Editor de Estructuras Conceptuales Evolutivas: Consideraciones Prácticas

Fernando Molina Ortiz[1], Lina García Cabrera[2], Nuria Medina Medina[3], María Visitación Hurtado Torres[4]

[1,3,4]Dpto. Lenguajes y Sistemas Informáticos. Universidad de Granada.
[2]Dpto. Informática. Universidad de Jaén.
[1]fmo@ugr.es [2]lina@ujaen.es [3]nmedina@ugr.es [4]mhurtado@ugr.es

Resumen En este artículo se presentan algunos aspectos de la implementación del prototipo JSemHP, concretamente los concernientes al subsistema de memorización. Este prototipo se basa en el modelo hipermedia semántico evolutivo SEM-HP, algunos de cuyos aspectos se han extendido durante el proceso de diseño. Se describe la adaptación del patrón de diseño modelo-vista-controlador a nuestro paradigma evolutivo, el entorno de desarrollo, la estructura básica de clases y la implementación de aspectos fundamentales como las estructuras conceptuales y la evolución del sistema.

Palabras clave: sistemas hipermedia, evolución, ontologías, Java

1. Introducción

La intención inicial fue desarrollar un prototipo basado en el modelo hipermedia SEM-HP [2], aunque finalmente se ha logrado un diseño bastante más versátil que puede ser utilizado en otros dominios. En concreto se está usando, además de en SEM-HP, en la edición de ontologías especiales enriquecidas con relaciones del modelo OO, utilizadas para integrar información procedente de Sistemas de Información y Sistemas de Ayuda a la Decisión [3]. En este artículo se describe una implementación inicial del subsistema de memorización, limitándonos a comentar los aspectos más destacables. En la sección 2 discutimos algunas ampliaciones hechas al modelo, para luego discutir el patrón modelo-vista-controlador desde nuestra perspectiva (sección 3). La sección 4 se dedica a describir con más detalle la implementación de nuestro prototipo, incluyendo el lenguaje y librerías empleadas (4.1), la estructura básica de nuestro sistema (4.2), la implementación de la estructura conceptual (4.3), el desarrollo de los aspectos evolutivos (4.4) y un comentario sobre los atributos genéricos de los nodos (4.5). Se finaliza con las conclusiones y el trabajo futuro (5).

2. Extensiones a SEM-HP

2.1. El modelo SEM-HP

El modelo SEM-HP es un modelo sistémico, evolutivo y semántico para el desarrollo de sistemas hipermedia adaptativos [2]. En este modelo un sistema hipermedia esta formado por tres sistemas interrelacionados. El *sistema de conocimiento*, que almacena y mantiene el conocimiento del autor tiene dos partes: el *subsistema de memorización*, que estructura el dominio conceptual y de información mediante una estructura conceptual, y el *subsistema de presentación* que permite crear diferentes vistas de ésta. Una estructura conceptual (EC) es una red semántica con dos tipos de nodos: conceptos (ideas etiquetadas) e ítemes (trozos de información). Los conceptos se relacionan entre ellos mediante asociaciones conceptuales, y los conceptos con ítemes mediante asociaciones funcionales. Una asociación se etiqueta mediante una relación que tiene un nombre y una serie de propiedades asociadas (como ciclicidad, transitividad, etc). Los nodos tienen un conjunto fijo de atributos: los conceptos una etiqueta y los ítemes otros como su identificador, el autor, fecha de creación, etc. El *sistema de navegación* determina como se puede navegar a través de las relaciones conceptuales, y por ultimo el *sistema de aprendizaje* asiste al usuario en su interacción con el sistema adecuando la navegación a las necesidades de información y los conocimientos que va logrando el lector.

