Repositorio de Metadatos de Componentes de Software Reutilizables Para Desarrolladores de la Iglesia Adventista del Séptimo Día

Rubén Jarib Sánchez Rosado[1], Jazmín Hernández Hernández[2], Germán Harvey Alférez Salinas[3]
Facultad de Ingeniería y Tecnología,
Universidad de Montemorelos
1. jaribs@ alumno.um.edu.mx , 2. jazminpalom@alumno.um.edu.mx,
3. harveyalferez@um.edu.mx

Technical Report COMP-025-2009

Abstract

El desarrollo de software basado en componentes se ha convertido actualmente en uno de los paradigmas más efectivos para la construcción de aplicaciones de software. Su uso se ha incrementado como estrategia clave de la ingeniería de software. Y resulta en la solución para la continua 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 propone la creación y utilización de un repositorio en línea de componentes reutilizables, para uso de desarrolladores de software de la Iglesia Adventista alrededor del mundo, con el afán de suplir la demanda de programas informáticos para el manejo de instituciones adventistas como son los 11,302 colegios y universidades, además de hospitales, canales de televisión, radio, entre otros.

1 Introducción

El *Desarrollo de Software Basado en Componentes* (DSBC) sienta las bases para el diseño y desarrollo de aplicaciones distribuidas basadas en componentes de software reutilizables. Esta disciplina cuenta actualmente con un creciente interés, tanto en los ámbitos académico e industrial [1], ya que permite que la construcción de nuevos sistemas sea más fácil, rápida y más económica [2]. Estos son los factores deseables que se buscan en todos los ámbitos de desarrollo de software; la Iglesia Adventista del Séptimo Día (IASD) y sus instituciones no son la excepción. Por esto se propone la creación y utilización de un repositorio en línea de componentes reutilizables clasificados por modelo de componentes.

Mediante la presente propuesta, las instituciones en la IASD con departamentos de desarrollo de software podrían 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 prefabricados de antemano, y el consecuente gasto inútil de mano de obra y tiempo en re-implementar cosas que ya estaban hechas, es decir, querer hacer todo uno mismo.

El segundo capítulo de este trabajo da una introducción de conceptos y métodos sobre los cuales se apoya el DSBC. El siguiente capítulo presenta la estructura propuesta para el repositorio, planteando la forma en 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 capitulo se presentan las definiciones más difundidas de los componentes, además de los beneficios de usarlos y los retos del desarrollo de software basado en componentes.

2.1 Definición de componentes

En la literatura existe gran diversidad de definiciones de componente de software. A continuación se presentan las más difundidas.

Un componente de software, de acuerdo con Szypersky es: "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 compuesto con otros componentes de forma independiente, en tiempo y espacio." [3]

También un componente de software puede ser definido por los siete criterios de Meyer [4]

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 en base a sus especificaciones
6. Es integrable con otros componentes
7. Puede ser integrado a un sistema rápida y suavemente.

Otro de los conceptos que envuelve el desarrollo basado en componentes es la reutilización de activos de software preexistentes. Esta 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 parte del sistema puede ser reusables, por lo que se deben escoger los aspectos de la aplicación que tengan el potencial de serlo, y elegir una arquitectura apropiada para construirla [5].

2.2 Beneficios de los componentes

El primer beneficio es que con componentes los desarrolladores minimizan el tiempo de desarrollo al reducir trabajo ya hecho. La fiabilidad del sistema es mayor.

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 [5].

Se mejora la productividad ya que las aplicaciones creadas a partir de componentes reusables requieren menor tiempo para el análisis, diseño y codificación consiguiendo la funcionalidad requerida por el usuario, pero con menos esfuerzo [2].

2.3 Retos del DSBC

Inherente a esta aproximación de desarrollo de software, están ligados 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 ninguna de estas resuelve el problema del todo. [1]

- **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. [6]

- **No Disponibilidad de un repositorio de componentes:** Esta es la falta de un repositorio donde los componentes puedan ser almacenados.

