A Framework for Managing Requirements of Software Product Lines

MaximilianoArias[‡] [*1], Agustina Buccella[‡] [*] and Alejandra Cechich [‡]

[‡]*GIISCo Research Group, Faculty of Informatics, UNComa University, Neuquen, Argentina*
[*]*Also National Scientific and Technical Research Council (CONICET)*
[1]*corresponding author:* maximiliano.arias@fi.uncoma.edu.ar

***Resumen*—An emerging problem in the Software Product Line (SPL) paradigm is the need for integral management of planned reuse. In SPL there are two instances where managing requirements gains importance. 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 which allows management of requirements using Natural Language Processing and Information Retrieval techinques, to structure, clean, index and find reusable functionalities according to those requirements. This framework is built in a way which allows the combination of such techniques to evaluate the best combinations for finding the correct functionalities in each SPL domain.**

***Index Terms*—Software Product Lines, Requirements Engineering, Natural Language Processing**

I. Introducción

Los paradigmas basados en reuso buscan diseñar y desarrollar artefactos de software pensados para ser reusados en un futuro, en ocasiones dentro de un mismo dominio de aplicación [1]. De esta manera, se invierte un gran esfuerzo inicial realizando un análisis del dominio y obteniendo los requerimientos que servirán para construir estos artefactos reusables. De aquí la importancia de poder estructurar la información obtenida del dominio y los requerimientos que derivarán en los artefactos de software. La información estructurada puede ser luego explorada para encontrar, más fácilmente, artefactos a reusar ante la aparición de nuevos requerimientos de software.

La Ingeniería de Requerimientos (IR) es el campo de las ciencias de la computación que se encarga de definir, documentar y mantener los requerimientos de software [2] durante el proceso de desarrollo. La naturaleza misma del proceso de elicitación de requerimientos, sugiere que los mismos serán obtenidos en lenguaje natural [3]. Por esta razón, numerosos trabajos [4, 5, 6] se han centrado en el uso de técnicas de procesamiento del lenguaje natural para poder reducir ambigüedades, detectar información faltante e incluso mejorar la trazabilidad con etapas posteriores.

En trabajos previos [7, 8], hemos estudiado varios aspectos del desarrollo de LPS, enfocándonos en particular dentro del dominio geográfico. Dentro de este paradigma, definimos una metodología de desarrollo en la que existe una serie de artefactos de software para cada etapa del proceso. Estos artefactos se apoyan en la correcta búsqueda de funcionalidades a partir de requerimientos específicos para favorecer el reuso efectivo, es decir para identificar y gestionar los mismos durante todo su ciclo de vida. En particular, en todo desarrollo de una LPS es necesario identificar, en base a requerimientos específicos, funcionalidades previamente desarrolladas que los implementan de manera de simplificar la tarea del desarrollador en cuanto al reuso de las mismas. Dicha tarea es sumamente necesaria en estos desarrollos debido a que la no identificación de funcionalidades existentes generará repeticiones de artefactos que implementen los mismos requerimientos, incluso en formas diferentes.

Considerando este contexto, nuestra propuesta se centra en la definición de un marco de trabajo que permite el análisis de los requerimeintos generados en lenguaje natural para simplificar la búsqueda de funcionalidades y su reuso posterior. Este marco se basa en el uso de técnicas de Procesamiento del Lenguaje Natural (PLN) y Recuperación de Información (RI), proveyendo los medios para poder elegir y evaluar variantes de dichas técnicas, así como reducir el esfuerzo que conlleva incorporar nuevas estrategias. De esta forma, la contribucion principal del presente trabajo esta basada en dos aspectos principales, 1) permitir estructurar requerimientos en lenguaje natural junto con un método efectivo para encontrar funcionalidades que los implementan, y 2) proveer un entorno para el intercambio de técnicas de PLN y RI de manera de permitir evaluar la combinación correcta de las mismas que provean los mejores resultados. Al mismo tiempo, esta flexibilidad permite también que el marco de trabajo pueda extenderse para ampliar los conjuntos de técnicas disponibles y así evaluar otras combinaciones.

El trabajo se encuentra organizado de la siguiente manera. En la Sección II se presenta un análisis de trabajos relacionados con nuestra propuesta. La Sección III muestra la motivación y los trabajos previos que impulsaron el presente trabajo. Luego, en la Sección IV se presenta el *marco de trabajo para la gestión de requerimientos.* La Sección V la evaluación experimental, con sus respectivos resultados y el análisis de los mismos. Finalmente, la Sección VI exhibe las conclusiones que se derivan del presente trabajo.

II. Trabajos relacionados

La gestión de requerimientos en paradigmas orientados a reuso se apoya en el uso del procesamiento del lenguaje

978-1-5386-3057-0/17/31,00©2017

natural para numerosas tareas. Existen técnicas bien conocidas y aplicadas ampliamente dentro de las que podemos encontrar el etiquetado de parte del discurso o POS Tagging [9], el reconocimiento de entidades con nombre o Named Entity Recognition (NER) [10] y la expansión semántica [11]. En particular, y con respecto a la expansión semántica, distintos acercamientos hacen uso de técnicas personalizadas en torno a bases de datos léxicas como WordNet [12], ontologías o taxonomías de dominio o algun otro tipo de herramienta que añada más conocimiento sobre una palabra. En la literatura podemos encontrar numerosos trabajos que intentan combinar alguna etapa de la IR con el procesamiento del lenguaje natural para facilitarla o incluso automatizarla. Estos trabajos buscan la generación automática o asistida de modelos [13, 14, 15], la extracción de requerimientos a partir de documentos de texto [16, 17, 18], la mejora y/o extensión de las especificaciones de requerimientos [19, 20] e incluso la gestión de artefactos de software reusables y documentos de requerimientos [21, 22, 23].

