DESARROLLO E IMPLEMENTACIÓN DE UN REPOSITORIO DE COMPONENTES PARA DAR SOPORTE A LA PRÁCTICA DE LA INGENIERÍA DEL SOFTWARE BASADA EN COMPONENTES DE LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN

ROBINSON DELGADO ROJAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS
MAESTRÍA EN INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2012

DESARROLLO E IMPLEMENTACIÓN DE UN REPOSITORIO DE COMPONENTES PARA DAR SOPORTE A LA PRÁCTICA DE LA INGENIERÍA DEL SOFTWARE BASADA EN COMPONENTES DE LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN

ROBINSON DELGADO ROJAS

Trabajo de investigación para optar al título de Magister en ingeniería de sistemas e informática.

DIRECTOR
Msc. FERNANDO ROJAS MORALES

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS
MAESTRÍA EN INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2012

TABLA DE CONTENIDO

LISTA DE FIGURAS

RESUMEN ESPAÑOL.

TITULO:
“DESAROLLO E IMPLEMENTACIÓN DE UN REPOSITORIO DE COMPONENTES PARA DAR SOPORTE A LA PRÁCTICA DE LA INGENIERÍA DE SOFTWARE BASADA EN COMPONENTES DE LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN.” *****

AUTOR:
Ing. ROBINSON DELGADO ROJAS.**

PALABRAS CLAVES:
Componentes, Ingeniería de Software basada en componentes, Repositorio de componentes.

CONTENIDO:

El modelo de desarrollo basado en componentes es un nuevo paradigma de desarrollo que permite la implementación de sistemas rápidamente, partiendo de componentes de software previamente desarrollados. El repositorio de componentes es la pieza principal de este modelo de desarrollo ya que permite la administración, selección, identificación, cualificación y ensamblaje de componentes de software.

En este artículo se explica el modelo de desarrollo, y como se diseñará un repositorio de componentes para dar soporte al proceso de desarrollo de software empresarial. Para el diseño se tendrán en cuenta todos los requerimientos funcionales y no funcionales que debe contemplar un repositorio de componentes, que soporte el modelo de desarrollo basado en componentes.

La implementación de este repositorio se aplicará a la División de Servicios de Información DSI, de la Universidad Industrial de Santander UIS, como mejora al modelo de desarrollo allí implantado, optimizando este modelo de desarrollo por componentes, ya que con el repositorio se permite disminuir los tiempos de búsqueda, selección e identificación de los componentes. Además de mejorar los tiempos de respuesta de atención a las solicitudes de mantenimiento y desarrollo que allí se generan, ya que las demoras se disminuyen al máximo al obtener e identificar de forma rápida los componentes que deben ser revisados o integrados en los sistemas solicitados. También se elimina el desarrollo de componentes de forma repetida y descontrolada por la falta de la herramienta que permita monitorear y centralizar los componentes desarrollados.

* Trabajo de investigación.
** Facultad de Ingenierías Físico – Mecánicas. Escuela de Ingeniería de sistemas. Director Ing. Fernando Rojas Morales.

ENGLISH SUMMARY.

TITLE:
DEVELOPMENT AND IMPLEMENTATION OF A COMPONENTS REPOSITORY TO SUPPORT THE PRACTICE OF COMPONENT-BASED SOFTWARE ENGINEERING OF INFORMATION SERVICE DIVISION.*****

AUTHOR:
Eng. ROBINSON DELGADO ROJAS.**

KEYWORDS:
Component, Component – Based Software Engineering, Component's Repository.

CONTENT:

The component-based development model is a new paradigm of development that allows the implementation of systems quickly, starting of software components developed previously. The component's repository is the main tool of this development model, this allows the administration, selection, identification and assembly of software components.

In this paper explain it the development model, and as will design a component's repository for give support to development process of business software. For design it will take into account all functional requirements and non functional that would have a component´s repository that give support at components-based development model.

The deployment of this repository will apply it to Information Services Division of Santander Industrial University UIS, for do it better this development the model implanted here, optimizing this development components model, this repository can do smaller times of searching, selection, and identification of components. Also it does better the response times of attention to the maintenance requirements and development that do it, because the delays decrease for get and identify quickly the components that it would be verified or integrated in the requested systems. Also eliminates the development of repeated components and don´t controlled for don´t exist tool that allows overlook and centralize the components developed.

* Trabajo de investigación.
** Facultad de Ingenierías Físico – Mecánicas. Escuela de Ingeniería de sistemas. Director Ing. Fernando Rojas Morales.

INTRODUCCIÓN

Los nuevos modelos de desarrollo de software proponen la implementación, compra o adquisición de 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 desarrollarlo 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 realizaría buscando los componentes en el repositorio, se combinarían y adaptarían estos componentes, en vez de estar codificando y construyendo las mismas aplicaciones cada vez.

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 utilizando estos componentes y no las piezas por separado de un lenguaje de programación convencional[1].

Para la optimización del proceso de desarrollo se hace indispensable contar con un sistema que permita hallar, proveer y administrar los componentes requeridos para el ensamble del sistema. Este sistema recibe el nombre de repositorio de componentes y debe permitir el almacenamiento, registro, búsqueda y administración de todos los artefactos de software producidos bajo el ciclo de vida basado en componentes. Es decir cumple una función mediadora en la creación de sistemas basados en componentes.

