ANÁLISIS Y APLICACIÓN DE TÉCNICAS PARA IDENTIFICAR ASPECTOS EN REQUERIMIENTOS

Facultad de Ciencias Exactas
tandil ~ argentina

Tesis entregada para el grado de
Ingeniero de Sistemas
en la Facultad de Ciencias Exactas

Por
Alejandro Rago

Bajo la supervisión de la
Dra. Claudia Marcos

Universidad Nacional del Centro de la Provincia de Buenos Aires
Tandil, Argentina
Mayo 2008

Agradecimientos

Resumen

Breve descripción de los objetivos, más una breve descripción de lo que es realizado en este trabajo.

Una explicación concisa de las conclusiones a las que se llegaron.

Índice de contenido

Índice de ilustraciones

Índice de tablas

Índice de código fuente

Capitulo I - Introducción

La identificación, modularización, representación y composición de concerns crosscutting es el tema de investigación sobre el cual esta centrado el paradigma de aspectos. La realización de estas tareas en etapas tempranas como al nivel de requerimientos o arquitectura es imperativa debido a que las propiedades de estos concerns tienen efectos de amplio alcance sobre otros componentes [4]. 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 su efecto sobre el sistema o sobre otros concerns. Además, la falta de la modularización de dichas propiedades 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.

Este trabajo presenta un análisis de los enfoques presentados en varias publicaciones, centrando la atención en aquellos que soportan la identificación semi-automatizada de aspectos en especificaciones de requerimientos tanto de sistemas nuevos como legados, y los compara utilizando un grupo de criterios establecidos con el propósito de criticar las características de cada uno de los enfoques existentes con respecto a los demás. Con la adquisición de este conocimiento, se propondrá una técnica de identificación que solucione los problemas observados, utilizando las mejores estrategias conocidas, para mejorar la técnica ya existente de una herramienta de detección semi-automática de aspectos en especificaciones de requerimientos.

Comentario [OSR1]: Ampliar

I.1 - Motivación

Ha habido un trabajo significante en las comunidades de ingeniería de requerimientos y de diseño arquitectónico (diferentes workshops realizados en las conferencias ECOOP [5] 1996, 2000, 2001; OOPSLA [6] 1999, 2000; ICSE [7] 2001, etc.) sobre la separación de concerns (por ejemplo, viewpoints, casos de uso, goals y modelos de análisis de trade-offs arquitectónicos). Sin embargo, estos enfoques no puntualizan explícitamente en los concerns crosscutting. El trabajo sobre aspectos tempranos, por lo tanto, complementa estos enfoques proporcionando una manera sistemática de manejar dichos concerns.

Recientemente un gran esfuerzo de investigación esta siendo llevado a cabo en el área de Desarrollo de Software Orientado a Aspectos [2, 3] para identificar y modelar aspectos desde las etapas más tempranas del ciclo de vida. Estos trabajos han progresado con el tiempo y se han publicado resultados interesantes y prometedores. Sin embargo, los enfoques no han sido del todo integrados al desarrollo de software debido a que estos todavía tienen algunos problemas.

Debido a que los trabajos realizados por los investigadores con respecto no han llegado a cumplir las expectativas de la comunidad, es de gran interés analizar las razones por las que estos no han logrado el éxito completo, de forma tal de poder realizar una propuesta que los resuelva.

Comentario [OSR2]: Mejorar y ampliar

I.2 - Objetivos

Se llevará a cabo una comparación con el fin de identificar, dentro de las propuestas existentes, los puntos fuertes y débiles que tengan, exponer las diferencias entre ellas, discernir en cuanto sus características se solapan o complementan entre sí, y finalmente mostrar los problemas inherentes de cada uno.

De esta manera se podrá proponer una técnica de identificación de aspectos, que mediante un consenso de las principales propiedades necesarias, solucione la mayoría de los problemas expuestos y concentre las mejores ideas. Esto permitirá mejorar la técnica utilizada por la herramienta Aspect Extractor Tool, que tiene muchas falencias debido a su simpleza. Entre estas se pueden nombrar que necesita que se ingrese información previa, tiene problemas con los sinónimos y las ambigüedades del lenguaje natural, no tiene ningún tipo de filtro para manejar la escalabilidad, y principalmente, que es muy poco efectiva.

Posteriormente se llevara a cabo una evaluación de la nueva técnica, aplicándola sobre un caso de estudio de tamaño pequeño, de forma tal poder razonar acerca de las ventajas y desventajas de su utilización en contraste a la técnica antigua.

Comentario [OSR3]: Ampliar .

I.3 - Organización del trabajo

En el Capitulo II se presentará una explicación los conceptos claves del Desarrollo de Sistemas Orientado a Aspectos, enfatizando principalmente en los Aspectos Tempranos y en la Minería de Aspectos. Adicionalmente se explicarán conceptos que serán necesarios durante el desarrollo de la técnica propuesta.

En el Capitulo III se llevará a cabo el análisis y la comparación de los trabajos de investigación actuales en el campo de la identificación semi-automática de aspectos en especificaciones de requerimientos.

En el Capitulo IV se desarrolla una nueva propuesta que soluciona los problemas encontrados en el análisis y comparación de los enfoques existentes, y se presentará la integración de esta técnica con la herramienta Aspect Extractor Tool.

En el Capitulo V se planteará un caso de estudio, el cual será analizado por la herramienta utilizando tanto la técnica nueva como la antigua. De los resultados obtenidos se obtendrán ventajas y desventajas de cada una de ellas.

En el Capitulo VI se expondrán el cierre del trabajo desarrollando las conclusiones del trabajo e indicando el rumbo futuro de la identificación de aspectos desde etapas tempranas.

Capitulo II - 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 (frecuentemente expresados desde distintos puntos de vista y siendo a menudo conflictivos entre sí) deben ser consensuados para encontrar una solución respetando los diversos intereses.

Comentario [OSR4]: Hacer resumen, y pasar esto a concerns.

II.1 - Orientación a aspectos

Se define un *concern* como "cualquier asunto de interés en un sistema de software" [45], por lo tanto, el desarrollo de software tiene que tratar con un gran número de concerns. Algunos de estos están relacionados al producto (el software) que debe ser creado, como la funcionalidad y la performance. Otros concerns están relacionados con el proceso de desarrollo en sí mismo, como los tiempos y costos del desarrollo.

La *SoC* (Separación de Concerns, Separation of Concerns) es un principio básico de la ingeniería de software [45]. Derivada del sentido común, esencialmente significa que para tratar satisfactoriamente problemas complejos, la única manera posible es dividiendo la complejidad en sub-problemas que pueden ser manejados y resueltos separadamente unos de otros. Las soluciones parciales pueden entonces ser combinadas en una solución completa. Un uso específico del principio de separación de concerns es la modularización: cada modulo es responsable por algunos concerns de un sistema de software, y todos los módulos juntos satisfacen todos los concerns del sistema de software.

Mientras que (por medio de técnicas convencionales de modularización u orientación a objetos) un tipo de concern puede ser fácilmente encapsulado dentro de artefactos como módulos, clases y operaciones a nivel de diseño o implementación, esto mismo no es posible para otro tipo de concern. Estos cortan transversalmente (*crosscut*) el diseño o implementación de unos cuantos o incluso muchos artefactos y por lo tanto son llamados *crosscutting concerns* [45]. 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. Primero, su diseño o implementación es dispersada (*scattered*) sobre varios artefactos en vez de ser localizada en uno solo (es llamado "el problema de *scattering*"). Y segundo, cuando un artefacto frecuentemente mezcla el diseño o implementación de más de un concern en vez de separarlos (es conocido como "el problema de *tangling*"). Los problemas de scattering y tangling generalmente ocurren juntos, aunque son dos conceptos diferentes [2]. Estos dos problemas tienen un número bien conocido de efectos negativos sobre el software afectado por ellos.

En consecuencia a este conjunto de dificultades, surge el *Desarrollo de Software Orientado a Aspectos* (Aspect Oriented-Software Development, AOSD) [45], 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. AOSD es entonces un enfoque para expandir el principio de SoC en el desarrollo de software.

II.2 - Programación orientada a aspectos

La *Programación Orientada a Aspectos* (Aspect-Oriented Programming, AOP) es un paradigma de programación cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos [10]. Gracias a la AOP se pueden encapsular los diferentes conceptos que componen una aplicación en entidades bien definidas, eliminando las dependencias entre cada uno de los módulos. De esta forma se consigue razonar mejor sobre los conceptos, se elimina la dispersión del código y las implementaciones resultan más comprensibles, adaptables y reusables.

El principal objetivo de la AOP es la separación de las funcionalidades dentro del sistema: por un lado funcionalidades comunes utilizadas a lo largo de la aplicación, y por otro lado, las funcionalidades propias de cada módulo. Cada funcionalidad común se encapsulará en una entidad.

II.3 - Aspectos tempranos

Sin embargo, 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 el orientación a aspectos desde el principio del ciclo de vida del software. De esta conclusión surge el concepto y definición de "*Early Aspects*" [2], la cual significa que es importante considerar a los aspectos tempranamente en el ciclo de vida de la ingeniería de software durante el análisis y el diseño (conceptual y detallado), en contraste a que se haga solamente en las etapas de implementación y testeo.

En el contexto de la Ingeniería de Requerimientos (*Requirements Engineering*, RE) los concerns de interés central son los requerimientos. Un requerimiento es un tipo especial de concern. Una especificación de requerimientos bien escrita está caracterizada por el hecho de que cada declaración de requerimientos no mezcla varios requerimientos sino que representa exactamente un requerimiento, por ejemplo, solamente un concern. Además, una buena trazabilidad (siendo este un factor de calidad crítico) es esperada de una especificación de requerimientos bien escrita. Entre otras, es una precondición importante para hacer explicitas las muchas relaciones semánticas que normalmente existen entre diferentes requerimientos. Un tipo especial de relación entre requerimientos es aquella que puede ser calificada como crosscutting. Los requerimientos que cortan transversalmente otros son denominados *requerimientos crosscutting* (crosscutting requirements) [45]. Asimismo, la expresión *influencia crosscutting* (crosscutting influence) [45] es usada como sinónimo para la relación entre dos requerimientos la cual es establecida por uno de ellos atravesando al otro. Una característica de las influencias crosscutting es que ellas pueden ser obvias, pero no necesariamente, son obvias si existen claras referencias hacia otros requerimientos y por lo tanto hacen explicitas las interdependencias respectivas. Las influencias crosscutting son más sutiles si no existen estas referencias claras, y por consiguiente no es fácil concluir por la lectura si un requerimiento es crosscutting o no, y si es sabido que lo es, es difícil determinar en que otros requerimientos tiene influencia. En estos casos las influencias crosscutting no son (o no explicitadamente) documentadas al nivel de los requerimientos.

Idealmente, una especificación de requerimientos documentaría todas las influencias crosscutting como también todas las otras relaciones semánticas entre requerimientos. La realidad, sin embargo, es bastante diferente, incluso si cada declaración de requerimientos se encarga de solo un concern y también es trazable, en la mayoría de los casos no todas las relaciones semánticas entre requerimientos son identificadas y explícitamente registradas. La razón principal de que esto es el hecho que normalmente no hay suficiente tiempo en un proyecto de software para transformar una especificación de requerimientos sub-optima en una optima caracterizada por una trazabilidad completa. Como mejor caso, los desarrolladores detectan esas influencias crosscutting tardíamente en las etapas de desarrollo y les permite reaccionar en consecuencia. En el peor caso, algunas influencias continúan ocultas hasta después de la entrega del sistema de software y son descubiertas indirectamente por los usuarios, porque algunos requerimientos no están totalmente satisfechos por el software. En ambos casos el resultado de las deficiencias de los requerimientos será el incremento de los costos (cuanto más tarde se detecten, más costoso será). El segundo caso, puede además causar que surjan otros problemas como la perdida de credibilidad de los desarrolladores o incluso la falla de componentes críticos del sistema.

II.4 - Ingeniería de requerimientos orientada a aspectos

La identificación de aspectos en etapas tempranas, según Sampaio, Rashid y Rayson en [46, 48], ayuda a lograr la SoC en el análisis inicial del sistema, en vez de que estas decisiones sean diferidas a etapas más tardías de diseño o implementación, y por lo tanto, teniendo que realizar

refactorizaciones costosas. Además, también mejora la evolución del software y reduce el tiempo de puesta en venta del software al mercado (*time to market*).

Para Rosenhainer en [45], la importancia de la identificación y documentación de requerimientos crosscutting y sus influencias son:

- Impacto de cambios: Al mantener una trazabilidad completa de los requerimientos, y tener documentado los requerimientos crosscutting y sus influencias, es posible que al ocurrir la necesidad de analizar cambios, se pueda fácilmente evaluar el impacto de estos en el sistema.
- Evolución de requerimientos: La capacidad de poder evaluar el impacto de cambios entre los requerimientos de una manera eficiente es un factor vital para la evolución de los requerimientos, por lo tanto, la documentación comprensiva de los requerimientos crosscutting y sus influencias también permite una simplificación de la evolución de requerimientos.
- Evitar pasar por alto influencias crosscutting: Si se realiza tempranamente, la identificación y documentación de los requerimientos crosscutting y sus impactos en otros requerimientos trae otra ventaja importante: que estos no sean olvidados en las fases siguientes. Siempre que se documenten las influencias crosscutting en algún requerimiento, será más fácil que estos sean tomados en consideración cuando se realice el diseño, la codificación, o una tarea de mantenimiento. Las influencias crosscutting al ser documentadas también permiten que se deriven los correspondientes casos de test, de los requerimientos influenciados o de los mismos requerimientos crosscutting, manualmente, semi-automáticamente o incluso automáticamente (con un grado suficiente de formalismo).

Es concebible identificar y documentar los requerimientos crosscutting junto con sus influencias en varias etapas del ciclo de vida del proceso de desarrollo y mantenimiento:

- Mientras se modelan los requerimientos: Luego de que se han elicitado los requerimientos, estos deben ser modelados. El modelaje es una actividad altamente analítica cuyos objetivos son por ejemplo estructurar los requerimientos, identificar y plasmar las dependencias y otras relaciones entre estos, e identificar y clarificar las ambigüedades y vaguezas. También sería natural y altamente deseable que en esta etapa se identifiquen y modelen los requerimientos crosscutting y sus influencias.
- Mientras se escriben las especificaciones de los requerimientos: La escritura de las especificaciones normalmente ocurre inmediatamente luego del modelaje de requerimientos. Sin embargo, frecuentemente una especificación es escrita directamente luego que los requerimientos han sido elicitados, salteando la etapa del modelado. En este caso la escritura de la especificación de los requerimientos también es un proceso altamente analítico. Obviamente, en este punto se debería entonces incluir la identificación y documentación de requerimientos crosscutting y sus influencias.
- Después de que se escriben las especificaciones de los requerimientos: Aunque es mejor identificar y documentar los requerimientos crosscutting y sus influencias cuando estos surgen y son descubiertos, se tiene que aceptar el hecho de que existen innumerables especificaciones de requerimientos escritas sin tener en mente los requerimientos crosscutting. Sin embargo, a posteriori puede ser deseable para los desarrolladores que estos sean identificados en los documentos. Entonces se convierte necesario que se puedan identificar y extraer los requerimientos crosscutting y sus influencias de los documentos existentes.
- Antes de que se diseñe el sistema: Alternativamente, antes de que comience el proceso de diseño del sistema comience, los desarrolladores podrían buscar en los requerimientos menciones de comportamientos típicamente aspectuales, como logging, trazabilidad o funcionalidad de depuración. Sin embargo, esto cubre solo un margen estrecho de los aspectos potenciales; no ayuda a identificar las influencias crosscutting que no están en alguna de esas categorías, o que pueden ser especificas del dominio.

- Durante las actividades de implementación: Por la naturaleza crosscutting de algunos requerimientos y sus influencias podrán y serán detectados durante las últimas actividades del desarrollo o el mantenimiento y deberán en ese punto ser documentadas. La forma común de realizarlo es primero mirar detalladamente el código fuente del sistema, y luego intentar descubrir el comportamiento enmarañado (tangled) y disperso (scattered) mientras este se hace visible. Este es un enfoque ad-hoc en el cual probablemente sea necesario rediseñar el sistema a medida que los aspectos son descubiertos.

Para identificar o modelar un amplio rango de aspectos tempranamente en el ciclo de vida, y evaluar en que puntos cortan transversalmente el sistema [17], los desarrolladores necesitan un soporte para el análisis de las relaciones entre todos los comportamientos descriptos en la documentación de requerimientos. También se necesita soportar la traslación de los resultados del análisis en modelos de diseño que puedan ser implementados.

La *Ingeniería de Requerimientos Orientada a Aspectos* (Aspect-Oriented Requirements Engineering, AORE) ha surgido como una nueva manera de modularizar y razonar acerca de los crosscutting concerns durante la ingeniería de requerimientos [23]. La AORE extiende la noción de SoC en la RE (por ejemplo viewpoints, casos de uso, objetivos, etc.) con aquellos aspectos al nivel de requerimientos. Estos aspectos modularizan los requerimientos que afectan y restringen otros requerimientos. Explicitando la modularización de los crosscutting concerns en el nivel de los requerimientos, la AORE posibilita razonar respecto a estos concerns tempranamente en el ciclo de vida del software.

No es posible esperar que los requerimientos que son crosscutting sean identificados tempranamente siempre [45]. Por ejemplo, durante la etapa de análisis de requerimientos, es conveniente utilizar un enfoque para el problema de la identificación desde un punto de vista de minería de aspectos, como buscar los requerimientos crosscutting en las especificaciones de requerimientos ya existentes.

II.5 - Mineria de aspectos

La *minería de aspectos* (Aspect Mining, AM) [18, 21, 30, 32-34, 45, 46, 54, 55] es un término usado para la identificación de aspectos (la cual encapsula los concerns crosscutting) en algunos requerimientos pre-existentes, documentos de diseño, código u otros artefactos de desarrollo. Los aspectos no están visibles fácilmente, por lo que deben ser "minados" de estos. Cabe destacar que generalmente en la literatura se utiliza el término minería de aspectos principalmente para referirse a la extracción de aspectos del código fuente de un sistema; sin embargo, se considera que este concepto es más amplio y puede involucrar a cualquier artefacto producido durante el desarrollo. Por ejemplo, los artefactos involucrados para la identificación de aspectos tempranos son las especificaciones de requerimientos.

La minería de aspectos de las especificaciones de requerimientos [45] es razonable debido a que la especificación de requerimientos se realiza sin considerar los requerimientos crosscutting. Además, parece ser realista no esperar que la orientación a aspectos en la ingeniería de requerimientos se convierta en un enfoque aceptado ampliamente en un futuro próximo, de tal manera que sea aplicado desde las etapas tempranas de los proyectos de software. Por esas razones, la minería de aspectos es necesaria y será más necesitada para permitir encontrar y documentar crosscutting concerns ocultos en las especificaciones existentes, y por lo tanto mejorar su calidad y utilidad.

Para el propósito de minar requerimientos crosscutting y sus influencias de las especificaciones de requerimientos, existen dos técnicas concebibles, siendo la primera manual y la segunda semi-automática:

- Identificación mediante inspecciones: Después que se ha escrito una especificación de requerimientos (o una parte de ella) esta debe ser inspeccionada para identificar y documentar cualquier tipo de deficiencias. Por lo que seria sencillo leer las especificaciones con el objetivo particular en mente de identificar requerimientos crosscutting y sus influencias. Generalmente para realizarse, es recomendable empezar desde un *requerimiento no funcional*

(Non-Functional Requirement, NFR) dado, y luego buscarlo en el texto de las especificaciones.

- Identificación mediante técnicas de recuperación de la información: También es posible soportar la identificación de requerimientos crosscutting y sus influencias mediante técnicas de *recuperación de información* (Information Retrieval, IR) [26] Un programa puede por ejemplo encontrar sentencias que parecen ser afectadas por un requerimiento que el analista considera ser crosscutting. De manera similar a lo anterior, el analista debe examinar cada sentencia y documento cuando se encuentra una influencia crosscutting.

La identificación de requerimientos aspectuales es sin embargo una tarea no trivial [23]. Llevar a cabo las tareas de identificación manualmente normalmente es muy costoso y consume mucho tiempo. Por lo tanto, como otras técnicas de RE, la AORE requiere el soporte efectivo y escalable de una herramienta (que utilice técnicas de IR) para aprovechar completamente sus beneficios cuando se analizan problemas de gran escala.

Las técnicas basadas en IR son especialmente prometedoras cuando existen grandes volúmenes de documentos [45]. Podrían ser usadas independientemente si el tiempo es demasiado corto para realizar una inspección completa, o en combinación con una inspección para detectar más influencias crosscutting no identificadas hasta el momento. Por un lado, inspeccionar una especificación substancial es un trabajo arduo donde muchos detalles de interés podrían ser pasados por alto debido a la falta de tiempo, fatiga, o monotonía. Por otra parte, comparado con un enfoque de IR, las inspecciones tienen sus principales fuertes en el trato de las peculiaridades del lenguaje natural como las ambigüedades y vaguezas[1], y sería por lo tanto mucho más fácil de lograr la completitud (detectar todas las influencias sobre los requerimientos). Sin embargo, aunque la completitud es altamente deseable, incluso la inspección no puede garantizar completamente que se satisfaga la completitud en alguna etapa, debido a que el enfoque esta basado en el juicio subjetivo y retroactivo del analista. Ambas técnicas también son aptas para la realización de un enfoque ágil para identificar influencias crosscutting (por ejemplo buscar influencias crosscutting solo en el momento que sea necesario por alguna tarea de desarrollo o mantenimiento).

II.6 - Desambiguación del sentido de las palabras

Debido a que las técnicas de IR sufren los problemas de ambigüedades y vaguezas propias de los lenguajes naturales, se han llevado a cabo investigaciones sobre el tema que según la literatura es denominado como *desambiguación del sentido de la palabra* (Word Sense Disambiguation, WSD). Estas investigaciones centran sus metas en desarrollar una técnica que realice el proceso de asignar un significado a una palabra particular basándose en el contexto en el cual esta ocurre [35].

Frecuentemente el conjunto de posibles significados para una palabra es conocido antes de tiempo, y es determinado por el inventario de significados de un *diccionario de maquina leíble* (Machine Readable Dictionary) o por una base de datos léxica. Cuando la WSD es planteada como el problema de seleccionar un sentido de un inventario existente, existen por lo menos dos metodologías que pueden ser aplicadas. Una opción es el *aprendizaje supervisado* (Supervised Learning), en donde un sistema es entrenado con ejemplos creados manualmente para desambiguar palabras correctamente en su contexto. La otra opción son los *enfoques basados en diccionarios* (Dictionary Based Approaches), los cuales tratan un diccionario o una fuente similar tanto como fuente del inventario de

[1] Una frase ambigua es la que tiene dos o más significados diferentes, pero cada uno de los cuales es preciso. Por su parte, una frase vaga es la que no tiene ningún significado preciso. Las frases ambiguas deben evitarse cuando hay riesgo de interpretación incorrecta, mientras que las frases vagas son necesarias si intentamos expresar un dato, pensamiento o estado de ánimo vagos. Hay varios tipos de ambigüedades, y se clasifican en ambigüedad referencial (cuando una palabra o frase puede referirse a dos o más propiedades u objetos) y ambigüedad gramatical (cuando la estructura de una frase permite dos o más interpretaciones).

1. *Retórica y composición hispánicas*. 1999 [cited; Available from: http://www.acs.ucalgary.ca/~gimenez/499_06_Fichas.htm..

sentidos como también de repositorio de información acerca de las palabras que pueden ser explotadas para distinguir los significados en el texto.

La presunción subyacente de estos métodos es que las palabras que ocurren juntas en una sentencia deben estar relacionadas en algún grado. Lo que queda incierto es cómo medir mejor la relación semántica, y qué métricas serán más eficaces para llevar a cabo la desambiguación.

El objetivo de un algoritmo de WSD consiste en asignar una palabra w_i que ocurre en un documento *d* con su significado apropiado o sentido *s*, mediante la exploración del contexto *C* en el cual w_i fue encontrada [19]. El contexto *C* para w_i es definido como un conjunto de palabras que se encuentran antes y después de w_i. El sentido s es seleccionado de un conjunto predefinido de posibilidades, usualmente conocido como *inventario de sentidos* (Sense Inventory, SI).

II.7 - WordNet

Como anteriormente se mencionaron los *inventarios de sentidos*, a continuación se explica uno en particular, llamado WordNet. Es una base de datos léxica que contiene información acerca de sustantivos, verbos, adjetivos y adverbios [12, 13, 35]. Esta organiza los conceptos relacionados en *conjuntos de sinónimos* (Synonym Set, Synsets) Cada synset puede considerarse como la representación de un concepto o *sentido* (Sense). Por ejemplo: {*car, auto, automobile, machine, motorcar*}

es un synset que representa el sense definido por la *definición* (Gloss): "*4-wheeled motor vehicle; usually propelled by an internal combustión engine*". De hecho cada synset representa un concepto o sense de la palabra, y se puede usar estos términos de manera indiferente en una frase (siempre y cuando se intente usar con este sense). Además de proveer estos grupos de sinónimos para representar un concepto, WordNet conecta los conceptos mediante una variedad de relaciones. Esto crea una red donde los conceptos relacionados pueden ser identificados mediante el cálculo de una distancia relativa entre ambos. Las relaciones provistas incluyen la *sinonimia* (synonymy), *antinomia* (antonymy), *es un* (is-a) y *parte de* (part-of). Las relaciones en WordNet generalmente no cruzan los limites impuestos por las partes del discurso, por lo que las relaciones semánticas y léxicas tienden a ser solamente entre conceptos de la misma parte del discurso.

