Extracción y clasificación de agentes KAOS desde requisitos textuales usando procesamiento de lenguaje natural

Carlos Andrés Vélez-Carvajal, Luis Alfonso Lezcano Rodríguez, Jaime Alberto Guzmán Luna

caavelezca@unal.edu.co, lalezcan@unal.edu.co, jaguzman@unal.edu.co

Universidad Nacional de Colombia
Carrera 80, No. 65 - 223
Medellín - Colombia

Resumen: *Este artículo presenta un método para la identificación y clasificación de agentes, según su definición en el enfoque de objetivos KAOS, partiendo de especificaciones de requisitos en español. Se basa en trabajos previos en procesamiento de lenguaje natural para la obtención del diagrama de clases. El método comienza con el tokenizado del texto, etiquetando cada palabra en su contexto y eliminando palabras vacías. Luego, se deben tomar los sujetos de cada oración y se agregan a la lista de conceptos encontrados. Finalmente, se aplican tres reglas para refinar la lista de conceptos: basado en las ocurrencias y frecuencia de palabras, se eliminan conceptos infrecuentes, al igual que los conceptos inactivos; de acuerdo a los hiperónimos, si el concepto se relaciona con humanos, se clasifica como agente de ambiente, de otro modo, si se relaciona con entidades o artefactos, es un agente de software. El método muestra resultados satisfactorios, aunque algunas simplificaciones deben eliminarse como trabajo futuro.*

Palabras clave: Procesamiento de lenguaje natural, NLP, KAOS, Ontologías de palabras, ingeniería de requisitos.

Abstract: *This article presents a method for identifying and classifying agents, as defined in KAOS framework, from specifications in Spanish natural language. It is based on previous works in natural language processing for obtaining the class diagram from requirements specifications. The method starts by tokenizing the text, tagging each word in its context and eliminating stop words. Then, it should be taken up the subjects of each sentence and they should be added in a list of concepts. Finally, three rules are applied for refining the list of agents: based on the occurrences and frequency of words, infrequent concepts are eliminated, and inactive concepts as well; according to the hypernyms, if the concept is related to humans is classified as environment agent, else, if is related to entities or artifacts, is a software agent. The method shows acceptable results, however, some simplifications should be solved as further work.*

Keywords: Natural Language Processing, NLP, KAOS, Ontologies words, requirements engineering.

1 Introducción

Una de las razones principales por las que se logra el éxito de un programa de software es por el grado en el que alcanza su propósito. Precisamente, KAOS, metodología con un enfoque de ingeniería de requisitos orientada a objetivos, surge de la necesidad de modelar y razonar sobre el sistema compuesto –software y ambiente–, capturando la base lógica de los elementos que lo componen, y teniendo como punto de partida los intereses de más alto nivel para la organización [1]. Dada la importancia de este enfoque, un trabajo significativo consiste en obtenerlo automáticamente de descripciones que hace el interesado en lenguaje natural.

En este artículo, se presenta un método para la obtención de agentes de acuerdo con la definición que se hace de ellos en el diagrama de objetivos de KAOS, partiendo de una especificación de éstos en lenguaje natural para el idioma español.

El método adapta algunas estrategias que se proponen para obtener el diagrama de clases con base en el lenguaje natural. Asimismo, incluye las siguientes tareas: tokenizado del texto, etiquetado de cada palabra, identificación de lexemas, y la identificación y eliminación de palabras vacías. Luego de hacer cálculos del total de palabras, ocurrencias y frecuencias, se utiliza Multilingual Central Repository (MCR) para identificar qué conceptos son relevantes. Después se proponen 3 reglas que sirven para descartar conceptos, y dejar los que probablemente serán agentes, además, permiten clasificar agentes de software y de ambiente.

Este artículo está organizado de la siguiente manera. En la sección 2, se sitúa a los objetivos como un elemento esencial en la ingeniería de requisitos, se introducen algunas nociones del enfoque de desarrollo KAOS y, además, se mencionan las problemáticas propias del lenguaje natural; la sección 3 explica la ventaja de ubicar una metodología, como la estudiada bajo un metamodelo y se menciona un metamodelo del diagrama de KAOS que aparece en la literatura. Luego, en la sección 4, se presenta el método propuesto, el cual arroja una lista de conceptos candidatos, que en la sección 5, se refinan. Precisamente, en la sección 5, se describen las reglas que se deben seguir con el fin de obtener agentes de software y de ambiente. En la sección 6, se muestran los resultados obtenidos bajo tres casos de estudio. Finalmente, se presentan las conclusiones y el trabajo futuro.

2 Procesamiento de lenguaje natural en la ingeniería de requisitos

En sí, un objetivo se puede definir como una meta que el sistema en consideración debe alcanzar. De manera que la formulación de un objetivo se refiere a propiedades que deben ser aseguradas [2].

