Desarrollo Basado en Reuso

Alejandra Cechich, Agustina Buccella, Andrés Flores, Gabriela Aranda, Nadina Martínez Carod, Juan Luzuriaga, Rodolfo Martínez, Marcelo Moyano, Rafaela Mazalu, Adriana Martín, Martín Garriga
Grupo de Investigación en Ingeniería de Software del Comahue (GIISCo)
http://giisco.uncoma.edu.ar
Departamento de Ingeniería de Sistemas – Facultad de Informática
Universidad Nacional del Comahue
Buenos Aires 1400, (8300) Neuquén
Contactos: {acechich, abuccel, afores}@fai.uncoma.edu.ar

Resumen

Hoy día, el reuso de sistemas de aplicaciones es a menudo la técnica más efectiva. Implica la reutilización de activos de grano grueso que pueden ser configurados para crear un nuevo sistema. En general, existen dos tipos de reuso de aplicaciones: la creación de nuevos sistemas integrando productos (*componentes*, *servicios*) existentes y el desarrollo de *líneas de productos*. Para desarrollar sistemas utilizando productos se tienen que realizar varias elecciones de diseño; por ejemplo, ¿qué productos ofrecen la funcionalidad adecuada? ¿cómo se intercambian los datos? ¿qué características de un producto se usarán realmente? También surgen diversos problemas a la hora de integrar esos productos; por ejemplo, problemas con la interoperabilidad, control sobre la evolución del sistema o soporte de los vendedores. Por otra parte, las líneas de producto software se diseñan para ser reconfiguradas. Esa reconfiguración puede implicar añadir o eliminar componentes del sistema, definir parámetros y restricciones, e incluir conocimiento de los procesos de negocios. Estos problemas han sido abordados desde diversas perspectivas, incorporando por ejemplo técnicas y herramientas basadas en conocimiento que exploran repositorios para reuso. Sin embargo, la reutilización efectiva de activos (recursos y servicios) entre las organizaciones requiere tratamientos que son materia de investigación en ámbitos complejos.

Palabras Clave: Líneas de Producto Software, Desarrollo basado en Componentes y en Servicios, Gestión de Conocimiento.

Contexto

La línea presentada se inserta en el contexto de los siguientes proyectos y acuerdos de cooperación:

- UNCo 04/E072 – Identificación, Evaluación y Uso de Composiciones Software. Acreditado por la Universidad Nacional del Comahue
- PAE-PICT-2312: Métodos y Herramientas para Sistemas masivamente Distribuidos
    - Investigaciones conjuntas con ISISTAN-UNICEN
- Acuerdo de Cooperación entre el Laboratorio de Investigación en Ecología Bentónica y el Laboratorio de Parasitología e Histopatología de Moluscos del Instituto de Biología Marina y Pesquera Almirante Storni.

Introducción

Cada vez más compañías ven su software como un activo valioso y están

promocionando la reutilización para incrementar sus beneficios en las inversiones software. La ingeniería de software basada en reuso es una aproximación del desarrollo que intenta maximizar la reutilización del software existente. Las unidades de software que se reutilizan pueden ser de tamaños diferentes. Por ejemplo:

1. *Reutilización de sistemas de aplicaciones*. La totalidad de un sistema de aplicaciones puede ser reutilizada incorporándola sin ningún cambio en otros sistemas, configurando la aplicación para diferentes clientes o desarrollando familias de aplicaciones que tienen una arquitectura común pero que son adaptadas a clientes particulares.
2. *Reutilización de componente y servicios*. Esta reutilización varía en tamaño desde subsistemas hasta objetos simples. El extremo más elemental nos permite la reutilización de componentes software que implementan una única función, como una función matemática o una clase de objetos. Este tipo de reutilización, basada en bibliotecas estándar, ha sido habitual en los últimos cuarenta años; estando disponibles diversas bibliotecas para distintas plataformas de desarrollo.

Los sistemas y componentes software son entidades reutilizables, pero su naturaleza específica significa a veces que el costo de modificarlos para una nueva situación resulta elevado. Una forma complementaria de reutilización es el reuso de conceptos, en el que la entidad reutilizada es más abstracta y se diseña para ser configurada y adaptada a una variedad de situaciones. El *reuso de conceptos* puede incluirse en aproximaciones tales como patrones de diseño, productos de sistemas configurables y generadores de programas. La ventaja obvia del reuso de software es que los costos totales de desarrollo deberían reducirse. Sin embargo, hay también costos y problemas asociados al reuso. En particular, hay un costo significativo asociado con el estudio de si un componente es apropiado para su reutilización en una situación concreta y con la prueba de ese componente para asegurar su confiabilidad. Estos costos adicionales pueden inhibir la introducción del reuso como paradigma de desarrollo.

