Uso de Técnicas de Aprendizaje para la Validación de Especificaciones

Jordi Alvarez, Núria Castell
Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics
Pau Gargallo 5, 08028 Barcelona
{jalvarez, castell}@lsi.upc.es

Resumen

En este artículo abordamos el proceso de validación de especificaciones preliminares escritas en lenguaje natural desde una nueva perspectiva: el uso de técnicas de aprendizaje automático. Describimos un sistema que a partir de ejemplos de especificaciones correctas construye o refina un modelo que nos permitirá validar nuevas especificaciones.

1 Introducción

La fase de ingeniería de requerimientos es una de las más importantes dentro del proceso de desarrollo del software. Se ha comprobado en diversos estudios que la mayoría de los errores detectados en la fase de codificación y mantenimiento provienen de una mala especificación [1, 7]. Además, el coste de corregir un error de especificación se incrementa desmesuradamente cuando se detecta en las fases de codificación o mantenimiento [14]. Esta fue una de las principales motivaciones por las que surgió el proyecto LESD [4]. Éste define cinco factores que nos permitirán validar la calidad de las especificaciones: trazabilidad, completitud, consistencia, verificabilidad y modificabilidad. LESD es continuado por el proyecto CICYT TIC93-0420 en el cual se enmarca este trabajo.

En este artículo intentamos dar una nueva perspectiva a la validación de especificaciones preliminares escritas en lenguaje natural. Tanto el hecho de que se trate de especificaciones preliminares como el que éstas estén escritas en lenguaje natural hace que sea realmente difícil dar una definición clara sobre los criterios que hacen que una especificación sea válida o no. Por ello proponemos usar técnicas de aprendizaje automático para adquirir, a partir de ejemplos correctos de especificaciones, un modelo de especificación válida.

Para la representación de las especificaciones usamos una red semántica. Por un lado, un mecanismo que permita representar fácilmente estructuras jerárquicas es adecuado, pues la mayoría de problemas que se nos presentan en ingeniería del software son de este tipo [11]. Por otro lado, las redes semánticas, con la ayuda de los tres mecanismos básicos de representación de [2] (agregación, clasificación y especialización) constituyen quizás el mecanismo de representación de conocimiento más adecuado para este tipo de problemas [6].

De este modo, a partir de especificaciones válidas, que podemos ver como compuestas por una serie de descripciones de objetos, abstraeremos una serie de conceptos basándonos en esas descripciones. Clasificaremos cada uno de estos objetos como instancia de uno de los conceptos creados, y a partir de ellos aprenderemos una serie de propiedades que podemos interpretar como la descripción genérica de cada concepto.

Posteriormente, para validar una especificación nos bastará con clasificar los objetos cuya descripción constituye la especificación y comprobar si cumplen las propiedades correspondientes al concepto bajo el cual han sido clasificados.

En resumen, lo que pretendemos es generar o enriquecer una base de conocimiento a partir de especificaciones ya validadas; y posteriormente usar esta base de conocimiento para validar (y posiblemente también ayudar en la corrección de) otras especificaciones.

En la figura 1 podemos ver un esquema general del proceso. El aprendizaje puede ser para crear un modelo a partir de cero o bien para enriquecer un modelo que ya tuvieramos previamente. En cualquier caso se trata de un proceso automatizado en el que el ingeniero de software sólo se encarga de proporcionar los ejemplos. En cambio la validación no es un proceso totalmente automático: una vez detectados los posibles errores, se genera un informe que pasa por el ingeniero de software, quien decide si se trata realmente de errores y cómo solucionarlos. Finalmente, una vez la especificación está validada, puede ser utilizada por el mecanismo de aprendizaje para refinar el modelo conceptual que ya teníamos.

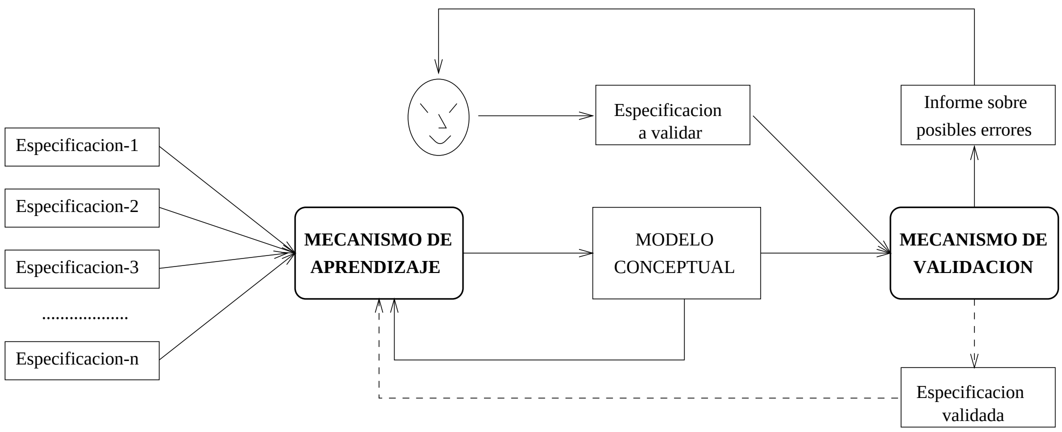

Figura 1: Esquema general del proceso de aprendizaje y validación

En este artículo nos ocupamos únicamente de los procesos de aprendizaje y validación una vez ya tenemos la representación conceptual de las especificaciones en nuestra red semántica. El paso de las especificaciones tal como las recibimos en lenguaje natural a dicha representación conceptual se describe en [19]; mientras que en [3] podemos encontrar una descripción global del proceso de control de especificaciones en el cual está inmerso el trabajo descrito aquí.

En lo que sigue, en la sección 2 se comenta la importancia de las estructuras, el elemento básico a partir del cual realizamos el aprendizaje. En la sección 3 distinguimos los diferentes niveles de conocimiento que tenemos en una especificación de requerimientos y sobre los cuales podemos aprender. En la sección 4 profundizamos un poco en los mecanismos de aprendizaje utilizados, comentando los distintos tipos de concepto que utilizamos para construir una jerarquía y en cómo ésta es generada. En cuanto al aprendizaje en sí mismo, la sección 5 describe el aprendizaje de las propiedades que cumplen las instancias de los conceptos aprendidos; cosa que nos servirá para, tal como se cuenta en la sección 6, validar nuevas especificaciones. Finalmente, la sección 7 describe las condiciones que deben cumplir las especificaciones de ejemplo que proporcionamos al sistema para que el aprendizaje sea el adecuado.

2 Aprendizaje a Partir de Estructuras

Cada especificación está formada por una serie de descripciones de objetos cada una de las cuales define una cierta interrelación entre ellos. Entendemos como objeto cualquier entidad, física o no, que está repersentada por un nodo en la red semántica. Así por ejemplo: actividades, restricciones, condiciones, eventos, etc. serán también objetos.