Los objetivos juegan un papel importante en el proceso de ingeniería de requisitos. Particularmente, son útiles para apoyar la exploración de diseños alternativos, y para

definir el comportamiento ideal y las responsabilidades de los agentes del sistema y del ambiente [3].

La ingeniería de software orientada a objetivos (GORE) se ha popularizado principalmente debido a que los enfoques tradicionales de análisis de software no son adecuados para afrontar sistemas de software complejos. Además, éstos modelan los datos y procesos, pero no capturan la justificación del sistema de software haciendo difícil entender los requisitos con respecto a problemas de alto nivel en el dominio. Por otra parte, se enfocan en estudiar el sistema aislado, y no razonan sobre el sistema y su relación con el entorno. La GORE trata de resolver estos problemas [1].

Dentro de los enfoques de GORE se encuentra KAOS, que es una metodología con un conjunto de técnicas de análisis formal. Consiste en un lenguaje de especificación de requisitos compuesto de un lenguaje multiparadigma, pues consta de submodelos relacionados a través de unas reglas de consistencia. Éstos son los modelos de objetivos, objetos, responsabilidad de agentes, interfaz de agentes y de operación. En la figura 1, tomada de [4] se presenta un diagrama con todos los elementos gráficos propios de KAOS. Se observa que un objetivo puede entrar en conflicto con otros objetivos, además, puede tener obstáculos asociados, se puede refinar en otros objetivos, y a ese refinamiento puede corresponder una propiedad de dominio que establece las declaraciones descriptivas de la subrogación. Adicionalmente, puede ser refinado por requisitos y expectativas, que son objetivos hoja, es decir, no se subrogan más, y son asignados a un único agente: de software, para el caso de requisitos; de ambiente, para las expectativas. A su vez, los requisitos se operacionalizan en operaciones, las cuales pueden ser causadas por algún evento, y tienen entrada y salida de información, que viene representada por las entidades como objetos autónomos y pasivos en el sistema.

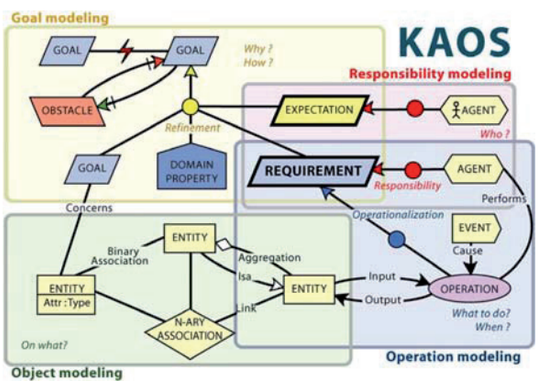

Figura 1: Diagrama KAOS mostrando todos sus componentes de representación gráfica. Fuente: [4].

Asimismo, el enfoque de KAOS tiene una estructura de dos capas: una capa de modelado conceptual para declarar conceptos (como objetivos, objetos y agentes) y relaciones entre los conceptos (como refinamiento de objetivos, asignación de responsabilidad de objetivos a agentes, etc.), tal y como se aprecia en la figura 1, y una capa de declaraciones formales para la especificación de conceptos y relaciones [3].

La idea central de KAOS consiste en representar cada objetivo del sistema y refinarlo a través de subobjetivos que describen cómo se alcanzará el objetivo subrogado. A su vez, cada objetivo (excepto el raíz, que es el objetivo estratégico de más alto nivel) se ve justificado por al menos otro objetivo que explica por qué se introdujo en el modelo [4], [5].

Los objetivos de más bajo nivel se deben asociar a agentes, los cuales son de: (i) software, si representan componentes automáticos responsables de alcanzar un objetivo que se denominará requisito, y (ii) ambiente, en el caso de agentes que hagan parte del entorno (por ejemplo, humanos) con la función de lograr un objetivo que se denominará expectativa [3], [4], [5].

La forma más común de expresar requisitos es a través de grandes volúmenes de texto, denominados aserciones en lenguaje natural (NL) que provienen de extractos de entrevistas, documentos y notas [6]. Sin embargo, el NL tiene básicamente las siguientes limitaciones: (i) es ambiguo, haciendo que el análisis preciso y riguroso sea complejo; (ii) el mismo sentido se puede representar de diferentes formas; (iii) pueden existir conceptos ocultos, ya que hay conceptos que no son expresados explícitamente en una fuente escrita [7]. Para atacar estos problemas, aunque no solucionarlos totalmente, se utilizan técnicas de procesamiento del lenguaje natural (NLP) con el fin de entender y capturar conceptos relevantes del texto. Además, se puede utilizar un mecanismo de representación del conocimiento del dominio, como lo son las ontologías, marcos, paradigma orientado a objetos, entre otros, y así, mejorar el desempeño de identificación de requisitos de software [8].

