Información Tecnológica
Vol. 26(6), 129-138 (2015)
doi: 10.4067/S0718-07642015000600015

Un Modelo para la Identificación de Elementos KAOS (Especificación Automática de Adquisición de Conocimientos) a partir de la Especificación de Requisitos en Lenguaje Natural

Luis A. Lezcano*, Jaime A. Guzmán y Carlos A. Vélez
Universidad Nacional de Colombia Sede Medellín. Carrera 80 No 65-223 (Núcleo Robledo) Medellín, Colombia. (e-mail: lalezcan@unal.edu.co; jaguzman@unal.edu.co; caavelezca@unal.edu.co)

* Autor a quien debe ser dirigida la correspondencia

Recibido May. 7, 2015; Aceptado Jun. 10, 2015; Versión final Jul. 30, 2015, Publicado Dic. 2015

Resumen

En este artículo, se propone un modelo para obtener de manera automática los elementos básicos del diagrama de objetivos de KAOS (objetivo, agente y entidad) y minimizar la ambigüedad semántica de tipo polisémica en la obtención de dichos elementos. Además, se valida el modelo propuesto mediante casos de estudio. El diagrama de objetivos de KAOS (Especificación Automática de Adquisición de Conocimientos; del inglés Knowledge Acquisition Automated Specification) suele ser utilizado en la fase de definición y análisis de la ingeniería de software para determinar los procesos que se efectúan en la organización que solicita el software. En los diferentes trabajos publicados en la literatura científica que usan el diagrama de objetivos de KAOS subsisten algunos problemas causados por la elaboración manual del diagrama por parte del analista (confusiones, subjetividad, ambigüedad, entre otros). Con la metodología propuesta los elementos básicos del diagrama de objetivos de KAOS se obtienen de manera automática evitando errores como los mencionados.

Palabras clave: ingeniería de software; lenguaje natural, validación de requisitos, desambiguación semántica, KAOS

A Model for the Identification of KAOS Elements (Knowledge Acquisition Automated Specification) from the Specification of Requirements in Natural Language

Abstract

In this work, a model to automatically obtain the basic elements of the goal diagram known as KAOS (goal, agent, and entity) and to minimize the polysemic semantic ambiguity in getting these elements is proposed. Furthermore, the proposed model is validated by using case studies. The KAOS goal diagram (Knowledge Acquisition Automated Specification) is often used in the definition and analysis phases of software engineering to determine the processes that are carried out in the organization that request the software. In different works of the scientific literature that use the KAOS diagram some problems caused by the manual elaboration of the diagram by the software analyst are present (confusions, subjectivity, and ambiguity, among others). With the proposed methodology in this work the basic elements of the goal diagram KASOS are obtained in automatic form avoiding the errors mentioned above.

Keywords: software engineering; natural language; validation of requirements; semantic disambiguation; KAOS

INTRODUCCIÓN

En la ingeniería de software los requisitos comprenden el conjunto de afirmaciones prescriptivas que el sistema a desarrollar debe satisfacer, es decir, son las necesidades de una organización que pueden ser logradas a través del software. Dentro de las metodologías existentes para la educción de requisitos, se encuentran aquellas orientadas a objetivos, cuya importancia está dada en que los objetivos permiten identificar la justificación de los requisitos (Lamsweerde, 2009). La literatura especializada en el tópico de estudio, presenta dos tipos de diagramas de objetivos, ambos utilizados en la fase de definición y análisis de requisitos: (i) el perteneciente a I*; y (ii) el correspondiente a la metodología KAOS. La metodología I* utiliza los siguientes elementos: objetivos, tareas, agentes y recursos. Éstos son representados a través de nodos relacionados entre ellos (Yu, 1995). Sin embargo, el diagrama de objetivos de I* no permite representar la jerarquía entre objetivos; ésta es necesaria para establecer el objetivo de alto nivel y realizar la subrogación que permita identificar los requisitos que deberá cumplir el software.

Por su parte, la metodología KAOS presenta: (i) una representación jerárquica de los objetivos; (ii) la forma de encadenar los objetivos con los requisitos de los interesados, con los agentes responsables de garantizar su satisfacción y las entidades necesarias para la interacción de los elementos del sistema; y (iii) facilidad de visualización, brindando así una forma de validar los requisitos entre analista e interesado. Adicionalmente, KAOS permite su articulación con la fase de diseño, en donde se utiliza con regularidad el lenguaje UML (Lamsweerde, 2009). Uno de los casos en donde se identifica dicha articulación, es en los objetivos y agentes, puesto que, a partir de éstos se logran obtener los casos de uso, los cuales son propios del diseño. Es decir, esta metodología permite una trazabilidad y consistencia durante el ciclo de vida del software. Dado lo anterior, se elige la metodología KAOS para el desarrollo de esta propuesta.

