La Biblioteca de Reutilización GIRO

Carmen Hernández Díez [§]
chernan@infor.uva.es

Miguel Angel Laguna Serrano [§]
mlaguna@infor.uva.es

Francisco J. García Peñalvo [¨]
fgarcia@gugu.usal.es

Resumen

Las bibliotecas de reutilización son elementos fundamentales dentro del proceso de reutilización de software pues sirven como enlace entre el desarrollo para reutilización, cuando se producen los elementos a reutilizar y el desarrollo con reutilización, cuando estos elementos son reutilizados. En este trabajo se presenta la biblioteca de reutilización construida por el grupo GIRO (Grupo de Investigación en Reutilización y Orientación a Objetos), dentro del proyecto Mecano, que da soporte al modelo de componente reutilizable propuesto por el grupo y que es accesible a través de la Web en la dirección http://marte.dcs.fi.uva.es. Además, se plantea el problema de la recuperación en las bibliotecas de software y se avanza la solución propuesta para la biblioteca GIRO.

Palabras clave

Reutilización, Bibliotecas de Software, Repositorio, Elemento Reutilizable, Mecano, Recuperación

1. Introducción

El proceso de producción de un determinado sistema software mediante reutilización sólo tiene sentido si existe un enlace entre el desarrollo para reutilización, donde los elementos software reutilizables de grano fino (*assets* a partir de ahora) son producidos, y el desarrollo con reutilización, donde éstos son utilizados. Esto lleva a la necesidad de contar con un almacén de *assets* que enlace estos dos procesos. Estos almacenes se conocen como repositorios de reutilización, constituyéndose en elementos centrales para el soporte operativo de la reutilización. Aunque inicialmente el concepto de repositorio se corresponde con una simple base de datos para el almacenamiento de *assets*, este ha ido evolucionando hacia el concepto de biblioteca de reutilización propugnado por el DoD (*Department of Defense*) de EEUU y los estándares de la OTAN para reutilización [1], donde la idea básica es que las bibliotecas de reutilización son modelos más aplicaciones o servicios [2].

Los repositorios de reutilización están dirigidos, por tanto, a facilitar la reutilización de componentes a lo largo del ciclo de vida del software para conseguir alcanzar los objetivos de reducción de costes y aumento de la productividad. La razón fundamental para la existencia de un repositorio es proporcionar acceso a los elementos reutilizables

[§] Departamento de Informática. Ed. De Tecnologías de la Información y las Telecomunicaciones. Universidad de Valladolid. Campus Miguel Delibes. 47011, Valladolid (España)

[¨] Departamento de Informática y Automática. Facultad de Ciencias. Universidad de Salamanca. Plaza de la Merced, s/n. 37008, Salamanca (España)

al equipo de desarrollo y de mantenimiento, y soportar la composición de sistemas y el prototipado rápido [3].

Según la perspectiva adoptada por DoD, un repositorio para la reutilización es una base de datos que almacena *assets*. Una biblioteca de reutilización es un repositorio más una interfaz de búsqueda, un esquema de indexación para los *assets* del repositorio y utilidades para gestión de cambios y garantía de calidad de los *assets*. En la actualidad, se hace difícil pensar que las herramientas para el almacenamiento, clasificación y recuperación de *assets* no puedan ser utilizadas a través de la red Internet mediante un servidor *Web*. La utilización de la flexibilidad y el poder que ofrece la tecnología Web debe aprovecharse para crear un entorno que dé soporte al desarrollo con y para la reutilización de forma distribuida.

De esta manera, aparece la noción de la interoperabilidad entre bibliotecas de reutilización. Esta se entiende como la capacidad para intercambiar *assets*, descripciones de *assets* y otras informaciones entre bibliotecas de reutilización. El *Reuse Library Interoperability Group* (RIG) ha elaborado el estándar IEEE 1420.1 [4] para intercambio de información entre bibliotecas de reutilización, también conocido como modelo BIDM (*Basic Interoperability Data Model*). Este modelo de datos describe la información intercambiable y la adopción de una interfaz de usuario, independiente del catálogo, orientada a los navegadores del Web. La interoperabilidad permite separar la actividad de catalogar *assets* de la de almacenarlos, de forma que los catálogos pueden tener referencias a *assets* almacenados en otras bibliotecas. Las bibliotecas ASSET, CARDS y DSRS han implementado esta capacidad de interoperabilidad y evolucionan hacia el concepto de biblioteca virtual.

