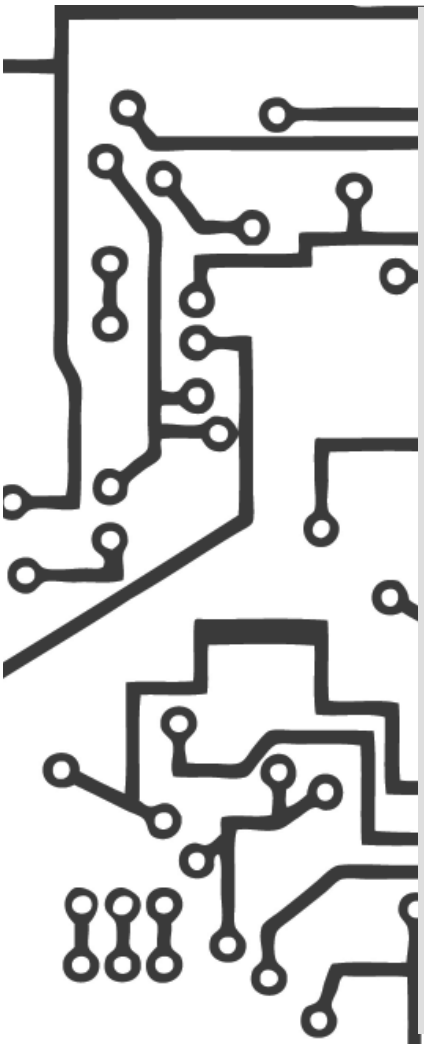
REVISTA Universidad EAFIT
Vol. 44. No. 151. 2008. pp. 93-106

UNC-Corpus

Corpus de diagramas UML para la solución de problemas de completitud en ingeniería de software*



Carlos M. Zapata J.
Ph. D. en Ingeniería de Sistemas. Profesor Asociado, Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia, Sede Medellín.
Integrante del Grupo de investigación en Lenguajes Computacionales de la misma institución.
cmzapata@unal.edu.co

Juan C. Hernández P.
Ingeniero de Sistemas e Informática, Universidad Nacional de Colombia, Sede Medellín.
Integrante del Grupo de investigación en Lenguajes Computacionales de la misma institución.
jcherna0@unalmed.edu.co

Raúl A. Zuluaga
Ingeniero de Sistemas e Informática, Universidad Nacional de Colombia, Sede Medellín.
Integrante del Grupo de investigación en Ingeniería de Software de la misma institución.
razulua0@unalmed.edu.co

Recepción: 10 de abril de 2008 | Aceptación: 18 de junio de 2008

* Este artículo se elaboró en el marco del proyecto “Un modelo de diálogo para la generación automática de especificaciones en UN-Lencep”, financiado por la DIME (División de Investigación de la Sede Medellín-Universidad Nacional de Colombia.

Resumen

Los corpus computacionales se utilizan como apoyo en el procesamiento del lenguaje natural (PLN) para resolver problemas de desambiguación, traducción y generación automática de textos, entre otras funciones. Para ello, se explota la característica principal de los corpus (el hecho de que poseen usos comprobados de un lenguaje) y se combina con análisis estadísticos y métodos de extracción de información basados en redes neuronales o algoritmos genéticos. En ingeniería de software, no existe evidencia del uso de corpus computacionales de diagramas. Un uso similar lo constituyen los repositorios de diagramas, que suelen manejar ejemplos reales de diagramas, especialmente para reutilización, pero sin usar la estadística o los métodos heurísticos para la extracción de información. En este artículo, se propone UNC-Corpus, una herramienta para el manejo de un corpus de diagramas construidos en el Lenguaje Unificado de Modelado UML, que aplica técnicas tradicionales de PLN en la solución de problemas de completitud en la ingeniería de software.

Palabras Clave

Corpus anotado
Diagramas UML
XMI
Repositorio
Metamodelado
PLN
Extracción de información

UNC-Corpus: a UML-diagram corpus to solve completeness problems in software engineering*

Abstract

Computational corpora are used as tools in Natural Language Processing (NLP) to solve disambiguation, translation and automated text generation problems. In order to complete these tasks, the main feature of computational corpora (the fact that they have proven uses of a language) is combined with statistical analysis along with information extraction methods based on neural networks or genetic algorithms. In software engineering, there is no evidence supporting the use of diagram computational corpora. Diagram repositories have a similar application working with real examples of diagrams (mainly for reuse purposes), but without using neither statistics nor heuristic methods for information extraction. In this paper, the UNC-Corpus, a tool for managing a corpus of UML (Unified Modelling Language) diagrams, which applies NPL traditional techniques in order to solve completeness problems in software engineering, is proposed.

Key words

Annotated corpus
UML diagrams
XMI
Repository
Metamodelling
NLP
Information extraction

*This article was prepared in the framework of the project "A dialogue model for the automated generation of specifications in UN-Lencep", funded by DIME (Research Division, Branch Medellin-National University of Colombia).

Introducción

Actualmente, crece la necesidad de disponer de muestras de información reales para resolver distintos problemas de las ciencias aplicadas. Esto implica que, de algún modo, hay que recopilar, en cantidades más o menos grandes, muestras de los elementos que constituyen la realidad que se quiere observar (Torruela y Llisterri, 1999). Los corpus computacionales son herramientas para la solución de esta necesidad en el área del procesamiento del lenguaje natural (PLN). En los corpus se dan las características necesarias para poder almacenar y organizar muestras reales de un dominio específico para su posterior análisis.