Ahora bien, aunque esta representación posee cierta semántica, se pueden presentar dificultades para la definición de los términos utilizados en el diagrama. Esto es, las palabras que se usan en los diagramas pueden tener distintos significados y, por tanto, dar lugar a diferentes interpretaciones para los requisitos. Esto es indeseable, debido a la importancia que tiene la consistencia en la ingeniería de software. Por lo tanto, se requiere minimizar la ambigüedad semántica, concretamente, la polisémica, ya que es la referida al correcto significado de una palabra (Lezcano et al., 2012). Es así, como el objetivo del artículo es: obtener de manera automática los elementos básicos que conforman el diagrama de objetivos de KAOS (objetivos, agentes y entidades), minimizar la ambigüedad semántica de tipo polisémica en dicho proceso, aproximar el proceso al discurso del interesado y permitir la interacción del usuario durante el proceso.

A continuación, se citan algunos autores de la literatura científica especializada que han usado o trabajado metodologías para la obtención o para el análisis de diagramas en la ingeniería de software, con el fin de mostrar la importancia, vigencia y diferenciación del método propuesto. Nicolás y Toval (2009) indican que las especificaciones de requisitos de software a través de lenguaje natural textual son un tema de interés que amerita dar continuidad a los trabajos de investigación realizados por la comunidad científica. Además, logran demostrar que las metodologías orientadas a objetivos juegan un papel importante en la ingeniería del software. Alves et al. (2010) señalan que la ingeniería de requisitos está ligada de forma directa con los objetivos correspondientes a un dominio de interés, en donde se definen funciones y restricciones del software futuro.

Ibrahim y Ahmad (2010) proponen una metodología para la obtención del diagrama de clases, partiendo de especificaciones en inglés, a través del procesamiento de lenguaje natural. Sin embargo, está enfocada en dicho diagrama, que hace parte de la fase de diseño, mientras que en la presente propuesta se trabaja sobre KAOS, que pertenece a la fase de definición y análisis. Además, en éste no se realiza ningún proceso de desambiguación semántica. Así, el método propuesto, abarca este aspecto. Barachisio et al. (2010) sostienen que la ejecución del proceso de análisis de requisitos en un dominio, sin el apoyo de una herramienta computacional, puede conducir a resultados infructuosos, incluso en áreas como la ingeniería de software y la ingeniería de requisitos. Esto demuestra la importancia del método para la automatización de los procesos de educción de requisitos.

Casagrande et al. (2014) presentan un método para la educción de objetivos a partir del procesamiento de lenguaje natural (NLP) en el contexto de KAOS. Para ello, emplearon técnicas de minería de datos que permiten apoyar al analista en la extracción y el modelado de objetivos. Concluyen indicando que los resultados obtenidos son buenos, pero que se requiere una mayor investigación lingüística para que la minería de datos se pueda ampliar además de objetivos, a requisitos, conceptos, restricciones, entre otros. A diferencia del método propuesto, allí se enfocan especialmente en la obtención de objetivos y su subrogación.

Lezcano (2014) y Lezcano et al. (2015) proponen un modelo para la obtención de cuatro elementos del diagrama KAOS, con resultados aceptables, aunque la desambiguación semántica que se realiza presenta un bajo desempeño. Finalmente, Puello, et al. (2014) presentan un trabajo en donde buscan diagnosticar la calidad de PSP (Personal Software Process). En éste utilizaron variables correspondientes al proceso de desarrollo, las cuales articularon con metodologías que presentan un enfoque sistemático, necesario para la verificación del conocimiento. En dicha investigación, lograron determinar que el modelo de diagnóstico de PSP es flexible para incorporar mejoras que permitan aumentar su precisión. Dado lo anterior, y considerando que el diagrama de objetivos de KAOS es utilizado en la fase de definición de requisitos, es decir, antes del desarrollo de software, se logra determinar la pertinencia del modelo propuesto en este artículo, puesto que, se podría aumentar la calidad y precisión al PSP si se utiliza la metodología KAOS con los aportes obtenidos en el modelo propuesto.