El problema central de una biblioteca es la recuperación: cómo encontrar en la biblioteca aquellas componentes que pueden ser utilizadas en la construcción de una aplicación particular. Como la recuperación depende de la concordancia entre el *asset candidato* y la *consulta del usuario*, la representación de ambas es una consideración importante, que hay que tener en cuenta cuando se está diseñando la biblioteca, de manera que la elección que se haga en este momento va a condicionar los posibles métodos de recuperación que se puedan construir sobre ella [5].

Este trabajo se estructura de la siguiente manera: El siguiente apartado presenta el modelo de Mecano propuesto por el grupo GIRO, los requisitos detectados para construir la biblioteca, el esquema de la base de datos que la soporta y la arquitectura del sistema que se ha construido. El apartado 3 introduce algunos conceptos sobre recuperación en bibliotecas de software y presenta una propuesta de actuación en este tema. El apartado 4 hace referencia a otros trabajos relacionados. Por último, aparecen las conclusiones y el trabajo que queda por hacer para obtener una biblioteca de reutilización completamente operativa.

2. La Biblioteca de Reutilización GIRO

La base fundamental del modelo de reutilización propuesto por el grupo GIRO [6] está constituida por un elemento reutilizable de grano grueso con soporte simultáneo de varios niveles de abstracción, que recibe el nombre de mecano.

El modelo de reutilización y la estructura de componente reutilizable propuestos están exhaustivamente expuestos y defendidos en [7].

En la Figura 1 se presenta el modelo del repositorio, donde aparece inmerso el modelo de componente reutilizable, el mecano, junto con toda la información necesaria para su clasificación, cualificación, recuperación, mantenimiento y uso.

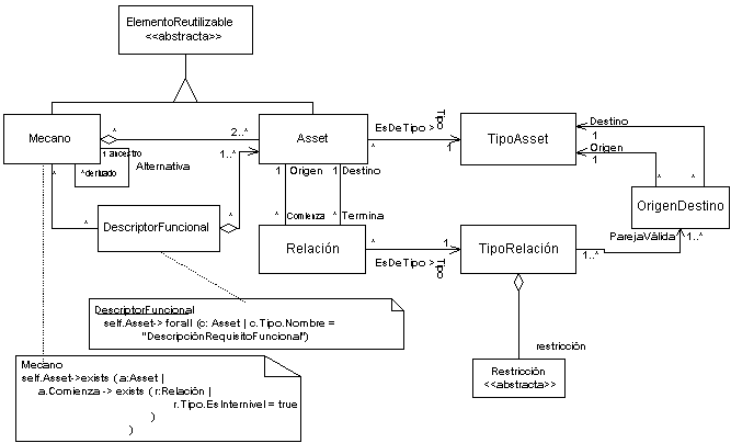

Figura 1. Modelo semi-formal de Mecano [7].

Como ya se ha comentado, el soporte operativo de la reutilización son las bibliotecas de software, de manera que, para hacer efectiva la propuesta del modelo de reutilización que se ha hecho, es necesario contar, inicialmente, con un repositorio que incorpore el concepto de mecano dentro de su estructura, para, posteriormente, crear una biblioteca en la que las facilidades prestadas por la capa del repositorio y las de la capa de servicios se ajusten al modelo establecido y a las reglas asociadas a dicho modelo.

Para definir el conjunto de funcionalidades que necesita soportar la biblioteca se ha asumido el papel del administrador del repositorio y el doble papel que puede jugar el usuario de la biblioteca: suministrador o desarrollador con reutilización, y se han desarrollado los casos de uso asociados a cada situación.

Cabe destacar entre los requisitos funcionales que se han establecido, los siguientes:

- El modelo de datos de la biblioteca debe ajustarse al estandar IEEE 1420.1, citado anteriormente.
- Se debe permitir tanto la *recuperación* de un elemento reutilizable que satisface una determinada petición, como la *inspección* (*browsing*) de cada uno de los elementos de la biblioteca.

En cuanto al ámbito de explotación de la biblioteca se estableció como requisito esencial el acceso a través de Internet, utilizando la interfaz del navegador web como base para realizar este acceso. La satisfacción de este requisito conduce a una arquitectura cliente/servidor. Además, en el interfaz de usuario se emplean grafos que reflejan la estructura del mecano visualmente, de manera que el usuario no sólo introduzca u obtenga el conjunto de *assets* que forma un mecano, sino que pueda ver, y en definitiva, comprender cómo se relacionan entre sí dichos componentes.