El estado del arte registra avances para obtener diferentes diagramas propios de la ingeniería de software, sin embargo, es poca la documentación que se encuentra en la literatura con respecto a la obtención de los elementos básicos del diagrama de objetivos KAOS a partir de lenguaje natural. Se destacan las propuesta de Lezcano et al. [9] y Zapata et al. [10]. Si bien ambas propuestas permiten obtener objetivos, su subrogación y agentes, se reducen a tener textos con oraciones bajo unos patrones morfosintácticos previamente definidos. El método que aquí se presenta no arroja los objetivos, ni su subrogación, pero sí los agentes. Además, el lenguaje del documento para este método es mucho menos restringido, pues no se reduce a algunas estructuras morfosintácticas, sino que se orienta a la voz activa, la cual es muy común en el lenguaje español.

3 Metamodelo de KAOS

Un metamodelo es una definición precisa de los constructos y reglas necesarias para la creación de modelos semánticos [11]. Los metamodelos pueden servir para múltiples propósitos como el establecer un lenguaje que soporte una metodología particular o proceso [12]. Por tanto, es necesario acudir a una representación ontológica del modelo estudiado. En [13] se presenta un metamodelo del diagrama de objetivos KAOS bajo el enfoque UEML, con el fin de realizar un análisis ontológico usando separación de referencia. Además, esta representación facilita la comparación, verificación de

consistencia, el proceso de actualización, sincronización y la traducción entre modelos.

El método propuesto en el presente artículo permite obtener agentes y su correspondiente clasificación en agentes de software y de ambiente. Actualmente, se viene trabajando en la obtención de objetivos, su clasificación y subrogación. En la figura 2, se muestra un fragmento del metamodelo desarrollado en [13] donde se observan los atributos de los agentes y sus relaciones más relevantes dentro del metamodelo en cuestión.

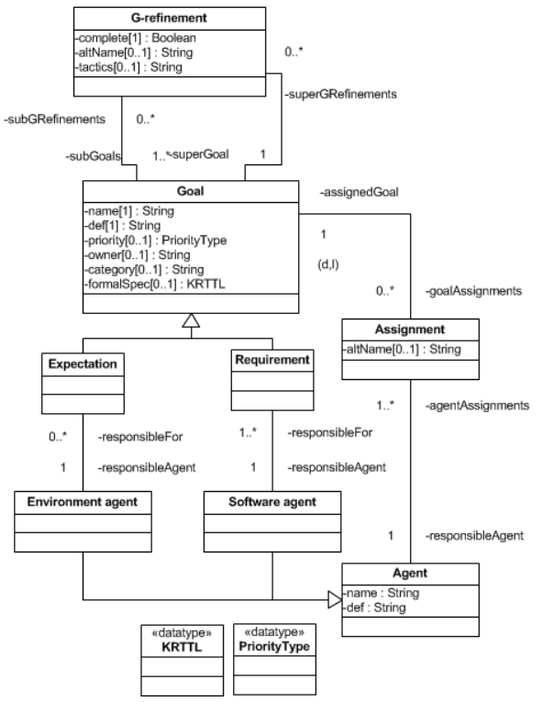

Figura 2: Fragmento del metamodelo de KAOS desarrollado en [13].

4 Método propuesto para la identificación de agentes desde requisitos en lenguaje natural

Con base en los trabajos de Ibrahim & Ahmad [6], Zhou & Zhou [14], Joshi & Deshpande [8] y More & Phalnikar [15], se propone el siguiente método para identificar conceptos que puedan asociarse a agentes. La figura 3 presenta la estructura básica del método mediante un diagrama de procesos en el cual se muestra cada una de las actividades y con flechas discontinuas, la información que entra o sale de almacenes de información. Lo primero es obtener el documento de descripción de los agentes, que es ingresado por el usuario. Este documento constituye el texto que se va a procesar, y con el cual se trabaja durante todo el método. Luego, un parser léxico-sintáctico se encarga del etiquetado morfosintáctico del texto (4.1). Después, se lematiza cada palabra del texto (4.2), se identifican palabras vacías (4.3). Para la siguiente acción, se calcula el número de palabras, ocurrencias y frecuencias, se almacenan las frases nominales en una lista de agentes candidatos, se eliminan las palabras vacías de dicha lista (4.4) y, finalmente, se buscan y almacenan los hiperónimos de cada concepto de la lista. Seguidamente, se aplican algunas reglas heurísticas que, haciendo uso de los datos de los pasos anteriores, eliminan agentes que no sean frecuentes, y clasifican si el agente es un humano o un dispositivo o software. Para ello, se consulta la lista de hiperónimos y se arroja la lista de agentes de software y de ambiente.