Para los sustantivos la relación más útil y común es la is-a. Esta existe entre dos conceptos cuando un conceptos *es un tipo de* (is-a-kind-of) otro concepto. Esto es también conocido como hipernomia (hypernym). Por ejemplo, *car* es un hipenomio de motor *vehicle*. También existe una jerarquía is-a para los verbos, aunque representa una *es forma de hacer* (is-way-of-doing), también conocido como *troponomia* (troponomy). Por ejemplo, *walking* es un troponimo de *moving*. Cada jerarquía (sea de sustantivos o de verbos) puede ser visualizada como un árbol que tiene un concepto muy general asociado al nodo raíz y conceptos más específicos asociados a las hojas.

II.8 - Patrones básicos de las sentencias en ingles

Generalmente, una buena aproximación para identificar el comportamiento potencialmente crosscutting, es identificar las acciones que se especifican en un documento de requerimientos. Estas acciones son verbos que se encuentran en el texto de las especificaciones. Sin embargo, debido a que un verbo dado puede aplicarse a más de un objeto en distintos lugares de los documentos y la acción que representa puede ser diferente en cada caso según el objeto afectado por el verbo, resulta necesario realizar la identificación de pares de verbos más los objetos directo que afectan, ya que permiten una mayor precisión y detalle sobre este comportamiento. Entonces es de utilidad identificar tanto el verbo como el objeto directo el cual es afectado por el verbo, en forma de pares (verbo - objeto directo). Para realizar esto, se debe analizar sintacticamente las frases, y mediante patrones identificar los pares.

De la misma manera que hay formas en geometría y en química, también hay fórmulas en la gramática [56]. Los lingüistas, personas que estudian el lenguaje, han encontrado patrones, o fórmulas, que arman el esqueleto estructural de casi todas las sentencias en el lenguaje inglés. El entendimiento de la estructura de la sentencias permite mejorar la habilidad de entender la gramática,

y poder identificar estructuras sintácticas en las sentencias. Para llevar a cabo esta tarea, las sentencias se clasificaron en diez patrones.

El primer paso para aprender los patrones es recordar las dos partes básicas de una sentencia, el sujeto y el predicado. El sujeto es usualmente el *quién* o el *qué* de la sentencia. Lo que se dice del sujeto es el predicado. Dos términos que deben recordarse son las frases sustantivas y las frases verbales. Las frases sustantivas actúan como sujeto, y las frases verbales como predicado. Otro factor importante es saber que el verbo es la parte central en la sentencia. Las variaciones en los patrones son en el predicado (o frase verbal). Por lo tanto se puede identificar con qué patrón se corresponde una sentencia solamente observando su frase verbal. Los diez patrones están divididos en cuatro grupos: los patrones "be", los patrones "linking verb", el patrón del "intransitive verb", y los patrones de "transitive verb".

- Patrones I al III (Patrones *Be*)

  El número de ranuras[2] en el predicado es dos. La primera contiene el verbo principal, o predicativo, que es una de las formas del verbo *be*. Algunos ejemplos son *is*, *am*, *are*, *was*, *were*, *being*, y *been*. Las formas expandidas son *have been*, *was being*, *might be* y *will be*. Lo que sigue al verbo principal en la segunda ranura determina que cual de los tres patrones es. En el patron I es un advervio de tiempo o lugar, en el II un adjetivo, y en el II una frase sustantiva.

- Patrones IV al V (Patrones *Linking verb*)

  Los patrones IV y V contienen dos ranuras en el predicado como los anteriores. Estos patrones contienen un verbo que actúa de nexo seguido por un sujeto complementario. La diferencia entre los dos es con qué se rellena la ranura del sujeto complementario. En el patrón IV se llena con un adjetivo, mientras que en el V esto se hace con una frase sustantiva.

  Los verbos que aparecen comúnmente en el patrón IV están relacionados con los sentidos, como *taste*, *smell*, *feel*, *sound* y *look*. Otros pueden llegar a ser *turn*, *appear*, *become*, *get*, *remain* y *prove*. Algunos verbos como *become*, *remain* y *seem* también son usados en el patrón V.

- Patrón VI (Patrón *intransitive verb*)

  En este patrón no hay complemento ni frase sustantiva o adjetiva que siga al verbo; sin embargo puede ser que se agregue información adverbial alrededor del verbo.

- Patrones VII al X (Patrones *transitive verb*)

  Los cuatro patrones transitivos tienen una cosa en común, cada uno contiene un objeto directo. El patrón VII es considerado el más básico y contiene solo un objeto directo a continuación del verbo. El patrón VII difiere un poco en que un objeto indirecto precede al objeto directo. En el patrón IX el objeto complemento que sigue al objeto directo es un adjetivo (modifica o describe el objeto directo). El objeto complemento en el patrón X es una frase sustantiva.

En la Tabla II-1 se muestran ejemplos de los diez patrones:

I	NP (Sujeto) *The team*	Verbo be *is*	Adverbio de tiempo o lugar *outside*
II	NP (Sujeto)	Verbo be	Adjetivo

[2] Se llama ranura a la unidad resultante de la división de un predicado o sujeto en partes. Es decir que representan un lugar donde se puede ubicar una palabra.

	The team	*is*	*hard working*	
III	NP (1) (Sujeto) *That team*	Verbo be *is*	NP (1) (Comp. del sujeto) *The Raiders*	
IV	NP (Sujeto) *The child*	Verbo nexo *seems*	Adjetivo *honest*	
V	NP (1) (Sujeto) *The children*	Verbo nexo *became*	NP (1) (Comp. del sujeto) *foster kids*	
VI	NP (Sujeto) *The club members*	Verbo intransitivo *arrived*		
VII	NP (1) (Sujeto) *The woman*	Verbo transitivo *passed*	NP (2) (Obj. Dir.) *the test*	
VIII	NP (1) (Sujeto) *The players*	Verbo transitivo *gave*	NP (2) (Obj. In.) *the other team*	NP (3) (Obj. Dir.) *the ball*
IX	NP (1) (Sujeto) *The members*	Verbo transitivo *find*	NP (2) (Obj. Dir.) *the club*	Adjetivo (Comp. del objeto) *interesting*
X	NP (1) (Sujeto) *She*	Verbo transitivo *considers*	NP (2) (Obj. Dir.) *her teacher*	NP (1) (Comp. del objeto) *the best*

Tabla II-1 - Patrones sintácticos del idioma ingles

Capitulo III - Técnicas de identificación de aspectos

En este capitulo se realizará una comparación de técnicas (o enfoques) existentes para interiorizar y comprender las mejores ideas exploradas por los investigadores, y luego poder componer aquellas que se complementen en una sola técnica, que resuelva los problemas de las individuales. Para esto, se tendrán en consideración las características de cada una de ellas, las cuales luego de la determinación de los criterios adecuados para el análisis, serán enfrentadas para mostrar las ventajas y desventajas de cada una.

Cabe destacar que si bien se tuvieron en consideración muchas técnicas y enfoques de identificación de crosscutting concerns propuestos y analizados por investigadores, muchas de estos fueron descartados a la hora de ser seleccionados para realizar la comparación, debido a que unas cuantas solamente proponían un modelo o proceso para la identificación de aspectos [14, 31, 33, 53], pero no hacían hincapié en la detección semi-automática de estos, o bien estaban pensados para que la minería se realice sobre código fuente de un sistema legado [18, 21, 29, 30, 32, 34, 49, 54, 55], o también presentaban enfoques desde puntos de vista arquitectónicos [20]. En la selección solo se tuvieron en cuenta aquellos enfoques que proponían un sistema semi-automatizado de detección de aspecto que utilizan técnicas para minar los concerns del texto de los documentos de requerimientos.

III.1 - Enfoques y técnicas existentes

Dentro de las técnicas analizadas, solamente cinco de estas terminaron siendo consideradas para realizar la comparación. Esto, se debe en gran medida a que los enfoques que asisten al desarrollador en la automatización de la AORE, si bien han conseguido un gran nivel de investigación, desarrollo y evaluación, todavía no han sido adoptados completamente por la comunidad y solo han sido tenido en cuenta por unos pocos grupos de investigadores a nivel mundial. Las técnicas que fueron seleccionadas para llevar a cabo la identificación son:

1. **Theme/Doc** [15-17]

   Es un enfoque que expone las relaciones entre los comportamientos del sistema. Este esta 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 clave y entidades clave, las cuales deben ser provistas por el desarrollador. La idea principal detrás del enfoque es que permite ir refinando las diferentes vistas de requerimientos provistas para que se pueda determinar que temas son crosscutting y cuales no, además de explicitar que lugares se manifiestan.

2. **EA-Miner** [23, 24, 37-44]

   Es un enfoque que permite identificar los aspectos que conciernen 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 principalmente por que no es necesario que el desarrollador tenga que leer previamente las especificaciones para minar y modelar los concerns crosscutting.

3. **IR for Identifying Crosscutting Concerns in Requirements Specifications** [45]

   Es un enfoque que se basa en la aplicación de técnicas de IR 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.

4. **On Demand Virtual Remodularization Using Program Graphs** [50-52]

   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. 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.

5. **Aspect Extractor Tool** [27, 28]

   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, como el stop-words y el stemming. Para determinar cuando se tiene un aspecto candidato, se realiza una búsqueda de comportamiento compartido entre casos de uso.

Para cada una de estos enfoques o técnicas, se realizara una explicación detallada exponiendo sus objetivos, su proceso, las tareas que declara y las características específicas de cada una de ellas. En particular, el enfoque 5 es el cual en el Capitulo IV será extendido mediante el perfeccionamiento del algoritmo de identificación de aspectos.

III.1.1 - Theme/Doc

El enfoque denominado *Theme* [15, 17], provee soporte para el desarrollo orientado a aspectos en dos niveles. Al nivel de los requerimientos, *Theme/Doc* provee vistas del texto de la especificación de requerimientos, exponiendo las relaciones entre los comportamientos en un sistema. Al nivel del diseño, *Theme/UML* permite al desarrollador modelar características y aspectos en un sistema, y especificar cómo estos deben ser combinados. El argumento central del enfoque es que Theme/Doc permite al desarrollador refinar las vistas de requerimientos de forma tal que se revele cuál funcionalidad del sistema es crosscutting y en qué partes del sistema ocurren. También lo asiste manteniendo la trazabilidad desde los requerimientos al diseño, ya que los requerimientos son mapeados directamente de las vistas de Theme/Doc a los modelos de Theme/UML. Esta trazabilidad además provee información acerca de la cobertura de los requerimientos en el diseño.

El enfoque Theme esta basado en las nociones de *temas* (themes). Un tema es un elemento de diseño: una colección de estructuras y comportamiento que representan una característica. Temas múltiples pueden ser combinados o integrados para formar un "todo funcional" de acuerdo a un modelo multidimensional, es decir, formar el sistema en sí. Existen dos tipos de temas: los *temas base*, los cuales pueden compartir con algunas estructuras y comportamientos con otros temas base, y los *temas crosscutting*, los cuales tienen comportamientos que se solapan con la funcionalidad de los temas base. Los temas crosscutting son aspectos.

Theme/Doc provee vistas y soporte funcional para la identificación y representación de la etapa de análisis. Theme/UML permite el modelado estándar UML de la estructura y comportamiento relevante para cada tema en la etapa de diseño. Theme/Doc se apoya en la hipótesis básica de que si dos comportamientos son descriptos en el mismo requerimiento, entonces están relacionados. Los comportamientos se pueden relacionar de tres formas: pueden ser relacionados erróneamente o por coincidencia, significando que el requerimiento podría ser reescrito de manera tal que no estén más acoplados; pueden estar relacionados jerárquicamente, en el que un comportamiento es un sub-comportamiento de otro; o pueden ser relacionados por crosscutting, significando que el requerimiento describe un comportamiento como un aspecto de otro. Usando la herramienta Theme/Doc, un desarrollador puede visualizar las relaciones entre comportamientos descriptos en la documentación de requerimientos, y además puede determinar qué comportamiento es base y cuál es crosscutting (aspecto).

La herramienta toma los documentos de requerimientos como entrada. El comportamiento en los requerimientos es identificado como un conjunto de palabras clave provistas por el desarrollador a la herramienta Theme/Doc. Este comportamiento es referido como *acciones* (actions). Hablando estrictamente, cualquier cadena de caracteres léxica, incluyendo un requerimiento no funcional, estado, etc. puede ser hecho una *acción clave* (key-action) si fue considerado un tema candidato. Sin embargo, está comprobado que usar acciones es un buen punto de comienzo para encontrar temas, y que los requerimientos que no parecen contener acciones pueden generalmente ser refinados para incluir acciones. El desarrollador también debe proveer un conjunto de *entidades clave* (key-entities) como entrada. Las entidades también pueden abarcar recursos. Estas entradas son usadas para generar las vistas de Theme/Doc.

El proceso y las reglas heurísticas que son aplicadas en Theme/Doc pueden ser descompuestas en tres sub-procesos principales: la identificación de acciones y entidades del texto de los requerimientos, la categorización de las acciones en temas, y la identificación de temas crosscutting. Cada sub-proceso incluye reglas heurísticas. La decisión sobre si un tema es crosscutting es definido explícitamente, en donde las reglas para identificar acciones y entidades y la decisión de cuáles acciones son suficientemente importantes para ser un tema son implícitas (es decir, "altamente intuitivas").

Sin entrar en detalle y sintetizando, las herramientas que Theme/Doc provee son vistas que exponen aquellos comportamientos que están localizados en varios requerimientos. La vista de acción mayor es usada para mostrar gráficamente la relación entre los requerimientos y las acciones. La vista de acción recortada es usada para representar los temas crosscutting. Aparte de estas vistas, la vista de tema es usada para planificar el diseño y el modelaje de los temas identificados. La vista de tema

difiere de las vistas de acción en que no sólo muestra requerimientos y acciones, sino que también muestra elementos clave del sistema que deben ser considerados para cada tema de diseño.

III.1.1.1 Vista de acción

Específicamente, para ayudar a la identificación de comportamiento crosscutting se utiliza la *vista de acción* (action view) de los documentos de requerimientos. Son necesarias dos tipos de entrada para crear la vista de acción: la lista de acciones identificadas por el desarrollador mediante la lectura de los documentos de requerimientos seleccionando los verbos sensibles, y los requerimientos tal cual están escritos en el documento original. Entonces Theme/Doc realiza un análisis léxico del texto y genera la vista de acción.

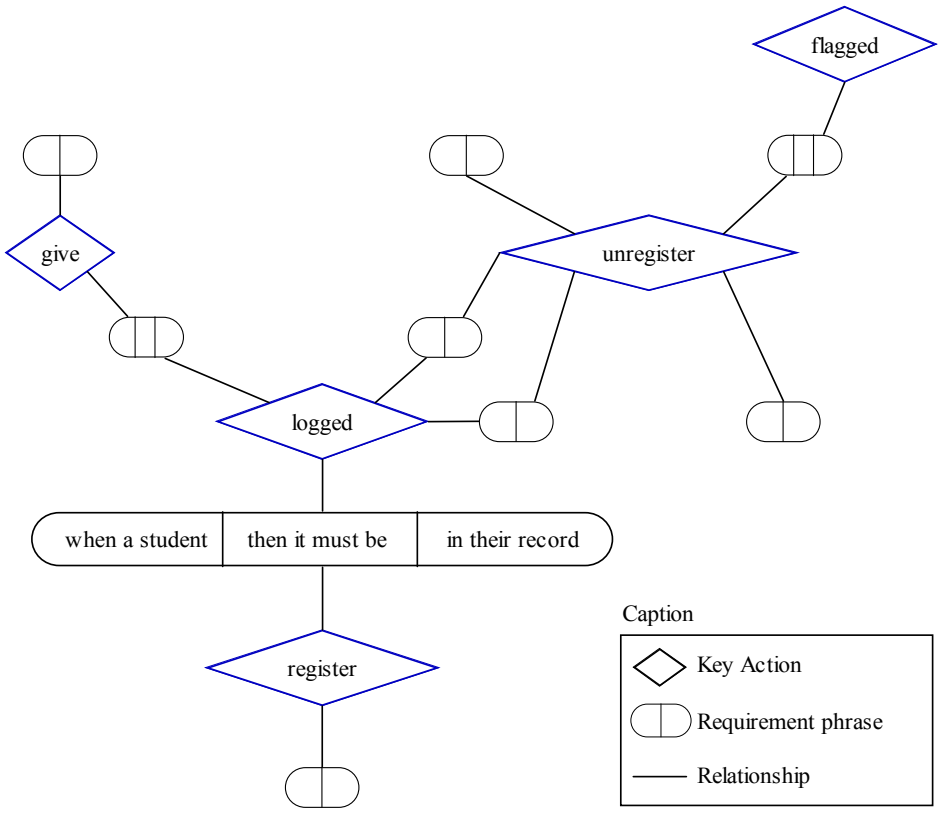

Ilustración III-1 - Vista de acción

En la Ilustración III-1 es posible observar las acciones representadas como diamantes, los requerimientos del texto son mostrados como cajas redondeadas las cuales restringen (son un fragmento) las palabras de la declaración original. La intención de la vista de acción es remarcar las relaciones entre las acciones, por lo que el texto de los requerimientos individuales no es de interés en esta etapa y por eso no se pretende que los fragmentos sean legibles. Cada acción es potencialmente un tema que deberá ser modelado en Theme/UML por separado. Como se dijo anteriormente, los temas permiten el diseño individual de las diferentes características del sistema. En la orientación a objetos, no todos los sustantivos de los documentos de requerimientos son diseñados como objetos o clases. Similarmente, en Theme, no todas las acciones son diseñadas como características individuales del sistema. En este paso, se trata de identificar las características, o temas del sistema, discerniendo las acciones que son lo suficientemente importantes para ser modeladas separadamente, de las que no lo son. Por lo tanto, esta vista es usada para determinar qué acciones deben ser temas, o solo

comportamiento (por ejemplo métodos) dentro de los temas. La decisión de qué acciones no son lo suficientemente importantes para ser temas es un proceso muy intuitivo.

III.1.1.2 Vista de acción principal

También existe la *vista de acción principal* (major action view), la cual ayuda a determinar qué aspectos son base, y cuáles son aspectos. Esta vista esta conformada solo con las acciones principales de las vistas de acción previas. Es casi idéntica a la vista de acción, y solo cambia porque se remueven algunos nodos. El objetivo del uso de esta vista esta en los requerimientos que son compartidos por más de un tema. Si un requerimiento es compartido por dos o más temas, entonces se debe decidir a cuál tema le corresponde proveer esa funcionalidad. Los requerimientos compartidos indican que puede que se haya identificado un aspecto en el sistema, ya que implica que dos temas no pueden operar sin depender el comportamiento de uno con respecto al otro, y viceversa. Es normal que algunos requerimientos estén asociados solamente con un tema. Es simple asumir que cualquiera que sea la funcionalidad debe asociarse con esos temas. Cuando no haya más requerimientos compartidos, se puede avanzar a examinar los temas individualmente, y empezar a planificar el diseño.

III.1.1.3 Vista de recorte de acción

Como los requerimientos frecuentemente se refieren a más que una sola acción, el enfoque tiene como objetivo separar y aislar los grupos de acciones y requerimientos en la vista de acción, obteniendo dos tipos de agrupamientos de acciones/requerimientos. Los del primer tipo son autónomos y no tienen requerimientos que se refieran a acciones en otros grupos, por lo que se denominan base. Los del segundo tipo son crosscutting y tienen requerimientos que refieren a acciones en otros grupos. Se usa la funcionalidad de *recorte* (clipping) de la herramienta para lograr esta separación y aislamiento. Primero se debe examinar cada requerimiento compartido para ver cuál acción esta más debidamente acoplada. Si el requerimiento es demasiado ambigüo para asociarlo con una acción u otra, se debe resolver la ambigüedad, tanto mediante la reescritura, división o refinamiento del requerimiento.

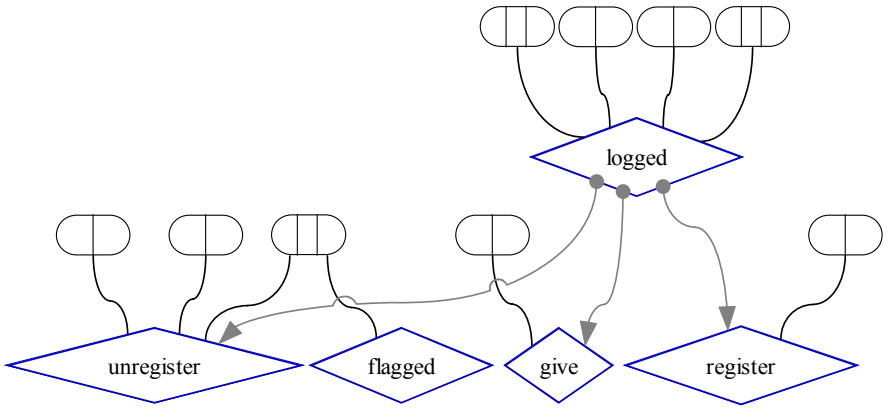

Ilustración III-2 - Vista de recorte acción

En la *vista de recorte de acción* (clipped action view), los temas crosscutting están ubicados sobre los temas que son cortados transversalmente. Como se puede observar en la Ilustración III-2, las flechas grises indicando el crosscutting en la vista recortada ayudarán a guiar la configuración de las relaciones entre aspectos y temas base en la etapa de diseño. Cada agrupamiento se convertirá en un tema que será deseable modelar usando Theme/UML.

III.1.1.4 Vista de tema

Para planificar el diseño y modelaje de los temas identificados en el paso anterior, se utiliza la *vista de tema* (theme view) de Theme/Doc mostrada en la Ilustración III-3. Estas vistas difieren de las vistas de acción en que no solo muestran los requerimientos y las acciones, sino que también muestran elementos claves del sistema que deberán ser considerados para cada tema diseñado en Theme/UML. Para construir una vista de tema, el desarrollador debe proporcionar adicionalmente el conjunto de palabras clave: las *entidades clave* (key entities). Como en la vista de acción, la vista de tema es creada a través de un análisis léxico de los documentos de requerimientos.

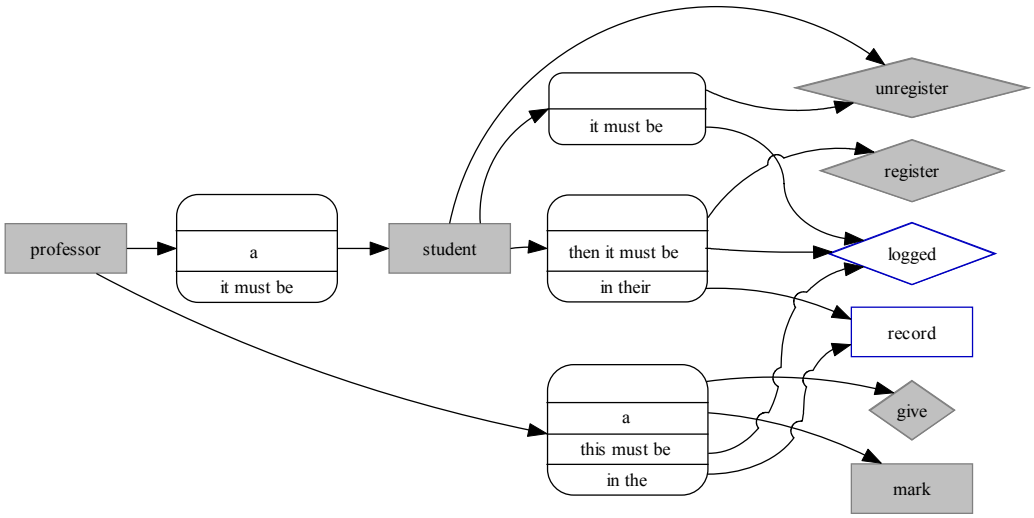

Ilustración III-3 - Vista de tema

Cuando se interpreta la representación de la vista de tema en Theme/Doc, se debe leer el primer elemento que apunta al registro de sentencias (sentence record), luego corresponde leer la primera frase en el registro, y entonces leer el elemento apuntado por esa frase. Si la frase esta vacía, significa que había solo un espacio entre el primer y segundo elemento, sin una frase de conexión entre ambas. Luego se lee la frase de conexión subsiguiente, y luego el elemento al cual apunta. Esto continúa hacia atrás y adelante entre el registro y sus elementos adjuntos hasta que el registro se termina. Para leer una sentencia, no se debe leer más de un elemento del registro de sentencias.

Durante la utilización de Theme/UML para el modelado, se deben usar las buenas prácticas del diseño orientado a objetos para ayudar a determinar qué clases y métodos serán descriptos. Cuando se modela un tema crosscutting puede que sea de utilidad modelar el comportamiento crosscutting en manera abstracta y potencialmente reusable. No se debe explicitar referencias a ninguna acción base o entidades. La vista de tema para temas crosscutting ayuda a identificar esos elementos remarcando de color gris las acciones y entidades que fueron encontradas en otros temas. Las acciones y entidades de color blanco restantes entonces pueden ser usadas para guiar el diseño de comportamientos crosscutting abstractos. Las acciones remarcadas en gris pueden también ser usadas para determinar la unión, o vinculación de comportamiento crosscutting a las acciones base remarcadas también de color gris.