Es común encontrar aplicaciones de software que almacenen artefactos ingenieriles o repositorios de artefactos (Berstein, 1998) que cuentan con características similares a los corpus computacionales. Sin embargo, a diferencia de estos últimos, dichas herramientas se enfocan en la administración de los objetos almacenados para facilitar de ese modo su posterior reutilización. Los corpus, en cambio, no se centran en la administración de la información que ellos contienen, sino en el análisis de la información de que disponen, como apoyo a otras herramientas y procesos de PLN.

En ingeniería de software, las muestras de elementos (en este caso diagramas) se deberían emplear para resolver problemas de completitud y consistencia, que son dos de los conceptos que miden la calidad de una aplicación de software (Zowghi & Gervasi, 2003). La definición de la completitud de especificaciones no es una tarea trivial, y es por esto que detectar las carencias de completitud de especificaciones no es fácil (Davis, 1993). Sin embargo, las colecciones de diagramas, denominadas “repositorios”, se emplean comúnmente para la reutilización de diagramas en modelos similares, pero, en este caso, no se emplean técnicas estadísticas u otras similares que se suelen aplicar a los corpus computacionales del PLN.

Por las razones anotadas, en este artículo se propone el UNC-Corpus, una herramienta para manipular de un conjunto de muestras almacenadas de diagramas que permite, a través de una consulta, conocer la información estadística de una palabra o grupo de palabras en relación con el uso correspondiente en los diagramas, con el fin de solucionar problemas de consistencia y completitud. Los usuarios que tienen acceso a esta herramienta pueden ser aplicaciones de software, como en el caso descrito por Zapata, Estrada y Arango (2007) o personas, que requieren definir a cuáles elementos de uno o varios diagramas corresponde una palabra y el respectivo grado de relevancia de cada una de las ocurrencias. El UNC-Corpus almacena la información de los diagramas de UML en dos formatos: uno correspondiente al estándar de las herramientas CASE (las aplicaciones de ingeniería de software asistida por computador) y otro correspondiente a una base de datos interna que posibilita la realización de consultas. Este trabajo se basa en la idea de que la información de un diagrama es útil, pero lo es más cuando se combina con la información de muchos ejemplos de diagramas almacenados de manera computacionalmente tratable, y se hacen consultas no solo sobre un tipo de diagrama en particular, sino sobre el conjunto de diagramas disponibles.

La exposición subsiguiente se estructura así: en la sección 1, se presentan la definiciones de corpus computacional, repositorios de artefactos software y el problema de la completitud en la ingeniería de software; en la sección 2, se realiza una revisión del estado del arte en repositorios y corpus de diagramas; en la sección 3, se muestra el UNC-Corpus; finalmente, en las dos últimas secciones se discuten las conclusiones y el trabajo futuro.

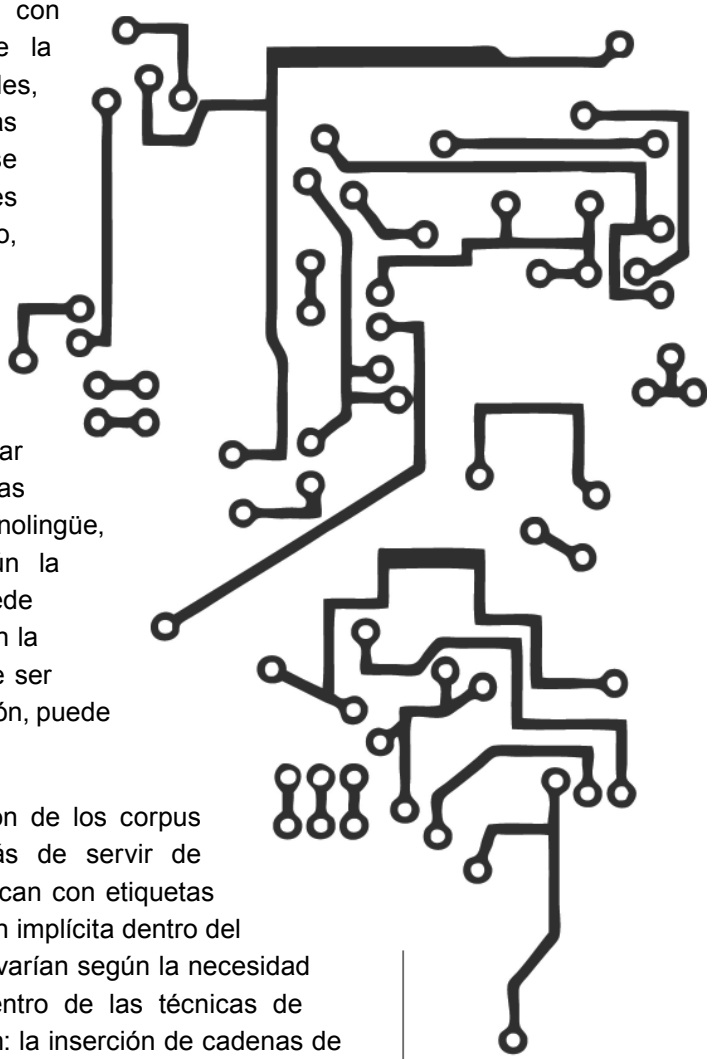
1. Marco Teórico

1.1 Corpus

Un corpus computacional es una colección de más de un documento que contiene usos comprobados de un lenguaje. Dichos documentos se seleccionan

y ordenan de acuerdo con criterios lingüísticos explícitos, para usarlos como ejemplo de dicho lenguaje.

