Publicación y Recuperación de Aspectos Reutilizables

Vidal Graciela y Casas Sandra

GISP - Instituto de Tecnología Aplicada
Universidad Nacional de la Patagonia Austral
Campus universitario Piloto Lero Rivera s/nro. Río Gallegos Santa Cruz
vidalgracielaunpa@gmail.com, sicasas@yahoo.com

Resumen

El Desarrollo de Software basado en Componentes posibilitó la creación de repositorios de software. En los repositorios es posible publicar, recuperar y reusar componentes. En los repositorios actuales se han identificado requerimientos no funcionales pero no son presentados explícitamente como aspectos, la abstracción central del Desarrollo de Software orientado a Aspectos. Por lo tanto los aspectos son requeridos para ser reutilizados, es decir que son publicables y recuperables. Los métodos de publicación y recuperación actuales serán analizados con el fin de adaptar o extender el más apropiado para la especificación de aspectos.

Palabras clave: reusabilidad, repositorio de software, componentes, AspectJ.

Contexto

Este trabajo forma parte del proyecto denominado "Usabilidad & AOP: desarrollo y evaluación de un framework de dominio"(29/A315) que se llevará a cabo en el período 2014-2015 por el grupo de investigadores del Instituto de Tecnología Aplicada, financiado por la Universidad Nacional de la Patagonia Austral, Unidad Académica Río Gallegos.

Introducción

Un repositorio de software es una librería en la cual se almacenan componentes software reutilizables y debe ofrecer una categorización de estos artefactos [1]. Para que su utilización sea efectiva, el usuario debe comprender claramente su contenido y luego determinar si sus requerimientos serán cubiertos por el mismo. El Desarrollo de Software basado en Componentes (DSBC) [2][3] ha posibilitado la creación de repositorios de software donde publicar, recuperar y reusar componentes.

Un componente de software reutilizable puede definirse como "cualquier componente desarrollado específicamente para ser utilizado en más de un contexto"[4]. Para que un componente sea almacenado en un repositorio se debe proporcionar su información general, información relacionada a su funcionalidad, restricciones, requerimientos y la documentación para su posterior integración y/o especialización.

Basándose en esta información pueden ser categorizados, lo cual es un proceso no trivial de gran importancia. Una buena especificación y categorización permitirá a los usuarios, acceder a los componentes que requieran, cuando sea necesario. Los componentes pueden ser especificados por medio de esquemas facetados [5], ontologías [6][7][8], frameworks[9], taxonomías[10], métodos topológicos[11] [12][13] y estructurales[14].

El Desarrollo de Software Orientado a Aspectos (DSOA)[15][16][17] es un paradigma propuesto para encapsular Requerimientos No Funcionales (RNF) [18][19], también conocidos como cross-cutting concerns [20]. Los RNF (por ejemplo logging, sincronización, replicación, etc.) entrecruzan la funcionalidad principal de un sistema, generando el denominado código disperso y enmarañado, el cual presenta varias dificultades, como por ejemplo, su mantenimiento. Un aspecto es la abstracción central del DSOA, el cual se compone de dos partes modulares: pointcuts y advices. Los advices son similares a los métodos, debido a que son fragmentos de código que serán incorporados al dominio en algún momento (after, before, around); los pointcuts son las expresiones que establecen ante qué eventos y condiciones estos fragmentos se ejecutarán, por ejemplo la llamada a un determinado método [15].

Los enfoques mencionados promueven el reuso de software, permitiendo alcanzar el desarrollo rápido de aplicaciones, mejorando la calidad y el rendimiento de los desarrolladores, por lo que resulta necesario que los repositorios de software incluyan aspectos.

Para determinar si los aspectos pueden ser almacenados y recuperados desde un repositorio, inicialmente se analizarán los mecanismos actuales destinados a la especificación, publicación y recuperación de componentes software.

El presente trabajo se enfoca en la adaptación y/o extensión de alguno de estos mecanismos aplicado a aspectos. Cualquiera sea el método de especificación seleccionado, deberá incluir la información relacionada a conceptos dinámicos y estáticos, propios de los aspectos, permitiendo su clasificación dentro del repositorio, como así también definir, diferentes niveles de reuso. Este último es un detalle muy importante, los aspectos poseen una estructura sobre la cual, es posible reutilizar el aspecto, un pointcut y/o un advice, dado que la definición de cada elemento estará disponible de forma individual.

Líneas de Investigación, Desarrollo e Innovación

El reuso o reutilización de aspectos ha sido abordado por diferentes trabajos en el campo de la investigación, a continuación se expondrán brevemente algunos de estos enfoques.

En [31] el reuso se enfoca en las dependencias entre aspectos, definiéndose una clasificación de acuerdo a la manera en que interactúan unos con otros y a la forma en que se dispara su funcionalidad. El aspecto se define en términos de los servicios que provee, los que requiere y los que puede eliminar sobre otros aspectos.