Actualmente existen portales Web donde venden diferentes clases de componentes para diferentes plataformas, algunos de estos portales son: Componentsource (www.componentsource.com), Flashline (www.flashline.com) y WrldComp

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

(www.wrldcomp.com), también encontramos marcos de trabajo (frameworks) que permiten utilizar componentes ya implementados para desarrollar nuevas aplicaciones. En cuanto a estándares para el desarrollo y comunicación de componentes existen varios modelos de componentes como son: .NET, COM (Component Object Model), DCOM (Distributed Component Object Model) de Microsoft, JavaBeans y EJB (Enterprise Java Beans) de Sun Microsystems y CORBA (Common Object Request Broker Architecture) del Object Management Group.

1. DESCRIPCION DE LA INVESTIGACION

1.1 MARCO DE REFERENCIA

1.1.1. MARCO CONCEPTUAL

El marco conceptual tiene como finalidad unificar los significados de conceptos y siglas empleados dentro del plan de investigación para efectos de claridad en el lenguaje empleado. De esta manera, se evitan interpretaciones diferentes de las palabras usadas en este documento.

1.1.2. TÉRMINOS UTILIZADOS

Desarrollo de Software Basado en Componentes (DSBD): Una perspectiva del desarrollo de software en donde todos los “artefactos" se pueden construir mediante ensamblaje, adaptación e integración de componentes existentes y en una variedad de configuraciones.

Componente: 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. (Szyperski, 1998)

Repositorio: El repositorio de componentes es una librería del sistema que da soporte para hallar, proveer y administrar los componentes para la construcción de una aplicación. Es una herramienta que almacena, registra y administra todos los artefactos producidos bajo en ciclo de vida basado en componentes sobre una arquitectura de componentes y soporta la reutilización de componentes. El repositorio es un mediador central que permite la inclusión, creación, verificación, configuración, administración y circulación de los componentes. Otros términos como: biblioteca o librería pueden ser utilizados para hacer referencia a un repositorio, de ahora en adelante en esta propuesta siempre que se hable de biblioteca, librería o repositorio se hará referencia al repositorio de componentes.

Meta data: La meta data son datos de los datos, que pueden contener información acerca de la estructura, relación y propiedades de los datos almacenados que permiten la búsqueda, administración, mantenimiento y almacenamiento de dicha información.

Servicio: Un servicio es una funcionalidad implementada que permite a un sistema externo realizar un conjunto limitado de procedimientos o funciones que se encuentran encapsulados dentro del proveedor de servicios. Estos servicios tienen la cualidad de ocultar al implementador la forma y estructura de la lógica del negocio.

Enterprise Java Beans (EJB): Es el modelo de componentes del lado servidor de Sun Microsystems.

Especificación: Un documento que prescribe de forma completa, precisa y fiable los requisitos, diseño, comportamiento o características de un sistema o de un componente.

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. Ejemplos de modelo de componentes: COM/DCOM, .Net, CORBA CCM, EJB, OSGi y Web Services[2].

Unified Modeling Language (UML): Lenguaje de modelado de sistemas software desarrollado por OMG.

1.1.3. SIGLAS

DSI	División de Servicios de Información
API	Application Programming Interface
ASP	Active Server Page
CBD	Component-Based Development
CBSE	Component-Based Software Engineering

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

DSBC	Desarrollo de Software basado en Componentes
DOM	Document Object Model
EJB	Enterprise JavaBeans
ISBC	Ingeniería del Software Basada en Componentes
JSP	Java Active Page
JSF	Java Server Faces
JVM	Java Virtual Machine
OMG	Object Management Group
RMI	Remote Method Invocation

1.2. MARCO TEÓRICO

En esta sección se presentan las bases teóricas de la investigación, centrada principalmente en el área de la Ingeniería del Software Basada en Componentes, el contenido del marco teórico se presenta en la siguiente figura.