En general, a pesar de utilizar distintas nomenclaturas, la gran mayoria de las propuestas siguen un proceso muy similiar, como puede verse en la Figura 1. En principio, se toman requerimientos de software en algún formato, siendo necesario en algunos casos extraerlos de algún medio (1.a) y en otros a partir de una especificación (1.b). Los requerimientos obtenidos son preprocesados (2), pasando por diversos procesos de etiquetado y extracción de información relevante. Luego, algunas propuestas [14, 22, 19, 23, 15, 20] realizan algún tipo de procesamiento semántico (3) antes de hacer uso de la información extraida. Existe una amplia variedad de usos (4) para los requerimientos preprocesados y expandidos, dependiendo de la etapa de la IR sobre la que se centre la propuesta. En general, a partir de algun proceso de uso de la información se obtiene un producto final, el cual varía según la propuesta pudiendo ser un modelo [13, 14, 15], una especificación de requerimientos [16, 18, 20], artefactos de software candidatos [21, 22], etc.

En particular, dentro del desarrollo de LPS, surgieron también varias propuestas [18, 17, 24] que aplican un proceso de gestión similar al de la Figura 1 enfoncándose dentro de actividades específicas del desarrollo. El origen de los requerimientos varía de propuesta a propuesta, tomando como entrada requerimientos de software en forma de una especificación, extrayéndolos de descripciones públicas de proyectos o de opiniones de usuario en sitios de descarga de aplicaciones restringidos a un dominio. Luego, estos requerimientos son preanalizados y procesados semánticamente con el objetivo de encontrar dentro de ellos funcionalidades comunes. Estas funcionalidades encontradas son luego utilizadas para detectar cuáles de ellas pueden plasmarse en forma de un modelo de características. De esta forma estas propuestas plantean el uso de técnicas que permiten en general el ida y vuelta de requerimientos a modelos de características facilitando la identificación de qué requerimientos van a ser implementados por qué funcionalidades.

La Tabla I muestra una clasificación de las propuestas analizadas con respecto a las técnicas del procesamiento del lenguaje natural y/o recuperación de información que se utilizaron para llegar al fin deseado. Las técnicas analizadas son el etiquetado de partes del discurso (POS tagging), reconocimiento de entidades con nombre (NER), la tokenización y la expansión semántica. Para esta última se tuvo en cuenta una subdivisión para especificar en qué se basan las propuestas para realizar la expansión, haciendo enfásis en aquellas que hacen uso de bases de datos léxicas y aquellas que hacen uso de taxonomías u ontologías. Se agregó además una categoría *Otro* para identificar aquellas propuestas que realizan expansión semántica mediante algún otro método.

Para clarificar nuestra contribución, la Tabla I también muestra nuestra propuesta, en donde podemos ver el uso de POS tagging, NER, tokenización y expansión semántica mediante diversos métodos para la recuperación de artefactos de software reusables a partir de requerimientos. A diferencia de otras propuestas, la nuestra se presenta de manera integral, haciendo uso de todas las técnicas nombradas con diversos fines en el marco del proceso presentado en la Figura 1. De esta manera, este trabajo se centra en todo el proceso, exceptuando la extracción de requerimientos (1.a), ya que, en nuestro contexto, los requerimientos son obtenidos de forma tradicional, mediante la elicitación a expertos del dominio.

III. Motivación y trabajos previos

En trabajos previos hemos definido una metodología de desarrollo de LPS [7, 8], que se basa en la creación de diferentes artefactos de software para especificar e implementar las funcionalidades comunes y variantes dentro de un dominio en particular. Dentro de estos desarrollos y aplicado a dicha metodología, surge la necesidad de realizar una correcta gestión de requerimientos centrada en dos instancias principales y bien definidas de un desarrollo de LPS que pueden observarse en la Figura 2. Por un lado, la primera instancia surge en la *Construcción de la LPS* (parte superior de la figura, marcada con 1) donde en base a requerimientos de los usuarios expertos, estándares del dominio e información sobre variabilidad (la cual surge del análisis de las dos entradas previas), se crea un conjunto de artefactos de software que se ven plasmados en la plataforma de la LPS. Como podemos observar en la figura los artefactos creados son una taxonomía de servicios del dominio, que contiene los servicios comunes y variantes posibles en un dominio particular, y las hojas de datos funcionales que contienen los modelos de variabilidad para cada funcionalidad. Estas hojas se crean siguiendo lo establecido en una arquitectura de referencia y buscando servicios en la taxonomía antes creada. Luego las mismas son implementadas como componentes de software.

La segunda instancia en donde es importante la gestión de requerimientos es en la etapa de *ingeniería de aplicación* de la plataforma de LPS previamente construída (parte inferior de la Figura 2). En esta etapa se deben instanciar los productos en base a los requerimientos específicos de los usuarios expertos de una organización en particular. Es decir, es aquí donde se usará la LPS construida para instanciar nuevos sistemas a partir de requerimientos particulares para cada uno de ellos. La

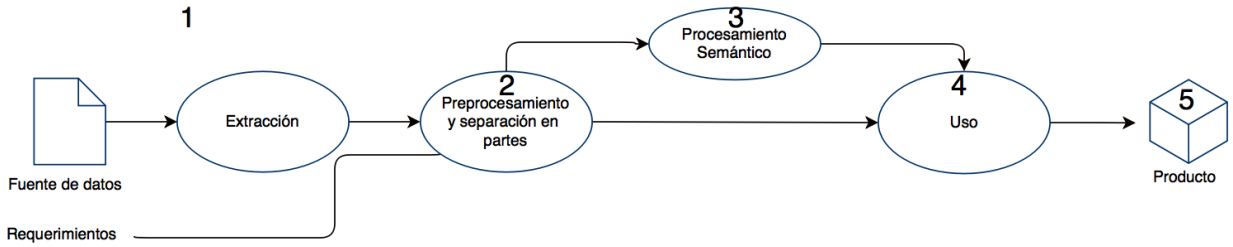

Figura 1. Proceso de gestión de requerimientos común entre las propuestas analizadas.