Concretamente, como entrada del proceso, se toma un documento en el lenguaje español, sin errores ortográficos ni gramaticales, que contiene las especificaciones de los requisitos funcionales del sistema en el cual se describen los agentes que hacen parte del sistema (ambiente o software a desarrollar). Pueden estar asociados a objetivos, o directamente a las operaciones de cada uno. La estructura morfosintáctica del texto que se procesa espera tener oraciones en voz activa, pues son las que indican claramente los sujetos de la acción. Así, la forma sería: Sustantivo – Verbo – Complemento.

Por ejemplo, "El personal de ambulancia envía información al operador de radio", "El operador de radio recibe información de la ambulancia seleccionada", donde claramente el agente es el conjunto de palabras que hay antes del verbo.

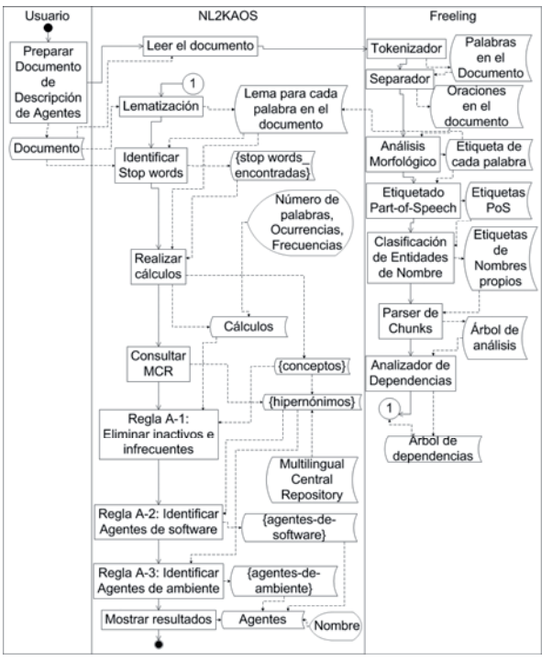

Figura 3: Diagrama de procesos del método propuesto (*NL2KAOS* representa el software que automatiza al método).

Sin embargo, no todas las frases que cumplen con esta forma se refieren a agentes, por ejemplo: "El tiempo necesario para atender la llamada y llenar el formulario del incidente no deben tomar más de cierto tiempo". Este tipo de oraciones que no se refieren a agentes, son las que se eliminan a través de la metodología.

Adicionalmente, todas las frases deben contener explícitamente el sujeto de la oración. Ésta es una

simplificación que se hace con el fin de evitar trabajar la resolución de anáfora, que es un problema adicional y que está por fuera del alcance del método propuesto. Se sugiere, como trabajo futuro, eliminar esta suposición.

A continuación, se detalla cada uno de los pasos de la metodología para la identificación de los agentes.

4.1 Etiquetado morfosintáctico

Sobre el documento ingresado, con las características anteriormente descritas, se realiza un análisis morfosintáctico con el fin de etiquetar cada palabra del texto de acuerdo con su categoría gramatical, es decir, la función que desempeña en su contexto. Este proceso se realiza mediante Freeling [16], herramienta que permite el procesamiento de textos en lenguaje natural, especialmente en el idioma español. Éste comprende un tokenizador para separar las palabras, un segmentador de oraciones, un etiquetador de nombres propios, y finalmente, un etiquetador morfosintáctico que, a cada palabra, le asocia una etiqueta de acuerdo con su función dentro del texto.

Adicionalmente, el Freeling posee un módulo analizador de dependencias, el cual recibe una lista de oraciones etiquetadas y las enriquece con un árbol de dependencias. Esta herramienta utiliza un conjunto de reglas manualmente definidas para construir el árbol: Primero, aplica reglas de completitud para transformar la salida del analizador de fragmentos (chunk parser) en un árbol de análisis completo (full parse tree). Luego, se convierte el árbol a dependencias, y se anotan las funciones.

La principal ventaja que posee el analizador de dependencias es que permite identificar el sujeto de la oración. Por ejemplo (ver figura 4), para la oración "El personal de ambulancia envía información al operador de radio.", el árbol de dependencias encuentra la palabra "personal" como sujeto. Pero, a su vez, establece que "El", "de" y "ambulancia" son palabras asociadas a ese sujeto, y en conjunto, forman el concepto "el personal de ambulancia".

El resultado de este proceso es un conjunto de etiquetas asociado a cada palabra y puntuación del texto, que dan información del lexema, función sintáctica y morfológica.

