Repositorio de componentes de software para sistemas de la Iglesia Adventista del Séptimo Día

Rubén Jarib Sánchez Rosado, Jazmín Hernández Hernández,
Germán Harvey Alférez Salinas
Facultad de Ingeniería y Tecnología, Universidad de Montemorelos

Abstract

El desarrollo de software basado en componentes se ha convertido en uno de los paradigmas más efectivos para la construcción de aplicaciones computacionales. Su uso se ha incrementado como estrategia clave de la ingeniería de software por la necesidad de acelerar el ciclo de desarrollo y a la vez la reducción de costos. Esto es porque el desarrollo se apoya en componentes de software ya desarrollados, que son combinados adecuadamente para satisfacer los requisitos del sistema.

Bajo este planteamiento, se proponen la creación y utilización de un repositorio en línea de componentes reutilizables para ser empleado por desarrolladores de software de la Iglesia Adventista alrededor del mundo, con el afán de suplir la demanda de programas informáticos en instituciones adventistas, como son los más de 11,000 colegios y universidades, además de hospitales, canales de televisión, estaciones de radio, entre otros.

1. Introducción

El Desarrollo de Software Basado en Componentes (DSBC) sienta las bases para el diseño y desarrollo de aplicaciones basadas en componentes reutilizables de software. Esta disciplina cuenta actualmente con un creciente interés, tanto en los ámbitos académico e industrial (Fuentes, Troya y Vallecillo, n.d.), ya que permite que la construcción de nuevos sistemas sea más fácil, más rápida y más económica (Stepanek, G., 2005). Por esto se propone la creación de un repositorio de componentes en línea para la Iglesia Adventista del Séptimo Día (IASD).

Con la creación de un repositorio de componentes de software,

las instituciones de la IASD con departamentos de desarrollo de software podrán colaborar mutuamente y beneficiar a las entidades con menores recursos ofreciendo un cúmulo de elementos de software listos para ser utilizados, corregidos y mejorados. Además, todas las dependencias podrían beneficiarse de componentes de software desarrollados en las principales universidades del sistema educativo adventista y servirse, recíprocamente, de los recursos que las demás instituciones provean.

Asimismo, los departamentos de sistemas de la IASD gozarían de medios para hallar los componentes de una forma rápida y sencilla. También se evitaría re-codificar componentes y el consecuente gasto inútil de mano de obra y tiempo en re-implementar cosas que ya estaban hechas.

El segundo capítulo de este trabajo da una introducción de conceptos y métodos sobre los cuales se apoya el DSBC. El capítulo tres presenta la problemática por resolver. El siguiente capítulo presenta la estructura propuesta para el repositorio, planteando la forma en la que se lleva a cabo la especificación de un componente de software. Finalmente, se presentan las conclusiones.

2. Desarrollo de software basado en componentes

En este capítulo se presentan la definición, los beneficios y los retos que conllevan los componentes de software.

2.1 Componente de software:

Un componente de software es una "unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y constituido con otros componentes de forma independiente, en tiempo y espacio" (Szypersky, 1997).

Un componente de software puede ser definido por los siete criterios (Meyer, 2003):

1. Puede ser utilizado por otros elementos de software
2. Puede ser usado por los clientes sin la intervención del desarrollador del componente
3. Incluye la especificación de todas sus dependencias
4. Incluye la especificación la funcionalidad que ofrece
5. Es usable con base en sus especificaciones
6. Es integrable con otros componentes
7. Puede ser integrado a un sistema rápida y suavemente.

Uno de los conceptos más importantes en el desarrollo basado en componentes es la reutilización de activos de software preexistentes. Esta reutilización promueve el desarrollo de aplicaciones de una manera más ágil, mejorando así la velocidad de respuesta a los cambios en la lógica del negocio. Sin embargo, no todas las partes del sistema pueden ser reusables, por lo que se deben escoger los aspectos de la aplicación que tengan el potencial de serlo, y elegir una arquitectura apropiada (Pressman, 2002).

2.2 Beneficios de los componentes: Uno de los mayores beneficios al utilizar componentes es que con ellos los desarrolladores minimizan el tiempo de desarrollo al reducir el trabajo ya realizado.

Además se mejora la calidad, ya que un componente de software desarrollado para ser reutilizado es más propenso a ser libre de errores, lo que asegura la calidad y fiabilidad del mismo (Pressman, 2002).