El reuso sistemático no se lleva a cabo sin más, sino que debe ser planificado e introducido ampliamente en una organización a través de un programa de reuso. Existen diversas técnicas para dar soporte al reuso. Éstas exploran el hecho de que los sistemas del mismo dominio de aplicación son similares y tienen potencial para el reuso. Dada esa cantidad de técnicas – que incluyen desde componentes tradicionales a componentes off-the-shelf y servicios – la cuestión clave es ¿cuál es la técnica más adecuada a utilizar? Obviamente, eso depende de diversos factores; por ejemplo, los requerimientos del sistema a desarrollar, la tecnología y los activos reutilizables disponibles, y a experiencia del grupo de desarrollo. Hoy día, cuando se intenta reusar componentes y servicios, se está limitado de forma inevitable por las decisiones de diseño que han sido tomadas por los implementadores de los componentes y servicios. Éstas varían desde algoritmos particulares que han sido utilizados para implementar, hasta los objetos y tipos de las interfaces. Cuando estas decisiones de diseño están en pugna con requerimientos particulares, reusar es imposible o requiere costos de adaptación exorbitantes. Una forma de solventar esto es reusar diseños abstractos

que no incluyen detalles de la implementación. Esta aproximación para reuso ha sido incluida en los llamados *patrones de diseño* [1][2]; sin embargo, su utilización efectiva requiere de un conocimiento profundo para decidir si se puede reutilizar o si se necesita desarrollar una solución de propósito específico. Otra forma de reuso a modo de estructura genérica puede verse en los llamados *frameworks orientados a objeto* [3] que se implementan como una colección de clases concretas y abstractas. Uno de los frameworks más conocidos y ampliamente usados para el diseño de interfaces de usuario es MVC (*Model-View-Controller*) propuesto originalmente en la década de los '80. El problema fundamental con los frameworks es su complejidad inherente y el tiempo que lleva aprender a utilizarlos. No hay dudas de que es una aproximación efectiva al reuso, pero supone un costo elevado al introducirla en los procesos de desarrollo.

En general, existen dos tipos de reuso de aplicaciones: la creación de nuevos sistemas integrando productos (*componentes, servicios*) existentes [7][8][9] y el desarrollo de *líneas de productos*[4][5][6].

En síntesis, aunque el desarrollo basado en reuso se está convirtiendo en una aproximación fundamental para el desarrollo de sistemas software, presenta varios problemas[10][11][12]:

1. Confiabilidad de los productos a ser reusados. Los productos deben tratarse como "cajas negras" y el código puede no estar disponible para los usuarios de dichos productos. En tales casos, ¿cómo saber que un producto es confiable? El producto puede no tener documentados modos de fallo que comprometen al sistema en el que se utiliza dicho producto o su comportamiento no funcional puede no ser el esperado.
2. Certificación del producto. Estrechamente relacionado con la confiabilidad, se halla la cuestión de la certificación. Se ha propuesto que asesores independientes deberían certificar la calidad de un producto a ser reusado; sin embargo, no está claro cómo se hace esto. Por ejemplo, ¿quién sería responsable si el producto no funciona bien de acuerdo con la certificación? ¿cómo podrían los certificadores limitar su responsabilidad?
3. Predicción de propiedades emergentes. Debido a que los productos son cajas negras, predecir sus propiedades es particularmente difícil. Como consecuencia, es posible encontrarse con que, cuando se integran productos, el sistema resultante tiene propiedades no deseadas que limitan su uso.
4. Equilibrio de requerimientos. Normalmente se tiene que encontrar un equilibrio entre los requerimientos ideales y los productos disponibles. Por el momento, alcanzar ese equilibrio es un poco intuitivo y se necesitaría un método más sistemático y estructurado para ayudar a los diseñadores a seleccionar y configurar productos.
5. Existencia de un mercado viable. La visión de desarrollo basado en reuso es que debería haber un mercado de productos viables donde los vendedores externos compitan para proporcionar componentes y servicios. Hoy día, esto no es una realidad. La principal razón para ello es que los usuarios de productos externos se enfrentan con los riesgos de que esos productos no funcionen tal y como se les dice. Si este es el caso, los costos de reuso superan a los beneficios.

6. Composiciones incompatibles. Cuando se crea un sistema mediante composición de productos, se puede observar que hay conflictos potenciales entre los requerimientos, la necesidad de entregar el sistema lo antes posible y de crear un sistema fácil de mantener. Por ejemplo, las decisiones a las que se debe llegar a un equilibrio son ¿qué composición es más eficiente para satisfacer los requerimientos del sistema? ¿qué composición permite adaptaciones para futuros cambios?

Estos problemas han sido abordados desde diversas perspectivas, incorporando por ejemplo técnicas y herramientas basadas en conocimiento que exploran repositorios para reuso. Sin embargo, la reutilización efectiva de activos (recursos y servicios) entre las organizaciones requiere tratamientos que son materia de investigación en ámbitos complejos; por ejemplo, la integración de arquitecturas empresariales o más recientemente, los denominados "ecosistemas de software" que consisten en un conjunto de empresas que extienden la funcionalidad de una plataforma suministrada por otra[1]. Ese tipo de ecosistemas están siendo ampliamente usados en distintos dominios (ejemplos incluyen SalesForce, Facebook y Amazon entre otros), donde los usuarios contribuyen con conocimiento, contenido, bienes y servicios, conexiones o comportamiento a una comunidad, y consecuentemente, la empresa suministra una plataforma que se transforma en un ecosistema social. El éxito en estos sistemas depende de las contribuciones de los usuarios para crear mayor valor incrementalmente. Eso implica investigar, entre otras cosas, en técnicas más avanzadas en la personalización de la información o en la composición de la solución usada por cada usuario.