4.2 Lematización

Los lexemas (o lemas) constituyen la mínima unidad con significado léxico de una palabra, lo cual resulta importante para poder comparar palabras sin verse afectado por las conjugaciones, plurales u otras modificaciones posibles.

Esta información proviene del análisis realizado por el Freeling y se utiliza precisamente para poder comparar con la lista de palabras vacías.

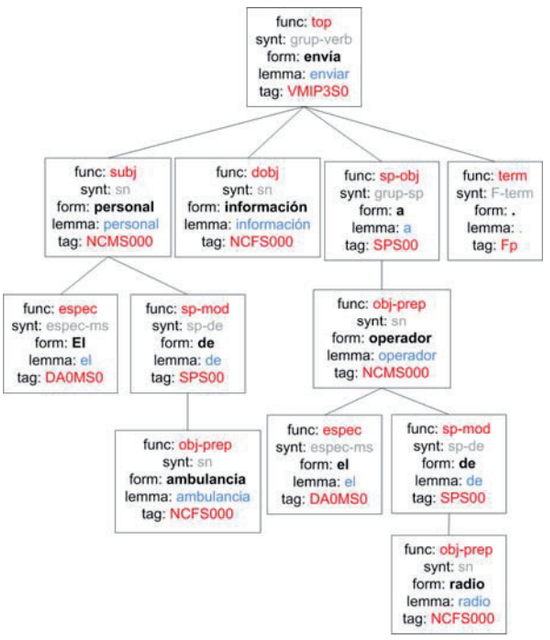

Figura 4: Árbol de dependencias para un fragmento.

4.3 Identificación de palabras vacías

Las palabras vacías son un conjunto de palabras sin significado en sí mismo, como lo son los artículos, pronombres, preposiciones, entre otros. Para identificarlas en el documento, se toman los lexemas de todas las palabras, ya que la lista considera todos los casos posibles de palabras vacías asociadas a los lexemas. Luego se compara cada una con la lista de palabras vacías del español que aparece en [17]. Las palabras que estén en ambos conjuntos se almacenan en una lista {stopwords_encontradas}.

4.4 Identificación de conceptos

Con el fin de obtener los conceptos realmente relevantes, se parte de suponer que si un agente es importante, se hablará con alguna frecuencia de él. La regla A-1, que se presenta más adelante, realiza esta selección, pero para poderse aplicar, necesita conocer la ocurrencia y frecuencia de cada palabra. Por tanto, se toma el conjunto de palabras del documento eliminando las palabras vacías que se identificaron en el paso anterior; luego, se cogen los lexemas y se realizan los siguientes cálculos: el número total de palabras Σ en los documentos, el número de ocurrencias Oi de cada palabra, la frecuencia F de cada palabra utilizando la ecuación (1).

$$F = Oi / \Sigma \qquad (1)$$

Para esto, hay que tener en cuenta que los cálculos se realizan con los lexemas. Por ejemplo, si en alguna parte del texto aparece "controlador" y luego "controladores", constituirían dos ocurrencias, pues el lexema para ambas es "controlador".

Por su parte, los agentes candidatos deben ser sujetos activos, es decir, que realicen alguna acción. Para identificarlos, se toman los conceptos que la herramienta

de etiquetado identifique como sujetos de la oración y se almacenan en la lista {conceptos}.

5 Reglas para la identificación de agentes

Una vez se tiene una lista refinada de conceptos, se aplican las siguientes reglas con el fin de identificar los agentes.

5.1 Regla A-1: Eliminar poco frecuentes

Si un concepto ocurre sólo una vez en el documento o su frecuencia es menor que el 2%, se puede ignorar este concepto.

5.2 Regla A-2: Identificar agentes de software

Si un concepto se relaciona con conceptos como "software", "sistema", "artefacto" o "dispositivo", se asigna a la lista de {agentes-de-software}. Para esto, se observan los últimos niveles en la lista de hiperónimos, es decir, la categoría conceptual más genérica que abarque cada concepto, según Multilingual Central Repository (MCR) [18]. Si la lista {hiperónimos} del concepto en cuestión, contiene alguno de los siguientes términos {sistema, método, código, equipo, equipamiento, instrumento, instrumental, utillaje, artefacto, cosa, objeto, objeto físico, objeto inanimado, entidad}, se puede decir que es un candidato a agente de software. La lista anterior sale de evaluar los hiperónimos de "software", "sistema" y "dispositivo", ya que los agentes de software pueden ser componentes automáticos, por ejemplo, dispositivos electrónicos que tengan alguna función dentro del sistema.

