ANÁLISIS SEMÁNTICO PARA LA IDENTIFICACIÓN DE ASPECTOS

Alejandro Miguel Rago

Facultad de Ciencias Exactas, UNICEN

Tandil, B7001BBO, Argentina

Teléfono/Fax: +542293439650/ int. 112

arago@alumnos.exa.unicen.edu.ar

Claudia Marcos

ISISTAN, Instituto de Sistemas Tandil

Facultad de Ciencias Exactas, UNICEN

Tandil, B7001BBO, Argentina

Teléfono/Fax: +542293439682/1

cmarcos@exa.unicen.edu.ar

Resumen

Este trabajo presenta una técnica de identificación de aspectos candidatos dada la especificación de requerimientos. Dicha técnica fue desarrollada con el objetivo de solucionar los problemas de las propuestas ya existentes, mejorando principalmente la precisión de la identificación. El proceso propuesto define una técnica automatizada que resuelve los defectos detectados, generalmente causados por las ambigüedades y vaguezas del texto natural. La técnica utiliza estrategias tales como análisis del texto con procesadores de lenguaje natural, utilización de algoritmos de desambigüación semántica, análisis de las sentencias utilizando patrones sintácticos, explotación de relaciones semánticas para agrupar palabras semánticamente relacionadas, generación de estructuras de navegación de los concerns (para realizar la identificación y filtrado de concerns), y ponderación de los concerns según su importancia.

Palabras clave

{aspectos, identificación temprana de aspectos, concerns crosscutting, análisis semántico de aspectos}

1. Introducción

El *Desarrollo de Software Orientado a Aspectos* (AOSD) [1, 2] centra su investigación en la identificación, modularización, representación y composición de concerns crosscutting. La realización de estas tareas desde etapas tempranas, tanto a nivel de requerimientos como arquitectura, es imperativa debido a que las propiedades de estos concerns tienen efectos de amplio alcance sobre otros componentes [3]. Es crucial que estas propiedades crosscutting sean correctamente controladas y administradas desde el principio del desarrollo. Si estos concerns no son identificados efectivamente, entonces no será posible razonar acerca de sus efectos sobre el sistema o sobre otros concerns. Además, la falta de la modularización de dichas propiedades desde el comienzo del desarrollo puede producir un efecto en cadena sobre otros componentes (requerimientos, arquitectura, etc.) cuando el sistema evoluciona. La provisión de métodos efectivos para manejar los aspectos tempranos hace posible establecer trade-offs críticos de manera temprana en el ciclo de vida del software.

Se ha realizado un trabajo significante en la comunidad de ingeniería de requerimientos y de diseño arquitectónico con respecto a la separación de concerns (ECOOP, OOPSLA, ICSE, etc.). El trabajo sobre aspectos tempranos complementa estos enfoques proporcionando una manera sistemática de manejar dichos concerns. Estos trabajos han obtenido resultados interesantes y prometedores. Sin embargo, los enfoques aún poseen algunas falencias, como la baja precisión en la identificación de aspectos, causados por las dificultades del análisis de texto escrito en lenguaje natural; la integración al desarrollo de software; entre otros.

Con el objetivo de solucionar los problemas identificados en los enfoques existentes, se ha definido un proceso que toma como entrada los casos de uso, sus especificaciones y el documento de requerimientos suplementarios, y realiza un análisis léxico, sintáctico y semántico sobre ellos, identificando los potenciales aspectos funcionales y no funcionales del sistema. El desarrollador sólo interactúa en el enfoque semi-automatizado para determinar, según su conocimiento y experiencia, aquellos aspectos que son parte del sistema en

cuestión y aquellos que no lo son. Finalizando, la técnica produce como salida un diagrama de casos de uso extendido, incluyendo los aspectos seleccionados por el desarrollador y sus relaciones con los casos de uso.

Este trabajo está organizado de la siguiente forma: la Sección 2 introduce el concepto de aspecto y su importancia, se presentan las técnicas de identificación existentes y se exponen sus problemas. La Sección 3 explica las características que consideramos relevantes para solucionar los problemas detectados, así como también se define el proceso de identificación de aspectos que materializa la técnica propuesta. En la Sección 4 se presentan los resultados de la evaluación de la técnica en dos casos de estudio, y finalmente, la Sección 5 describe las conclusiones y el trabajo propuesto a futuro.

2. Desarrollo de Software Orientado a Aspectos

El desarrollo de software es típicamente una tarea de alta complejidad para ser llevada a cabo, debido a que el software a desarrollar es en sí mismo de una naturaleza altamente compleja. Un largo número de necesidades, deseos y requerimientos deben ser consensuados para encontrar una solución respetando los diversos intereses. Se define un *Concern* como "cualquier asunto de interés en un sistema de software" [22], por lo tanto, el desarrollo de software tiene que tratar con un gran número de concerns. La *Separación de Concerns* (Separation of Concerns, SoC) [22] trata de separar un problema en subproblemas que pueden ser manejados y resueltos separadamente. Las soluciones parciales entonces pueden ser combinadas en una solución completa.

Mientras que algunos tipos de concern pueden ser fácilmente encapsulados dentro de artefactos como módulos, clases y operaciones a nivel de diseño o implementación, esto mismo no es posible para otros. Estos últimos *cortan transversalmente* (crosscut) el diseño o implementación de unos cuantos o incluso muchos artefactos y por lo tanto son llamados *crosscutting concerns* [22]. Típicos ejemplos de crosscutting concerns incluyen el logging, la sincronización y la distribución. Debido a su naturaleza, los crosscutting concerns causan dos problemas principales en el desarrollo de software [12]. Primero, su diseño o implementación es *dispersada* (scattered) sobre varios artefactos en vez de ser localizada en uno sólo, denominado "el *problema de scattering*". Y segundo, cuando un mismo artefacto mezcla el diseño o implementación de más de un concern, conocido como "el *problema de tangling*". Los problemas de scattering y tangling generalmente ocurren juntos, aunque son dos conceptos diferentes [1]. Estos dos problemas tienen un número bien conocido de efectos negativos sobre el software afectado por ellos.