Un corpus computacional se refiere a una colección de documentos codificados electrónicamente, ya sea en una base de datos o un conjunto de archivos que se integran a un sistema de almacenamiento (Sánchez, 1999). Según este autor, ese tipo de herramientas tiene como objetivo ofrecer un conjunto de documentos donde se muestran datos reales y exhaustivos, muestras del mundo real que enseñan el uso, el comportamiento y/o las tendencias de conceptos y patrones. Normalmente, los corpus se utilizan con fines lingüísticos, donde se requiere la validación de reglas y formas gramaticales, morfológicas, sintácticas, semánticas y pragmáticas de la lengua que se estudia. Como parte de sus principales aplicaciones en el ámbito lingüístico, los corpus se utilizan en traducción automática, obtención de conocimiento léxico y terminológico, recuperación de información y desambiguación.



Existen diferentes maneras de clasificar un corpus: según el número de lenguas que intervienen en éste, puede ser monolingüe, multilingüe, bilingüe o paralelo; según la cantidad de texto seleccionado, puede ser textual, de referencia o léxico; según la especialidad de los documentos, puede ser general o específico; según la codificación, puede ser simple o anotado.

Los corpus anotados son una variación de los corpus (McEnery, 2003), en la cual, además de servir de almacén de documentos, estos se marcan con etiquetas que permiten hacer explícita información implícita dentro del documento. Las técnicas de anotación varían según la necesidad para la que el corpus se diseñe. Dentro de las técnicas de anotación más comunes, se encuentran: la inserción de cadenas de texto, las etiquetas dentro de los mismos documentos y la reescritura de los documentos con el fin de clasificar su contenido.

Aprovechando la anotación de un corpus, se puede proveer una solución a partir de las propuestas del enfoque lingüístico y, en particular, aquellas con un enfoque estadístico, como las descritas en Levow (1997) o aquellas en que la frecuencia de aparición de un término en documentos de determinado contexto se utiliza como criterio de decisión para darle significado al mismo término que se consulta.

Éste es un acercamiento supervisado (Uzuner, 1998), pues requiere cierto nivel de marcación de los componentes del corpus, en su contexto y en su corrección, pero se aleja del enfoque tradicional de corpus anotado, en el que los elementos dentro de cada documento simplemente se marcan por medio de etiquetas.

1.2 Repositorios de artefactos software

Bernstein (1998), define un repositorio como una base de datos de información acerca de artefactos ingenieriles, tales como software, documentos, mapas o sistemas de información. El desarrollo de estos artefactos requiere el uso de otras herramientas implicadas en el proceso de elaboración de software, como las herramientas CASE. Como funciones de un repositorio de artefactos, se consideran: el almacenamiento de modelos para soportar operaciones realizadas por las herramientas CASE, el control de versiones y la administración de los procesos de elaboración de software, entre otros.

Al almacenar un objeto en un repositorio, es posible guardar información de clasificación del objeto que facilite su búsqueda, lo cual es equivalente a una etiqueta en un corpus. Cada objeto se extrae de manera parcial o total, como una copia que se utiliza en otros procesos, permitiendo así el ahorro de tiempo en las distintas fases del desarrollo de una aplicación. Los repositorios de código fuente son los ejemplos típicos de este tipo de almacenamiento. En ellos se puede contar con el código fuente allí contenido, para su posterior uso en otra fase del proyecto de desarrollo o en otros proyectos.

1.3 Completitud

Para considerar “completa” una especificación de requisitos, ésta debe poseer tres características fundamentales (Boehm, 1984): no debe haber información sin especificar, la información no debe contener objetos o entidades no definidas y, por último, toda la información necesaria para definir el problema debe estar en la especificación. Según estas características, es casi imposible la definición y medida de la completitud de especificaciones. El objetivo consiste, entonces, en determinar si una especificación de requisitos es suficientemente completa para el tipo de sistema modelado y una forma de realizarlo es a través de los esquemas conceptuales elaborados a partir de tales especificaciones (Zapata, Estrada y Arango, 2007). Los modelos que parten de especificaciones incompletas, pueden estar correctos, pero no completos, lo que influye luego en el desarrollo del proyecto y la calidad del software construido.

2. Revisión de la Literatura

Buena parte de las herramientas CASE cuenta con repositorios que administran los diagramas que se construyen a través de ellas mismas. Los repositorios en este tipo de herramientas ofrecen diversas funcionalidades, tales como la administración de los objetos que almacenan, la administración de las relaciones y dependencias entre estos objetos y la administración de versiones, entre otros (Berstein, 1998). Existe una tendencia generalizada a agregar nuevas funcionalidades a los repositorios más allá de la reutilización. Un ejemplo de ello se puede observar en el trabajo de López, Laguna y García (2002), en el que se almacenan diagramas de requisitos clasificados según una familia de modelos ya definida y un dominio de aplicación, para luego construir un catálogo que permite ver una lista de los mismos y ubicar los diagramas almacenados bajo determinada categoría. Otros acercamientos cambian la función pasiva del repositorio para convertirlo en un programa activo. Por ejemplo, el proyecto SENSOR (Ritter & Steiert, 2000) propone una herramienta para el almacenamiento y consulta de diagramas UML y, basado en reglas en lenguaje OCL, busca implementar la consistencia entre modelos al momento de ingresarlos al repositorio. En dichos repositorios se suele utilizar como método de almacenamiento de información la jerarquía de clases, en la cual se elabora un metamodelo de la información que se desea guardar, como se implementa en el trabajo de Keller, Bédard y Saint-Denis (2001).

Los repositorios de artefactos de software ofrecen grandes funcionalidades cuando actúan como sistemas de almacenamiento, administración y consulta de la información que almacenan. Sin embargo, los repositorios actuales no cuentan con la capacidad de indicar cuál es el uso más común de una palabra al interior de los diagramas o artefactos que el mismo repositorio almacena, y ésta es una de las preguntas que se debe contestar cuando se está buscando que un modelo sea completo.

