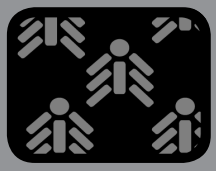
PROPUESTA DE UN PROCESO DE DESARROLLO DE COMPONENTES SOFTWARE REUTILIZABLES



AUTOR
Mg(c) VERA RIVERA, Fredy Humberto
Magíster(c) en Ingeniería de Sistemas
e Informática.
División de Servicios de Información – UIS.
Grupo Inv. Sistemas y Tecnologías
de la Información.
frehuver@uis.edu.co, freve9@hotmail.com
COLOMBIA

AUTOR
Mcc. ROJAS MORALES, Fernando
Magíster en Ciencias Computacionales.
Profesor Escuela de Ing. de Sistemas - UIS.
Grupo Inv. Sistemas y Tecnologías
de la Información.
frojas@uis.edu.co
COLOMBIA

Fecha de Recepción del Articulo

Fecha de Aceptación del Articulo

Articulo Tipo 1: Artículo de Investigación Científica y Tecnológica.

RESUMEN

El presente artículo corresponde a un avance de la investigación para obtener el título de magister titulada: "Propuesta de un proceso de desarrollo de componentes software reutilizables", mediante la cual se busca establecer los pasos necesarios para crear componentes software reutilizables en Java Edición Empresarial 5 (Java EE 5). En primer lugar se hace una introducción planteando la problemática que se evidencia en el desarrollo de software empresarial y cómo la Ingeniería del Software Basada en Componentes (ISBC) puede ayudar a resolverla; se aclara la definición de componente y se plantean las preguntas de investigación. Posteriormente se explica la metodología utilizada en la investigación que comprende la investigación descriptiva e investigación tecnológica aplicada. Después dentro de los resultados de la investigación se plantea la estructura de un componente software reutilizable siguiendo el modelo de componentes de Java, el cual consta principalmente de Entidades (pojos, antiguos EJB de entidad), EJBs (de sesión o manejador de mensajes), componentes o controles personalizados para la interfaz de usuario y servicios web que exponen las funcionalidades encapsuladas en los EJBs como servicios web. Después se propone un modelo de selección de componentes software reutilizables y por último se establecen las alternativas de arquitectura que se pueden utilizar para implementar este tipo de componentes, dentro de estas arquitecturas se pueden definir: la arquitectura por capas, arquitectura modelo – vista – controlador y la arquitectura orientada a servicios.

PALABRAS CLAVES

Ingeniería del Software Basada en Componentes.
Componentes Software Reutilizables.
Modelo de Componentes.
Enterprise Java Bean.
Servicios Web.

ABSTRACT

This article is a look ahead to the research to obtain the master degree: "Proposal of a process of the development or reuse software components" by which it is wanted to set the necessary steps to create reuse software components in Java Enterprise Edition 5 (Java

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

EE 5). First of all, an introduction is made to set out the problem that is evident in the development of enterprise software and how the Component Based Software Engineering (CBSE) can help to solve it; the definition of component is clarified and the research questions are set. Later it is explained the methodology used in the research that comprises the descriptive research and applied technologic research. After in the result of the research, it is set up the structure of a reuse software component following the component model of Java, which consist mainly of Entities (pojos, old EJB entities), EJB (session beans, message driver bean), components or personalized controllers for the user interface and web services that present the covered operations in the EJBs like web services. Then it is proposed a model of selection of reuse software components and lastly it is established the alternatives of architecture that can be used to introduce this kind of components. Among these architectures it can be defined: multi – tired, model – view – controller and services – based architecture.

KEYWORDS

Component – Based Software Engineering.
Reuse software component.
Component model.
Enterprise Java Bean.
Web Services.

INTRODUCCIÓN

A lo largo de los últimos años el desarrollo de los sistemas informáticos ha venido evolucionando de tal forma que la complejidad en los lenguajes, en las herramientas utilizadas para el desarrollo y en el mismo proceso de desarrollo se ha evidenciado. Esta complejidad trae diferentes problemas que se pueden apreciar en los proyectos de desarrollo de software de muchas empresas de nuestro medio:

- Los proyectos no terminan en plazo planeado.
- Los proyectos no se ajustan al presupuesto inicial.
- Baja calidad del software generado.
- Software que no cumple las especificaciones.
- Código inmantenible que dificulta la gestión y evolución del proyecto.

Los métodos para estimar el esfuerzo requerido de tiempo y personal necesario para un proyecto software no han demostrado ser eficientes. Cuando se fijan plazos normalmente no se cumplen, y a esto se añade que las aplicaciones de hoy son mucho más complejas de lo que solían ser hace sólo una década, haciéndolas inmanejables por una sola persona. [11]

Dentro del conjunto de herramientas que ofrece la Ingeniería de Software para reducir esta problemática, se encuentra la Ingeniería del Software Basada en Componentes (ISBC), que se centra en el diseño y construcción de sistemas basados en computadora que utilizan componentes software reutilizables [1]. La ISBC busca entonces construir los sistemas informáticos utilizando componentes ya elaborados que se puedan reutilizar en varios proyectos de desarrollo, reduciendo así el tiempo de desarrollo, aumentado la calidad del software generado y haciendo más fácil el mantenimiento.

En la figura 1 se puede apreciar una representación de la idea principal de la ISBC, donde los sistemas empresariales se construyen buscando los componentes software en el repositorio, se combinan y adaptan estos componentes, en vez de estar codificando y construyendo las mismas aplicaciones cada vez.

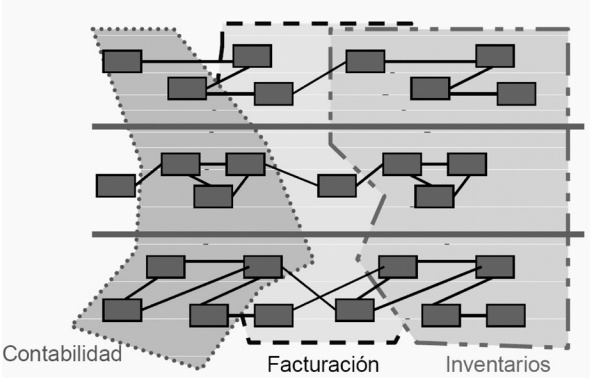


Figura 1. Desarrollo de Software basado en componentes reutilizables.

Sobre la base del objetivo primario de la ISBC, es importante aclarar la definición de componente software reutilizable y los requisitos que debe tener un elemento software para poder considerarlo como componente.

Definición de Componente Software: 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", en otras palabras un componente es una unidad software utilizada para ensamblar o componer un sistema más grande y complejo, debe contener una especificación que permita identificarlo y reutilizarlo.