Hay que hacer notar que una fuente de información fundamental han sido las experiencias adquiridas durante el uso de un repositorio anterior basado en EUROWARE [8]. En efecto, como primera experiencia en el uso de una biblioteca de reutilización se optó por la adaptación de un motor de repositorios ya existente, EUROWARE, para el soporte de la estructura de mecano. Esta experiencia ha sido fundamental para la detección y obtención de los requisitos que debía cumplir el nuevo repositorio para soportar completamente la estructura y filosofía de reutilización propias de los mecanos.

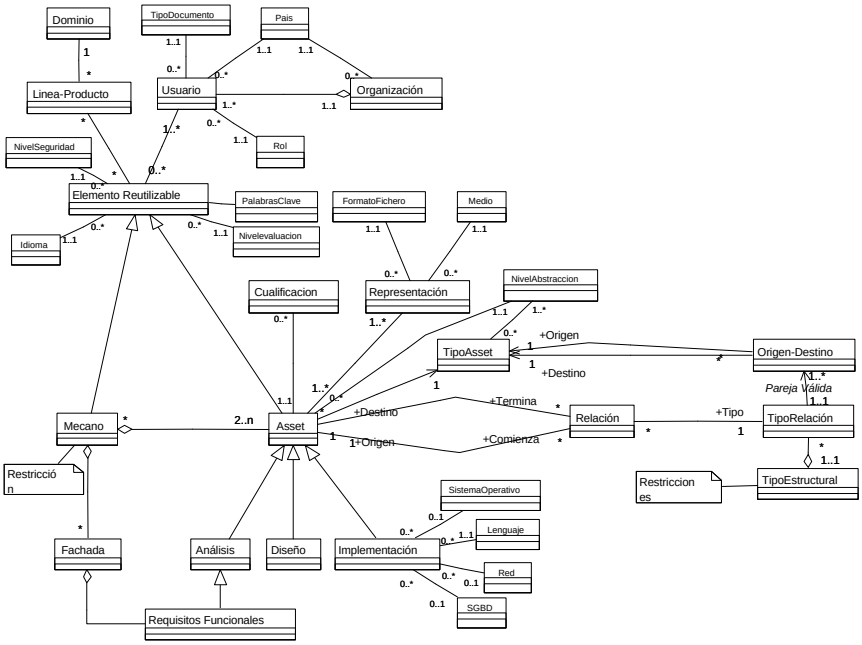

Figura 2. Esquema de la base de datos

2.1.Esquema de la base de datos

El esquema conceptual de la base de datos construida como soporte de almacenamiento de la biblioteca coincide con el modelo de repositorio expuesto en el apartado anterior. El sistema gestor de bases de datos que se ha utilizado es el SGBD objeto/relacional **ORACLE 8i**, para el que se definió el esquema lógico que se muestra en la Figura 2.

Para la implementación de la base de datos se han utilizado tablas objeto-relacionales y los métodos de las clases están introducidos en la base de datos como procedimientos almacenados escritos en Java, de manera que se encuentran empaquetados junto con los datos [9, 10]. El repositorio está actualmente accesible en la dirección http://marte.dcs.fi.uva.es/.

2.2. Arquitectura de la biblioteca de reutilización

Se ha implementado una arquitectura a tres niveles: Cliente, Nivel de Servidor Web y Nivel de la Base de Datos, de forma que un cliente interactúa con la base de datos a través de una capa intermedia que establece las interfaces de los servicios ofrecidos por

el servidor. De esta forma, se pueden cambiar el servidor o el cliente sin que las demás capas se vean afectadas, siempre y cuando se mantengan las interfaces. Toda la información del sistema se almacena, como ya se ha comentado, en la base de datos objeto-relacional **ORACLE 8i**.