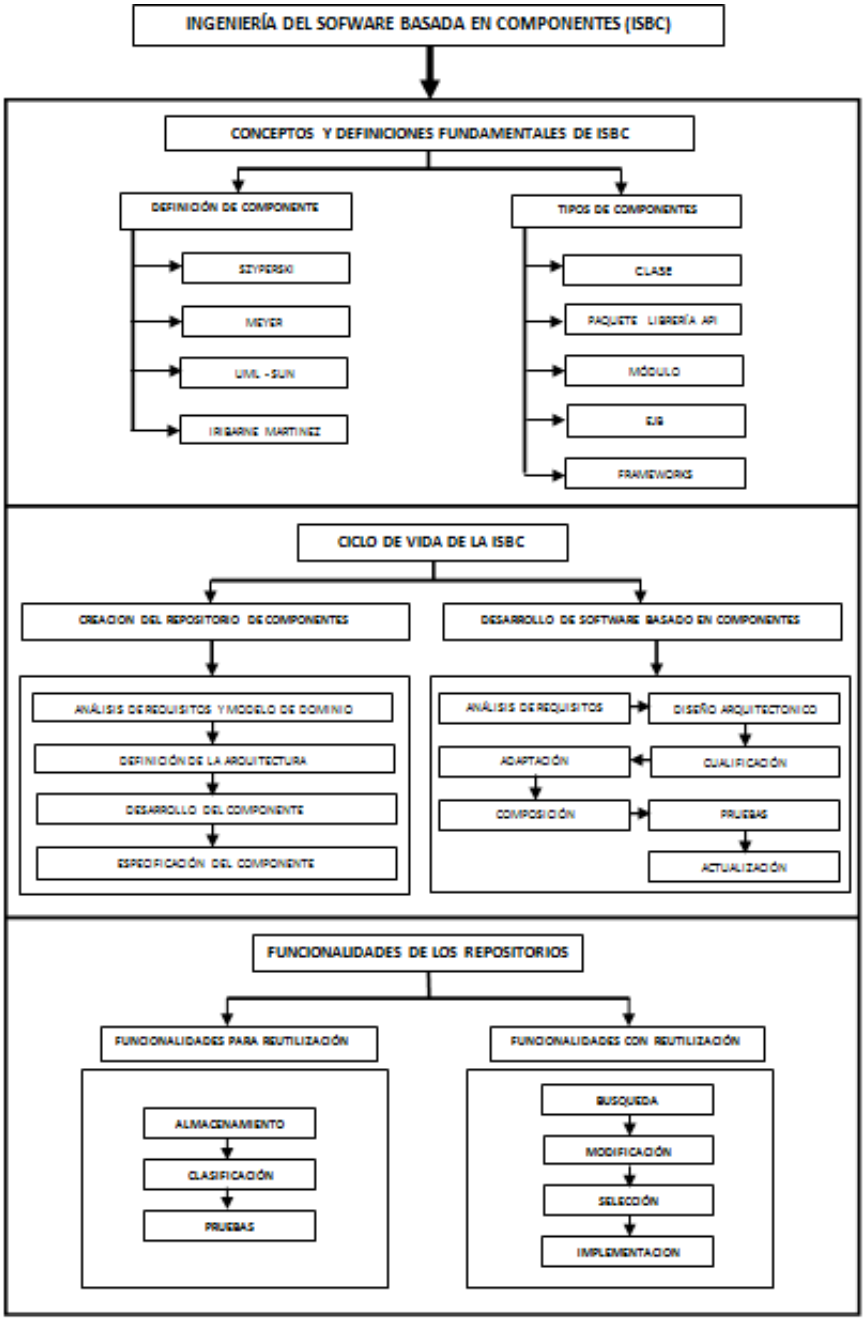

Figura 1. Marco Teórico de la Investigación.

1.2.1. CONCEPTOS Y DEFINICIONES FUNDAMENTALES DE LA ISBC

En esta sección se darán los conceptos y definiciones más importantes que se tratan en la ISBC para poder abarcar y entender su ciclo de vida y las actividades que este comprende. En primer lugar, el concepto de componente muchos autores lo definen de forma diferente:

Según Szyperski[3], "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[4] define 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 en base solo a su especificación.
6. Se puede componer con otros componentes para formar un software.
7. Puede ser integrado rápidamente y fácilmente.

En el Lenguaje Unificado de Modelamiento (UML), más específicamente en la metodología de desarrollo utilizada por Sun Microsystems[5], definen a 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. En esta metodología consideran

[3] Szyperski, C. (1998). Component Software. Beyond Object-Oriented Programming. Addison-Wesley.
[4] Meyer, B. (1999). The Significance of Components. Beyond Objects column, Software Development.
[5] Object-Oriented Analysis and Design Using UML. Copyright 2003 Sun Microsystems,

como componente software a los elementos de la base de datos, a una clase java distribuida en código fuente, a una librería de clases .jar, a un servlet, a una aplicación completa.

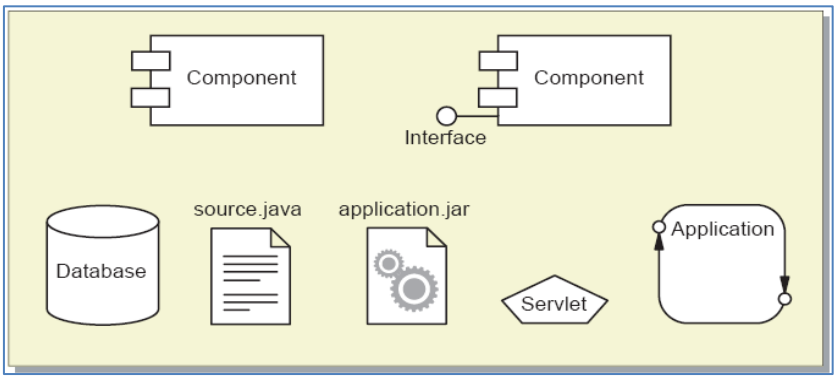

Figura 2. Ejemplos de Componentes.

.

Hyung Cho y John D. McGregor[6], definen el término componente como piezas software que pueden ser combinadas para construir algo más grande y completo (otro componente, subsistema, sistema).

Iribarne Martínez[7] 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.

Después de ver las definiciones que diferentes autores dan al concepto de componente, se puede resumir que un componente es 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. Para clasificarse como, un elemento software componente debe cumplir con los siguientes criterios:

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

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

- 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 las que fue implementado.
- Debe permitir el mantenimiento y la actualización de forma individual, sin afectar estructuralmente al sistema que compone.

1.2.1.1 TIPOS DE COMPONENTES REUTILIZABLES

A continuación se detallan la forma como se utilizan ciertos elementos software como componentes reutilizables, este tipo de componentes se encuentran en páginas Web que venden componentes, en artículos y libros que hablan de la ingeniería del software basada en componentes y en el desarrollo de software en general.

1.2.1.2 Clase

Una clase es una descripción generalizada que describe una colección de objetos similares. Las clases encapsulan datos (atributos) y los métodos que manipulan esos datos. Entre las clases existe una jerarquía de clases, en la cual los atributos y operaciones de la superclase son heredados por subclases que pueden añadir, cada una de ellas sus propios atributos y métodos. Al tener herencia las clases no se pueden instalar como una unidad software independiente, ya que algunas clases tienen relaciones con otras clases padres y/o clases hijas. También las clases no hacen referencia explicita de sus dependencias y requisitos.