Tabla I
COMPARACIÓN DE PROPUESTAS EN EL USO DE PLN E IR PARA LA INGENIERÍA DE REQUERIMIENTOS.

Técnica/ Propuesta		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]	Nuestra Propuesta
POS tagging		●	●	●	●	●	●		●		●	●	●	●
NER		●										●		●
Tokenización		●		●	●		●			●	●		●	●
Expansion semántica			●											
	Relaciones léxicas							●	●	●	●	●	●	●
	Taxonomía/Ontología						●	●	●	●				●
	Otro							●						

plataforma de la LPS con sus artefactos de software previamente construidos permitirá a los desarrolladores configurar e instanciar cada uno de estos productos. A su vez, ante la aparición de nuevos requerimientos, se debe explorar la LPS en busca de funcionalidades que los implementan y la configuración de variabilidad necesaria para tal fin.

Debido a que las hojas de datos funcionales describen funcionalidades completas, éstas son el eje del proceso de instanciación de productos, pues engloban todos los servicios de la taxonomía que involucra una funcionalidad y las relaciones de variabilidad entre ellos. Las relacionas de variabilidad se describen mediante un modelo ortogonal de variabilidad [25], teniendo en cuenta variabilidad opcional, alternativa, obligatoria y relaciones de uso y exclusión.

Así, la instanciación de un sistema requiere explorar la plataforma en busca de funcionalidades capaces de implementar los nuevos requerimientos. Esta fase de búsqueda de requerimientos reusables debe ofrecer los mejores resultados posibles, buscando reducir el esfuerzo y la posibilidad de generación de funcionalidades que ya existían (duplicated features). Las funcionalidades presentes en la plataforma se originaron a partir del análisis de dominio y se encuentran, en general, descriptas en lenguaje natural. De esta manera, los avances en técnicas de PLN y recuperación de información pueden asistir a la tarea de instanciación de productos.

IV. DEFINICIÓN DE UN MARCO DE TRA BAJO PARA LA GESTIÓN DE REQUERIMIENTOS EN LPS

Nuestra propuesta se centra en la definición de un marco de trabajo que permite estructurar los artefactos reusables de una LPS y recuperarlos cuando nuevos requerimientos se hagan presentes. De acuerdo a la Figura 1, nuestra propuesta sigue el proceso de los pasos 1 al 5. Así, los requerimientos son presentados como una lista de requerimientos elicitados, que son luego preprocesados y divididos en sus partes constituyentes. Los requerimientos preprocesados son expandidos semánticamente y utilizados para recuperar artefactos de software durante el paso 4. El producto obtenido de nuestro proceso es una lista de artefactos de software reusables.

El marco de trabajo posee dos objetivos principales, uno de ellos es brindar al desarrollador una herramienta que le permita crear una estructura para describir un dominio particular y definir cúales son las funcionalidades reusables con las que cuenta. El segundo objetivo es permitir que el desarrollador sea capaz de explorar esta estructura definida y encontrar funcionalidades reusables a partir de requerimientos de software nuevos.

La arquitectura del marco de trabajo puede verse en el Figura 3, la cual está compuesta por dos elementos principales, uno encargado de la definición de la estructura del dominio (lado izquierdo de la figura) y otro encargado de explotar esta estructura para satisfacer las necesidades de reuso (lado derecho de la figura).

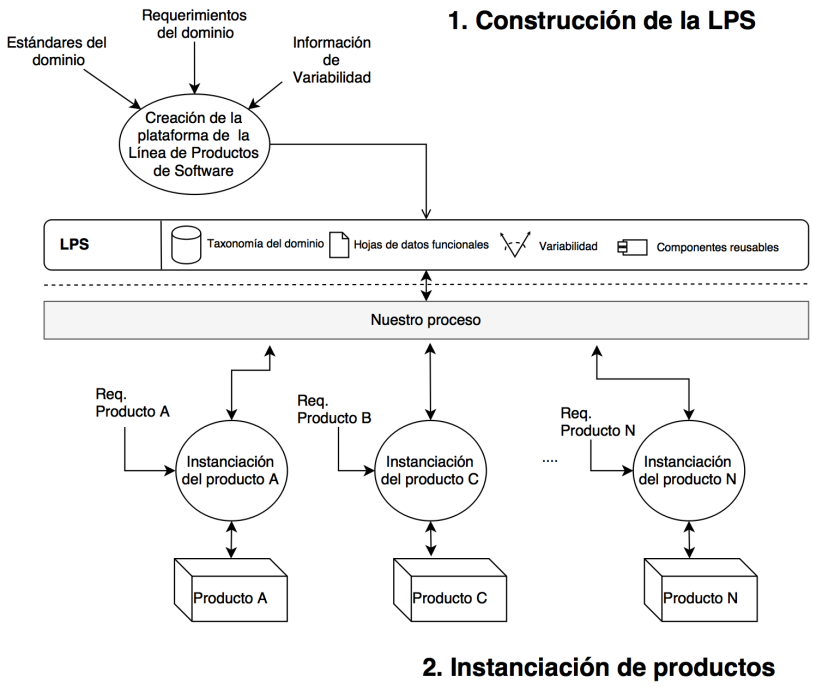

Figura 2. Proceso de construcción y uso de una LPS.