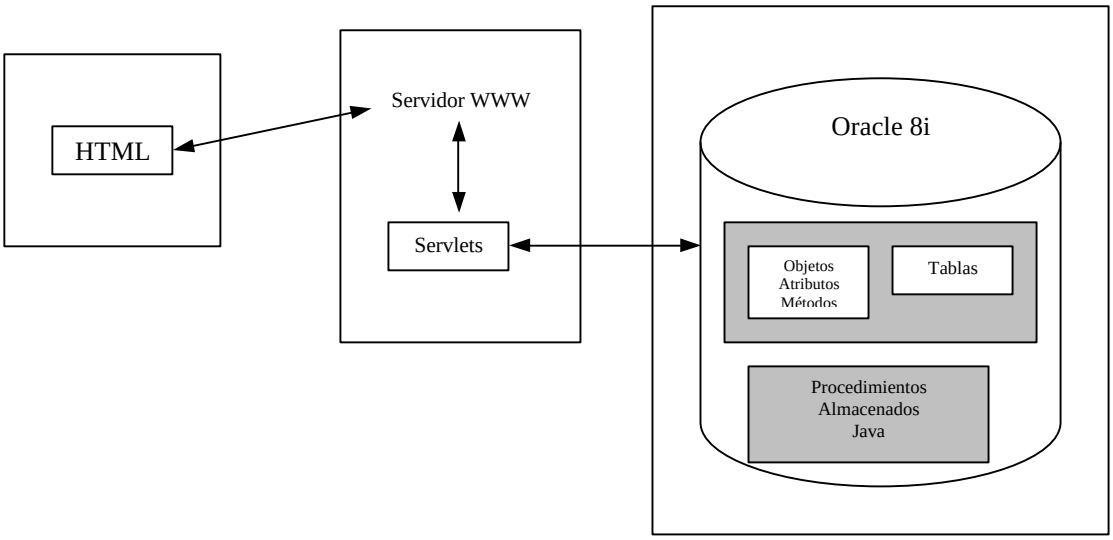

Figura 3. Esquema de la arquitectura de la biblioteca.

Incidiendo en el hecho de que el acceso al repositorio se realiza por medio de un navegador, la comunicación se establece a través de un servidor Web que escucha las peticiones de los clientes (navegadores) y devuelve páginas HTML con la información solicitada. En el nivel del Servidor Web se encuentran los *servlets* que se encargan de la comunicación entre la base de datos y el navegador. Dado que la interfaz de usuario requiere en algunas ocasiones de operaciones gráficas que superan las capacidades de HTML, se ha recurrido, en estos casos, a *applets* Java que suplan estas carencias.

En la Figura 3 se muestra un esquema de la arquitectura cliente/servidor completa que presenta la biblioteca de reutilización GIRO.

2.3. Interfaz de usuario

La interfaz de usuario recoge los elementos básicos que necesita el usuario para interaccionar con la biblioteca de reutilización según sea el rol con el que accede a ella.

Desde el punto de vista de la gestión de mecanos, el usuario va a disponer de una visualización del grafo correspondiente al mecano que está consultando o construyendo. Dicha representación presenta como nodos a los *assets* que forman el mecano. Los distintos niveles de abstracción del *asset* así como alguna otra característica particular se representan en el interfaz a partir de una asignación imagen-color que se ha establecido para cada tipo de *asset*. Igualmente, la semántica y polaridad de las relaciones definidas convenientemente en la propuesta del elemento reutilizable [7, 11] se han representado gráficamente de la misma manera.

Ante esta visualización, el usuario tiene la posibilidad de consultar las propiedades de los elementos del grafo de forma individual, simplemente escogiendo el *asset* o relación de la que desea conocer sus detalles. La información también es presentada en una

forma textual, pero permitiendo una rápido movimiento por los elementos del mecano, a través de visores y barras de desplazamiento.