3 Necesidad de tener un repositorio para 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 desarrollados.

La IASD necesita, por lo tanto, una forma de producir software evitando de esta forma "reinventar la rueda" en ámbitos de desarrollo de software. Esto otorgará a los departamentos de sistemas un sitio centralizado auxiliar para hallar elementos que podrían ser de utilidad para sus desarrollos.

4 Diseño del repositorio

Qué 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 zip que contienen al componente y su documentación serán almacenados en un servidor FTP.

En la Figura1 se muestra la estructura del árbol de proyecto. Dentro de 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.

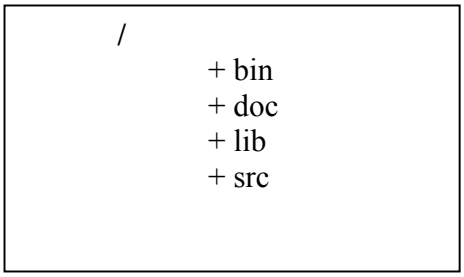

Figura 1.- Estructura en árbol de la información del componente.

Las búsquedas se hacen en base a la descripción y nombre del componente requerido. Además se ofrece la posibilidad de que el usuario seleccione alguna categoría y de esta forma refinar la búsqueda.

La descripción de los componentes se hace a través de metadatos. Estos se basan en el estándar interdisciplinario *Dublin Core* [7] 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 tenido 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 cuales son los derechos que se ceden a los clientes de un producto de software por ejemplo: c*opyright*, GPL, GNU.

9. **Fecha:** la fecha en que se libero la versión actual.

Estos son añadidos por nosotros:

