Búsqueda de Funcionalidades basada en Expansión de Consultas para SPLs

Maximiliano Arias*‡, Agustina Buccella*‡, and Alejandra Cechich*

*GIISCO Research Group, Universidad del Comahue, Neuquén, Argentina

‡Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
{maximiliano.arias,agustina.buccella,alejandra.cechich}@fi.uncoma.edu.ar

Abstract. El desarrollo de Líneas de Productos de Software (LPS) es un paradigma basado en reuso que se basa en la instanciación rápida de productos similares de un dominio particular. Este paradigma hace uso intensivo del reuso de funcionalidades con un grado de personalización y configuración muy alto. Esto hace evidente la necesidad de un adecuado conocimiento del conjunto de funcionalidades actualmente existentes en la línea, debiendo sostenerse en una completa documentación. De esta manera, un correcto procesamiento de los requerimientos de software favorece al reuso efectivo de las funcionalidades de la línea. En este este trabajo se propone un proceso de búsqueda de funcionalidades basado en métodos de expansión de consultas. Estos métodos demostraron mejorar la efectividad de la búsquedas y la precisión en las primeras posiciones del ranking de sugerencias.

Keywords: Líneas de productos de software, expansión de consultas, requerimientos de software

1 Introducción

El desarrollo de Líneas de Productos de Software [5, 9] (LPS) es un paradigma centrado en reuso que busca identificar servicios comunes y variables que definen las funcionalidades de un dominio particular. Encontrar servicios comunes y variantes deriva en la generación de una plataforma de la cual se pueden instanciar productos de software, estableciendo configuraciones sobre los servicios variantes o bien agregando funcionalidades particulares de ese producto. De esta manera, una LPS conlleva un elevado tiempo de desarrollo inicial que puede verse compensado a medida que se instancian más productos.

Generar un mecanismo para el reuso efectivo de las funcionalidades implementadas no es trivial [10] y requiere un esfuerzo que debe acompañar todo el proceso de desarrollo de la LPS misma. Siempre resulta de importancia ser capaz de interpretar de alguna manera los requerimientos que se toman del usuario en lenguaje natural, para poder extraer el conocimiento de los mismos. Este conocimiento es importante porque si se cuenta con el mecanismo indicado puede favorecer a la búsqueda de requerimientos ya implementados, evitando

que se generen funcionalidades duplicadas que puedan perjudicar la evolución de la línea.

En trabajos previos [6, 7] hemos desarrollado una metodología completa para la generación de Líneas de Productos de Software, enfocándonos en particular dentro del dominio geográfico. Dentro de esta metodología existe una serie de artefactos de software que se apoyan en la correcta búsqueda de requerimientos para favorecer el reuso efectivo. Para lograr esto se lleva a cabo una serie de transformaciones que refinan los requerimientos y los transforman en subproductos que se utilizan en subsiguientes búsquedas. En este trabajo presentamos un *proceso de búsqueda de funcionalidades* en donde nos centramos en los métodos de expansión de consultas para lograr resultados más efectivos. Como contribución principal de este trabajo, se presentan alternativas en el mecanismo de búsqueda de funcionalidades implementadas dentro de la LPS utilizando nuevas relaciones semánticas, reconocimiento de entidades e identificación de categorías sintácticas.

El trabajo se encuentra organizado de la siguiente manera. En la Sección 2 se describen los trabajos previos que motivaron al desarrollo de los métodos de preparación y expansión de consultas presentados. En la Sección 3 se presenta un análisis de trabajos relacionados con nuestra propuesta. Luego se presenta el *proceso de búsqueda de funcionalidades* dentro de LPS a partir de requerimientos del usuario, haciendo hincapié en los métodos de preparación, limpieza y expansión de consultas. La Sección 5 muestra una evaluación experimental, con sus respectivos resultados, realizada para medir la eficacia de los métodos presentados. Finalmente, la Sección 6 exhibe las conclusiones y trabajos futuros que se derivan del presente.