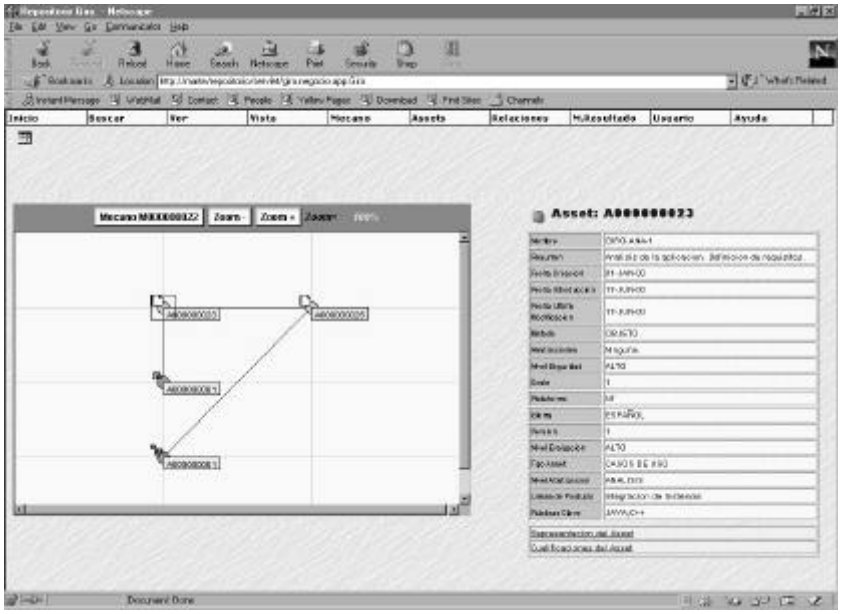

Figura 4. Una ventana de la interfaz de usuario.

La Figura 4 muestra una de las ventanas que aparecen como parte de la interfaz de usuario del cliente.

3. El Sistema de Recuperación de la Biblioteca GIRO

Entre los servicios que se deben incorporar al repositorio, se encuentra el que permite la recuperación y la inspección de los elementos almacenados. La diferencia entre inspeccionar y recuperar es que mientras la última consiste en identificar y extraer *assets* que satisfacen un *criterio de concordancia* (*matching*) predefinido, la primera consiste en inspeccionar *assets* para su posible extracción, sin ningún criterio predefinido [12]. A continuación se exponen algunos conceptos asociados a la recuperación de los *assets*.

Como la recuperación depende de la concordancia entre el *asset* candidato y la *consulta del usuario*, la representación de ambas es una consideración importante. Una vez determinada esa estructura, se pueden distinguir dos aspectos en la actividad de recuperación de *assets*, mediante los cuales se revisa una biblioteca para llegar a identificar los *assets* relevantes: *navegación*, que determina qué *assets* son visitados, y eventualmente en qué orden son visitados; y *concordancia* (matching), que determina la condición bajo la cual un *asset* visitado es seleccionado.

La concordancia consiste en verificar una condición que involucra a un *asset* dado (o su subrogado) para determinar cuando el *asset* debe ser seleccionado. Se distinguen dos tipos de condiciones: la *condición de relevancia* (o *criterio de relevancia*) que es la condición bajo la cual un *asset* es considerado relevante con respecto a la intención de

la consulta; y la *condición de concordancia* (o *criterio de concordancia*) que es la condición bajo la cual un *asset* es seleccionado. La condición de concordancia puede verse como una implementación del criterio de relevancia, pero, en la práctica, estas dos condiciones pueden ser muy diferentes, bien porque el criterio de relevancia sea demasiado complejo de automatizar o porque el subrogado sea demasiado abstracto y no permita una caracterización precisa de los *assets* relevantes.

Se pueden distinguir dos metas de una operación de recuperación de *asset*: *recuperación exacta*, cuyo propósito es identificar los *assets* que son correctos con respecto a la consulta introducida; *recuperación aproximada*, cuyo propósito es identificar a los *assets* que pueden ser modificados con un esfuerzo mínimo para satisfacer la consulta.

En [12] se realiza una revisión de los métodos para almacenar y recuperar software en una biblioteca, en el que se clasifican estos métodos en seis familias: Métodos basados en Recuperación de la Información, Métodos Descriptivos, basados en Semántica Operacional, basados en Semántica Denotacional, Métodos Topológicos y Métodos Estructurales.

Todos estos métodos se basan en una serie de atributos que caracterizan a las bibliotecas de software y que necesariamente se han de tener en cuenta cuando se define una nueva biblioteca: Naturaleza del *asset*, Marco de Uso de la Biblioteca, Representación de la Consulta, Representación del *Asset*, Estructura de Almacenamiento, Esquema de Navegación, Meta de la Recuperación, Criterio de Relevancia y Criterio de Concordancia.

Según la anterior clasificación, y teniendo en cuenta los valores que los atributos del párrafo anterior toman en la biblioteca GIRO, se concluye que ésta cumple los criterios que se han descrito para poder utilizar como método de recuperación algún método topológico; es decir, un método que identifica componentes relevantes minimizando alguna medida de distancia entre ellos y la consulta.

