Propuesta de aplicación de la ingeniería del software basada en componentes en el desarrollo de software empresarial

Fredy Humberto Vera Rivera
Magíster en Ingeniería Área Informática y Ciencias de la Computación.
Universidad Industrial de Santander
Docente Universidad Santo Tomás USTA
Grupo de Investigación UNITEL - USTA
Bucaramanga, Colombia
freve9@hotmail.com

Fernando Rojas Morales
Magíster en Ciencias Computacionales.
Profesor Escuela de Ing. de Sistemas - UIS.
Grupo Investigación en Ingeniería Biomédica – UIS
Bucaramanga, Colombia
frojas@uis.edu.co

Resumen— Este artículo presenta una investigación realizada en conjunto por la Escuela de Ingeniería de Sistemas y la División de Servicios de Información (DSI) de la Universidad Industrial de Santander (UIS), en la cual se plantea un proceso de desarrollo de componentes software reutilizables en Java Versión Empresarial 5, el cual fue utilizado en el desarrollo de software empresarial en la UIS, con resultados positivos, dentro de los cuales se destacan: aumento en la productividad, reutilización de software, mayor rapidez en la construcción de software , disminución de los esfuerzos de mantenimiento, y se puede mejorar la calidad y la oportunidad en el desarrollo de nuevos sistemas. En el artículo se aclara la definición de componente software desde el punto de vista de diferentes autores, plantea el desarrollo de un componente software reutilizable con las principales librerías que componen la versión Java empresarial, el cual consta de Ejbs de entidades, Ejbs de sesión, componentes personalizados para la interfaz de usuario y servicios web, posteriormente plantea el proceso de desarrollo de componentes software reutilizables, el cual detalla las fases necesarias para crear una biblioteca de componentes software de este estilo y por último se dan las conclusiones del trabajo realizado.

Palabras clave— Componente, Ingeniería del Software basada en componentes, Reutilización de software.

Abstract— This paper presents a research by the School of Systems Engineering, and Information Services Department of the Universidad Industrial de Santander (UIS). This research consists of a development of reusable software components in Java Enterprise Edition 5. The results of this development showed several advantages, such as an increase in productivity, software reuse, quicker building of the software application, and a decrease in software maintenance efforts. All of these advantages resulted in an improvement of quality and timeliness in the development of new systems. This article presents the definition of software component from the point of view of different authors. It also proposes the development of a reusable software component using the main libraries that make up the Java Enterprise Edition 5, (consisting of entity EJBs, Session EJBs, custom components for the user interface and web services). Then, this article describes the steps needed to create a library of reusable software components, and finally gives the conclusions of the work.

Keywords— Component, Component – Based Software Engineering, Reuse of Software.

I. INTRODUCCIÓN

Los principios de reutilización de software han estado presentes a lo largo de muchos años, y para un ingeniero de software es conveniente contar con un conjunto de unidades software o aplicaciones modulares que se puedan ensamblar para formar un sistema nuevo más grande y complejo, en vez de tener que desarrollar el sistema nuevo desde cero. A estas unidades software se les conoce como componentes. Al poder utilizar estos componentes software, que ya han sido probados y verificados se puede disminuir el tiempo de desarrollo y hacer sistemas informáticos más confiables y seguros.

Cuando se requiera construir un nuevo sistema, se buscarían los componentes en el repositorio, se combinarían y adaptarían, en vez de codificar y construir las mismas aplicaciones cada vez. Las empresas desarrolladoras de software y la industria en general, deben aplicar este modelo de desarrollo basado en componentes para poder crear su propia biblioteca de componentes para hacer más fácil, seguro y rápido el desarrollo de nuevos sistemas informáticos.

El ensamblaje de componentes lleva a una reducción en el tiempo de desarrollo, una reducción en los costos del proyecto y un aumento en el índice de productividad. Aunque estos resultados están en función de la robustez de la biblioteca de

componentes, no hay duda que el ensamblaje de componentes proporciona ventajas significativas para los ingenieros de software y las empresas en general.