ARQUITECTURA DEL MODELO PROPUESTO

El modelo consta de tres módulos: el morfosintáctico, el semántico y el de recálculo. El insumo de entrada son las especificaciones de los requisitos, redactados en idioma español –sin errores ortográficos–, y el conjunto de las reglas morfosintácticas para la identificación de los elementos KAOS. La salida es el conjunto de elementos identificados con su desambiguación semántica. En este artículo, se analiza para los siguientes elementos de la metodología KAOS (Guzmán et al., 2013): objetivos, agentes y entidades. El sistema que implementa este modelo se denominó “NEONPL”. Su arquitectura se muestra en la figura 1 y sus componentes se explican a continuación.

Inserción de reglas morfosintácticas

Los datos de entrada del modelo contemplan dos elementos: (i) la especificación en lenguaje natural de los requisitos de la organización; y (ii) las reglas morfosintácticas, las cuales son sentencias que describen la estructura que debe tener la especificación, para que el sistema identifique los elementos KAOS. Cada regla se define a través de un antecedente, que establece la estructura sintáctica de la frase (verbi gratia, si primero va un sustantivo que un verbo, o una palabra específica) que describe el elemento en lenguaje natural, y un consecuente, que indica la forma en que se muestra el elemento identificado. Cada regla se asocia a un elemento de KAOS específico.

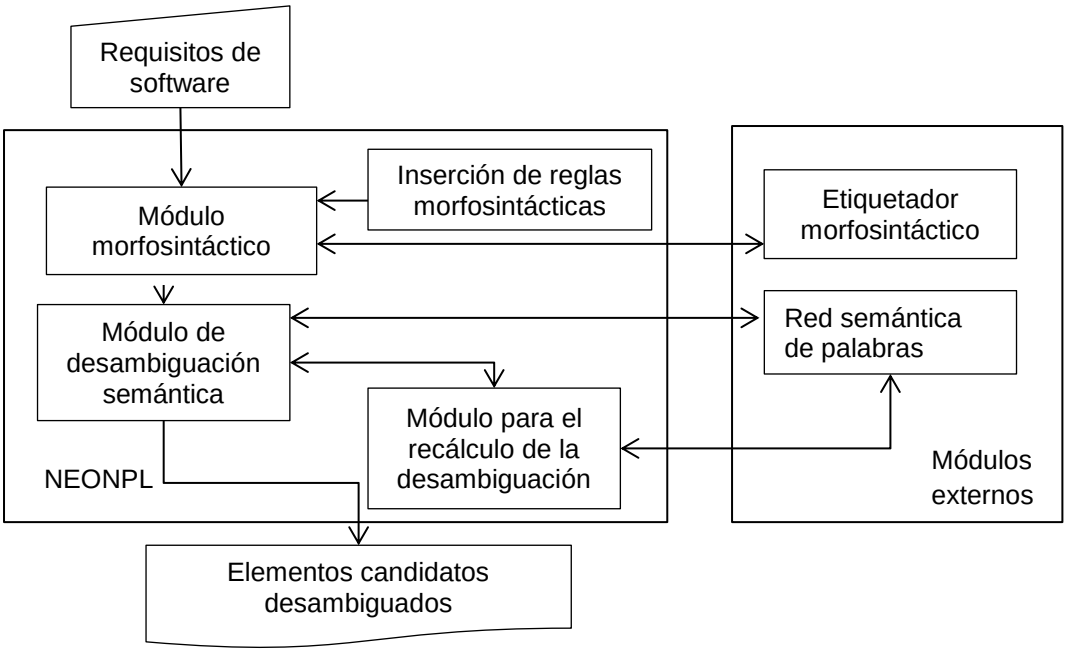

Fig. 1. Arquitectura del sistema propuesto. Fuente: elaboración propia.

En la tabla 1 se detallan las reglas para la identificación de elementos KAOS desde el lenguaje natural. Si bien imponen restricciones al lenguaje de entrada, convirtiéndose en un lenguaje controlado, no es completamente cerrado, ya que el análisis morfosintáctico tiene en cuenta las conjugaciones, declinaciones y entidades de nombre.

Tabla 1. Reglas para la identificación de objetivos, agentes y entidades KAOS (Guzmán et al., 2013).

Regla	*Antecedente* *(Estructura que debe aparecer en la frase)*	*Consecuente* *(Estructura que se extrae de la frase)*	*Elemento*
FMOB1	Sintagma Nominal + Perífrasis verbal modal + complemento.	Verbo principal + Complemento	Objetivo
FMOB2	Que + Sintagma Nominal + Verbo no copulativo + Complemento	Verbo principal + Complemento	Objetivo
FMOB3	Verbo en infinitivo + que + Sintagma Nominal + Verbo no copulativo + Complemento	Verbo principal + Complemento	Objetivo
FMOB4	Sintagma Nominal + Perífrasis verbal modal + Verbo auxiliado + Complemento	Verbo principal + Complemento	Objetivo
FMAG	Sustantivo o sintagma nominal + Verbo no copulativo + Complemento	Sintagma nominal sin artículos	Agente
FME1	Sustantivo 1 + preposición (de/del) + Sustantivo 2	Sustantivo 2 lematizado	Entidad
FME2	Sustantivo 1 + Verbo (tiene) + Sustantivo 2	Sustantivo 1 lematizado	Entidad