Adicional a esta definición y teniendo en cuenta las propuestas por Meyer [3], Sun Microsystem [4] e Iribarne Martínez [5], se puede llegar determinar que

un elemento software, para poder clasificarse como componente software reutilizable, debe cumplir con los siguientes requisitos:

- Debe ser una unidad software sin dependencia estructural de otras unidades.
- Debe tener un alto grado de cohesión y un bajo acoplamiento.
- 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.

Con el desarrollo de esta investigación se pretende crear un proceso de desarrollo de componentes software basado en el estándar Java EE 5, para ser utilizado en el desarrollo de software empresarial principalmente para la División de Servicios de Información de la Universidad Industrial de Santander (UIS), y poder resolver así la problemática que se presenta en el desarrollo de los sistemas informáticos. En la propuesta se pretende hacer el estudio de los pasos que se deben llevar a cabo para poder implementar de forma adecuada los componentes reutilizables utilizando el estándar Java EE 5. En la figura 2 se puede apreciar una propuesta de diagrama de componentes del sistema de información de Recursos humanos de la UIS.

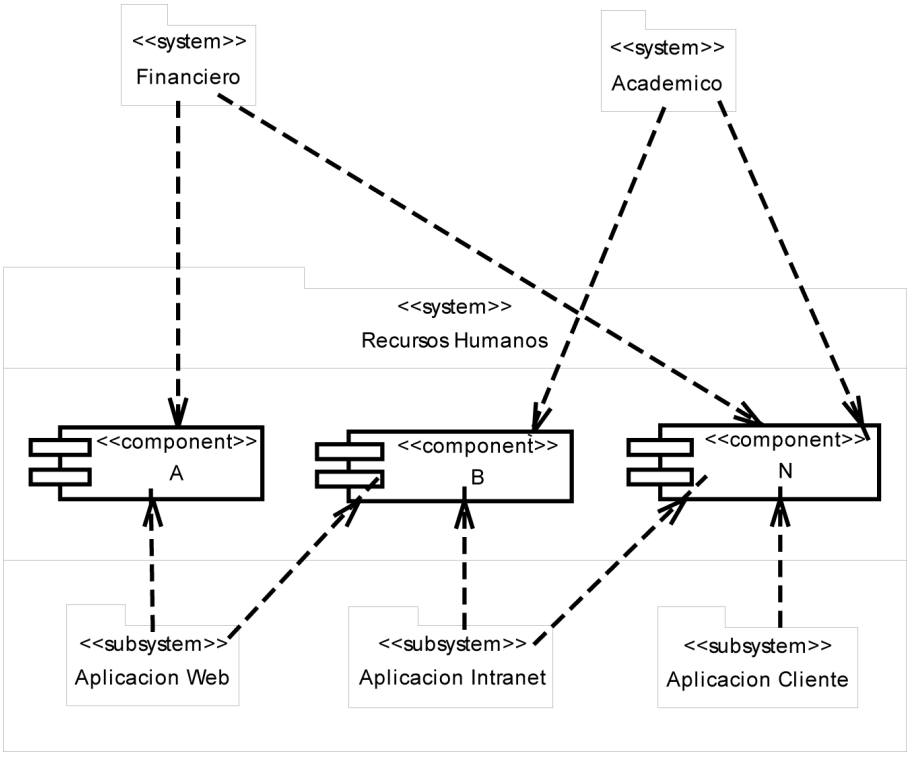


Figura 2. Propuesta Diagrama de Componentes Sistema de Recursos Humanos de la UIS.

En el diagrama, las aplicaciones informáticas del sistema de Recursos Humanos se basan en componentes ya creados, utilizando las funciones y facilidades que cada componente ofrece. También las aplicaciones de los otros sistemas de información pueden utilizar estos componentes.

En Pressman [1], en Drake, Medina, y González [14] e Iribarne Martínez [5], donde se analiza el ciclo de vida de la ISBC, se puede apreciar una división importante de este ciclo de vida en dos grupos fundamentales: Ingeniería de Dominio y Desarrollo de basado en componentes, la ingeniería de dominio pretende la creación de la biblioteca de componentes reutilizables.

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

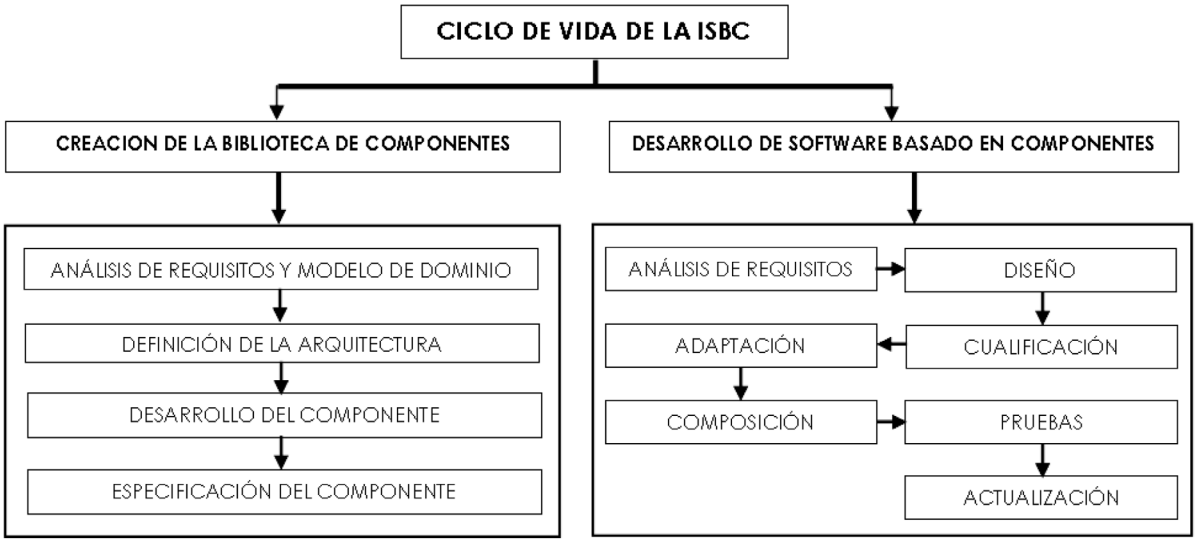


Figura 3. Ciclo de Vida de la ISBC

Con base en la fundamentación conceptual presentada se formula el siguiente problema de investigación y sus preguntas asociadas:

Problema de Investigación:

¿Cómo elaborar componentes software reutilizables que sirvan de apoyo al proceso de desarrollo software?