Para dar respuesta a este conjunto de dificultades, surge el *Desarrollo de Software Orientado a Aspectos* (Aspect Oriented-Software Development, AOSD) [22], el cual tiene como objetivo aliviar estos problemas mediante la modularización de los crosscutting concerns. Los mismos son encapsulados por artefactos modulares llamados *aspectos* (aspects) y, por lo tanto, restringidos a ubicaciones separadas. El AOSD es entonces un enfoque para expandir el principio de SoC en el desarrollo de software.

Con el paso del tiempo, las comunidades de investigadores han llegado a la conclusión de que no es suficiente utilizar los enfoques orientados a aspectos solamente durante las actividades de diseño detallado y/o implementación, debido a que los crosscutting concerns no están confinados únicamente a los artefactos tratados en estas actividades. Es beneficioso soportar la orientación a aspectos desde el principio del ciclo de vida del software. De esta conclusión surge el concepto y definición de *Aspectos Tempranos* (Early Aspects, EA) [1, 7], la cual expone la importancia de tener en cuenta los aspectos tempranamente en el ciclo de vida de la ingeniería de software. Esto implica que deben identificarse también durante el análisis y el diseño conceptual, en contraste a que se realice solamente en las etapas de diseño detallado e implementación.

2.1. Técnicas actuales

Han surgido un conjunto de enfoques, técnicas y/o herramientas semi-automatizadas que permiten la identificación de los concerns crosscutting, extrayéndolos de los documentos de requerimientos. Estas herramientas tienen diferentes objetivos, y utilizan diferentes técnicas para realizar su propósito:

Theme/Doc [5-7]: Es un enfoque que expone las relaciones entre los comportamientos del sistema. Éste está basado en la noción de temas, los cuales representan un comportamiento característico del sistema. Para identificar los temas se basa en una lista de acciones y entidades clave, las que deben ser provistas por el desarrollador. La idea principal detrás del enfoque es que permite refinar las diferentes vistas de requerimientos provistas para que se pueda determinar que temas son crosscutting y cuales no, además de explicitar en que lugares se manifiestan.

EA-Miner [8, 16-21]: Es un enfoque que permite identificar los aspectos que conciernen al sistema de manera rápida, sin importar como los documentos de requerimientos están estructurados. Para realizarlo se apoya en la herramienta WMATRIX, la cual realiza un análisis sensible al contexto de la documentación provista. Se caracteriza principal-mente porque no es necesario que el desarrollador tenga que leer previamente las especificaciones para minar y modelar los concerns crosscutting.

IR for Identifying Crosscutting Concerns in Requirements Specifications [22]: Es un enfoque que se basa en la aplicación de técnicas de *Recuperación de Información* (Information Retrieval, IR) [9] para explorar las especificaciones de requerimientos en busca de algún concern en particular. El resultado de esta búsqueda es, mediante el análisis de las influencias crosscutting localizadas, determinar si el concern es crosscutting.

On Demand Virtual Remodularization Using Program Graphs [23, 24]: Es un trabajo realizado para soportar la remodularización de código orientado a objetos a código orientado a aspectos y la búsqueda de concerns particulares, utilizando una novedosa estructura denominada *Grafo de Identificación de Aspectos* (Aspect Oriented Identification Graph, AOIG). Este grafo es generado mediante un análisis del código fuente, enfocándose en los verbos y objetos directos utilizados principalmente en las signaturas de los métodos, y en menor medida, en los comentarios. Para minar los aspectos solamente se debe navegar este grafo.

Aspect Extractor Tool [10, 11]: Es un enfoque que tiene como objetivos identificar, especificar, integrar y evaluar los aspectos, basándose en los diagramas y especificaciones de caso de uso propuestas por UML. La automatización se realiza utilizando técnicas de IR [9], como stop-words y stemming. Para determinar cuándo se tiene un aspecto candidato, se realiza una búsqueda de comportamiento compartido entre casos de uso.

2.2. Problemas de las técnicas actuales

Se llevó a cabo una comparación de estas herramientas, teniendo en cuenta factores tales como dependencia de la estructura de los documentos, nivel de automatización provisto, tipo de análisis de la documentación, escalabilidad, efectividad, integra-bilidad, trazabilidad, visualización, velocidad de ejecución y evolución de los requerimientos, con el objetivo de exponer tanto sus virtudes como sus defectos. Se arribó a la conclusión de que las herramientas tienen problemas debido a las siguientes causas:

- El análisis de especificaciones de requerimientos no estructurados (no aprovechan la información adicional de un documento estructurado).
- La necesidad de ingresar información previa.
- El débil análisis sintáctico y la prácticamente ausencia de análisis semántico del texto.
- La falta de escalabilidad, principalmente cuando las herramientas son utilizadas en problemas reales y complejos.
- La poca o nula integrabilidad, trazabilidad y visualización provistas por los enfoques que no se apoyan en una metodología de modelamiento estándar.
- Los problemas de efectividad de los enfoques, al no tratar adecuadamente las ambigüedades y vaguezas del lenguaje natural. Como así también, no resolver adecuadamente el uso de sinónimos, o de palabras iguales con significados diferentes.