En algunos lenguajes orientados a aspectos como AspectJ[32] se utiliza la herencia como mecanismo para implementar el reuso, sin embargo presenta algunos problemas. Para solucionarlo se han definido cinco reglas y se considera el desarrollo de al menos tres aspectos para llevar a cabo el reuso [33].

Una diferencia entre objetos y aspectos se encuentra en la dependencia que poseen con respecto a otros componentes de una aplicación. Los objetos bien definidos tienen limitado alcance con mínima dependencia y alta cohesión. Los aspectos son válidos bajo ciertas condiciones de contexto [34], se requiere comprobar la compatibilidad y los requerimientos que definen su contexto de uso. La utilización de pre y post condiciones de la programación de diseño por contratos son una alternativa a utilizar para la especificación de un aspecto reusable.

En [35] se aborda el reuso sobre propiedades que aportan calidad a una aplicación y no sobre la funcionalidad como es habitual. Estas propiedades, son difíciles de identificar, clasificar y catalogar, por lo que no es suficiente basarse en su taxonomía[36][37].El punto de partida del artículo es enfocarse en los RNF [18][19], ingeniería de requerimientos orientada a objetivos[38], programación orientada a aspectos[15], reuso de software[39] y gestión de calidad.

Sin embargo ninguno de estos enfoques propone el reuso de un aspecto, como un módulo alojado en un repositorio en el que puede ser publicado y recuperado.

En [21] se llevó a cabo una exploración sobre un conjunto de repositorios de software [22][23][24][25][26] en los cuales se confirma la presencia de RNF, lo que demuestra que son requeridos para ser reutilizables, por lo tanto son publicables y recuperables, sin embargo no están modularizados como aspectos. Para que los aspectos puedan ser almacenados, es fundamental definir qué información debe proporcionar su especificación para su publicación y recuperación. Existe información común tanto para componentes software como para aspectos, pero estos últimos poseen ciertos conceptos que lo diferencian notablemente. Estos conceptos son dinámicos (pointcuts y advices) y estáticos (herencia e introducciones) y deben estar incluidos en la especificación del aspecto.

Resultados y Objetivos

El objetivo de este trabajo es definir y desarrollar mecanismos que permitan la especificación de aspectos reusables. Luego se desarrollará una herramienta estilo "prueba de conceptos" que implemente esta especificación, permitiendo publicar y recuperar estos módulos sobre un repositorio.

Objetivos Específicos

- Adaptar o extender mecanismos estandarizados de componentes para aspectos.
- Desarrollar un esquema que permita especificar y clasificar aspectos.
- Desarrollar una herramienta estilo prueba de conceptos que soporte el esquema
- Desarrollar y ejecutar un plan de validación para comprobar que los aspectos pueden ser almacenados y recuperados.

Al finalizar este trabajo debe ser posible: almacenar, clasificar, buscar y recuperar aspectos reusables en un repositorio, basándose en la especificación y mecanismos definidos. Esta especificación conjuntamente a una ontología o clasificación facetada, darán soporte a una herramienta sobre la cual se deberá probar lo objetivos definidos.

Posteriormente el usuario debería ser capaz de acceder a los aspectos categorizados, realizar consultar, seleccionar y recuperarlos para ser reutilizados en diferentes aplicaciones.

Formación de Recursos Humanos

El equipo de investigadores está conformado por docentes de la Universidad Nacional de la Patagonia Austral, estudiantes de la Licenciatura en Sistemas y de la Maestría en Informática y Sistemas que dicta la universidad. En particular este proyecto incluye una tesista de la Maestría en Informática y Sistemas de la UNPA.