5.3 Regla A-3: Identificar agentes de ambiente

Si un concepto se relaciona con personas, se asigna a la lista de {agentes-de-ambiente}. Se parte de la afirmación que hace Annie Antón [19] cuando asegura que un agente es software o persona(s), y como ya se tienen los agentes del software a desarrollar, quedarían las personas que interactúan con él. Naturalmente, pueden existir elementos de software que sean externos al sistema, y éstos, a pesar de ser software, se deberían catalogar como agentes de ambiente. Sin embargo, suponiendo que el software a desarrollar interactuará exclusivamente con humanos, sería válido decir que todo agente de ambiente se refiere a personas. De esta forma, se busca dentro de MCR si cada concepto, en sus últimas categorías de hiperónimos, se relaciona con "ser_vivo", "humano", "agente", "agente_causal", "individuo", "organización", "grupo_social".

6 Validación

Este método se probó en tres casos de estudio que describen los requisitos de un sistema a desarrollar, los cuales han sido descritos previamente en la literatura: el sistema de la ambulancia [3], el sistema de una pizzería [20] y el caso de estudio de un sistema asociado a un elevador [4]. Las adaptaciones que se hicieron a esos casos de estudio y que se utilizaron como pruebas se muestran en la figura 5.

Caso de Estudio 1: Ambulancia

El personal de ambulancia recibe instrucciones de localización del operador de radio. El personal de ambulancia envía información al operador de radio. El asistente de control atiende las llamadas de emergencia. El asistente de control llena el formulario del incidente. El asistente de control indica en el formulario la localización del incidente y la hora en la que la llamada fue hecha. El tiempo necesario para atender la llamada y llenar el formulario del incidente no deben tomar más de cierto tiempo. El localizador de recursos examina el formulario del incidente. El localizador de recursos identifica cuál es la ambulancia más cercana. El localizador de recursos selecciona la ambulancia más cercana al incidente. El localizador de recursos acepta propuesta del CAD. El localizador de recursos rechaza propuesta del CAD. El sistema CAD propone al localizador de recursos la ambulancia más apropiada. El sistema CAD puede proponer otra ambulancia, en caso de que el localizador de recursos o el asistente de control rechacen la propuesta inicial. El despachador llama al personal de ambulancia, si ésta se encuentra en la estación. El despachador le da los datos al personal de ambulancia para llegar al lugar del incidente. El despachador da información al operador de radio para que guíe a la ambulancia. El operador de radio recibe información de la ambulancia seleccionada. El operador de radio da instrucciones de localización al personal de ambulancia. El operador de radio recibe información del personal de ambulancia.

Caso de Estudio 2: Pizzería

El gerente de la pizzería desea aumentar la rentabilidad del negocio. La pizzería tiene un despachador, un cocinero y dos repartidores. La pizzería ofrece a los clientes diversos tipos y tamaños de pizza. También, la pizzería ofrece a los clientes la posibilidad de ordenar aditivos para la pizza. Los clientes pueden ordenar aditivos para la pizza. La pizzería posee una zona de cobertura determinada. La pizzería tiene una promoción que consiste en que la pizzería debe entregar el pedido al cliente en menos de 30 minutos, de lo contrario la pizzería debe regalar el pedido al cliente. La pizzería envía a los repartidores a entregar la pizza. Los repartidores entregan la pizza en menos de 30 minutos, de lo contrario, la pizzería cobra a los repartidores el pedido. Así, la pizzería evita la pérdida de dinero.

Caso de Estudio 3: Ascensor

La compañía encargada del elevador está en la capacidad de proveer una forma de escape. La compañía provee una interfaz basada en botones. La compañía garantiza la existencia de un botón de emergencia. La compañía construye la estructura del ascensor. La compañía garantiza energía de emergencia. La compañía asegura el funcionamiento del software. El controlador del elevador garantiza que las puertas no se abran mientras esté en movimiento. El controlador detiene el elevador, si hay una falla de energía. El controlador enciende la luz de emergencia cuando sea necesario. El controlador puede abrir las puertas, cuando esté en el nivel indicado. El controlador informa a los pasajeros del

estado de su petición. El controlador reporta a los pasajeros las condiciones de sobrepeso.

Para visualizar más fácilmente cómo funciona el método, se muestran a continuación los resultados de cada paso, para el caso de estudio de la ambulancia: Del etiquetado morfosintáctico, sin tener en cuenta las etiquetas para la puntuación, se obtienen las etiquetas respectivas de las 241 palabras, de las cuales 96 son palabras vacías y se identifican los 6 sujetos de las oraciones que se convierten en conceptos candidatos.