Estas herramientas fallan ya que si bien en conjunto tienen muchas ideas interesantes y complementarias, por sí solas todas carecen de alguna característica esencial para su uso en un desarrollo real (es decir, no un caso de estudio).

3. Identificación automatizada de aspectos

De acuerdo a las conclusiones determinadas anteriormente, se estableció que según los objetivos perseguidos, una técnica de identificación de aspectos en especificaciones de requerimientos debería contar con las siguientes características:

I. Aprovechar la información provista por una documentación estructurada para mejorar la identificación de aspectos candidatos.

II. Incorporar la técnica a alguna metodología que se integre al proceso de desarrollo del sistema de manera tal permitir tener una mayor visibilidad que al utilizar un enfoque genérico (que no detalla estas características), y que al mismo tiempo, provea de un conjunto de vistas que faciliten la lectura de los concerns.

III. Tratar, en lo posible, de evitar que el desarrollador deba ingresar información adicional para la realización del análisis para aumentar tanto la velocidad de la aplicación como la transparencia.

IV. Proveer un análisis léxico, sintáctico y semántico del texto de la documentación, que aumente la efectividad y reduzca los problemas causados por las ambigüedades del lenguaje natural.

V. Incluir una variedad de filtros que permitan reducir la información generada por el análisis hasta obtener una cantidad manejable por el analista (para cuando éste deba intervenir manualmente en la selección).

Para cumplir los objetivos de manera satisfactoria, se analizaron diferentes alternativas que permitieran llevar a cabo las características anteriormente mencionadas. Además, se tuvieron en cuenta las decisiones de implementación tomadas por los investigadores en cada uno de los enfoques o técnicas estudiadas en la comparación. Se arribó a las siguientes elecciones:

- Utilizar un *Procesador de Lenguaje Natural* (Natural Language Processor, NLP) que realice el análisis léxico y sintáctico del texto. Este análisis etiqueta las palabras de cada sentencia (indicando si es verbo, sustantivo, etc.).
- Realizar un análisis semántico de cada palabra en el que se determine precisamente qué sentido se le da a la palabra según el contexto que la rodea.
- Realizar un análisis de las palabras que permita agrupar aquellas identificadas con significados semánticos similares, utilizando diccionarios semánticos [4]. También son tenidos en cuenta diccionarios de sinónimos y algoritmos de stemming.
- Contar con información estadística de cada una de las palabras, para poder establecer la relevancia de las mismas.

3.1. Desarrollo de la técnica de identificación de aspectos

La mayoría de los análisis de identificación de aspectos se basan en la búsqueda de verbos para identificar comportamiento crosscutting. Sin embargo, sería más adecuado y preciso (siempre que sea posible) identificar sobre qué objetos actúan estas acciones o verbos, para aumentar la efectividad del enfoque. Esto es debido a que una misma acción o verbo, al ser aplicada sobre diferentes objetos, puede llegar a considerarse como diferentes comportamientos. De acuerdo a esto, al ser mayor la cantidad de información que se tiene de cada concern crosscutting potencial, la identificación de pares (verbo, objeto directo) conlleva a una mayor efectividad de análisis (cumpliendo en parte la Característica III). Por esta razón, el algoritmo principal realiza un análisis basándose en un esquema similar al de los grafos AOIG propuesto en [24], buscando pares (verbo, objeto directo) para identificar las acciones clave. Aunque como en el texto natural no siempre se incluye información sobre el objeto al que afecta el verbo, es necesario también considerar los verbos solos (sin objeto directo), para no perder información. Al mismo tiempo, el uso de una estructura similar al grafo AOIG permite realizar diferentes tipos de análisis sobre los concerns, debido a la gran cantidad de información que provee.

Se utiliza UML como base de la aplicación de identificación de aspectos (Característica II). Las ventajas de su uso son la trazabilidad e integrabilidad que provee, además que gracias a su extensibilidad mediante “profiles”, se permite especificar los aspectos en un desarrollo de sistemas integrando los aspectos de manera transparente en el proceso de desarrollo. Esto mejora la visibilidad y la comprensión de los desarrolladores sin necesidad de realizar conversiones de herramienta en herramienta. Finalmente, al ser UML considerado un estándar en los desarrollos de software actuales, la adopción de la metodología no conlleva ningún aprendizaje previo para los desarrolladores, agilizando el tiempo de utilización.

Se aprovecha el formato estructurado de las especificaciones de los caso de uso para identificar aquel comportamiento crosscutting que deba ser considerado funcional o no funcional. Se considera como no funcional cada verbo y objeto directo que se encuentre en la sección requerimientos especiales (también llamado “no funcional”) de la especificación de casos de uso o en cualquier documento de requerimientos suplementarios, mientras que en el resto de las especificaciones de caso de uso, como el flujo básico, alternativo, condiciones, etc. será considerado como funcional (Característica I). El análisis del algoritmo se realiza en dos partes separadas y consiste principalmente del etiquetado de parte del discurso (usando un NLP) y la desambigüación semántica de las palabras (satisfaciendo en parte la Característica IV) [13, 15]. Primero, se realiza sobre los textos de los flujos básicos y alternativos de los casos de uso. Segundo, se realiza sobre los requerimientos no funcionales del sistema y sobre los requerimientos adicionales de cada una de las especificaciones. Esta diferenciación se realiza para poder tener noción de cuándo el par encontrado está relacionado con un concern funcional y otro no funcional (cumpliendo parcialmente la Característica III).

Para llevar a cabo la desambigüación semántica, se utilizan bases de datos léxicas. Se optó por esta solución, debido a que cuando el dominio es desconocido, estas BBDD se comportan estadísticamente de manera estable.