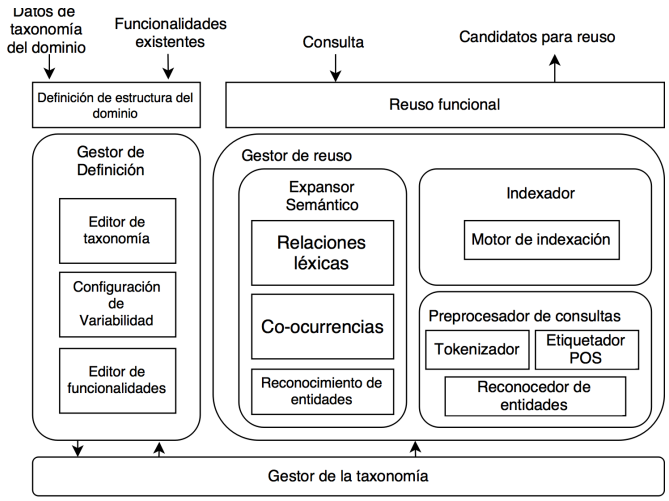

Figura 3. Arquitectura del marco de trabajo para la gestión de requerimientos en el desarrollo de LPS.

El encargado de la creación de la estructura del dominio y las funcionalidades existentes es el *Gestor de definición* (sobre el lado izquierdo) y cuenta con tres componentes: el editor de taxonomía, la configuración de variabilidad y el editor de funcionalidades (derivado de trabajos previos [26]). Para esto, nuestra propuesta toma una metodología de trabajo que hemos definido en trabajos previos [8] basada en el modelado de funcionalidades complejas a partir de servicios definidos por una taxonomía de dominio. Para modelar funcionalidades contamos con servicios, donde cada servicio es único e indivisible y cada funcionalidad es una estructura que define relaciones entre ellos. De esta manera, las funcionalidades son diseñadas con un lenguaje común, la taxonomía, lo que facilita la indexación y posterior búsqueda para reuso. La configuración de la variabilidad es definida durante la construcción de las funcionalidades, identificando variabilidad de tipo opcional, alternativa y obligatoria.

El gestor encargado de utilizar la estructura definida para recomendar funcionalidades reusables existentes es el *Gestor de reuso* y puede verse en el lado derecho de la Figura 3. El desarrollo de LPS se centra en hacer uso de funcionalidades previamente construidas específicamente para ser reusadas (reuso planeado [27]). Así, se necesita una manera de gestionar las funcionalidades creadas y poder así, ante la aparición de un nuevo requerimiento, reusar (evitando la creación de una funcionalidad duplicada). Una vez creadas e indexadas la taxonomía y las funcionalidades entra en juego este gestor.

En la parte inferior de la Figura 3 se puede observar un tercer gestor, el *Gestor de la taxonomía*, que es transversal a los dos anteriores y se encarga de manejar las escrituras y

lecturas realizadas sobre la taxonomía. Durante la definición de la estructura del dominio este gestor se utiliza en modo escritura cuando la taxonomía está siendo creada y en modo lectura cuando se están generando funcionalidades. Cuando se buscan candidatos para reuso, la taxonomía se usa en modo lectura, pues ésta se utiliza como intermediario entre los requerimientos y las funcionalidades complejas [28]. En este trabajo nos enfocamos en el *Gestor de reuso*, explicando detalladamente su estructura y funcionamiento interno.

IV-A. Gestor de reuso

Dentro del *Gestor de reuso* podemos identificar tres módulos, el *Preprocesador de consulta*s, el *Expansor de consultas* [29] y el módulo *Indexador.* El primero cuenta con tres componentes, el *Tokenizador* (encargado de la separación en términos), el *Etiquetador POS* (encargado de la detección de tipos de palabras) y el *Reconocedor de entidades* (encargado de identificar entidades con nombre). El segundo módulo, se encarga de la expansión semántica y cada componente representa una estrategia de expansión. Actualmente cuenta con tres estrategias para realizar la expansión, una basada en el uso de relaciones léxicas, otra basada en una base de datos de co-ocurrencias de palabras y otra que se basa en el reconocimiento de entidades con nombre. Finalmente, el módulo de indexación y búsqueda toma los subproductos generados por los otros dos módulos y los utiliza para explorar la estructura definida por e*l Gestor de definición*, haciendo uso del *Gestor de la taxonomía*.

El gestor de reuso sigue el flujo de trabajo definido en la Figura 4, la cual se encuentra dividida en tres columnas verticales que representan los tres módulos que conforman al gestor y cada hexágono representa un subproceso de transformación que es llevado a cabo por alguno de los componentes internos al módulo en cuestión. Podemos observar además que, de izquierda a derecha, estas columnas se corresponden con los pasos del proceso general identificado (marcados como circulos numerados) en la sección de trabajos relacionados de la Figura 1, respectivamente.

A continuación se detallan cada uno de los pasos definidos en la Figura 4. A modo de aclaración, utilizaremos el siguiente requerimiento para ejemplificar cada uno de ellos :

"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) Preprocesador de consultas: Dentro de este módulo se realizan tres pasos, cada una de ellos a cargo de uno de los tres componentes que lo forman. Estos tres pasos pueden verse numerados de 1 a 3 en la Figura 4, y son llevados a cabo por los componentes *Etiquetador POS*, *Reconocedor de entidades* y *Separador de términos* respectivamente.

a) Paso 1 - Etiquetado de tipos de palabras (POS tagging): Es importante considerar que una misma palabra puede tener distintos significados según el contexto o forma en la cual sea utilizada. Así, la primer tarea a realizar es etiquetar las palabras según la forma en la que éstas fueron utilizadas dentro de la consulta en lenguaje natural. Para esta tarea se utilizó una variante del etiquetador presentado en [9] (part-of-speech) y se etiquetaron según las 36 categorías definidas por el *Penn Treebank Project*[1] que incluyen distintos tipos de sustantivos, verbos en distintas formas, conjunciones, adverbios etc.