Así mismo se mejora la productividad, ya que las aplicaciones creadas a partir de componentes reusables requieren un menor tiempo para el análisis, diseño y codificación consiguiendo la funcionalidad requerida por el usuario, pero con menos esfuerzo (Stepanek, 2005).

2.3 Retos del DSBC: Inherentes a esta aproximación de desarrollo de software, están una serie de problemas específicos como son:

• Evolución del componente: La gestión de la evolución de un componente es un problema serio, ya que las mejoras y reparaciones que se le puedan hacer al componente podrían resultar en la inserción de errores al sistema. Existen diferentes enfoques para abordar este problema, como la inmutabilidad de las interfaces y el replanteamiento de las mismas. Sin embargo, ninguno de estos enfoques resuelve el problema del todo (Fuentes, Troya y Vallecillo, n.d.).

• Interoperabilidad: Es la habilidad de dos o más entidades de comunicarse y cooperar sin importar la diferencia del lenguaje de implementación, el ambiente de ejecución o el modelo de abstracción. Este es un problema, ya que no siempre se cuenta con las especificaciones concretas (Madiajagan y Vijayakumar, 2006).

3. Necesidad de tener un repositorio de componentes en la Iglesia Adventista

Actualmente los departamentos de sistemas de la IASD se encuentran incomunicados en cuanto a los avances y desarrollos que cada uno está llevando a cabo. Esto genera desperdicio de recursos temporales, económicos y humanos, lo cual es delicado para una institución sin ánimo de lucro.

Por otra parte, como característica inherente de los miembros de la IASD, cada departamento de desarrollo sería un colaborador al compartir y permitir, por parte de los otros organismos, perfeccionar los componentes ya desarrollados. Además, se estaría evitando el problema de "re-inventar la rueda" en el desarrollo de software.

4. Diseño del repositorio

Para poder llevar a cabo el proceso de desarrollo de software basado en la reutilización de componentes, es importante tener un catálogo de software que sea fuente de información, para describir los componentes hasta el momento desarrollados en otros proyectos de software.

En el repositorio propuesto se incluyen las librerías requeridas, el código fuente, los archivos

ejecutables, así como una clara documentación tanto de su uso como de la descripción de su interfaz. Esto nos lleva a proponer la estructura de un árbol de proyecto para almacenar esta información dentro de un único archivo comprimido con extensión zip. Los archivos comprimidos que contienen al componente y su documentación serán almacenados en un servidor FTP. En la figura 1 se muestra la estructura del árbol del componente. En un directorio raíz (/) se incluyen los directorios que almacenarán los archivos ejecutables (bin), la documentación (doc), las librerías (lib) y el código fuente (src). Toda esta información es requerida.

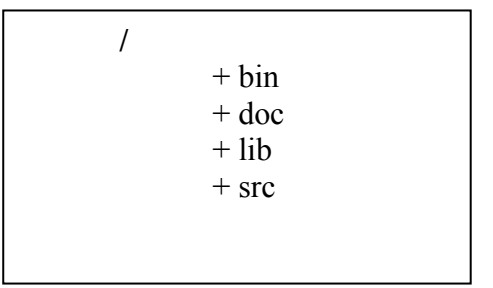

Figura 1. Estructura en árbol para componentes en el repositorio

Las búsquedas de los componentes se hacen con base en la descripción y nombre del componente requerido. Además, se ofrece la posibilidad de que el usuario seleccione alguna categoría y de esta forma se refine la búsqueda.

La descripción de los componentes se hace a través de metadatos. Estos se basan en el estándar interdisciplinario Dublin Core (Dublin Core® Metadata Initiative (DCMI), n.d.) que expone los siguientes atributos:

1. Nombre: el nombre dado a un recurso, habitualmente por el autor.
2. Lenguaje: tipo de código en el que fue desarrollado el componente.
3. Categoría: es el área temática del componente, por ejemplo: seguridad, reportes, búsquedas, etc.
4. Creador: la persona que haya hecho una contribución intelectual significativa.
5. Contribuyente: persona que ha aportado para la mejora del componente.
6. Descripción: una descripción textual de la funcionalidad del componente.
7. URL: dirección de recursos en la red.
8. Licencia: términos y condiciones de uso de los contenidos publicados. Estos regulan cuáles son los derechos que se ceden a los clientes de un producto de software, por ejemplo: copyright, GPL GNU.