Se construye un grafo (Ilustración 1) que muestra la utilización de los nodos (verbo, objeto directo) y nodos (verbo), pero en vez de hacerlo directamente a su nodo de uso en la especificación se agrega un nodo intermedio que agrupa pares de "verbos semánticamente relacionados" con "sustantivos semánticamente relacionados". Esta agrupación de palabras tiene su origen en la navegación de las relaciones semánticas entre términos de la base de datos semántica, en la que se buscan términos similares. Estos agrupamientos, al estar realizados posteriormente a la desambigüación semántica, resuelven los problemas de sinónimos, ambigüedades y vaguezas del lenguaje natural, aumentando de manera notoria la efectividad del enfoque (Característica IV).

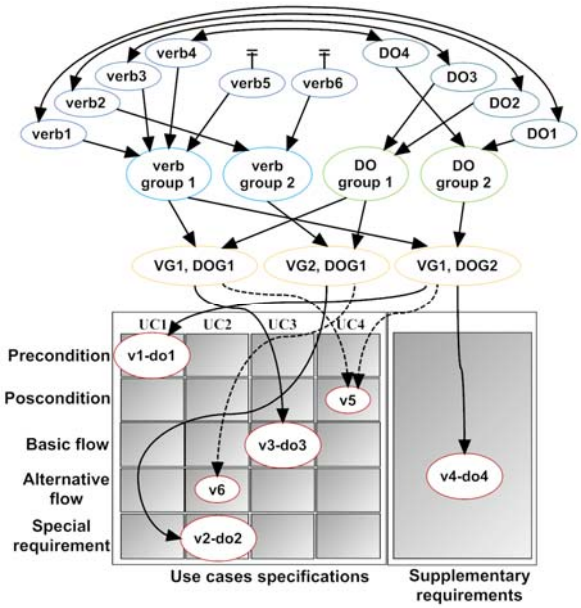

Ilustración 1 - Grafo de identificación de aspectos

Luego de la creación del grafo, se realiza un simple recorrido del mismo para determinar aspectos potenciales, teniendo en cuenta sus usos en más de una especificación de casos de uso. Cada uno de estos aspectos potenciales es clasificado en funcional y no funcional según el verbo del cual forma el par. El resultado obtenido es ordenado realizando un ranking en el que cada uno de los nodos es ponderado por las estadísticas generadas en el análisis del texto previamente efectuado más la información de crosscutting que es obtenida del recorrido del grafo.

La lista ordenada es presentada al analista para que seleccione y determine los aspectos a considerar. Se pueden utilizar filtros para reducir la lista según criterios preestablecidos, basándose en la estructura del grafo de identificación para efectuar el filtrado, lo cual permite una mayor escalabilidad del enfoque (Característica V). Entre los criterios de filtrado, es posible encontrar los filtros de verbo, de objetos, etc.

3.2. Proceso de identificación de aspectos candidatos

Las tareas que componen el proceso de identificación fueron agrupadas en bloques, los que forman conjuntos de tareas que permiten llevar a cabo objetivos específicos para la ejecución de la técnica propuesta. El proceso (Ilustración 2) consta principalmente de tres bloques diferentes: el *análisis del procesador de lenguaje natural* (NLP Analysis), *la generación del grafo* (Graph Generation), y el *buscador de aspectos candidatos* (Aspect Finder).

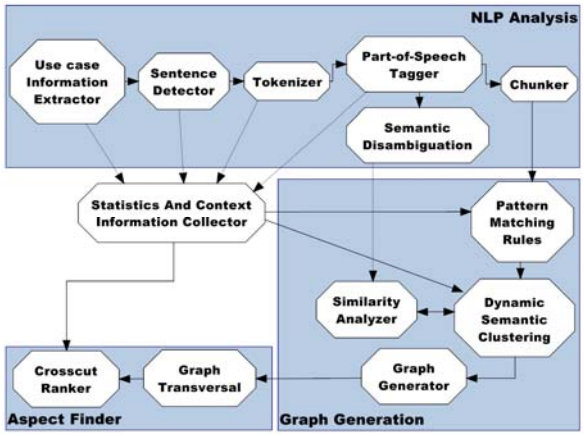

Ilustración 2 - Proceso de identificación

3.2.1. NLP Analysis

El bloque de tareas *NLP Analysis* tiene como finalidad llevar a cabo un análisis léxico, sintáctico, y semántico del texto de las especificaciones, así como también efectuar un estudio estadístico y de ubicación de este texto. Consta de seis tareas explícitas que llevan a cabo sus objetivos: *extracción de información de los casos de uso* (Use case Information Extractor), *separación de sentencias* (Sentence Detector), *separación de palabras* (Tokenizer), etiquetado POS (Part-of-Speech Tagger), *desambigüación semántica* (Semantic Disambiguation) y *agrupamiento en grupos sintácticos* (Chunker). Todas las tareas se realizan con el procesador NLP, excepto la tarea *Semantic Disambiguation*, que se ocupa de desambigüar el

significado de las palabras según el contexto que la rodea. Esto es necesario para tratar de solucionar los problemas de ambigüedad y vaguezas propios de los lenguajes naturales.

Para realizar la desambigüación de los sentidos de la palabra se utiliza una modificación al algoritmo propuesto en [14] denominado *"Maximum Relatedness Disambiguation Algorithm"*, que utiliza métricas de relaciones semánticas. Este algoritmo desambigüa una palabra polisémica eligiendo aquel sentido de la palabra que maximiza su relación con aquellas palabras cercanas que estén dentro de la ventana de contexto.