1.2.1.3 Paquete / Librería / API

Un paquete es un conjunto de clases, usualmente agrupadas conceptualmente. Un paquete no es un elemento ejecutable, debe incluirse una referencia en la aplicación que quiere utilizar las clases que posee. Un paquete permite un alto grado de reutilización y es orientado a objetos.

Por ejemplo, el API que se utiliza para generar los archivos pdf en aplicaciones java, y en aplicaciones Web, se distribuye en un archivo .jar o en el paquete com.logawie.* Estos componentes son de la forma mostrada en la siguiente figura.

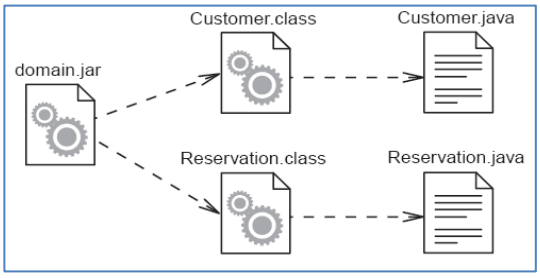

Figura 3. Ejemplo de Librería

Otro ejemplo de este tipo de componentes, son los componentes java que se encuentran en la página de Component Source. Esta empresa distribuye estos componentes, en un archivo jar con las clases que van a manejar ciertas funcionalidades para un sistema informático. Se tomó como ejemplo un componente generador de informes, que consta de javaBeans para ser utilizados en las aplicaciones que se requiera.

1.2.1.4 Módulo

Un módulo es un conjunto de clases y otros elementos no orientados a objetos, como pueden ser procedimientos y funciones. Un módulo se utiliza para implementar las clases, procedimientos y funciones generales que se van a utilizar en toda la aplicación. Es posible que un mismo módulo se utilice en varias aplicaciones, pero al igual que las clases no tiene una especificación que permita identificarlos claramente para hacer más

fácil su reutilización, también al actualizar un módulo afecta notablemente el funcionamiento de la aplicación que lo utiliza. Ejemplo de este tipo de módulos son las dll desarrolladas en Visual Basic, que pueden ser utilizadas en aplicaciones web y aplicaciones de windows.

1.2.1.5 Enterprise Java Beans

La especificación de JavaBeans Enterprise define una arquitectura para un sistema transacional de objetos distribuidos basado en componentes. La especificación establece un modelo de programación (API EJB). Este modelo de programación proporciona a los desarrolladores de Beans y a los vendedores de servidores EJB un conjunto de contratos que definen una plataforma de desarrollo común. El objetivo de estos contratos es asegurar la portabilidad a través de los vendedores y el soporte de un rico conjunto de funcionalidades. Se puede apreciar en la figura 4, que los enterprise java bean se encuentran alojados en el servidor de aplicaciones. Las paginas jsp, servlets y aplicaciones cliente, los utilizan para manejar ciertas funcionalidades encapsuladas que los EJB ofrecen. Los EJB se encargan de manejar el acceso a datos, las transacciones y sesiones de la aplicación.

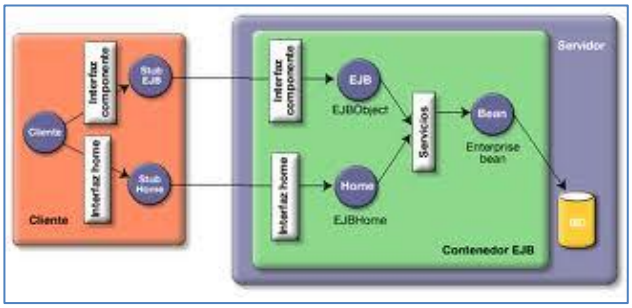

Figura 4. Componentes Enterprise JavaBean

1.2.2. CICLO DE VIDA DE LA ISBC

Un ciclo de vida define el conjunto de actividades que se llevan a cabo para desarrollar un producto software desde su concepción hasta su instalación y mantenimiento. En la ISBC se desarrollan dos actividades fundamentales: La ingeniería de dominio y el desarrollo

basado en componentes[8]. En la figura 5 se puede apreciar el ciclo de vida propuesto por Pressman para la ingeniería del software basada en componentes.

La ingeniería del dominio explora un dominio de aplicaciones con la intención de encontrar específicamente los componentes de datos funcionales y de comportamiento candidatos para la reutilización. Estos componentes se encuentran en bibliotecas de reutilización.

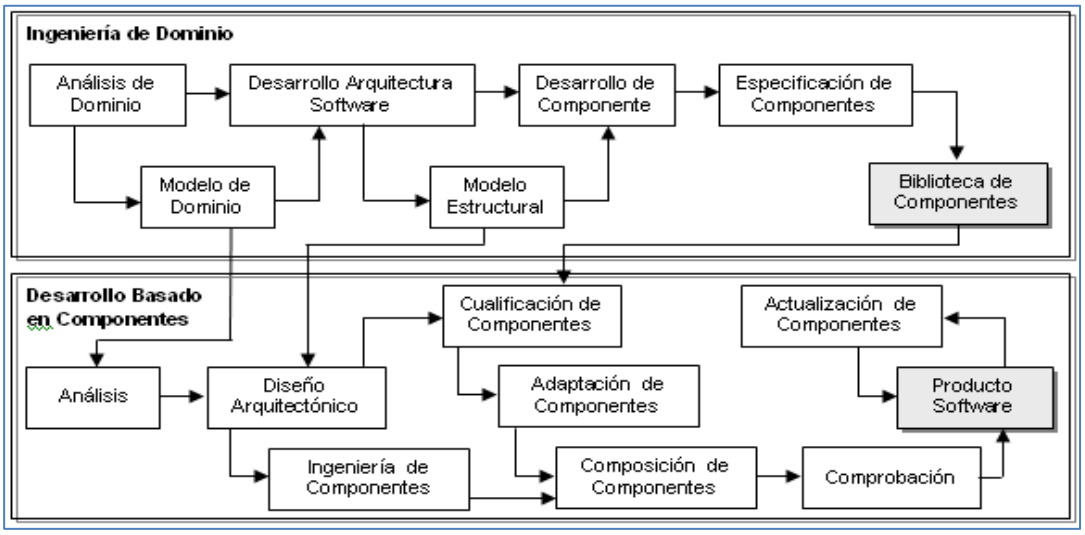

Figura 5. Actividades fundamentales de la ISBC.