III.1.1.5 Vista aumentada

Luego que los temas han sido diseñados usando Theme/UML, se debe volver a analizar las vistas de tema de Theme/Doc en una forma de asistir a la verificación de que las decisiones de diseño hechas están alineadas con los requerimientos. En la *vista aumentada* de tema (augmented view), la relacion “tiene” (has) es mostrada usando una flecha invertida en el elemento contenedor, apuntando

al elemento contenido. Las relaciones “invoca” (calls) es expresada con una flecha punteada. Se aumentan las vistas agregando métodos a las vistas, las cuales son mostradas como acciones, y también clases, que son mostradas como entidades. Para distinguir estos agregados se marcan las esquinas de los contenedores. Estos agregados son especificados por el usuario e introducidos en la herramienta Theme/Doc, la cual genera la vista aumentada (Ilustración III-4).

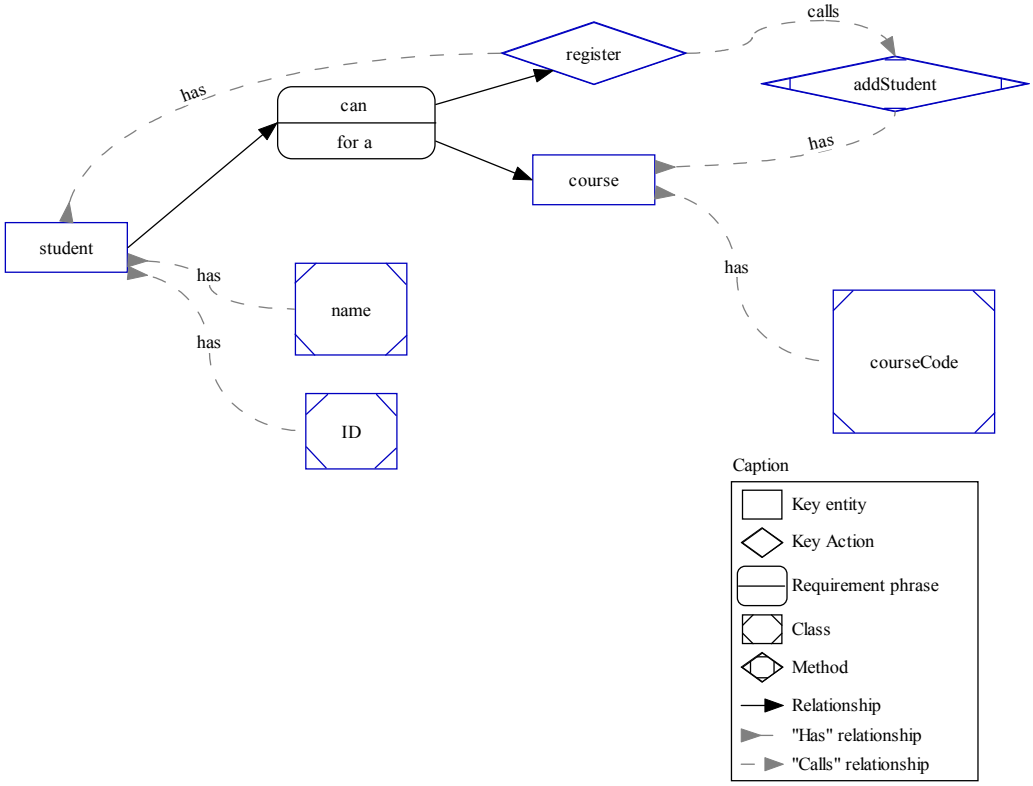

Ilustración III-4 - Vista aumentada

En este punto, se tendría que observar el diseño de Theme/UML aliñado con los requerimientos mostrados en Theme/Doc. Todos los elementos de diseño (clases y métodos) deberían ser mostrados, y esto permitirá evaluar la correctitud del diseño con solo mirar los requerimientos. Cabe destacar que esta vista no tiene como intención proveer una base formal para el análisis. Más bien, su intención es ubicar, en el contexto de los requerimientos, las decisiones de diseño tomadas. Este contexto le da al desarrollador una vista para verificar las decisiones de diseño propias.

III.1.2 - EA-Miner

El enfoque propuesto por Sampaio y otros en [48] y ampliado en [46] describe cómo diferentes fuentes de requerimientos no estructuradas (entrevistas, descripciones del sistema en lenguaje natural, etc.) pueden ser minadas automáticamente para ayudar al ingeniero de requerimientos a rápidamente identificar y construir un modelo orientado a aspectos de los requerimientos del sistema. El objetivo principal del *Método de Identificación Temprana de Aspectos* (Early Aspects Identification Method, Early-AIM) es determinar posibles aspectos candidatos en la documentación de requerimientos sin importar como estos están estructurados. El enfoque usa técnicas de *Procesamiento de Lenguaje Natural* (Natural Language Processing, NLP), las cuales proveen soporte para el análisis sensible al contexto de los requerimientos. Se provee una herramienta de soporte, EA-Miner, que ayuda al desarrollador a minar automáticamente y modelar los concerns crosscutting sin tener que haber leído previamente los documentos de requerimientos.

III.1.2.1 Proceso AORE

Este enfoque esta englobado en lo que los autores denominan el *proceso AORE* (AORE Process) [24]. Este tiene un nivel de abstracción mayor, el cual permite realizar metódicamente una serie de pasos para llevar a cabo las actividades necesarias de la ingeniería de requerimientos. El proceso esta soportado por una suite de herramientas, entre las cuales se encuentra EA-Miner, y el respectivo modelo de Early-AIM. El trabajo realizado por los autores es un proceso AORE general sintetizado de los varios procesos AORE propuestos en la literatura. Todas las tareas en el proceso (Ilustración III-5) pueden ser llevadas a cabo iterativamente y no necesariamente en el orden presentado.

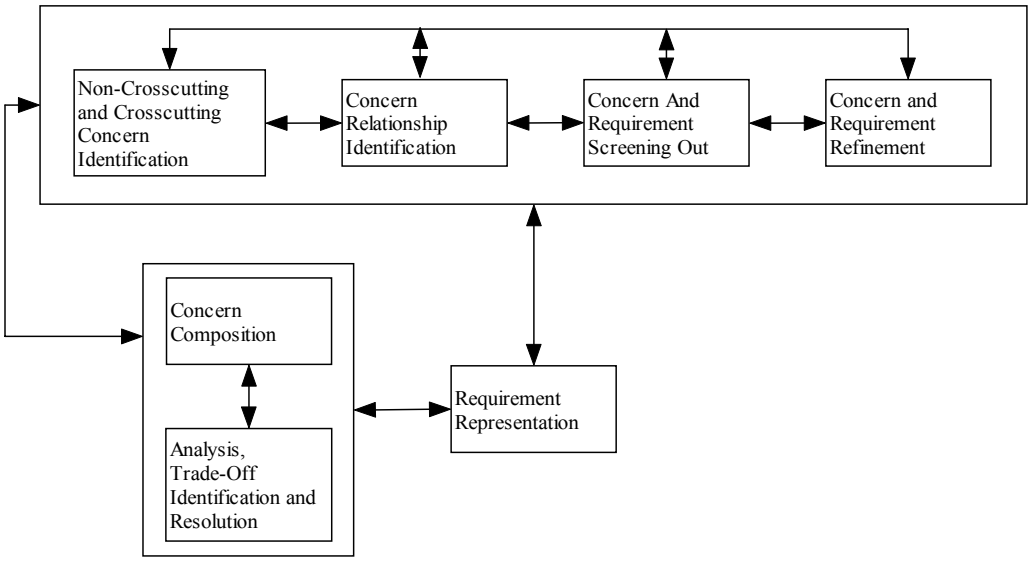

Ilustración III-5 - Proceso AORE

Una vez que esta disponible un conjunto inicial de documentos de requerimientos, el proceso comienza con el paso de *identificación de concerns no crosscutting y crosscutting* (Non-Crosscutting and Crosscutting Concern Identification). Aquí los varios concerns base (por ejemplo puntos de vista, casos de uso, etc.) como también los concerns crosscutting (por ejemplo los aspectos que atraviesan los concerns base) son identificados.

La identificación de concerns es seguida por la *identificación de las relaciones entre concerns* (Concern Relationship Identification). Esto es necesario de manera tal de entender cómo los concerns

se relacionan entre si. Más importante, sin embargo, es la necesidad esencial de que se razone acerca de las influencias crosscutting y las restricciones impuestas por los aspectos sobre los otros concerns.

Habiendo identificado los concerns y sus relaciones, se obtiene el conocimiento necesario para decidir cuáles concerns, relaciones y requerimientos son pertinentes para el sistema de software pretendido, y si hay repeticiones en la lista identificada. Este paso es denominado *descarte de concerns y requerimientos* (Concern and Requirement Screening Out).

Los concerns pertinentes restantes, sus requerimientos y relaciones son entonces representados en un formato determinado (textual, gráfico, u otros). La selección de la representación esta dictada por el enfoque AORE específico usado para el análisis. Este paso es necesario para la producción del documento de especificación de requerimientos final y es llamado *representación de requerimientos* (Requirement Representation).

Sin embargo, durante la representación de los concerns el desarrollador puede llegar a identificar la necesidad del refinamiento de estos, y pueden surgir, como consecuencia, nuevos concerns o requerimientos. Similarmente, pueden ser identificadas relaciones nuevas o alternativas. Por lo tanto, la representación de los requerimientos y los pasos previos de identificación están conectados. El paso donde se refinan se denomina *refinamiento de concerns y requerimientos* (Concern and Requirement Refinement)

Finalmente, los requerimientos representados en la notación seleccionada necesitan ser analizados para encontrar las influencias ejercidas por los aspectos como también por los conflictos potenciales entre aspectos[3] [36]. Estos análisis son facilitados por los pasos de *Composición de concerns* (Concern Composition) y el de *análisis, identificación y resolución de trade-offs* (Analysis, Trade-Off Identification and Resolution). Los resultados de la composición son analizados, los conflictos identificados y resueltos mediante consultas a los stakeholders.

III.1.2.2 Suite de herramientas

La suite de herramientas consiste de varias herramientas integradas en la cual cada una de ellas trabaja sobre la salida recibida de la herramienta aplicada previamente. Estas soportan la ingeniería de requerimientos llevada a cabo en varias de las tareas del proceso AORE. Las herramientas, las cuales se pueden observar en la Ilustración III-6, juegan dos tipos de roles: generadores de información y consumidores de información. Los generadores de información analizan los documentos de entrada y los complementan con información y anotaciones lingüísticas, semánticas y estadísticas. Las herramientas consumidoras de información usan estas anotaciones e informaciones adicionales para realizar múltiples tipos de análisis. Dentro de las herramientas generadoras de información se encuentra WMATRIX, que será explicado en detalle posteriormente, y las herramientas consumidoras incluyen la herramienta de *análisis de requerimientos* (Requirement Analysis Tool, RAT), la herramienta de *minería temprana de aspectos* (Early Aspect Mining, EA-Miner), la herramienta de *palabra clave en la frase* (Key Word In Phrase, KWIP) y la herramienta de *composición de requerimientos aspectuales y soporte de decisión* (Aspectual Requirements Composition And Decision Soport, ARCADE).

Tanto WMATRIX como RAT soportan la etapa de identificación de concerns y relaciones del proceso AORE. RAT usa la información estadística producida por WMATRIX para identificar los requerimientos que contienen las palabras estadísticamente significativas usadas en el texto de entrada. La herramienta KWIP se encarga de soportar la etapa de descarte. KWIP es aplicado para soportar el descarte manual de aquellos concerns y requerimientos redundantes. Esta extiende la técnica de concordancia estándar de WMATRIX variando la cantidad de contexto que rodea las palabras clave basándose en un conjunto de filtros heurísticos.

[3] Un conflicto entre aspectos es cuando más de un aspecto esta asociado al mismo artefacto y estos aspectos no son totalmente independientes, causando que el comportamiento del sistema pueda ser impredecible. Este es uno de los problemas de los enfoques y técnicas actuales de AOSD, ya que no tienen soporte (o este es mínimo) para la definición y el manejo de los conflictos entre aspectos.

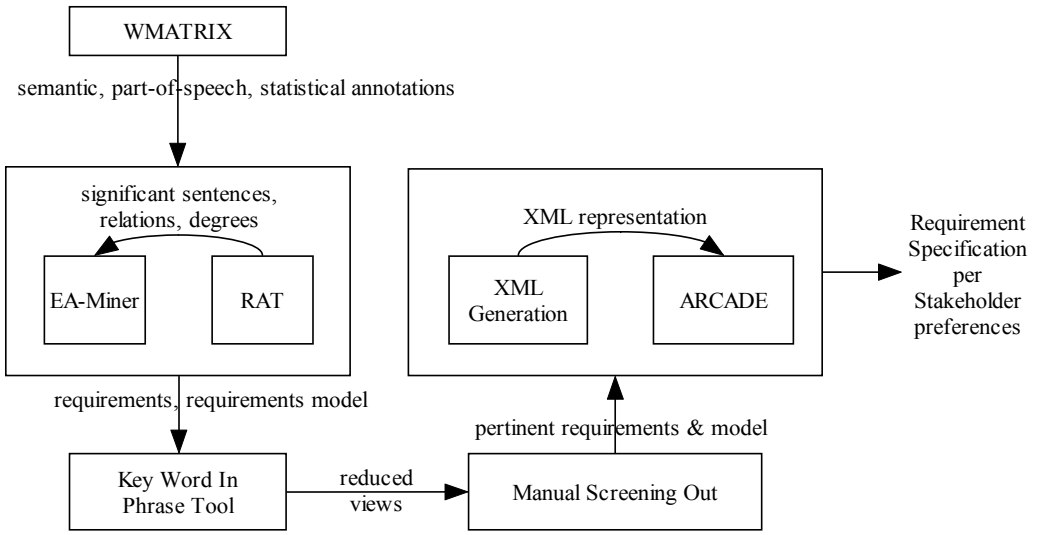

Ilustración III-6 - Suite de herramientas de soporte para el proceso AORE

La herramienta ARCADE se encarga de soportar las etapas de representación y composición de concerns como también de etapa de análisis de influencias, trade-offs y la resolución de trade-offs. Esta usa representaciones XML, gobernadas por esquemas XML, para especificar los puntos de vista, aspectos y las reglas de composición asociadas. Las reglas de composición son derivadas de la información generada por RAT. La representación semi-estructurada hace más fácil analizar los requerimientos puramente textuales basándose en las reglas de composición. El mecanismo de composición de ARCADE proyecta los aspectos sobre los puntos de vista de forma tal de hacer posible al ingeniero de requerimientos ver claramente cómo un aspecto potencial influencia o restringe puntos de vista específicos de los requerimientos. También incluye un componente analizador de trade-offs el cual identifica el solapamiento entre aspectos con referencias a los puntos de vista de los requerimientos a los cuales influencian.

III.1.2.3 Early-AIM y WMATRIX

Casi todas las aplicaciones de NPL a la ingeniería de requerimientos han usado técnicas basadas en reglas. Esto restringió severamente su aplicabilidad a documentos de requerimientos bien formados, que usan un subconjunto controlado del lenguaje natural y significa que textos no controlados (también denominados no estructurados) como transcripciones de las entrevistas a los stakeholders no son tratables usando este tipo de procesamiento. En la práctica, la síntesis automática de los requerimientos de estos documentos no era factible. Sin embargo, como se ha estudiado en varios proyectos (por ejemplo, REVERE), se ha podido usar satisfactoriamente el NLP para ayudar al analista a identificar conceptos importantes (como objetos, agentes, funciones, etc.). Estas herramientas están basadas sobre técnicas estadísticas las cuales proveen robustez a la hora de aplicarse sobre documentos no controlados.

Este enfoque (Early-AIM), como se dijo previamente, usa particularmente el procesador NLP WMATRIX [11], que provee características como *parte del discurso* (part-of-speech) y *etiquetado semántico* (semantic tagging), *análisis de frecuencia* (frequency analysis) y *concordancias* (concordances) para identificar conceptos del dominio de potencial significancia. El análisis de parte del discurso automatiza la extracción de sustantivos y verbos del texto (con un 98% de precisión según estudios prácticos); el análisis semántico agrupa palabras relacionadas y expresiones de varias palabras en conceptos (también llamadas categorías conceptuales), incluso cuando son usadas muchas formas de las palabras en los documentos (por ejemplo, las palabras “vehicle(s)”, “driver(s)” y “traffic” son agrupadas en el campo semántico o “etiquetados” como “land transport”) o cuando una palabra tiene muchos significados y puede ser asignada a más de una categoría semántica el análisis

ayuda a identificar la categoría correspondiente de la palabra según el contexto donde ocurre; y el análisis de frecuencia provee información estadística con respecto a cada termino de los documentos, incluyendo cantidad de ocurrencias, varianzas, etc. Este procesador es capaz de etiquetar cualquier tipo de texto escrito en ingles.

		Item	O1	%1	O2	%2	LL	Semantic Group
List	Concordance	I2.2	61	14.32	2738	0.28 +	357.93	Business: Selling
List	Concordance	X4.2	21	4.93	3108	0.32 +	75.34	Mental object: Means, method
List	Concordance	I1.1	9	2.11	2654	0.27 +	21.07	Money: Affluence

KEY:
O1 is observed frequency.
O2 is observed frequency for normata.
%1 and %2 values shows relative frequencies in the texts.
+ indicates overuse in O1 relative to O2.
- indicates underuse in O1 relative to O2.
The table is sorted on log-likelihood (LL) value to show key items at the top.

Tabla III-1 - Datos generados por la herramienta WMATRIX

Los autores utilizaron un caso de estudio de ejemplo para mostrar los resultados del analisis de la herramienta WMATRIX. Este esta basado en *sistema de remates* (Auction System), que sera usado para comprar y vender bienes a traves de subastas online. La Tabla III-1 muestra los datos generados por WMATRIX. De esta se puede observar que los requerimientos son principalmente acerca de "Selling" (LL357.93) con términos relacionados de "auction", "buying", "selling", "customers" y "bids"; "Online system" del "Mental object" que el texto describe (LL 75.34) y "Money" (LL 21.07) con términos relacionados como "credit", "credit card" y "invest". La lista de conceptos para cada categoría semántica en el documento puede ser vista del hipervínculo "List", y su ocurrencia en el documento del hipervínculo "Concordance".

III.1.2.4 EA-Miner

Como el lenguaje natural es una forma conveniente de expresar requerimientos, las investigaciones realizadas para el desarrollo de EA-Miner se enfocaron en usar las características de WMATRIX que ayudan en la automatización de la identificación de algunos conceptos como objetos, roles de usuario, funcionalidades y aspectos tempranos en los documentos de requerimientos. En particular EA-Miner ayuda a identificar automáticamente elementos como *puntos de vista* (viewpoints) y *casos de uso* (use cases) basándose en algún criterio como identificar verbos que denotan acciones como posibles casos de uso o sinónimos como posibles puntos de vista.

La herramienta EA-Miner utiliza las características provistas por WMATRIX, como el etiquetado de parte del discurso y el análisis semántico, con el propósito específico de identificar aspectos tempranos[4] y las relaciones entre los requerimientos, y además se complementa con el uso de su propio léxico para requerimientos crosscutting y documentos de modelos de representación.

Para la identificación de concerns crosscutting no funcionales (los cuales frecuentemente son fuertes candidatos en convertirse en aspectos), según lo expresado en [23], el léxico de EA-Miner esta

[4] Se utiliza las palabras "aspectos tempranos" refiriéndose a la traducción de Early Aspect, la cual es sinónimo de aspecto candidato, es decir, aspectos detectados tempranamente en el desarrollo que deben ser analizados por el analista para decidir si serán o no un aspecto (final) del sistema.

construido sobre (y extiende) los árboles de requerimientos no funcionales del *Framework de Requerimientos No Funcionales* (NFR Framework [25]). Los aspectos no funcionales son identificados mediante la asignación las palabras cercanas semánticamente a los subgrupos para cada categoría de los NFR, por lo que todos los aspectos relacionados a los NFR son identificados. Adicionalmente, un léxico para cualquier nuevo aspecto no funcional que no es incluido en el NFR Framework, pero que es considerado ser útil, puede ser incorporado en el léxico de la herramienta. Para la identificación de concerns crosscutting funcionales, EA-Miner utiliza una estrategia similar a la de Theme/Doc, detectando las ocurrencias repetidas de action words, las cuales sugieren la presencia de un aspecto funcional. En la práctica, el léxico de EA-Miner ayuda a identificar las palabras semánticamente relacionadas a un solo concern (no funcional) para sugerirlo como aspecto, y también alerta a los ingenieros de requerimientos cuando una funcionalidad es mencionada en varios requerimientos, lo que indica alguna asociación crosscutting de esa funcionalidad.

La herramienta también facilita la producción del modelo aspectual y la representación de los documentos de requerimientos con las estructuras elegidas. Por ejemplo, ayuda a identificar automáticamente puntos de vista usando las anotaciones de parte del discurso provistas por WMATRIX (todos los sustantivos son identificados como puntos de vista potenciales. Debido a que la lista de sustantivos para documentos grandes puede ser muy larga, EA-Miner aplica varias estrategias de reducción mediante:

1. Usando *lematización* (lemmatization) para reconocer palabras que tienen la misma raíz (por ejemplo, “customer” y “customers”) y tratándolas como el mismo viewpoint.
2. Usando diccionarios de sinónimos para fusionar palabras con el mismo significado (por ejemplo, “client” y “customer”).
3. Usando las normas de frecuencia de WMATRIX para considerar como viewpoints sugeridos solo aquellos sustantivos significativos (los que han sido usados muchas veces).

Entonces EA-Miner provee la lista de requerimientos relacionados a cada viewpoint y a sus aspectos relacionados. Hay que notar que EA-Miner no esta limitado a la estructuración de requerimientos basada en viewpoints. Otros modelos de requerimientos pueden ser integrados dentro de la herramienta. Por ejemplo, puede ser aplicado al desarrollo de documentación del estilo de los casos de uso mediante la identificación de los posibles casos de uso de los verbos de acción, y relacionando los requerimientos correspondientes a ellos. Son utilizables las mismas estrategias de reducción explicadas anteriormente en estas instancias.

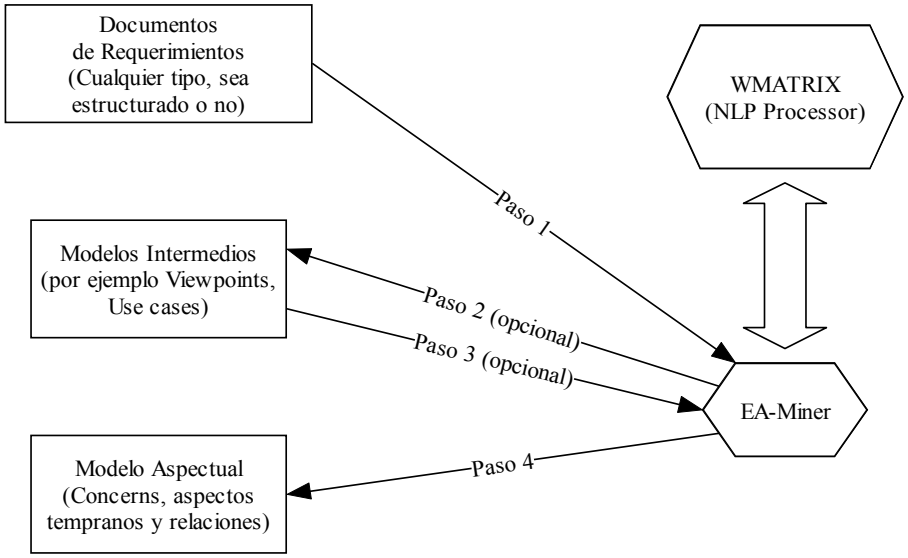

Ilustración III-7 - Modelo de minería de aspectos Early-AIM

El enfoque, mostrado en la Ilustración III-7, empieza en el Paso 1 mediante el análisis de documentos existentes que son fuentes para la elicitación de requerimientos como entrevistas hechas con los stakeholders o descripciones informales del sistema. La herramienta lee estos documentos y se los entrega a WMATRIX para que sean procesados, el cual produce análisis para ayudar a la identificación de concerns y viewpoints usando las técnicas previamente descriptas de NLP. La herramienta EA-Miner permite filtrar la información que fluye hacia y desde WMATRIX (como por ejemplo, mostrando solo un subconjunto de los verbos identificados basándose en algún criterio). Al final del paso 1 una especificación de requerimientos más estructurada (*Modelo Intermedio*, Intermediate Model) puede ser producida como salida. Es importante denotar que esta especificación no esta completamente automatizada por las herramientas. Los siguientes dos pasos (Paso 2 y Paso 3) son opcionales y tienen como propósito facilitar la producción de modelos aspectuales en el Paso 4 mediante la producción de un modelo de requerimientos más estructurado como entrada. Estos pasos consisten en leer directamente de los documentos informales o del modelo intermedio, y procesar la información para buscar requerimientos crosscutting y aspectos candidatos. La herramienta de minería tiene el rol de configurar los criterios (por ejemplo, aplicar filtros para extraer la información desde la herramienta) que será usada por WMATRIX para procesar la información. Además, la herramienta puede filtrar los resultados de tal manera de identificar los aspectos candidatos y producir un modelo que represente la relación entre los requerimientos. En estos pasos se puede reusar el modelo de viewpoints intermedio. La herramienta EA-Miner (vía WMATRIX) examina la dispersión de los aspectos candidatos entre los varios requerimientos. Si el aspecto candidato esta bien disperso (por ejemplo, aparece en muchos requerimientos) entonces la posibilidad de que ese aspecto candidato sea un aspecto concreto es más fuerte. En el Paso 4 se produce un modelo que muestra los concerns, aspectos tempranos y sus relaciones. La implementación de este paso esta basado en las siguientes cuestiones:

- Palabras de acción (Action words):

  La herramienta puede automáticamente buscar action words, identificadas como verbos por la herramienta WMATRIX, y producir un modelo que represente las relaciones entre los requerimientos, basándose en datos estadísticos (para razonar acerca de que palabras afectan otras).