Pregunta de Investigación 1:

Es necesario definir los criterios de selección de los componentes, para saber identificar que puede ser o no un componente, poder identificar que funcionalidades se pueden reutilizar en otros desarrollos para ser encapsuladas dentro de un componente. Conforme a lo enunciado nace el primer interrogante a resolver:

¿Cómo identificar los componentes candidatos a implementar y cuáles son los criterios de selección de estos componentes?

Pregunta de Investigación 2:

Para el desarrollo de los componentes reutilizables se requiere definir su modelo estructural, es decir, se requiere definir la forma como se pueden interrelacionar los elementos que forman el componente, por lo tanto surge la siguiente pregunta de investigación:

¿Qué alternativas de arquitectura se pueden utilizar para el desarrollo de los componentes reutilizables?

Pregunta de Investigación 3:

Otros asuntos importantes para el desarrollo de componentes son la forma como se implementan de modo que se puedan reutilizar y la manera de realizar su especificación. Las técnicas que existen actualmente para la especificación de los componentes y de las funcionalidades que ofrece no están muy claras, como lo dicen Jean-Guy Schneider y Jun Han en su artículo [6], por lo tanto se puede formular otro interrogante para el estudio a realizar.

¿Cómo se implementarían y especificarían los componentes software, de tal forma que se pueda hacer fácil su reutilización?

Pregunta de Investigación 4:

Una vez implementado y especificado el componente es necesario definir el modo como los desarrolladores lo pueden utilizar, es importante saber la forma de distribuir el componente, y de aquí surge el último interrogante de la presente investigación:

¿Cuál es el esquema de distribución y de utilización más adecuado de los componentes, de tal forma que se pueda reutilizar en varios proyectos de desarrollo?

Se puede apreciar que las preguntas de investigación corresponden a fases del ciclo de vida de la creación de la biblioteca de compontes, resolviendo cada pregunta se obtiene la propuesta del proceso de desarrollo de componentes.

1. METODOLOGÍA

La metodología del presente trabajo de investigación hace uso de conceptos de la investigación descriptiva y de la investigación tecnológica aplicada.

Por medio de la investigación descriptiva se pretende describir y analizar el proceso de desarrollo de componentes software reutilizables, estudiando los métodos que existen en la actualidad para el desarrollo de los componentes, analizando y definiendo los pasos que se deben llevar a cabo para crear los componentes, además de los beneficios y desventajas que tiene este nuevo paradigma de desarrollo.

En cuanto a la investigación tecnológica aplicada, la cual aplica los conocimientos a la solución de un problema práctico inmediato generalmente particular, se concreta en el diseño e implementación de un componente software reutilizable para la División de Servicios de información de la UIS, teniendo como base los fundamentos descritos en la parte de la investigación descriptiva.

1.1 Proceso de investigación

El proceso de investigación define las actividades a realizar en el desarrollo de la investigación. Las fases fundamentales que se van a llevar a cabo se detallan en la figura 4. Se inicia la investigación con recopilación de información necesaria para tener los fundamentos teóricos de la investigación, después se realiza un diagnostico del problema que se pretende resolver, esta fase se realiza iterativa e incrementalmente junto con las siguientes fases: Estudio de alternativas para el desarrollo de componentes, Diseño e implementación del componente, y Distribución del componente; hasta resolver las preguntas de investigación planteadas y cumplir con los objetivos. Por último se realiza la elaboración del informe final y la divulgación de los resultados.

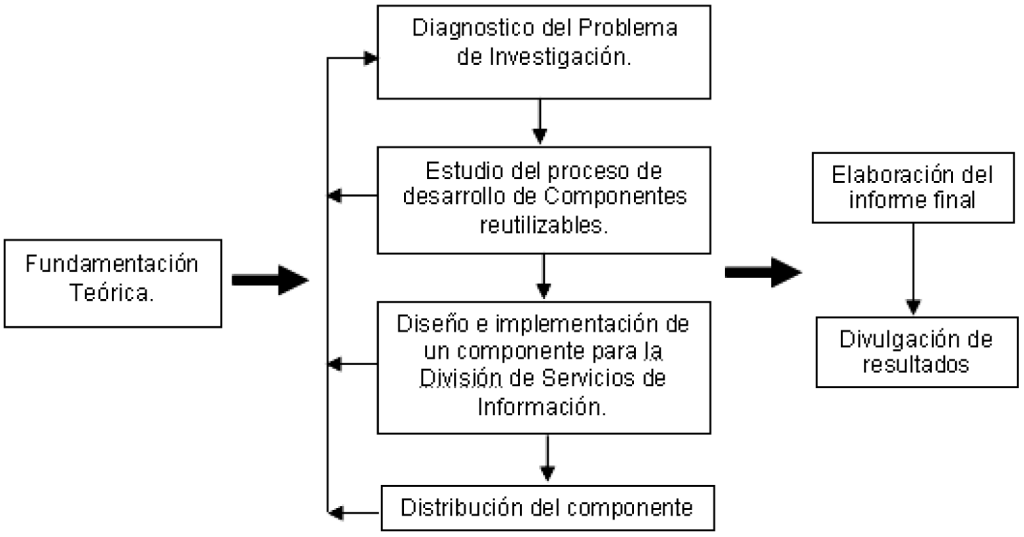


Figura 4. Proceso de Investigación

2. RESULTADOS

En primer lugar es importante aclarar cómo son los componentes software reutilizables que se pretenden implementar, de tal forma que se pueda entender el alcance y la complejidad de la investigación. La figura 5 describe el contenido y las características fundamentales de un componente software reutilizable.

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

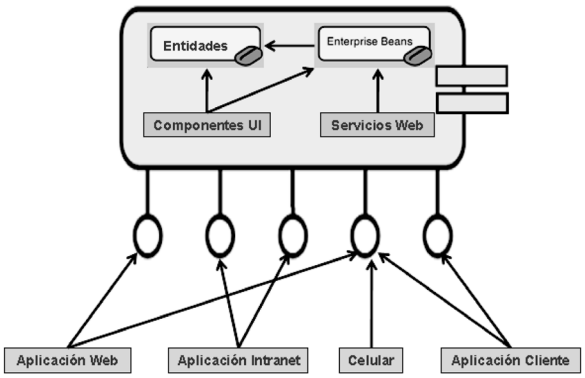


Figura 5. Componente Software Reutilizable

El componente de estudio de la presente investigación mostrado en la figura 5, es un conjunto de subcomponentes que siguen el modelo de componentes de Java, los cuales se detallan a continuación:

- **Entidades** o antiguos EJB de Entidad, son las que contienen el modelo del dominio de aplicación del componente, y manejan la persistencia y almacenamiento en la base de datos.
- **Enterprise Java Beans (EJB)**, son los encargados de manejar o controlar la lógica del negocio del dominio de aplicación para el cual será implementado el componente y modelan los servicios que va a prestar el componente.
- **Componentes UI:** Son componentes o controles personalizados para la interfaz de usuario. Utilizados para implementar los servicios que ofrece el componente ya sea en una aplicación Web, aplicación intranet o por cualquier aplicación que utilice controles.
- **Servicios Web,** implementan la lógica de negocio del componente para ser utilizados remotamente ya sea por otros sistemas o por aplicaciones que requieran esos servicios. Se encargan de exportar las funcionalidades ofrecidas por los EJBs como servicios web.