Referencias

1. Bakshi, A. Bawa S., Development of a software repository for the precise search

and exact retrieval of the components, International Journal of Advanced Research in Computer Science and Software Engineering 3 (2013), no. 8, pp.1116-1126.
2. González R., Torres. M. Algunas Implicaciones del Desarrollo Basado en Componentes. paper. Bogotá, Colombia: Departamento de Ingeniería de Sistemas Universidad Javeriana; (2005).
3. Montilva J.C., Arapé N., Colmenares J.A, "Desarrollo de software basado en componentes," IV Congreso de Automatización y Control, Maracaibo , Mérida - Venezuela, (2003) pp. 9
4. Kalathipparambil J. and Vasantha, A Mixed Method Approach for Efficient component retrieval from a component repository. Journal of Software Engineering and Applications, (2011).
5. Prieto-Diáz, R.: Implementing Faceted Classification for Software Reuse. In: Communications of the ACM 34 (1991) .
6. Happel H. and Seedorf S.. Applications of Ontologies in Software Engineering Lindeberg, T., feature detection with automatic scale selection. International Journal of Computer Vision, 30 (2004) 77-116.
7. Dragan G., Nima K., Milan M., Ontologies and Software Engineering (2008).
8. Collaborative Software engineering chapter 6: Application of Ontology in Collaborative Software Development, Springer-Verlag, (2010).
9. Overhage S. Towards a Standardized Specification Framework for Component Development, Discovery, and Configuration. Augsburg University. Dept. of Application Engineering and Business Information Systems (2003).
10. Maninder S., Shivani, G., "Identifying asset type for various reusable component storage/retrieval methods," Proceedings of National Conference on Challenges & Opportunities in Information Technology - RIMT-IET, Mandi Gobindgarh, (2007) pp.38-41
11. G. Forbes, McCartan C., Ruairi O'Donnell, Niall S. and Leon R. The integration of information retrieval techniques within a software reuse environment. Department of Information Science, Strathclyde University, 26 Richmond St., Glasgow, G1 1XH, UK (2000).
12. Girardi, M.R. Classification and Retrieval of Software through Their Description in Natural Language, Technical Report 2782, University of Geneva, Geneva, Switzerland (1995).
13. Pahl, C. Ontology-based description and Reasoning for Component.-based Development on the Web en Proceedings of Specification and Verification of Component Based Systems (SAVCBS'03) – ACM SIGSOFT Symposium on the Foundations of Software Engineering Workshop, Sept. 1-2, 2003, Helsinki,Finland, ACM, (2003) pp. 84- 87.
14. Klein, M. and A. Bernstein. Searching for Services on the Semantic Web Using Process Ontologies. The First Semantic Web Working Symposium. Stanford, CA, USA. (2001)
15. Kiczales G., Lamping J., Mendhekar A., et al. Aspect-Oriented Programming. In: Springer-Verlag, editor. Finland.: European Conference on Object-Oriented Programming; (1997). pp25.
16. J. a. T. Brichau , D'Hondt, "Aspect-oriented software development," 2005, pp. 20.
17. Brichau J., Theo D'., Rashid A., Mezini, M., Haupt M., Fabry J. Chitchyan R., Truyen E., Joosen W., Garcia A., Jackson A., Fuentes L., Pinto M., Tekinerdogan B., Krechetov I., Katz, Sh. Curriculum Fundamentals of AOSD. Integración: AOSD – EUROPE.(2006).
18. Mylopoulos, J., Chung, L., Nixon, B.: Representing and using nonfunctional requirements: A process-oriented approach. IEEE Trans. Softw. Eng. 18 (1992) pp.483–497.
19. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishing (2000).
20. Filman, R., et al., Aspect-Oriented Software Development. Addison-Wesley (2004).
21. Vidal G., Casas S. y Marcos C. Exploración de repositorios de softwa-

re y análisis de potenciales extensiones a aspectos. Aprobado por Resolución 0705/13-R-UNPA.ISSN 1852-4516 disponible en http://ict.unpa.edu.ar. (2013)
22. http://uima.apache.org/ (2014)
23. http://www.sourcecodeonline.com (2014)
24. http://sir.unl.edu/php/index.php (2014)
25. http://www.vclcomponents.com/ (2014)
26. http://www.netlib.org/ (2014)
27. Nedhal A., Al Saiyd, Intisar A. Al Said, Takrori AHA. Semantic-Based Retrieving Model of Reuse Software Component: Computer Science Department (2010).
28. Heineman, G., Councill, W, Component-Based Software Engineering, Addison Wesley, (2001).
29. Pressman R., Software Engineering A Practitional's Approach, Sixth Ed., McGraw Hill, (2005).
30. Kuljit Kaur PK, Jaspreet Bedi, and Hardeep Singh. Towards a Suitable and Systematic Approach for Component Based Software Development. World Academy of Science, Engineering and Technology (2007).
31. Kienzle J., Yu Y., Xiong J. On Composition and Reuse of Aspects. School of Computer Science McGill University. Montreal, QC H3A 2A7. Canada(2003)
32. Sitio web de AspectJ www.eclipse.org/aspectj (2014)
33. Hanenberg S., Rainer U., "Using and reusing aspects in aspectj," Institute for Computer Science - University of Essen, Germany, (2001) p. 6.
34. D. Wampler, "The challenges of writing reusable and portable aspects in aspectj: Lessons from contract4j," Aspect Research Associates and New Aspects of Software, (2006) pp.9.
35. Sampaio do Prado Leite J., Yu Y., Liu L., Yu E. and Mylopoulos J. Quality-Based Software Reuse. Departmento de Informatica, Pontifícia Universidade Católica do Rio de Janeiro, RJ 22453-900, Brasil. Department of Computer Science, University of Toronto, M5S 3E4 Canada. School of Software, Tsinghua University, Beijing, 100084, China. (2005)
36. Sommerville, I.: Software Engineering, 4th Ed. Addison-Wesley (1992)
37. Boehm, B.W., Brown, J.R., Lipow, M.: Quantitative evaluation of software quality. In: ICSE, International Conference on Software Engineering, IEEE Computer Society Press (1976) pp. 592–605.
38. Mylopoulos, J., Chung, L., Yu, E.: From object-oriented to goal-oriented requirements analysis. CACM 42 (1999) pp.31–37
39. Krueger, C.: Software reuse. ACM Computer Survey 24 (1992) pp.131–183.