Para dotar de capacidad de *evolución* a los sistemas desarrollados el modelo incorpora la figura del Metasistema. Este proporciona al autor un conjunto de acciones evolutivas (ACe) que le permiten modificar los componentes del sistema. Cada ACe tiene un conjunto de restricciones que el Metasistema comprobará para determinar si ésta podrá ser llevada a cabo de forma consistente. En ocasiones un cambio en un subsistema genera la necesidad de nuevos cambios en ese subsistema o en otros. Para ésto el Metasistema realiza de forma automática la propagación interna y externa del cambio.

2.2. Extensiones

Algunas de las ampliaciones realizadas han surgido con el objetivo de aumentar la versatilidad de la herramienta (atributos genéricos), y otras como consecuencia del propio proceso de implantación de los aspectos teóricos del modelo (dominios semánticos).

Mediante el uso de atributos genéricos se permite que tanto los conceptos como los ítemes de información tengan un número y tipo de atributos variable. Por esto proporcionamos mecanismos para definir los atributos que tiene cada tipo de nodo y las propiedades de cada atributo, lo que hace necesaria la inclusión de nuevas acciones evolutivas. Los atributos genéricos permiten personalizar los datos que almacena cada nodo con muy poco esfuerzo, lo cual hace muy flexible la herramienta.

Durante la elaboración de la herramienta también surgió otro concepto, al que llamamos *dominio semántico* (DS). Un DS se refiere a una parcela de conocimiento, caracterizada principalmente por los atributos de los nodos y las

relaciones conceptuales disponibles. Además, en el mismo DS la misma etiqueta tiene siempre el mismo significado. La existencia explícita de los DSs nos permite asociar distintas ECs al mismo DS, y por lo tanto reutilizar la definición de éste.

3. Evolución y Modelo-Vista-Controlador (MVC)

En el patrón de diseño MVC tradicional, la vista y el controlador constituyen el interfaz de usuario para modificar el modelo, de forma que las acciones del controlador cambian directamente el modelo. En nuestro caso, al ser el Metasistema el encargado de modificar el modelo, el controlador debe enviar las acciones evolutivas al Metasistema. Una vez el modelo haya cambiado notificará a sus vistas el cambio para que éstas se actualicen, como indica el patrón MVC. La figura 1 ilustra el patrón descrito en esta sección.

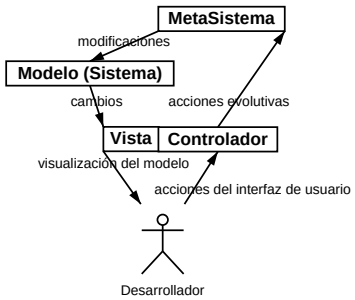

Figura 1. Modelo-Vista-Controlador Evolutivo

4. Implementación

4.1. Lenguaje de Programación, Entorno y Librerías

El lenguaje de programación elegido ha sido Java, debido a su carácter multiplataforma, su integración con la Web, y la disponibilidad de múltiples librerías. Para implementar la edición de estructuras conceptuales nos hemos basado en la librería JGraph [1], que es libre y de código abierto, está hecha en Java, respeta el patrón MVC y está en un estado maduro de desarrollo. De todos modos, el diseño intenta ser lo más independiente posible de la librería de grafos utilizada. Se puede ver más información sobre el entorno de desarrollo utilizado en [4,5].

4.2. Estructura Básica

La clase principal, encargada de almacenar la estructura conceptual (EC) del subsistema de memorización se llama CSm (Conceptual Structure memorization). Ésta se corresponde con el Modelo en el patrón MVC (figura 1). Cada objeto CSm tiene asociado un Metasistema (clase CSmMetaSystem), que permite su evolución. CSmMetaSystem, además de permitir evolucionar a CSm, contiene metainformación sobre la EC, como son las posibles relaciones que puede

tener cada asociación, los atributos que han de tener los conceptos y los ítemes, etc. Así, CSmMetaSystem guarda la información relativa al dominio semántico (DS). Un CSmMetaSystem puede hacer evolucionar varios CSm distintos, siempre y cuando todos estén bajo el mismo DS.