El desarrollo basado en componentes obtiene los requisitos del cliente y selecciona el estilo arquitectónico adecuado para cumplir los objetivos del sistema que se va a construir, y a continuación: (1) selecciona posibles componentes para la reutilización; (2) cualifica los componentes para asegurarse de que encajan adecuadamente en la arquitectura del sistema; (3) adapta los componentes si se deben hacer modificaciones para poderlos integrar adecuadamente; (4) integra los componentes para formar subsistemas y la aplicación completa.

Las actividades fundamentales de la ingeniería del software basada en componentes se pueden dividir en dos grades grupos, en primer lugar, las actividades necesarias para la

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

creación de biblioteca de componentes que serán detalladas en el punto de repositorios, y el segundo, las actividades necesarias para el desarrollo de software basado en componentes.

1.2.2.1 ACTIVIDADES NECESARIAS PARA EL DESARROLLO DE SOFTWARE BASADO EN COMPONENTES

El desarrollo basado en componentes abarca las actividades necesarias para desarrollar un producto software ensamblando componentes reutilizables disponibles en una biblioteca. El desarrollo basado en componentes se adapta a cualquier metodología de desarrollo software, después de establecer requisitos se establece un diseño arquitectónico pero en lugar de entrar inmediatamente en tareas de diseño detalladas, el equipo examina los requisitos para determinar cuáles de ellos están dispuestos para la composición, y no para la construcción.

Las actividades más importantes dentro del desarrollo basado en componentes se describen a continuación:

Análisis de Requisitos: En esta fase se realiza la selección de los componentes candidatos a reutilizar, que cumplen con los requisitos establecidos del sistema informático a crear y se adaptan al diseño arquitectónico del sistema. También se especifican los nuevos componentes que es necesario implementar para cumplir a satisfacción con los requerimientos del cliente.

Diseño Arquitectónico: Una vez seleccionados los componentes necesarios para ensamblar el sistema informático, se debe diseñar la arquitectura del sistema, la arquitectura define la estructura del sistema, la cual esta formada por componentes software y sus relaciones. El diseño arquitectónico debe satisfacer los requisitos funcionales y no funcionales del sistema.

Selección y cualificación de Componentes: En esta actividad se asegura que el componente candidato cumple con los requisitos funcionales y con el diseño arquitectónico del sistema a crear, es decir que encaje adecuadamente en la arquitectura

del sistema, también se debe verificar que cumpla con las características de calidad (por ejemplo, rendimiento, fiabilidad, usabilidad) necesarias para la aplicación.

Adaptación de Componentes: La adaptación de componentes tiene que ver con la modificación de los componentes cualificados para eliminar los conflictos que se puedan presentar al integrarlos al sistema informático a crear. Para tratar estos conflictos se suele utilizar una técnica de adaptación llamada encubrimiento de componentes, existen varios tipos de adaptación:

- El encubrimiento de caja blanca examina los detalles del procesamiento interno del componente y realiza las modificaciones a nivel de código para eliminar los conflictos.
- El encubrimiento de caja gris se aplica cuando la biblioteca de componentes proporciona un lenguaje de extensión de componentes, o API, que hace posible eliminar o enmascarar los conflictos.
- El encubrimiento de caja negra requiere la introducción de un pre-procesamiento o post-procesamiento en la interfaz de componentes para eliminar o enmascarar conflictos.

El equipo de software debe determinar si se justifica el esfuerzo requerido para envolver adecuadamente un componente o si por el contrario se debería diseñar un componente personalizado.

Composición de Componentes: La tarea fundamental de la composición de componentes es el ensamblaje de todos los componentes cualificados, adaptados y diseñados para armar el sistema informático. Este ensamble se hace teniendo como base el diseño arquitectónico del sistema.

La especificación de la interfaz lista las signaturas de los métodos suministrados por el componente, los roles que juega por el componente y las precondiciones y poscondiciones para cada método. El modelo de componentes asegura que los componentes desarrollados en distintos lenguajes de programación que residan en distintas plataformas puedan ser interoperables.

Pruebas: Una vez ensamblado el sistema se deben realizar las pruebas funcionales del sistema completo, como también las pruebas de integridad para verificar que el sistema cumple con los requisitos establecidos.

Actualización: La actualización del sistema se realiza a nivel de componente y no de todo el sistema, se realiza cambiando al componente de forma que no afecte el funcionamiento de todo el sistema. Al ensamblar una actualización de un componente se debe asegurar que no afecta al sistema que compone, debe seguir cumpliendo con los requisitos para los que fue hecho.

1.2.3. REPOSITORIO DE COMPONENTES.

Los repositorios de componentes son elementos fundamentales dentro del proceso de reutilización porque sirven como elemento de enlace entre el desarrollo para la reutilización, cuando se producen los elementos a reutilizar, y el desarrollo con reutilización, cuando estos elementos son reutilizados.