La ingeniería del software basada en componentes (ISBC) es un proceso que se centra en el diseño y construcción de sistemas basados en computadora que utilizan componentes de software reutilizables. La ISBC lucha por conseguir un conjunto de componentes de software preconstruidos y estandarizados que estén disponibles para encajar en un estilo arquitectónico específico para algún dominio de aplicación. La aplicación se ensambla entonces con estos componentes y no las piezas por separado de un lenguaje de programación convencional, como se enuncia en [1].

La ISBC, enmarca un conjunto de actividades necesarias para llevar a cabo la construcción de un sistema a partir de componentes reutilizables, en forma general estas actividades son:

- Ingeniería de Dominio: Consiste en la creación de la biblioteca de componentes. Establece el conjunto de componentes de software que el ingeniero del software puede reutilizar.
- Desarrollo basado en componentes: Enmarca las actividades necesarias para la construcción del sistema informático a partir de componentes.

La presente investigación se centró en la ingeniería de dominio, más específicamente en la forma de crear un componente software reutilizable que se puede usar en el desarrollo de muchas aplicaciones de software.

En el presente artículo se pretende, en primer lugar, aclarar la definición de componente, sección II; en la sección III, se plantea un componente software reutilizable para ser implementado en java versión empresarial; en la sección IV se plantea el proceso de desarrollo de componentes; por último se plantean los resultados y conclusiones del trabajo realizado.

II. DEFINICIÓN DE COMPONENTE SOFTWARE

En esta sección se aclararán los conceptos y definiciones más importantes del término componente, y se especificarán los requisitos que una unidad software debe cumplir para considerarse como componente.

En primer lugar, el concepto de componente muchos autores lo definen de forma diferente:

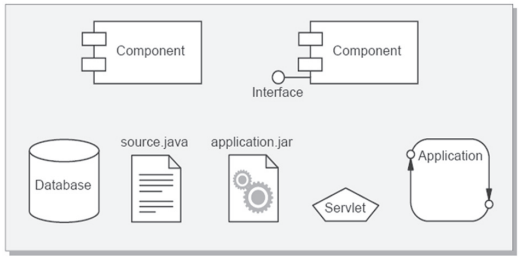
Según Szyperski [2], "Un *componente* es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio". También dice que "Un componente es un módulo y una serie de recursos. Un módulo es una serie de clases, procedimientos y funciones. Un recurso es una colección congelada de elementos tipados".

Meyer define en [3] siete criterios que debe cumplir un elemento software para poder definirse como componente:

1) "Debe ser usado por otros elementos software.
2) Debe ser usado por clientes sin la intervención del desarrollo de componentes.
3) Incluir una especificación de todas las dependencias.
4) Incluir una especificación de las funcionalidades que ofrece.
5) Es usado solamente con base en su especificación.
6) Se puede componer con otros componentes para formar un software.
7) Puede ser integrado rápidamente y fácilmente".

En el documento escrito por Sun Microsystems [4], se define un componente como una unidad de software que debe tener una interfaz que lo exporte como un servicio a otros componentes, puede ser algo grande y abstracto, también definen como componentes a los elementos mostrados en la Fig. 1.

FIG. 1. EJEMPLOS DE COMPONENTES

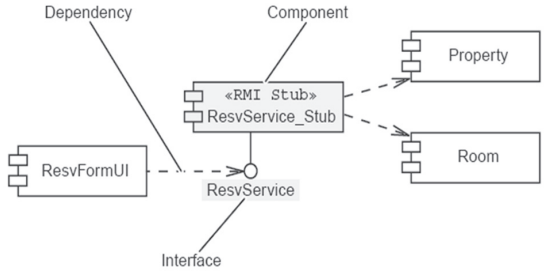


Fuente: Sun Microsystems [4]

En UML se utiliza el diagrama de componentes para mostrar las dependencias, relaciones e interfaces entre los componentes software y para representar las dependencias entre las estructuras de datos implementadas. También se utilizan para detallar el modelo de arquitectura y el modelo de distribución de un sistema informático. En la Fig. 2 se pueden apreciar unos ejemplos de diagramas de componentes.

Hyung Cho y John D. McGregor [5], definen el término componente como piezas software que pueden ser combinadas para construir algo más grande y completo (otro componente, subsistema, sistema).