En nuestro requerimiento de ejemplo, luego de realizar el proceso de etiquetado podemos observar, entre otras etiquetas, que *When* corresponde a la categoría WRB (Adbervio Wh-), *used* a la categoría VBN (Verbo en pasado participio), *by* a la categoría IN (preposición) y así sucesivamente.

b) Paso 2 - Reconocimiento de entidades (NER): Luego de etiquetar las palabras según su tipo, aplicamos la técnica de reconocimiento de entidades con nombre (NER[2]) para mejorar la tokenización. Una entidad con nombre es una palabra o conjunto de palabras que forma una entidad que puede ser clasificada dentro de una serie de categorías predefinidas. Estas categorías incluyen: *Lugar, Persona, Organización, Dinero, Porcentaje, Fecha* y *Tiempo*. Así aplicamos el NER presentado en [10], refinando la lista de palabras inicial. Aquellas entidades que estaban conformadas por más de una palabra se agrupan nuevamente y se etiquetan con la clase de entidad que les corresponde.

Volviendo al ejemplo presentado anteriormente, el proceso identificaría tres entidades*, Calin Rovinescu , Air Canada* y *Montreal,* etiquteandolas como *Persona*, *Organización* y *Lugar* respectivamente. Esto permitiría por ejemplo que la palabra *Air* y *Canada* no sean luego expandidas por separado, generando información que no aporta al sentido real de la consulta.

c) Paso 3 - Separación en términos: Por un lado, se debe comenzar con la separación de las consultas en sus palabras constituyentes (tokenización) de modo que cada una de ellas pueda ser tratada por separado. Luego, el listado de tokens debe pasar por un proceso de eliminación de símbolos, letras, palabras u otros elementos no deseados o que no aportan valor semántico a la consulta, denominados *stopwords*.

Continuando con el ejemplo mencionado anteriormente, y teniendo en cuenta que durante el reconocimiento de entidades *Calin Rovinescu y Air Canada* fueron detectados como entidades, el vector de terminos tendría la forma [*used*, *Calin Rovinescu, president, inside, Air Canada,...*]. Puede notarse que las palabras *when, by, the,* fueron eliminadas del vector, pues entran en la categoría de *stopwords*.

2) Expansor semántico: La expansión de una consulta refiere al agregado de valor semántico mediante algún proceso de expansión y puede llevarse a cabo siguiendo distintos métodos propios del campo de la recuperación de información. Éstos buscan explorar relaciones semánticas entre términos para expandir el conjunto de palabras que conforman la consulta. Como dichas relaciones semánticas pueden variar de acuerdo a cada método, utilizamos tres métodos de expansión, explicados a continuación, de manera de poder evaluar la eficacia de cada uno de ellos dentro del proceso definido en la Figura 4.

a) Paso 4.a - Expansión de consultas utilizando WordNet: Inicialmente se deben buscar las raíces de los términos

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

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

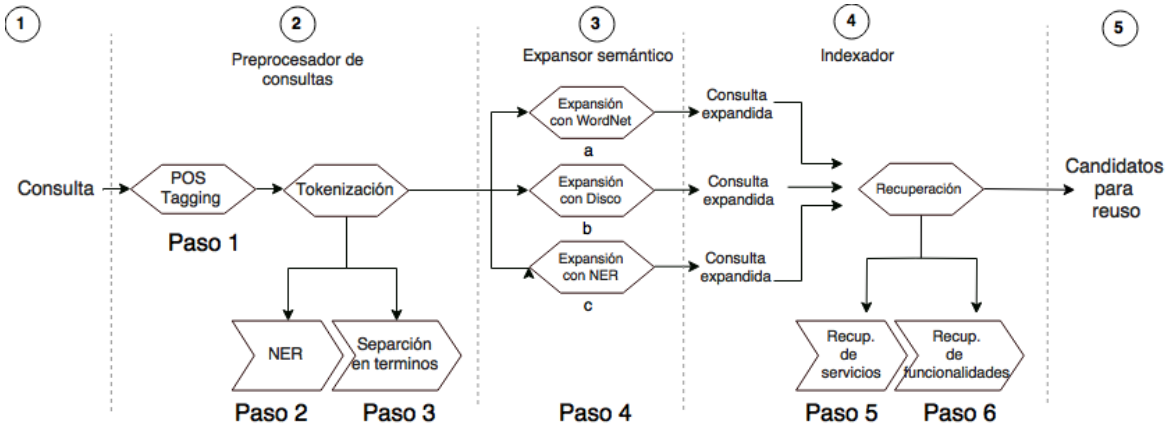

Figura 4. Flujo de trabajo del gestor de reuso para cada uno de los módulos que lo componen.

(*stemming*) utilizando el algoritmo provisto por WordNet. Así, comenzamos a construir el vector de términos expandido que utilizamos para consultar al motor de búsqueda.

La utilización del significado correcto de una palabra es un aspecto clave cuando se utiliza Wordnet pues éste ofrece un conjunto de definiciones para cada palabra organizadas según el tipo de palabra. Para seleccionar la definición se hizo uso del etiquetado de Part of Speech definido en la etapa de preprocesamiento. Debido a que el etiquetado se hizo utilizando 36 categorías de palabras y Wordnet solo cuenta con 4 *(noun, verb, adjective* y *adverb*), se realizó un mapeo entre las categorías. Así, por ejemplo, las categorías *NNS (Noun, plural)* y *VBN (Verb, past participle)* de PTBP se mapearon a las categoría *noun* (sustantivo) y *verb* (verbo) de Wordnet respectivamente. A pesar de poder identificar el tipo de palabra, una palabra puede poseer más de un significado para el mismo tipo, en cuyo caso se selecciona arbitrariamente uno de los significados del conjunto.

b) Paso 4.b - Expansión de consultas utilizando DISCO: Esta herramienta establece relaciones de similitud entre palabras mediante un análisis de co-ocurrencias. Es decir, las palabras similares a otra palabra son aquellas que aparecen junto a la palabra inicial dentro de un corpus de texto extraído de Wikipedia. De esta manera, para cada palabra del vector de términos que ya fue limpiado se obtiene una lista de co-ocurrencias ordenada de manera descendente, se toman las k primeras palabras y se agregan al vector. Así, la nueva consulta contiene las palabras iniciales más las k co-ocurrencias más frecuentes de cada una de éstas.

c) Paso 4.c - Expansión de consultas utilizando NER: Nuestra propuesta de expansión mediante NER hace uso de una base de datos de conocimiento a la cual consultar sobre la entidad detectada. Nuestra implementación actual utiliza la API de Wikipedia[3] para consultar toda la información sobre una entidad. Las entidades reconocidas y etiquetadas en pasos previos son convertidas en mensajes para la API, obteniendo como respuesta, la información asociada en formato JSON[4]. Esta información es procesada con un algoritmo que encuentra los términos más frecuentes que no sean considerados *stop-words*. Así, los k términos más frecuentes son agregados a la consulta. De esta manera la base de datos de conocimiento podría ser cambiada por otra siempre que exista una interfaz definida para la recuperación de la información almacenada.

