Hacia un catálogo práctico de componentes de F/OSS para la reutilización

Jorge Ramirez, Gustavo Gil, María Laura Massé Palermo

Centro de Investigación y Desarrollo en Informática Aplicada (CIDIA) – Universidad Nacional de Salta
{ramirezj, gsgil, mlmassep}@cidia.unsa.edu.ar

Abstract: El software libre y de código abierto (F/OSS por sus siglas en inglés) ofrece nuevas posibilidades para la reutilización. Para que tal reutilización sea posible, es necesario que los desarrolladores cuenten con herramientas para encontrar componentes entre la variedad de recursos disponibles bajo licencias de F/OSS. Aquí presentamos las características esperadas y los lineamientos que orientan el desarrollo de un nuevo catálogo de componentes libres para la reutilización

Palabras clave: Software libre y de código abierto, reutilización, repositorios de componentes, desarrollo basado en componentes.

El software libre y de código abierto (al que aquí llamaremos F/OSS por sus siglas en inglés) se caracteriza por la distribución de las aplicaciones junto con el código fuente, brindando la posibilidad de modificar el producto. Esto significa que a partir de sus características intrínsecas, el F/OSS pone en juego la posibilidad de adaptar aplicaciones o utilizarlas para desarrollos nuevos

La enorme cantidad de software distribuido bajo licencias de este tipo abre un campo novedoso para la reutilización; al igual que en un mercado de componentes comerciales, los desarrolladores disponen de una gran cantidad de recursos desarrollados por terceros y pasibles de incorporar a nuevos desarrollos[1].

Si bien no existe un único modelo[2], el desarrollo de F/OSS comienza frecuentemente con la publicación de una primera versión y continúa con una evolución en la que participan nuevos desarrolladores y usuarios. Esa forma de desarrollo supone una incidencia muy alta de actividades de mantenimiento, tanto para corregir defectos como para incorporar características nuevas[3]; estas características sugieren la posibilidad de relacionar el desarrollo exitoso de un proyecto de F/OSS con una buena mantenibilidad y, por ende, con cierta facilidad para adaptar y modificar.

Como dice Sametinger[4], el software no puede ser reutilizado a menos que se lo encuentre. Los trabajos de Morisio, Menzies[5][6] y otros coinciden en que los factores decisivos que inciden en el éxito de la reutilización no son de carácter

técnico; no obstante, un problema que puede inhibir la adopción de prácticas de reutilización es la dificultad para encontrar qué reutilizar. La búsqueda de componentes se dificulta [7][4] por la inexistencia de consenso o uniformidad respecto de la forma de describir la funcionalidad de los componentes, la interfaz de los mismos, los requerimientos que atiende, etc. En el caso de la reutilización de F/OSS, el carácter frecuentemente distribuido que adoptan tanto el desarrollo y como la distribución de software, acentúa las dificultades para la recuperación de productos que puedan reutilizarse.

La búsqueda de código reutilizable puede dirigirse hacia los repositorios de F/OSS (Sourceforge, Savannah, entre otros), o utilizar los buscadores orientados a la obtención de código (Koders, Codase).

Los repositorios generales no suelen dar soporte a la búsqueda específica para reutilización; por otra parte, la calidad de la información disponible en los mismos es muy heterogénea[8][9].

Los motores de búsqueda de código se basan principalmente en el contenido textual del código, y no en las características generales del software sobre el que se realiza la búsqueda. La información obtenida mediante estos motores resulta muchas veces insuficiente para determinar las posibilidades de reutilización, dado que suelen devolver muchos resultados que no están relacionados con los objetivos del desarrollador. Esta limitación está relacionada con la indexación basada en texto libre utilizada sobre elementos elementos de naturaleza no lingüística, como el código fuente[10]

En este trabajo presentamos las características de un nuevo catálogo de F/OSS para la reutilización, cuyo desarrollo se encuentra en sus inicios en el ámbito del Centro de Investigación y Desarrollo en Informática Aplicada (CIDIA) de la Universidad Nacional de Salta.