- Análisis semántico y dominio léxico (Semantic analysis and Domain lexicon):

Un léxico especifico del dominio, basado en los *catálogos de requerimientos no funcionales* (NFR Catalogues), categorizan crosscutting concerns bien conocidos. EA-Miner puede entonces analizar sintacticamente el texto y etiquetar los posibles aspectos tempranos, ayudado por el etiquetado semántico previamente hecho por WMATRIX y por el léxico del domino, para evitar la selección de candidatos erróneos.

- Meta-modelos y heurísticas (Meta-models and Heuristics):

  Para producir la salida del Paso 4 (el modelo aspectual), EA-Miner busca los aspectos candidatos identificados y también modelos intermedios ingresados específicamente, como sus meta-modelos y heurísticas de construcción.

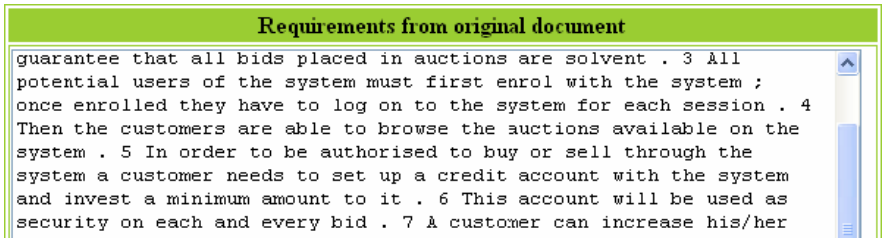

Ilustración III-8 - Captura de pantalla 1 de EA-Miner

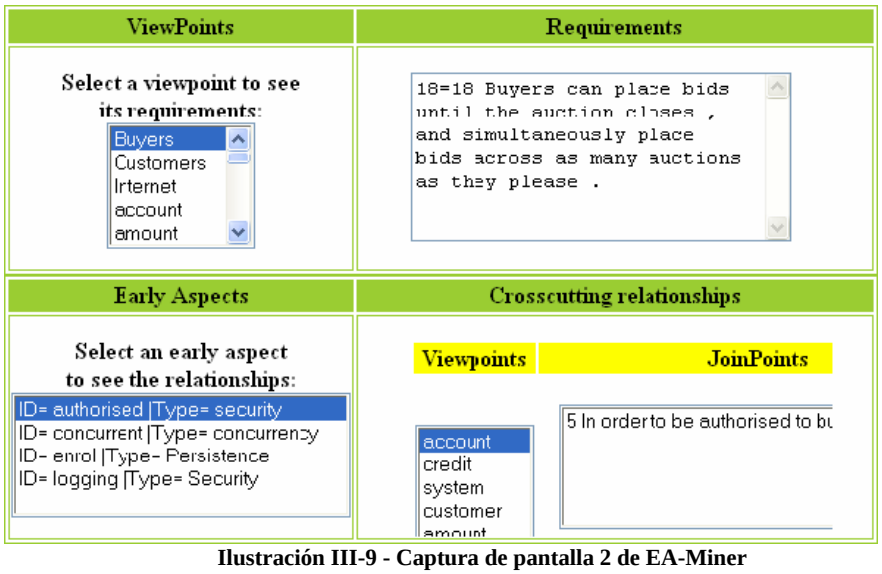

Ilustración III-9 - Captura de pantalla 2 de EA-Miner

Específicamente, y entrando en detalle, la herramienta WMATRIX provee soporte en las siguientes tareas:

- Identificar viewpoints:

  La herramienta WMATRIX ha sido anteriormente usada para identificar stakeholders en los documentos de requerimientos. El criterio usado aquí, era identificar sustantivos de agentes humanos con terminaciones comunes para títulos de trabajo (tales como “er”, o “et” o “or” o “man”) como por ejemplo “controller” o “pilot”. Esto puede ser extendido a identificar otros tipos de viewpoints que no son humanos. La lista de viewpoints candidatos es producida automáticamente y mostrada al ingeniero de requerimientos para que pueda seleccionar los relevantes y poder producir los modelos intermedios previamente descriptos. Asimismo la herramienta de minería puede también insertar etiquetas en el texto que ayuden al IR a

encontrar rápidamente requerimientos relacionados con los viewpoints. En la Ilustración III-10 se observa como WMATRIX es usado para analizar las descripciones informales dadas y lista los viewpoints candidatos. Los viewpoints candidatos son automáticamente identificados por WMATRIX mediante el listado de los sustantivos en el texto acompañados por su clase de parte del discurso y su frecuencia de ocurrencia. El usuario puede ver partes de pequeñas oraciones donde todas las ocurrencias de una palabra específica suceden, y tiene la opción de ver una perspectiva más detallada del texto para cada ocurrencia de una palabra en particular. Estas características habilitan al usuario a minar rápidamente los viewpoints y sus requerimientos relacionados debido a que debe enfocarse sólo en leer pequeñas partes del texto.

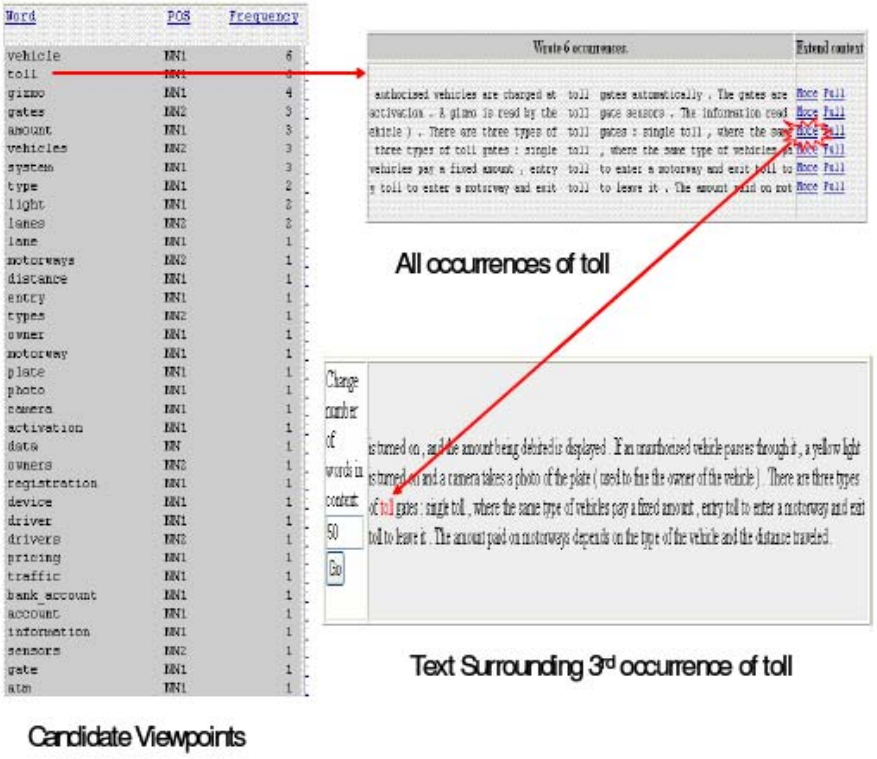

Ilustración III-10 - Identificación de viewpoints

- Identificar requerimientos crosscutting:

  Esta tarea puede ser soportada por las herramientas de diferentes maneras. En uno de los enfoques (similar a Theme/Doc), la herramienta puede automáticamente buscar action words, que son identificadas como verbos por la herramienta WMATRIX, y producir un modelo que represente las relaciones entre los requerimientos. El ingeniero no tiene que proveer las action words por adelantado, y además, el NPL permite un análisis de las palabras con mayor sensibilidad al contexto reconociendo por ejemplo que las acciones “collect” y “pick up” son del mismo campo semántico. La Ilustración III-11 muestra cómo WMATRIX es usada para listar las action words reconocidas como verbos en el texto de la especificación de requerimientos.

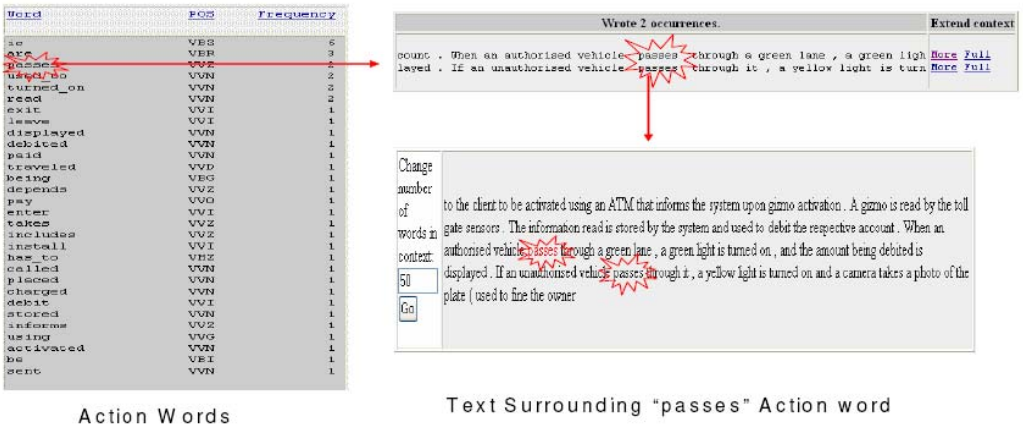

Ilustración III-11 - Identificación de palabras de acción

La herramienta provee las características necesarias para filtrar la lista de action words, por ejemplo, excluyendo aquellos verbos que no representen una acción como los verbos auxiliares (“is”, “are”, “be”, etc.) Además, la lista podría ser filtrada para presentar los distintos verbos, que tienen el mismo significado semántico en el contexto, como la misma acción. Otro enfoque para identificar las influencias crosscutting podría ser configurar filtros los cuales busquen tipos conocidos de palabras que se mapean a requerimientos no funcionales como la seguridad, persistencia y performance, y sugerirlos como concerns. El problema de este enfoque es que algunas veces esos concerns son implícitos y difíciles de ser minados automáticamente, dependiendo más en el buen juicio de los desarrolladores. En la Ilustración III-12 se muestra un ejemplo de cómo el análisis semántico puede ser utilizado para agrupar palabras por sus clases semánticas (la herramienta identifica tres ocurrencias de la clase semántica S7.4, que significa “permission”).

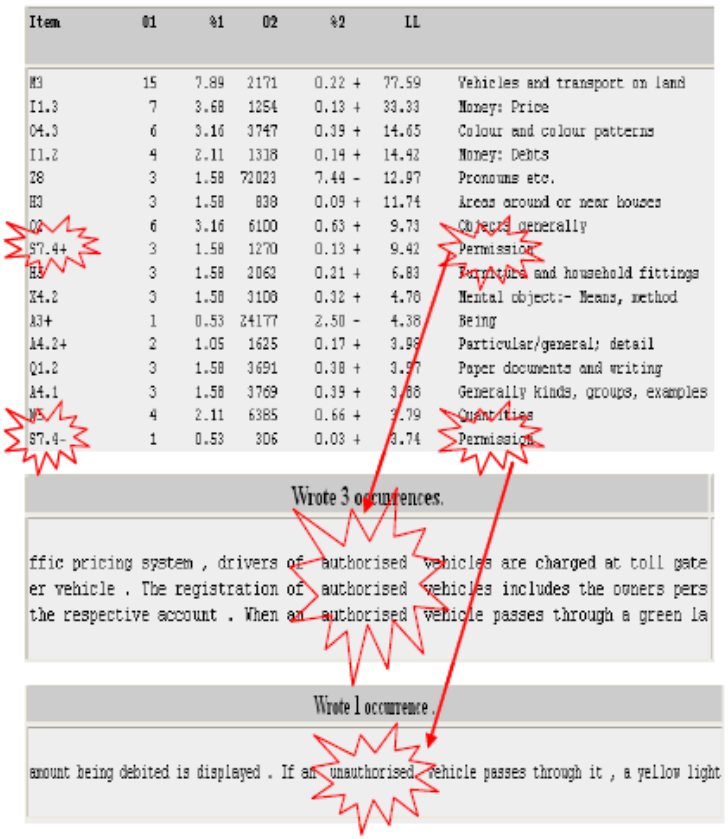

Ilustración III-12 - Identificación de concerns mediante el análisis semántico

En síntesis, el enfoque propuesto en Early-AIM y soportado por la herramienta EA-Miner (ver Ilustración III-8 y Ilustración III-9) puede ser utilizado sin importar la estructura de los documentos provistos como entrada. Los dominios de documentación "pesada" basados en regulaciones, estándares y varios tipos de documentaciones extensivas pueden en consecuencia beneficiarse de este enfoque. La herramienta permite el desarrollo acelerado debido a que no impone ningún tipo específico de formato y no depende del conocimiento previo de los requerimientos por parte del ingeniero. Es importante mencionar que la herramienta requiere la intervención de los ingenieros de requerimientos para llevar a cabo la creación de los modelos intermedios.

III.1.3 - IR for Identifying Crosscutting Concerns in Requirements Specifications

El enfoque presentado por Rosenheiner en [45], se basa en la minería de aspectos sobre las especificaciones de requerimientos mediante la aplicación de algoritmos de IR [26]. El estudio esta centrado principalmente en explorar las interacciones entre los requerimientos no funcionales y los funcionales. El procedimiento semi-automático propuesto trabaja localizando influencias crosscutting de requerimientos en particular (provistos como entrada por el analista) en esas especificaciones. Aunque el estudio del enfoque solo se realizó sobre dos casos de prueba, uno cuya documentación estaba basada en casos de uso, y otra cuyas especificaciones estaban escritas de la manera tradicional (sin casos de uso ni otra técnica de modelaje), no se descarta que pueda funcionar sobre otros estilos de documentación. Sin embargo, los autores no lo aseguran y exponen que se deberían realizar investigaciones futuras para afirmarlo.

Los resultados de la técnica fueron que si bien es relativamente fácil identificar requerimientos no funcionales crosscutting si estos son mencionados en una sección separada especialmente dedicada a los requerimientos (o calidades) no funcionales, no es tan fácil identificar funcionalidad crosscutting cuando no esta explícitamente mencionada en una subsección. Sin embargo, durante las pruebas sobre los casos de estudio con la técnica de identificación propuesta, se detectaron ciertos concerns que afectaban los casos de uso. Pero en este caso también era notoria la influencia en el contenido de la especificación. Además, para encontrar los requerimientos que son cortados transversalmente por otros, el estudio de las especificaciones de requerimientos debe realizarse con el aspecto en mente, es decir, uno debe saber lo que se quiere encontrar, y no que la herramienta lo encuentre por uno.

Durante el desarrollo de los casos de estudio, se tuvo en mente un requerimiento (no funcional) que afectaba a otros, y que acontecía en ambos casos de estudio. Este requerimiento posteriormente fue utilizado para aplicar las técnicas de IR. Analizando las influencias crosscutting solamente en los requerimientos funcionales, la búsqueda fue restringida a las secciones que sólo trataban sobre funcionalidad. Debido a que ambas especificaciones de requerimientos son de diferentes tipos (una basada en casos de uso, y la otra no) y por lo tanto una parte de la terminología usada en ambas difiere, para unificarlas los elementos más detallados serán referidos como *hojas de requerimientos* (Requirement Leaves, RL). Si una RL es atravesada por algún otro requerimiento sus padres (elementos de requerimientos superiores) son afectados de igual manera. Mientras que un RL relevante para el caso de estudio basado en casos de uso son los pasos de los escenarios (o flujos), una precondición o una poscondición, en el realizado de forma tradicional lo son las descripciones de requerimientos, precondiciones, poscondiciones o los efectos secundarios (side effects).

Fueron aplicadas tanto la inspección como las técnicas basadas en IR para encontrar las RL que eran cortadas transversalmente por el requerimiento previamente seleccionado, en ambas especificaciones de requerimientos. Las experimentaciones preliminares mostraron que el siguiente simple procedimiento semi-automático basado en la IR (Código fuente III-1) es relativamente efectivo para la tarea de la identificación:

```
1. Usar expresiones regulares que según los patrones <NOMBRE DEL
   SISTEMA> <PALABRA QUE EMPIECE CON MINUSCULA> y <NOMBRE DEL SISTEMA>
   deberá <PALABRA QUE EMPIECE CON MINUSCULA> para encontrar todas las
   frases que cumplan con el criterio de búsqueda (por ejemplo "El
   sistema responde…" o "El sistema deberá generar…").
2. Ordenar los resultados de la búsqueda y remover los duplicados.
3. Remover manualmente todas aquellas frases del resultado de la
   búsqueda que sean irrelevantes (por ejemplo aquellas que no estén
   relacionadas con el requerimiento tenido en mente para realizar la
   consulta).
4. Encontrar todas las líneas y sus contextos mediante expresiones
   regulares formadas de los resultados restantes (aquellos no
   removidos).
5. De los resultados obtenidos, manualmente encontrar todas las líneas
   que tengan declaraciones de requerimientos que estén afectados por el
   requerimiento seleccionado previamente, y ordenar todas las líneas
   irrelevantes.
```

Código fuente III-1 - Pseudo-código del enfoque de IR

En los pasos 3 y 5 puede llegar a ser necesario consultar el texto de las especificaciones de requerimientos si una línea o su contexto no contienen la información suficiente para tomar la decisión correcta. Sin embargo, estas consultas fueron raramente utilizadas durante el estudio.

III.1.4 - On Demand Virtual Remodularization Using Program Graphs

La investigación llevada a cabo por Shepard y otros[5] en [50-52] esta enfocada en el desarrollo de un grafo (graph) que pueda soportar la re-modularización virtual en demanda de un programa (Virtual On-Demand Program Remodularization). El grafo puede ser usado para generar una vista del código base que presenta segmentos de código relacionados como si estuviesen en un solo archivo. Se han construido herramientas concretas, como un ubicador de características, un recuperador de conjuntos de trabajo y una herramienta de minería de aspectos (Feature Location, Working Set Recovery, y Aspect Mining Tool) que demuestran la utilidad de este grafo, y el potencial de acelerar los bloques de construcción de las tareas de programación modernas. Los puntos novedosos del enfoque son:

1. El uso de un procesador de lenguaje natural (NLP) como herramienta de análisis del programa.
2. El desarrollo de una representación subyacente para soportar tanto el NLP como el análisis de programas tradicional.
3. La representación de un programa orientado a objetos en función de sus verbos.

III.1.4.1 Utilizacion de los pares Verb-DO

Debido a que los enlaces de dependencia estructurales que relacionan el código generalmente son largos y oscuros, se investigo el uso del NLP para conectar los verbos dentro de un programa, conectando fuertemente las piezas diseminadas de una acción. En un lenguaje de programación, los verbos se corresponden a las acciones (u operaciones) y los sustantivos se corresponden a los objetos. Frecuentemente los verbos actúan en muchos objetos diferentes en un mismo programa. Por lo tanto, es importante considerar el tema para identificar precisamente una acción específica. Hay una fuerte relación específica entre los verbos y sus temas en el lenguaje inglés. Un tema es el objeto sobre el cual la acción (implicado por el verbo) actúa, y usualmente se expresa como un objeto directo (direct object, DO). Un ejemplo de una relación verbo - objeto directo (verb-DO) en el lenguaje natural es (parked,car) en la sentencia “The person parked the car”. La relación verb-DO puede ser usada para varios propósitos, por ejemplo, para clasificar sustantivos. Sustantivos similares pueden ser agrupados en una jerarquía examinando los verbos con los cuales los sustantivos son usados como DO. Los autores se centran más en los DO que en los sujetos, porque en el código orientado a objetos, el objeto adjunto es usualmente el sujeto. De forma tal de recuperar esta información acerca de los programas, se procesa el código fuente para extraer los pares verb-DO. Entonces, se podrán construir herramientas que usen estos pares para asistir al usuario en la navegación y visualización del código de una manera que atraviese la descomposición dominante. La representación de estos pares será similar a un índice inverso (como el usado en IR), donde se mapearan los pares verb-DO a las ocurrencias en el código actual.

III.1.4.2 Grafos AOIG

Los autores han diseñado un modelo de programa novedoso que captura las relaciones orientadas a acciones entre identificadores en el código fuente de un programa. Específicamente, el modelo representa explícitamente las ocurrencias de verbos y objetos directos (DOs) en un programa, como implica el uso de identificadores definidos por el usuario. Solamente se analizan las ocurrencias de los verbos y DOs en las *declaraciones de métodos*, y en los *comentarios* que estén dentro o se refieran a las declaraciones de métodos, porque las declaraciones de métodos son el centro de los concerns. El modelo identificador orientado a acciones es definido como un grafo, llamado *grafo identificador orientado a acciones* (Action-Oriented Identifier Graph, AOIG), el cual contiene cuatro tipos de nodos:

- Un *nodo de verbo* (verb node) existe para cada verbo diferente que ocurre en el programa.

[5] Si bien esta técnica fue principalmente diseñada para ser utilizada sobre código fuente, es interesante debido a que también se realiza un análisis sobre los comentarios del código fuente, los cuales están escritos en lenguaje natural.

- Un *nodo de objeto directo* (DO node) existe para cada objeto directo único en el programa.
- Un *nodo verbo-objeto directo* (verb-DO node) existe para cada par verb-DO identificado en el programa. Un par verb-DO se define por dos identificadores, localizados uno a continuación del otro, donde se determina que el primer identificador es una acción o verbo, y el segundo identificador esta siendo usado con el rol de objeto directo de la acción del primer identificador.
- Un *nodo de uso* (use node) existe para cada ocurrencia de un par verb-DO en los comentarios o en código del programa.

  También hay dos tipos de enlaces en el AOIG:

- Un enlace de emparejamiento (pairing edge) tiene un nodo de verbo *v* o un nodo de DO *d* como fuente, y un nodo verb-DO como sumidero, cuando se determina que el verbo *v* y el DO *d* son usados juntos. Un nodo de verbo dado (o un nodo de DO) pueden tener enlaces a múltiples nodos verb-DO; sin embargo, un nodo verb-DO dado solo tiene dos enlaces entrantes, uno desde el nodo de verbo y uno desde el nodo de DO involucrados en la relación.
- Para cada ocurrencia (o uso) de un par verb-DO dado en el programa, hay aun diferente enlace de uso (use edge) que mapea el nodo verb-DO correspondiente al nodo de uso representando el uso de ese par en el programa.

  Un ejemplo de este grafo se puede observar en la Ilustración III-13.

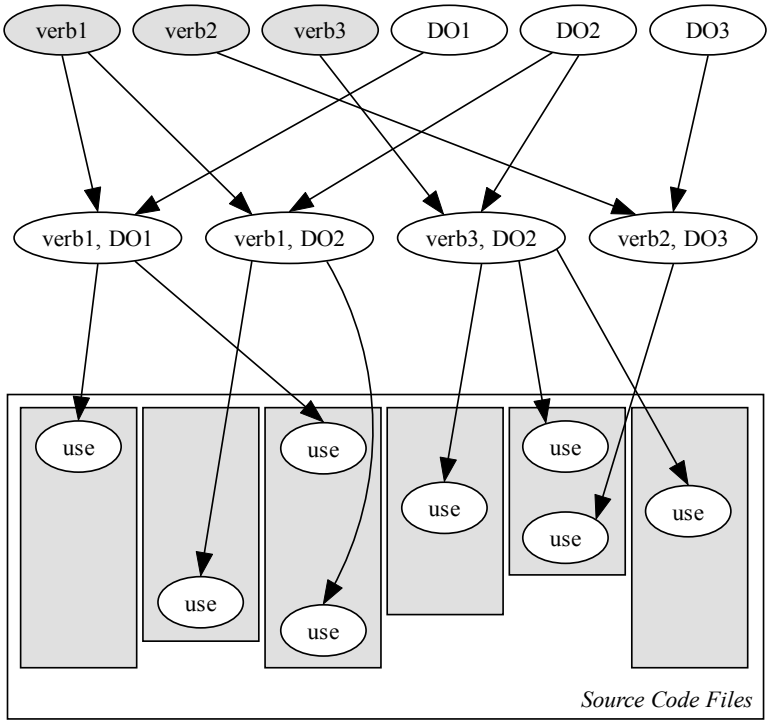

Ilustración III-13 - Ejemplo de un AOIG

III.1.4.3 AOIGBuilder

La extracción del AOIG del código fuente implica el análisis del lenguaje natural (para los comentarios) como también de las estructuras del programa como las declaraciones (para el código).

En este punto, y teniendo en mente los objetivos de este trabajo, solamente nos interesa el modelo propuesto por los autores que se encarga del análisis de los comentarios.

El proceso de extraer pares verb-DO de los comentarios es ilustrado en la Ilustración III-14. Una vez que los pares son extraídos, estos son usados para crear los nodos y enlaces apropiados en el AOIG. Esta tarea es automatizada por la herramienta AOIGBuilder.