Los repositorios de reutilización están destinados, por tanto a facilitar la reutilización de componentes a lo largo del ciclo de vida del software para lograr alcanzar los objetivos de reducción de costos y aumento de la productividad. La razón fundamental de la existencia de un repositorio es permitir el acceso a los componentes desarrollados.

Un repositorio es conocido también como una biblioteca de componentes o librería de reutilización. Es una herramienta que facilita el desarrollo de software con reutilización de componentes con el objetivo de obtener mayor productividad en el proceso de desarrollo de sistemas software.

En la actualidad existen varias implementaciones de repositorios de componentes en el mercado, pero como es de esperarse cada una de ellas tiene algo que es relevante para un usuario específico pero le falta algo para estar completo según los requerimientos o necesidades de cada casa desarrolladora de software.

ACTIVIDADES NECESARIAS PARA LA CREACIÓN DE LA BIBLIOTECA DE COMPONENTES

Análisis de Requisitos y Modelo de Dominio: El análisis de dominio se utiliza para identificar los componentes reutilizables que son relevantes para el dominio de aplicación de una situación determinada, intenta definir un conjunto de características que sean compartidas por todos los componentes que van a ser parte de la biblioteca de componentes que se quiere formar. También se obtienen los requisitos funcionales y no funcionales que van a manejar cada uno de los componentes. El producto final del análisis de dominio es un modelo de dominio que contiene los componentes identificados para la reutilización.

Definición de la Arquitectura Software: En esta fase se pretende caracterizar el modelo estructural del dominio de aplicación de la biblioteca de componentes que se esta formando. El modelo estructural es un estilo arquitectónico que puede y debe reutilizarse en aplicaciones pertenecientes al dominio, representa la estructura, las propiedades de los componentes reutilizables y las interrelaciones que tienen lugar entre todos los elementos arquitectónicos del dominio de aplicación.

Desarrollo de Componentes: Esta fase de Desarrollo comprende las actividades necesarias para implementar los componentes reutilizables que van a ser parte de a biblioteca de componentes. Para su implementación se puede utilizar cualquier lenguaje de programación, pero teniendo en cuenta que el desarrollo es para la reutilización.

Especificación de Componentes: Uno de los aspectos más importantes en el proceso de creación de la biblioteca de componentes reutilizables es la especificación del componente, que permite identificar y describir el componente, detallando las funcionalidades que presta, para hacer más fácil su búsqueda dentro de la biblioteca de componentes. Un componente tiene que mostrar información suficiente para poder ser implementado y usado.

Otro de los aspectos importantes dentro del desarrollo y especificación del componente es la forma como el ingeniero de software puede disponer del componente que realmente necesita, el ingeniero recibe el componente de la biblioteca ejecutando consultas, si hubo muchos componentes candidatos para ser utilizados, el usuario debe seleccionar el que mejor se adapte a sus necesidades. También es importante que una vez implementado el componente sea clasificado de una manera acertada dentro de la biblioteca de componentes, existen esquemas de clasificación para componentes de software reutilizables algunos de ellos son: métodos de las ciencias de la documentación y de bibliotecología, métodos de inteligencia artificial y sistemas de hipertexto.

Como la principal actividad de un repositorio de componentes es permitir y facilitar la reutilización de componentes de software, dentro de sus principales beneficios se puede mencionar[9]:

- **Clasificación**.

  Permite determinar en qué grupo de componentes debe quedar alojado un componente en particular, esta clasificación puede hacerse por medio de varios criterios de clasificación según las necesidades del usuario. Algunas de las características más utilizadas son:

  - Proveedor.
  - Funcionalidad.
  - Área de desarrollo.
  - Arquitectura.
- **Búsqueda**.

  Proceso que permite hallar un componente de software en el repositorio que se adapte a la funcionalidad requerida, para ello deben estar bien definidas las características de los componentes y sus funciones realizadas.
- **Modificación.**

  En este proceso se contempla la anulación de posibles conflictos en la integración del componente al nuevo sistema de software desarrollado.
- **Pruebas**.

[9] A reusable software component-based development process model, M.R.J. Qureshi, S.A. Hussain.

Esta característica es considerada de gran importancia ya que permite antes de la implementación del componente en el nuevo sistema, comprobar su funcionamiento y el grado de adaptabilidad a los requerimientos exigidos.

- **Implementación**.
  Los componentes que son seleccionados del repositorio deben ser de fácil implementación en el nuevo sistema, bajo esta premisa se debe permitir la consulta de la forma de implementación en el nuevo sistema.

- **Almacenamiento**.
  Esta es la característica de mayor importancia en el proceso de desarrollo de software basado en componentes, ya que todo componente desarrollado debe estar alojado en el repositorio, de no ser así este componente básicamente no existirá en el proceso.
- **Selección**.
  Proceso que permite indicar que componente de una lista de posibles elegidos cumple con los requerimientos y es seleccionado como el componente a implementar en el nuevo sistema.

Los repositorios de componentes pueden orientarse o desarrollarse para ser aplicados a una metodología de diseño específica, es decir puede basarse en modelos[10], como el CARDS Command Center Library, o basarse en componentes[11], como la biblioteca GIRO o el CRPS[12] , u orientarse a otras metodologías como basadas en conocimiento o basadas en procesos.

ALGUNOS REPOSITORIOS DE COMPONENTES DESARROLLADOS.

Los componentes pueden clasificarse en 2 clases.

- **Repositorios Comerciales**.

[10] Model-Based Reuse Repositories, James Petro and Michael E. Fotta.
[11] Biblioteca de Reutilizacion GIRO, Carmen Hernandez Diez and Miguel Angel Laguna Serrano.
[12] Design and implementation of component repository, Jung Eun Cha, Young Jung Yang.