3. Indexador: De acuerdo con el proceso general identificado en la Figura 1, el paso 4 representa el uso de los requerimientos expandidos y la obtención de un producto. En nuestro caso, el uso que damos a los requerimientos expandidos es la *recuperación* de funcionalidades, cuya salida es la lista de funcionalidades candidatas.

a) Paso 5 - Recuperación de servicios: El *Indexador* toma las consultas generadas a partir de los requerimientos y las utiliza para recuperar servicios de la taxonomía, luego genera una lista ordenada de servicios candidatos que son relevantes para esa consulta. Estos servicios son las partes constituyentes de las funcionalidades diseñadas, por lo que se toman un numero n de servicios de la lista de relevantes y se convierten en consultas utilizadas para buscar funcionalidades. De esta manera, la taxonomía funciona como un pivote entre los requerimientos y las funcionalidades complejas.

b) Paso 6 - Recuperación de funcionalidades: Las n consultas generadas a partir de los servicios de la taxonomía recuperados se utilizan para buscar las funcionalidades almacenadas. Para cada una de ellas se obtiene un conjunto de funcionalidades candidatas, por lo que se cuenta con n conjuntos. Utilizando el valor de relevancia que el motor de indexación asigna y agrupando aquellas funcionalidades candidatas iguales, se genera un nuevo listado unificado de candidatos que es entregado al usuario.

IV-B. Implementación

Hemos implementado una primera versión del marco de trabajo utilizando JAVA y Spring Resftul[5] para la lógica y una interfaz web desarrollada con Angular[6]. Esta herramienta permite el intercambio y agregado de nuevas técnicas de expansión e indexadores.

[3]https://www.mediawiki.org/wiki/API:Main_page
[4]http://www.json.org/
[5]https://spring.io/guides/gs/rest-service/
[6]https://angularjs.org/

V. Evaluación experimental

Nuestro marco de trabajo se encuenta implementado en un prototipo que toma cualquier taxonomía que respete un formato determinado y cualquier conjunto de funcionalidades descriptas mediante hojas de datos funcionales. En este contexto, hemos construido una taxonomía [8] de servicios del dominio geográfico de acuerdo al estandard ISO 19119[7]. Se construyó e indexó un conjunto de funcionalidades iniciales que conforman la plataforma de la LPS.

Para poder evaluar el trabajo realizado, se planteó un experimento, cuyo objetivo es poder analizar el rendimiento de las distintas combinaciones posibles de componentes intercambiables del marco de trabajo (estrategias de expansión e indexadores) que se encuentran implementadas actualmente. El rendimiento de una combinación de componentes debe poder medir qué tan efectivo es el marco de trabajo para identificar funcionalidades reusables a partir de un conjunto de requerimientos entre sí. Actualmente existen tres técnicas de expansión semántica y tres indexadores implementados, por lo que existen 9 combinaciones posibles que deben ser comparadas. Además contamos con una ejecución del experimento que no realiza ningún tipo de expansión semántica y que también es tomada en consideración.

V-A. Configuración del experimento

Debido a que nuestra propuesta actúa como intermerdiaria entre los desarolladores y la LPS para la recuperación de las funcionalidades existentes ante la aparición de nuevos requerimientos, nuestro experimento se centra en este aspecto. Para ello, resulta necesario que establezcamos un escenario similar al que ocurre durante la instanciación de productos, contando con una LPS implementada y un conjunto de requerimientos nuevos para la instanciación de un producto. Para poder generar este escenario hemos definido nuestro experimento de la siguiente manera:

1. *Definición de los requerimientos de prueba:* se debe definir cúal es el conjunto de requerimientos de prueba, es decir los requerimientos que se utilizarán para la instanciación de un nuevo producto.
2. *Configuración del set de datos:* Es necesario preparar el conjunto de datos sobre el cual se aplicará el proceso, utilizando como entrada los requerimientos definidos en el paso 1. En nuestro escenario, el set de datos son todos los elementos que describen las funcionalidades de una LPS (taxonomía de servicios y hojas de datos funcionales).
3. *Definición de las funcionalidades a recuperar:* Tras definir los requerimientos y el set de datos, es necesario establecer cúales son los resultados esperados para poder realizar la evaluación. Así, es necesario establecer las funcionalidades que deberían recuperarse a partir de dichos requerimientos.
4. *Ejecución:* Tras determinar todos los elementos que formarán parte de la ejecución, se debe ejecutar el experimento.
5. *Análisis de los resultados:* El último paso es analizar los resultados y determinar si las funcionalidades que se estableció que debían recuperarse en el paso 2 fueron realmente recuperadas y medir así la eficacia mediante las métricas *recall* y *precisión acumulada en N*[30].

A continuación se detallan cada uno de los pasos enumerados anteriormente.