Por ejemplo, la regla FMOB4 indica que su antecedente está conformado por una frase que deberá contener los siguientes elementos sintácticos en su respectivo orden: sintagma nominal, una perífrasis verbal modal y un complemento. El consecuente, que indica el texto del elemento KAOS identificado, se forma con: el verbo principal en infinitivo y su complemento.

En la figura 2 se detalla la interfaz de este módulo que permite generar las reglas anteriormente descritas. Para tal fin, se permite identificar el nombre del elemento. La regla se genera mediante una sección que representa el antecedente, conformado por una lista ordenada de elementos. Estos elementos pueden ser de dos tipos: uno asociado a la categoría gramatical, y un segundo tipo asociado a una palabra de tipo libre o específica. Una palabra libre es, como su nombre lo indica, cualquier palabra ingresada por el usuario; la específica es alguna palabra en particular que el usuario debe indicar. El orden de esta lista representa el orden en que deberán ir los elementos de la frase. Por su parte, para el consecuente, se seleccionan componentes del antecedente, indicando así la forma en que se presenta el elemento identificado. En la interfaz de usuario se adicionó una descripción, para escribir textualmente cómo quedará la regla.

Lo siguiente es que el sistema identifique los elementos desde la lista de frases etiquetadas, haciendo uso de las reglas definidas. Esto implica para cada frase verificar si cumple o no con las reglas. El algoritmo propuesto que implementa esta función se presenta en la figura 3. Así, el resultado de este módulo es una lista de elementos, donde cada uno se compone de una lista de palabras que, a su vez, poseen una etiqueta morfosintáctica.

Módulo de desambiguación semántica

Los elementos KAOS identificados pretenden ser una representación formal que elimine la ambigüedad propia del lenguaje natural. Por lo tanto, en la detección automática de estos elementos es necesario realizar un proceso de desambiguación. El tratamiento del lenguaje se puede estudiar desde seis categorías: fonética, morfología, sintaxis, semántica, pragmática y análisis del discurso. En cada una de ellas, se puede dar lugar a ambigüedades. Se dice que un texto es ambiguo cuando existen diferentes estructuras lingüísticas posibles para un mismo texto (Jurafsky y Martin, 2000). En este modelo se propone trabajar el problema de la ambigüedad semántica, específicamente, la polisémica, que se define como aquella que se origina cuando una palabra tiene diferentes significados, y se debe llegar al significado correcto según el contexto e intencionalidad del emisor del mensaje (Navigli, 2009).

Fig. 2. Ejemplo de inserción de regla. Fuente: elaboración propia.

```
1. function identificarElementos(S, user_name) returns List<Elemento>
2.      foreach (Si ∈ S) { //Para cada oración Si de la lista de oraciones identificadas en el texto de entrada:
3.          foreach (Rk ∈ Reglas(user_name)) { //Para cada regla Rk definida por el usuario:
4.              aciertos ← 0;
5.              foreach (Al ∈ Rk.antecedente) { //Para cada variable del antecedente de la regla actual:
6.                  foreach (Wj ∈ Si) //Para cada variable del antecedente de la regla actual:
7.                      if (Wj.tag = Al){ // La regla se está cumpliendo
8.                          aciertos++;
9.                          j ← S.size(); //Salir del ciclo de la línea 6.
10.                     } else { // Ya no cumple la regla
11.                         j ← S.size(); //Salir del ciclo de la línea 6.
12.                         l ← Al.size(); //Salir del ciclo de la línea 5.
13.                     }
14.                 }
15.             }
16.             if (aciertos = Al.size()) { //Si se cumple la regla (hay tantos aciertos como elementos del antecedente)
17.                 texto ← "";
18.                 foreach (Cm ∈ Rk.consecuente) { //Para cada variable del consecuente de la regla actual:
19.                     texto.append( SCm.getLemma());//El texto del consecuente se construye con palabras de la
20.                     // frase donde se identifica el elemento.
21.                 }
22.                 elementos.add( new Elemento(texto));
23.             }
24.         }
25.     }
26.     return elementos;
27. endfunction
```

Fig. 3. Algoritmo para la identificación de elementos. Fuente: elaboración propia.