Los métodos topológicos están basados en la premisa de que, dada una consulta que describe alguna característica necesaria, estamos interesados en identificar los *assets* que están más cerca de proporcionarnos esta característica. Estos métodos dependen críticamente de lo que se entiende por *estar cerca de*, que a su vez depende de alguna definición de distancia entre la consulta y el *asset* candidato.

La medida de la distancia que se ha de proponer debería reflejar el alcance de la similaridad entre las propiedades funcionales de la consulta y las del candidato; es decir, medir la *actuación* semejante y, según se establece en [13], debería basarse en la concordancia de las especificaciones (*specification matching*).

4. Trabajos Relacionados

Siguiendo la línea planteada en este trabajo, se presentan algunos métodos topológicos que se han ido proponiendo, sin pretender ser exhaustivos:

En [14] Ostertag, Hendler, Prieto-Diaz y Braun presentan una biblioteca llamada *AIRS* (AI-based Reuse System). Este sistema se basa en un enfoque híbrido, que incluye una enfoque de facetas [15] y de redes semánticas. El método de recuperación se basa en la computación de métricas de similaridad que permiten comparar tanto componentes como paquetes.

En [16, 17] Spanoudakis y Constantopoulos introducen un lenguaje de modelado conceptual (bajo el nombre de *TELOS*) que se adapta muy bien a la representación de artefactos software, específicamente dentro del contexto del análisis y diseño orientado a objetos. El lenguaje soporta la representación de clases objeto, y distingue entre atributos y entidades. Utilizan este lenguaje para representar tanto consultas como *assets*, y definen una medida de la distancia entre consultas y *assets* basándose en un análisis de su representación en TELOS.

Otra biblioteca de software es *Software Information Base (SIB)* desarrollada dentro del proyecto *Ithaca* [18]. Esta biblioteca está organizada según el lenguaje de representación del conocimiento *Telos*; las entradas en SIB son clases Telos [19].

En [20, 21] Penix y Alexander discuten un método basado en especificaciones formales para el almacenamiento y recuperación de componentes software en una biblioteca. Para mejorar la recuperación (*recall*), utilizan una recuperación aproximada cuando la exacta falla, porque no ha encontrado *assets* o ha encontrado demasiados.

5. Conclusiones

En este trabajo se presenta la primera versión de la biblioteca de reutilización GIRO. Esta biblioteca ha sido desarrollada completamente en el seno del grupo de investigación GIRO para soportar de forma nativa el concepto de mecano y del modelo de reutilización que de él se deriva [7]. Esta primera versión se ha centrado en la funcionalidad necesaria para poder poblar el repositorio de una manera eficaz y distribuida, presentando una interfaz de usuario gráfica e intuitiva basada en la metáfora de navegación propia del mundo Web. El nivel de almacenamiento se ha implementado sobre **ORACLE 8i**.

El trabajo futuro se centra en transformar esta primera versión en una verdadera biblioteca en la que se satisfagan todos los servicios detectados como necesarios. En particular, en este trabajo se aborda el servicio de recuperación de los elementos reutilizables, ya se trate de *assets* o de mecanos. De entre los distintos métodos estudiados, se ha optado por un método topológico, por que se ajusta de forma óptima a la estructura del elemento reutilizable que se propugna.

6. Agradecimientos

Este documento ha sido realizado en el seno del grupo **GIRO** (*Grupo de Investigación en Reutilización y Orientación al objeto*) compuesto por miembros del Departamento de Informática de la Universidad de Valladolid, del Departamento de Informática y Automática de la Universidad de Salamanca y del Area de Lenguajes y Sistemas

Informáticos de la Universidad de Burgos. Este trabajo ha sido parcialmente financiado por el proyecto **CICYT TIC2000-1673-C06-05**.

7. Bibliografía

[1] NATO (1992). NATO standard for management of a reusable software component library. Volume 2 (of 3 Documents). NATO Communications and Information Systems Agency (NACISA).

[2] Wallnau, K.C. (1992). "Towards an extended view of reuse libraries". En *Proceedings of 5th Workshop on Institutionalizing Software Reuse (WISR-5)*, Palo Alto, California (USA).

[3] Jiang Guo y Luqi (2000). "A Survey of Software Reuse Repositories". En *7th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS)*, April 3-7, 2000. Napier University, Edinburg.