Cada una de estas descripciones está constituida por una agregación de descripciones más sencillas que comunmente se llaman atributos o slots y que aquí, para lo que nos interesa, trataremos siempre como relaciones entre nodos. Así por ejemplo, en la figura 2, la agregación de las relaciones TITULO, ISBN, AÑO-PUBLICACIÓN y AUTOR entre LIBRO-1 y los nodos correspondientes de la red semántica constituye la descripción de LIBRO-1.

Es esta agregación, uno de los tres mecanismos básicos que hemos mencionado en la sección anterior, lo que nos permite construir una representación de la estructura presente en la especificación.

Definiendo una estructura como un conjunto de nodos relacionados entre sí, en la figura 2 podemos ver una estructura bastante compleja (sin contar la relación de instanciación tenemos ocho nodos involucrados). Ésta se puede descomponer entre la descripción de LIBRO-1 y la de Quim Monzó; y también se puede descomponer en estructuras lineales más simples (parte derecha de la figura) que son las que utilizaremos en el proceso de aprendizaje.

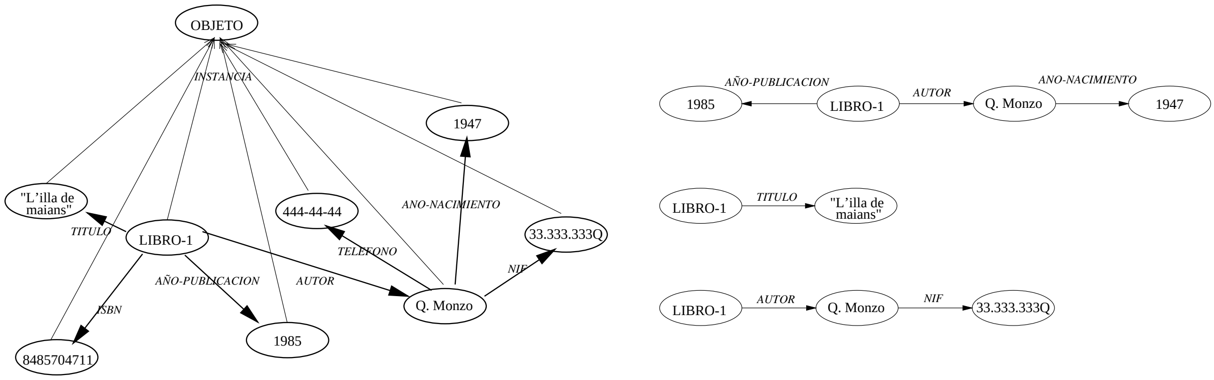

Figura 2: Ejemplo de estructuras básicas a partir de la definición de la representación de un libro y su autor en la red semántica.

Así como [18] hace uso de los tres mecanismos antes mencionados para calcular la similitud entre diferentes estructuras, nosotros sólo usamos directamente el mecanismo de agregación. Aunque el uso que hacemos de él nos permite, como veremos más adelante, sacar provecho de la información que nos puedan proporcionar los mecanismos de clasificación y generalización.

El aprendizaje consistirá en capturar en forma de concepto aquellas estructuras o grupos de estructuras que a priori pensamos que pueden sernos más útiles. De este modo, la extensión de un concepto estará constituida por aquellos objetos que cumplan determinada propiedad estructural.

Una vez tenemos los conceptos, aprendemos a partir de los ejemplos aquellas propiedades (también estructurales) que cumplen sus instancias[1].

Así pues, cada uno de los conceptos llevarán asociadas una serie de propiedades estructurales de las cuales distinguiremos entre aquellas que definen el concepto y aquellas que están asociadas a él.

Las primeras establecen una doble implicación entre las instancias de un concepto y éste. Es decir, el hecho de que un objeto esté relacionado de tal forma que cumpla la o las propiedades definitorias de un concepto, es una condición necesaria y a la vez suficiente para que dicho objeto sea instancia de ese concepto. Esto nos será muy útil para que posteriormente podamos clasificar automáticamente todos los objetos en la jerarquía aprendida.

Las segundas representan propiedades estructurales que las instancias de un concepto acostumbran a cumplir, aunque no son ni una condición necesaria ni suficiente para que un objeto sea instancia de un concepto. Estas propiedades nos servirán para detectar relaciones entre diferentes propiedades estructurales y por lo tanto nos serán útiles para comprobar la completitud y la consistencia de las especificaciones.

En la figura 3 podemos observar las propiedades definitorias y asociadas de cuatro conceptos formados a partir de estructuras simples extraidas de la descripción de LIBRO-1.

[1]Evidentemente, las propiedades que definen el concepto serán un subconjunto del total de propiedades que cumplen sus instancias

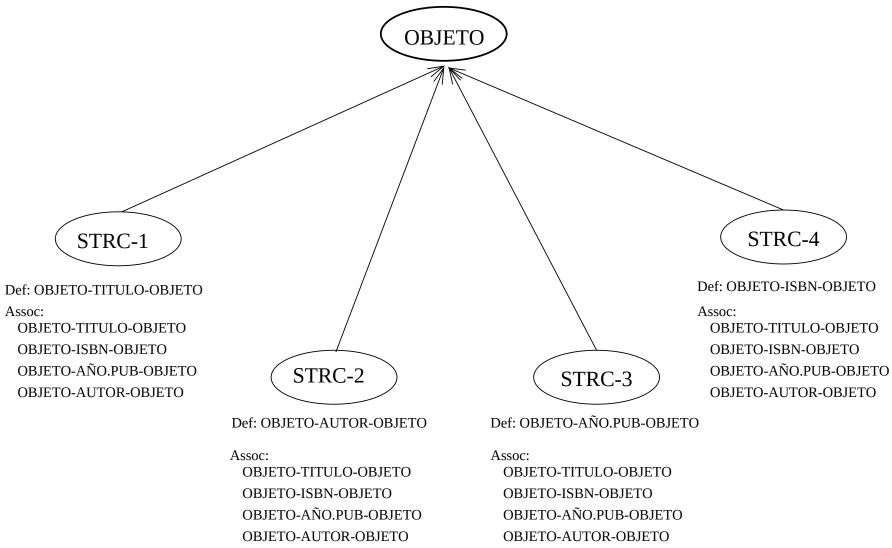

Figura 3: Propiedades definitorias de los conceptos generados a partir de la representación de LIBRO-1 de la figura 2.

Las propiedades definitorias están implementadas simplemente como una lista de nodos y relaciones intercalados que el mecanismo de clasificación de la red semántica (no confundir con el concepto de clasificación concepto-instancia) comprueba cada vez que clasifica un objeto en la jerarquía.