Existen aproximaciones que pretenden encontrar en los discursos de requisitos, frases con información incompleta. Entre estas aproximaciones, Zapata, Jaramillo y Arango (2006) emplean la gramática de casos para, a partir de los componentes que deben acompañar un verbo en una oración, sugerir la información que falta en las oraciones.

Lo que se quiere explotar, como solución al problema de la completitud, es la frecuencia de uso de un diagrama o parte del mismo. Si muchos analistas o diseñadores diferentes realizan diagramas similares para un mismo contexto y los usan repetidamente, se podría considerar como una forma común de modelado. El hecho de que estos diagramas tengan diferentes formas de representar un mismo concepto o que, incluso, presenten ciertos errores en el uso de los símbolos, puede constituir una barrera para el uso de los mismos. Esta barrera se supera si se cuenta con una gran cantidad de diagramas ya que, al disponer de un gran volumen de estos, se logra mostrar la tendencia de las estructuras más comunes y, por ende, las de mayor uso, posicionando los modelos incorrectos como casos rechazables que no influyen significativamente en las tendencias de los modelos correctamente diseñados.

Lo anterior permite utilizar el concepto de la frecuencia de uso de los modelos, cuando se quiera conocer qué elementos se deberían incluir o eliminar de un modelo para contribuir a las expectativas de completitud que se posean.

3. UNC-Corpus

La iniciativa de un corpus de diagramas surge como una propuesta para aliviar en parte el problema de la completitud de los diagramas utilizados en la ingeniería de software, ofreciendo así una herramienta de apoyo al analista en la fase de modelado del sistema.

Los diagramas almacenados por UNC-Corpus poseen un archivo elaborado en la herramienta CASE de origen, que presenta un nivel de etiquetado, es decir, cada documento contiene la información que define un diagrama y todos sus elementos. Para el almacenamiento de dicha información en su propia base de datos, el UNC-Corpus toma las etiquetas y los elementos correspondientes a los diagramas que maneja. Debido a esta característica, es posible considerar el UNC-Corpus como un corpus anotado, con las ventajas que se mencionaron en la sección 1.1.

En los procesos de desarrollo de productos de software, los diagramas de UML se construyen utilizando herramientas CASE (Quintero *et al.*, 2005). Estas herramientas poseen un estándar de intercambio de información, denominado XML de Intercambio de Metadatos (XMI) (OMG, 2008), el cual se basa en el lenguaje de marcado extensible XML (W3C, 2008a).

La solución que se propone en este artículo toma los diagramas de UML en formato XMI, que se generan con la herramienta CASE Poseidon™ (Gentleware AG, 2008). Para poder considerar los elementos susceptibles a estudio y consultas, se tiene que, a cada archivo XMI, se le extraen las etiquetas de los diagramas que trabaja el UNC-Corpus, las cuales se guardan en una base de datos relacional con su respectiva información semántica. Es decir, se requiere la definición de una estructura de almacenamiento de datos coherente con las características de los modelos y una conversión de los datos desde XMI hacia las tablas de las bases de datos. Se podrían, simplemente, almacenar los archivos XMI y

realizar las consultas necesarias sobre dichos archivos, pero se implementó una base de datos por dos razones fundamentales: desempeño y facilidad en la consulta. Sobre un documento XMI es posible realizar consultas, pero sólo dentro del mismo documento y, dado que el objetivo de este trabajo es acumular tantos diagramas como sea posible para lograr resultados estadísticamente significativos, hacer consultas sobre archivos individuales no es una opción.

En el UNC-Corpus, se pueden hacer consultas por pantalla o simplemente acceder a la base de datos y realizar consultas escritas en el Lenguaje Estructurado de Consulta SQL desde cualquier aplicación, empleando la estructura de la base de datos que se presenta en este artículo y solicitando el nombre del servidor, la base de datos, el usuario y la contraseña correspondientes.

La gestión de la información, dentro del corpus de diagramas, la constituyen dos partes: la definición de los diagramas y las instancias de dichas definiciones. En cuanto a la primera parte, un diagrama se puede almacenar como un conjunto de entidades y diferentes tipos de relaciones. Por ejemplo, en el diagrama de clases, se pueden entender como entidades: “Clase”, “Atributo”, “Operación” y “Generalización”, y como relaciones: “tiene”, “es de tipo”, “es padre” y “es hijo”. Con estos elementos, se pueden generar relaciones más complejas entre clases, por ejemplo: “la Clase *tiene* Atributo”, “la Clase *tiene* Operación”, “el Atributo *es de tipo* Clase”, “la Clase *es padre* en la Generalización” y “la Clase *es hijo* en la Generalización”.

El UNC-Corpus maneja, actualmente, diagramas de clases, casos de uso y máquina de estados con notación estándar de UML, pero puede soportar cualquier diagrama que se pueda reducir a las entidades y relaciones que forman un diagrama.

3.1 Funcionamiento del UNC-Corpus

El proceso parte de la elaboración de un diagrama de clases, casos de uso o máquina de estados en Poseidon™ (Gentleware AG, 2008) y la generación del archivo en formato XMI correspondiente. El interesado ingresa al módulo de carga de archivos de la aplicación y allí agrega el archivo XMI que contiene la información del diagrama que quiere incluir (Figura 1). El sistema muestra una ventana donde solicita el archivo XMI que se desea ingresar y pide al interesado que seleccione un contexto de los existentes sobre el que aplica el diagrama que se pretende cargar o que, en su defecto, escriba un nuevo contexto. El UNC-Corpus carga el archivo XMI y lo somete a una serie de consultas para extraer la información presente en el documento; tales consultas se implementaron en las tecnologías basadas en el Lenguaje de recorrido de XML (XPATH) (W3C, 2008b) y se definieron para los diagramas de clases, casos de uso y máquina de estados. Los resultados de cada consulta se almacenan indicando qué entidad o relación se consultó y, cuando esta ocurrencia ya existe en la base de datos, se incrementa en una unidad el atributo “frecuencia” que cada registro tiene, indicando así que la ocurrencia de la entidad o la relación, se encuentra repetida.