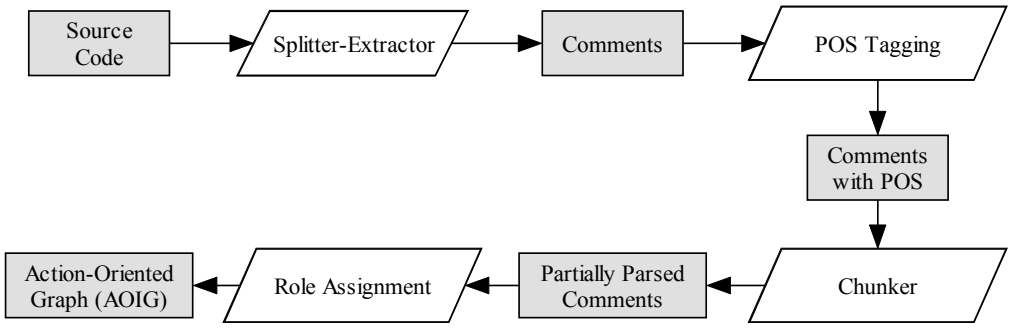

Ilustración III-14 - Proceso de construcción del AOIG de los comentarios del código fuente

Para extraer los pares verb-DO de los comentarios, se siguen tres pasos principales:

1. El *etiquetado de parte del discurso* (Part-of-speech, POS) de cada comentario.
2. El *agrupamiento en frases simples* (chunking) de las etiquetas POS de los comentarios.
3. La *deteccion mediante patrones* (pattern matching) en cada documento para determinar pares verb-DO.

Para crear los nodos verb-DO de los comentarios, es asignada una etiqueta POS (por ejemplo verbo, adjetivo, sustantivo, etc.) a cada palabra en cada comentario. Estas etiquetas POS son usadas para fragmentar las sentencias en frases básicas como frases sustantivas base (basic noun phrase), secuencias de grupos de verbos (verb group sequences) y frases preposicionales (prepositional phrases). Mientras que los lenguajes naturales son notorios por sus ambigüedades, esta fragmentación (también conocida como “robust parsing” o “partial parsing”) puede ser realizada de forma precisa. Además, esta fragmentación es suficiente para el propósito de detectar verbos y sus objetos directos. Los objetos directos son detectados por un simple “pattern matching”, encontrando las frases sustantivas que siguen inmediatamente a los verbos en voz activa, salteando algunos modificadores verbales como los adverbios, o escaneando por los sujetos en el caso de grupos de verbos pasivos. Una vez que los verbos y los DO fueron detectados, se construyen los nodos de uso para representar el comentario y conectarlo a su nodo verb-DO correspondiente, creando uno si fuese necesario, lo cual podría causar la creación en cascada de un nodo de verbo o un nodo de DO, si alguno de ellos no ha sido creado todavía. A modo de ejemplo, consideremos el comentario:

“Remove an item from the cart”.

Este se etiquetaría:

Remove$_{<v>}$ an$_{<dt>}$ item$_{<n>}$ from$_{<p>}$ the$_{<dt>}$ cart$_{<n>}$.

Se parsearia parcialmente:

Remove$_{<v>}$ [an item]$_{<NounPhrase>}$ [from [the cart]$_{<NounPhrase>}$]$_{<PrepositionalPhrase>}$.

Y usando extracción de patrones se determinaría el par verb-DO:

(Remove,item).

Cuando se construye el nodo de uso de este comentario, se adjuntaría al nodo verb-DO (Remove,item), creando uno si es necesario.

III.1.4.4 CCVerbFinder

Para el desarrollo de la herramienta de minería propuesta, los autores primero se plantearon una serie de requerimientos (u objetivos) que esta debía cumplir. Se estableció que debía ayudar al desarrollador minar los concerns que cuya existencia conoce como aquellos que desconoce, pero que están modularizados pobremente. Esto segundo, es particularmente importante para aquel desarrollador que desea mejorar la legibilidad, mantenabilidad y usabilidad de su código de manera sistemática. También debería establecer un ranking de concerns, yendo de los peores modularizados a los mejores, de manera que el desarrollador pueda concentrar sus esfuerzos de refactorización en los concerns que tienen más chances de retrasar el proceso de desarrollo.

De forma de validar la utilidad para la minería de aspectos, se construyó una herramienta simple que usa el AOIG. Esta herramienta representa un primer paso en el uso del AOIG para este propósito, y se espera construir sobre él y extender su funcionalidad en el futuro. Esta herramienta, llamada CCVerbFinder, encuentra los verbos que cortan transversalmente el sistema, y los propone como buenos candidatos para la refactorización[6]. CCVerbFinder encuentra los verbos crosscutting recorriendo el AOIG, empezando de cada nodo de verbo individual en el conjunto de nodos de verbo del AOIG (o de los nodos verb-DO si la información deseada es respecto a los pares verb-DO). Para cada nodo de verbo *v*, CCVerbFinder recorre los enlaces desde *v* hasta los nodos verb-DO alcanzables, y luego recorre los enlaces desde los nodos verb-DO alcanzados terminando en los nodos de uso, alcanzados desde el nodo de verbo de comienzo. La herramienta cuenta el número de archivos únicos que contienen nodos de uso alcanzables, y reportan esta cuenta de crosscut (crosscutcount) para el verbo *v*.

En la Ilustración III-13, si el CCVerbFinder estaba contando por el nodo de verbo “verb1”, recorrería el AOIG a <verb1, DO1> y <verb1, DO2>. De cada uno de esos nodos verb-DO, recorrería los enlaces de uso para encontrar dos usos de cada par verb-DO, para un total de cuatro usos de “verb1”, el cual ocurre en tres archivos diferentes. Al final, los verbos o pares verb-DO que generen grandes contadores de archivos son los más probables que implementen concerns crosscutting, porque su implementación esta dispersada en un gran número de archivos. En el futuro los autores desearían realizar esto con un nivel de granularidad más fino.

[6] El termino refactorización esta utilizado con el significado de transformar el código de fuente orientado a objetos en código orientado a aspectos.

III.1.5 - Aspect Extractor Tool

En el trabajo de tesis de grado de Haak y Díaz [27, 28], estos presentan un enfoque denominado Aspects Extractor, que tiene como objetivos identificar, especificar, integrar y evaluar los aspectos en etapas tempranas del ciclo de desarrollo del software. Para lograrlos, se automatizan las tareas que conforman el modelo de ingeniería de requerimientos propuesto. Esta automatización se realiza sobre la información de la funcionalidad del sistema brindada por el analista y se obtiene una lista de aspectos candidatos. El analista decide cuáles son los aspectos definitivos a partir de tal lista. Esta automatización contribuye al trabajo del analista, ya que no debe inspeccionar la especificación de casos de uso de un sistema en la búsqueda de aspectos.

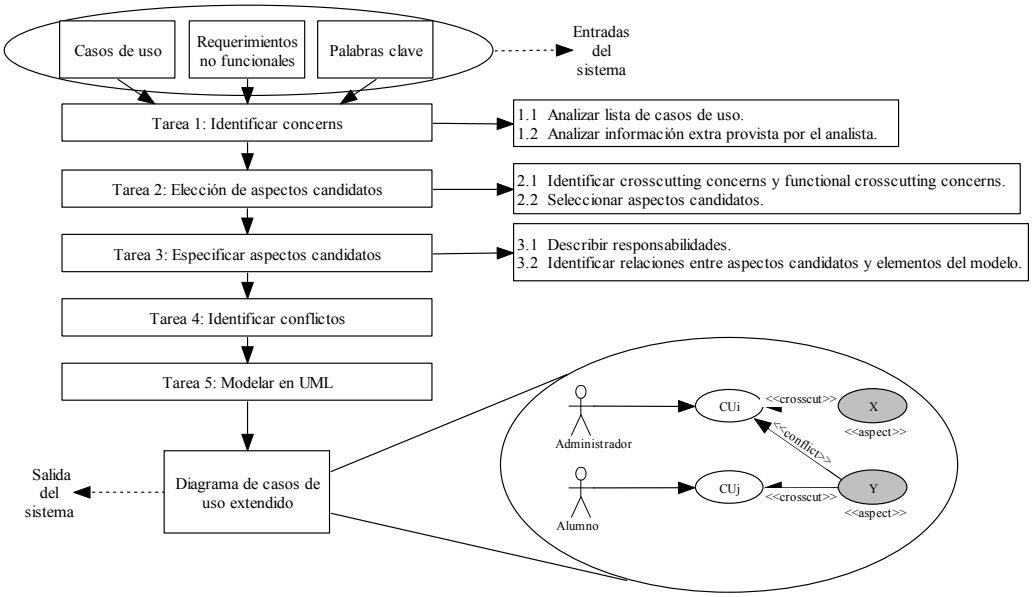

Ilustración III-15 - Modelo de ingeniería Aspect Extractor

En la Ilustración III-15 se muestra el modelo de ingeniería de requerimientos Aspects Extractor, el cual esta conformado por cinco tareas principales: Identificar concerns, Elección de los aspectos candidatos, Especificar aspectos candidatos, Identificar conflictos y por último, Modelar en UML, que se explican brevemente a continuación:

- Tarea 1: Identificar concerns

  Consiste en identificar a partir de los requerimientos, los posibles concerns de un sistema. En esta tarea se genera automáticamente información de interés que se utilizará en las tareas restantes.

- Tarea 2: Elección de aspectos candidatos

  En esta tarea se identifican los crosscutting concerns y los functional crosscutting concerns. Una vez identificados, el analista seleccionará algunos de ellos de acuerdo a su experiencia e intuición. Estos concerns pasan a ser aspectos candidatos.

- Tarea 3: Especificar aspectos candidatos

  El analista debe describir el objetivo y la funcionalidad de los aspectos candidatos seleccionados en la tarea anterior. Además, el sistema determina con cuáles elementos del modelo está relacionado un determinado aspecto candidato.

- Tarea 4: Identificar conflictos

  Se identifican todas las posibles situaciones conflictivas que se dan cuando un elemento del modelo es afectado por más de un aspecto candidato. Si existe dicha situación, entonces será indicada de forma que pueda ser tratada posteriormente por el analista.

- Tarea 5: Modelar en UML

  Basados en la información obtenida en la realización de las tareas previas, se construirá un modelo visual, con el agregado de los aspectos finalmente seleccionados por el analista y las posibles situaciones conflictivas entre los aspectos candidatos.

Es de mayor interés en este trabajo analizar sólo las tareas que son automatizadas por la herramienta Aspect Extractor Tool de manera tal que minan las especificaciones de requerimientos, las cuales en el modelo de ingeniería propuesto se corresponden a las sub-tareas 1.1, 1.2 y 2.1. Estas, se analizan en detalle a continuación:

III.1.5.1 Tarea 1: Identificar concerns

Esta tarea consiste en identificar a partir de los requerimientos, los posibles concerns de un sistema. Para especificar estos requerimientos existen diferentes tipos de técnicas; en este caso se ha elegido para este enfoque la utilización de casos de uso, dado que UML [8] puede hoy considerarse un estándar. Además, es importante que la entrada no sea un documento de texto sin estructura, ya que el análisis sería más difícil dado que pueden existir documentos muy informales que no sirvan verdaderamente o dificulten el tratamiento, e infinidad de problemas que pueden surgir al no estandarizar la entrada.

La tarea de identificación de concerns, está dividida en dos sub-tareas: analizar la lista de casos de uso y analizar información extra provista por el analista. En ambas sub-tareas se genera automáticamente información de interés que se almacena en una tabla que se utilizará en las tareas restantes. De todos modos, algunas de las columnas se especificarán en tareas posteriores; por ejemplo, Responsabilidad que se completa en la tarea 3.

En dicha tabla (Tabla III-2), por cada concern identificado automáticamente, se especifican:

- Concern candidato: nombre del concern.
- Caso de uso: caso de uso al que pertenece o con el que está relacionado.
- Actor: actor relacionado con dicho concern.
- Responsabilidad: responsabilidad de dicho concern, es decir las obligaciones que tiene que cumplir, causas de su creación, etc.

Concern candidato	Casos de uso	Actor	Responsabilidad
Nombre del concern	Lista de casos de uso afectados	Actor relacionado con tal concern	Obligaciones que debe cumplir

Tabla III-2 - Información generada por Aspect Extractor Tool

III.1.5.2 Sub-tarea 1.1: Analizar lista de casos de uso

Para identificar concerns, se analiza la especificación provista por el analista. Dicha especificación es modelada con casos de uso. La lista de concerns candidatos será producida automáticamente, utilizando la técnica de recuperación de información.

- Filtrado y estandarización de información

  La idea es utilizar la técnica de recuperación de información, y a partir de la utilización de la misma poder estandarizar la información. La recuperación de información es el conjunto de tareas o técnicas mediante las cuales el usuario localiza y accede a los recursos de información que son pertinentes para la resolución del problema planteado. En este caso el problema es representar los casos de uso por medio de palabras claves que permitan identificar los concerns.

  Para tal fin se combinan dos técnicas, las cuales no tienen en cuenta los posibles errores ortográficos del analista al momento de realizar la especificación de los casos de uso:

  - Stop-Words: Inicialmente se eliminan las denominadas *stop-words*, que son palabras qué, desde el punto de vista no lingüístico, no contienen información relevante. Las stop-words difieren de acuerdo al lenguaje, en el caso del idioma Español podemos mencionar artículos, preposiciones, etc.
  - Stemming: Dado que diferentes variantes de una misma palabra pueden ser problemáticas al realizar análisis de texto porque tienen diferente deletreo pero el mismo significado, por ejemplo: aprender, aprenden, aprendió, aprender, se decidió utilizar *stemming*, que es el proceso de transformar una palabra en su *raíz* (stem). Para tal ejemplo, todas esas palabras estarían identificadas por su raíz, es decir “aprend”.

- Recuperación de información

  La mayor parte de los modelos y técnicas empleados en recuperación de la información utilizan en algún momento comparaciones y/o recuentos de frecuencias de los términos que aparecen en los documentos y en las consultas. Esto implica la necesidad de normalizar dichos términos, de manera que las comparaciones y/o recuentos puedan efectuarse de manera adecuada. Se tiene el caso de las palabras derivadas del mismo lema, a las que cabe atribuir un contenido semántico muy próximo.

  Las posibles variaciones de los derivados, junto con formas flexionadas, alteraciones en género y número, etc. hacen aconsejable un agrupamiento de tales variantes bajo un único término. Lo contrario produce una dispersión en el cálculo de frecuencias de tales términos, así como la dificultad de comparar consultas y documentos.

  La idea principal del trabajo es encontrar aspectos candidatos, los cuales serán deducidos a partir de la obtención de los diferentes tipos de concerns. Debido a ello, es que se eliminan en una primera instancia, todos los términos irrelevantes que se encuentran en el interior de un caso de uso, para lo cual se compara dicho caso de uso contra la lista de stop words. Una vez eliminadas las stops words, queda como resultado una lista de palabras que representan al caso de uso. Posteriormente, a cada una de las palabras de dicha lista se le aplica el algoritmo de stemming, con lo cual se obtendrá la raíz de la misma, clasificándola en verbo o sustantivo para lo cual se utiliza un diccionario de verbos.

  Asimismo, tal lista es filtrada para representar verbos y sustantivos diferentes que tienen el mismo significado semántico en el contexto, como por ejemplo la misma acción, que es el caso de los verbos levantar y recoger. Es decir, una vez que se llevó a la palabra a raíz, se verifica si existe un sinónimo de la misma. Si así fuera, se reemplaza por la raíz de dicho sinónimo para unificar vocabulario y no perder información o que el enfoque identifique dos aspectos candidatos cuando en realidad hay uno solo.

Al final de estos pasos, cada caso de uso quedará representado por dos listas. Una de ellas es una lista de verbos (raíces de los verbos), y la otra es una lista de sustantivos (raíces de sustantivos).

Una vez que se aplicaron dichas técnicas, se facilita la búsqueda de palabras similares en el interior de los distintos casos de uso.

III.1.5.3 Sub-tarea 1.2: Analizar información extra provista por el analista

El analista suministrará cierta información al sistema en forma de palabras claves, y requerimientos no funcionales, con el propósito de enriquecer la información almacenada en el sistema para la obtención de los concerns. Las palabras claves pueden representar un aspecto en el sistema ya sea funcional o no funcional. El sistema comienza con un conjunto de palabras predefinido al cual el analista le podrá agregar nuevas o eliminar existentes. La librería de palabras claves irá aumentando su tamaño a medida que el analista ingrese más palabras. Un beneficio que surge a partir de esta modalidad es que las palabras agregadas serán familiares al vocabulario que utilice el analista en sus especificaciones de casos de uso. De este modo, la posibilidad de matching entre dichas palabras y los casos de uso será mayor.

- Palabras claves funcionales y no funcionales

  Se podrán suministrar dos tipos de palabras claves. Por un lado se pueden suministrar palabras relacionadas con requerimientos no funcionales, tales como seguridad, persistencia, performance. Luego, se buscan ocurrencias de las mismas dentro de los casos de uso y en caso de encontrarse, se las sugiere como crosscutting concern. Por otro lado, se pueden suministrar palabras claves relacionadas con cierta funcionalidad que se pueda reutilizar, y de este modo encontrar functional crosscutting concerns. Por ejemplo, métodos, ordenamiento, visualización, técnicas, etc. El problema que surge en este punto es que a veces estos concerns se encuentran implícitos y su detección automática se torna difícil. Cuando el analista ingresa una palabra, se comprueba que no exista, se la reduce a su raíz y se almacena la palabra original y su raíz en la base de datos.

- Requerimientos no funcionales del sistema

  El otro tipo de información que el analista tendrá posibilidad de suministrar son los requerimientos no funcionales del sistema. Para identificar los concerns a partir de ellos, se toma un requerimiento no funcional por vez y se analiza sus influencias crosscutting sobre los casos de uso. Es decir, primeramente el analista ingresa un requerimiento no funcional del sistema. Este es almacenado, luego se le eliminan las stop words, y posteriormente se le realiza un stemming. De este modo, el requerimiento no funcional quedará representado por sus palabras relevantes. El análisis de sus influencias crosscutting estará dado por la búsqueda de ocurrencias de las palabras que representan el requerimiento no funcional y las palabras que representen al caso de uso.

III.1.5.4 Tarea 2: Elección de aspectos candidatos

Esta tarea se subdivide en dos sub-tareas: Identificar crosscutting concerns y functional crosscutting concerns, y elección de aspectos candidatos por parte del analista. Solamente la primera es de interes para este trabajo.

Sub-tarea 2.1: Identificar crosscutting concerns y functional crosscutting concerns

Existen dos formas de identificar crosscutting concerns. La primera de ellas se da cuando un concern corta transversalmente más de un elemento del modelo (casos de uso), para lo cual se observa si el tamaño de la lista de casos de uso de la columna Casos de uso de la Tabla 3.1 es mayor que uno. En este punto, la lista concerns candidatos será presentada al analista quien deberá eliminar aquellos que considere irrelevantes. La segunda forma se da cuando se identifica un caso de uso que cuenta con un requerimiento especial (no funcional) en su especificación. Dicho requerimiento no funcional será un concern candidato.

Para identificar functional crosscutting concerns, se debe verificar si alguno de los concerns se corresponde con las palabras claves que el sistema tiene almacenadas para tal fin. Una forma adicional de detectar concerns candidatos, ya sea funcionales o crosscutting, se da solamente si el sistema hubiese sido utilizado con anterioridad y se hubiesen identificado aspectos candidatos. Dichos aspectos, fueron almacenados en la base de datos para ser reutilizados posteriormente. Si alguno de los concerns se corresponde con alguno de los aspectos, entonces dicho concern será un concern candidato y su tipo dependerá del mismo tipo que el aspecto. Es decir, si el aspecto es crosscutting, el tipo del concern candidato también será crosscutting. De esta manera, se reutilizan los aspectos y el conocimiento del analista. A medida que se utilice el sistema, dicho conocimiento es incrementado, ayudando así a la identificación de aspectos.

III.2 - Determinación de criterios de comparación

Para poder realizar una comparación práctica y objetiva, en esta sección se describirán los criterios que establecimos, los cuales permitirán evaluar y medir las características provistas por cada una de las técnicas de identificación de concerns crosscutting en especificaciones de requerimientos. Los criterios son explicados a continuación:

- Dependencia de la estructura: Analiza sobre qué tipo de documentos es aplicable el enfoque. Cada técnica puede llegar a tener limitaciones sobre la estructura de las especificaciones de requerimientos que son usadas como entrada para la herramienta de minería, o no tener ninguna y poder trabajar sobre cualquier tipo de documento independientemente de la organización de la información.
- Nivel de automatización: Se refiere al grado de interacción que tiene el desarrollador con la herramienta. Como generalmente las herramientas son del tipo semi-automatizadas, este criterio se enfoca principalmente en si es necesario tener un conocimiento previo de los documentos de requerimientos o del sistema en si, sí es necesario ingresar información adicional con respecto al sistema, la cantidad de etapas donde el desarrollador debe tomar decisiones, entre otras.
- Tipo de análisis de la documentación: Clasifica el análisis que es realizado a la información suministrada por los documentos de requerimientos, según si este tiene un nivel léxico, sintáctico, o semántico. El nivel léxico se logra cuando se analizan las palabras, el sintáctico cuando se analiza la estructura de las sentencias escritas en los documentos, y el semántico cuando se interpreta el significado de las palabras según el contexto donde se encuentren.
- Escalabilidad: Describe las capacidades que proveen las herramientas, si las hay, para poder aplicar los enfoques a problemas de gran tamaño, de qué manera lo realizan y qué tan útiles son para ayudar al desarrollador a la identificación de los concerns. Además, debe tenerse en cuenta si la herramienta es capaz de analizar grandes cantidades de información en un tiempo aceptable, es decir, que no se produzcan demoras de tamaño considerable al usar las técnicas sobre problemas de dimensiones amplias.
- Efectividad: Describe la precisión a la hora de determinar los crosscutting concerns, sean funcionales o no funcionales, en las especificaciones de requerimientos; es decir, el grado de completitud con el que la herramienta detecta los aspectos. Esto se refleja principalmente en aquellos concerns que están demasiado ocultos y dispersos en las especificaciones de requerimientos, que son difíciles de discernir debido a las vaguezas del lenguaje natural (sinónimos, ambigüedades, etc.), o que son dependientes del dominio del sistema. Un ejemplo de estos son la gran mayoría de los crosscutting concerns funcionales, los cuales generalmente por ser funcionales, usan palabras que son difíciles de interpretar como un posible concern crosscutting de manera manual.
- Integrabilidad: Se refiere a la capacidad del enfoque analizado de acoplarse a otras técnicas para el modelaje y diseño de soluciones orientadas a aspectos, basadas en las salidas de las herramientas de minado de aspectos. También será tomada en consideración la inclusión de una asistencia para el modelado dentro de la herramienta, o si sugiere alguna metodología en particular para llevar a cabo las etapas posteriores del desarrollo del sistema.
- Trazabilidad: Especifica el grado de trazabilidad provisto por la herramienta para realizar un seguimiento de los requerimientos crosscutting y sus influencias sobre los demás requerimientos. Debe ser tenido en consideración que tanta trazabilidad es provista por el enfoque, por ejemplo si provee la navegación entre los requerimientos por sus relaciones.

- Visualización: Clasifica las diferentes herramientas analizadas según la forma de representar los concerns, de qué manera lo hacen, y analiza las ventajas y desventajas de su uso. También debe tenerse en cuenta la posibilidad de generar vistas intermedias para llevar a cabo la identificación de concerns, la posibilidad de generar modelos intermedios que ayuden a refinar los documentos de requerimientos existentes, y la salida de modelos aspectuales que representen los diferentes concerns crosscutting detectados y sus influencias sobre el resto de los concerns del sistema.
- Velocidad: Se refiere a la medición del tiempo empleado para llevar a cabo el análisis de cada una de las técnicas, para poder compararlas tanto con respecto al análisis manual como con respecto a las otras técnicas. Esta métrica esta estrechamente relacionada con el criterio de nivel de automatización, pero es de nuestro interés debido a que no necesariamente un enfoque más automatizado debería ser más rápido de aplicar que otro menos automatizado. Es de vital importancia analizar la velocidad, debido a que el ahorro de tiempo es el factor principal a la hora de seleccionar una técnica u otra.
- Evolución: Representa el grado de adaptabilidad, tanto de los enfoques como de las herramientas proporcionadas, para soportar el cambio de los requerimientos a través del tiempo. Esto implica analizar los efectos que se producirían sobre la información previamente generada al evolucionar los requerimientos del sistema, incluyendo la inclusión, exclusión, refinamiento o modificaciones de gran porte sobre estos. En gran parte esta relacionado con el criterio de trazabilidad previamente explicado, pero este criterio esta enfocado en cómo podría manejarse un cambio en los requerimientos desde la herramienta, la asistencia proporcionada al desarrollador para llevarlo a cabo y las virtudes de estas.

III.3 - Realización de la comparación

En esta sección se analizan cada una de las técnicas o enfoques, reflejando sus ventajas y desventajas, teniendo en cuenta las diferencias de unas respecto a las otras. Estas fueron enfrentadas teniendo en mente cada uno de los criterios de comparación previamente descriptos.