El algoritmo fue modificado para que se aplique a todas las palabras del texto y con el tamaño de ventana que abarca la oración donde se encontró la palabra objetivo. Las medidas utilizadas para determinar cuánto se relacionan las palabras en el algoritmo son dos: la propuesta por Lesk (Lesk Gloss Overlap Measure, Ilustración 3), que es rápida pero no muy efectiva, y la otra es la desarrollada por Pedersen y otros (Lesk Extended Gloss Overlap Measure, **Ilustración 4**) que es más efectiva pero también más costosa en velocidad.

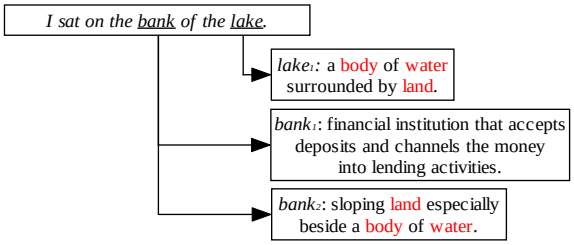

Ilustración 3 - Medida de Lesk

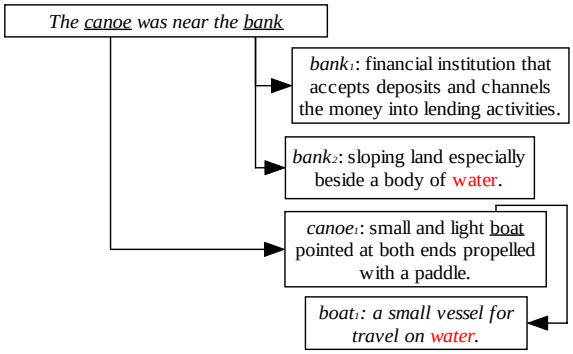

Ilustración 4 - Medida de Lesk extendida

En el algoritmo propuesto, el inventario de donde se obtienen las definiciones se corresponden con la base de datos semántica WordNet [4]. Ésta es una base de datos léxica que contiene información acerca de sustantivos, verbos, adjetivos y adverbios. El objetivo es organizar los conceptos relacionados en *conjuntos de sinónimos* (Synonym Set, Synsets). Cada synset puede considerarse como la representación de un concepto o *sentido* (Sense). Además de proveer estos grupos de sinónimos para representar un concepto, WordNet conecta los conceptos mediante una variedad de relaciones. Como consecuencia se crea una red donde los conceptos relacionados pueden ser identificados mediante el cálculo de una distancia relativa entre ambos.

3.2.2. Graph Generation

Tiene como propósito la realización del grafo de identificación de aspectos propuesto llevando a cabo aquellos pasos previos necesarios para la construcción del mismo. El bloque está formado por cuatro tareas que ejecutan esta elaboración: *detección de verbos y objetos directos* (Pattern Matching Rules), *análisis de similitud* (Similarity Analyzer), *agrupamiento dinámico de verbos y sustantivos* (Dynamic Semantic Clustering) y *creación del grafo* (Graph Generator).

La tarea *Pattern Matching Rules* se ocupa de llevar a cabo la identificación de los verbos y los objetos directos en las sentencias del texto, tomando como entrada la salida del agrupador sintáctico. Se aplican una serie de reglas que permiten identificar los verbos buscando el verbo dentro de las frases verbales e identificando en las frases sustantivas posteriores el objeto directo. Para realizarlo, se determina el tipo de verbo que se detectó en la frase verbal y en base a ello se identifica el patrón sintáctico que le corresponde. De esta manera es posible saber si la frase contiene o no un objeto directo, y si existe, en que lugar se encuentra. Se identifican pares *Verbo-Objeto directo* en aquellas frases que cuenten con objeto directo, y solamente el simple *Verbo* en aquellas que no tengan uno.

La tarea *Similarity Analyzer* está encargada de realizar el análisis de similitud entre verbos (y objetos directos) para posteriormente agruparlos en grupos semánticamente relacionados. Está basado en el resultado de la desambigüación semántica realizada previamente que tiene como salida la determinación del sentido o significado correcto de los términos según su contexto. Dados dos términos (verbos u objetos directos), se establece la métrica de similitud entre éstos utilizando las relaciones semánticas provistas por el inventario de sentidos, navegando estas relaciones en busca de las palabras que representan los synsets navegados, teniendo como origen del recorrido el sentido desambigüado de cada término. La forma de establecer el puntaje es

contando el número de términos que tienen en común los dos conceptos, considerando estos dos casos:

- Si se comparan dos verbos, se usaran dos de las jerarquías provistas por WordNet para ampliar las palabras que representan los verbos: *Hypernyms*, donde el verbo Y es un hypernym del verbo X, si la acción X es un tipo de Y ("*travel*" y "*movement*"); y *Troponyms*, donde el verbo Y es un troponym del verbo X, si la acción Y es realizar X de alguna forma ("*to lisp*" y "*to talk*").
- Si se comparan dos sustantivos (objetos directos), se utilizaran dos de las jerarquías provistas por WordNet para ampliar las palabras que representan los sustantivos: *Hypernyms*, donde Y es un hypernym de X, si cada X es un tipo de Y ("*canine*" y "*dog*"); y *Hyponyms*, donde Y es un hyponym de X, si cada Y es un tipo de X ("*dog*" y "*canine*").

Esto quiere decir que para cada término que se compara, se realiza una explosión (Ilustración 5) de cada palabra para poder tener un mejor entendimiento del concepto que representa el término. Para cada uno de los dos términos que se deben comparar, se realiza la explosión (está controlada a no más de tres niveles en cualquier dirección) y se almacenan las palabras que representan a cada synset alcanzado por la explosión.