2 Trabajos previos

En trabajos previos [6, 7] se propuso una metodología para el desarrollo de LPS la cual abarca 4 pasos. Como podemos observar en la Figura 1, estos pasos hacen uso de *artefactos de software* numerados de 1 al 6. En primer lugar, durante el paso 1 se deben analizar las fuentes de información del dominio que permiten generar y/o modificar una *taxonomía de servicios del dominio* (artefacto 1) . Tras analizar las fuentes de información y generar la taxonomía, se debe conceptualizar el dominio (paso 2) generando un nuevo artefacto de software, denominado *hojas de datos funcionales* (artefacto 3), que contienen los modelos de variabilidad de cada funcionalidad. Estas hojas se crean siguiendo lo establecido en el artefacto 2, la *arquitectura de referencia* y buscando servicios en la taxonomía antes creada. Tras generar las hojas de datos funcionales se debe realizar un estudio de reusabilidad de los componentes existentes. Los diseños son refinados en el paso 3 obteniendo como resultado un esquema de componentes abstracto (artefacto 4). Luego, en el paso 4 se debe diseñar la plataforma definiendo una arquitectura concreta para la misma (artefacto 5). La arquitectura concreta será la que se seguirá estrictamente para la etapa de implementación, que generará los *componentes concretos* (artefacto 6) implementados.

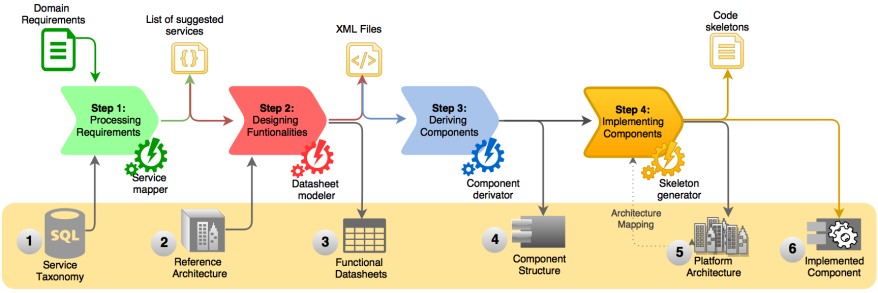

Fig. 1. Proceso de desarrollo de una LPS.

Como podemos observar, dentro de este proceso los artefactos de software cumplen un papel fundamental como soporte para las etapas del proceso de desarrollo. Este trabajo se enfoca en los pasos 1 y 2 del mismo, más particularmente en el proceso de correspondencia de requerimientos del dominio con servicios de la taxonomía. Este trabajo fue también implementado en un herramienta de soporte, llamada *Service Mapper,* la cual genera una lista de servicios candidatos a partir de un conjunto de requerimientos del dominio.

3 Trabajos relacionados

Las técnicas de procesamiento del lenguaje natural y recuperación de información se han utilizado ampliamente para el procesamiento de requerimientos de las LPSs [4]. Podemos distinguir un gran número de trabajos relacionados con la extracción de conocimiento de requerimientos de software y centrados en el procesamiento del lenguaje natural. Por ejemplo, en [2] se presentan técnicas de recuperación de información para el procesamiento de documentación del dominio. En este trabajo los autores procesan grandes documentos de distintas organizaciones y buscan encontrar similitudes que puedan ser plasmadas en un modelo de variabilidad. En [8] los autores proponen el uso de técnicas de clustering para la identificación de relaciones de necesidad entre requerimientos. Estas relaciones son luego utilizadas para asistir a los ingenieros en la construcción de un modelo de funcionalidades (feature model). Finalmente, en [12] se presenta un framework para la generación de un documento de requerimientos anotado a partir de texto plano. Este framework analiza el texto que describe un dominio en busca de agentes, acciones, restricciones, y otras categorías definidas por los autores. Finalmente, la elicitación etiquetada es utilizada para asistir en la construcción de un modelo de variabilidad.