III.3.1 - Dependencia de la estructura

Theme/Doc, si bien no especifica explícitamente que debe aplicarse a documentos de requerimientos estructurados, presupone que está descripto de tal manera que se amolda al analizador léxico de la herramienta. Esto implica que no se puede aplicar a dominios de desarrollos centrados en la documentación, donde la gran mayoría de los requerimientos son obtenidos de varias fuentes como entrevistas con los stakeholders, documentos legados, y extensos documentos de requerimientos (miles de páginas textuales).

La técnica de IR propuesta por Rosenheiner, si bien se aplica a especificaciones tradicionales de requerimientos (similares al texto natural), estos son estructurados (se dividen en subsecciones especificas para cada parte del sistema), y por lo general el ingeniero debe determinar y restringir sobre qué secciones del documento debe aplicarse.

La técnica utilizada por EA-Miner, gracias al uso del NLP, permite identificar efectivamente los aspectos en la documentación de requerimientos, sin importar que tan complejos sean (por ejemplo si es el documento de una reglamentación o una ley) o si son no estructurados.

La técnica que utiliza los AOIG para identificar concerns crosscutting en los comentarios del código fuente, al igual que EA-Miner, se puede aplicar a cualquier tipo de documento textual de requerimientos sin importar su estructura, debido a que utilizan un NLP.

Finalmente, en la herramienta Aspect Extractor Tool, queda establecido en el enfoque que debido a la dificultad de tratar con textos sin estructura, es necesario estandarizar la entrada permitiendo solo la utilización de casos de uso y sus respectivas especificaciones (debido al nivel de aceptación que tiene UML) para realizar la identificación de aspectos.

III.3.2 - Nivel de automatización

La herramienta Theme/Doc se basa en un esquema semi-automatizado para la detección de comportamiento crosscutting. En este se necesita realizar un trabajo preparatorio, en donde el desarrollador debe ingresar como entrada a la herramienta no solo las especificaciones de requerimientos, sino también un conjunto de palabras clave (intuitivamente), por lo que éste debe leer y comprender la totalidad de la documentación antes de empezar el análisis.

En el enfoque que utiliza técnicas de IR para minar aspectos en documentos de requerimientos también presenta un esquema semi-automatizado, y aunque también es necesario realizar un trabajo preparatorio previo, no es necesario leer toda la documentación de requerimientos como en Theme/Doc, en cambio se debe proveer a la herramienta con ciertos datos (como el nombre del sistema) para realizar el análisis. Además, como se mencionó anteriormente, se deben limitar las búsquedas a aquellas secciones del documento que son de interés. Y finalmente, siendo la critica de mayor importancia, la búsqueda de concerns crosscutting está basada en suposiciones del analista sobre algún concern que cree que es crosscutting, de tal manera que la herramienta analiza solo aquellas partes que se refieren a esta suposición.

La herramienta EA-Miner carece de los problemas anteriores, debido a que el uso del NLP permite que haya que ingresar como entrada solamente los documentos de requerimientos. Además, la identificación de los posibles aspectos es realizada de forma completamente automatizada. Adicionalmente de identificar los concerns, la herramienta también automatiza la detección de viewpoints y action words. Los únicos pasos en los que el analista interviene son en la generación del modelo intermedio y en la determinación de cuales concerns, por su relevancia en el sistema, deben ser modelados como aspectos. También cabe destacar que se provee información extraída por el catalogo de NFR.

La herramienta CCVerbFinder, al igual que EA-Miner, presenta un esquema semi-automatizado, que basándose en el AOIG generado por la herramienta AOIGBuilder (la cuál utiliza un NLP), permite minar concerns (tanto los que se saben que existen como aquellos que se desconocen) que están dispersos. Esta genera como salida una lista de posibles concerns candidatos a convertirse en aspectos. El analista sólo interactúa decidiendo cuáles de ellos serán convertidos.

Por último, en el enfoque Aspect Extractor, las tareas de identificación de aspectos candidatos son semi-automatizadas, pero se debe agregar información extra suministrada por el analista para mejorar la detección. Se deben agregar a la entrada de la herramienta un conjunto de palabras claves funcionales y no funcionales (de forma similar a Theme/Doc), como también requerimientos no funcionales del sistema (similar a la técnica de IR), para poder realizar un análisis efectivo.

III.3.3 - Tipo de análisis de la documentación

Tanto Theme/Doc y Aspect Extractor consisten en un análisis léxico de la información del texto de la documentación de requerimientos, en las que se buscan determinadas palabras claves provistas por el desarrollador.

La técnica de IR esta basada en la búsqueda de aspectos específicos, los cuales son determinados por el analista. Para llevarse a cabo, se utiliza un análisis sintáctico (técnicas de pattern matching) para obtener los resultados de las búsquedas.

En EA-Miner y CCVerbFinder, en cambio, se realiza un análisis léxico, sintáctico producto de la utilización del NLP, y EA-Miner agrega un análisis semántico del las especificaciones de requerimientos.

III.3.4 - Escalabilidad

El enfoque Theme/Doc clasifica las acciones en dos tipos, mayores y menores. De esta manera provee dos niveles de zoom para las vistas de acción: se puede ver todas las acciones de la vista, o solo aquellas que sean mayores. Sin embargo, cuando el tamaño del sistema se hace muy grande, este esquema de dos niveles puede llegar a ser insuficiente o incluso infactible. Se ha investigado bastante en [16] sobre el uso de múltiples pistas en los documentos de requerimientos para ayudar a escalar las vistas de concerns. Sin embargo, no se han llegado a conclusiones concisas, sino que los autores dejaron abiertas las interrogantes planteadas.

La técnica de IR provee cierto nivel de escalabilidad, ya que como su objetivo no es encontrar todos los concerns crosscutting sino centrarse de a uno por vez, permite que esta se aplique bien a sistemas voluminosos.

La herramienta EA-Miner no tiene problemas de escalabilidad, y puede aplicarse indiferentemente a sistemas chicos como grandes, debido a que la técnica solamente hace un análisis del texto, sin proveer ningún tipo de vistas como Theme/Doc. Además, la herramienta suministra características de filtrado de las listas generadas, por ejemplo, excluyendo aquellos verbos que no representan ninguna acción como los verbos auxiliares (“is, “are”, “be”), o incluso se podría filtrar la lista para que muestre aquellos verbos que tengan el mismo significado semántico en el contexto que los rodea. Estas características permiten que el analista pueda, en un sistema con un gran número de requerimientos, limite la presentación de las listas a solamente aquellos verbos los cuales esta interesado.

CCVerbFinder de manera similar a EA-Miner no tiene problemas de escalabilidad, aunque difiere en que no tiene ningún tipo de forma de filtrar la información.

La herramienta Aspect Extractor Tool, si bien no presenta problemas de escalabilidad a la hora de realizar el análisis, puede llegar a ser tedioso para el analista cuando este tiene que seleccionar entre los aspectos candidatos, ya que esta lista crece hasta un tamaño inmanejable.

III.3.5 - Efectividad

La herramienta Theme/Doc ha probado ser más efectiva que los análisis manuales para la detección de aspectos, principalmente debido a que subsana las propias ambigüedades del lenguaje

natural. Los problemas causados por los sinónimos fueron resueltos mediante un diccionario de sinónimos. Sin embargo, esto no fue suficiente para tratar con palabras iguales con diferentes significados. También se comprobó que puede identificar correctamente las ambigüedades encontradas en los requerimientos, que son visualizadas en las vistas y posteriormente corregidas en la etapa de refinamiento. Además, soporta efectivamente la resolución de conflictos entre temas crosscutting, estableciendo un orden entre ellos. Varios autores están de acuerdos en que el enfoque de Theme/Doc es altamente efectivo cuando se desea lograr un alto grado de completitud en la identificación de todos los concerns y sus relaciones.

En la técnica de IR, el algoritmo probó tener una alta precisión y recall en los dos casos de estudios que se analizaron por los autores. Además, luego de un análisis posterior, se determinó que los errores que habían ocurrido, el 67% de estos fueron causados por malas decisiones del analista (por ejemplo, descartar aspectos candidatos que en realidad debían ser tomados en cuenta). No hay ninguna consideración por los sinónimos ambigüedades, y especificaciones de mala calidad.

Se ha determinado que la herramienta EA-Miner es muy efectiva para la identificación de aspectos tempranos, principalmente causado por la alta precisión del etiquetado sintáctico y semántico (98% y 91% respectivamente) de WMATRIX. La información generada por WMATRIX es complementada con un léxico propio para concerns crosscutting y modelos de representación de documentos de requerimientos. A diferencia de Theme/Doc, este enfoque diferencia las heurísticas de identificación de concerns crosscutting, entre aquellos que son funcionales de los que no. Para la identificación de concerns crosscutting no funcionales el léxico de la herramienta se basa en los árboles de requerimientos no funcionales del Framework de NFR, y se permite agregar información nueva a este léxico; por lo que la identificación de estos es muy efectiva. Para la identificación de concerns crosscutting funcionales se utiliza un esquema similar al mostrado en Theme/Doc, el cual también tiene una alta precisión. En contraste a Theme/Doc, aunque EA-Miner también usa un diccionario de sinónimos para evitar los problemas que causan, no tiene el problema de las palabras iguales con diferentes significados, debido a que el NLP permite realizar un análisis contextual más sensitivo de las palabras reconociendo estas palabras en diferentes campos semánticos. También se realizan técnicas de IR como la lematización (stemming) para aumentar la precisión y se eliminar palabras sin significado semántico (stop-words); y además se analizan las frecuencias de los verbos o sustantivos para considerar solamente aquellos que sean significativos y reducir el “ruido”. En las evaluaciones de la herramienta EA-Miner en [23], se obtiene como resultado que es efectiva tanto para la identificación de viewpoints (100% con respecto al análisis manual de un desarrollador senior) como también para la de aspectos (casi 100%). Además, se expone que el error de la identificación es casi nulo.

CCVerbFinder, de manera similar a EA-Miner es muy efectivo a la hora de etiquetar las palabras, debido al uso del NLP. Sin embargo, difiere con respecto de la técnica de identificación de aspectos, ya que propone una estrategia de pattern matching para encontrar pares de verbo-objeto directo que estén repetidos en los comentarios del código fuente, y según la evaluación de sus autores tiene una precisión del 97% y un recall del 78%. Este es menor al de EA-Miner debido a la simpleza del extractor de patrones utilizado en el análisis de los comentarios y los problemas generados por no considerar las ambigüedades del lenguaje. También es necesario destacar que los resultados del análisis, es decir, los aspectos candidatos, son organizados en un ranking que los ordena desde aquellos que son más crosscutting a los que son menos.

La herramienta Aspect Extractor Tool utiliza una estrategia basada en la IR la cual permite identificar los aspectos candidatos observando aquellas repeticiones de palabras en las especificaciones. Utiliza un diccionario de sinónimos para estandarizar las palabras, se aplican las técnicas de stemming y stop-word para aumentar la eficacia. Sin embargo, este análisis que se realiza sobre los documentos esta menos refinado (tiene problemas como Theme/Doc) y a una gran distancia de la complejidad algorítmica de las otras herramientas. Entre las características adicionales de Aspect Extractor, con respecto a los demás enfoques, podemos considerar que permite reutilizar el conocimiento previo de otros sistemas analizados, permite analizar los conflictos generados entre los aspectos, y permite discernir entre aspectos funcionales y no funcionales (como EA-Miner).

III.3.6 - Integrabilidad

El enfoque Theme, al utilizar UML para diseñar la solución de los temas, permite que los resultados generados se integren al gran conjunto de herramientas disponibles para el estándar UML. De manera similar, aunque enfatizando esto un poco más, la herramienta Aspect Extractor Tool se integra de manera similar a UML, pero este restringe que las entradas sean diagramas y especificaciones de casos de uso, integrándose como un paso más en el desarrollo de software con UML.

En el enfoque de IR, no hay ningún tipo de mención a la integración de este en algún tipo de metodología o proceso de desarrollo. Simplemente sirve solo para identificar aspectos y esta pensado como una actividad complementaria para el análisis de sistemas legados.

El proceso AORE, donde se encuentra incluido EA-Miner, no menciona la integrabilidad con ninguna metodología ni proceso de desarrollo, ya que esta pensado para que se adapte a cualquiera de estos, sin importar cual sea.

CCVerbFinder, al estar su análisis centrado en el código fuente, no propone ningún tipo de integrabilidad.

III.3.7 - Trazabilidad

Theme/Doc provee trazabilidad desde los documentos de requerimientos pasando por las vistas de Theme/Doc, hasta el modelo de diseño realizado en Theme/UML, explicitando los enlaces entre las porciones de los documentos de requerimientos relevantes a las vistas provistas por el enfoque.

La técnica de IR y la herramienta EA-Miner no proveen ningún tipo de trazabilidad ni a los requerimientos ni a los artefactos desarrollados posteriormente.

CCVerbFinder al estar enfocada para su uso en código, tampoco provee ningún tipo de trazabilidad hacia otros artefactos, ya que se realiza desde las últimas etapas del desarrollo.

La herramienta Aspect Extractor Tool, debido a las características del enfoque, permite utilizar la trazabilidad que provee UML para el desarrollo. Los aspectos son integrados en un diagrama de casos de uso extendido (la salida de la herramienta) y estos son especificados formalmente, analizando las relaciones entre aspectos y casos de uso, analizando los conflictos entre aspectos, etc. Para integrarse al modelo de UML se aplica un profile de UML para aspectos.

III.3.8 - Visualización

Theme/Doc provee una serie de vistas que representan el texto de las especificaciones de requerimientos, exponiendo las relaciones entre los comportamientos del sistema. Estas vistas asisten al desarrollador a determinar cuáles elementos de la funcionalidad son aspectos y cuáles de estos son base. Desde un punto de vista más global, estas vistas suministran una visión orientada a características del conjunto de requerimientos. Todas estas vistas son mapeadas a las vistas de diseño de UML mediante Theme/UML.

La técnica de IR no provee ningún tipo de visualización o modelos intermedios.

EA-Miner permite (opcionalmente) la generación de modelos de requerimientos intermedios, de tal manera de producir una descripción más estructurada, visualmente, del análisis realizado por la herramienta. Estos no están limitados a ninguna metodología en particular, soportando viewpoints, casos de uso, etc. Al final del análisis se genera un modelo aspectual, el cual incluye todos los concerns, aspectos tempranos y sus relaciones.

El enfoque de AOIG presenta al grafo de AOIG para realizar la identificación de aspectos, aunque no es necesario utilizarlo para llevarla a cabo debido a que la herramienta CCVerbFinder lo utiliza para detectar aspectos de manera automática.

La herramienta Aspect Extractor Tool esta basada en la suposición de que los requerimientos están modelados en UML por medio de diagramas y especificaciones de casos de uso (los cuales son la entrada de información), y la salida de la herramienta es un diagrama de casos de uso extendido, el cual representa a los concerns crosscutting identificados como aspectos en el diagrama de casos de uso y a sus relaciones e influencias sobre los casos de uso y los demás aspectos.

III.3.9 - Velocidad

Si bien no hay hechos ningún tipo de comparación entre las herramientas, si hay unos cuantos trabajos cuya investigación se centró en comparar las ganancias de tiempo obtenidas entre la identificación de aspectos de forma manual vs. la identificación semi-automatizada [15, 17, 22, 47, 52].

Theme/Doc permite la automatización de varias tareas consumidoras de tiempo que permiten identificar los concerns de un sistema. Sin embargo, el hecho de que se tenga que leer y comprender toda la documentación para ingresar como entrada una lista de palabras clave, lo relega en el ahorro de tiempo con respecto a los demás enfoques. Igualmente, sigue siendo mucho más rápido que el análisis manual en busca de crosscutting concerns.

La técnica de IR es analizada en el caso de estudio provisto en el mismo enfoque [45], donde se muestran ganancias de tiempo del 35% al 60%, dependiendo principalmente de la longitud de los documentos de requerimientos analizados (a mayor longitud de texto se ahorra una mayor cantidad de tiempo). Además, por las características propias de la técnica, es la más rápida para analizar las influencias de un requerimiento crosscutting cuando el analista sabe cuál es (es decir, que ya esta identificado).

Estudios llevados a cabo por los autores de EA-Miner han probado que el uso de la herramienta WMATRIX no es un cuello de botella cuando se utilizan grandes volúmenes de documentación. Este análisis puede ser llevado a cabo en un tiempo reducido, creciendo este de forma lineal al tamaño de la documentación. Además, debido a que EA-Miner automatiza las actividades más consumidoras de tiempo, como la identificación de concerns, viewpoints y action words, se puede llegar a decir que es el más rápido de los enfoques presentados. En la evaluación de la herramienta EA-Miner en [23], se obtiene como resultado que la ganancia de tiempo del enfoque semi-automatizado (en el caso de estudio propuesto) es acerca de 20 veces más rápido, con respecto a un análisis manual.

En la herramienta presentada en el enfoque de AOIG, tanto el costo de construcción del AOIG como el recorrido de este para identificar los aspectos son lineales con respecto al número de

nodos en el grafo. Esto permite obtener una ganancia de tiempo similar a la expuesta en EA-Miner, ya qué todas las tareas son realizadas de manera automática, excepto la selección de qué aspectos candidatos serán considerados concretos.

El enfoque Aspect Extractor, de manera similar a Theme/Doc, tiene una ganancia de tiempo menor que los demás enfoques debido a que se debe ingresar información previa para realizar el análisis de identificación. Sin embargo, debido a que no es tan efectiva y se genera mucho ruido en los aspectos detectados, puede que el analista tarde todavía un poco más que Theme/Doc.

III.3.10 - Evolución

Theme/Doc si bien habla de la evolución de los requerimientos del sistema, no provee soluciones concretas, ya que pretende que se regeneren todas las vistas para llevar a cabo el análisis de impacto de los cambios.

Tanto la técnica de IR como la que usa AOIG, no tienen ningún tipo de consideración por los cambios no anticipados de los requerimientos.

En la presentación del enfoque AORE no se hace ninguna mención sobre cómo se manejaría el impacto de cambios en los requerimientos, ni en el proceso ni en la herramienta EA-Miner, aunque esto no significa que se pueda plantear y analizar algún tipo de propuesta para tenerlos en cuenta.

La herramienta Aspect Extractor Tool, al estar basada en UML, podría llegar a ser complementada con alguna propuesta para administrar los cambios de los requerimientos. Sin embargo, en su estado actual, la herramienta tendría que regenerar todo el análisis para llevarlo a cabo.

III.4 - Conclusiones de la comparación

Si bien no es crucial que una herramienta pueda tratar con documentos no estructurados, ya que se pueden evitar problemas y ganar información al limitarse a algún tipo de metodología que utilice especificaciones estructuradas, si es notorio que al utilizar las técnicas de NLP (EA-Miner, AOIG) estas consiguen un nivel de conocimiento del texto de los documentos que ninguna de las otras técnicas puede lograr. Además, permite que no haga falta ni tener que leer los documentos ni ingresar datos adicionales.

Los enfoques como Theme/Doc, la técnica de IR y Aspect Extractor tienen problemas con respecto a la automatización debido a que es necesario que se ingrese información previa para llevar a cabo la exploración de posibles aspectos, y estas dependen de la calidad de esta información para realizar un buen análisis (es decir, la herramienta depende de la capacidad del analista que lo usa, y no de la herramienta en si misma). En contraste, las técnicas que utilizan los NLP pueden extraer esta información de manera automática, y de forma precisa evitando depender de la capacidad del desarrollador.

La capacidad de realizar un análisis semántico de la información, provee un mayor conocimiento de los requerimientos. Una de las características más importante de éste es que evitan problemas de otros tipos de análisis (léxico y sintáctico) al agrupar palabras con el mismo significado en el contexto de los requerimientos del sistema.

Las técnicas que basan su análisis de manera visual tienen problemas de escalabilidad debido a que no pueden proveer una manera concreta de reducir o aumentar el nivel de zoom. Las que se basan en un análisis automatizado sobre el texto, carecen de este problema, pero deben tener cuidado con la cantidad de información que presentan, ya que por lo general son de gran tamaño e intratables para el desarrollador. La única forma de solucionar esto es proveer alguna manera de filtrarla mediante reglas o agrupar la información semánticamente relacionada.

El uso de técnicas semi-automatizadas para la detección de aspectos, en todos los enfoques ha logrado resultados efectivos, con una alta precisión y recall. Esto esta relacionado con que los documentos de requerimientos, generalmente son significativamente aspectuales.

La integrabilidad de los enfoques en los procesos de desarrollo utilizados actualmente es un gran problema que tienen estas técnicas, excepto Theme/Doc y Aspect Extractor Tool, debido a que no queda claro como se acoplarían al desarrollo de un sistema. La excepción son las herramientas que están restringidas a utilizar UML, y si bien pierden aplicabilidad a otros sistemas (no basados en UML, como viejos sistemas legados), ganan a la hora de la integración en un proceso de desarrollo actual.

Ninguna de las herramientas enfatiza el tópico de trazabilidad, a excepción de aquellas basadas en UML, las cuales proveen el seguimiento propuesto por este estándar.

La representación visual de los aspectos es una parte muy importante de los enfoques, los cuales en su salida deben presentar un modelo aspectual de todos los concerns del sistema y sus relaciones. Es importante considerar que muchas de las técnicas que utilizan NLP, las cuales han probado ser las más efectivas, no enfatizan sobre el uso de modelos de representación visuales.

La velocidad de detección de aspectos cuando se compara con una inspección manual de las especificaciones, ha sido comprobado mediante la experimentación, que son altamente veloces. Sin embargo, no hay ningún tipo de análisis entre las herramientas entre si. Igualmente se presupone que aquellas técnicas más automatizadas, que no requieren que se ingrese información adicional para la realización de la identificación, deberían ser las más veloces.

Ninguna de las herramientas presentadas consideró en profundidad el análisis de la evolución no anticipada de los requerimientos.

Cabe remarcar que aunque se realizó una comparación entre los cinco enfoques presentados, esto no es precisamente justo, ya que el objetivo de la comparación es determinar las mejores características de estos para ampliar y perfeccionar la técnica de detección de aspectos de la

herramienta Aspect Extractor Tool, pero cada uno de ellos tienen objetivos diferentes y cada uno se compromete para cumplir estos objetivos y no precisamente los que son relevantes para este trabajo en particular.

A continuación, para obtener una visión más global, se resume la información obtenida de la comparación de las técnicas analizadas en forma de una matriz (Tabla III-3).

Criterio / Enfoque	Theme/Doc	Técnica de IR	EA-Miner	CCVerbFinder (AOIG)	Aspect Extractor
Dependencia de la estructura	Estructurados (textos amoldados a la herramienta).	Estructurados en secciones.	Independiente de la estructura.	Independiente de la estructura.	Estructurados (especificaciones de casos de uso).
Nivel de automatización	Entrada de información adicional (se debe leer toda la documentación).	Entrada de información adicional (no se debe leer toda la documentación) Las búsquedas se basan en suposiciones del analista.	No necesita la entrada de información adicional. No depende del nivel de conocimiento del analista.	No necesita la entrada de información adicional. No depende del nivel de conocimiento del analista.	Entrada de información adicional.
Tipo de análisis	Léxico.	Léxico y sintáctico.	Léxico, sintáctico y semántico.	Léxico, sintáctico.	Léxico.
Escalabilidad	Semi-escalable (tiene problemas para manejar el tamaño de las vistas)	Escalable	Escalable (soporta filtros para manejar grandes cantidades de información)	Escalable (no tiene soporte para manejar grandes cantidades de información)	Escalable (no tiene soporte para manejar grandes cantidades de información)
Efectividad	Efectivo para encontrar todos los concerns de un sistema. Problemas con palabras semánticamente similares.	Efectivo para encontrar concerns particulares de un sistema. Problemas con sinónimos, ambigüedades y malas especificaciones.	Muy efectivo No tiene problemas. Discierne entre aspectos funcionales y no funcionales	Muy efectivo No discierne entre aspectos funcionales by no funcionales Problemas con sinónimos, ambigüedades y malas especificaciones.	Poco efectivo Problemas con palabras semánticamente similares Discierne entre aspectos funcionales y no funcionales
Integrabilidad	Se integra al diseño con UML.	No se integra.	Esta pensado para ser integrable con cualquier metodología o proceso.	No se integra.	Se integra con UML.
Trazabilidad	Poca.	No provee.	No provee.	No provee.	Suficiente.
Visualización	Vistas de requerimientos y de diseño.	No provee.	Vistas intermedias de requerimientos y vista de salida aspectual.	Vista del grafo AOIG. No provee vistas específicas.	Diagramas de casos de uso extendido (incluye los aspectos).
Velocidad	No tan rápida.	Rápida.	Muy rápida.	Muy rápida.	No tan rápida.
Evolución	Solo es considerado, pero no suministra soporte.	No considera.	No considera.	No considera.	No considera.

Tabla III-3 - Comparación de técnicas de identificación de aspectos

Estas herramientas fallan porque si bien en conjunto tienen muchas ideas interesantes y complementarias, por si solas todas carecen de alguna característica esencial para su uso en un desarrollo real (es decir, no un caso de estudio). Además, al no estar centrados los objetivos planteados por cada herramienta en los mismos que se persiguen en este trabajo, hay puntos donde todas las herramientas presentan falencias. Theme/Doc no realiza análisis sintáctico y semántico, no es demasiado escalable, tiene problemas con las ambigüedades, y necesita que el analista ingrese información previa. La técnica de IR no realiza un análisis semántico de las palabras, se debe ingresar alguna información previa, y esta diseñado para encontrar aspectos de a uno por vez (busqueda). EA-Miner no se integra con ninguna metodología en particular y no considera la trazabililidad ni la evolución de los requerimientos. CCVerbFinder no provee un análisis semántico, tiene problemas con

las ambigüedades, no se integra a ninguna metodología, no provee vistas específicas y no maneja apropiadamente la escalabilidad. Aspect Extractor Tool necesita que se ingrese información adicional de entrada, no realiza análisis sintácticos ni semánticos, no tiene soporte para la escalabilidad, es poco efectivo debido a las falencias de su técnica de identificación de aspectos.