A continuación, presentamos un panorama de los trabajos relacionados con las características y funciones propias de un repositorio de software reutilizable; en la sección siguiente exponemos los roles que intervienen en la utilización de un repositorio; a continuación, presentamos de manera general la metodología de desarrollo y las características de la herramienta a construir; finalmente, presentamos las conclusiones.

Trabajos relacionados

En los años 90' se produjo un auge de trabajos relacionados con la construcción de repositorios de recursos reutilizables[7]. La idea de un mercado de bibliotecas que nació con la ponencia de McIlroy en 1968[11] no se había plasmado en la realidad, y no se había extendido un mercado de donde buscar recursos.

Las bibliotecas (o “librerías” como se traduce con frecuencia) no constituyen el único recurso pasible de reutilización en nuevos desarrollos de software; también se

puede volver a utilizar requerimientos, aspectos del diseño y diversos documentos que se producen durante el desarrollo de un sistema de software[12]. La búsqueda y recuperación de elementos reutilizables implica desafíos característicos que no aparecen en sistemas de búsqueda en otros dominios.

En ese tiempo surgieron propuestas de utilizar una clasificación facetada[13] en lugar de la organización en un árbol de categorías, para facilitar la búsqueda según criterios diferentes.

Como mencionamos, Prieto-Díaz y otros [13] propusieron un esquema de clasificación facetada para los recursos reutilizables. Por ejemplo, las facetas sugeridas en el trabajo de 1987 eran : función, objeto, medio, tipo, área funcional y settings. En el proyecto REBOOT[14], en tanto, se propusieron las siguientes facetas: abstracción (usualmente, los componentes OO pueden caracterizarse por un sustantivo), operaciones, opera sobre, dependencias.

Henninger[10], por su parte, presentó un enfoque evolutivo y adaptativo que se ajuste paulatinamente a las expectativas de los usuarios; el sistema propuesto utiliza una herramienta para detectar software reutilizable, y un sistema denominado CodeFinder que modifica la representación de los componentes almacenados en función de las actividades de búsqueda realizadas efectivamente por los usuarios.

Ya en el 2000, Meling y otros propusieron[15] el desarrollo de un Administrador de Descripciones de Componentes (CDM, Component Description Manager) para superar las limitaciones de la clasificación basada en lenguaje natural o en un conjunto limitado de palabras clave.

En C.R.U.I.S.E.(Component Reuse In Software Engineering)[12], los autores plantean una serie de requerimientos que deberían considerarse al construir un repositorio de componentes para la reutilización. Algunas características propuestas que resultan relevantes para nuestro trabajo son:

- **Búsqueda**: Debe proveer mecanismo de búsqueda que permitan a los usuarios encontrar los recursos necesarios. Debería permitir la búsqueda de texto libre, por palabras claves, clasificación facetada (de acuerdo con la propuesta de Prieto-Díaz [13]).
- **Navegación**: Debe ofrecer vías de navegación sencillas
- **Generación de informes**: Ofrecer informes acerca de la utilización del repositorio, permitiendo -por ejemplo- conocer cuáles son las búsquedas más comunes, los recursos descargados con mayor frecuencia, la participación de los usuarios en el repositorio, etc.
- **Notificaciones a los usuarios**: Posibilidad de informar a usuarios registrados cuando se agreguen nuevos activos, o situaciones relacionadas con sus perfiles.
- **Administración de versiones**: Llevar el registro de las diferentes versiones de los activos o recursos que gestione el repositorio.

- **Servicios de retroalimentación**: Permitir que los usuarios provean retroalimentación sobre los activos que utilizan.
- **Soporte para múltiples repositorios**: debe poder gestionar activos de diversos repositorios

Lucredio y otros elaboraron en 2004 un informe[7] sobre la investigación en el área de la búsqueda y recuperación de componentes. Estos autores revisan los aportes realizados por diferentes investigadores y proponen un conjunto de requerimientos que debería cumplir un mecanismo de búsqueda y recuperación de componentes.