- **Repositorio +1:** Desarrollado por Software Engineering Co. Trabaja sobre plataforma Sun y sistema operático Solaris. Soporta reutilización de diseño, documentación, código fuente, pruebas e información de modelado.
- **SALMS (Software Asset Library Management System):** Este repositorio permite su instalación en PC's o estaciones de trabajo UNIX, estando accesible por todos los desarrolladores en una organización. La interface al usuario se presenta bajo un ambiente Web que permite observar todos los artefactos producidos bajo el ciclo de vida, artefactos como: restricciones, modelos de arquitectura, especificación de diseño, código fuente o Scripts de pruebas.
- **ASRR (Automated Software Reuse Repository):** Este repositorio está compuesto por 2 partes principales, la herramienta de administración y el repositorio de reutilización. La primera permite el manejo de los usuarios y sus privilegios en el repositorio, y la segunda posibilita la ejecución de funciones como búsqueda, almacenaje y descarga de componentes.
- **Repositorio Universal**: Desarrollado por Unisys. Desarrollado para facilitar a los desarrolladores moverse en un ambiente basado en repositorios. Este repositorio está orientado a objetos y permite adicionarle módulos propios desarrollados para complementarlo según la necesidad del usuario.

- **Repositorios Gubernamentales**.
  - **DSRS (Defense Software Repository System)**: El DSRS es un repositorio automatizado que permite almacenar y recuperar componentes de software. Este repositorio tiene la habilidad de conectarse a otros repositorios con el fin de intercambiar componentes de virtuales ubicados en otros repositorios.
  - **LID (Library Interoperability Demostration)**: Es un prototipo de repositorio que permite ilustrar como una librería puede ser descompuesta en distintas capas funcionales y conectadas en interfaces abiertas tal como lo especifica (ALOAF) Asset Library Open Architecture Framework.

  Existen otros repositorios conocidos en la actualidad como ICASE, MORE, SAIC/ASSET, PAL entre otros[13]

[13] A survey of software Reuse Repositories – Jiang Guo and Luqi, Department of Computer Science.

1.3. PLANTEAMIENTO DEL PROBLEMA

Dentro del proceso de mejoramiento continuo en los procesos y procedimiento de la Universidad, la División de Servicios de Información (DSI) determinó cambiar su modelo de desarrollo de software al modelo de desarrollo basado en componentes, gracias a los resultados derivados del trabajo de investigación de maestría realizado por el Msc Fredy Humberto Vera Rivera, quien planteó la necesidad de centralización de los componentes. Es así como nace esta propuesta de investigación. Este modelo de desarrollo ha permitido no solo mejorar la documentación y el tiempo de implementación de algunos sistemas, sino que también se han corregido varios problemas de desarrollo que se habían estado presentando como:

- No terminar los desarrollos en el plazo definido.
- Requerir más presupuesto del inicialmente solicitado.
- Baja calidad del software generado
- Software que no cumple con las especificaciones.
- Código inmantenible que dificulta la gestión y evolución de los sistemas desarrollados.

Si bien han disminuido los problemas de desarrollo con la aplicación del nuevo proceso, es cierto también que han surgido algunos inconvenientes que han hecho la labor de construcción de software un poco complicada. Uno de estos es la falta de una forma de organizar los componentes, funciones y servicios que los diferentes grupos de desarrollo ofrecen según su área de desempeño, ya que si bien se encuentran centralizados, no se cuenta con una forma que favorezca la disposición de un servicio o componente que pueda ser requerido por otro grupo de desarrollo en un momento dado. Esto conlleva al desarrollo de funcionalidades y componentes que hacen o cumplen la misma función. Es importante observar que el retrabajo es una práctica que está en contra de la aplicación del proceso de desarrollo basado en componentes adoptado por la DSI, que ignora las recomendaciones y principios del desarrollo de software basado en componentes.

Esta problemática se puede evidenciar en la División de Servicios de Informacion, en las diversas áreas de desarrollo existentes, áreas como la académica, financiera, recursos

humanos entre muchas otras. Cada una de estas áreas cuenta con un número considerable de desarrolladores, los cuales implementan componentes, funciones y servicios sin contar con un sitio donde especificar, centralizar y almacenar los componentes desarrollados. Además por la falta de información de los componentes desarrollados y la premura del desarrollo, los componentes o funcionalidades se están desarrollando por personas sin el conocimiento de la lógica del negocio. Lo cual influye directamente en la calidad y eficiencia del componente desarrollado, ya que su estructura y servicios seguramente no son los óptimos. Dentro de cada área de desarrollo existe un número de ingenieros encargados de la implementación de cada caso de uso, por ende del desarrollo de componentes que permitan la implementación de dicho caso, al no poseer información de existencia de software para reutilización, podrían crear nuevos componentes, servicios o funciones que en realidad ya estaban creadas y que por desconocimiento no son reutilizadas.

En la actualidad existen cerca de 30 ingenieros de desarrollo dedicados de tiempo completo al desarrollo de 5 sistemas de información, y 20 estudiantes de pregrado que contribuyen con el desarrollo dentro del marco de la realización de su práctica empresarial. Además, hay líderes de área quienes diseñan, organizar y supervisan el desarrollo de los sistemas, el mantenimiento y adecuación de los sistemas existentes; por la complejidad del desarrollo es prácticamente imposible revisar la reutilización o no de funciones existentes.