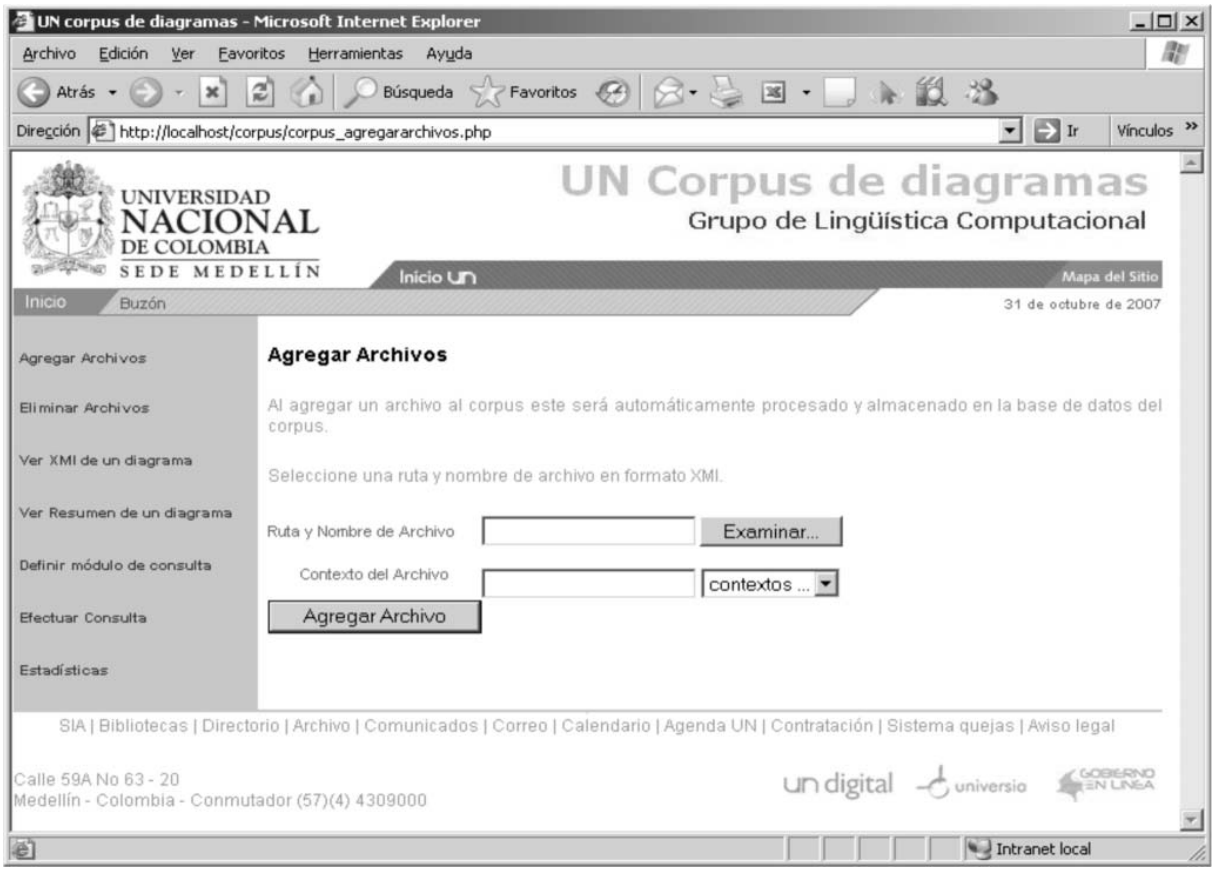
Las siguientes, son las tablas relacionales sobre las que se guarda la información extraída de los archivos XMI (véase la Figura 2).

Definición de Diagramas: Las primitivas y las relaciones entre ellas son las que conforman los diagramas. Así, existen tres tablas que permiten almacenar la estructura de cada diagrama: *diagrama, primitiva* y *relación*.

Almacenamiento de Instancias de primitivas y relaciones: todas las ocurrencias de las primitivas y las relaciones de un diagrama que se capturan del archivo en formato XMI, se almacenan en las tablas *instanciaPrimitiva* e *instanciaRelación*. Si la ocurrencia encontrada ya se creó, se incrementa en una unidad el campo ‘frecuencia’ de la instancia de la primitiva o la relación según el caso.