Con la implementación de este tipo de componentes, al ingeniero desarrollador se le presentan un conjunto de subcomponentes encapsulados en el propio componente, los cuales pueden ser utilizados independientemente de la forma como el desarrollador crea conveniente, puede utilizar las entidades, los EJB, los controles personalizados o invocar 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.

Los subcomponentes son desarrollados siguiendo el modelo de componentes Java EE5 [7], en el cual se pueden encontrar los frameworks necesarios para implementar el componente: Para las entidades, se encuentra el API de persistencia de Java, que es el encargado de controlar la comunicación de las entidades con la base de datos, para los Enterprise Beans el API EJB 3.0, que permite la creación de los controladores de la lógica del negocio para la cual es creado el componente; para los componentes UI se encuentra el API Java Server Faces, y también se detalla el framework para implementar Servicios Web en Java. Con este conjunto de frameworks que ofrece Java EE 5 es posible implementar este tipo de componentes software reutilizables.

2.1 Definición del proceso de desarrollo de componentes.

Teniendo identificada la forma y el contenido del componente software reutilizable, se puede definir la propuesta del proceso de desarrollo de componentes software reutilizables, el cual tiene su fundamento en Pressman [1], Iribarne Martínez [5], Weitzenfeld [8], Bruegge y Dutoit [9]. Las fases propuestas para el proceso de desarrollo son las siguientes: Análisis de requisitos del dominio de aplicación, Modelo de selección de componentes para el dominio de aplicación, Definición de la arquitectura, desarrollo y especificación del componente y por último el esquema de distribución del componente, la propuesta se puede apreciar mejor en la figura número 6 donde se encuentra el diagrama de actividades UML del proceso y se destacan los resultados principales de cada actividad.

La presente investigación hace parte del proyecto de desarrollo de las nuevas versiones de los sistemas Académico, de Recursos Humanos y Financiero de la División de Servicios de Información de la UIS, actualmente se está trabajando en la creación de los componentes generales□ del sistema de Recursos Humanos, utilizando la presente propuesta consiguiendo así probar su efectividad, refinar el proceso de desarrollo de componentes y obtener componentes reales para ser utilizados en el desarrollo de los sistemas informáticos de la Universidad, también al final de la investigación se tendrá por lo menos un caso de estudio documentado para su publicación.

A continuación se describen las fases del proceso de desarrollo de componentes planteado.

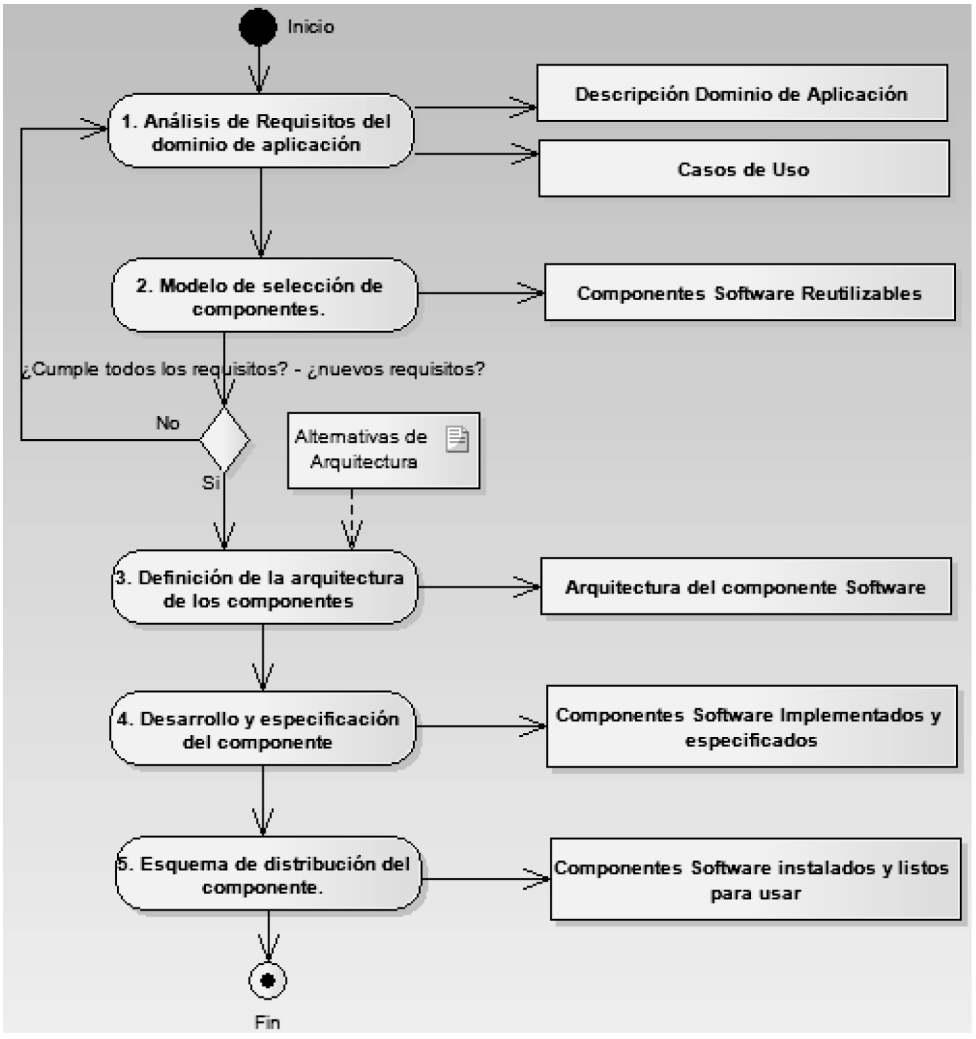


Figura 6. Propuesta del proceso de desarrollo de componentes software reutilizables.

2.2 Análisis de requisitos del dominio 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.

Weitzenfeld dice que "el propósito del modelo de requisitos es comprender completamente el problema y sus implicaciones". El modelo de requisitos se centra principalmente en la información del dominio del problema y en identificar y establecer las funcionalidades que se van a ofrecer con los componentes. Con el modelo de requisitos se pretende:

- Tener una descripción clara del dominio de aplicación.
- Identificar los actores y principales clientes de las funcionalidades del dominio de aplicación.
- Identificar los casos de uso del dominio de aplicación.
- Identificar las restricciones, precondiciones, postcondiciones y requerimientos no funcionales para los casos de uso.
- Analizar las posibilidades de reutilización tanto a nivel de aplicaciones del mismo dominio, como a aplicaciones externas al dominio de aplicación.