Capitulo IV - Identificación de aspectos

En esta sección se estudiará en profundidad cuál sería el enfoque más adecuado para proponer una nueva técnica de identificación de aspectos que solucione los problemas anteriormente mencionados, y que reúna las mejores características estudiadas para poder realizar de una extensión de la herramienta Aspect Extractor Tool. Esta es de vital importancia, ya que se desea aumentar la probabilidad de que los aspectos identificados por la herramienta sean verdaderos aspectos candidatos, y no falsos positivos (ruido).

IV.1 - Desarrollo de la técnica de identificación de aspectos propuesta

En esta sección se definen las características necesarias para llevar a cabo los objetivos propuestos y las herramientas que permiten resolverlas adecuadamente, para finalmente llegar a la especificación del algoritmo de identificación de aspectos.

De acuerdo a las conclusiones arribadas de la comparación se determinó 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 e integrar algún tipo de metodología que se integre al proceso de desarrollo del sistema, de tal manera de tener una mayor visibilidad que al utilizar un enfoque genérico (que no detalla estas características), al mismo tiempo que provea de un conjunto de vistas que facilitan 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 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 dejen 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, siempre teniendo 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 alternativas:

- Utilizar una herramienta de NLP que realice el análisis (léxico y sintáctico) del texto. Este análisis etiqueta las palabras de cada sentencia (indicando si es verbo, sustantivo, etc.).
- Se realiza un análisis semántico de cada palabra en el cual se determina precisamente que sentido se le esta dando a la palabra según el contexto que la rodea.
- Se realiza un análisis de las palabras que permite agrupar las palabras identificadas con significados semánticos similares, utilizando diccionarios semánticos [12]. También son tenidos en cuenta diccionarios de sinónimos y algoritmos de stemming.
- Se debe contar con información estadística de cada una de las palabras, para poder establecer la relevancia de estas palabras.

La mayoría de los análisis están basados en la búsqueda de verbos para identificar comportamiento crosscutting. Sin embargo nos parece más adecuado y preciso identificar sobre que objeto actúan estas acciones o verbos, para aumentar la efectividad del enfoque (debido a que una acción, al ser aplicada sobre diferentes objetos, puede llegar a considerarse como varios

comportamientos diferentes, aunque solo sea un verbo el qué está en cuestión). 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 [52], buscando pares (verbo, objeto directo) para identificar las acciones clave. Al mismo tiempo, el uso de una estructura similar al grafo AOIG, nos permite realizar muchos tipos de análisis sobre los concerns, debido a la gran cantidad de información que provee.

Se utilizará UML como base de la aplicación de identificación de aspectos (esto cumple con la 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 y 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 aprovechara la estructura fija de las especificaciones de caso de uso para identificar aquel comportamiento crosscutting que deba ser considerado funcional o no funcional. Se considerara como no funcional cada verbo y objeto directo que se encuentre en la sección requerimiento especial (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 (Esto cumple la 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 desambiguación semántica de las palabras (satisfaciendo en parte la Característica IV). Primero, se realiza sobre los textos de los flujos básicos y alternativos de las especificaciones de casos de uso. Segundo, se realiza sobre los requerimientos no funcionales del sistema y sobre los requerimientos adicionales de cada uno 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 desambiguación semántica, se consideraron dos posibles alternativas: las técnicas de IR basadas en lenguaje supervisado y las bases de datos léxicas. Del análisis llevado a cabo, se opto por descartar las primeras, debido a que estas dependen en gran medida del entrenamiento que han recibido, y si bien con un buen entrenamiento los resultados son bastante mejores que las segundas, son extremadamente rígidas con respecto al uso de dominios particulares (donde no han sido entrenadas), mientras que las bases de datos léxicas se comportan manteniendo una precisión cuasi constante.

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

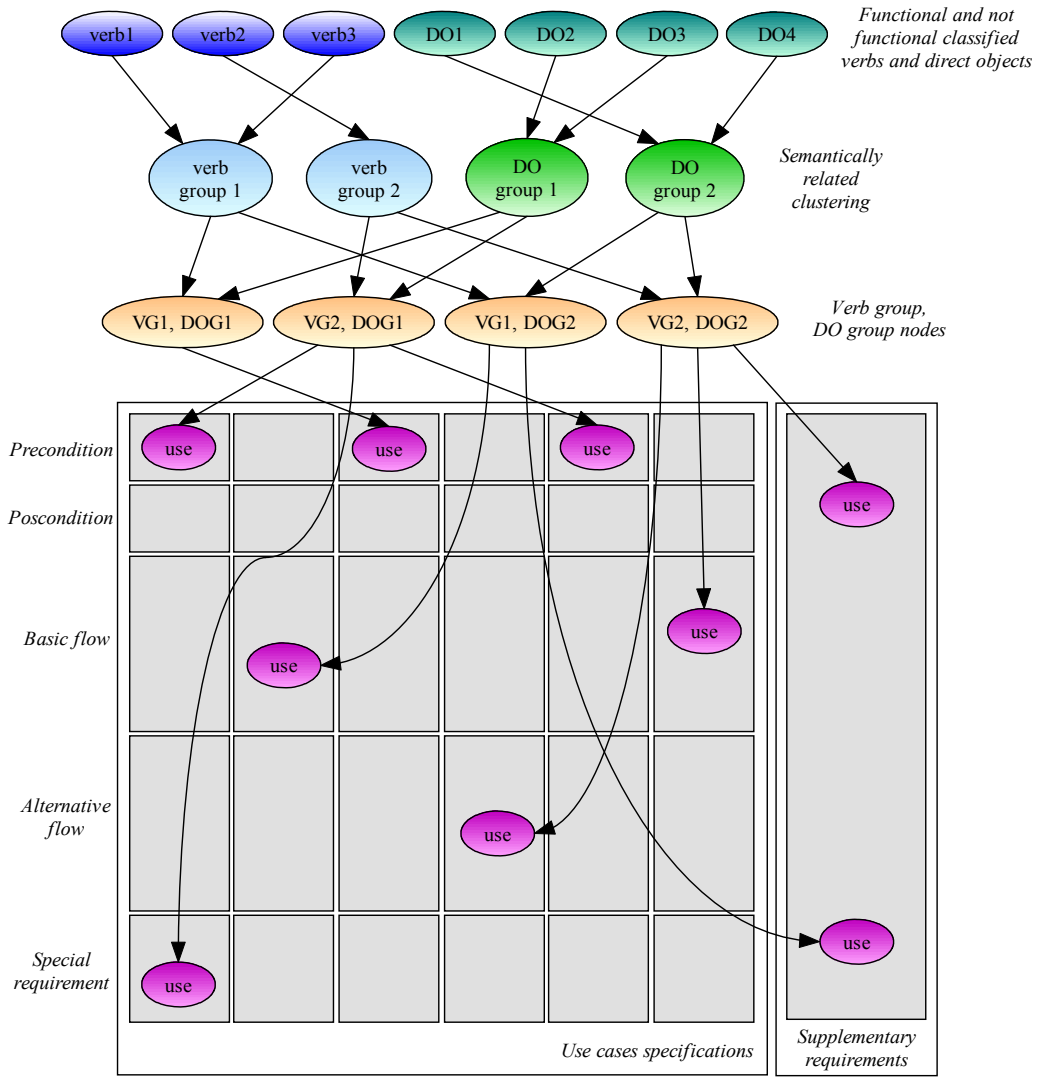

Ilustración IV-1 - Ejemplo del grafo propuesto

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 en no funcional según el verbo del cual forma el par. El resultado obtenido es ordenado realizando un ranking en cual cada uno de los nodos es ponderado por las estadísticas generadas en el análisis del texto previamente realizado más la información de crosscutting que es obtenida del recorrido del grafo.

Esta 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, basandose en la estructura del grafo de identificación para efectuar el filtrado, permitiendo una mayor escalabilidad del enfoque (lo cual satisface la Característica V). Entre los criterios de filtrado, podemos encontrar los filtros de verbo, de objetos, etc.

IV.1.1 - Proceso de identificación

En esta sección se explicara en detalle cada una de las tareas llevadas a cabo en el proceso propuesto. El proceso (Ilustración IV-2) consta principalmente de tres partes diferentes: el *análisis del procesador de lenguaje natural* (NLP Analysis), *la generación del grafo* (Graph Generation), y el *buscador de aspectos candidatos* (Aspect Finder).

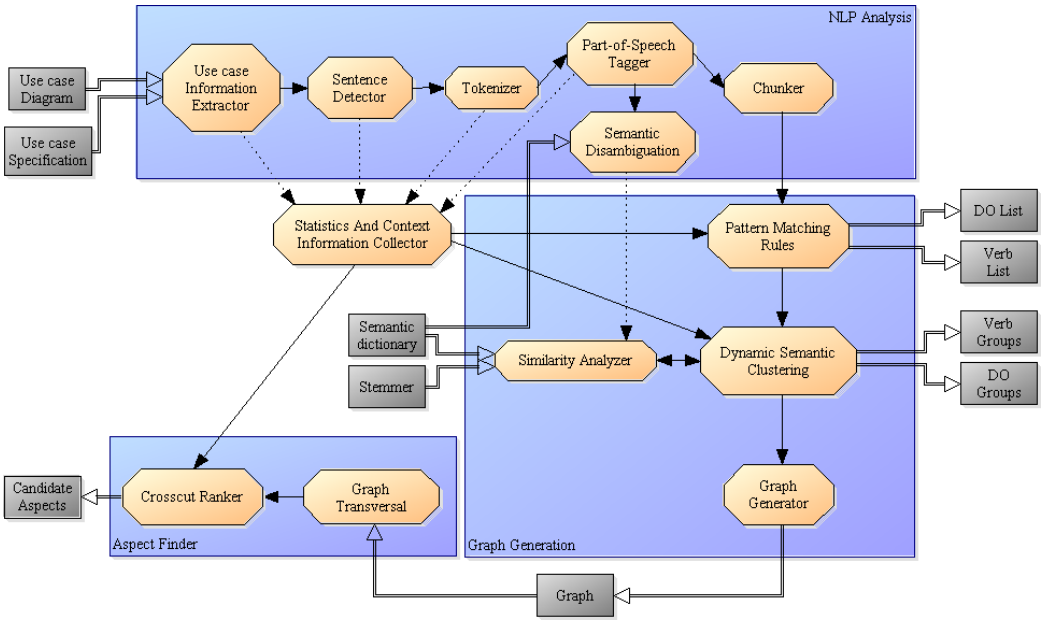

Ilustración IV-2 - Proceso de identificación de aspectos candidatos propuesto

IV.1.2 - Análisis del procesador de lenguaje natural

Este bloque de tareas tiene como finalidad llevar a cabo un análisis léxico, sintáctico, semántica del texto de las especificaciones, así como también llevar a cabo un estudio estadístico y de ubicación de este texto. Consta de seis tareas explicitas 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)
- Desambiguación semántica (Semantic Disambiguation)
- Agrupamiento en grupos sintácticos (Chunker)

Y también de una tarea implícita:

- Recolección de estadísticas e información de contexto (Statistics And Context Information Collector)

IV.1.2.1 Extracción de información de los casos de uso

El objetivo de esta tarea es analizar la información suministrada por el diagrama de casos de uso (inclusiones, extensiones, herencias, etc.) y las especificaciones de casos de uso, de tal manera de

poder extraer el texto y las restricciones relevantes para el análisis del NLP. Tiene la responsabilidad de determinar que texto de los casos de uso es clasificada como funcional o no funcional.

IV.1.2.2 Separación de sentencias

Esta tarea se encarga de realizar la separación de los párrafos entregados por la tarea anterior en oraciones o sentencias. Esta se realiza para los flujos básicos, alternativos, requerimientos especiales, etc. de los casos de uso, como también para los requerimientos suplementarios entregados por la tarea anterior, y es una tarea previa para la ejecución del etiquetado POS.

Esta tarea no es tan trivial, ya que si bien se podría pensar en dividir el párrafo en sentencias cada vez que ocurre un punto, es probable que debido a las ambigüedades del texto natural esto lleve a realizar una equivocación. Esto pasa debido a que los signos de puntuación (".", "!", "?", etc.) también pueden aparecer dentro de una sentencia, formando parte de esta (es el caso de las abreviaciones, los números decimales, etc.).

Por ejemplo, el párrafo:

Mr. Jones went shopping. His grocery bill came to $23.45.

Si se dividiera solamente considerando los signos de puntuación quedaría:

Mr.	Jones went shopping.	His grocery bill came to $23.	45.

Cuando en realidad el resultado necesario era:

Mr. Jones went shopping.	His grocery bill came to $23.45.

Para hacer esto, los signos de puntuación son considerados marcadores de fin de sentencia potenciales (en vez de definitivos). Entonces el texto es escaneado y cada vez que se encuentra uno de estos caracteres, se evalúa una serie de predicados que, analizando el contexto alrededor del carácter, determinan si es un marcador de fin de sentencia o no.

IV.1.2.3 Separación de palabras

Esta tarea esta encargada de dividir una sentencia en un conjunto de palabras. Esta tarea se realiza para sentencias entregadas por la tarea anterior, y es la tarea previa para la ejecución del etiquetado POS. De la misma manera que la tarea anterior, esta tarea no es tan trivial debido al uso de caracteres especiales en el texto natural (por ejemplo las contracciones).

Por ejemplo, si tenemos la sentencia:

I don't drink coffee.

Y lo dividiéramos solo considerando los espacios quedaría:

I	don't	drink	coffee	.

Cuando en realidad la salida correcta era:

I	do	n't	drink	coffee	.

Esto ocurre porque la palabra "don't", para ser pasada al etiquetador POS, debe ser dividida en "do" y "n't", ya que "do" representa al verbo y "n't" es una contracción de "not" y un adverbio que modifica al verbo precedente.

IV.1.2.4 Etiquetado POS

Esta tarea tiene como finalidad realizar la asignación de etiquetas de parte del discurso a cada palabra recibida de la tarea anterior. Esto es necesario para realizar el agrupamiento en frases sintácticas. El *etiquetado de parte del discurso* (Part-of-Speech Tagging, POS Tagging, o POST),

también llamado etiquetado gramatical, es el proceso de marcar las palabras en el texto con su correspondiente parte del discurso, basándose tanto en su definición como en el contexto que la rodea [9]. Un ejemplo puede ser la identificación de sustantivos, verbos, adverbios, etc.

Refiriéndose al algoritmo particular utilizado en este trabajo, este etiquetado se basa en el *modelo de máxima entropía* (Maximum Entropy Model). El etiquetador que se va a utilizar fue entrenado usando texto del diario "Wall Street Journal" y el "Brown Corpus". Los POS Tags son abreviaciones en código, realizadas en conformidad al esquema del corpus lingüístico desarrollado en la Universidad de Pennsylvania denominado "Penn Treebank[7]". En la Tabla IV-1 se puede observar las posibles etiquetas.

Entonces si se tuviera que etiquetar la siguiente frase:

The	suburb	of	Saffron	Park	lay	on	the	sunset	side	of	London	.

El etiquetador devolvería como resultado:

DT	NN	IN	NNP	NNP	VBD	IN	DT	JJ	NN	IN	NNP	.

CC	*Coordinating conjunction*	**RP**	*Particle*
CD	*Cardinal number*	**SYM**	*Symbol*
DT	*Determiner*	**TO**	*To*
EX	*Existential there*	**UH**	*Interjection*
FW	*Foreign word*	**VB**	*Verb, base form*
IN	*Preposition / subordinate conjunction*	**VBD**	*Verb, past tense*
JJ	*Adjective*	**VBG**	*Verb, gerund/present participle*
JJR	*Adjective, comparative*	**VBN**	*Verb, past participle*
JJS	*Adjective, superlative*	**VBP**	*Verb, non-3rd ps. sing. present*
LS	*List item marker*	**VBZ**	*Verb, 3rd ps. sing. present*
MD	*Modal*	**WDT**	*Wh-determiner*
NN	*Noun, singular or mass*	**WP**	*Wh-pronoun*
NNP	*Proper noun, singular*	**WP$**	*Possessive wh-pronoun*
NNPS	*Proper noun, plural*	**WRB**	*Wh-adverb*
NNS	*Noun, plural*	**``**	*Left open double quote*
PDT	*Predeterminer*	**,**	*Comma*
POS	*Possessive ending*	**''**	*Right close double quote*
PRP	*Personal pronoun*	**.**	*Sentence-final punctuation*
PRP$	*Possessive pronoun*	**:**	*Colon, semi-colon*
RB	*Adverb*	**$**	*Dollar sign*
RBR	*Adverb, comparative*	**#**	*Pound sign*
RBS	*Adverb, superlative*	**-LRB-**	*Left parenthesis*
		-RRB-	*Right parenthesis*

Tabla IV-1 - Etiquetas POS

IV.1.2.5 Desambiguación semántica

Esta tarea se ocupa de desambiguar 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. En [35] se presenta un algoritmo, denominado “Maximum Relatedness Disambiguation Algorithm”, que utiliza métricas de relaciones semánticas para realizar la

[7] The Penn Treebank Project - http://www.cis.upenn.edu/~treebank/

desambiguación de los sentidos de la palabra. Este algoritmo desambigua una palabra polisémica eligiendo aquel sentido de la palabra que maximiza la relación con otras palabras en una ventana de contexto dada. Este algoritmo es general en el sentido de que puede llevar a cabo la desambiguación usando cualquier métrica existente (que retorne un puntaje de relación o similitud para un par de sentidos de una palabra). Se denomina a las palabras de una ventana de contexto como $w_1, w_2, \ldots, w_n$, donde w_t, $1 \leq t \leq n$, es la *palabra objetivo* (target word) a la cual se le debe asignar un sentido. Se asume cada palabra w_i tiene m_i posibles sentidos, denotados como $s_{i1}, s_{i2}, \ldots, s_{imi}$. El objetivo es seleccionar uno de los sentidos del conjunto $\{s_{t1}, s_{t2}, \ldots, s_{tmt}\}$ como el sentido más apropiado para la palabra objetivo w_t. El algoritmo realiza la desambiguación usando una métrica de relación semántica genérica que es denotada como ***relatedness*** $= s_{ij} \times s_{kl} \rightarrow R$, donde s_{ij} y s_{kl} representan cualquier par de sentidos en la ventana de contexto y R representa el conjunto de números reales. En otras palabras, la *relatedness* es una función que toma como entrada dos sentidos, y retorna como salida un numero real (el algoritmo supone que este numero de salida es un indicador del grado de relación semántica entre dos sentidos de entrada).

El pseudo-código es descripto en el Código fuente IV-1. Para cada palabra w_j en la ventana de contexto, el algoritmo computa la *relatedness* entre s_{ti} y cada sentido s_{jk}, $k < 1 < m_j$, de cada palabra w_j y selecciona la puntuación más alta de *relatedness*. Entonces suma el puntaje de cada uno de los sentidos de las palabras en la ventana, y entonces este se convierte en el puntaje para el sentido s_{ti} de la palabra objetivo. Aquel sentido con el puntaje más alto es devuelto como el sentido correspondiente a la palabra objetivo.

```
Para cada sentido s_ti de la palabra objetivo w_t {
      set score_i = 0
      Para cada palabra w_j en la ventana de contexto {
            Saltear a la proxima palabra si j == t
            Para cada sentido s_jk de w_j {
                  temp_score_j = relatedness(s_ti, s_jk)
            }
            winning_score = maximo_puntaje(temp_score_j)
            if (winning_score > threshold)
                  set score_i = score_i + winning_score
      }
}
Retornar i, tal que score_i ≥ score_j , para todo j, 1 ≤ j ≤ n, n = número de
palabras en la sentencia
```

Código fuente IV-1 - Pseudo-código de algoritmo de desambiguación

En este trabajo se utiliza una modificación del algoritmo, que se aplica a todos las palabras del texto y con el tamaño de ventana que abarca la oración donde se encontró la palabra objetivo. Se utilizan dos medidas de relatedness, una es la propuesta por Lesk (Lesk Gloss Overlap Measure) que es rápida pero no muy efectiva, y la otra es la desarrollada por Pedersen y otros (Lesk Extended Gloss Overlap Measure) que es más efectiva pero también más costosa en velocidad. Ambas están basadas en el algoritmo original propuesto por Lesk para identificar relaciones semánticas entre palabras utilizando su definición o descripción (Gloss) del diccionario. En el algoritmo que se propone, el inventario de donde se obtienen las estas definiciones se corresponden con la base de datos semántica WordNet.

El algoritmo original selecciona un significado para una palabra objetivo en particular comparando las definiciones del diccionario de sus posibles sentidos con aquellas definiciones de las palabras en la ventana de contexto. Éste está basado en la intuición que los sentidos de las palabras que están relacionadas entre sí frecuentemente son definidos en el diccionario utilizando las mismas palabras. Este algoritmo considera las definiciones como una bolsa de palabras desordenadas y simplemente cuenta el número de palabras que se solapan o repiten entre cada sentido de la palabra objetivo y los sentidos de las otras palabras en la sentencia. El algoritmo selecciona el sentido de la palabra objetivo que tiene más solapaciones con los sentidos de las palabras que la rodean.

Un ejemplo del algoritmo de Lesk original se muestra en la Ilustración IV-3, suponga la sentencia siguiente, donde se desea desambiguar la palabra *bank*. Suponga que la palabra *bank* tiene dos sentidos y la palabra *lake* solamente uno. Aplicando el algoritmo, se ve que entre $bank_1$ y $lake_1$ no hay ninguna solapación, pero entre $bank_2$ y $lake_1$ hay tres solapaciones (*land*, *body* y *water*). Por lo tanto el algoritmo determinaría que $bank_2$ es el sentido más apropiado según el contexto de la sentencia.

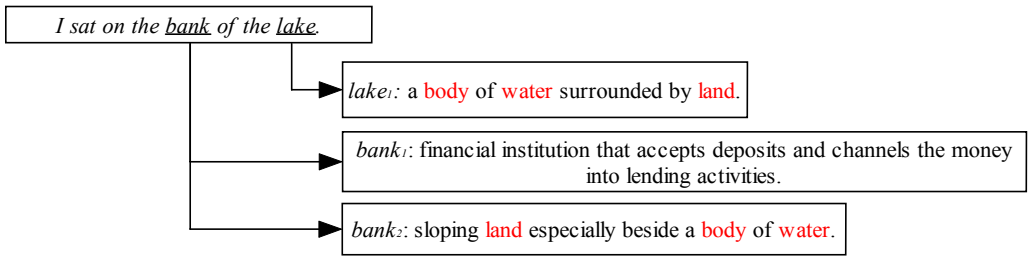

Ilustración IV-3 - Ejemplo del algoritmo de Lesk original

Aunque los solapaciones de las definiciones del diccionario son una forma muy prometedora de medir la similitud porque pueden ser usados para hacer comparaciones entre conceptos de diferentes POS, pueden llegar a ser inefectivas si las definiciones son necesariamente cortas y no proveen la suficiente información para determinar la similitud. Por esta razón, las investigaciones de Banerjee y Pedersen se enfocaron en cómo sobreponerse a las limitaciones de las definiciones cortas. Su propuesta difiere con la original de Lesk en que se buscan las solapaciones no sólo entre las definiciones de los dos conceptos que están siendo medidos, sino que también entre aquellos conceptos que están relacionadas entre estos dos primeros. Además, en vez de contar la cantidad de palabras que son compartidas entre las definiciones (es decir que se cuenta de a una palabra), se cambia la puntuación de tal manera que según la cantidad de palabras consecutivas que se solapan entre dos sentidos diferentes se incrementa la cantidad de puntaje obtenido, siendo esta el cuadrado de la cantidad de palabras consecutivas repetidas.

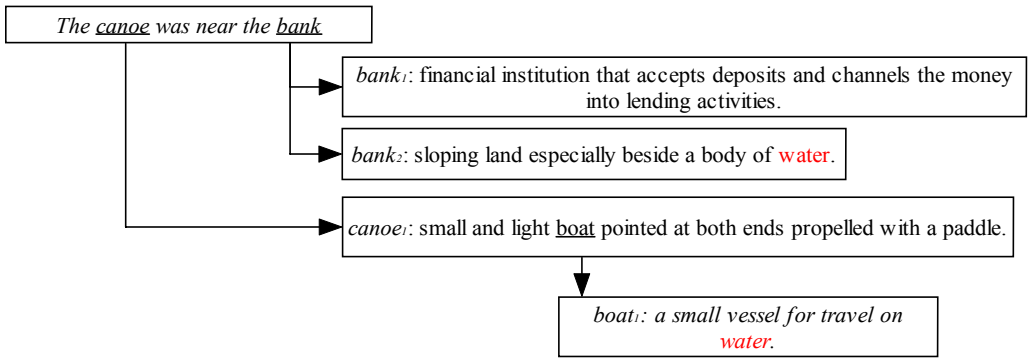

Ilustración IV-4 - Ejemplo del algoritmo de Lesk extendido

Para mostrar un ejemplo del algoritmo de Lesk extendido, suponga la sentencia siguiente, donde se desea desambiguar la palabra *bank* (Ilustración IV-4). En el algoritmo original, no hay solapaciones entre el $canoe_1$ y ninguno de los sentidos de *bank* ($bank_1$ y $bank_2$). Sin embargo si lo hay entre $boat_1$ y $bank_2$, ya que las definiciones de estos dos conceptos comparten la palabra *water*, lo cual indica que están relacionadas. El hecho que los conceptos que están relacionados con los conceptos originales (en este caso $canoe_1$ y $boat_1$) también tengan solapaciones lleva a los autores de la investigación a la conclusión que el algoritmo extendido tiene mayor efectividad, y las relaciones entre conceptos que están implícitas pueden ser encontradas mediante la expansión de las definiciones.