1. Definición de requerimientos de prueba: El primer paso es establecer cúal es el conjunto de requerimientos que formarán la entrada del proceso a evaluar. Debido a que nuestra LPS se enmarca en el dominio de ecología marina, se utilizaron requerimientos de este dominio. Se contactaron expertos del dominio pertenecientes a distintas organizaciones (Centro Nacional Patagónico[8] y el Instituto de Biología marina Almirante Storni[9]) y se obtuvo un conjunto de 58 requerimientos funcionales escritos en lenguaje natural. Todos los requerimientos están escritos en idioma inglés, ya que todos los procesamientos se realizan en este idioma. Un requerimiento funcional tiene la forma que puede observarse en la Tabla II. Como se puede observar, los requerimientos se agrupan en forma de *Funcionalidades,* cada funcionalidad contiene varios requerimientos relacionados. Por ejemplo, en esta tabla se muestra la funcionalidad para realizar *Zoom* y posee cuatro requerimientos funcionales que la describen, como por ejemplo el requerimiento de agregar el selector de la herramienta de zoom al mapa.

2. Configuración del set de datos: En nuestro escenario experimental nuestro set de datos posee dos componentes, por un lado la taxonomía de servicios y por otro las funcionalidades definidas en las hojas de datos funcionales. Contamos con una taxonomía[8] que posee alrededor de 150 servicios y que se encuentra indexada y almacenada en una base de datos. Haciendo uso de esta taxonomía y utilizando el gestor de definición se construyeron los diseños de 20 funcionalidades que combinan, mediante distintas relaciones de variabilidad, algunos de los servicios de la taxonomía. Cada funcionalidad se define dentro de una hoja de datos funcional, que almacena los nombres de todos los servicios de la taxonomía que involucra y las relaciones de variabilidad que los vinculan.

3. Determinación de las funcionalidades a recuperar: Con el objetivo de poder medir los resultados de la recuperación de funcionalidades, se consideró el conjunto de requerimientos definido anteriormente. De esta manera, necesitamos poder saber de antemano cuales son las funcionalidades que corresponden a los requerimientos, de modo que luego se pueda verificar si se obtuvieron resultados correctos. Para esta tarea, los requerimientos fueron inicialmente correspondidos con las funcionalidades existentes por un grupo de ingenieros en Sistemas del departamento de Ingeniería de sistemas de la Universidad Nacional del Comahue. Para cada requerimiento se establecieron cuáles eran las hojas de datos funcionales que el proceso debía retornar luego de la ejecución. Así, construimos una lista de funcionalidades candidatas para cada requerimiento, que luego sería comparada contra los resultados obtenidos.

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

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

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

Tabla II
EJEMPLO DE REQUERIMIENTO FUNCIONAL.

Funcionalidad	Requerimientos funcionales
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.

4. Ejecución: Tras definir los elementos necesarios para llevar a cabo el experimento y luego establecer las mediciones correspondientes, se realizó la ejecución del proceso considerando diez combinaciones de técnicas distintas:

- Una combinación que no utiliza expansión semántica.
- Tres combinaciones distintas para cada una de las tres técnicas de expansión. Cada una de éstas utiliza un indexador distinto.

El objetivo de realizar todas estas ejecuciones es reducir la implicancia del motor de indexación con respecto a los resultados obtenidos. Buscando demostrar la independencia del indexador se utilizaron los indexadores Lucene[10], Sphinx[11] y Minion[12]. Así, buscamos evaluar el desempeño de los tres métodos de expansión presentados independientemente del motor de indexación elegido. Además, deseamos comparar los métodos de expansión contra los motores de indexación sin utilizar expansión semántica, lo que nos da un total de cuatro elementos a comparar para cada indexador: sin expansión semántica, expandiendo mediante WordNet, expandiendo mediante DISCO y expandiendo mediante NER. Además, cabe destacar que de la lista de resultados entregada por cada indexador se tomaron los 10 más relevantes para estudiar.

5. Análisis de los resultados: Para analizar el desempeño se consideraron dos métricas pertenecientes al campo de la recuperación de información: *precisión-en-n (acumulada)* y *Recall*. La primera calcula la precisión en diferentes puntos de la lista de candidatos retornados durante la selección. Formalmente, *precisión-en-n* para una consulta individual se define como:

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

Donde $RetRel_n$ es el número de servicios de la taxonomía relevantes recuperados en las primeras n posiciones.

La métrica *recall* mide el desempeño del proceso de selección de servicios de la taxonomía a partir de los servicios relevantes que recupera. Y se define como:

$$Recall = \frac{recuperados}{relevantes}$$

Donde recall es 1 cuando cada servicio relevante es devuelto en la lista de candidatos.

[10]http://lucene.apache.org/core/
[11]http://sphinxsearch.com
[12]https://minion.java.net/

El Gráfico A de la Figura 5 muestra los valores de *recall* para cada una de las técnicas de expansión existentes (diferenciadas por un patrón de relleno) y para cada indexador (diferenciados con un color). Para mejorar la visualización de los resultados, se tomaron los valores promedio de *recall* y *precisión-en-n* correspondientes a no utilizar expansión semántica (de todos los indexadores) dejando un total de 10 resultados a comparar para cada métrica.

Para poder comprender los resultados obtenidos, es necesario describir los valores de referencia para cada una de estas métricas. Para la métrica de *precisión-en-n,* los valores varían entre 0 y 1 para cada valor de N, donde N es la posición de la lista de resultados. Del total de resultados para cada combinación de técnicas se observó cuantos resultados correctos aparecían en la posición 1, en la 2, y así sucesivamente. Debido a que utilizamos precisión acumulada, existe un valor de N a partir del cual el valor se vuelve 1, la posición en la lista para la cual todos los elementos relevantes fueron recuperados. En el peor de los casos, este valor, corresponde con el valor total de elementos. Para la métrica de *recall*, el valor varía entre 0 y 1 y se mide con respecto a una ventana de la lista de elementos recuperados (en nuestro caso los 10 primeros). Así, un recall de 0 representa el caso donde ninguno de los elementos que debían recuperarse fue recuperado y 1 el caso donde todos los elementos fueron recuperados.