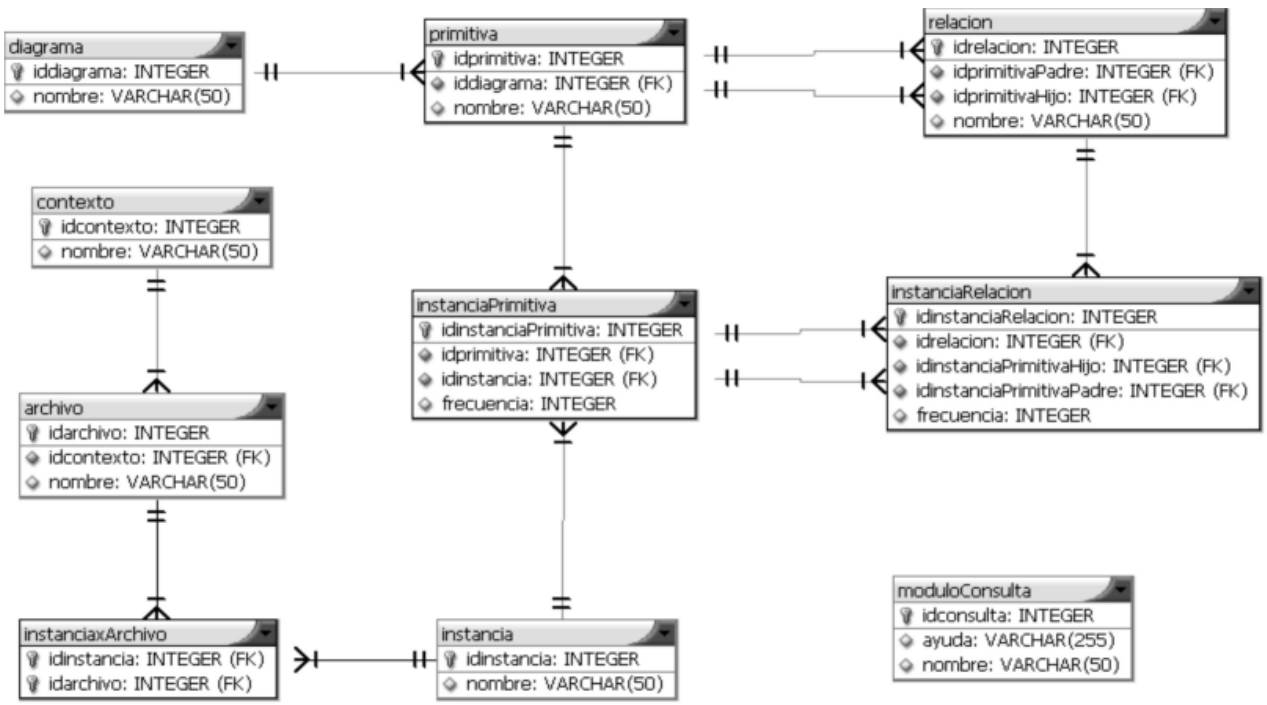
Definición de consultas: el UNC-Corpus cuenta con una interfaz SQL que permite definir consultas sobre la información almacenada.

Figura 1. Módulo de carga de archivo del UNC-Corpus



Fuente: Elaboración propia.

Figura 2. Diagrama entidad-relación correspondiente al UNC-Corpus



Fuente: Elaboración propia.

Una serie de consultas XPATH se asocia con las definiciones de los diagramas, para permitir la extracción de las instancias y relaciones que se desean guardar en la base de datos, a partir del archivo XMI. Una consulta común en XPATH, la que retorna todas las entradas de la entidad Clase en UML, sería la siguiente:

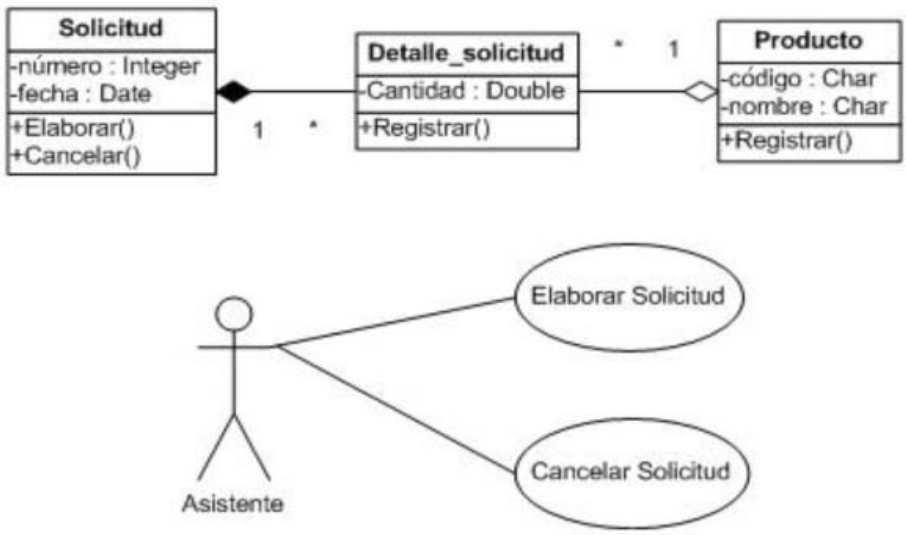
*/XMI/XMI.content/UML:Model/ UML:Namespace.ownedElement/ UML:Class/@**

Esta consulta retorna una colección con todos los nodos dentro del documento XMI que se encuentren en esta ruta que, igualmente, equivalen a la primitiva "Clase" dentro del diagrama de clases para la definición de XMI en su versión 1.2.

3.2 Caso de Estudio

En cuanto al manejo de aplicación, supóngase que en el UNC-Corpus se guardaron varios diagramas que pertenecen al manejo de solicitudes en diferentes contextos. Supóngase, adicionalmente, que los diagramas de la Figura 3 pertenecen a un modelo particular que se desea modelar, por ejemplo el contexto de las solicitudes en una empresa manufacturera. Es probable que los diagramas similares almacenados en el UNC-Corpus posean "Solicitud", "Detalle_solicitud" y "Producto" como instancias de la primitiva "Clase" del diagrama de clases. Por otra parte, "Asistente" puede ser una instancia de la primitiva "Actor" y "Elaborar solicitud" y "Cancelar solicitud" pueden ser instancias de la primitiva "Caso de uso". "Actor" y "Caso de uso" pertenecen al diagrama de casos de uso. Así, cualquiera de las palabras que se encuentran en esos diagramas podría estar vinculada con muchas instancias de primitivas dentro del UNC-Corpus, cada una de las cuales tendrá asociada una frecuencia.