Dentro de las técnicas para el procesamiento natural podemos nombrar el análisis semántico mediante el uso de WordNet [11], la búsqueda de palabras similiares mediante el uso DISCO[1] y el proceso reconocimiento de entidades con nombre (Name Entity Recognition - NER) [1]. La primera de estas técnicas

[1] http://www.linguatools.de/disco/disco_en.html

sugiere el uso de un diccionario léxico que organiza las palabras de manera jerárquica, para establecer relaciones de sinonimia, hiponimia e hiperonimia. La segunda hace uso de una base de datos de co-ocurrencias de palabras que establece similitud entre conceptos de acuerdo a la cantidad de veces que éstas se utilizan en conjunto, analizando un gran corpus de datos. Por último, NER sirve para el reconocimiento y clasificación de entidades dentro de un conjunto de categorías predefinidas.

En nuestro trabajo, proponemos la aplicación de técnicas de procesamiento del lenguaje natural y recuperación de información para llevar a cabo la búsqueda de funcionalidades implementadas dentro de una LPS. De esta manera, buscamos generar un mecanismo de reuso efectivo que permita reducir el riesgo de duplicar funcionalidades, eliminando tiempos de desarrollo innecesarios. Así, buscamos comparar la eficiencia de la aplicación de distintas técnicas de expansión de consultas entre sí.

4 Proceso de Búsqueda de Funcionalidades

En nuestro proceso de desarrollo, mostrado en la Figura 1, una funcionalidad es introducida como un conjunto de requerimientos del dominio o de la organización y se transforma para generar los subsiguientes artefactos de software. Como una funcionalidad puede ya existir total o parcialmente implementada en la plataforma, resulta necesario verificar esta existencia para realizar un reuso efectivo. De esta manera, cada artefacto de software se relaciona con otros artefactos de etapas posteriores, generando una secuencia de artefactos que relacionan los requerimientos con el diseño y la implementación.

Para poder garantizar esto es necesario que la búsqueda de funcionalidades dentro de la plataforma sea eficiente. Las funcionalidades se representan mediante requerimientos escritos en lenguaje natural, que debe ser procesado. De esta manera en nuestra propuesta, buscar funcionalidades completas a partir de un conjunto de requerimientos conforma una actividad compleja que se apoya inicialmente en la búsqueda de servicios en la taxonomía del dominio [3]. A medida que una descripción en lenguaje natural crece, también crece la complejidad para procesarla. Así, resulta diferente tratar requerimientos atómicos [3] a tratar funcionalidades completas, haciendo visible la necesidad de algún nivel de control sobre la estructura con la que se escriben los requerimientos. Por esto, este trabajo también propone el uso reconocimiento de entidades [1] y desambiguación de palabras según su clasificación sintáctica.

Como muestra la Figura 2 los requerimientos son ingresados al buscador de funcionalidades (*Functional mapper*) en forma de consultas. Estas consultas son preprocesadas por el *preprocesador de consultas* y utilizadas por el motor de búsqueda para encontrar servicios dentro de la taxonomía de servicios. De esta manera se obtiene un conjunto de servicios candidatos para cada requerimiento, ordenados según el criterio de similitud del buscador. Los n candidatos que obtuvieron la mejor clasificación se convierten en nuevas consultas que se utilizan para buscar dentro del repositorio de hojas de datos funcionales.

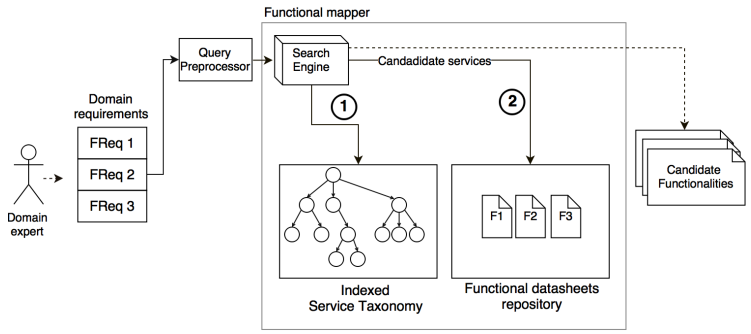

Fig. 2. Búsqueda asistida de una funcionalidad en el repositorio de funcionalidades.