Los resultados de este modelo són: la descripción detallada del dominio de aplicación y los diagramas de casos de uso.

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

2.3 Modelo de selección de componentes software reutilizables.

Con el modelo de selección de componentes se pretende elegir los componentes a implementar para el dominio de aplicación establecido, teniendo en cuenta los casos de uso definidos en la fase anterior.

El modelo de selección de componentes se puede basar en técnicas comunes de Ingeniería de software, más específicamente en el proceso de desarrollo de software orientado a objetos propuesto por Weitzenfeld en [8], haciendo las adaptaciones correspondientes.

La figura 7 representa el diagrama de actividades del modelo de selección de componentes propuesto, el cual consta de 5 actividades principales: modelo de requisitos, modelo del dominio del problema, modelo de análisis, modelo de componentes y análisis de reutilización.

Con el modelo de selección de componentes se pretende identificar los componentes candidatos a implementar y establecer los criterios de selección de componentes para que sean reutilizables, para un dominio de aplicación establecido.

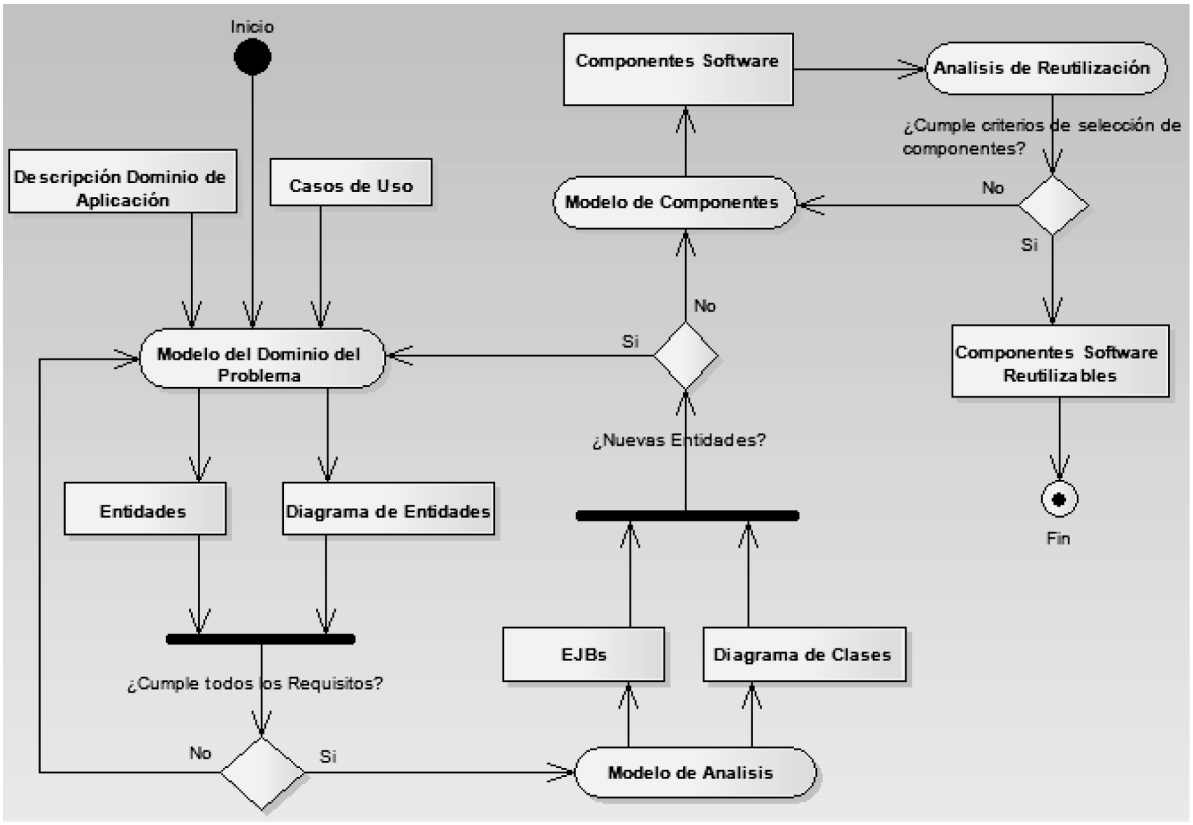


Figura 7. Modelo de Selección de Componentes.

A continuación se detallan las actividades que comprende el modelo de selección de componentes:

2.3.1 Modelo del Dominio del problema.

Según Weitzenfeld, el modelo del dominio del problema define un modelo de clases común para todos los involucrados. El modelo de dominio se centra principalmente en la identificación de entidades. Una entidad es un objeto de dominio que tiene persistencia en la base de datos. Para identificar las entidades se hace uso de una técnica de ingeniería de textos así:

- Resaltar los sustantivos en la descripción del dominio de aplicación.
- Obtener un listado de entidades candidatas.
- Seleccionar las entidades que tengan significado para el dominio de aplicación que se está trabajando.
- Eliminar las entidades que correspondan aspectos

de la interfaz de usuario y las que correspondan a actores del sistema.

Los resultados de este modelo son las entidades del dominio de aplicación y el diagrama de entidades.

El diagrama de entidades establece:

- Atributos y llaves primarias de las entidades.
- Las relaciones existentes entre las entidades.
- La multiplicidad y roles en las relaciones.
- La direccionalidad de las relaciones.

2.3.2 Modelo de análisis.

En este modelo se pretende establecer los Enterprise Java Bean (EJB) del dominio de aplicación. Los EJB encapsulan la lógica del negocio de la aplicación y actúan como los controladores del componente. Para el modelo de análisis es importante tener en cuenta qué:

- Se debe definir por lo menos un EJB por caso de uso. Dependiendo de la complejidad del caso de uso se pueden utilizar varios EJB.
- Dependiendo de los requerimientos funcionales y no funcionales del caso de uso se decide el tipo de EJB a utilizar: con estado, sin estado o manejador de mensajes.
- Se pueden manejar varios tipos de EJB para un mismo caso de uso, depende de los requerimientos del caso de uso.
- Se deben encapsular los servicios del caso de uso.
- Se deben identificar las interfaces locales y remotas para cada EJB según el caso de uso.
- Se debe adicionar o eliminar las entidades que se estime conveniente.

Los resultados de este modelo son los EJBs y el diagrama de clases donde se relacionan los EJBs y las entidades que manejan; de este diagrama se puede apreciar la dependencia estructural de cada EJB.