9. Fecha: la fecha en que se liberó la versión actual.

Los siguientes atributos fueron añadidos en este estudio:
1. Versión: número que indica el nivel de desarrollo de un programa.
2. Institución: dependencia encargada del desarrollo del componente.
3. País: país en donde se encuentra la institución.
4. Dirección electrónica: dirección electrónica del creador del componente.
5. Fax: número de fax del creador del componente.
6. Idioma: lenguaje de publicación, de especificación y documentación del componente.
7. Palabras clave: conjunto de palabras que se relacionan con la naturaleza o género del contenido del componente.

En la Figura 2 se muestra un documento XML con la lista de atributos que describen al componente. Cada elemento <META> especifica una tupla de atributo junto con su valor. Los principales atributos que tiene son name, content. El atributo name identifica la propiedad y content le asigna un valor.

<META name=" Nombre " content=" KinAuthenticate"/>
<META name=" Version" content=" V3.0"/>
<META name=" Lenguaje" content=" Java "/>
<META name=" Autor " content=" Rubén González Pinedo "/>
<META name=" Contribuyente " content=" Juan Díaz Espinoza "/>
<META name=" Descripción " content="Provee una autenticación de usuario y permite a los administradores, administrar la información de seguridad, tales como usuarios en un grupo y los grupos a los cuales un usuario está asignado. KinAuthenticate elimina la necesidad de tener separadas el nombre de usuario y contraseña, en aplicaciones java "/>
<META name=" Url" content=" http:/fit.um.edu.mx/repositorio/seguridadyadministracion/KinAuthenticatev3.zip "/>
<META name=" Licencia" content=" GPL "/>
<META name=" Fecha de creación " content=" 2009-03-02"/>
<META name=" Institución " content=" Universidad de Montemorelos "/>
<META name=" País " content=" Mexico"/>
<META name=" Palabras clave " content=" Control de acceso "/>
<META name=" Dirección electrónica " content=" rubgp@gmail.com "/>
<META name=" Fax" content=" 8262630900 "/>
<META name=" Idioma" content=" Español"/>

Figura 2. Documento XML de metadatos descriptivos para componentes

5. Descripción del repositorio

Para el funcionamiento del repositorio es imprescindible contar con una herramienta de recuperación de información para resolver el problema de la búsqueda rápida y eficiente de componentes. Para lograr este fin se propone la utilización de una de las librerías más flexibles y poderosas del mercado, que ha tenido gran éxito y escalabilidad en muchas aplicaciones (Lucene; The Apache Foundation, n.d).

Lucene indexa los documentos XML que contiene la aplicación para que después a través de una consulta del usuario, la librería busque en el índice y muestre los resultados. Además, puede indexar y buscar datos almacenados en páginas Web localizados en servidores Web remotos, documentos almacenados en sistemas de archivos locales, archivos de texto simple, archivos HTML, o PDF, o en cualquier otro formato del que se pueda extraer información textual. De igual forma, con la ayuda de Lucene se pueden indexar datos almacenados en base de datos, dándoles a los usuarios la capacidad de búsqueda "full-text" (Gospodnetic y Hatcher, 2005).

5.1 Especificación de requisitos funcionales del repositorio de componentes: A continuación se presentan los casos de usos que describen los requisitos funcionales del sistema en forma gráfica (figura 3) y textual (figuras 4-6).