Las propiedades asociadas están implementadas como relaciones fuzzy en la descripción del concepto. Se trata de relaciones entre nodos, como las que aparecen en cualquier red semántica, pero con un peso asociado. A la manera de las conexiones que nos encontramos en las redes neuronales. Esto nos permite guardar el grado en que un concepto cumple una propiedad. Aunque parezca que esto limita las propiedades asociadas que podemos guardar a simples conexiones entre dos nodos, la iteración del proceso de aprendizaje (como veremos más adelante) nos permitirá acceder a propiedades cada vez más complejas.

3 Conocimiento y Metaconocimiento

Hasta ahora hemos hablado de agrupar en conceptos los objetos cuyas descripciones componen las especificaciones; pero no hemos hablado del tipo de conocimiento que el sistema debe aprender. Típicamente, una especificación contiene dos niveles de información diferentes.

Por un lado, información sobre objetos tangibles (aquello que tradicionalmente en las redes semánticas se ha dado en llamar token y que en [5] constituye el nivel de instancias), que describe la realidad en sí. Así por ejemplo, en una especificación de un sistema de gestión de una biblioteca, cuando hablamos de “la biblioteca de la Facultad de Informática de Barcelona”, estamos hablando de una entidad presente en la realidad del sistema especificado.

Y por otro lado nos encontramos con descripciones de conceptos del dominio (nivel de dominio en [5]). Siguiendo con el mismo ejemplo, en la especificación de un sistema de gestión de una biblioteca nos encontraremos seguramente con una descripción (adecuada a nuestro problema, eso sí) de librería, libro, etc.

A nivel de representación, el mecanismo de clasificación nos permite distinguir entre estos dos niveles de conocimiento y nos permite ir todavía más allá. Si nuestro sistema de representación es suficientemente flexible para tratar los conceptos como simples objetos, puede hacer que cualquier concepto, como objeto que es, sea a la vez instancia de otro concepto. Esto es utilizado por sistemas como TELOS [13] o KAOS [5, 20] para colocar un nivel de metaconocimiento por encima del nivel de los conceptos pertenecientes al dominio de aplicación.

La información de este metanivel constituye un metamodelo que permite imponer unas ciertas restricciones sobre el nivel de conocimiento básico de la misma forma que éste hace con el nivel de los tokens. Este metanivel constituye pues una herramienta potente para encontrar incompletitudes e inconsistencias en el nivel de descripción del dominio (el modelo).

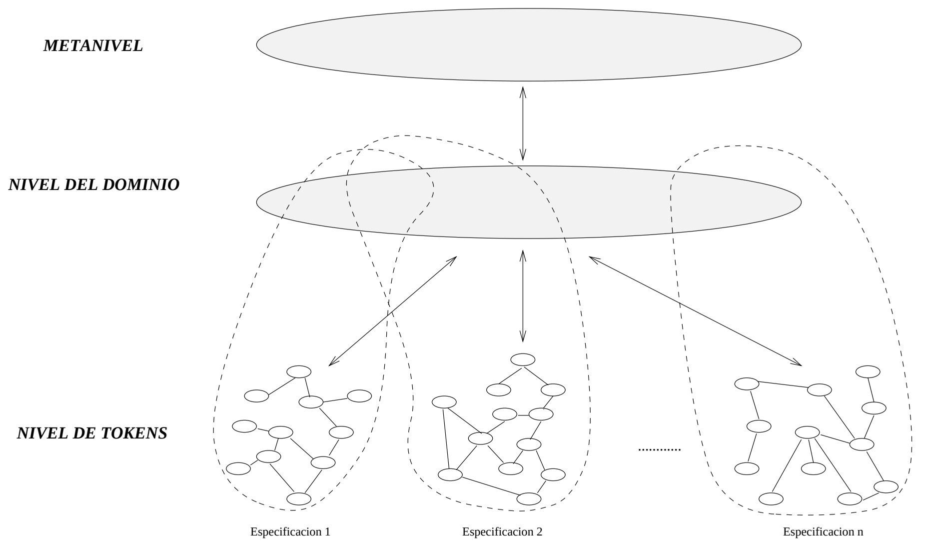

Figura 4: Flujo de información entre los tres niveles de conocimiento. El aprendizaje nos permite abstraer propiedades que cumplen los objetos de un nivel a partir de objetos del nivel inferior; mientras que el mecanismo de clasificación nos permite tener un modelo de un nivel inferior a partir de su inmediato superior.

A nivel de aprendizaje, debemos ser capaces de aprender tanto sobre los tokens como sobre el modelo. Para ello, seguiremos la misma filosofía que los sistemas de representación mencionados. Tendremos tres niveles: el de los tokens, el conocimiento del dominio y el metanivel[2].

Así podremos separar el proceso global de aprendizaje en dos niveles distintos: el aprendizaje sobre el conocimiento del dominio (a partir del nivel de tokens) y el aprendizaje del metamodelo (realizado a partir del conocimiento del dominio). Según nos interese, podemos aplicar el primero, con lo que completaremos y aprenderemos las propiedades de la jerarquía de conceptos del dominio; aplicar el segundo, con lo que generaremos un metamodelo (o completaremos el que ya teníamos); o bien aplicar los dos. De esta forma, tal como se muestra en la figura 4, se produce un flujo de información bidireccional (por un lado el aprendizaje y por otro un supuesto sistema de tipos regido por el mecanismo de clasificación) entre el nivel de las instancias y el nivel del dominio [8]; y entre este último y el metanivel.

4 Generación de Conceptos

La base sobre la cual vamos a construir nuestra jerarquía (bien sea la correspondiente al modelo o la del metamodelo) son las propiedades estructurales de los objetos. Para ello, los conceptos que crearemos se definirán a partir de cadenas estructurales conceptualizadas; entendiendo por cadena estructural una serie de nodos ordenados y unidos cada uno con el siguiente (excepto el

[2] Aunque en [13] se menciona la posibilidad de tener más de tres niveles de conocimiento, la realidad es que los verdaderamente útiles (al menos para el caso que nos ocupa) son los tres primeros, pues de lo que se trata es de modelizar la información que aparece en la especificación. Como hemos visto, dicha información pertenece a los dos primeros niveles: el de tokens y el de conceptos básicos. La modelización de un nivel viene dada por el nivel inmediatamente superior, que es el único que puede imponer condiciones sobre él. Por eso, con añadir un tercer nivel (que no necesitamos modelizar sino simplemente utilizar) nos basta.

último) mediante una relación; y entendiendo por conceptualizadas el hecho de que no aparezca en la cadena el objeto que aparece en la especificación inmerso en la estructura, sino que el que aparezca sea el concepto del cual es instancia. En la figura 5 podemos ver las estructuras simples de la figura 2 conceptualizadas.