FIG. 2. EJEMPLOS DE DIAGRAMA DE COMPONENTES SOFTWARE



Fuente: Sun Microsystems [4]

Iribarne Martínez Luis F dice que un componente software puede ser desde una subrutina de una librería matemática, hasta una clase en Java, un paquete en Ada, un objeto COM, un JavaBeans, o incluso una aplicación que pueda ser usada por otra aplicación por medio de una interfaz especificada [6].

Después de ver las definiciones que diferentes autores dan al concepto de componente, se puede definir a un componente como una unidad software utilizada para ensamblar o componer un sistema más grande y complejo, un componente debe contener una especificación que permita identificarlo y reutilizarlo. Un elemento software para poder clasificarse como componente debe cumplir con los siguientes criterios:

- Debe ser una unidad software sin dependencia estructural de otras unidades.
- Debe tener un alto grado de cohesión.
- Debe tener una identificación, una descripción y una especificación de las funcionalidades que realiza.
- Debe seguir un modelo de componentes.
- Debe poder ensamblarse de forma rápida y fácil con otros componentes para formar un sistema más grande.
- Debe tener una interfaz que permita utilizar, modificar y adaptar las funcionalidades que maneja.
- Debe estar certificado y probado para asegurar que cumple con las funcionalidades para lo que fue implementado.
- Debe permitir el mantenimiento y la actualización de forma individual, sin afectar estructuralmente al sistema que compone.

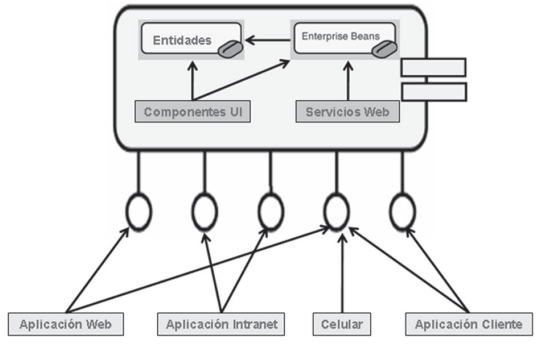
Después de revisar el concepto de componente es importante mirar la definición de Modelo de Componentes, que es otro de los términos más utilizados en el ámbito del desarrollo de software basado en componentes.

Un **modelo de componentes** define la forma como se especifica un componente y la forma como se ensambla el componente, también incluye una serie de tipos de componente, sus interfaces y una especificación de los patrones aceptables de interacción entre tipos de componentes [1]. Ejemplos de modelo de componentes: COM/DCOM, .Net, CORBA CCM, EJB, OSGi y Web Services [1].

III. COMPONENTE SOFTWARE REUTILIZABLE PROPUESTO

En la Fig. 3 se aprecia el contenido y las características fundamentales del componente software reutilizable propuesto.

FIG. 3. COMPONENTE SOFTWARE REUTILIZABLE



Como se puede observar en la figura, el componente de estudio de la presente investigación es un conjunto de subcomponentes, los cuales se detallan a continuación [13]:

- Entidades o antiguos EJB de Entidad, Mapean el modelo del dominio del problema con una base de datos relacional.
- Enterprise Java Beans (EJB), Se implementan los servicios que va a prestar el componente software reutilizable.
- Componentes UI: Son componentes personalizados para la interfaz de usuario. Utilizados para implementar los servicios que ofrece el componente ya sea en un ambiente Web o por cualquier aplicación.
- Servicios Web, son la implementación de la lógica de negocio del componente para ser utilizados remotamente siguiendo el estándar de servicios web.

El ingeniero desarrollador puede utilizar este tipo de componente, instanciando los subcomponentes encapsulados en el propio componente, los cuales pueden ser utilizados ya sea con los EJB, los controles personalizados o invocando los servicios Web que le ofrece el componente; así el componente se podría utilizar en el desarrollo de aplicaciones Web, aplicaciones cliente e inclusive en aplicaciones móviles.

La edición empresarial de Java versión 5 proporciona un conjunto de frameworks con los cuales es posible implementar el componente propuesto. Los subcomponentes son desarrollados según el modelo de componentes Java EE 5 [7]. En la Tabla 1 se muestran las principales librerías utilizadas para desarrollar el componente software reutilizable.