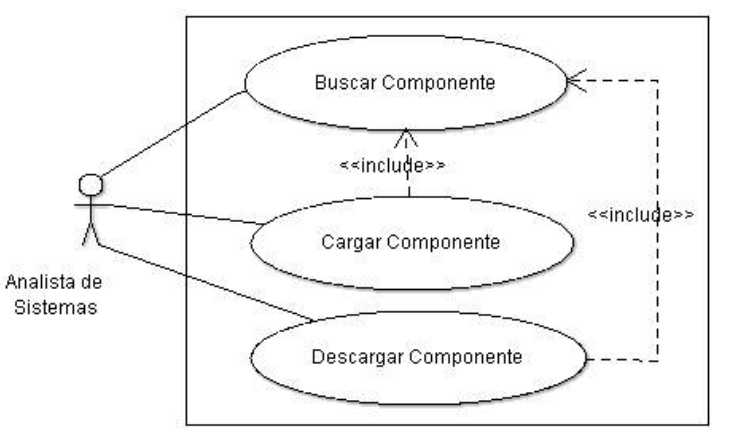

Figura 3. Diagrama de casos de uso del repositorio

UC-0001	Buscar componente	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se desea encontrar un componente que supla una funcionalidad requerida* o durante la realización de los siguientes casos de uso: [UC-0003] Descargar componente	
Precondición	Se requiere la funcionalidad provista por un componente.	
Secuencia normal	Paso	Acción
	1	El actor Analista de Sistemas (ACT-0001) *escribe las palabras clave relacionadas con el componente requerido*
	2	El sistema *despliega la lista de componentes reaccionadas con las palabras clave presentando los datos en orden de relevancia.*
	3	El actor Analista de Sistemas (ACT-0001) *selecciona el componente buscado.*
	4	El sistema *despliega los datos que describen al componente.*
Postcondición	Se encuentra el componente que satisface la funcionalidad requerida.	
Excepciones	Paso	Acción
	2	Si *no se encontraron registros que correspondan al componente buscado,* el sistema *presenta el mensaje: "No hubo coincidencias".*

Figura 4. Descripción del caso de uso buscar componente

UC-0002	Cargar Componente	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se actualice un componente existente en el repositorio o se dé de alta un nuevo componente.*	
Precondición	Hay un componente que necesita ser actualizado o existe un nuevo componente que se quiere compartir.	
Secuencia normal	Paso	Acción
	1	El actor Analista de Sistemas (ACT-0001) *empaqueta en un archivo .zip las librerías, código fuente, el ejecutable y el manual de usuario.*
	2	El actor Analista de Sistemas (ACT-0001) *selecciona el archivo .zip a través del botón examinar.*
	3	El actor Analista de Sistemas (ACT-0001) *presiona el botón cargar.*
	4	El sistema *almacena el archivo en el servidor FTP donde se encuentra localizado el repositorio.*
	5	El sistema *presenta una página con un formulario con los datos descriptivos del componente.*
	6	El actor Analista de Sistemas (ACT-0001) *llena el formulario y presiona el botón "Generar Metadatos".*
	7	El sistema *genera el archivo XML y lo guarda en la base de datos.*
Postcondición	La última versión del componente es almacenada en el repositorio.	

Figura 5. Descripción del caso de uso Cargar Componente

UC-0003	Descargar Componente	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se ha encontrado el componente deseado.*	
Precondición	Se realizó la búsqueda de un componente.	
Secuencia normal	Paso	Acción
	1	Se realiza el caso de uso Buscar Componente (UC-0001)
	2	El actor Analista de Sistemas (ACT-0001) *presiona el botón descargar.*
Postcondición	El Analista del Sistema finaliza exitosamente la descarga del componente.	

Figura 6. Descripción del caso de uso Descargar Componente

Conclusiones y trabajo futuro

En el presente artículo se describió un repositorio basado en componentes para ser utilizado por parte de los departamentos de desarrollo de software de la IASD. También se mostraron cuáles son las ventajas de compartir componentes y así mismo se motiva a compartir software.

Se espera que la implementación del presente proyecto sea de gran utilidad en la comunidad adventista gracias a que los desarrollos basados en componentes son rápidos y económicos. Además los componentes pueden ser modificados, corregidos y mejorados para suplir las diferentes necesidades de la IASD.

En el trabajo futuro se mostrarán los resultados de la implementación del repositorio de componentes.

Referencias

Dublin Core® Metadata Initiative (DCMI). (n.d.). Home. Retrieved 4 March 2010 from: http://dublincore.org/

Fuentes, L., Troya, J.M. y Vallecillo A. (n.d.). Desarrollo de software basado en componentes. Retrieved 4 March 2010 from: www.lcc.uma.es/~av/Docencia/Doctorado/tema1.pdf

Gospodnetic, O. y Hatcher, E. (2005). Lucene in Action. Manning Publications.

Madiajagan, M., y Vijayakumar, B. (2006). Interoperability in Component Based Software Development. World Academy of Science, Engineering and Technology. Retrieved 4 March 2010 from: www.waset.org/journals/waset/v22/v22-13.pdf

Meyer, B. (2003). The Significance of Components in Software Development. Retrieved 4 March 2010 from: www.sdmagazine.com/documents/s=7207/sdm9911k/

Pressman, R. (2002). Ingeniería del software. Un enfoque práctico. 5 ed. McGraw-Hill.

Stepanek G. (2005). Software Projects Secrets: Why Software Projects Fail. Apress.

Szypersky, C. (1997). Component Software. Beyond Object-Oriented Programming. Addison-Wesley Professional.

The Apache Foundation. (n.d.). Apache Lucene. Retrieved 4 March 2010 from: http://lucene.apache.org/java/docs/index.html