En su trabajo de tesis, Bhatia propuso un modelo de etiquetado en el que los propios usuarios de los repositorios elaboren clasificaciones de manera colectiva y distribuida[16], tomando como referencia el método de indexación social conocido como “folcsonomías”. Este enfoque tiene la ventaja de que la propia comunidad vaya estableciendo el léxico común mediante la utilización real del repositorio y el etiquetado colaborativo. La propuesta de este autor contempla la posibilidad de asignar diferentes niveles de visibilidad a los elementos que incorporen los usuarios, posibilitando la utilización de la misma herramienta dentro del ámbito de una institución o empresa en particular. La estructura de la aplicación sugerida por Bhatia consta de un módulo encargado de la incorporación de información de recursos (mediante la recuperación de información de la Web o la carga por parte de los usuarios) y otro que se ocupa de sostener el etiquetado colaborativo

Ayala y otros[17] analizaron el mercado de componentes F/OSS y su interacción concreta con un conjunto de pequeñas y medianas empresas de desarrollo; en los casos analizados, los autores encontraron que los reutilizadores experimentados buscaban componentes en los sitios o repositorios que conocen, mientras que los menos experimentados se inclinaron por utilizar motores de búsqueda de carácter general (como Google) más que buscadores específicos de código.

Diferentes roles en repositorios

Un repositorio que permita la búsqueda y selección de F/OSS será necesariamente de carácter externo para los desarrolladores. Como vimos al principio, los repositorios de proyectos F/OSS existentes no están orientados a la reutilización; es necesario, entonces, superar esta brecha mediante herramientas que hoy no están disponibles.

Típicamente, un repositorio para la reutilización supone la existencia de tres roles[18]:

- producción de componentes
- administración del repositorio
- consumo o utilización de componentes

En el caso que nos ocupa, la producción se realiza según la lógica propia del desarrollo F/OSS, lo que implica la inexistencia de planes centralizados y pautas de aplicación general para la documentación de los productos. La producción de F/OSS (muchas veces a cargo de comunidades) difícilmente incorpore en sus procesos de forma sistemática la publicación, la documentación y la notificación a los reutilizadores de las novedades de los productos que desarrollan; esa tarea deberá ser asumida, en la medida de lo posible, por el propio repositorio.

El acceso a los productos de F/OSS también ofrece múltiples orígenes posibles; entre ellos podemos mencionar a los repositorios que albergan proyectos de este tipo (como SourceForge), los sitios Web de los productos y las distribuciones de GNU/Linux.

La heterogeneidad en el alojamiento de los potenciales componentes y la heterogeneidad de la información disponible para cada uno de ellos, nos llevan a plantear la creación de un catálogo, antes que un repositorio propiamente dicho. Los usuarios a quienes estaría destinado principalmente esta aplicación son desarrolladores, que accederían a este catálogo a través de la Web.

Esas características sugieren la conveniencia de adoptar un método de desarrollo abierto y participativo, característico de muchos proyectos F/OSS[19].

Construcción del catálogo

En las secciones anteriores expusimos las características que debería cumplir un repositorio de software reutilizable, según el estado del arte sobre el tema.

El catálogo que empezamos a desarrollar deberá incluir información general de cada componente, su evolución (la historia del producto), la o las licencias bajo las cuales se distribuye, comentarios de usuarios que lo utilizaron en sus proyectos y ejemplos de utilización e integración. En principio, nos interesa una aplicación orientada sólo a la búsqueda y recuperación de componentes F/OSS

Un objetivo central del desarrollo es el de conformar paulatinamente una comunidad de usuarios/desarrolladores. Este rasgo requiere de métodos que permitan la evolución del producto.