Para la desambiguación semántica, se utilizaron en este modelo las siguientes técnicas: MFS (Preiss et al., 2009), UKB (Agirre y Soroa, 2009) y Lesk adaptado (Banerjee y Pedersen, 2002). Los pseudocódigos simplificados que corresponden a los algoritmos de las técnicas antes enunciadas se pueden observar en la tabla 2. El Lesk adaptado realiza un conteo comparando la definición de cada sentido posible de cada palabra a desambiguar, con una bolsa de palabras compuesta por las definiciones de las palabras en el contexto y sus respectivos: sinónimos (relación de equivalencia), hiperónimos (generalización), holónimos (especialización) y merónimos (parte de). Por su parte, el MFS, al ser el sentido más frecuente, no tiene en cuenta estas relaciones semánticas. Finalmente, el UKB ordena la lista de sentidos posibles teniendo en cuenta primero los sentidos que son más cercanos semánticamente. La cercanía se mide a través del algoritmo BFS para el recorrido y ordenando para hallar la ruta más corta.

El contexto para desambiguar una palabra puede definirse de distintas formas. Una de ellas es tomar las demás palabras funcionales (no vacías) de la oración en la que ella se encuentra. De esta forma, los algoritmos utilizados en esta propuesta son completamente escalables a cualquier cantidad de frases en el documento de entrada.

Estas relaciones semánticas entre los sentidos de las palabras se pueden consultar en una red semántica, siendo utilizado la de Multilingual Central Repository (González et al., 2012). Las consultas se realizan recuperando las definiciones y relaciones semánticas del sentido requerido. En el módulo anterior se obtuvo la lista de elementos identificados y las palabras que los componen. Después de este módulo semántico, lo que se tiene es esta lista de palabras, pero enriquecida con el mejor sentido, de acuerdo con el algoritmo indicado por el usuario.

Tabla 2. Algoritmos de desambiguación Adapted Lesk, MFS y UKB presentados en forma simplificada.

```
function AdaptedLesk (phrase){
 foreach wi ϵ phrase {
   foreach (sensej ϵ wi){
   foreach (sensek ϵ wi & k != j) {
scoreij[j][k] ← overlap(ada_senses(j),
                        ada_senses(k));
    }}
  }
 //Ordena de mayor a menor score
 return score.sort();
}

function ada_senses (i) {
return
definition(synonyms(senses(i))

+hyperonyms(senses(i))
              +holonyms(senses(i))

+meronyms(senses(i)));
}
```

```
function MFS (word){
 dictionary ← loadMFS();
 senses                          ←
dictionary.get(word).senses;
 return senses;
}
function loadMFS(){
  dictionary ← dictionary<>;
  kb
←load(scores_senses_database);
  foreach k in kb {
    if(!dictionary.contains(k)){
      dictionary.put(k,          k.lemma,
k.sense);
    }
    else{
      dictionary.setscore(k, k.score++);
    }
  }
//Ordena según el sentido más
frecuente
  return dictionary.sort(x => x.score);
}
```

```
function UKB (phrase){
 kb ←loadWN();
 for i in 1..kb.size {
  for j in kb(j)-context..kb(j)+context {
    sensesj[] ← synsets(kb(j));
    Gkb ← relevant_concepts(kb(j) –
            context, kb(j) + context);
    for k in 1..sensesj.size {

shortest_path←BFS(senses(k),Gkb);
     Gd.add(shorest_path);
    }
   }
   score ← PageRank(Gd);
   dictionary.add(kb(i), score);
 }
 return dictionary;
}
```

Salida del sistema

Ya que se tienen los elementos identificados y los conceptos desambiguados, se presentan los resultados al usuario. En la figura 4 se detalla la interfaz asociada a este módulo para un caso específico. En su parte superior se detalla el título que asocia la frase original del texto y los elementos identificados. En la parte inferior, para cada palabra que se asocia a la frase se asocia una columna que contiene el lema de la palabra, su etiqueta que identifica internamente en el sistema la palabra, el sentido de la palabra que relaciona el identificador en la red semántica de palabras y la definición que denota el texto, extraída de dicha red semántica.

Como se puede observar, ésta no es la representación final del diagrama KAOS, pero es un paso intermedio y necesario en la identificación de los elementos candidatos presentes en la definición del sistema por parte del interesado. De esta forma, el modelo propuesto en este artículo lo que permite es automatizar el proceso de obtención de los elementos candidatos y su desambiguación semántica polisémica; además de presentarlos al analista para su validación.