1. **Versión:** es un número que indica el nivel de desarrollo de un programa.
2. **Institución:** dependencia encargada del desarrollo del componente.
3. **País:** lugar en donde se encuentra la institución.
4. **Correo electrónico:** permite el intercambio de mensajes a través de sistemas de comunicación electrónicos.
5. **Fax:** método y aparato de transmisión y recepción de documentos mediante la red telefónica conmutada.
6. **Idioma:** lenguaje de publicación del componente su especificación y documentación.
7. **Palabras clave:** conjunto de palabras que se relacionan con la naturaleza o genero 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/KinAuthen
ticatev3.zip "/>
<META name=" Licencia" content=" GPL "/>
<META name=" Fecha de creación " content=" 2009-03-02"/>
<META name=" Institucion " content=" Universidad de Montemorelos
"/>
<META name=" Pais " content=" Mexico"/>
<META name=" Palabras clave " content=" Control de acceso "/>
<META name=" Correo electrónico " content=" "/>
<META name=" Fax" content=" "/>
<META name=" Idioma" content=" Español"/>
```

Figura 2.- Documento XML de metadatos descriptivos.

5 Funcionamiento 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 información. Para lograr este fin se utiliza una de las librerías más flexibles y poderosas del mercado, que ha tenido gran éxito y escalabilidad en muchas aplicaciones, Lucene. Esta librería fue creada (Open Source) y totalmente escrita en Java [8]

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 en servidores Web remotos, documentos almacenados en sistemas de archivos locales, archivos de texto simple, archivos html, o pdf, o cualquier otro formato del que se pueda extraer información textual [9]. De igual forma, con la ayuda de Lucene se pueden indexar datos almacenados en base de datos, dándoles a los usuarios capacidades de búsqueda de texto completo. Que muchas bases de datos no proveen [10].

La aplicación propuesta ha sido concebida con una arquitectura de tres capas, en la Figura 3 se muestra un esbozo de esta.

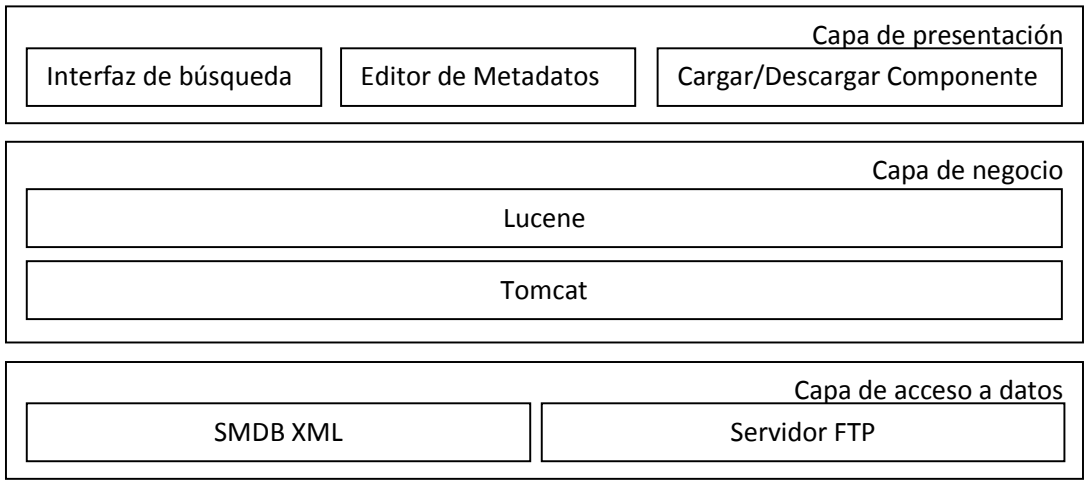

Figura 3. Arquitectura de tres capas del sistema

Creando la interfaz de búsqueda, la carga y la descarga del componente mediante la Servlet/JSP, la parte del modelo ha sido desarrollada en base a al servicio de búsqueda de Lucene y usando el servidor Tomcat como contenedor de los JSP y logrando la persistencia de los metadatos mediante el uso de eXist.

5.1 Especificación de requisitos funcionales del repositorio de componentes.

A continuación se presentan los casos de usos de que describen los requisitos funcionales del sistema, de manera gráfica (Figura 4) y textual, además de los requisitos no funcionales.

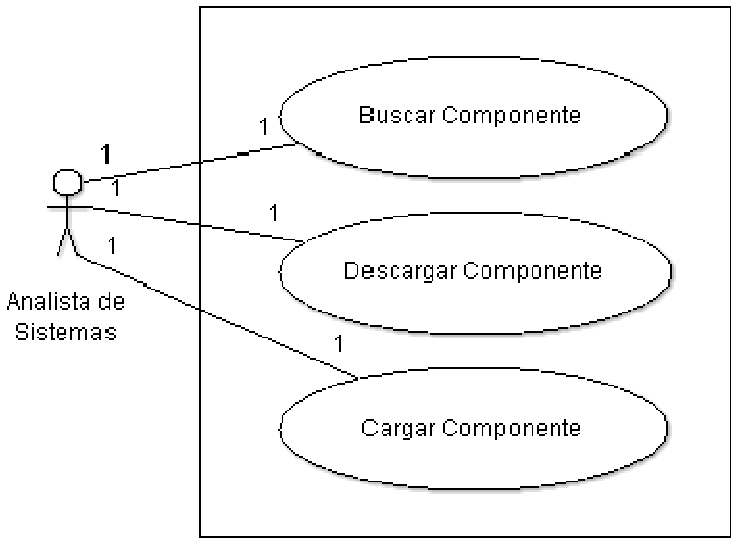

Figura 4.- Diagrama de caso de uso del sistema.

UC-0001	**Buscar componente**	
Versión	1.0 (09/03/2009)	
Autores	Jarib Sanchez Jazmin Hernadez	
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) *tipea las palabras clave, relacionadas con el componente requerido*
	2	El sistema *despliega la lista de componentes reaccionadas a 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 satisfaga 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"*, a

Importancia	vital

UC-0002	**Cargar Componente**	
Versión	1.0 (09/03/2009)	
Autores	Jarib Sanchez Jazmin Hernadez	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se actualice un componente preexistente en el repositorio o se dé de alta un nuevo componente.*	