TABLA I
IMPLEMENTACIÓN DEL COMPONENTE EN JEE5

Subcomponente: Entidades	
Lenguaje o Librería Java	Descripción
Api de persistencia de Java - JPA 1.0.	Contiene clases y anotaciones especiales para realizar el mapeo entre objetos y una base de datos relacional. Permite realizar la persistencia automática de las entidades. Para mayor información ver [8]
JPQL (Java Persistence Query Language, Lenguaje de consulta para persistencia en java.	Es un lenguaje de consulta parecido al SQL, que permite realizar consultas sobre las entidades. Para más detalles ver [7]
Jboss SEAM	Contiene clases y anotaciones para permitir el acople entre los diferentes niveles de una aplicación Java empresarial, es decir, permite la comunicación directa entre las entidades, EJBs. En [9] se pueden ver más detalles.
Subcomponente: Enterprise Java Beans – EJB	
Lenguaje o Librería Java	Descripción
Api EJB 3.0	Clases que permiten la creación de la lógica del negocio de una aplicación, contiene servicios de seguridad, administración, transacción y configuración de los componentes en el contenedor. Para más detalles ver [7]
Jboss SEAM	Permite el acople y comunicación entre los EJBs que encapsulan los servicios del componente y la capa de presentación, en esta caso serían los Servicios Web y los Controles personalizados para la Interfaz de Usuario. En [9] se pueden ver más detalles.
Subcomponente: Controles UI.	
Lenguaje o Librería Java	Descripción
Api java server faces (JSF)	Contiene controles para ser utilizados en la interfaz de usuario, como: combos, menús, cuadros de textos, tablas y otros. Permite la validación de los datos introducidos por el usuario, manejo de eventos y conexión con la capa de presentación y la lógica de negocio. En [7] se pueden ver más detalles.
Api richfaces CDK – Kit de desarrollo de componentes.	Richfaces es una librería de componentes para la interfaz de usuario, parecido a los JSF, pero incorporan la tecnología AJAX en los componentes. También contiene un kit de desarrollo de componentes, el cual facilita la implementación de componentes richfaces. En [10] se pueden ver más detalles.
Apache Maven	Framework que permite automatizar las tareas relacionadas con el desarrollo de componentes o de cualquier aplicación. Las tareas que permite automatizar son: compilar, copiar, distribuir, limpiar, etc.
Subcomponente: Servicios Web	
Lenguaje o Librería Java	Descripción
Api JAX-WS	Conjunto de clases y herramientas necesarias para desarrollar, probar y distribuir servicios web y sus clientes, los cuales pueden ser basados o no en la plataforma java. Para más detalles ver [7]

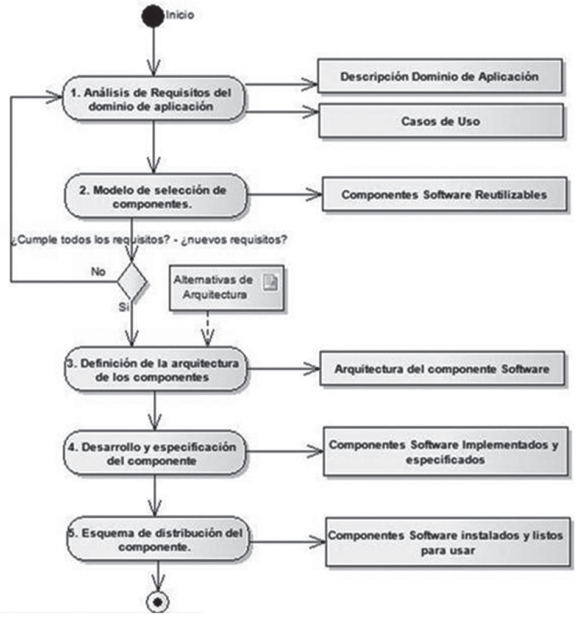
Con las anteriores librerías se puede desarrollar el componente propuesto. En esta investigación se desarrollaron componentes de este estilo y fueron utilizados en el desarrollo de software empresarial de la División de Servicios de Información de la Universidad Industrial de Santander (UIS), con resultados positivos.