Así, como las hojas de datos se construyeron a partir de la taxonomía, el buscador la utiliza como intermediaria del proceso de búsqueda de funcionalidades. Los requerimientos escritos en lenguaje natural son mapeados a servicios de la taxonomía normalizando la subsiguiente búsqueda. Entonces, si por ejemplo se partiera de una necesidad expresada por un usuario en donde se solicita la posibilidad de que el sistema exporte sus mapas en formato un TIFF esta necesidad se podría expresar mediante un conjunto de requerimientos como:

- *FREQ1: Permitir exportar mapas del usuario*
- *FREQ2: Permitir exportar un mapa en formato TIFF*

Estos dos requerimientos se utilizarían para buscar dentro de la taxonomía de servicios, sugiriendo un conjunto de servicios candidatos que incluiría por ejemplo el servicio *PS-T3 Export visible map* (donde *PS-T3* representa el código que identifica a dicho servicio y *Export visible map* su descripición). La tarea siguiente del buscador es recorrer la base de datos de funcionalidades en busca de hojas de datos funcionales que incluyan a éste y a los demás servicios candidatos (del conjunto de n servicios mencionado anteriormente) ordenándolos según el número de servicios incluidos. En nuestro ejemplo, el servicio encontrado se utilizó durante la creación de la línea de productos para implementar la funcionalidad *Export maps,* que tiene su correspondiente hoja de datos funcional. En este contexto la hoja de datos *Export maps* formaría parte del conjunto de funcionalidades candidatas para la solicitud del usuario, junto con otras funcionalidades similares.

Al mismo tiempo, podemos notar que resulta de gran interés que los servicios de la taxonomía recuperados sean correctos pues estos resultados se utilizarán luego para continuar la búsqueda. Identificamos dos actividades importantes que debe realizar el preprocesador de consultas la *limpieza (query cleaner)* y la *expansión (query expander)*[3]. Dichas actividades se muestran en la Figura 3.

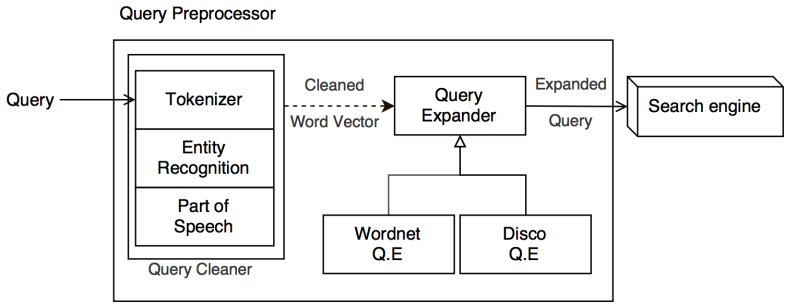

Fig. 3. Esquema del módulo de preprocesamiento de consultas.

4.1 Limpieza

En primera instancia, es necesario preparar la consulta (requerimiento funcional escrito en lenguaje natural) de modo que esta pueda ser expandida correctamente y luego utilizada para la búsqueda. De esta manera, se deben realizar una serie de transformaciones que convertirán cada una de las consultas en un vector de palabras a expandir. 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*.

Además de filtrar las palabras que no aportan valor semántico, 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 [1], 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.

A su vez, hay que considerar que una misma palabra puede tener distintos significados según el contexto o forma en la cual sea utilizada [13] . Así, aquellas palabras que no fueron reconocidas como entidades se deben etiquetar con su categoría sintáctica para luego facilitar la tarea de expansión. Para esta tarea se utilizó el etiquetador presentado en [13] (part-of-speech) y se etiquetaron según las 36 categorías definidas por el *Penn Treebank Project*[3] que incluyen distintos tipos de sustantivos, verbos en distintas formas, conjunciones, adverbios etc.

[2] http://nlp.stanford.edu/software/CRF-NER.html
[3] https://www.cis.upenn.edu/~treebank/

4.2 Expansión

La expansión de una consulta refiere al agregado de valor semántico mediante algún proceso de expansión, 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 dos métodos de expansión, explicados a continuación, y comparamos los resultados obtenidos de la búsqueda de funcionalidades con cada método entre sí y con respecto a la búsqueda sin ningún tipo de expansión de consultas (realizada en trabajos previos). La comparación se realizará no solo considerando los métodos en sí mismos, sino dependiendo de las entradas ingresadas a los mismos.