El hecho de hacer el paso de la conceptualización viene motivado porque el conocimiento que queremos aprender (recordemos que queremos aprender un modelo para una serie de objetos) está a nivel de conceptos y no a nivel de instancias. Por lo tanto sería una pérdida de tiempo intentar razonar sobre las instancias cuando aportan el mismo conocimiento que los conceptos correspondientes. Son los conceptos quienes dan un significado a sus instancias, que por sí mismas no lo tienen[3].

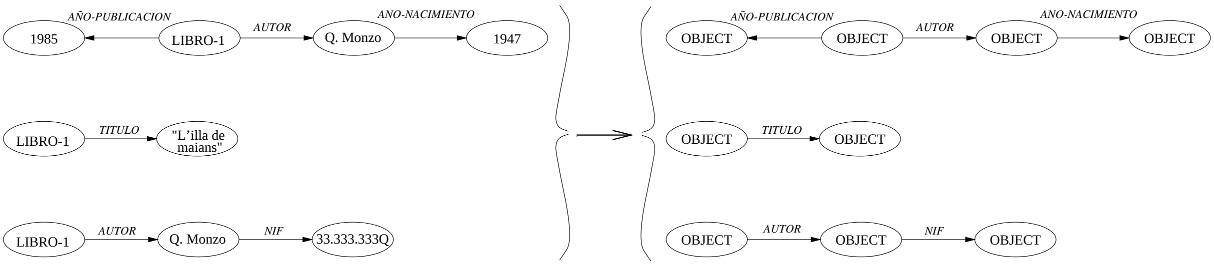

Figura 5: Conceptualización de las estructuras de la figura 2 teniendo en cuenta que todos los objetos son instancia de objeto.

La conceptualización de las estructuras es lo que nos permite sacar provecho de los mecanismos de clasificación y generalización. Respecto al primero, su uso en la conceptualización de estructuras es evidente al cambiar los objetos por los conceptos de los que son instancia. En cuanto al segundo, la posterior clasificación en la jerarquía del concepto generado hace que obtengamos estructuras más generales que otras simplemente porque los conceptos y relaciones que participan en unas son más generales que los que participan en las otras.

Así por ejemplo, en la figura 6 podemos ver como los conceptos STRC-1 y STRC-3 son más generales que STRC-2 y STRC-4 respectivamente porque repositorio es más general que biblioteca.

4.1 Tipos de Conceptos

Para construir una jerarquía a partir de cadenas estructurales tenemos tres tipos de conceptos distintos: conceptos estructurales simples, conceptos conjuntivos y conceptos disyuntivos. Los primeros vienen definidos por una única propiedad estructural que es una cadena estructural conceptualizada. Los conceptos conjuntivos, como su nombre indica, vienen definidos por la conjunción de las propiedades estructurales que definen a los padres. Y por último, los conceptos disyuntivos se definen a partir de la disyunción de las propiedades estructurales que definen a sus hijos.

Los conceptos estructurales simples constituyen las piezas básicas sobre las que los conceptos conjuntivos y disyuntivos nos permitirán hacer especializaciones y generalizaciones. Con estos tres tipos de conceptos combinados adecuadamente podremos obtener fácilmente conceptos con definiciones estructurales complejas. En la figura 7 podemos ver cómo sirviéndonos de conceptos conjuntivos y estructurales simples construimos una jerarquía en la que especificamos las descripciones mínimas de distintos tipos de publicaciones.

[3]Al principio del proceso de aprendizaje, si partimos de cero, la jerarquía tendrá un único concepto al que podemos dar el nombre genérico de object (o meta-object, según el caso). Eso significa que al principio medimos a todos los objetos por el mismo rasero, y object aparecerá como único nodo en todas las posiciones de todas las cadenas estructurales. Esta situación está totalmente acorde con el hecho de que si en un principio no tenemos ninguna jerarquía, lo único que tenemos para aprender es la estructura.

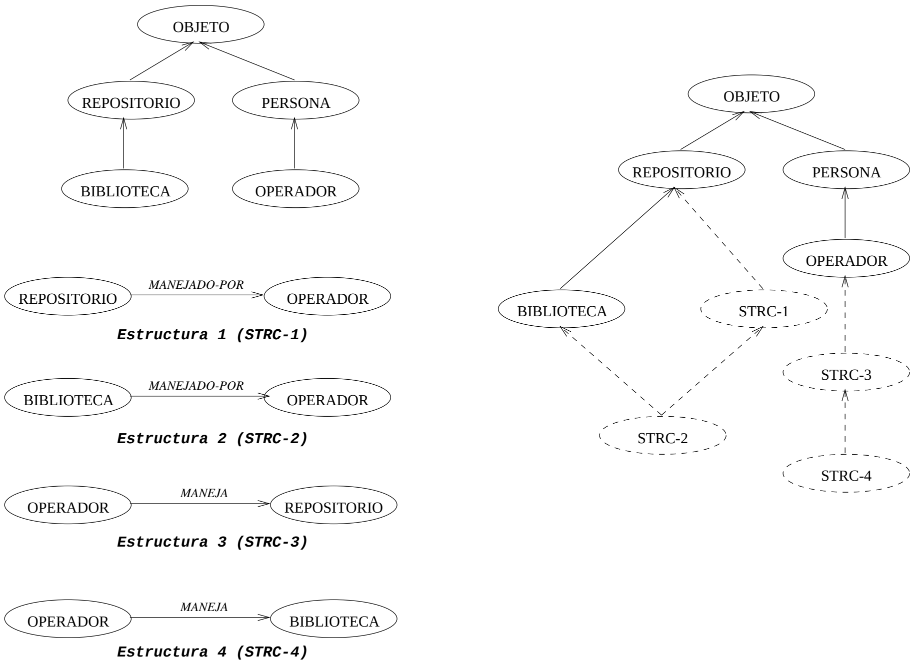

Figura 6: Clasificación de estructuras según los conceptos que participan en ellas. A la izquierda tenemos una simplificación de la jerarquía que tenemos inicialmente y las estructuras que vamos a introducir. A la derecha podemos ver como queda la jerarquía después de introducir los nuevos conceptos y clasificarlos automáticamente.

4.2 Analogía con el Aprendizaje Inductivo

Mirando la generación de conceptos desde el punto de vista del aprendizaje inductivo de Michalski y sus reglas de generalización [12], podemos hacer una analogía entre los conceptos generados y las fórmulas lógicas. De este modo, podemos ver los conceptos estructurales simples como átomos, la conceptualización de estructuras como el paso de constantes a variables, la relación entre los conceptos conjuntivos y sus padres como la aplicación de la regla de eliminación de conjuntandos y la relación entre los conceptos disyuntivos y sus padres como la aplicación de la regla de adición de disyuntandos.