Precondición	hay un componente que necesita ser actualizado o existe un nuevo componente	
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) *Se 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 *presentara 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.	
Importancia	vital	

UC-0003	**Descargar componente**	
Versión	1.0 (09/03/2009)	
Autores	Jarib Sanchez Jazmin Hernadez	
Dependencias	Ninguno	
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.	
Importancia	vital	

NFR-0001	**Rendimiento**
Versión	1.0 (09/03/2009)
Autores	Jarib Sanchez Jazmin Hernadez
Dependencias	Ninguno
Descripción	El sistema deberá *realizar los procesos en línea con tiempos de respuesta aceptables.*
Importancia	Vital

NFR-0003	**Seguridad**
Versión	1.0 (09/03/2009)
Autores	Jarib Sanchez Jazmin Hernadez
Dependencias	Ninguno
Descripción	El sistema deberá *permitir el ingreso por medio de sesión de usuario.*
Importancia	vital

NFR-0004	**Portabilidad**
Versión	1.0 (09/03/2009)
Autores	Jarib Sanchez Jazmin Hernadez
Dependencias	Ninguno
Descripción	El sistema deberá *garantizar compatibilidad con navegadores de uso común.*
Importancia	Importante

6 Conclusiones y trabajo futuro

El presente artículo mostró los beneficios de tener un repositorio basado en componentes en los departamentos de desarrollo de software de la IASD, también nos muestra cuales son las ventajas de compartir los desarrollos y así mismo motiva a desarrollar previendo que probablemente alguien más necesitara y utilizara tales componentes.

Este proyecto es de gran utilidad en la comunidad Adventista, gracias a la facilidad de realizar desarrollos de software basado en componentes de una manera rápida y económica,

los cuales pueden ser modificados, corregidos y mejorados para las diferentes necesidades de la IASD.

En definitiva, un repositorio de componentes de software impulsaría enormemente a las instituciones educativas de la IASD otorgando a los estudiantes y profesores un sitio en el cual darse a conocer, incrementando así su experiencia en el desarrollo de sistemas de cómputo.

En el próximo trabajo se mostrara la manera en que se implemento el repositorio de metadatos de componentes de la IASD, el cual contendrá un diseño detallado de la interfaz, tanto como los diagramas del diseño y la implementación del mismo.

En el proceso de implementación de este proyecto ciertamente habrá algunas modificaciones en el diseño del repositorio de metadatos de componentes, ya que pueden surgir nuevas ideas que se podrían aplicar para mejorar su funcionalidad.

7 Referencias

[1] J. M. Lidia Fuentes, "Desarrollo de Software basado en componentes," p. 1, 2003.

[2] G. Stepanek, *Software projects Secrets : Why software projetcts fail*. Apress, 2005.

[3] C. Szypersky, "Component Software. Beyond Object-Orientend Programming," 1998.

[4] B. Meyer, *The significance of the components. Beyond Objects Colum, Software Development* vol. 11, 7 ed. 1997.

[5] R. Pessman, *INGENIERÍA DEL SOFTWARE. Un enfoque práctico*, 5 ed. McGrawHill, 2002.

[6] M. M. and Vijayakumar, "Interoperability in Component Based Software Development," *World Academy of Science, Engineering and technology*, vol. 16, Nov. 2006.

[7] (2009, Feb. 25). *Dublin Core Metadata Initiative* [Online]. Available: http://dublincore.org/

[8] T. A. Foundation. (2009, Feb. 19). *Apache Lucene* [Online]. Available: http://lucene.apache.org/java/docs/index.html

[9] A. G. J. (2007). *Indización y Búsqueda a través de Lucene* [Online]

[10] O. G. Erik Hatcher, *Lucene in Action*. Manning, 2005.