[1]http://www.software-ecosystems.com/Software_Ecosystems/Ecosystems.html

Líneas de Investigación y Desarrollo

El perfil del grupo GIISCo hoy puede definirse en base a las actividades a las que da soporte en la Investigación y Transferencia en tópicos relacionados con Ingeniería de Software. Los temas específicos se adaptan dinámicamente a una disciplina que presenta desafíos diferentes asociados al crecimiento de la Tecnología de la Información y las Comunicaciones. Actualmente, abordamos los siguientes aspectos:

- Identificación, Evaluación y Composición de Servicios.
- Sistemas de Información Geográficos: Integración y consulta de datos distribuidos, desarrollo basado en reuso.
- Accesibilidad Web: Desarrollo y evaluación de sitios Web en vistas de un acceso universal de la información.
- Mejora de procesos y productos de la Ingeniería de Software aplicados a e-Gov.
- Reuso de conocimiento

Resultados y Objetivos

En [13], hemos enumerado una serie de contribuciones anteriores. Durante el año 2011, hemos profundizado la investigación en aspectos de desarrollo de *líneas de productos*, generando métodos de desarrollo e implementaciones en el área de ecología marina; *identificación de servicios*, generando métodos en base a testing y en colaboración con

investigadores de ISISTAN, UNICEN; *reuso de conocimiento*, abordando el tema de información en foros técnicos*; e-gobierno*, generando técnicas para facilitar la incorporación de notificaciones electrónicas en el ámbito judicial; y *accesibilidad web*, generando métodos para su diseño desde etapas tempranas.
Las líneas de investigación convergen en el tratamiento del desarrollo de software basado en reuso desde perspectivas que marcan tendencias en la investigación y desarrollo del área. Desde el análisis de dominios y el modelado de líneas de productos, que permite avanzar en el reuso a escala de sistemas globales, hasta el reuso de servicios específicos – utilizando las últimas plataformas y avances tecnológicos incluyendo semántica y estandarización – y pasando por el reuso de conocimiento que inherentemente da soporte a desarrollos en todo el espectro; la visión de esta línea de investigación se resume en:

"Desarrollar técnicas y herramientas que permitan avanzar en el estado del arte y de la práctica actual en procesos específicos de desarrollo basado en reuso, que son pilares esenciales para la consolidación de la construcción de aplicaciones software como una actividad sistemática de ingeniería".

Formación de Recursos Humanos

GIISCo reúne aproximadamente a 25 investigadores, entre los que se cuentan docentes y alumnos de UNComa y asesores externos. Varios de los docentes-investigadores de GIISCo-UNComa han terminado o se encuentran próximos a terminar carreras de postgrado. Se cuenta actualmente entre los miembros del grupo con 6 doctores (dos de ellos han sido designados investigadores-asistentes CONICET), 1 magister (defensa 2011), 3 doctorandos (dos de ellos becarios CONICET) y 1 maestrando. Dirección de Tesis durante 2011: Doctorado (4 tesistas), Maestría (4 tesistas), Grado (7 tesistas).

Referencias

[1] Gamma et al. (1994) Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley
[2] Buschman et al. (1996). Pattern-Oriented Software Architecture. Wiley & Sons.
[3] Fayad et al (1999). Implementing Application Frameworks. Wiley & Sons.
[4] Pohl et al. (2005). Software Product Line Engineering. Springer.
[5] Clements & Northrop (2002). Software Product Line. Addison-Wesley.
[6] Van der Linden et al (2207). Software Product Lines in Action. Springer.
[7] Bell (2008). Service-Oriented Modeling. Wiley & Sons.
[8] Zimmermman et al (2003) Perspectives on Web Services. Springer.
[9] Erl et al (2009) Web Service Contract Design & Versioning for SOA. Prentice-Hall.
[10] Cechich et al. (2006). Trends on COTS Component Identification. Proceedings of the Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, IEEE Computer Society, p. 90-99.
[11] Bosch (2006) The Challenges of Broadening the Scope of Software Product Families. Communications of the ACM. December 2006.
[12] Bosch (2006) Expanding the Scope of Software Product Families: Problems and Alternative Approaches. Proceedings of the 2nd International Conference on the Quality of Software Architectures (QoSA 2006).
[13] Alejandra Cechich, Agustina Buccella, Andrés Flores, Gabriela Aranda, Nadina Martínez Carod, Juan Luzuriaga, Rodolfo Martínez, Marcelo Moyano, Rafaela Mazalu, Adriana Martin, Martín Garriga, Evaluación y Uso de Composiciones Software, XIII WORKSHOP DE INVESTIGADORES EN CIENCIAS DE LA COMPUTACION, Rosario, 2011