Módulo para el recálculo de la desambiguación

Con este módulo, se busca brindarle al usuario final la posibilidad de recalcular los sentidos hallados. Para esto, según el algoritmo de desambiguación elegido por el usuario, el sistema recalcula el siguiente sentido de la palabra elegida, de acuerdo con el score obtenido en la desambiguación.

PRUEBAS DE VALIDACIÓN

Según (Navigli, 2009), la validación de los sistemas de clasificación, como es el caso de las técnicas de desambiguación semántica, suele medirse a través de las siguientes métricas: cobertura-*C* (1), recuerdo-*R* (2), precisión-*P* (3) y métrica F1 (4). En este contexto, la cobertura se define como el porcentaje de elementos de software que tienen un resultado por parte del sistema, sobre la cantidad total de elementos; el recuerdo es el porcentaje de elementos que han sido correctamente desambiguados con respecto a la totalidad de elementos presentes en el texto ingresado por el usuario; la precisión es el porcentaje de elementos correctamente desambiguados, frente a los elementos que fueron desambiguados (no la totalidad de ellos). Finalmente, la métrica F1, comprendida entre 0 y 1, indica la efectividad de la desambiguación, al agrupar la precisión y el recuerdo.

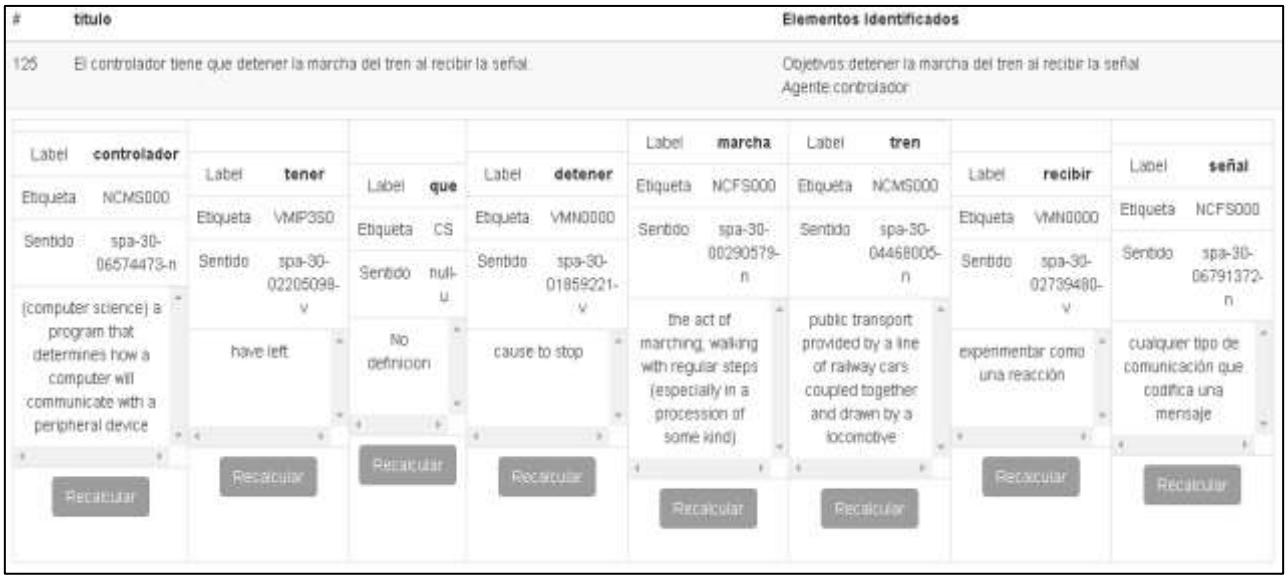

Fig. 4. Ejemplo de resultado y opción de recalcular un sentido. Fuente: elaboración propia

Su formulación matemática se observa en las ecuaciones (1) a (4), respectivamente. En la descripción de estas tres métricas, *T*, es la cantidad total de elementos KAOS del conjunto; *V*, es el número de elementos KAOS correctamente desambiguados; y *U*, el número total de elementos KAOS desambiguados.

$$\text{Cobertura:} \quad \mathrm{C}_{wsd} = \frac{U}{T} \times 100 \tag{1}$$

$$\text{Recuerdo:} \quad \mathrm{R}_{wsd} = \frac{V}{T} \times 100 \tag{2}$$

$$\text{Precisión:} \quad \mathrm{P}_{wsd} = \frac{V}{U} \times 100 \tag{3}$$

$$\text{Métrica F1:} \quad \mathrm{F1}_{wsd} = \frac{2 * P * R}{P + R} \tag{4}$$

Con base en los parámetros descritos en las ecuaciones (1) a (4), se proponen las siguientes definiciones: (i) un elemento KAOS se considera desambiguado si posee un sentido para cada palabra funcional que lo compone, es decir, si todas las palabras no vacías (e.g. adjetivos, verbos, sustantivos, adverbios) poseen un sentido asociado; (ii) un elemento se considera correctamente desambiguado, si y sólo si, todos los sentidos de las palabras funcionales que lo componen son correctos, según el contexto entregado por usuario; y (iii) un elemento se considera no desambiguado si al menos uno de los sentidos de las palabras funcionales que lo componen no posee un sentido asociado.