Los diferentes roles o funciones mencionados más arriba, así como la estructura de las aplicaciones que sostienen los repositorios evolutivos relevados anteriormente[16] [10] nos sugieren separar dos funciones: la incorporación a la base de datos de la representación de componentes, y las tareas de indexación y etiquetado colaborativo.

En la primera etapa del desarrollo nos planteamos reutilizar un spider o motor de búsqueda F/OSS,; la experiencia de utilización, reutilización y comparación entre los candidatos se incorporará a la información disponible en el catálogo y permitirá establecer un conjunto inicial de etiquetas. Algunas alternativas F/OSS que están en evaluación al momento de escribir estas líneas son: sphider, phpDig, php Crawler, etc.

A partir de la información recolectada por el motor seleccionado, y de la experiencia de búsqueda y evaluación del propio motor, se conformará una base de datos inicial, estableciéndose un pequeño conjunto de etiquetas iniciales, que podrán ser modificadas por la comunidad.

Para soportar el etiquetado colaborativo, también nos planteamos evaluar un conjunto de productos alternativos, de modo que el mismo proceso permita incorporar datos sobre los componentes que se evalúen.

El desarrollo, entonces, seguirá un modelo de prototipado evolutivo; el crecimiento del sistema se vinculará a su eventual utilización y a la conformación de una comunidad de desarrolladores que la utilicen, de un modo similar al modelo de bazar postulado por Raymond[19]

Para controlar la evolución del sistema, proponemos la revisión periódica de las etiquetas y la evaluación de la precisión y la exhaustividad, tomando un conjunto de consultas típicas realizadas por los usuarios.

Los comentarios de los usuarios también constituyen una fuente de información sobre los componentes; el sistema propuesto deberá brindar la posibilidad de realizar búsquedas sobre los aportes de los usuarios.

Se realizará una evaluación periódica de la precisión y la exhaustividad a partir de búsquedas relevantes para nuestra institución, de modo de promover modificaciones futuras en el catálogo.

Las búsquedas que se realicen en ocasiones devolverán varios componentes candidatos; un aspecto que podría ser de ayuda al momento de preferir un componente por sobre otro es la facilidad de adaptarlo, modificarlo o integrarlo a un nuevo desarrollo. Tomando en cuenta este aspecto, y vinculado a la actividad de investigación que llevamos adelante en la UNSa[20][21], nos planteamos incorporar indicadores de la mantenibilidad[22], de modo que el usuario ordene los resultados según esos indicadores

Conclusiones

La necesidad de disponer de un catálogo que permita encontrar software libre y de código abierto desde el punto de vista de su potencialidad para reutilizarse, nos llevó a indagar sobre las características que debería tener esa herramienta.

La herramienta que proponemos atiende una necesidad de nuestro centro de investigación y -potencialmente- de utilidad para muchos desarrolladores, el modelo de desarrollo distribuido caracterizado como “bazar” por Raymond[19] aparece como particularmente adecuado. En ese sentido, citando el mismo ensayo de Raymond, la propuesta apunta a “rascarse una picazón” típica del desarrollador de F/OSS.

El relevamiento bibliográfico realizado ofrece una guía para construir la herramienta buscada; la viabilidad de ese desarrollo se vincula a los fundamentos del Software Libre

La exposición de esta idea en ámbitos vinculados al software libre apunta también a impulsar el crecimiento futuro de la herramienta.

La evolución y, en definitiva, la viabilidad del producto cuyo desarrollo hemos comenzado, estará ligada a la conformación de una comunidad en torno al proyecto, siguiendo un modelo basado en las necesidades de los usuarios antes que en el modelo de negocios de una organización en particular.

Referencias:

1.-Hissam, S.A.; Weinstock, C.B.. **Open Source Software: The Other Commercial Software**. En *In 1st Workshop on Open Source Software at ICSE*. 2001.

2.-Scacchi, W.; Feller,J.; Fitzgerald,B.; Hissam,S.A.; Lakhani, K.: Understanding Free/Open Source Software Development Processes, *Software Process: Improvement and Practice*, 11, 2006.