En nuestra herramienta se separa la evolución de la EC y la del DS, de forma que un CSmMetaSystem, que almacena un DS, puede evolucionar también. Y, siguiendo con nuestro paradigma, la evolución de CSmMetaSystem será llevada a cabo por un meta-meta-sistema independiente (CSmMetaMetaSystem), el cual recibirá las acciones evolutivas del desarrollador a través de su controlador. Desde el punto de vista del desarrollador del sistema hipermedia, la vista-controlador que modifica un CSm a través de CSmMetasystem sería un editor de estructuras conceptuales, y la que modifica un CSmMetasystem a través de CSmMetaMetaSystem sería un editor de dominios semánticos. Podemos ver un diagrama en la figura 2.

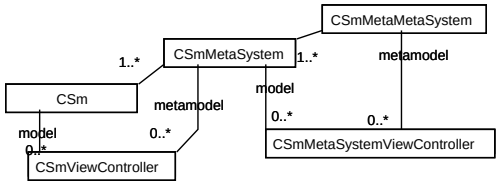

Figura 2. Estructura básica

4.3. La Estructura Conceptual

La EC se ha implementado usando la librería JGraph, que proporciona un interfaz y una clase para los grafos que sirven como modelo de sus vistas. Como se puede observar en la figura 3, utilizamos su implementación (heredando de DefaultGraphModel, etc), pero definimos nuestros propios interfaces para tratar con grafos. El resto de la herramienta utiliza estos interfaces, de forma que para utilizar otra librería de grafos sólo tendremos que implementarlos con la nueva librería. No se han tenido que cambiar mucho las vistas de JGraph, aunque los controladores sí requieren más atención debido a que ahora tienen que enviar acciones evolutivas, y no solamente cambiar el grafo (sección 3).

4.4. Evolución

Para representar las ACe se ha decidido utilizar una clase de Java para cada tipo de acción, siendo un objeto de esa clase una instanciación de la acción (con unos parámetros actuales determinados). De esta forma, para añadir un concepto se creará un objeto de la clase AddConcept que tenga la etiqueta del concepto a crear (único parámetro de la acción evolutiva), y luego se llamará al Metasistema con la acción creada y la EC sobre la que la queremos ejecutar. Se podría haber decidido añadir simplemente un método al Metasistema para cada acción evolutiva, pero hemos decidido tener acciones explícitas porque esto

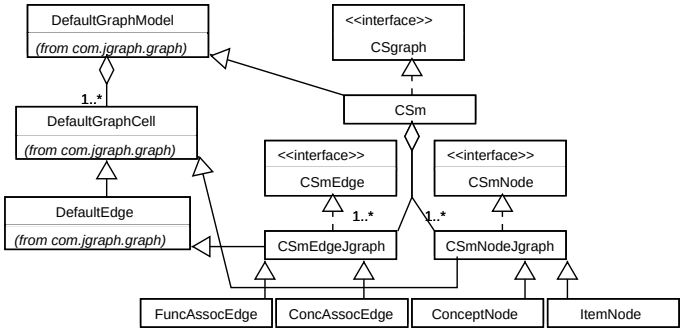

Figura 3. La estructura conceptual (CSm)

facilita la propagación del cambio (enviamos la acción realizada y su resultado), y además haría más fácil llevar una historia estructural del sistema en caso de que se considere necesario. Aunque conceptualmente es el Metasistema el que realiza las acciones evolutivas (para realizarlas tenemos que llamar a un método suyo), el código que las efectúa está en cada ACe. Todo esto nos permite añadir nuevas ACe sin tener que tocar el Metasistema.