La validación del modelo propuesto se realizó utilizando la metodología KAOS (Lamsweerde, 2001, 2009), en la obtención de objetivos, agentes y entidades. Asimismo, fueron utilizados tres casos de estudio registrados en la literatura científica: sistema de ambulancia de Londres (C1) (Letier, 2001), sistema de elevador (C2) (Respect IT, 2007), y el caso Carrefour (C3) (Lezcano, 2014).

En el experimento se midieron las cuatro métricas y definiciones descritas anteriormente. En la tabla 3 se presentan los resultados obtenidos. Además, se establecen comparaciones entre los tres algoritmos de desambiguación. De acuerdo con los resultados presentados en dicha tabla, la precisión del Lesk adaptado presenta graves falencias en su cobertura y, por lo tanto, en el recuerdo y en F1. Esto se explica principalmente porque el algoritmo es totalmente dependiente de las palabras contenidas en las definiciones de los sentidos. Así, la deficiencia de glosas o definiciones en las redes semánticas para el español, afectan considerablemente el desempeño de las técnicas relacionadas con Lesk.

Tabla 3. Resultados de la desambiguación en elementos del diagrama KAOS

Elementos KAOS	*Ambulancia*			*Ascensor*			*Carrefour*		
	MFS	*UKB*	*ALesk*	*MFS*	*UKB*	*ALesk*	*MFS*	*UKB*	*ALesk*
Totales	16			11			10		
Desambiguados	15	15	1	10	10	1	8	8	2
Correctamente desambiguados	9	12	1	8	9	1	5	5	1
Cobertura	93.75%	93.75%	6.25%	90.91%	90.91%	9.09%	80.00%	80.00%	20.00%
Precisión	60.00%	80.00%	100.00%	80.00%	90.00%	100.00%	62.50%	62.50%	50.00%
Recuerdo	56.25%	75.00%	6.25%	72.73%	81.82%	9.09%	50.00%	50.00%	10.00%
F1	58.06%	77.42%	11.76%	76.19%	85.71%	16.67%	55.56%	55.56%	16.67%

Con respecto al MFS y UKB, como requieren un entrenamiento previo en un corpus etiquetado, las falencias de la red no afectan su desempeño en la misma medida que el Lesk. Ahora, si bien el MFS y el UKB logran desambiguar la misma cantidad de elementos en los tres casos de estudio, el UKB tiene mejores resultados: en promedio, se obtiene una precisión del 67.50% y un recuerdo del 59.66%, para el MFS; y del 77.50% y 68.94% respectivamente para el UKB. Si se realiza la validación por cada palabra, y no por el sentido completo de la frase, se tiene una precisión del 78.57% y del 73.81% para el MFS y UKB, respectivamente; y un recuerdo del 79.37% y 82.22% para los mismos.

CONCLUSIONES

Con el trabajo presentado en este artículo se logró: (i) implementar un modelo que permite el procesamiento de textos en la fase de definición del ciclo de vida del software; (ii) articular a la metodología KAOS con herramientas de desambiguación que han sido utilizadas mundialmente en otras disciplinas; (iii) minimizar la ambigüedad semántica de tipo polisémica en la obtención de los elementos básicos del diagrama de objetivos de KAOS; (iv) automatizar el proceso que permite obtener los elementos básicos del diagrama de objetivos de KAOS; (v) diseñar implementar y validar el sistema que se denominó “NEONPL”, el cual permitió desarrollar la fase experimental del proyecto. Asimismo, se realizó un análisis a las técnicas de desambiguación (UKB, MFS y Lesk) utilizadas en el modelo propuesto y se pudo establecer que con la técnica UKB se obtienen mejores resultados en precisión y en recuerdo.

Con el aporte realizado en este trabajo, se da un paso importante en el proceso utilizado en la fase de definición del ciclo de vida del software a través de la metodología KAOS, puesto que, los elementos básicos del diagrama de objetivos de KAOS se obtienen de manera automática y utilizando un proceso de desambiguación semántica. Además, por ser automático permite: (i) que el analista pueda dedicar más tiempo a otros procesos propios de su especialidad; y (ii) acercar el proceso al discurso del interesado.