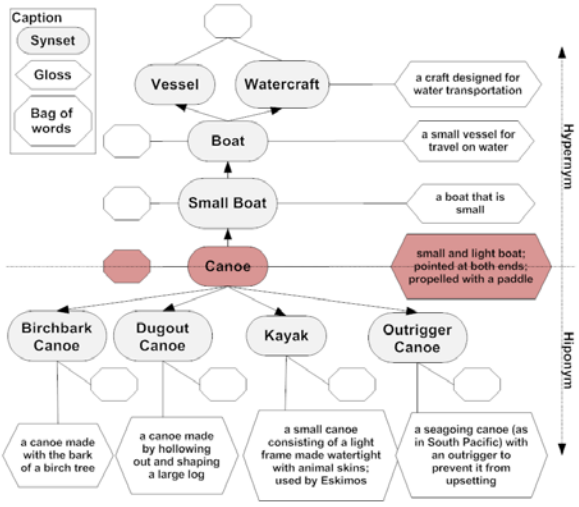

Ilustración 5 - Desglose de información

La tarea *Dynamic Semantic Clustering* se encarga de que a medida que se va analizando el texto de los casos de uso, los verbos y objetos directos descubiertos se vayan agrupando dinámicamente. Para lograrlo se utiliza el análisis de similitud entre la palabra y las palabras de cada grupo. Para aquellos grupos donde la inclusión de esa palabra mantenga (en promedio) el puntaje de similitud en un número que supere un umbral preestablecido, se la almacena como candidata a integrarse a ese grupo. Para aquel grupo donde el puntaje es el máximo, se intenta agregarlo a este grupo. Puede darse el caso de que al agregar la palabra, el grupo deba ser dividido para que el agrupamiento sea más efectivo (cuando el promedio mínimo no se respeta, o cuando el grupo es muy disperso). También puede ocurrir que no se integre a ningún grupo en particular (cuando no existen muchos grupos o cuando la palabra no está relacionada con ningún agrupamiento) y se agregue a un grupo de un sólo elemento.

Por último, la tarea *Graph Generator* lleva a cabo la creación del grafo propuesto anteriormente (Ilustración 1). Para realizarlo, se cuenta con la información generada por las tareas anteriores, principalmente con las listas y agrupamientos de verbos y objetos directos que son agregados al grafo e interconectados, de manera tal que el mismo pueda ser recorrido. Además, se agregan al grafo los nodos *Grupo de verbo* y *Grupo de objetos directo* (GV,GOD) que representan la acción de uno de los verbos del grupo sobre uno de los objetos directo del grupo, y los nodos de "uso" que relacionan estos nodos con un caso de uso en particular. Luego de esta agregación, se insertan al grafo aquellos nodos simples (verbos sin objetos directos) detectados. Esto representa una diferencia respecto del grafo AOIG original, y se realiza agregando los verbos simples a todos aquellos nodos (GV,GOD) cuyo grupo de verbo contenga al verbo en cuestión. Si no existe ningún nodo (GV,GOD) que lo contenga, se crea uno nuevo que no contiene ningún GOD.

3.2.3. Aspect Finder

Tiene como propósito, apoyándose en los artefactos generados por las tareas previas (principalmente en el grafo), la identificación de aspectos candidatos y el ordenamiento de éstos según su relevancia. El bloque está formado por dos tareas que ejecutan esta elaboración: *recorrido del grafo en búsqueda de concerns crosscutting* (Graph Transversal) y *ordenamiento de los aspectos candidatos* (Crosscut Ranker), además de la tarea adicional *filtrado de aspectos candidatos* (Candidate Aspects Filtering).

La tarea *Graph Transversal* es la que tiene la responsabilidad de identificar los aspectos candidatos. Para efectuarla, se realiza un recorrido del grafo contando aquellos grupos de verbos que cortan transversalmente una cantidad de casos de uso que

supera un umbral predeterminado. Para cada grupo de verbo que es considerado crosscutting, se realiza una exploración en busca del término (un verbo) más representativo del grupo, utilizando las estadísticas recolectadas previamente, para proponerlo como nombre candidato del concern detectado. Adicionalmente, también se establece si el concern es funcional o no funcional, analizando de donde provienen los verbos.

La tarea *Crosscut Ranker* tiene como objetivo establecer un ordenamiento en los grupos de verbos crosscutting detectados. Para establecer el orden de los aspectos candidatos, se toman en consideración varios parámetros de cada aspecto potencial: el número de casos de uso cortados transversalmente, el número de cortes transversales totales (puede haber más de una referencia en un caso de uso específico) del grupo de verbos, la relevancia de ocurrencia del grupo de verbos, la relevancia de ocurrencia de los grupos de objetos directos relacionados con el grupo de verbos en cuestión, y por último si el grupo de verbos es considerado funcional o no funcional. Estos parámetros son reunidos en una sola función (Ilustración 6), que fue definida de acuerdo a nuestras observaciones, cuyo resultado es un número real representando la importancia del aspecto candidato.

El puntaje es:

$$\Delta * (\frac{ucc}{\sum UC} * \alpha + \frac{vgcc}{\sum UC} * \beta + (\frac{vgr}{\sum V} + \frac{dogsr}{\sum DO}) * \gamma)$$

Sujeto a:

$\alpha = 10$
$\beta = 6$
$\gamma = 3$

$\Delta = 3$ si el grupo de verbos es no funcional
$\Delta = 2$ si el grupo de verbos es funcional

Donde:
ucc : use cases crosscutted
vgcc: verb group crosscutted count
vgr: verb group relevance (occurrence count)
dogsr: direct objects groups relevance (occurrence count)

Ilustración 6 - Fórmula de ordenamiento

La tarea *Candidate Aspects Filtering* es necesaria para poder manejar la selección, especificación y administración de los aspectos candidatos por parte desarrollador, luego de que la herramienta los haya encontrado. Resulta necesario contar con alguna forma de poder reducir la información generada por el enfoque en un subconjunto manejable basándose por ejemplo en determinados tipos de criterios.

