of the evaluation process. As our SPL was developed for the marine ecology subdomain, our requirements also belong to this subdomain. We contacted experts of different marine biology organizations (CEntro Nacional Patagónico [8] and Almirant Storni Marine Biology Insitute [9]) and obtained 58 functional requirements, described in natural language.

Due to all processing is peformed for the english language, all of the requirements are written in this language. A functional requirement is described as shown in Table 2. All requirements are grouped in the form of *Funcionalities*, where each functionality contains several related requirements. For example, the table shows the *Zoom* functionality, which is described by four functional requirements including, for example, the *Add zoom tool selector to the map* requirement.

[8] http://www.cenpat-conicet.gob.ar/

[9] http://www.ibmpas.org/pagina.php?id=2

Table 2
Functional requirement example.

Functionality	Functional requirements
Zoom	Add zoom tool selector to the map.
	Allow the users to zoom in and out by a specific ammount.
	Allow the users to zoom in to the entire map.
	Allow the users to zoom to fit a particular layer,
	allowing them to detect possible misplaced data.

2. Dataset configuration

In our experimental scenario the dataset is composed of two main components – a service taxonomy and a set of functional datasheets. At the moment, there is a taxonomy [8] of around 150 services, indexed and stored in a database. Using this taxonomy and the *Definition Manager*, we defined 20 functionalities that combine, trough different variability relations, some of the taxonomy's services. Each functionality makes a functional datasheet, which stores the name of each involved service and the relations between them.

3. Definition of retrievable functionalities

With the objective of measuring the results of the retrieval process, we considered the previously defined set of functionalities. This way, we must define beforehand which functionalities should be retrieved from which requirements, in order to verify the validity of the results. To achieve this objective, requirements were initially mapped with the existing functionalities by a group of software engineers from the Software Engineering Department of the National University of Comahue. For each requirement, they established which datahseet should have been returned after the execution of the experiment. Thus, we constructed a list of candidate functionalities for each requirement that was later compared to the obtained results.

4. Execution

After defining all the necesary elements to perform the experiment, and then establishing all the corresponding measurements, we executed the experiment considering ten different combinations of techniques:

- One combination that skips the semantical expansion process.
- Three different combinations for each of the three semantical expansion techniques, one for every implemented indexer.

The main objective of performing such number of combinations was to reduce the implication of the indexer over the obtained results. Trying to prove the independence of the

indexers, we used the indexing engines Lucene [10], Sphinx [11] and Minion [12]. In this way, we look to evaluate the behavior of the three expansion methods despite the chosen indexer. In addition, we wished to compare the expansion methods against the indexers without any type of semantical expansion, which gave a us a total of four elements to compare for each indexer: no semantical expansion, expansion using WordNet, expansion using DISCO and expansion using NER. Also, it is important to highlight that for each indexer we took the 10 most relevant results into consideration.

5. Results analysis

To analyze performance, we considerd two well known metrics from the information retrieval field: *cumulative precision at n* and *recall*. The first one, calculates the precision at different points of the candidate list returned during the selection. Formally, *cumulative precision at n* for an individual query is defined as:

$$Precision - at - n = \frac{RetRel_n}{n},$$

where $RetRel_n$ is the number of relevant taxonomy services retrieved in the first n positions.

The *recall* metric measures the performance of the taxonomy services selection process, using the relevant retrieved services. The metric is defined as:

$$Recall = \frac{retrieved}{relevant}$$

where recall is 1 when all relevant services are retrieved in the candidate list.

Graphic A of Figure 5 shows *recall* values for each of the existing expansion techniques (differenced by a filling pattern) and for each indexer (differenced by a color). To improve visualization of the results, we used the average *recall* and *precision-at-n* values (of all indexers) for the non expansion execution, leaving a total of 10 comparable results for each metric.

To understand the obtained results, it is important to describe the reference value for each metric. For the *precision-at-n* metric, values vary from 0 to 1 for each value of N, where N is the position in the result list. From the total of results for each combination, we observed how many correct results where at position 1, at position 2, and so forth. Since we used cumulative precision, there is a value of N from which precision becomes 1, this position on the list represents the value for which all relevant results where already retrieved. In the worst case scenario, this value corresponds with the total of elements. For the *recall* metric, the value goes from 0 to 1 and it is measured with respect to a window of elements from the retrieved list (in our case the 10 first results). Hence, a recall value of 0 represents the case where none of the elements that should have been retrieved was actually retrieved; and 1 the case where all of the elements were retrieved.