Desde este punto de vista, cada uno de los conceptos es en realidad una función de reconocimiento potencial de las propiedades que queremos aprender.

No nos debe importar el que algunas reglas de generalización de las usadas no preserven la equivalencia lógica entre una fórmula y su generalización[4] [10]. El aprendizaje real de las propiedades se realiza sobre los conceptos generados en una fase posterior; y puede revelarnos algunos de estos conceptos como totalmente inútiles.

Lo que sí nos debe preocupar es que hayamos generado como mínimo todos aquellos conceptos que representen verdaderamente funciones de reconocimiento sobre las propiedades que queremos aprender.

Como veremos en el siguiente apartado, el mecanismo de aprendizaje de las propiedades requiere que el nodo relativo al concepto se encuentre explícitamente representado en la red semántica. De no ser así, el aprendizaje se realizará sobre conceptos representativos de funciones de reconocimiento que por muy parecidas que sean a la verdadera, caerán en la incompletitud o inconsistencia.

La única manera que tenemos de asegurar que todos los conceptos que representen funciones de reconocimiento que nos interesan sean generados es creando todos los conceptos posibles. Pero esto es computacionalmente imposible dado el elevado número de estructuras distintas con que

[4] De las tres que hemos mencionado, la única que no la preserva es la de paso de constantes a variables.

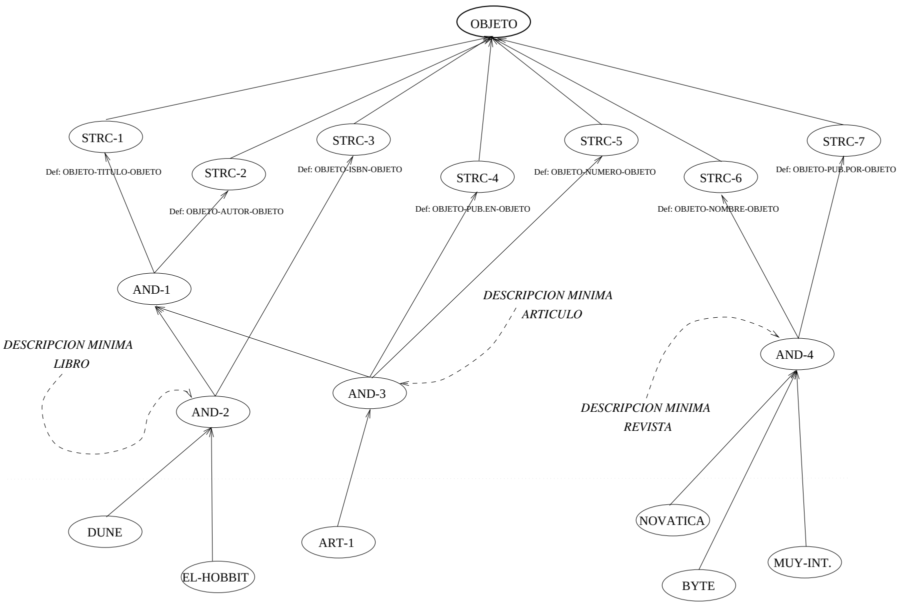

Figura 7: Distinción de distintos tipos de publicación mediante el uso de propiedades estructurales definitorias.

nos podemos encontrar. Por ello, no nos queda más remedio que utilizar heurísticas que nos guien en la generación de conceptos.

4.3 Modelo Heurístico de Generación de Conceptos

Este es un punto que todavía tenemos que explorar en profundidad; si bien hasta este momento hemos utilizado un sistema bastante sencillo que funciona aceptablemente para conocimiento generado aleatoriamente a partir de un metamodelo previo que es destruido justo antes de empezar el aprendizaje (parecido al utilizado en KAOS [5]).

El proceso consiste en generar tantas estructuras simples como aparezcan en las descripciones de ejemplos que tengamos. Para no disparar la complejidad de los cálculos las estructuras generadas no deben sobrepasar una longitud máxima preestablecida. De todas maneras, si con estructuras de una determinada longitud no tuviéramos suficiente, la iteración del proceso nos permitiría, tal como veremos más adelante, acceder a estructuras más largas de forma automática.

Una vez tenemos las estructuras simples usamos conceptos conjuntivos para generar una estructura compleja para cada descripción de objeto que tengamos y clasificamos cada objeto como instancia del concepto correspondiente. La herencia múltiple la podemos realizar siempre a nivel conceptual; cosa que permite al mecanismo de aprendizaje abstraer propiedades relacionadas con la conjunción de varios conceptos simples).

Para terminar de construir la jerarquía usamos técnicas de clustering [9] sobre las estructuras básicas generadas. La distancia de un concepto a un cluster se calcula como la máxima de las distancias entre él y todos los elementos del cluster. Y la distancia entre distintos conceptos se calcula a partir de la descripción de un concepto, que nos viene dada por el conjunto de propiedades asociadas a él. Por lo tanto, es necesario haber realizado el aprendizaje de propiedades, explicado en la próxima sección, sobre los conceptos a los que se aplica el clustering.

De forma intuitiva, la distancia (y también la similitud) entre las diferentes descripciones de conceptos se calcula a partir de la información común (propiedades asociadas iguales) que tienen, penalizando la ausencia de información. Esta ausencia de información se mide a partir del total de tipos de propiedades asociadas que puede tener un concepto. Su penalización hace que la supuesta distancia entre dos elementos no sea tal distancia sino simplemente un heurístico; ya que una de las propiedades que debe cumplir es que la distancia entre un elemento y él mismo sea siempre 0 [10]. En nuestro caso sólo será 0 si su descripción contiene todos los tipos de propiedades asociadas existentes.

De todas maneras, esta penalización nos proporciona una medida heurística que se muestra como la mejor de las que hemos probado. Además, fuerza a construir una jerarquía de forma que los objetos con mucha información (ya sean conceptos o instancias de ellos) estarán situados en la parte profunda de la jerarquía mientras que los que tienen poca información estarán situados más arriba.

No generamos un único nivel de clusters sino que empezamos con un umbral bajo de distancia (podemos empezar con 0, por ejemplo) y vamos iterando el proceso mientras incrementamos este umbral hasta llegar a 1 (la máxima distancia entre dos conceptos). De esta forma, acabaremos generando un único cluster en la parte más alta de la jerarquía que se corresponderá con la raíz de ésta, con lo cual la habremos completado.

5 Aprendizaje de las Propiedades

Una vez ya tenemos los nodos explícitamente representados en la red semántica, procedemos con el aprendizaje de las propiedades estructurales asociadas a cada concepto. Esta fase se puede realizar cuando los conceptos están ya todos generados o bien incrementalmente a medida que se van generando (de hecho, las técnicas de clustering del apartado anterior requieren que en los conceptos sobre los que se aplica el clustering ya se haya hecho la abstracción de las propiedades asociadas).