IV. EL PROCESO DE DESARROLLO DE COMPONENTES SOFTWARE PROPUESTO

En [13] se planteó la investigación realizada para proponer el proceso de desarrollo de componentes software reutilizables, se hizo un estudio de las metodologías de desarrollo software y de la ISBC para proponer el presente modelo, después de tener planteado el proceso se utilizó en el desarrollo de software empresarial en la División de Servicios de Información de la UIS, con lo que se pudo depurar cada una de las fases, para llegar a los resultados que se presentan a continuación.

Los fundamentados principales del proceso se centran en los planteamientos realizandos en Pressman [1], Iribarne Martínez [6], Weitzenfeld [11], Bruegge y Dutoit [12]. Las fases propuestas para el proceso de desarrollo se pueden apreciar en la Fig. 4 donde se encuentra un diagrama de actividades UML del proceso y se destacan los resultados principales de cada actividad.

FIG. 4. EL PROCESO DE DESARROLLO DE COMPONENTES SOFTWARE REUTILIZABLES



A. Análisis de requisitos del área de aplicación

En esta fase se pretende establecer el dominio de aplicación y el modelo de requisitos para la biblioteca de componentes que se pretende crear. Para establecer el dominio de aplicación es importante identificar y delimitar las áreas posibles de aplicación, estas áreas se definen dependiendo de la necesidad que la empresa de desarrollo de software tenga para implementar componentes y su grado posible de reutilización por los desarrolladores de software.

Después de seleccionar las áreas posibles de aplicación es importante evaluar los siguientes criterios (Grado de utilización, dificultad de implementación, complejidad, tolerancia al cambio, nivel de abstracción, necesidad de implementación y posibilidad de encontrar componentes ya desarrollados), para poder decidir el dominio de aplicación de la biblioteca de componentes.

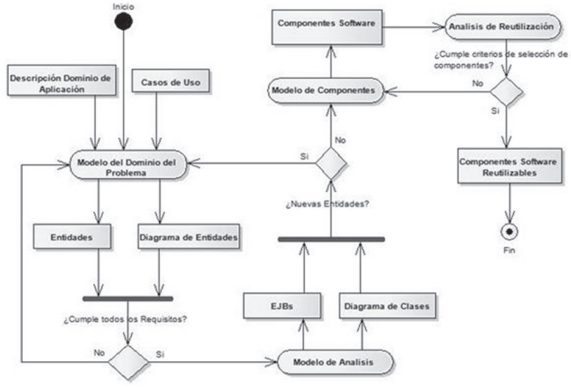
Para cada criterio se define una escala y se da un puntaje de evaluación. Esta evaluación la debe realizar el arquitecto software de la organización; para el caso, el jefe de la DSI. Los resultados de esta fase son la descripción del dominio de aplicación y el modelo de casos de uso.

B. Modelo de selección de componentes

Con el modelo de selección de componentes se pretende elegir los componentes que se van a implementar para el dominio de aplicación establecido, de acuerdo con los casos de uso definidos en fase anterior. En la Fig. 5 se presenta el diagrama de actividades del modelo.

Los resultados de esta fase son los componentes software reutilizables candidatos para ser implementados.

FIG. 5. MODELO DE SELECCIÓN DE COMPONENTES



C. Definición de la arquitectura

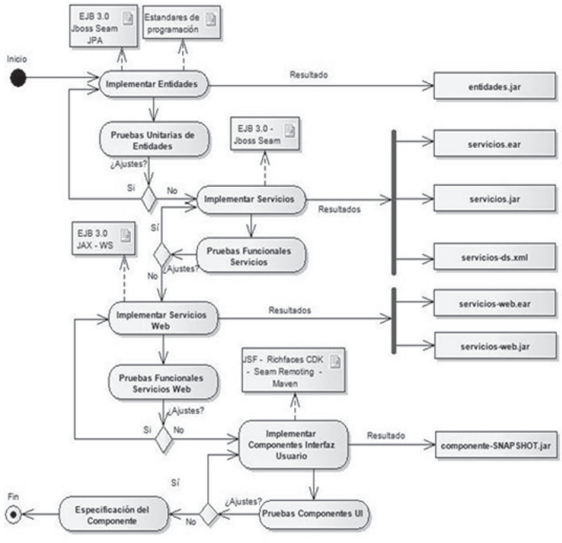
Una vez definidos y seleccionados los componentes, es necesario definir el estilo arquitectónico que se va a utilizar en su implementación.