After establishing these ranges of values, it becomes possible to observe several particularities on the graphics. On the one side, recall is significantly low for (less than 0.5) for the non expansion column, meaning that half of the functionalities that should have been retrieved, were not. From the techniques that use DISCO and WordNet, we can observe a clear superiority for DISCO, above all of the indexers. Finally, it is imporant to note that

[10] http://lucene.apache.org/core/

[11] http://sphinxsearch.com

[12] https://minion.java.net/

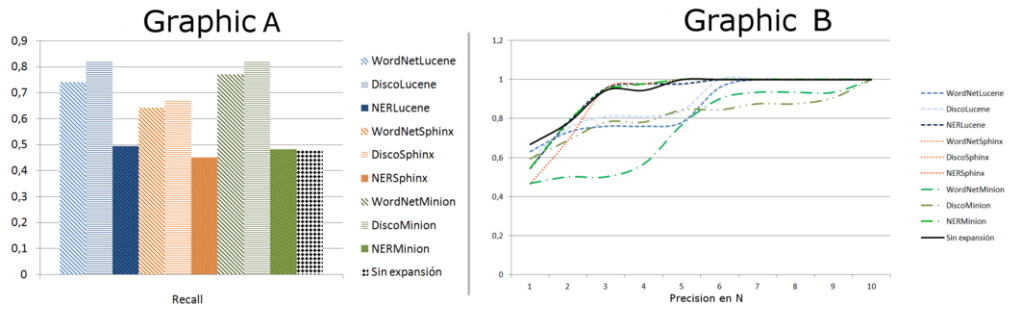

Figure 5. Measurements of *recall* and *Precision at N* for all expansion techniques and indexers.

the NER technique is not significantlly superior than the one that does not use semantical expansion.

Graphic B of Figure 5 shows the values of the *Precision at N* metric, for indexers and semantical expansion techniques, including the values of precision for the non expansion execution of the experiment, and for the 10 first results of the candidate list. As it can be observed in the reference, each indexer is identified by a pattern, and each technique by a color. We can observe that cumulative precision for the non-expansion technique starts with a value of 0.6 and rapidly climbs to a value of 1. All other techniques start with values above 0.5, meaning that for all techniques, half of the relevant functionalities were found in the first position. It is important to note that precision is a measurement that is calculated independently from recall, and both factors should be taken into account in order to evaluate performance.

5.2 Discussions and threats to validity

Analyzing both metrhics we can observe that semantical expansion techniques present a significant increment in the correct retrieval of functionalities compared to the same process without expansion. Also, cumulative precision at n reaches 100% for lists of 10 candidates in all cases. We can observe a pretty marked diference in the recall metric between all techniques that spreads across all indexers, being DISCO the one that offers the best values followed by WordNet and followed by NER.

There are at least two main threats to validity. On the one hand the dataset could be much larger in order to obtain better quantitative results. On the other hand, experiments were realized over a single domain, which might expose a correlation between the expansion techniques and the application domain. Thus, the lower values of recall for the NER expansion technique might be due to this particular domain, but could perform significantly better in another context.

6 Conclusion and Future Work

One of the pillars that support the LPS paradigm is the reuse of previously developed functionalities. For this, it is crucial to carry out a good management of the whole process, mainly of the domain requirements, so that existing functionalities will not be implemented again when new requirements appear, avoiding the generation of duplicates. Throughout

this article, we have defined a working framework for the management of software requirements of an LPS, establishing the architecture required for its implementation and presenting each of the elements that conform it.

The architecture for the framework allowed us to define two interchangeable elements: the strategies for the process of semantic expansion of queries and indexers, giving rise to to the systematic evaluation of all of them. Thus, we have evaluated the performance of different combinations of expansion techniques and indexers, analyzing the results obtained with metrics well known in the field of information retrieval.

This work lays the groundwork for future assessments of new techniques setting reference values that allow us to evaluate possible improvements. As future work we are defining new experiments in other domains to eliminate any possible correlation between the domain and the framework.

References

[1] H. Mili, A. Mili, S. Yacoub, and E. Addy, *Reuse-based Software Engineering: Techniques, Organization, and Controls*. New York, NY, USA: Wiley-Interscience, 2001.

[2] G. Kotonya and I. Sommerville, *Requirements engineering: processes and techniques*. Wiley Publishing, 1998.

[3] D. Zowghi and C. Coulin, *Requirements Elicitation: A Survey of Techniques, Approaches, and Tools*, pp. 19–46. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[4] V. Ambriola and V. Gervasi, "Processing natural language requirements," in *Proceedings 12th IEEE International Conference Automated Software Engineering*, pp. 36–45, Nov 1997.

[5] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in *29th International Conference on Software Engineering (ICSE'07)*, pp. 499–510, May 2007.

[6] K. Ryan, "The role of natural language in requirements engineering," in *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, pp. 240–242, Jan 1993.