Figura 3. Un ejemplo de diagrama de clases



Fuente: Elaboración propia.

Por otra parte, empleando la entidad "moduloConsulta" del diagrama entidad-relación de la Figura 2, en el UNC-Corpus se puede almacenar una consulta en SQL como la siguiente:

```
SELECT
      (ip.frecuencia/cantidad.nro*100) as frecuencia,
      p.nombre as primitiva,
      i.nombre as instancia
FROM
       instanciaPrimitiva ip,
       instancia i,
       primitiva p,
       ( SELECT
        sum( ip.frecuencia ) as nro
   FROM
       instanciaPrimitiva ip
   WHERE ip.idinstancia in
( SELECT
          idinstancia
        FROM
          instancia i
        WHERE
          nombre LIKE '%$palabra%')
  ) cantidad
 WHERE
        ip.idinstancia = i.idinstancia AND
        ip.idprimitiva = p.idprimitiva AND
        i.nombre LIKE '%$palabra%'
ORDER BY frecuencia DESC
```

Una forma de uso del UNC-Corpus sería que un usuario humano accediera a la aplicación e ingresara por la opción “Efectuar consulta”, para luego digitar el término “solicitud”. En ese caso, la respuesta sería la que se muestra en la Figura 4. La otra opción es que una aplicación se programara para acceder al UNC-Corpus, con una serie de comandos de conexión como el siguiente, que se encuentra en lenguaje C#:

```
MySqlConnection oMySqlConnnumconeccion = new MySqlConnection();

if (oMySqlConnnumconeccion != null)

oMySqlConnnumconeccion.Close();

oMySqlConnnumconeccion.ConnectionString = "Server=servidorcorpus.edu.co;Database
=corpus;Uid=visitante;Pwd=123;";

//Los datos correspondientes al servidor, la base de datos, el usuario y el password, se
deben solicitar al administrador del UNC-Corpus.

oMySqlConnnumconeccion.Open();

MySqlDataReader readernumconeccion = null;
```

Figura 4. Resultados arrojados por la aplicación luego de consultar los roles que puede tomar el término *solicitud*.

Fuente: Elaboración propia.

Finalmente, si se entrega el valor *$palabra* = 'solicitud', el resultado sería el mismo de la Figura 4, pero los resultados se podrían capturar en la aplicación correspondiente.

El apoyo que puede suministrar el UNC-Corpus a los diagramas del ejemplo, se relacionan con los usos identificados en la consulta. Por ejemplo, aparecen otros casos de uso que el autor de los diagramas no tomó inicialmente en cuenta: "Abrir solicitud", "Anular solicitud", "Aceptar solicitud", "Abrir solicitud de compra", "Imprimir solicitud" y "Rechazar solicitud". Es probable que no todas estas instancias de la primitiva caso de uso sean aplicables al dominio particular de los diagramas de la Figura 3, pero suministran una oportunidad para interrogar al interesado al respecto. Igual ocurre con las instancias "Solicitud de compra" y "Solicitud prestamo", pertenecientes a la primitiva "clase". Esas instancias suministran argumentos para establecer un diálogo de completitud con el interesado, con el fin de precisar los términos a los que se refiere su dominio particular. Zapata, Estrada y Arango (2007) emplean el UNC-Corpus de manera similar al ejemplo planteado, pero se enfocan específicamente en la determinación de atributos y operaciones que se ligan con una clase particular, complementando la herramienta UNC-Diagramador que presentan Zapata, Ruiz y Villa (2007) con un módulo que verifica la completitud del diagrama de clases.

Finalmente, es necesario anotar que, a medida que se incrementa el número de diagramas ingresados en el UNC-Corpus, las formas de modelado habituales de los diferentes conceptos se van incrementando en frecuencia, relegando a porcentajes cada vez más bajos los usos erróneos de los diagramas. Por ejemplo, tendrá un porcentaje muy bajo una palabra como "solicitar" cuando se usa como clase, pero tendrá un porcentaje muy alto cuando se emplee como operación del diagrama de clases o como interacción de un caso de uso.

Conclusiones

Los repositorios de diagramas son herramientas comúnmente utilizadas en el modelado, especialmente para reutilización, creación de catálogos de diagramas o verificación de la consistencia entre ellos. Sin embargo, los repositorios no tienen capacidades de análisis estadístico o heurístico, que sí son comunes en ciertos usos del PLN, como es el caso de los corpus. Estas aplicaciones, generalmente conformadas por muchos documentos que reúnen usos identificados de un lenguaje, posibilitan el análisis profundo de una lengua por las grandes capacidades de análisis que le brindan a las herramientas que trabajan con PLN. Los corpus anotados, son especialmente útiles en labores como traducción automática o extracción de información.

Tratando de dotar a los repositorios de software con capacidades de análisis que permitan examinar la completitud de diagramas, en este artículo se presentó el UNC-Corpus, una herramienta para el manejo de corpus de diagramas, se definió su estructura interna y se ejemplificó la manera en que puede contribuir a solucionar algunos de los problemas relativos a la completitud de diagramas.