Este proceso permite que se aplique en grandes desarrollos tornando la herramienta en un enfoque escalable, es decir que, sin importar la cantidad de aspectos candidatos que haya, por más grande que sea el sistema a desarrollar, el analista podrá especificarlos y manejarlos de forma eficiente y práctica. Para lograrlo, se utiliza la estructura del grafo generado para poder reducir la información basándose varios criterios. Por ejemplo, *Verbo*, en el cual se especifica el nombre de un verbo, y cada aspecto candidato que contenga este verbo será retornado. *Objeto directo*, donde se especifica el nombre de un objeto directo, y cada aspecto candidato que contenga este objeto directo en alguno de sus usos será retornado. *Cuenta crosscutting*, en el que se especifica un número mínimo de cuenta crosscutting, y cada aspecto candidato cuya cuenta crosscutting iguale o supere este número será retornado. Finalmente, *Caso de uso*, donde se especifica un caso de uso y todos aquellos aspectos candidatos que contengan algún uso en este caso de uso será retornado.

4. Evaluación

Se llevó a cabo una evaluación de la técnica propuesta en dos casos de estudio: un sistema de administración de estudiantes (SIMS), de tamaño pequeño (especificación de 3 páginas), y un sistema de revisión de artículos automatizado (ARS), de tamaño mediano (especificación de 15 páginas). Los mismos fueron ejecutados enfrentando tres tipos de análisis: uno con el enfoque original de la herramienta Aspect Extractor Tool (AET), y dos con la técnica definida en este trabajo. Esta ultima fue utilizada de dos formas: una (T1) con los parámetros del análisis semántico y el agrupamiento dinámico limitadas al mínimo, y la otra (T2) configurando los parámetros de análisis y agrupamientos al máximo.

	SIMS			ARS		
	AET	T1	T2	AET	T1	T2
Efectividad	62%	75%	62%	30%	65%	90%
Ruido	50%	33%	37%	53%	58%	55%
Identificados	10	9	8	17	31	40
Positivos	5	6	5	6	13	18
Negativos	5	3	3	9	18	22
Velocidad	2s.	30s.	1m.	5s.	60m.	90m.

Tabla 1 - Resultados de la evaluación

Los resultados encontrados (Tabla 1) fueron prometedores. Se observó un aumento de la

efectividad entre las técnicas, principalmente debido a la identificación de aspectos funcionales muy concretos (funcionalidad específica no fácilmente discernible dispersada por los casos de uso) ocultos en las declaraciones del texto, como también la identificación de las relaciones entre aspectos y los casos de uso de forma más precisa y correcta. Esto se debe en gran parte a las ventajas de llevar a cabo las tareas de desambigüación de los términos mediante el análisis del contexto y la potencia semántica de la agrupación dinámica. Sin embargo, se notó que su funcionamiento mejora a medida que se tiene como entrada especificaciones más grandes y completas. También se determinó que según los resultados, la técnica resuelve correctamente evitar el ingreso de información previa, automatizando al máximo la detección de aspectos. El objetivo de solucionar los problemas causados por los sinónimos y las ambigüedades del lenguaje fue satisfecho, ya que se observaron las grandes mejoras del agrupamiento de términos.

Un problema observado fue un leve aumento del ruido. Esto ocurrió en la gran mayoría de los casos por el uso de sentencias (como “the use case ends”) dentro de las especificaciones en información que no es relevante para el sistema. Dichos problemas están justificados porque no se filtra esta información irrelevante de los casos de uso a la hora de presentar los aspectos candidatos al desarrollador. Sin embargo, puede ser solucionado fácilmente al utilizar los filtros de aspectos candidatos, según los criterios convenientes establecidos por el desarrollador en cada caso. Otra desventaja encontrada con esta técnica fue el consumo de tiempo necesario para llevar a cabo la ejecución del proceso. Esta situación puede o no ser justificable según el nivel de precisión y efectividad que se crea necesario a la hora de identificar aspectos candidatos, como también del tamaño y complejidad del sistema analizado. Con respecto a los parámetros de la técnica nueva, las diferencias encontradas entre ambos análisis nos llevan a la conclusión de que a medida que se incrementa el análisis semántico y las estrategias de agrupamiento, se mejora la efectividad y se reduce el ruido, pero también se incrementa el tiempo de ejecución.

Se espera en trabajos posteriores, realizar una evaluación de las demás herramientas analizadas para poder contrastar la efectividad de la técnica con las existentes actualmente.

5. Conclusión

En este trabajo se presentó una técnica de identificación de aspectos candidatos de las especificaciones de requerimientos. La técnica propuesta fue definida como un intento para resolver los problemas encontrados en los enfoques y herramientas existentes. Se materializó en un proceso formado por tareas bien especificadas y con objetivos concretos. La técnica se basó principalmente en un análisis semántico de la información de los documentos de requerimientos, poniendo el énfasis en resolver los problemas al analizar un texto escrito en lenguaje natural.

Básicamente, esta técnica se diferencia de las existentes debido a que centra el análisis de texto natural enfatizando en las relaciones semánticas de los términos, utilizando como base investigaciones de WSD (Word Sense Desambiguation).

Se resolvieron varios problemas comunes en otros enfoques, como los sinónimos, vaguezas y ambigüedades de los lenguajes naturales. Adicionalmente, esta técnica se acopla a cualquier proceso de desarrollo que utilice UML como lenguaje de modelamiento, con todas las ventajas que esto conlleva, y se proveyeron filtros que permiten escalabilidad y soportan el análisis semi-automatizado en sistemas de gran tamaño. Sin embargo, quedan algunas cuestiones por resolver, como el tiempo de ejecución y los problemas que pueden llegar a surgir debido al ruido.