Desde el punto de vista de la ISBC, la arquitectura software de los componentes define la forma como se distribuyen e interrelacionan los elementos estructurales del componente (Entidades, EJBs, Componentes UI y Servicios Web) y define el medio de comunicación entre ellos. Del análisis de los estilos arquitectónicos y de la arquitectura de JEE5, se pueden definir las siguientes alternativas de arquitectura: 1. Modelo – Vista – Controlador, 2. Arquitectura por capas, 3. Arquitectura Orientada a Servicios (SOA), que se detallan en Vera y Rojas [13].

D. Procedimiento de desarrollo, pruebas y especificación de los componentes

Con el procedimiento de desarrollo, pruebas y especificación se pretende definir los pasos necesarios para implementar los componentes software reutilizables en JEE5, para garantizar su calidad y describir detalladamente las funcionalidades, propiedades y elementos del componente por medio de su especificación. En la Fig. 6 se puede apreciar el procedimiento propuesto.

FIG. 6. PROCEDIMIENTO DE DESARROLLO, PRUEBAS Y ESPECIFICACIÓN DE LOS COMPONENTES



E. Distribución de los componentes

Los resultados de la implementación del componente son: archivos de tipo jar para las entidades y para la implementación de los servicios; ear para la implementación de los servicios, un archivo de configuración de la conexión con la base de datos dataSource.xml, archivos jar para la implementación de los servicios Web y para la implementación de los componentes personalizados para la interfaz de usuario, JavaDocs de las clases java implementadas, una ficha de especificación y modelos UML del componente.

Para la distribución se debe crear un archivo build.xml con la herramienta ANT por medio del cual se pueda instalar en el servidor de aplicaciones el componente de forma automática, luego empaquetar los archivos en un archivo *.tar o *.zip para ser colocado en el servidor de versiones de la organización. En el servidor de versiones de la empresa de desarrollo el programador encontrará un único archivo con toda la documentación y especificación del componente.

V. CONCLUSIONES

La investigación planteó la elaboración de componentes, por medio de los cuales el desarrollo de los sistemas informáticos se realice más eficientemente y con mejor calidad que de la forma tradicional; fortaleza que fue aprovechada para el desarrollo del proyecto de las nuevas versiones de los sistemas de información Académica, Financiera y de Recursos Humanos de la UIS en Java versión empresarial 5; estos sistemas dan soporte a la toma de decisiones en los procesos más importantes que se llevan a cabo en la Universidad.

Se pudo evidenciar que por medio de la utilización de componentes software reutilizables la competitividad de la DSI mejora notoriamente, porque:

La producción de aplicaciones software se hace con mayor rapidez, habida cuenta que los componentes reutilizables ya creados, se usan en muchos proyectos de desarrollo sin necesidad de implementarlos de nuevo.

Los sistemas informáticos ofrecidos a la comunidad universitaria se basan en componentes ya creados, probados y estandarizados que reúnen las mejores prácticas de desarrollo, logrando así una mejor calidad en el producto.

La reutilización de software trae beneficios muy importantes para la actividad de desarrollo profesional de software entre los cuales se encuentran:

- Oportunidad: Se tiene menos software que hacer y se puede construir con mayor rapidez.
- Mantenimiento: Disminución de los esfuerzos de mantenimiento.
- Fiabilidad: Al tener buenos componentes ya usados y probados con anterioridad se puede mejorar la calidad y la oportunidad en el desarrollo de nuevos sistemas.
- Eficiencia: Los componentes son realizados por expertos en el dominio de aplicación en el que se desarrollaron, mediante los mejores algoritmos, patrones y estructuras de datos.
- Inversión: El componente no se desarrolla sólo para un proyecto, se puede utilizar en muchos proyectos, por lo que se reduce el tiempo de desarrollo del proyecto y talento humano asignado.

Con la elaboración del proceso de desarrollo de componentes software reutilizables se beneficiarían principalmente las empresas desarrolladoras de software empresarial. En el contexto de este trabajo, la División de Servicios de Información de la UIS, dispone ahora de una forma más detallada, basada en el estándar Java EE 5, para elaborar componentes software reutilizables que se puedan usar en el desarrollo de los sistemas empresariales.