Después se almacenan hiperónimos para cada uno de los conceptos de la lista anterior, o en caso de no arrojar ningún resultado, de cada palabra no vacía de cada concepto. Así, queda la siguiente lista de agentes candidatos: {asistente de control, personal de ambulancia, operador de radio, sistema CAD, despachador}. Aplicando la regla A-1, no se eliminan conceptos, pues todos son frecuentes en el texto. Con la A-2 se obtiene el agente de software {sistema CAD}. Si bien CAD no aparece dentro del repositorio MCR, sí se encuentra que sistema se refiere a un artefacto/objeto/instrumento. La aplicación de la regla A-3 arroja la siguiente lista de agentes de ambiente: {asistente de control, personal de ambulancia, operador de radio}. Los agentes {despachador, localizador de recursos} no se logran clasificar, ya que no aparecen dentro del repositorio. Así, se identifican los 6 agentes que están explícitos en la descripción del sistema: personal de ambulancia, operador de radio, asistente de control, localizador de recursos, sistema CAD y despachador; los cuales corresponderían a los nombres de cada agente. Sin embargo, solo se logra recuperar la categoría de 4 de ellos.

En la figura 5, se muestran dos gráficas. En la primera, se presentan los agentes que el método permitió encontrar en los tres casos de estudio, y lo compara con el total de agentes realmente presentes, previamente inferidos por un analista. Con el fin de evaluar el desempeño del método, en la segunda gráfica se mide su eficiencia (ver ecuación (2)).

$$\eta_{agt} = \left(1 - \frac{|Agentes\ hallados - Agentes\ presentes|}{Número\ de\ agentes\ en\ la\ muestra}\right) \times 100 \quad (2)$$

La ecuación evalúa la diferencia entre el conjunto de agentes hallados y agentes presentes, pues interesa conocer cuáles de los agentes hallados no pertenecen al conjunto de agentes presentes; luego, este número se divide por el número de agentes presentes, se resta a uno y se multiplica por 100, para así obtenerlo como un porcentaje de efectividad. De esta forma, si no hay diferencia entre ambos conjuntos, el índice sería de 100%.

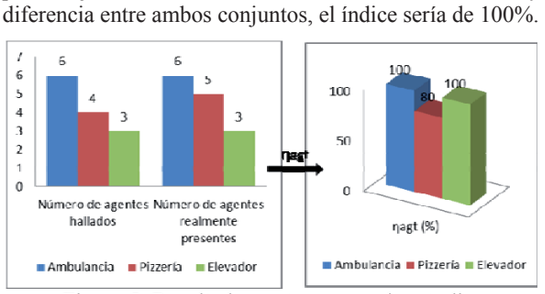

Figura 5: Resultados en tres casos de estudio.

De los tres casos de estudio, el de la pizzería fue donde el método tuvo el peor desempeño. Precisamente, ese caso de estudio realiza una descripción estructural del sistema, más no funcional. Es decir, afirma que "la pizzería tiene un repartidor, un cocinero, un despachador" (descripción de la estructura de la organización), pero no clarifica las funciones que realiza cada uno (descripción funcional). De hecho, conceptos como "cocinero" se mencionan sólo una vez, y al aplicar la regla A-1, se eliminan. Seguramente, si se hablara más de las funciones que desempeña cada uno, aparecerían con más frecuencia. De hecho, un agente debe realizar alguna acción, pues es un elemento activo [21].

En general, se puede decir que la metodología arroja resultados satisfactorios, sin embargo, presenta problemas cuando no se describen los agentes suficientemente.

7 Conclusiones y trabajos futuros

Este trabajo presenta un método para la identificación de agentes bajo el enfoque KAOS. Se propone utilizar Freeling para el etiquetado morfosintáctico y la lematización. Además, se proponen tres reglas para identificarlos y clasificarlos. Sobresale el uso de los hiperónimos de los agentes candidatos, en el MCR, con el fin de definir si son de software o de ambiente. A través de tres casos de estudio, se muestra que las reglas cumplen el objetivo satisfactoriamente. Sin embargo, deben realizarse más pruebas.

El método propuesto en este artículo inicia con el etiquetado morfosintáctico de un texto escrito en español con la descripción de agentes en voz activa. Luego, se eliminan palabras vacías del documento para más adelante poder realizar cálculos de frecuencias y ocurrencias de palabras, y agrupar conceptos. Se encuentran los sujetos de las oraciones, los cuales se almacenan en una lista de conceptos candidatos. Posteriormente, se hallan los hiperónimos de cada concepto, con lo cual se identifica si los conceptos hacen referencia a agentes de software o de ambiente, dependiendo si semánticamente se relacionan con artefactos o humanos. Finalmente, se proponen 3 reglas para eliminar conceptos que no son agentes y clasificar los agentes.