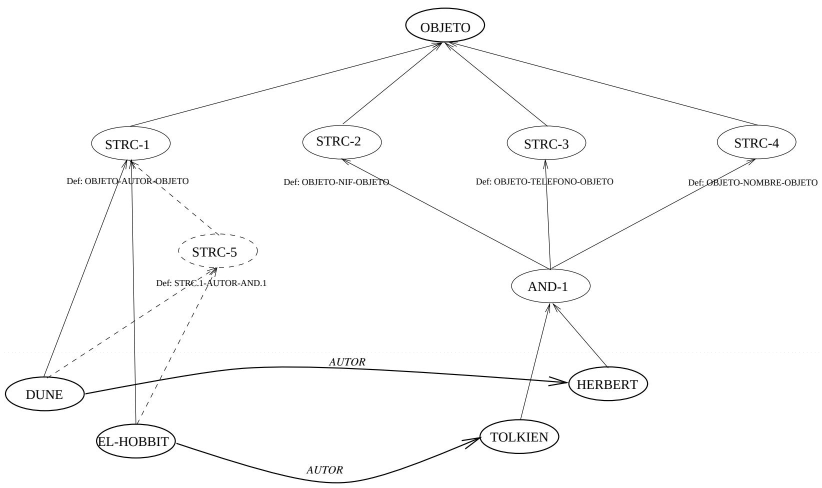

Figura 8: Iteración del proceso de generación de conceptos para refinar la base de conocimiento.

Como se puede observar en la figura 8, las propiedades (tanto definitorias como asociadas), por complejas que sean, se pueden reducir a propiedades asociadas más simples entre conceptos más complejos. Por ello podemos reducir la representación de tales propiedades a la representación de simples links o relaciones (con un peso que indica el grado en que la propiedad se cumple). Y por tanto, podemos reducir también el aprendizaje al aprendizaje de simples links.

En el dibujo se puede observar como tras la primera iteración del proceso (trazado de linea continua) se genera la propiedad definitoria OBJETO-AUTOR-OBJETO para el concepto STRC-1. En la segunda iteración (trazado punteado) generamos un nuevo concepto que especializa la propiedad definitoria de STRC-1 a STRC-1-AUTOR-AND-1. Y como AND-1 es un concepto definido a partir de propiedades estructurales, la propiedad definitoria de STRC-5 equivale a la conjunción de: STRC-1-AUTOR-OBJETO-NIF-OBJETO, STRC-1-AUTOR-OBJETO-TELEFONO-OBJETO y STRC-1-AUTOR-OBJETO-NOMBRE-OBJETO. Lo mismo ocurre para las propiedades asociadas ya que a base de iterar el proceso de aprendizaje, aparecen en ellas cada vez nodos con definiciones estructurales más complejas.

En definitiva, el aprendizaje consiste pues en decidir en qué grado los conceptos generados están en relación con otros conceptos. Para ello, inspirándonos libremente en el modelo propuesto por Shastri en [17], damos a cada link entre dos conceptos el valor correspondiente al porcentaje de instancias del primer concepto que están relacionadas mediante este tipo de link a instancias del segundo concepto. De esta forma, suponiendo que los nodos de nuestra red semántica se activaran y que cuando se activara un objeto se activaran todos los conceptos del cual es instancia, el peso de un link es una medida de lo a menudo que el nodo destino se activa cuando el nodo origen está activado; lo cual se ajusta extremadamente bien a una interpretación Hebbiana de los pesos sinápticos en las redes neuronales [17]. Pero a diferencia de Shastri, nuestros links están etiquetados (sus etiquetas son las relaciones); por lo que podemos tener varios links diferentes entre los mismos dos conceptos.

6 Validación de Especificaciones

Una vez tenemos realizado el proceso de aprendizaje, podemos empezar a validar especificaciones. La validación es sencilla. Si interpretamos la especificación que vamos a validar como un conjunto de descripciones de los objetos que intervienen en el sistema especificado, sólo necesitamos clasificar en la jerarquía aprendida cada uno de estos objetos y posteriormente comparar sus propiedades con las del concepto bajo el cual ha sido clasificado.

Las propiedades definitorias de los conceptos nos servirán para clasificar los objetos, mientras que las propiedades asociadas nos servirán para verificar que su descripción sea completa y consistente.

Si un concepto tiene alguna propiedad estructural asociada que alguna de sus instancias no tiene, habremos detectado una posible incompletitud[5]. La razón es sencilla: si un grupo de objetos que tienen en común parte de su estructura (las propiedades estructurales definitorias del concepto que los agrupa) tienen todos ellos (o la mayoría) cierta propiedad y nos encontramos con otro objeto que tiene en común toda la estructura menos esta última propiedad, sospecharemos la posibilidad de que la descripción de este objeto sea incompleta. La evidencia que tengamos para ello dependerá del porcentaje de instancias que cumplan la propiedad.

Recíprocamente, si un objeto tiene alguna propiedad que el concepto bajo el cual ha sido clasificado no tiene, sospecharemos la existencia de una posible inconsistencia o bien que la descripción del objeto contiene información redundante.

Pero no todo termina ahí. Hay otro caso en el cual podemos detectar una posible incompletitud. ¿Qué pasa si la descripción de un objeto es tan pobre (y tan incompleta) que hace que la clasificación se quede a medias? Al estar clasificado bajo un antecesor del concepto bajo el cual tendría que estar clasificado, es muy posible que no detectemos buena parte de las incompletitudes.

Una solución directa a este problema es hacer un proceso iterativo: cada vez que vamos completando la descripción del objeto y eliminando incompletitudes, reclasificamos el objeto, con lo que probablemente encontremos nuevas incompletitudes. Así hasta que ya no encontramos más. Este proceso nos servirá para la mayor parte de los casos, pero no podemos asegurar que nos sirva siempre.

Para asegurarnos, existe una solución complementaria. En el caso de que nuestro objeto haya sido clasificado bajo una concepto que no tiene instancias directas (sino que sus instancias son instancias de algún concepto más específico)[6], podemos deducir que su descripción es tan incompleta que no permite una clasificación total del objeto. De esta forma, si el -llamémosle- concepto abstracto no permite iniciar el proceso iterativo que acabamos de comentar, podemos intentar guiarnos por las propiedades definitorias de los conceptos hijos. Es decir; intuimos que el objeto no puede ser instancia directa del concepto supuestamente abstracto, por lo que debe ser instancia (directa o no, da igual) de alguno de sus hijos. Por lo tanto, intentamos averiguar

[5] En el próximo apartado ahondaremos en el porqué del término posible.