El UNC-Corpus genera su valor agregado a partir del uso reiterado de una forma de modelado en un determinado dominio. Esto implica que, a medida que se vayan incorporando diagramas al UNC-Corpus, las formas comunes de modelar los términos de un dominio se irán imponiendo, suministrando una información cada vez más confiable a las aplicaciones o usuarios que lo puedan usar. Así, el UNC-Corpus se puede convertir en un patrón de solución dentro del contexto donde se definen los diagramas, al suministrar sugerencias de instancias de primitivas y relaciones que se puedan validar con el interesado para mejorar la completitud de los diagramas.

La efectividad del UNC-Corpus en la solución de problemas de completitud depende de la cantidad de diagramas que se ingresen en su base de datos, de forma tal que se puedan cubrir múltiples dominios y de forma muy detallada.

Trabajo Futuro

Dentro de la herramienta UNC-Corpus, un primer trabajo futuro que se puede plantear es el mejoramiento del sistema de búsquedas para hacerlo más inteligente, permitiendo resolver a un cierto nivel: la proximidad sintáctica, sinónimos y accidentes gramaticales que puedan estar presentes en los términos de búsqueda.

En un segundo trabajo, se requiere la implementación de algunos otros usos de la información contenida en el corpus, por ejemplo la realización de sugerencias para la corrección de errores de consistencia en algún diagrama para un dominio específico, la conversión entre diagramas de un mismo tipo (por ejemplo, la obtención del diagrama entidad-relación a partir del diagrama de clases y viceversa) o el estudio de completitud de un diagrama a partir de otro que tenga información complementaria (por ejemplo, complementar las operaciones del diagrama de clases tomando en consideración las interacciones de los casos de uso).

Un tercer trabajo futuro que se puede derivar de este artículo es el estudio, mediante técnicas de análisis de información, de los diagramas que están allí contenidos, para extraer la estructura de un "Metamodelo" de los diferentes componentes de una especificación de software, buscando sugerir diferentes tipos de preguntas que se pueden realizar en el proceso de captura y especificación de requisitos.

Bibliografía

Bernstein, P. (1998). "Repositories and Object Oriented Databases", *ACM SIGMOD, 27*(1). New York, pp. 88-96.

Boehm, B. (1984). "Verifying and validating software requirements and design specifications", *IEEE Software, 1*(1). Los Alamitos, pp. 75–88.

Davis, A. M. (1993). *Software Requirements: Analysis and Specification* (2ª ed.). New Jersey: Prentice Hall.

Gentleware AG. (2008). *Poseidon for UML*. http://www.gentleware.com/products.html (Agosto 20 de 2008).

Keller, R.; Bédard, J. & Saint-Denis, G. (2001). "Design and implementation of a UML-based design repository", *Lecture notes in computer science, (2068)*. Londres, pp. 448-464.

Levow, G. (1997). "Corpus-based techniques for word sense disambiguation", *Technical Report AIM-1637*. Massachussets, pp. 1-20.

López, O.; Laguna, M. & García, F. (2002). "Reuse based analysis and clustering of requirements diagrams", *Eighth International Workshop on Requirements Engineering: Foundation for Software Quality*. Essen, Germany: Essener Informatik Beiträge, pp. 71-83.

McEnery, T. (2003). "Corpus Linguistics". Mitkov, R. (Ed.) *The Oxford Handbook of Computational Linguistics,* New York: Oxford University Press, capítulo 24. pp. 448-463.

OMG. (2008). *XMI Specification*. http://www.omg.org/technology/documents/formal/xmi.htm (8 de abril de 2008).

Quintero, J.; Anaya, R.; Marín, J. y Bilbao, A. (2005). "Un estudio comparativo de herramientas para el modelado con UML", *Revista Eafit, 41*(137). Medellín, pp. 60-76.

Ritter, N. & H. Steiert. (2000). "Enforcing modelling guidelines in an ORDBMS-based UML-Repository", *Proceedings of the 2000 information resources management association international conference on Challenges of information technology management in the 21st century*. Anchorage: IGI Publishing, pp. 269-273.

Sánchez, A. (1999). "Una propuesta de codificación morfosintáctica para corpus de referencia en lengua española", *Estudios de Lingüística del Española, (3)*. http://elies.rediris.es/elies3/ (8 de abril 8 de 2008).

Torruela, J. y J. Llisterri. (1999). "Diseño de corpus textuales y orales", J. M. Blecua; G. Clavería; C. Sánchez y J. Torruela (eds.): *Filología e informática: Nuevas tecnologías en los estudios filológicos*. Barcelona: Milenio, pp. 45-77.

Uzuner, O. (1998). "Word Sense Disambiguation Applied to Information Retrieval". Master's thesis. Massachussets: Massachussets. Institute of Technology.

W3C. (2008a). *XML Specification*. http://www.w3.org/TR/REC-xml/ (Abril 8 de 2008).

W3C. (2008b). *XPATH Specification*. http://www.w3.org/TR/xpath (Abril 8 de 2008).

Zapata, C. M.; Estrada, B. y Arango, F. (2007). "Un método para el refinamiento interactivo del diagrama de clases de UML", *Dyna, 152-153* (74). Medellín, pp. 253-266.

Zapata, C. M.; Jaramillo, A. y Arango, F. (2006). "Una propuesta para mejorar la completitud de requisitos utilizando un enfoque lingüístico", *Ingeniería y Desarrollo, 19*. Barranquilla, pp. 1-16.

Zapata, C. M.; Ruiz, L. M: y Villa, F. (2007). "UNC-Diagramador: una herramienta Upper CASE para la obtención de diagramas UML desde Esquemas Preconceptuales", *Revista EAFIT, 43* (147). Medellín, pp. 68-80.

Zowghi, D. & V. Gervasi. (2003). "On the interplay between consistency, completeness, and correctness in requirements evolution", *Information and Software Technology, 45* (14). Newton, pp. 993-1009.