IV.1.2.6 Agrupamiento en grupos sintácticos

Esta tarea se ocupa de agrupar las palabras previamente etiquetadas con POS Tags en conjuntos de palabras (conocidos con el nombre de frases sintácticas). La finalidad de esta es poder realizar posteriormente el análisis sintáctico de las sentencias, de forma tal de poder identificar los pares de verbos y objetos directo. Las palabras de una sentencia se agrupan en partes más grandes, y cada una de estas partes es etiquetada con la unidad sintáctica que le corresponda (por ejemplo, una frase sustantiva o una frase verbal). Los Chunk Tags son abreviaciones en código, que identifican un tipo en particular de unidad sintáctica (Tabla IV-2).

ADJP	**Adjective Phrase**
ADVP	**Adverb Phrase**
CONJP	**Conjunction Phrase**
INTJ	**Interjection**
LST	**List marker**
NP	**Noun Phrase**
PP	**Prepositional Phrase**
PRT	**Particle**
SBAR	**Clause introduced by a subordinating conjunction**
UCP	**Unlike Coordinated Phrase**
VP	**Verb Phrase**

Tabla IV-2 - Etiquetas Chunk

Un ejemplo del chunking es, dada la siguiente sentencia:

This is n't the greatest example sentence in the world because I 've seen better.

Primero se realiza el POS Tagging sobre la sentencia, quedando:

This	is	n't	the	greatest	example	sentence	in	the	world	because	I	've	seen	better	.
DT	VBZ	RB	DT	JJS	NN	NN	IN	DT	NN	IN	PRP	VBP	VBN	RB	.

Y recién entonces se efectua el chunking sobre la sentencia:

This	is	n't	the	greatest	example	sentence	in	the	world	because	I	've	seen	better	.
NP	VP		NP				PP	NP		SBAR	NP	VP		ADVP	.

IV.1.3 - Recolección de estadísticas e información de contexto

Esta tarea se encarga de mantener información estadística acerca de las tareas realizadas, como la cantidad de ocurrencias de las palabras. Además mantiene las ubicaciones de cada ocurrencia, con referencias a su contexto inmediato. Es de utilidad, ya que varias tareas necesitan esta información para llevar a cabo sus actividades.

IV.1.4 - Generación del grafo

Este bloque de tareas 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 esta formado por cuatro tareas que ejecutan esta elaboración:

- Detección de verbos y objetos directos (Pattern Matching Rules)
- Análisis de similaridad (Similarity Analyzer)
- Agrupamiento dinámico de verbos y sustantivos (Dynamic Semantic Clustering)
- Creación del grafo (Graph Generator)

IV.1.4.1 Detección de verbos y objetos directos

Esta tarea se ocupa de llevar a cabo la identificación de los verbos y los objetos directos en las sentencias, apoyándose en la salida del agrupador sintáctico, aplicando una serie de reglas. Estas reglas identifican 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 detecto en la frase verbal, y en base a esto se identifica el patrón sintáctico que le corresponde. Esto permite saber si la frase contiene o no un objeto directo, y si existe, en que lugar se encuentra. En el Código fuente IV-2 se observa detalladamente la forma de identificar los verbos y objetos directos, considerando todos los casos posibles.

```
Para todos los agrupamientos
      Encontrar una frase verbal
      Determinar donde empieza la próxima frase verbal
      Determinar el verbo más representativo
      Si encontré un verbo y no es un verbo be, un verbo nexo un
      verbo intransitivo
            Obtener todos los agrupamientos desde la frase verbal
            hasta la próxima frase verbal que sean frases sustantivas
            o frases adjetivas
            Si no hay ninguna frase sustantiva o adjetiva después del
            predicado
                  No hay que hacer nada
            Sino si hay solo una frase sustantiva
                  Obtengo los sustantivos de la frase sustantiva
                  Obtengo el sustantivo dominante que juega el rol de
                  objeto directo
            Sino si hay dos frases
                  Si hay una frase sustantiva seguida de una adjetiva
                        Obtengo los sustantivos de la frase sustantiva
                        Obtengo el sustantivo dominante que juega el
                        rol de objeto directo
                  Si hay dos frases sustantivas
                        Obtengo los sustantivos de la primera frase
                        sustantiva
```

```
                Obtengo el sustantivo dominante que juega el
                rol de objeto directo
        Sino si hay más de dos frases
            Obtengo los sustantivos de la primera frase
            sustantiva
            Obtengo el sustantivo dominante que juega el rol de
            objeto directo
        Si encontré un objeto directo
            Creo el verbo
            Creo el objeto directo
            Los relaciono entre si
```

Código fuente IV-2 - Pseudo-código de identificación de verbos y objetos directos

El análisis anterior sirve sólo para las frases escritas en voz activa. Aquellas que se encuentren en voz pasiva no son consideradas (de momento) ya que no es usual que estas se utilicen en una especificación de requerimientos.

IV.1.4.2 Análisis de similaridad

Esta tarea esta encargada realizar el análisis de similaridad entre verbos (y objetos directos) para posteriormente agruparlos en grupos semánticamente relacionados. Esta basado en el resultado de la desambiguación 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 similaridad entre estos 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 desambiguado 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:

1. Si se comparan dos verbos, se usará la jerarquía de tropónimos e hiperónimos provista por WordNet para ampliar las palabras que representan los verbos.
2. Si se comparan dos sustantivos (objetos directos), se utilizará la jerarquía de hiperónimos e hipónimos provista por WordNet para ampliar las palabras que representan al sustantivo.

Esto quiere decir, que para cada término que se compara, se realiza una explosión (véase la Ilustración IV-5) en las direcciones previamente dichas (tropónimos, hiperónimos e hipónimos) para poder tener un entendimiento del concepto que representa el termino, y mas que nada de las palabras que lo representan para realizar la comparación. Para cada uno de los dos términos que se piensan comparar, se realiza la explosión (esta controlada a no ir más de tres niveles en cualquier dirección) y se almacenan las palabras que representan a cada synset alcanzado por la explosión.

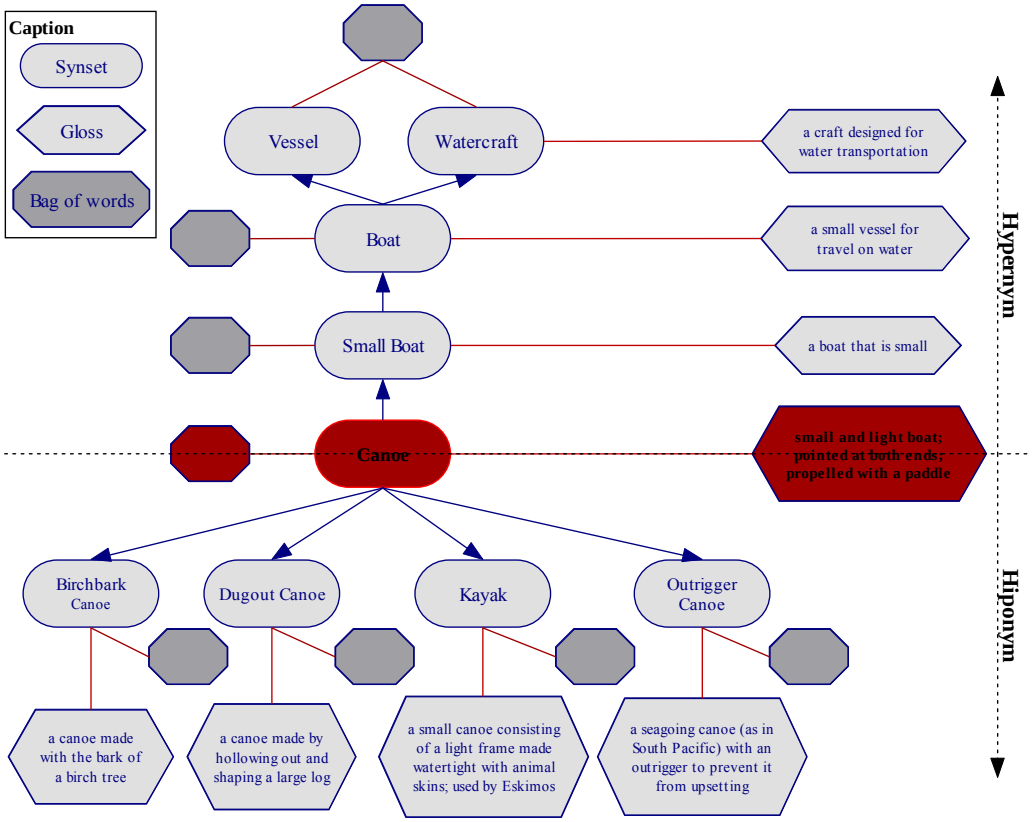

Ilustración IV-5 - Aumentación de información sobre un sustantivo

Cabe destacar que a las palabras se les aplica un proceso de stemming para aumentar la efectividad de las comparaciones. Esta medida es utilizada posteriormente en la próxima tarea para llevar a cabo el agrupamiento.

Comentario [OSR5]: Ampliar , agregar pseudocódigo, y ejemplo en forma de ilustración.

IV.1.4.3 Agrupamiento dinámico de verbos y sustantivos

Esta tarea 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 similaridad entre la palabra y las palabras de cada grupo. Para aquellos grupos donde la inclusión de esa palabra mantenga (en promedio) el puntaje de similaridad en un número que supere un umbral preestablecido, se la almacena como candidata de 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 esta relacionada con ningún agrupamiento) y se agregue a un grupo de un solo elemento. Es importante remarcar que se utiliza las estadísticas recolectadas por el proceso para considerar solo aquellos verbos y objetos directos que son relevantes.

IV.1.4.4 Creación del grafo

En esta tarea se lleva a cabo la creación del grafo propuesto anteriormente (Ilustración IV-1). Para realizarlo se cuenta con la información generada por las tareas anteriores, principalmente con las listas y agrupamientos de verbos y objetos directos. Estos son agregados al grafo e interconectados, de manera tal de que este permita ser recorrido. Además, se agregan al grafo los nodos “grupo de verbo”-

“grupo de objetos directo” 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.

IV.1.5 - Buscador de aspectos candidatos

Este bloque de tareas tiene como propósito, apoyándose en los artefactos generados de las tareas previas (principalmente en el grafo), la identificación de aspectos candidatos y el ordenamiento de estos según su relevancia. El bloque esta formado por dos tareas que ejecutan esta elaboración:

- Recorrido del grafo en búsqueda de concerns crosscutting (Graph Transversal)
- Ordenamiento de los aspectos candidatos (Crosscut Ranker)

Además de la tarea adicional:

- Filtrado de aspectos candidatos (Candidate Aspects Filtering)

IV.1.5.1 Recorrido del grafo en búsqueda de concerns crosscutting

Esta tarea es la que tiene la obligación 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. En el Código fuente IV-3 se puede apreciar el algoritmo de navegación del grafo.

```
crosscut_count_verbgroups[] = 0
Para cada "grupo de verbos" vg {
      crosscut_count_verbgroup = 0
      crosscut_count_usecase[] = 0
      Para cada par "grupos de verbos-grupo de objetos directo" vg-dog {
            Para cada nodo de uso u {
                  crosscut_count_verbgroup += 1
                  crosscut_count_usecase[u] += 1
            }
      }
      Si crosscut_count_verbgroup > threshold
            crosscut_count_verbgroups[vg] = crosscut_count_verbgroup
      Si no
            crosscut_count_verbgroups[vg] = 0
}
Retornar los grupos de verbos que son crosscutting.
```

Código fuente IV-3 - Pseudo-código del recorrido del grafo

Para cada grupo de verbo que es considerado crosscutting, se realiza una exploración en busca del termino (un verbo) mas 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. Si el verbo provino de un caso de uso, si es funcional se pondera con un 1 y si es no funcional con -1. Se analizan todos los verbos que componen el grupo, y si la suma es positiva entonces es un concern funcional, y si es negativo se considera no funcional. Asimismo, si algúno de los verbos provino de un requerimiento suplementario, entonces se considera que el concern crosscutting es no funcional, sin tener en cuenta la cuenta la ponderación de los casos de uso.

IV.1.5.2 Ordenamiento de los aspectos candidatos

Esta es la ultima tarea del proceso, y 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:

1. El número de casos de uso cortados transversalmente.
2. El número de cortes transversales totales (puede haber más de uno en un caso de uso especifico) del grupo de verbos.
3. La relevancia de ocurrencia del grupo de verbos.
4. La relevancia de ocurrencia de los grupos de objetos directos relacionados con el grupo de verbos en cuestión.
5. Si el grupo de verbos es considerado funcional o no funcional.

Estos parámetros son reunidos en una sola función que da como resultado un número real representando la importancia del aspecto candidato (Ilustración IV-6).

El puntaje es:

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

Sujeto a:

$\alpha = 10$

$\beta = 6$

$\gamma = 3$

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

$\Delta = 2$, 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 IV-6 - Fórmula de ordenamiento de aspectos candidatos

Los parámetros α, β y γ toman los valores α = 10, β = 6 y γ = 3, de forma arbitraria. De esta manera se ordenan todos los aspectos candidatos, de forma tal que representen su relevancia en el sistema.

IV.1.5.3 Filtrado de aspectos candidatos (Candidate Aspects Filtering)

Esta tarea es necesaria para poder manejar la selección, especificación y administración del los aspectos candidatos por parte desarrollador, luego de que la herramienta los haya encontrado. Es imprescindible contar con alguna forma de poder reducir la información generada por el enfoque, basándose en determinados tipos de criterios, en un subconjunto manejable. Esto permite que se

aplique en grandes desarrollos, tornando la herramienta en un enfoque escalable, es decir, que sin importar la cantidad de aspectos candidatos haya, por más grande que sea el sistema a desarrollar, el analista podrá especificarlos y manejarlos de forma eficiente y práctica.

Para hacer esto, se propone utilizar la estructura del grafo generado, para poder reducir la información basándose en estos criterios:

- Verbo: Se especifica el nombre de un verbo, y cada aspecto candidato que contenga este verbo será retornado.
- Objeto directo: 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: Se especifica un número mínimo de cuenta crosscutting, y cada aspecto candidato cuya cuenta crosscutting iguale o supere este número será retornado.
- Caso de uso: Se especifica un caso de uso, y todos aquellos aspectos candidatos que contengan algún uso en este caso de uso será retornado.

IV.2 - Extensión de Aspect Tool Extractor

Resumen

IV.2.1 - Diseño de la extensión

Resumen

Grafico del diagrama general

Paquete LogicaExtendida

Paquete Adicional

Paquete AnalisisNLP

Paquete BuscadorAspectos

Paquete Filtros

Paquete GeneracionGrafo\

Paquete Grafo

IV.2.2 - Implementación de la extensión

Resumen

Capitulo V - Evaluación de la técnica propuesta

Resumen

V.1 - Problemática

V.2 - Análisis del caso de estudio con Aspect Extractor Tool

V.2.1 - Resultados sin la mejora de la técnica de identificación de aspectos

V.2.2 - Resultados con la mejora de la técnica de identificación de aspectos

V.3 - Comparación de ambas aplicaciones y conclusiones

Capitulo VI - Conclusiones

Actual estado de las técnicas de identificación de aspectos

Críticas a las técnicas

Análisis de mis mejoras

Explicar donde fallan las técnicas de WSD, y como se podrían mejorar.

Explicar donde falla WordNet. En el uso de dominios particulares.

Posibles mejoras

La mejora de la composición de aspectos.

Mejora de agregar algo que arregle los errores ortográficos.

Futuro de las técnicas de identificación de aspectos y de los aspectos tempranos en los procesos de desarrollo de software

Abreviaturas

- **AM** - *Aspect Mining*
- **AOIG** - *Action-Oriented Identifier Graph*
- **AOP** - *Aspect-Oriented Programming*
- **AORE** - *Aspect-Oriented Requirements Engineering*
- **AOSD** - *Aspect Oriented-Software Development*
- **ARCADE** - *Aspectual Requirements Composition And Decision Soport*
- **DO** - *Direct Object*
- **EA** - *Early Aspects*
- **EA-Miner** - *Early Aspect Mining*
- **Early-AIM** - *Early Aspects Identification Method*
- **FR** - *Functional Requirement*
- **IR** - *Information Retrieval*
- **KWIP** - *Key Word In Phrase*
- **NFR** - *Non-Functional Requirement*
- **NLP** - *Natural Language Processor*
- **POS** - *Part-of-Speech*
- **RAT** - *Requirement Analysis Tool*
- **RE** - *Requirements Engineering*
- **RL** - *Requirement Leave*
- **SI** - *Sense Inventory*
- **SoC** - *Separation of Concerns*
- **Synset** - *Synonym Set*
- **UML** - *Unified Modeling Language*
- **WSD** - *Word Sense Disambiguation*

Bibliografía

1. *Retórica y composición hispánicas*. 1999 [cited; Available from: http://www.acs.ucalgary.ca/~gimenez/499_06_Fichas.htm.
2. *Aspect-Oriented Software Development Net*. 2008 [cited; Available from: http://aosd.net/.
3. *Aspect Oriented Software Development*. 2008 [cited; Available from: http://en.wikipedia.org/wiki/Aspect-oriented_software_development.
4. *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*. 2008 [cited; Available from: http://www.early-aspects.net/.
5. *European Conference on Object-Oriented Programming (ECOOP)*. 2008 [cited; Available from: http://www.ecoop.org/.
6. *International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. 2008 [cited; Available from: http://www.oopsla.org/.
7. *International Conference on Software Engineering (ICSE)*. 2008 [cited; Available from: http://www.icse-conferences.org/.
8. *Lenguaje Unificado de Modelado*. 2008 [cited; Available from: http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado.
9. *Part-of-speech tagging*. 2008 [cited; Available from: http://en.wikipedia.org/wiki/Part-of-speech_tagging.
10. *Programación Orientada a Aspectos*. 2008 [cited; Available from: http://es.wikipedia.org/wiki/Programaci%C3%B3n_Orientada_a_Aspectos.
11. *Wmatrix Corpus Analysis and Comparison Tool*. 2008 [cited; Available from: http://ucrel.lancs.ac.uk/wmatrix/.
12. *WordNet*. 2008 [cited; Available from: http://en.wikipedia.org/wiki/WordNet.
13. *The WordNet Home Page*. 2008 [cited; Available from: http://wordnet.princeton.edu/.
14. Araújo, J., et al. *Aspect-Oriented Requirements with UML*. in *Second International Workshop on Aspect-Oriented Modelling with UML (held in conjunction with the Fifth International Conference on the Unified Modeling Language)*. 2002. Dresden, Germany.
15. 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.
16. Baniassad, E. and S. Clarke, *Investigating the Use of Clues for Scaling Document-Level Concern Graphs*, in *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with OOPSLA 2004 Conference*. 2004: Vancouver, Canada.
17. 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.
18. Barbosa, F.S., *Comparing Three Aspect Mining Techniques*, in *Simpósio Doutoral em Engenharia Informática (DSIE'08)*. 2008: Porto, Portugal.
19. Basile, P., et al. *The JIGSAW Algorithm for Word Sense Disambiguation and Semantic Indexing of Documents*. in *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*. 2007. Prague, Czech Republic.
20. Bass, L., M. Klein, and L. Northrop. *Identifying Aspects using Architectural Reasoning*. in *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with AOSD Conference*. 2004. Lancaster, UK.
21. Cojocar, G.S. and G. Şerban. *On Some Criteria for Comparing Aspect Mining Techniques*. in *Proceedings of the 3rd Workshop on Linking Aspect Technology and Evolution*. 2007. Vancouver, British Columbia, Canada: ACM.
22. Chitchyan, R., et al., *A Survey of Analysis and Design Approaches*. 2005, AOSD-Europe.

23. Chitchyan, R., et al. *Evaluating EA-Miner: Are Early Aspect Mining Techniques Effective?* in *Proceedings of the First International Workshop Towards Evaluation of Aspect Mining*. 2006. Nantes, France.
24. 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.
25. Chung, L., et al., *Non-Functional Requirements in Software Engineering*. International Series in Software Engineering. Vol. Vol. 5. 2000: Kluwer Academic Publishers.
26. Figuerola, C., *La Investigación sobre Recuperación de la Información en Español*. 2000, Facultad de Documentación, Universidad de Salamanca.
27. Haak, B., et al., *Aspects Extractor: Identificación de Aspectos en la Ingeniería de Requerimientos*. 2005, Facultad de Ciencias Exactas, UNICEN.
28. Haak, B., et al., *Identificación Temprana de Aspectos.* Revista Sociedad Chilena de Ciencia de la Computación, 2005. **Volumen 6**(Workshop de Ingeniería de Software).
29. 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.
30. Kellens, A., K. Mens, and P. Tonella, *A Survey of Automated Code-Level Aspect Mining Techniques.* Transactions on Aspect-Oriented Software Development (TAOSD), 2007. **Volume IV**(Special issue on Software Evolution): p. 145 - 164.
31. Lohmann, D. and J. Ebert, *A Generalization of the Hyperspace Approach using Meta-Models*, in *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with AOSD Conference 2003*. 2003: Boston, Massachusetts, USA.
32. Loughran, N. and A. Rashid, *Mining Aspects*, in *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (held in conjunction with AOSD Conference 2002)*. 2002: University of Twente, Enschede, The Netherlands.
33. Marin, M., L. Moonen, and A.v. Deursen. *A Common Framework for Aspect Mining Based on Crosscutting Concern Sorts*. in *13th Working Conference on Reverse Engineering*. 2006.
34. Moldovan, G.S. and G. Şerban. *Quality Measures for Evaluating the Results of Clustering Based Aspect Mining Techniques*. in *Proceedings of the First International Workshop Towards Evaluation of Aspect Mining*. 2006. Nantes, France.
35. Pedersen, T., S. Banerjee, and S. Patwardhan, *Maximizing Semantic Relatedness to Perform Word Sense Disambiguation*. 2005, University of Minnesota Supercomputing Institute.
36. Pryor, J. and C. Marcos. *Solving Conflicts in Aspect-Oriented Applications*. in *Proceedings of the Fourth Argentine Symposium on Software Engineering (ASSE´2003), 32 JAIIO (Jornadas Argentinas de Informática e Investigación Operativa)*. 2003. Buenos Aires, Argentina.
37. 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.
38. 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.
39. Rayson, P., *Matrix: a Statistical Method and Software Tool for Linguistic Analysis Through Corpus Comparison*. 2002, Lancaster University.
40. Rayson, P., et al. *The REVERE Project: Experiments with the application of probabilistic NLP to Systems Engineering*. in *Proceedings of 5th International Conference on Applications of Natural Language to Information Systems (NLDB'2000)*. 2000. Versailles, France.
41. Rayson, P., R. Garside, and P. Sawyer, *Language Engineering for the Recovery of Requirements from Legacy Documents (REVERE Project Report)*. 1999, Lancaster University.
42. 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.

43. Rayson, P., R. Garside, and P. Sawyer. *Assisting Requirements Engineering with Semantic Document Analysis*. in *Proceedings of RIAO 2000 (Recherche d'Informations Assistie par Ordinateur, Computer-Assisted Information Retrieval) International Conference*. 2000. Collège de France, Paris, France.
44. 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.
45. 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.
46. Sampaio, A., et al., *Mining Aspects in Requirements*, in *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with AOSD 2005 Conference*. 2005: Chicago, Illinois, USA.
47. Sampaio, A. and A. Rashid, *Report on Evaluation of Aspect Identification Tool (EA-Miner) in Case Studies*. 2007, AOSD-Europe.
48. Sampaio, A., A. Rashid, and P. Rayson. *Early-AIM: An Approach for Identifying. Aspects in Requirements*. in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*. 2005. Paris, France: IEEE Computer Society.
49. Seiter, L. *Automatic Mining Of Context Passing In Java Programs*. in *Proceedings of the First International Workshop Towards Evaluation of Aspect Mining*. 2006. Nantes, France.
50. Shepherd, D., et al., *Using Natural Language Program Analysis to Locate and Understand Action-Oriented Concerns*, in *International Conference on Aspect Oriented Software Development (AOSD 2007)*. 2007: Vancouver, British Columbia.
51. 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.
52. 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.
53. Sousa, G., et al. *Separation of Crosscutting Concerns from Requirements to Design: Adapting an Use Case Driven Approach*. in *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with AOSD Conference*. 2004. Lancaster, UK.
54. Störzer, M., U. Eibauer, and S. Schoeffmann. *Aspect Mining for Aspect Refactoring: An Experience Report*. in *Proceedings of the First International Workshop Towards Evaluation of Aspect Mining*. 2006. Nantes, France.
55. TAN, W., *Introduction to Aspect Mining*. 2006, Department of Computer Science, Sun Yat-sen University.
56. Wilson, M. *Sentences*. 1998 [cited; Available from: http://www.uncp.edu/home/canada/work/caneng/sentence.htm.