AGRADECIMIENTOS

Este artículo se realizó en el marco del proyecto de investigación: “*Modelo de procesamiento terminológico basado en ontologías para la desambiguación verbal en la educción de requisitos de software*” código 18717, financiado a través de la “Convocatoria del programa nacional de proyectos para el fortalecimiento de la investigación, la creación y la innovación en posgrados de la Universidad Nacional de Colombia 2013-2015”.

REFERENCIAS

Agirre, E., y Soroa, A., *Personalizing pagerank for word sense disambiguation*. Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, 33–41, Athens, Greece (2009).

Alves, V., Niu, N., Alves, C., Valença, G., *Requirements engineering for software product lines: A systematic literature review*. Information and Software Technology, Elsevier, 806–820 (2010).

Banerjee, S., y Pedersen, T., *An adapted Lesk algorithm for word sense disambiguation using WordNet. Computational Linguistics and Intelligent Text Processing*, Springer Berlin Heidelberg, 136–145 (2002).

Barachisio, L., Cardoso V., Lucredio, D., Santana, A., Romero S., Pontin R., *A systematic review of domain analysis tools*. En: Information and Software Technology, Elsevier, 1–13 (2010).

Casagrande, E., Woldeamlak, S., Lee W., Zeineldin, H., Svetinovic, D., *NLP-KAOS for Systems Goal Elicitation: Smart Metering System Case Study*. Software Engineering, IEEE Transactions, 40 (10) 941-956 (2014).

González, A., Laparra, E., y Rigau, G., *Multilingual Central Repository version 3.0*. Proceedings of the 8th International Conference on Language Resources and Evaluation, ELRA, 2525–2529 (2012).

Guzmán, J., Lezcano, L., y Gómez, S., *Caracterización de los Elementos del Diagrama de Objetivos de KAOS a partir de Lenguaje Natural*. Revista Colombiana de Tecnologías de Avanzada, 138 - 144 (2013).

Ibrahim, M., y Ahmad, R., *Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques*. Actas de la 2ª Conferencia en Investigación y Desarrollo en Computación, 200–204 (2010).

Jurafsky, D., y Martin, J., *Speech and Language Processing*, Prentice Hall, New Jersey (2000).

Lamsweerde, A. Van. *Goal-Oriented Requirements Engineering : A Guided Tour*. Actas del 5° Simposio Internacional IEEE, 249-262 (2001).

Lamsweerde, A. Van., *Requirements Engineering - From system goals to UML models to software specifications*, Wiley, West Sussex, Inglaterra (2009).

Letier, E., *Reasoning about Agents in Goal-Oriented Requirements Engineering*, Tesis de Magíster, Departamento de Ingeniería Informática, Universidad Católica de Lovaina, Lovaina, Bélgia (2001).

Lezcano, L., Guzmán, J., Tamayo, P., *The application of and unresolved problems regarding the use of objectives in software engineering*. Ingeniería E Investigación, 63 – 67 (2012).

Lezcano, L., *Un modelo de procesamiento terminológico para la obtención y validación de requisitos de software basado en el diagrama de objetivos de KAOS*. Tesis Doctoral. Universidad Nacional de Colombia. Departamento Ciencias de la Computación y de la Decisión, Medellín, Colombia (2014).

Lezcano, L., Guzmán, J., y Gómez, S., *Obtaining Agents and Entities from Natural Language*, Lecture Notes, Electrical Engineering: Springer Verlag Berlin Heidelberg, 29-36 (2015).

Navigli, R., *Word sense disambiguation: A survey*, ACM Computing Surveys (CSUR), 41(2), 10 (2009).

Nicolás, J., Toval, A., *On the generation of requirements specifications from software engineering models: A systematic literature review.* En: Information and Software Technology, Elsevier, 1291-1307 (2009).

Preiss, J., Dehdari, J., King, J., y Mehay, D., *Refining the most frequent sense baseline.* Proceedings of the NAACL HLT Workshop on Semantic Evaluations: Recent Achievements and Future Directions, Association for Computational Linguistics, 10–18, Boulder, Colorado (2009).

Puello, P., Oviedo, S., Franco, D., *Metodología para el Diagnóstico de Prácticas del Modelo Proceso Personal de Software*. Información Tecnológica, 25(2), 57-66 (2014)

Respect IT, *A KAOS Tutorial. Objectiver*, 1–46 (2007).

Yu, E., *Modelling Strategic Relationships for Process Reengineering*. PhD Thesis. University of Toronto. Department of Computer Science (1995).