Con el desarrollo de los componentes software reutilizables se garantiza la reutilización de lógica del negocio de la empresa de desarrollo de software en muchos proyectos, el componente no se centra sólo en resolver un problema específico para la interfaz de usuario, sino que por el contrario se puede reutilizar un conjunto de requisitos software en varios proyectos de desarrollo.

La ISBC es una disciplina que se encuentra en evolución; actualmente se desarrollan muchas investigaciones y tiene un futuro notable debido a sus grandes beneficios que trae para la industria desarrolladora de software.

El proceso de desarrollo de la ISBC se centra en dos grandes actividades: la ingeniería de dominio, que es la encargada de analizar, desarrollar y administrar la biblioteca de componentes reutilizables; la otra actividad es el desarrollo basado en componentes, que enmarca las actividades necesarias para ensamblar sistemas informáticos a partir de la biblioteca de componente reutilizables, además el desarrollo basado en componentes diseña nuevos componentes a partir de los requisitos particulares de un sistema nuevo.

La ISBC se adapta completamente a las metodologías de desarrollo de software tradicionales, aunque se realizan cambios sustantivos en las fases de diseño e implementación, debido a la utilización de componentes reutilizables.

Las técnicas de análisis y diseño de componentes reutilizables se basan en los mismos principios y conceptos que forman parte de las buenas prácticas de ingeniería del software.

EL modelo de objetos suele ajustarse a uno o más estándares de componentes (por ejemplo, OMG/CORBA) que definen la forma en que una aplicación puede acceder a los objetos reutilizables. Los esquemas de clasificación capacitan al desarrollador para hallar y recuperar componentes reutilizables y se ajustan a un modelo que identifica conceptos, contenidos y contextos.

La ISBC permite hacer más económico el desarrollo de sistemas informáticos, ya que permite construir menos y reutilizar más. La dificultad que tiene la ISBC se centra en la construcción de la biblioteca de componentes, en la adaptación e integración de los componentes reutilizables, otra dificultad es la falta de un estándar universal para la especificación, comunicación e integración de componentes, existen varios modelos de componentes de diferentes empresas.

REFERENCIAS

[1] Roger S. Pressman. Ingeniería del Software un enfoque práctico. Quinta Edición. Mc. Graw Hill. 2002. pp 473.

[2] Szyperski, C. Component Software. Beyond Object-Oriented Programming. Addison-Wesley. 1998.

[3} Meyer, B. The Significance of Components. Beyond Objects column, Software Development. 1999.

[4] Object-Oriented Analysis and Design Using UML. Sun Microsystems. 2003

[5] Hyung Cho y John D. McGregor. IEEE – 2005 Artículo: Component Specification for Enterprise software Development on Web Services Environment.

[6] Iribarne Martínez, Luis F. Un Modelo de Mediación para el Desarrollo de Software basado en Componentes COTS. Tesis Doctoral. Universidad de Málaga. España. 2003.

[7] SUN MICROSYSTEM, The Java EE5 tutorial. 2008.

[8] DEBU, Panda; REZA, Rahman y DEREL Lane. EJB in Action. Manning Publications Co. 2007.

[9] DAN, Allen; Seam in Action. Manning Publications. 2008.

[10] SERGEY, Smirnov. CDK Developer Guide. Jboss Comunitiy. 2008.Transl. J. Magn. Jpn., vol. 2, pp. 740-741, August 1987 [Dig. 9th Annual Conf. Magn. Jpn., p. 301, 1982].

[11] A. Weitzenfeld. “Ingeniería de Software Orientada a Objetos, Teoría y Práctica con UML y Java”. Editorial Itam. México. 2005 pp 139 .

[12] B. Bruegge, A. H. Dutoit. “Object-Oriented Software Engineering Using UML”. Patterns and Java. 2ª. ed. Ed. Prentice Hall. 2004. Pp 223 - 300 .

[13] VERA R, Fredy H. ROJAS, Fernando. Artículo: Propuesta de un procedo de desarrollo de componentes software reutilizables. Revista Gerencia Tecnológica Informática. Volumen 7 – Número 19. Diciembre 2008.

[14] MEYER, Bertrand. Construcción de Software Orientado A Objetos. Prentice-hall 2002.