[4] IEEE (1995). IEEE Standard for Information Technology – Software Reuse - Data Model for Reuse Library Interoperability Data Model (BIDM). IEEE Std 1420.1, IEEE Computer Organization.

[5] Atkinson, S. (1996). "A Formal Model for Integrated Retrieval from Software Libraries". En *Proc. Technology of Object-Oriented Languages and Systems (TOOLS 21)*, pp. 153-167, Prentice-Hall.

[6] García, F.J.; Marqués, J.M.; Laguna, M.A.; Maudes, J.M. (1998). "Estructuras complejas de reutilización: Definición de mecano estático". En las *Actas de las II Jornadas de Trabajo del Grupo MENHIR*. Editor José A. Carsí (Valencia, 19-20 Febrero, 1998):135-141.

[7] García, F.J. (2000). *"Modelo de Reutilización soportado por Estructuras Complejas de Reutilización denominadas Mecanos"*. PhD thesis, Universidad de Salamanca.

[8] García, F.J.; Romay, M.P.; Marqués, J.M.; Crespo, Y. (1999). "Mecanos: Exposición de resultados y líneas de trabajo abiertas en la Reutilización Sistemática del Software". En las *Actas de las II Jornadas Iberoamericanas de la Ingeniería de Requisitos y Ambientes de Software (IDEAS'99)*. Editores Pastor, O., González, C., Trejos, I., Insfrán, E. (Alajuela, 24-26 Marzo, 1999):193-204.

[9] Pulpón, A.; Pérez, E. (2000). "Diseño y Desarrollo de un Repositorio para la Reutilización de Mecanos". Proyecto Fin de Carrera de la Ingeniería Informática de la Universidad de Valladolid. Departamento de Informática (ATC, CCIA y LSI). Dirigido por Valentín Cardeñoso y Carmen Hernández.

[10] Marticorena, R. (2000). "Diseño y Desarrollo de una Interfaz de Usuario para el Manejo de Mecanos en un Repositorio para la Reutilización". Proyecto Fin de Carrera de la Ingeniería Informática de la Universidad de Valladolid. Departamento de Informática (ATC, CCIA y LSI). Dirigido por Valentín Cardeñoso y Carmen Hernández.

[11] García, F.J.; Marqués, J.M.; Laguna, M.A.; Maudes, J.M. (1998). "Influencia de las relaciones entre elementos software reutilizables en la generación de mecanos". En las

Actas de las III Jornadas de Ingeniería del Software (JIS'98). Editores Ambrosio Toval y Joaquín Nicolás Ros (Murcia, 11-13 Noviembre, 1998):155-166.

[12] Mili, H.; Mili, F.; Mili, A. (1998). "A Survey of Software Reuse Libraries". *Annals of Software Engineering*, 5: 349-414.

[13] Rollins, E.J.; Wing, J.M. (1991). "Specifications as Search Keys for Software Libraries". *Proc. 8th International Conference on Logic Programming* (Paris, June 1991): 173-187.

[14] Ostertag, Hendler, Prieto-Diaz, Braun (1992). "Computing Similarity in a Reuse Library System: An AI-based Approach". *ACM Transactions on Software Engineering and Methodology*, 1(3): 205-228.

[15] Prieto-Diaz (1990). "Classification of Reusables Modules". *Software Reusability, Volume I: Concepts and Models*, Frontier Series. ACM Press / Addison Wesley, Reading, Ma.

[16] Spanoudakis, Constantopoulos (1993). "Similarity for Analogical Software Reuse: A Conceptual Modelling Approach". *Proceedings, CaiSE'93, LNCS vol. 685*.

[17] Spanoudakis, Constantopoulos (1994). "Measuring Similarity between Software Artifacts". *International Conference on Software Engineering and Knowledge Engineering*, Jurmala, Latvia.

[18] Ithaca (1993). "Integrated Toolkit for Highly Advanced Computer Applications". Technical Report 2705, EEC-Esprit II Project.

[19] Koubarakis (1989). "TELOS: Features and Formalization". Technical Report KRR-TR-89-1, University of Toronto.

[20] Penix, Alexander (1995). "Design Representation for Automating Software Component Reuse". *Proceedings, 5th International Workshop on Knowledge Bases Systems for the (Re)Use of Software Libraries*.

[21] Penix, Alexander (1996). "Efficient Specification based Component Retrieval". Technical report, University of Cincinnati, Knowledge Based Software Engineering Laboratory, ECECS.