[6] Aquello que en terminología de lenguajes orientados a objetos se llamaría una clase abstracta.

qué propiedad o propiedades definitorias de los hijos debería cumplir el objeto y evidentemente no cumple (si no, la clasificación hubiera continuado).

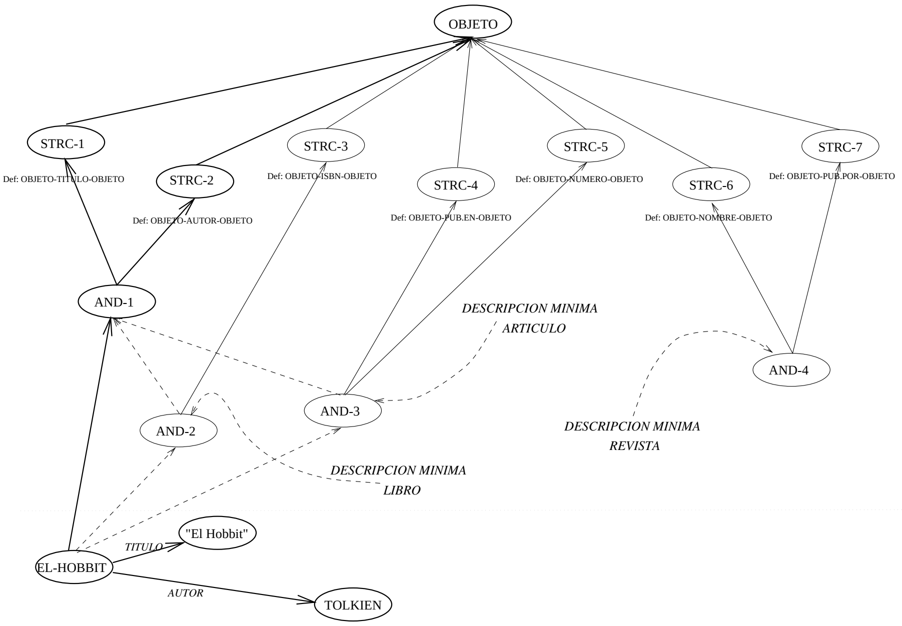

Figura 9: Validación de una descripción de libro a la cual le falta el atributo ISBN.

En la figura 9 podemos ver como una descripción incompleta de un libro se clasifica bajo AND-1. Al no tener éste ninguna instancia directa, el sistema sospecharía que la descripción es incompleta y propondría al usuario completar la descripción de EL-HOBBIT con propiedades correspondientes a AND-2 y AND-3 (según lo que conoce el sistema, la descripción del objeto puede evolucionar hacia un libro o hacia una revista).

Mientras en los otros casos de detección de posibles incompletitudes e inconsistencias estaba claro cuales eran las propiedades estructurales en cuestión, aquí tan sólo sabemos que se trata de un subconjunto de las propiedades estructurales que diferencian a los hijos del concepto abstracto de éste. Es el ingeniero de software quien debe decidir cuales son las propiedades que faltan (si es que realmente falta alguna).

7 Conjunto de Entrenamiento

En cualquier método de aprendizaje a partir de ejemplos, la elección del conjunto de ejemplos a partir de los cuales vamos a realizar el entrenamiento (conjunto de entrenamiento) es de vital importancia. En nuestro caso, el conjunto de entrenamiento está formado por especificaciones. Después de hablar de la validación en el apartado anterior, analizaremos aquí qué propiedades deben cumplir las especificaciones del conjunto de entrenamiento para que el aprendizaje y la posterior validación sean eficaces.

Y para que el aprendizaje sea eficaz, todas en conjunto y cada una en particular deberán cumplir algunas propiedades que comentamos a continuación.

7.1 Representatividad

En primer lugar, las especificaciones deben ser ampliamente representativas dentro del dominio que estamos tratando. Si todas ellas se restringen a un subconjunto del dominio, tendremos una parte del dominio sobre la que no podremos aprender nada.

Esto puede dar lugar a dos errores distintos. Por un lado, cuando el sistema se encuentra con una estructura (o conjunción de estructuras) que no se ha encontrado anteriormente, dará un aviso de inconsistencia. Por otro lado, cuando el sistema encuentra que falta una estructura que siempre ha visto, dará un aviso de incompletitud.

Evidentemente no podemos pretender que cualquier situación posible en nuestro dominio esté presente en alguna de las especificaciones de ejemplo, con lo que no podemos asegurar que las incompletitudes e inconsistencias detectadas por el sistema lo sean de verdad. Por ello, la decisión final la debe tomar siempre el ingeniero de software [16] y debemos entender los avisos del sistema de validación como tales: avisos sobre posibles incompletitudes o inconsistencias que el sistema deja al juicio del ingeniero de software decidir si realmente lo son o no.

Todo esto no quiere decir que el sistema que aquí hemos presentado no sirva para nada. Después de un extenso análisis de problemas propios de ingeniería del software, [11] argumenta que la mayor parte de ellos pertenecen a un conjunto tratable de clases jerárquicas. Entonces, si las abstracciones realizadas en la fase de aprendizaje son suficientemente buenas, y para un mismo dominio de aplicación y entorno de desarrollo, el número de situaciones nuevas que puede presentar una especificación respecto a un conjunto de ellas (el de entrenamiento) decrece rápidamente si el tamaño del conjunto crece.

Además, una vez validadas, las nuevas especificaciones se pueden ir incorporando al conjunto de entrenamiento, con lo que los posibles avisos erróneos se van corrigiendo y la fiabilidad del sistema de validación va en aumento.

7.2 Consistencia

Evidentemente, cada una de las especificaciones debe ser consistente. De no ser así, el sistema aprenderá las inconsistencias de los ejemplos y el resultado de la validación perderá cualquier valor que pudiera tener.

7.3 Minimalidad

Es importante que las especificaciones que formen parte del conjunto de entrenamiento sean mínimas. De este modo nos aseguramos que todas las propiedades que aprende el sistema son estrictamente necesarias en la especificación.

Esto es especialmente importante para la comprobación de la completitud, ya que justifica al sistema para poder asegurar que, cuando detecta la ausencia de una de esas propiedades, existe una incompletitud. De no ser mínimas las especificaciones, podríamos aprender propiedades innecesarias, lo que haría que el sistema pudiera detectar incompletitudes allí donde no las hay.

7.4 Completitud

Recíprocamente, también es importante que las especificaciones sean completas. Si no, el sistema podría aprender descripciones incompletas de objetos y por tanto detectar menos incompletitudes de las realmente existentes.

Además, la incompletitud de las especificaciones del conjunto de entrenamiento también afectaría a la detección de inconsistencias. El sistema no aprendería algunas de las propiedades necesarias, y cuando se las encontrara, las reportaría como inconsistencias.