A partir de este artículo, se generan nuevos asuntos que pueden dar continuidad al trabajo realizado, los cuales se citan a continuación:

(i) Queda por establecer la asignación de agentes a objetivos, ya que dicha descripción es tarea del analista, siendo necesario primero evaluar las características de cada agente y así determinar si está en capacidad de alcanzar dicho objetivo o no.

(ii) Otro elemento importante a tratar de resolver es el de la anáfora, ya que no siempre se estará repitiendo en el lenguaje natural el sujeto de la acción, sino que se suele recurrir al uso de los pronombres para referirse a ellos. En los casos de estudio se repetía siempre cada agente, por ejemplo: "El personal de ambulancia recibe instrucciones de localización del operador de radio. El personal de ambulancia envía información al operador de radio". Una oración más natural sería la siguiente "El personal de

ambulancia recibe instrucciones de localización del operador de radio. Además, éste envía información al operador de radio".

(iii) Definir reglas adicionales que puedan generar variaciones en los agentes identificados.

Agradecimientos

Este artículo se realizó en el marco del proyecto de investigación: "Un modelo de procesamiento terminológico para la obtención de requisitos de software basado en el diagrama de objetivos de KAOS" código quipú 20101008208, financiado por la Dirección de Investigaciones de la Universidad Nacional de Colombia Sede Medellín DIME, a través de la convocatoria "DIME 2012 financiación de proyectos de investigación".

Referencias bibliográficas

[1] A. Lapouchnian, "Goal-Oriented Requirements Engineering : An Overview of the Current Research," 2005.

[2] A. Van Lamsweerde, "Goal-Oriented Requirements Engineering : A Guided Tour," 2001.

[3] E. Letier, "Reasoning about Agents in Goal-Oriented Requirements Engineering Reasoning about Agents in Goal-Oriented Requirements Engineering," 2001.

[4] Respect IT, "A KAOS Tutorial," *Objectiver*, pp. 1–46, 2007.

[5] A. Dardenne, S. Fickas, and A. Van Lamsweerde, "Goal-directed Concept Acquisition in Requirements Elicitation," vol. 97403, pp. 14–21, 1991.

[6] M. Ibrahim and R. Ahmad, "Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques," *2010 Second International Conference on Computer Research and Development*, pp. 200–204, 2010.

[7] I. Song and K. Yano, "A taxonomic class modeling methodology for object-oriented analysis," *Information Modeling Methods and Methodologies*, pp. 216–240, 2005.

[8] S. D. Joshi, "Textual Requirement Analysis for UML Diagram Extraction by using NLP," vol. 50, no. 8, pp. 42–46, 2012.

[9] L. A. Lezcano, J. A. Guzmán, and S. A. Gómez, "Caracterización de los Elementos del Diagrama de Objetivos de KAOS a partir de Lenguaje Natural," in *IX Congreso Internacional de Electrónica y Tecnologías de Avanzada*, 2012.

[10] C. M. Zapata, L. Lezcano, and P. A. Tamayo, "Validación del Método para la Obtención Automática del Diagrama de Objetivos desde Esquemas Preconceptuales," *Revista EIA*, no. 8, pp. 21–35, 2007.

[11] UML, "Unified Modeling Language Specification," *Object Management Group (OMG)*, 2004.

[12] I. S. S. Brito, "Aspect-Oriented Requirements Analysis," Universidade Nova de Lisboa, 2008.

[13] R. Matulevičius and P. Heymans, "KAOS Construct Analysis using the UEML Approach Template," Namur, Belgium, 2005.

[14] X. Zhou and N. Zhou, "Auto-generation of Class Diagram from Free-text Functional Specifications and Domain Ontology," pp. 1–20, 2008.

[15] P. More, "Generating UML Diagrams from Natural Language Specifications," vol. 1, no. 8, pp. 19–23, 2012.

[16] L. Padró and E. Stanilovsky, "Freeling 3.0: towards wider multilinguality," in *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, 2012, pp. 2473–2479.

[17] Tartarus, "Snowball," 2012. [Online]. Available: http://snowball.tartarus.org/algorithms/spanish/stop.txt. [Accessed: 18-Jun-2013].

[18] A. Gonzalez-Agirre, E. Laparra, and G. Rigau, "Multilingual Central Repository version 3.0," *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pp. 2525–2529, 2012.

[19] A. Antón, "Goal Identification and Refinement in the Specification of Software-Based Information Systems," Georgia Institute of Technology, Atlanta, 1997.

[20] C. M. Zapata, S. Villegas, and F. Arango, "Reglas de consistencia entre modelos de requisitos de UN-Método," *Revista Universidad Eafit*, vol. 42, no. 141, pp. 40–59, 2006.

[21] R. Matulevicius and P. Heymans, "Analysis of KAOS Meta-model," *Computer Science Department, Namur University*. Reporte técnico, Bélgica, 2005.