3.-Samoladas, I.; Stamelos,I.; Angelis,L.; Oikonomou, A.: Open source software development should strive for even greater code maintainability, *Commun. ACM*, 47, 2004.

4.- **Sametinger, J.**: *Software engineering with reusable components*, .Springer-Verlag New York, Inc., 1997.

5.-Morisio, M.; Ezran,M.; Tully, C.: Success and Failure Factors in Software Reuse, *IEEE Trans. Softw. Eng.*, 28, 2002.

6.-Menzies, T.; Di Stefano, J.S.: More Success and Failure Factors in Software Reuse, *IEEE Trans. Softw. Eng.*, 29, 2003.

7.-Lucrédio, D.; Prado,A.F.D.; Almeida, E.S.D.. **A Survey on Software Components Search and Retrieval**. En *EUROMICRO*. 2004.

8.-Howison, J.; Crowston, K.. **The perils and pitfalls of mining SourceForge**. En *Proc. of Mining Software Repositories Workshop at the International Conference on Software Engineering (ICSE)*. 2004.

9.-Rainer, A.; Gale, S.. **Sampling Open Source Projects from Portals: Some Preliminary Investigations**. En *METRICS '05: Proceedings of the 11th IEEE International Software Metrics Symposium*. 2005.

10.-Henninger, S.: An Evolutionary Approach to Constructing Effective Software Reuse Repositories, *ACM Trans. Softw. Eng. Methodol.*, 6, 1997.

11.-McIlroy, M.D.: «*Mass Produced Software Components*», en *Software Engineering Concepts and Techniques*, ,NATO Science Committee, 1968.

12.- **de Almeida, E.S.; Alvaro,A.; Garcia,V.C.; Cordeiro Pires,J.; Burégio,V.A.; Nascimento,L.M.; Lucrédio,D.; Lemos Meira, S.**: *C.R.U.I.S.E: Component Reuse in Software Engineering*, 1st .C.E.S.A.R e-book, 2007.
13.-Prieto-Díaz, R.: Implementing faceted classification for software reuse, *Commun. ACM*, 34, 1991.
14.-Sindre, G.; Conradi,R.; Karlsson, E.: The REBOOT approach to software reuse, *Journal of Systems and Software*, 30, 1995.
15.-Meling, R.; Montgomery,E.J.; Ponnusamy,P.S.; Wong,E.B.; Meh, D.. **Storing and Retrieving Software Components: A Component Description Manager**. En *in Proc. of the 2000 Australian Software Engineering Conf*. 2000.
16.-Bhatia, K.. Collaborative Tagging for Software Reuse. *Computer Science \& Engineering Department, Thapar Institute of Engineering \& Technology, Deemed University*. 2006.
17.-Ayala, C.P.; Hauge,O.; Conradi,R.; Franch,X.; Li,J.; Velle, K.S.. **Challenges of the Open Source Component Marketplace in the Industry**. En *OSS*. 2009.
18.- Apperly, H.. **Component-based software engineering**. In . Heineman, George T. and Councill, William T.. Vol. .2001.
[19] RAYMOND, E.. The Cathedral and the Bazaar. http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.
20.-Ramirez, J.; Gimson,L.; Gil, G.. **Evaluación de la Evolución del Diseño en F/OSS:un Caso de Estudio**. En *VII Workshop Ingeniería de Software - XVI Congreso Argentino de Ciencias de la Computación*. 2010.
21.-Ramirez, J.; Gil,G.; Romero,G.; Gimson, L.. **Indicadores de la Utilización del Bug Tracker en Proyectos F/OSS.**. En *XV Congreso Argentino de Ciencias de la Computación*. 2009.
22.-Ramirez, J.; Gil, G.: Mantenibilidad y Evolutividad en el Software Libre y de Código Abierto, 2010. Trabajo Final Integrador para la obtención del título de Especialista en Ingeniería de Software, Universidad Nacional de La Plata, Argentina-