7.5 Relajación de las Condiciones

El hecho de que para realizar el aprendizaje de las propiedades asociadas a cada concepto no se utilicen técnicas de aprendizaje simbólico sino que únicamente se mantenga un ratio del porcentaje de instancias que cumplen dichas propiedades hace que el sistema sea bastante robusto y resistente a errores.

Por ello, no es necesario que las especificaciones del conjunto de entrenamiento cumplan al cien por cien las condiciones especificadas. El sistema construirá un modelo de validación más o menos correcto aunque las especificaciones no sean totalmente correctas. Eso sí, cuantos más errores, más ejemplos necesitaremos para obtener una validación fiable.

8 Conclusiones y Trabajo Futuro

Hemos presentado en este artículo un sistema que a partir de especificaciones ejemplares es capaz de abstraer un modelo de especificación para comprobar la completitud y la consistencia de nuevas especificaciones.

La complejidad del problema (sobre todo por lo que se refiere a la completitud), la total vinculación de los términos completitud y consistencia al dominio de aplicación y la metodología usada para el desarrollo del software en cada caso concreto, así como el alto grado de informalidad que implica el que se trate de especificaciones preliminares escritas en lenguaje natural hacen que la construcción manual de una base de conocimiento que nos permita validar especificaciones resulte altamente inviable y costosa. Por ello, el uso de técnicas de aprendizaje automático para la construcción de dicha base de conocimiento parece una buena solución.

Por el momento hemos realizado pruebas satisfactorias, generando automáticamente conocimiento a partir de un metamodelo parecido al usado en el sistema KAOS y aprendiendo a partir del conocimiento generado. El resultado es que se aprenden (entre otros) los mismos conceptos que estaban en el metamodelo y con las mismas propiedades.

En un futuro próximo planeamos usar especificaciones reales, lo cual nos servirá para comprobar si el sistema es capaz de abstraer un modelo y un metamodelo a partir de ejemplos sobre los que no tenemos un metamodelo previo.

Esto nos ayudará a hacer un estudio más profundo sobre el conjunto de heurísticas que utilizamos para generar la jerarquía. Puede resultar interesante añadir algunas más de las reglas de generalización de Michalski como por ejemplo la de subir en el árbol de generalización, que nos permitiría realizar automáticamente la generalización de la figura 6.

Por otro lado, para incrementar la potencia del sistema de representación puede ser interesante introducir en las mismas estructuras conceptos reflexivos sobre ésta misma. Así por ejemplo, podríamos introducir el concepto SELF para referirse de una manera estándar al primer elemento de la estructura. Esto nos permitiría entre otras cosas descubrir propiedades de las relaciones temporales, comprobando que la cadena estructural CONCEPTO–ANTES–CONCEPTO–ANTES–SELF nunca se cumple y por tanto es inconsistente que A pase antes que B si B pasa antes que A. A un nivel más abstracto podríamos aprender qué relaciones son inversas de otras mediante la cadena: CONCEPTO–RELACION–CONCEPTO–INVERSA–SELF.

Por último, también pensamos trabajar sobre un sistema que nos permita ir eliminando todo el conocimiento generado que no nos sea útil. El sistema genera gran cantidad de conceptos y conexiones que después resultan ser inútiles. Si para problemas reales no se elimina buena parte de esta basura, la cantidad de espacio ocupado puede ser desmesurada.

9 Agradecimientos

Este trabajo está parcialmente subvencionado por la CICYT (TIC93-420) y por la CIRIT (GRQ93-3015 y una beca predoctoral a Jordi Alvarez).

Referencias

[1] V. Basili and B. Perricone. Software errors and complexity: An empyrical investigation. *Communications of the ACM*, 27(1):42–52, Jan. 1984.

[2] A. Borgida. Conceptual modeling of information systems. In M. Brodie and J. Mylopoulos, editors, *On Knowledge Base Management Systems*, chapter 31, pages 461–469. Springer-Verlag, 1986.

[3] N. Castell and A. Hernández. Filtering software specifications written in natural language. In *Proceedings of 7th Portuguese Conference on Artificial Intelligence, (EPIA'95)*, pages 447–455, Madeira, Portugal, Oct. 1995. LNAI-990, Springer Verlag.

[4] N. Castell, O. Slavkova, Y. Toussaint, and A. Tuells. Quality control of software specifications written in natural language. In *Proceedings of 7th International Conference on Industrial & Engineering Applications of AI & Expert Systems (IEA-AIE'94)*, pages 37–44, Austin, Texas, June 1994. Gordon and Breach Science Publishers.

[5] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20:3–50, 1993.

[6] P. Devanbu, R. Brachman, P. Selfridge, and B. Ballard. Lassie: A knowledge-based software information system. *Communications of the ACM*, 34(5):35–49, May 1991.

[7] A. Endres. An analysis of errors and their causes in system programs. *IEEE Transactions on Software Engineering*, 1(2):140–149, June 1975.

[8] M. Feather. Requirements reconnoitering at the juncture of domain and instance. In *Proceedings of 1st International Symposium on Requirements Engineering*, pages 73–76, San Diego, California, Jan. 1993. IEEE CS Press.

[9] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[10] Y. Kodratoff. *Introduction to Machine Learning*. Morgan Kaufman Publishers, 1988.

[11] N. Maiden, P. Mistry, and A. Sutcliffe. How people categorise requirements for reuse: a natural approach. In Requirements Engineering [15], pages 194–203.

[12] R. Michalski. A theory and methodology of inductive learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning, An Artificial Intelligence Approach*, pages 83–130. Morgan Kaufman Publishers, 1983.

[13] J. Mylopoulos, M. Jarke, A. Borgida, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362, Oct. 1990.

[14] R. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, New York, NY, USA, third edition, 1992.

[15] *Proceedings of 2nd International Symposium on Requirements Engineering*, York, England, Mar. 1995. IEEE CS Press.

[16] H. Reubenstein and R. Waters. The requirements apprentice: Automated assistance for requirements acquisition. *IEEE Transactions on Software Engineering*, 17(3):226–240, Mar. 1991.

[17] L. Shastri. A connectionist approach to knowledge representation and limited inference. *Cognitive Science*, 12:331–392, 1988.

[18] G. Spanoudakis and P. Constantopoulos. Similarity for analogical software reuse: A computational model. In *Proceedings of 11th European Conference on Artificial Intelligence*, Amsterdam, Aug. 1994.

[19] Y. Toussaint. *Méthodes Informatiques et Linguistiques pour l'aide a la Spécification de Logiciel*. PhD thesis, Paul Sabatier University, Toulousse, 1992.

[20] A. van Lamsweerde, R. Darimon, and P. Massonet. Goal-directed elaboration of requirements for a meeting scheduler: Problems and lessons learnt. In Requirements Engineering [15], pages 194–203.