2.3.3 Modelo de componentes.

Se utiliza para definir los componentes software candidatos a implementar, se encarga de encapsular las entidades y los EJB dentro de un solo componente. Para realizar esta encapsulación se debe tener en cuenta qué:

- Se deben agrupar los EJB y entidades que manejen información similar y se puedan reutilizar.
- Si se encuentran relaciones de generalización, agregación o composición entre las entidades identificadas, estas entidades deben ir en el mismo componente.
- Las transacciones se deben manejar en lo posible dentro de un solo componente.
- Los EJBs o las entidades que son altamente acopladas deben ir en el mismo componente.

Los resultados de esta actividad son los componentes software candidatos a implementar y un diagrama de componentes que muestra la interacción entre ellos.

2.3.4 Análisis de reutilización.

En esta actividad se pretende asegurar que los componentes resultantes sean reutilizables, para esto es necesario definir unos criterios de selección de componentes. Bruegge y Dutoit [9] definen los objetivos de diseño en su proceso de desarrollo software, los cuales se pueden adaptar para definir los criterios de reutilización y de selección de los componentes. También Bertoa y Vallecillo [15] hacen una adaptación del modelo de calidad ISO 9126 aplicado a los componentes, de los cuales se pueden tomar algunos criterios propuestos por ellos para aplicarlos en los criterios de selección.

En la tabla 1 se pueden ver los criterios de selección de componentes, para cada criterio se presenta su descripción y una escala para realizar su evaluación. La evaluación se realiza con una escala de 1 – 5 indicando el grado de cumplimiento del criterio.

Tabla 1. Criterios de selección de Componentes.

Criterio	Descripción	Evaluación
Dependencia estructural.	Se determina por medio de la evaluación del grado de cohesión y el grado de acoplamiento del componente. Debe tener un alto grado de cohesión y un bajo acoplamiento para que la reutilización sea más fácil de obtener.	1- Muy alta. 2- Alta 3- Media 4- Baja 5- Muy baja

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

Facilidad mantenimiento	El mantenimiento del componente debe ser transparente, debe afectar lo menos posible a los clientes que lo utilizan. Se mide el grado de dificultad del mantenimiento.	1-Muy Difícil 2- Difícil. 3- Media. 4- Fácil. 5-Muy fácil.
Grado de Adaptabilidad.	Determina la capacidad que tiene el componente para adaptar las funcionalidades o servicios que presta a situaciones imprevistas o no manejadas. El grado de adaptabilidad debe ser alto.	1-Muy Bajo. 2- Bajo. 3- Medio. 4- Alto. 5- Muy alto
Grado extensibilidad.	Determina el grado de dificultad y el impacto de añadir nuevas funcionalidades al componente. Para obtener la reutilización este grado de dificultad debe ser bajo.	1- Muy Alto. 2- Alto. 3- Medio. 4- Bajo. 5- Muy bajo
Grado de Utilización.	Determina el número de posibles clientes o usuarios del componente.	1-Muy Bajo. 2- Bajo. 3- Medio. 4- Alto. 5- Muy alto
Portabilidad.	Determina la capacidad que tiene el componente de utilizarse en diferentes ambientes al que fue implementado. Si se quiere que el componente tenga alta reutilización se debe poder utilizar en diferentes lenguajes, plataformas y sistemas operativos.	1-Muy Baja. 2- Baja. 3- Media. 4- Alta. 5- Muy alta.
Complejidad.	Determina el grado de complejidad de la lógica que maneja el componente. Entre más complejo sea, más difícil será su implementación.	1- Muy Alta. 2- Alta. 3- Media. 4- Baja. 5- Muy baja
Cambiabilidad.	Define la posibilidad de cambio de los requisitos para los cuales fue implementado el componente.	1- Muy Alta. 2- Alta. 3- Media. 4- Baja. 5- Muy baja
Posibilidad de componente ya desarrollado.	Define la posibilidad de encontrar un componente ya desarrollado que se adapte a la mayoría de las funcionalidades que va a manejar el componente a implementar.	1- Muy Alta. 2- Alta. 3- Media. 4- Baja. 5- Muy baja

Después de evaluar cada criterio de selección se suma los valores dados para obtener una clasificación con los componentes ordenados de forma descendente. Los componentes para ser candidatos a implementar deben tener una calificación mayor a 30 puntos.

Los resultados de esta actividad son los componentes software reutilizables listos para ser implementados.

2.4 ALTERNATIVAS DE ARQUITECTURA DE LOS COMPONENTES SOFTWARE REUTILIZABLES

Una vez definidos y seleccionados los componentes, es necesario definir el estilo arquitectónico que pueden seguir para poderlos implementar correctamente.

Bruegge y Dutoit [9] definen el término arquitectura como la forma de organizar e interrelacionar los diversos componentes de un sistema informático. La arquitectura del sistema incluye descomposición del sistema, control del flujo global, manejo de condiciones de frontera y protocolos de intercomunicación de subsistemas.

Desde el punto de vista de la ISBC y para la presente investigación 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) y define el medio de comunicación entre los componentes. Los componentes software reutilizables que se pretenden desarrollar se deben basar en el modelo arquitectónico Java EE 5 y deben seguir el modelo de componentes Java EE 5. La arquitectura básica y general de las aplicaciones empresariales se puede apreciar en la figura 8.

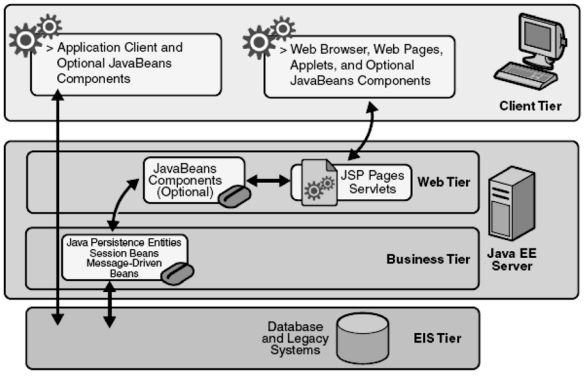


Figura 8. Arquitectura Aplicaciones Java EE 5[7].

Las aplicaciones Java EE 5 constan de tres capas principales: La capa cliente, la capa del servidor Java EE, que se divide en dos: la capa web y de lógica del negocio, y la capa de información empresarial.
La ingeniería de software define diferentes estilos arquitectónicos a utilizar en el desarrollo de aplicaciones software, del análisis de los estilos arquitectónicos y de la arquitectura de Java EE 5, se pueden definir las siguientes alternativas de arquitectura para los componentes software reutilizables:

- Arquitectura modelo–vista–controlador.
- Arquitectura por capas.
- Arquitectura orientada a servicios.