Con estos rangos de valores establecidos podemos observar varias particularidades en el gráfico. Por un lado el valor de recall es significativamente bajo (menor al 0.5) para la columna sin expansión, es decir que más de la mitad de las funcionalidades que debían haber sido recuperadas no lo fueron. Para las técnicas utilizando DISCO y WordNet, podemos ver una superioridad en la técnica DISCO sin importar el indexador utilizado. Finalmente, la otra particularidad observable es que la técnica de expansión con NER no es significativamente superior a no utilizar expasión.

El Gráfico B de la Figura 5 muestra los valores de la métrica *precisión-en-n* para todos los indexadores y técnicas de expansión, incluyendo los valores de precisión sin utilizar expansión semántica, en los 10 primeros resultados de la lista de candidatos. Como se puede observar en las referencias, cada indexador se encuentra identificado por un patrón y un color según la técnica utilizada. Como particularidades de este gráfico, podemos observar que la precisión acumulada de la técnica sin expansión comienza con un valor mayor a 0.6 a partir de la primera posición y trepa rápidamente a 1. Todas las demás técnicas comienzan con valores de precisión superiores al 0.5, es decir que para todas las técnicas más de la mitad de las funcionalidades relevantes fueron encontradas en la

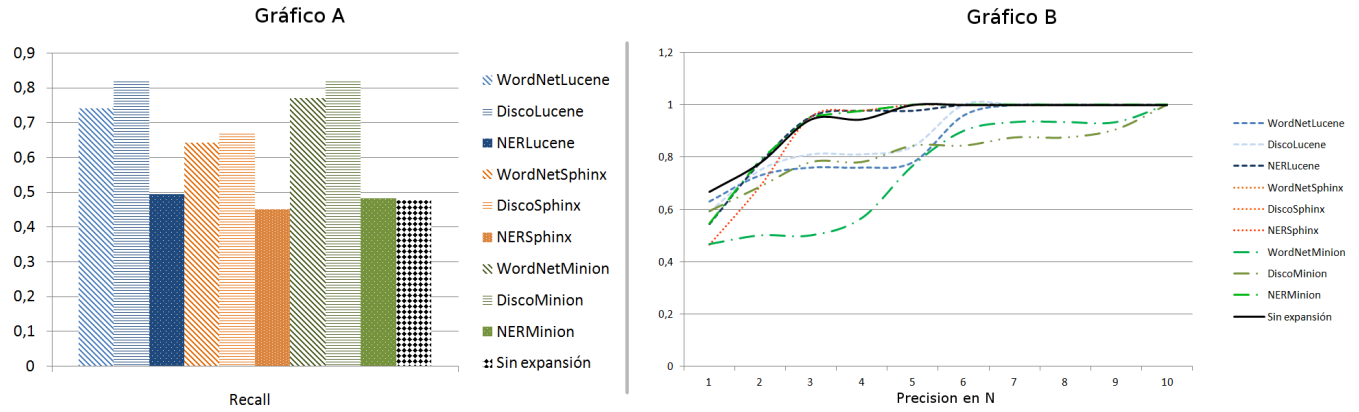

Figura 5. Medidas de *recall* y *Precisión en N* de las distintas estrategias de expansión e indexadores.

primera posición. Cabe destacar que precisión es una medida que se calcula independientemente del valor de recall, por lo que ambos factores deben ser tenidos en cuenta para estudiar la eficacia.

V-B. Discusión y amenazas a la validez

Analizando ambas métricas podemos ver que las técnicas de expansión semántica presentan un significativo incremento en la recuperación correcta (es decir que debían ser recuperadas y lo fueron) de funcionalidades con respecto a la recuperación sin expansión y que la *precisión-en-n* acumulada llega al 100 % para listas de 10 candidatos en todos los casos. Podemos observar que existe una marcada diferencia de *recall* entre las distintas técnicas que se repite independientemente del indexador utilizado, siendo la que utiliza DISCO la que mejor nivel de recall otorga, seguida por la que utiliza WordNet y más abajo la que utiliza NER.

Podemos decir que existen dos amenazas a la validez principales. Por un lado el set de datos utilizado podría ser aún más grande para otorgar mejores valores. Por otro lado, los experimientos se realizaron sobre un único dominio por lo que puede existir una relación entre la eficiencia de la técnicas de expansión y el dominio. Así, los bajos valores de *recall* para la expansión mediante el uso de NER puede deberse al bajo nivel de entidades reconocibles en los requerimientos para este dominio particular, pero podría ser más significativa en otro contexto.

VI. Conclusiones y trabajos futuros

Uno de los pilares que sostienen el paradigma de las LPS, es el reuso de funcionalidades previamente desarrolladas. Para esto, es crucial llevar a cabo una buena gestión de todo el proceso, principalmente de los requerimientos del dominio, de modo que no se vuelvan a implementar funcionalidades existentes ante la aparición de nuevos requerimientos, evitando la generación de duplicados. A lo largo de este artículo, hemos definido un marco de trabajo para la gestión de requerimientos de software de una LPS, estableciendo la arquitectura necesaria para su implementación y presentando cada uno de los elementos que lo conforman. La arquitectura para el marco de trabajo nos permitió la definición de dos elementos intercambiables: las estrategias para el proceso de expansión semántica de consultas y los indexadores, dando lugar a la evaluación sistemática de todas ellas. Así, hemos evaluado el desempeño de las distintas combinaciones de técnicas de expansión e indexadores, analizando los resultados obtenidos con métricas bien conocidas del campo de la recuperación de información.

Este trabajo sienta las bases para futuras evaluaciones de nuevas técnicas estableciendo valores de referencia que nos permitan evaluar correctamente las posibles mejoras. Como trabajos futuros debemos plantear nuevas experimentaciones en otros dominios para eliminar cualquier posible corelación entre el dominio y el marco de trabajo.

Referencias

[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, "BÃºsqueda de servicios para asistir en el desarrollo de una lÃnea 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.