En cuanto a las restricciones de las ACe, cada una se implementa mediante una clase de Java que contiene el código necesario para comprobarla. De este modo, cada ACe tiene una lista de las restricciones que hay que comprobar al ejecutarla (es decir, las que se podrían violar como consecuencia de realizar esta acción) (figura 4). Algunas de las restricciones se pueden comprobar antes de realizar el cambio, pero otras son mucho más fáciles de comprobar después de haber cambiado la EC. Esto último nos ha llevado a implementar un mecanismo de modificación de la EC orientado a transacciones, de forma que podamos volver al estado original si las restricciones no se cumplen. Todo el proceso de ejecutar la acción, comprobar restricciones y rechazar la acción, volviendo al estado anterior si no se cumplen lo efectúa el método runOn de la clase abstracta EvAction, por lo que sus subclases normalmente sólo tendrán que definir sus parámetros, inicializar su conjunto de restricciones e implementar el método que verdaderamente hace los cambios (actualRunOn).

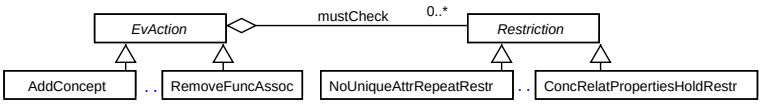

Figura 4. Algunas acciones evolutivas y restricciones

4.5. Atributos genéricos

Debido a que los atributos que contiene cada tipo de nodo pueden cambiar, el formulario de edición del nodo es generado en tiempo de ejecución. Los atributos implementan el interfaz CSmAttribute, mediante el que se pueden definir

las propiedades del atributo. Para cada tipo de atributo se puede obtener su CSmAttributeGui asociado, que se encargará de interactuar con el usuario dentro del formulario de edición del nodo. Es muy sencillo definir los atributos que tendrá cada tipo de nodo, de hecho se permitirá al desarrollador hacerlo a través de un interfaz de usuario en el editor de dominios semánticos. Añadir un nuevo tipo de atributo se reduce a implementar un CSmAttribute y un CSmAttributeGui a partir de los existentes.

5. Conclusiones y Trabajo Futuro

En la etapa inicial del desarrollo del prototipo JSemHP hemos implementado la parte correspondiente al subsistema de memorización, de forma que el editor de ECs pueda ser fácilmente aplicable en otros ámbitos. El modelo SEM-HP ha sido ampliado con atributos genéricos y el concepto de dominio semántico. Además definimos una versión extendida del patrón MVC pensada para sistemas evolutivos. La implementación se basa en la librería de grafos JGraph, aunque el diseño permite utilizar otras. Se hace especial hincapié en el carácter evolutivo del modelo, para lo que nos basamos en muchos conceptos utilizados por el grupo GEDES en otros prototipos, como Metasistema y acciones evolutivas explícitas modeladas mediante clases (que añaden flexibilidad y facilitan la propagación y seguimiento del cambio). Se ha implementado un mecanismo de modificación de ECs basado en transacciones que facilita dejar la EC consistente al rechazar acciones evolutivas. Además, el diseño puede ser ampliado con nuevas acciones evolutivas, restricciones y atributos con facilidad. En este caso se ha comprobado que el desarrollo de una herramienta basada en modelos no estándar requiere más esfuerzo en la fase de diseño, y además se dificulta la integración con software ajeno. No obstante, esta implementación nos permite mostrar que el modelo es factible, y que en el futuro podrá ser útil fuera del mundo académico.

El trabajo futuro pasa por terminar la implementación de los distintos subsistemas de SEM-HP, entrando con más profundidad en temas como la propagación externa del cambio o la interacción con el sistema operativo o un navegador web.

Referencias

1. Alder, Gaudenz. "The JGraph Swing Component". http://www.jgraph.com
2. García-Cabrera, L; Rodríguez-Fórtiz, MJ; Parets-Llorca, J. "Evolving hypermedia systems:a layered software architecture" Journal of Software Maintenance and Evolution: Research and Practice. John Wiley & Sons, Ltd. 17 pgs. (pendiente de publicar)
3. Hurtado-Torres, M.V.; Parets-Lorca J. Evolutionary Information and Decisión Support Systems:An integration Based on Ontologies. Lecture Notes in Computer Science LNCS 2178, pp 146-159, 2001.
4. Java Development Kit. http//java.sun.com
5. Netbeans Integrated Development Environment. http://www.netbeans.com