2.4.1 Arquitectura modelo–vista–controlador.

Franky [10] describe esta arquitectura de la siguiente manera: "El Modelo, maneja las reglas del negocio y las estructuras de datos; la vista, maneja presentación de los datos del sistema al usuario; y el controlador: Transforma pedidos del usuario en operaciones sobre los objetos del modelo y selecciona la vista para mostrar los resultados al usuario". La adaptación de este estilo arquitectónico para los componentes software reutilizables se puede apreciar en la siguiente figura:

En una parte del modelo estarían las entidades soportadas por el API de persistencia de Java, en la vista se ubican los componentes o controles para la interfaz de usuario y los servicios web, que exponen las funcionalidades del componente como servicios web, el control seria manejado por los EJBs soportados por los frameworks Enterprise Java Beans 3.0 y Jboss Seam.

2.4.2 Arquitectura por capas.

Este tipo de arquitectura pretende dividir los elementos del software en capas que se comunican entre sí hasta entregar el resultado al usuario.

Para los componentes de estudio, se definieron las siguientes capas: capa de datos, capa de entidades, capa de EJBs y la capa de presentación. Este estilo de arquitectura se puede apreciar mejor en la figura 9.

Capa de Datos: Esta capa se encarga de la conectividad con la base de datos, define el modelo físico de datos, el cual va estar ligado a las entidades.

Capa de Entidades: Definen las clases de entidad que van a tener persistencia en la base de datos, soportan la lógica del negocio del área de aplicación.

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

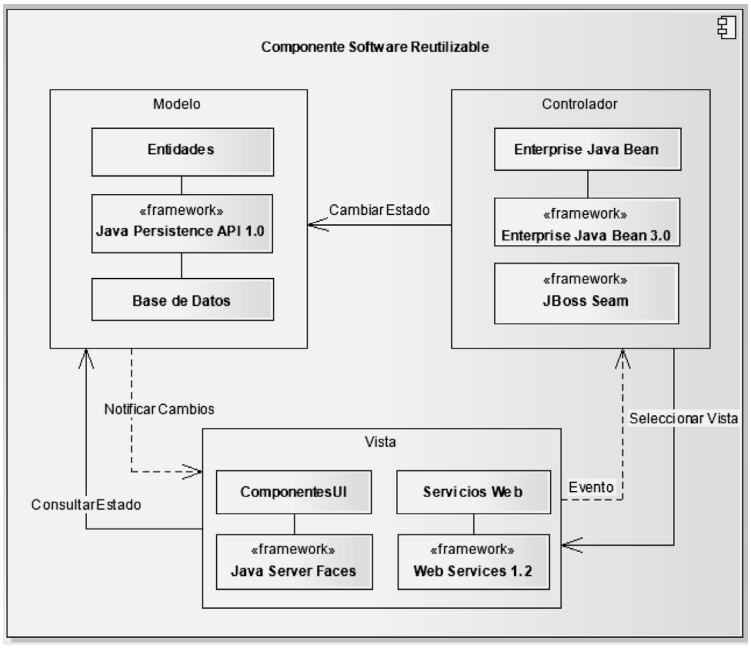


Figura 9. Arquitectura Modelo – Vista - Controlador

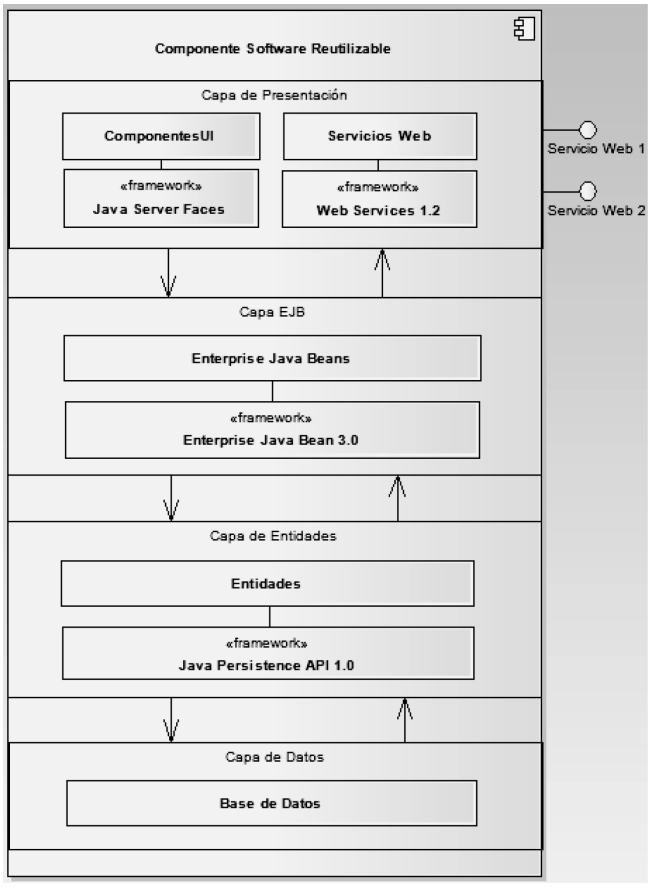


Figura 10. Arquitectura por Capas

Capa de EJBs: Establecen la lógica del negocio del área de aplicación, definen los servicios y funcionalidades que va a tener el componente.

Capa de Presentación: Es la encargada de utilizar los servicios propuestos por los EJB para exportarlos a clientes como controles de usuario personalizados para el área de aplicación o como servicios web para poder utilizarlos remotamente o por cualquier otro sistema.

2.4.3 Arquitectura orientada a servicios.

Según Toro [11], la arquitectura orientada a servicios consiste en la exposición y uso de servicios por parte de los sistemas informáticos. Los sistemas informáticos se dividen en varios niveles de servicios: servicios de conectividad con otros sistemas, servicios de datos, servicios de negocio, procesos de negocio y presentación de servicios para ser consumidos por diferentes sistemas y dispositivos.

En la figura 11 se aprecia la aplicación de la arquitectura orientada a servicios en los componentes software reutilizables.

En este estilo de arquitectura se dividen los elementos estructurales del componente en niveles de servicios que son exportados y usados por el nivel superior. Los niveles de servicios del componente son:

Nivel de Servicios de Conectividad: Expone los mecanismos de conectividad con cualquier base de datos.

Nivel de Servicios de Datos: En este nivel se encuentran las entidades soportadas por el API de persistencia de Java. En este nivel se encuentran los servicios de creación, modificación, eliminación y control de los datos.

Nivel de Servicios de Negocio: se ofrecen los servicios de la lógica del negocio del dominio de aplicación de los componentes; se encarga de manejar las transacciones y procesar las peticiones de los clientes.