Expansión de consultas utilizando Wordnet Uno de los métodos de expansión presentado en este trabajo hace uso de la base de datos léxica WordNet. Inicialmente se deben buscar las raíces de los términos (*stemming*) utilizando el algoritmo provisto por WordNet. Así, comenzamos a construir el vector de términos expandido que utilizamos para consultar al motor de búsqueda. Como cada palabra está relacionada con un gran numero de definiciones, las etiquetas que determinan las categorías sintácticas de cada palabra son usadas para buscar más precisamente dentro del conjunto. Luego se buscan sinónimos (palabras con exactamente el mismo significado), hipónimos (palabra con un significado más específico) e hiperónimos (palabras con un significado más general) y se agregan todos ellos al vector.

Expansión de consultas utilizando DISCO El segundo de los métodos de expansión presentados hace uso de la herramienta 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.

5 Evaluación experimental

El experimento presentado tiene como objetivo evaluar el desempeño de los dos métodos de expansión de consultas, en el contexto del proceso búsqueda de funcionalidades entre sí, y con respecto a la búsqueda de funcionalidades por parte de ingenieros de software. De esta manera, se busca definir cuál de los métodos

descriptos genera los mejores resultados. Para este experimento se contó con la colaboración de un grupo de ingenieros de software que buscaron manualmente funcionalidades dentro de un repositorio a partir de un conjunto de requerimientos de software. El objetivo de los ingenieros era el de encontrar una funcionalidad que se asemeje lo más posible a los requerimientos de software solicitados. Debido a que las funcionalidades guardadas en el repositorio se crearon a partir de los servicios de la taxonomía, los ingenieros utilizaron una aplicación que permitía la visualización de la misma. Así, los requerimientos de software fueron mapeados manualmente a servicios de la taxonomía y luego a funcionalidades.

5.1 Configuración del experimento

Se definió un escenario experimental en el que fueron considerados los 120 servicios de la taxonomía previamente preprocesados e indexados. Con el objetivo de poder medir los resultados de la recuperación de servicios, se consideró un conjunto de 58 requerimientos de usuario definidos en conjunto con un grupo de biólogos marinos del Instituto de Bióloga Marina y Pesquera Almirante Storni (IBMPAS)[4] y el Centro Nacional Patagónico (CENPAT)[5] que se encuentran expresadas en lenguaje natural. Los requerimientos fueron inicialmente correspondidos con servicios de la taxonomía por un experto del dominio para poder luego analizar los resultados.

En trabajos previos [3] se obtuvieron los resultados de corresponder estos requerimientos con servicios de la taxonomía manualmente (llamado *método manual*) y mediante el uso de un buscador que realizaba una expansión básica utilizando WordNet (llamado *WordNet básico*). Estos resultados fueron contrastados contra las dos nuevas variantes que hemos definido: utilizando el método de expansión de WordNet con reconcimiento de entidades (llamado *WordNet mejorado*) y categorías sintácticas, y utilizando el método de expansión de palabras similares con DISCO.

5.2 Resultados

Para analizar el desempeño se consideraron dos métricas pertenecientes al campo de la recuperación de información: *Precision-at-n (acumulada)* y *Recall*. La primera calcula la precisión en diferentes puntos de la lista de candidatos retornados durante la selección. Formalmente, Precision-at-n para una consulta individual se define como $Precision - at - n = \frac{RetRel_n}{n}$, donde $RetReln$ 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. La medida *Recall* es 100% cuando cada servicio relevante es devuelto en la lista de candidatos.

[4] http://www.ibmpas.org/
[5] http://www.cenpat-conicet.gob.ar/

Métrica	Manual	WordNet_Básico	WordNet_Mejorado	DISCO
Precision-at-1	-	0.65	0.63	0.59
Precision-at-2	-	0.75	0.73	0.75
Precision-at-3	-	0.82	0.76	0.81
Precision-at-4	-	0.86	0.76	0.81
Precision-at-5	-	0.96	0.78	0.84
Precision-at-6	-	1	0.96	1
Precision-at-7	-	1	1	1
Recall	0.56	0.74	0.76	0.82

Tabla 1. Resultados obtenidos del experimento para cada uno de los métodos de expansión.