En el futuro, como posibles mejoras se tiene en mente perfeccionar la manera de realizar los agrupamientos, integrar los objetos directos a la identificación de concerns crosscutting, filtrar aquella información irrelevante de la lista de aspectos candidatos, optimizar los tiempos de ejecución del proceso y adicionar el soporte del lenguaje español a la técnica ya que, actualmente, la técnica está orientada las especificaciones de casos de uso escritas en inglés.

6. Bibliografía

[1] *Aspect-Oriented Software Development Net*. 2008 [cited; Available from: http://aosd.net/.

[2] *Aspect Oriented Software Development*. 2008 [cited; Available from: http://en.wikipedia.org/wiki/Aspect-oriented_software_development.

[3] *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*. 2008 [cited; Available from: http://www.early-aspects.net/.

[4] *WordNet*. 2008 [cited; Available from: http://en.wikipedia.org/wiki/WordNet.

[5] Baniassad, E. and S. Clarke, *Finding Aspects in Requirements with Theme/Doc*, in *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in*

conjunction with AOSD Conference. 2004: Lancaster, UK.

[6] Baniassad, E. and S. Clarke. *Theme: An Approach for Aspect-Oriented Analysis and Design*. in *Proceedings of the 26th International Conference on Software Engineering (ICSE)*. 2004: IEEE Computer Society.

[7] Baniassad, E., et al., *Discovering Early Aspects.* IEEE Software, 2006. **Volume 23**(Special Issue on AOSD): p. 61 - 70.

[8] Chitchyan, R., et al. *A Tool Suite for Aspect-Oriented Requirements Engineering*. in *Proceedings of the 2006 International Workshop on Early Aspects (Held At ICSE 2006)*. 2006. Shanghai, China: ACM.

[9] Figuerola, C., *La Investigación sobre Recuperación de la Información en Español*. 2000, Facultad de Documentación, Universidad de Salamanca.

[10] Haak, B., et al., *Aspects Extractor: Identificación de Aspectos en la Ingeniería de Requerimientos*. 2005, Facultad de Ciencias Exactas, UNICEN.

[11] Haak, B., et al., *Identificación Temprana de Aspectos*. Revista Sociedad Chilena de Ciencia de la Computación, 2005. **Volumen 6**(Workshop de Ingeniería de Software).

[12] Hannemann, J. and G. Kiczales. *Overcoming the Prevalent Decomposition in Legacy Code*. in *Workshop on Advanced Separation of Concerns, 23rd International Conference on Software Engineering (ICSE)*. 2001. Toronto, Ontario, Canada.

[13] Li, K., R.G. Dewar, and R.J. Pooley. *Object-Oriented Analysis Using Natural Language Processing*. in *International Conference for Young Computer Scientists (ICYCS '05)*. 2005. Beijing, China.

[14] Pedersen, T., S. Banerjee, and S. Patwardhan, *Maximizing Semantic Relatedness to Perform Word Sense Disambiguation*. 2005, University of Minnesota Supercomputing Institute.

[15] Perez-Gonzalez, H.G. and J.K. Kalita. *Automatically Generating Object Models from Natural Language Analysis*. in *Conference on Object-Oriented Programming Systems Languages and Applications*. 2002. Seattle, Washington.

[16] Rashid, A., A. Moreira, and J. Araújo. *Modularisation and Composition of Aspectual Requirements*. in *Proceedings of the 2nd International Conference on Aspect-Oriented Software Development (AOSD 2003)*. 2003. Boston, Massachusetts.

[17] Rashid, A., et al. *Early Aspects: A Model for Aspect-Oriented Requirements Engineering*. in *Proceedings of IEEE Joint International Conference on Requirements Engineering (RE)*. 2002: IEEE Computer Society.

[18] Rayson, P., *Matrix: a Statistical Method and Software Tool for Linguistic Analysis Through Corpus Comparison*. 2002, Lancaster University.

[19] Rayson, P., R. Garside, and P. Sawyer, *Language Engineering for the Recovery of Requirements from Legacy Documents (REVERE Project Report)*. 1999, Lancaster University.

[20] Rayson, P., R. Garside, and P. Sawyer. *Recovering Legacy Requirements*. in *Proceedings of REFSQ'99. Fifth International Workshop on Requirements Engineering: Foundations of Software Quality*. 1999. Heidelberg, Germany: University of Namur.

[21] Rayson, P., R. Garside, and P. Sawyer, *Assisting Requirements Recovery from Legacy Documents*, in *Systems Engineering for Business Process Change: collected papers from the EPSRC research programme*, P. Henderson, Editor. 2000, Springer-Verlag: London, UK. p. 251 - 263.

[22] Rosenhainer, L., *Identifying Crosscutting Concerns in Requirements Specifications*, in *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with OOPSLA 2004 Conference*. 2004: Vancouver, Canada.

[23] Shepherd, D., L. Pollock, and T. Tourwé. *Using Language Clues to Discover Crosscutting Concerns*. in *Proceedings of the 2005 International Workshop on Modeling and Analysis of Concerns (MACS 2005), co-located with International Conference on Software Engineering (ICSE 2005)*. 2005. St. Louis, Missouri.

[24] Shepherd, D., L. Pollock, and K. Vijay-Shanker. *Towards Supporting On-Demand Virtual Remodularization Using Program Graphs*. in *Proceedings of the 5th International Conference on Aspect Oriented Software Development (AOSD 2006)*. 2006. Bonn, Germany.