Nivel de Presentación: En este nivel se encuentran los componentes para la interfaz de usuario y la exportación de los servicios ofrecidos por los EJBs como servicios Web. Estos elementos usan los servicios encapsulados en los EJBs del nivel de servicios de negocio o los servicios de datos propuestos por las entidades.

Además cada nivel de servicios puede exponer al desarrollador directamente los servicios que implementa, el desarrollador puede directamente acceder a los servicios ofrecidos por las entidades o utilizar los servicios de los EJBs directamente.

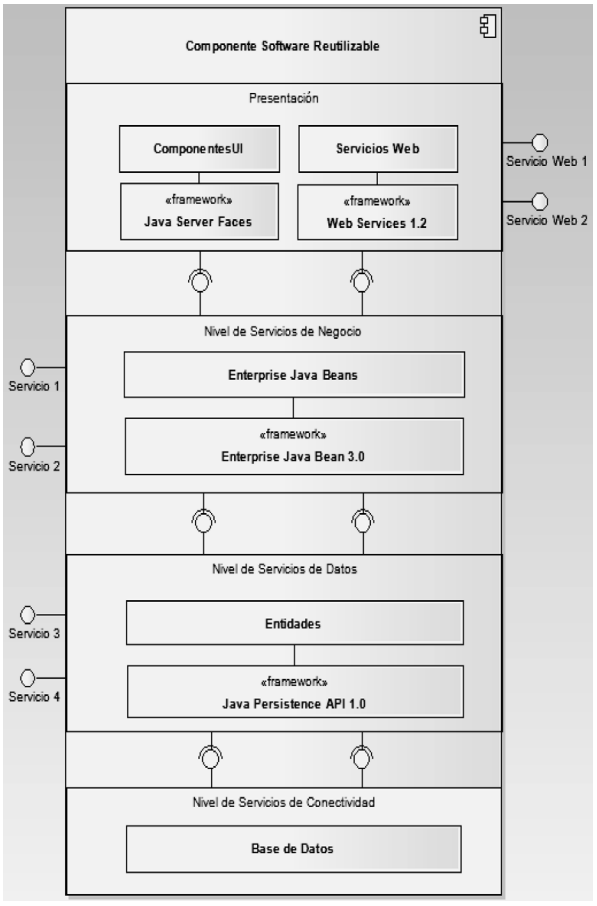


Figura 11. Arquitectura orientada a servicios.

3. CONCLUSIONES

La reutilización de software trae beneficios muy importantes para la actividad de desarrollo profesional de software, ya que permite reutilizar elementos ya elaborados, haciendo así más rápida su implementación. Los beneficios más importantes de la reutilización de software propuestos por Meyer [12] son:

- **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 el desarrollo de nuevos sistemas.
- **Eficiencia**: Los componentes en muchas ocasiones son desarrollados por expertos en el dominio que se desarrollaron, usando los mejores algoritmos, patrones y estructuras de datos.

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

- **Consistencia**: Los componentes deben tener un énfasis estricto en un diseño regular y coherente, ya que el estilo del componente influye en el estilo del sistema desarrollado.
- **Inversión**: El componente no se desarrolla solo para un proyecto, tiene la oportunidad de utilizarlo en muchos proyectos. Se puede preservar en el componente el conocimiento de los mejores desarrolladores.

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 Universidad, que dispondría 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.

Los componentes implementados ofrecerían facilidades para el desarrollo software, como son: componentes para la interfaz de usuario personalizados listos para ser utilizados, invocación remota de las funcionalidades del componente utilizando el modelo de Servicios Web, además, se podrían utilizar los EJB y las entidades encapsuladas en el componente para el desarrollo de cualquier sistema informático donde se requieran los requisitos del dominio de aplicación que abarca el componente.

Por medio del modelo de selección de componente se pueden identificar los componentes candidatos a implementar para un dominio de aplicación particular, asegurando por medio del análisis de reutilización la posibilidad de utilizarse en varios proyectos de desarrollo software.

Hasta el momento se han resuelto dos interrogantes de la presente investigación, y se está trabajando en la depuración y refinamiento de un proceso de desarrollo de componentes software, con el cual se resuelvan todas las preguntas de esta investigación, y el problema planteado. Además, como se puede apreciar de la descripción realizada en este artículo, hasta este momento se han logrado resultados significativos que pueden ayudar a mejorar el desarrollo software en general y en especial en la División de Servicios de Información de la Universidad Industrial de Santander.

4. REFERENCIAS

[1] PRESSMAN, Roger S. Ingeniería del Software un enfoque práctico. Quinta Edición. Mc Graw Hill. 2002.

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

[3] MEYER, Bertrand. The Significance of Components. Beyond Objects column, Software Development. 1999.

[4] SUN MICROSYSTEM. Tutorial: Object-Oriented Analysis and Design Using UML. Copyright 2003.

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

[6] SCHNEIDER, Jean Guy y HAN, Jun. Artículo: Components – the Past, the Present ,and the Future. School of Information Technology, Swinburne University of Technology. Hawthorn, Victoria, Australia.

[7] SUN MICROSYSTEM, The Java EE5 tutorial. 2007

[8] WEITZENFELD, Alfredo. Ingeniería de Software Orientada a Objetos, Teoría y Práctica con UML y Java. México. Itam. 2002.

[9] BRUEGGE, Bernd y DUTOIT, Allen H. Object-Oriented Software Engineering Using UML, Patterns and Java. 2ª Edición. Prentice Hall. 2004.

[10] FRANKY, Maria Consuelo. Curso: Desarrollo de aplicaciones en Java EE 5. CincoSOFT LTDA. 2007.

[11] TORO, Víctor Manuel. Conferencia: Panorama sobre la Ingeniería del Software. CincoSOFT LTDA. 2007.

[12] MEYER, Bertrand. Construcción de Software Orientado A Objetos. Prentice-hall 2002. Capitulo 4.

[13] RUMBAUCH, James; JACOBSON, Ivar y BOOCH, Grady. El lenguaje unificado de modelado. Manual de referencia. Addison Wesley. Madrid – España. 2000.

[14] DRAKE, José M; MEDINA, Julio Luís y GONZALEZ HARBOUR, Michael. Artículo: Entorno para el Diseño de Sistemas Basados en Componentes de Tiempo Real. Grupo de Computadores y Tiempo Real. Universidad de Cantabria. España.

[15] BERTOA, Manuel F. y VALLECILLO, Antonio. Calidad de Componentes Software. Departamento de Lenguajes y Ciencias de la Computación. Universidad de Málaga. 2004

Nota: Los diagramas UML presentados en el presente artículo fueron elaborados por medio de la herramienta software Enterprise Architect de Sparx System, licenciada por la División de Servicios de Información de la UIS.