La Tabla 1 muestra los resultados de precisión acumulada y recall en los experimentos realizados. En ella se puede ver la primera columna que muestra los resultados de la búsqueda manual y las siguientes 4 columnas que representan los resultados de los distintos métodos de preprocesamiento de consultas descriptos previamente.

Como se puede observar en la tabla, la medida de recall para el método de expansión de disco presenta una 8% de mejora con respecto al método básico de WordNet y un 6% de mejora con respecto al mejorado. Con respecto a las medidas de precisión acumulada, ambos algoritmos de WordNet se encuentran por arriba de disco en la primera posición de la lista. El algoritmo mejorado de WordNet presenta una desmejoría en su precisión para las posiciones intermedias de la lista, alcanzando el 100% de precisión en la posición 7. Todos los mecanismos de expansión han demostrado ser más de un 18% mejores en encontrar los servicios candidatos con respecto a la búsqueda manual, con su pico máximo en el algoritmo que utiliza DISCO (26% mejor).

6 Conclusiones

El reuso efectivo de las funcionalidades que conforman una LPS se basa en utilizar una funcionalidad existente cuando se obtiene un requerimiento que ya ha sido implementado. De esta manera, es necesario llevar un estricto control sobre cúales funcionalidades ya se encuentran implementadas, de modo que un requerimiento no conlleve al desarrollo de funcionalidades que ya existen generando duplicados.

A lo largo de este trabajo hemos presentado distintas técnicas que permiten estructurar y buscar funcionalidades dentro de una LPS. Utilizando herramientas construidas a partir de estas técnicas un ingeniero de software podría verificar si un requerimiento entrante necesita o no ser implementado. De esta manera, el reuso efectivo permitirá la instanciación de productos en menor tiempo y favorecerá la evolución reduciendo la redundancia funcional. Hemos demostrado además que el uso de distintas técnicas de expansión de consultas mejoró la efectividad y la precisión de las búsquedas en ciertas posiciones de la lista de candidatos.

References

1. *Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling*, ACL '05, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.
2. V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, and A. Rummler. An exploratory study of information retrieval techniques in domain analysis. In *Software Product Line Conference, 2008. SPLC '08. 12th International*, pages 67–76, Sept 2008.
3. Maximiliano Arias, Alan De Renzis, Agustina Buccella, , Alejandra Cechich, and Andres Flores. Bï¿œsqueda de servicios para asistir en el desarrollo de una lï¿œnea de productos de software. In *Proceedings of ASSE 2015*, Rosario, Argentina, 2015.
4. Noor Hasrina Bakar, Zarinah M. Kasirun, and Norsaremah Salleh. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software*, 106:132 – 149, 2015.
5. Günter Böckle, Frank J van der Linden, and Klaus Pohl. *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.
6. Agustina Buccella, Alejandra Cechich, Maximiliano Arias, Matias Pol'La, Maria del Socorro Doldan, and Enrique Morsan. Towards systematic software reuse of gis: Insights from a case study. *Computers & Geosciences*, 54:9–20, 2013.
7. Agustina Buccella, Alejandra Cechich, Matias Polla, Maximiliano Arias, Maria del Socorro Doldan, and Enrique Morsan. Marine ecology service reuse through taxonomy-oriented {SPL} development. *Computers & Geosciences*, 73(0):108 – 121, 2014.
8. Kun Chen, Wei Zhang, Haiyan Zhao, and Hong Mei. An approach to constructing feature models based on requirements clustering. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 31–40, Aug 2005.
9. P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002.
10. Orlena CZ Gotel and Anthony CW Finkelstein. An analysis of the requirements traceability problem. In *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pages 94–101. IEEE, 1994.
11. George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.
12. 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*, pages 194–198, Dec 2009.
13. Kristina Toutanova and Christopher 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, pages 63–70, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.