[7] A. Buccella, A. Cechich, M. Arias, M. Pol'La, M. del Socorro Doldan, and E. Morsan, "Towards systematic software reuse of gis: Insights from a case study," *Computers & Geosciences*, vol. 54, pp. 9–20, 2013.

[8] A. Buccella, A. Cechich, M. Polla, M. Arias, M. del Socorro Doldan, and E. Morsan, "Marine ecology service reuse through taxonomy-oriented {SPL} development," *Computers & Geosciences*, vol. 73, no. 0, pp. 108 – 121, 2014.

[9] K. Toutanova and C. D. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," in *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, EMNLP '00, (Stroudsburg, PA, USA), pp. 63–70, Association for Computational Linguistics, 2000.

[10] *Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling*, ACL '05, (Stroudsburg, PA, USA), Association for Computational Linguistics, 2005.

[11] E. M. Voorhees, *Query Expansion using Lexical-Semantic Relations*, pp. 61–69. London: Springer London, 1994.

[12] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[13] T. Bhowmik, N. Niu, J. Savolainen, and A. Mahmoud, "Leveraging topic modeling and part-of-speech tagging to support combinational creativity in requirements engineering," *Requirements Engineering*, vol. 20, no. 3, pp. 253–280, 2015.

[14] S. Gulia and T. Choudhury, "An efficient automated design to generate uml diagram from natural language specifications," in *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*, pp. 641–648, IEEE, 2016.

[15] Y. Mu, Y. Wang, and J. Guo, "Extracting software functional requirements from free text documents," in *Information and Multimedia Technology, 2009. ICIMT'09. International Conference on*, pp. 194–198, IEEE, 2009.

[16] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated extraction and clustering of requirements glossary terms," *IEEE Transactions on Software Engineering*, 2016.

[17] E. Boutkova and F. Houdek, "Semi-automatic identification of features in requirement specifications," in *2011 IEEE 19th International Requirements Engineering Conference*, pp. 313–318, Aug 2011.

[18] A. Ferrari, G. O. Spagnolo, and F. Dell'Orletta, "Mining commonalities and variabilities from natural language documents," in *Proceedings of the 17th International Software Product Line Conference*, pp. 116–120, ACM, 2013.

[19] S. J. Körner and T. Brumm, "Natural language specification improvement with ontologies," *International Journal of Semantic Computing*, vol. 3, no. 04, pp. 445–470, 2009.

[20] Y. Wang, "Semantic information extraction for software requirements using semantic role labeling," in *Progress in Informatics and Computing (PIC), 2015 IEEE International Conference on*, pp. 332–337, IEEE, 2015.

[21] G. Capobianco, A. D. Lucia, R. Oliveto, A. Panichella, and S. Panichella, "Improving ir-based traceability recovery via noun-based indexing of software artifacts," *Journal of Software: Evolution and Process*, vol. 25, no. 7, pp. 743–762, 2013.

[22] G. J. Hahm, M. Y. Yi, J. H. Lee, and H. W. Suh, "A personalized query expansion approach for engineering document retrieval," *Advanced Engineering Informatics*, vol. 28, no. 4, pp. 344–359, 2014.

[23] S.-P. Ma, C.-H. Li, Y.-Y. Tsai, and C.-W. Lan, "Web service discovery using lexical and semantic query expansion," in *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*, pp. 423–428, IEEE, 2013.

[24] N. H. Bakar, Z. M. Kasirun, N. Salleh, and H. A. Jalab, "Extracting features from online software reviews to aid requirements reuse," *Applied Soft Computing*, vol. 49, pp. 1297–1315, 2016.

[25] G. Böckle, F. J. van der Linden, and K. Pohl, *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.

[26] M. A. B. M. P. M.Arias and A. Cechich, "Datasheet modeler: una herramienta de soporte para el desarrollo de funcionalidades en lps," in *XXI Congreso Argentino de Ciencias de la Computación*, 2015.

[27] W. Frakes and C. Terry, "Software reuse: Metrics and models," *ACM Comput. Surv.*, vol. 28, pp. 415–435, June 1996.

[28] M. Arias, A. Buccella, and A. Cechich, "Búsqueda de funcionalidades basada en expansión de consultas para spls," in *XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016).*, 2016.

[29] M. Arias, A. D. Renzis, A. Buccella, A. Cechich, and A. Flores, "Búsqueda de servicios para asistir en el desarrollo de una línea de productos de software," in *Proceedings of ASSE 2015*, (Rosario, Argentina), 2015.

[30] D. Godoy and A. Amandi, "Hybrid content and tag-based profiles for recommendation in collaborative tagging systems," in *Latin American Web Conference, 2008. LA-WEB'08.*, pp. 58–65, IEEE, 2008.