Por este motivos y otros como los beneficios que se pueden obtener por la utilización de un repositorio se hace indispensable el diseño, implementación e implantación de un repositorio de componentes propio de la DSI, que no solo permita el registro, consulta y acceso a los componentes desarrollados , sino que también ofrezca funcionalidades adicionales para los servicios y funciones desarrolladas, además de permitir observar la forma de implementación de un componente dentro del desarrollo de un nuevo sistema, verlo en funcionamiento antes de ser elegido, entre otras características.

Una vez analizada la situación que se presenta actualmente en el desarrollo de software en la DSI se puede formular el problema de investigación:

Problema de Investigación:

¿Cómo optimizar la disposición de componentes implementados para mejorar el proceso de desarrollo de software en la División de Servicios de Información?

Como la ISBC busca desarrollar productos software a partir del ensamblaje de módulos independientes llamados componentes. Estos componentes deben estar disponibles en una biblioteca o repositorio, donde cada ingeniero desarrollador los puede obtener para utilizarlos en el nuevo sistema. Al basarse en componentes ya existentes se tiene menos software por desarrollar y por tanto los nuevos sistemas se pueden desarrollar con más rapidez.

Con la propuesta de utilización de un repositorio se pretende centralizar en una librería todos los componentes software desarrollados con el fin de contar con un sitio único de almacenaje. De esta forma se podría pensar en tener una forma única y eficiente de registro, consulta, selección e implementación de componentes con el fin de evitar la duplicidad y la demora en la búsqueda e implementación de componentes ya desarrollados.

Entre los beneficios que se obtiene al contar con un repositorio de componentes tenemos:

- Centralización de los componentes.
- Facilidades de consulta de componentes.
- Mejor identificación de las funcionalidades de los componentes desarrollados.
- Aumento sustancial en la documentación, implementación y desarrollo de los componentes, funciones y servicios.
- Optimiza la selección de los componentes o servicios al verlos funcionando sin necesidad de su implementación en el nuevo sistema.
- Permite la realización de pruebas de forma más eficiente.
- Desarrollo de sistemas en menor tiempo.

Considerando la relevancia de contar con un repositorio de componentes de software, la pregunta planteada se puede desagregar en nuevas preguntas con el propósito de dividir el trabajo de investigación de forma estratégica.

Debido a que es necesario definir las características relevantes de un componente para poder especificarlo de forma acertada, y de esta forma, se pueda identificar un componente en particular dentro del repositorio, nace el primer interrogante a resolver:

Pregunta de Investigación 1:

¿Cómo identificar los componentes en el repositorio que cumplen los requerimientos para su reutilización?

Para el desarrollo de un repositorio de componentes reutilizables se requiere definir su estructura, es decir, se requiere definir la forma como se pueden relacionar los componentes entre si y el repositorio mismo, por lo tanto surge la siguiente pregunta de investigación:

Pregunta de Investigación 2:

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

También se deben tener en cuenta las características o funcionalidades que el repositorio debe proveer a los usuarios de tal forma que sea funcional. De esta forma se hace necesario hacer realizar procesos de análisis y diseño del repositorio con el fin de contar con la estructura apropiada que permita el registro de toda la información relevante para el repositorio. De esto se deriva un nuevo interrogante.

Pregunta de Investigación 3:

¿Qué estructura de datos se debe crear con fin de implementar un repositorio funcional y completo?

Y finalmente, se deben identificar los atributos o características que pueden establecer la correspondencia del componente encontrado con las funcionalidades o requisitos requeridos por el desarrollador. De esta necesidad o requerimiento se puede plantear el siguiente interrogante de investigación:

Pregunta de Investigación 4:

¿Cómo se debe catalogar o especificar un componente de software para su plena identificación?

1.4. OBJETIVOS

1.4.1. OBJETIVO GENERAL

Desarrollo e implementación de un repositorio de componentes para dar soporte a la práctica de la ingeniería del software basada en componentes de la División de Servicios de Información.

1.4.2. OBJETIVOS ESPECÍFICOS

- Diseñar e implementar un repositorio de componentes de software reutilizables para la División de Servicios de Información.

- Revisar las características relevantes de un componente, con el fin de diseñar la ficha de especificación de un componente.

- Refactorizar el procedimiento para especificación de componentes de software reutilizables.

- Especificar, almacenar y registrar por lo menos el 10% de los componentes de software implementados por el grupo de desarrollo académico de la División de Servicios de Información.

- Elaborar el esquema de almacenamiento, búsqueda, selección, distribución y utilización del componente desde el repositorio.

1.5. MARCO METODOLÓGICO

1.5.1. TIPO DE TRABAJO

El presente trabajo de investigación se basa principalmente en los conceptos de la Investigación descriptiva y la investigación tecnológica aplicada.

La investigación descriptiva tiene como objetivo describir y analizar lo que existe en un dominio de conocimiento con respecto a las variaciones o a las condiciones de una situación. Con ellos se obtiene información acerca de las características y comportamiento actual de fenómenos, hechos, conjunto de sujetos o áreas de interés.[14]. Para el caso del presente trabajo de investigación se pretende analizar y estudiar la forma de solucionar toda la problemática de la DSI generada por las falencias existentes en el modelo de desarrollo de software utilizado allí. Además se estudiará los beneficios y ventajas existentes que tiene el uso de un repositorio de componentes en este modelo de desarrollo.

En cuanto a la investigación tecnológica aplicada, en la que se debe poner en práctica los conocimientos a la solución de un problema práctico generalmente particular, se va a diseñar e implementar un repositorio de componentes software reutilizables para la División de Servicios de información, teniendo como base los fundamentos descritos en la parte de la investigación descriptiva.

1.5.2. ESTRATEGIAS DE RECOLECCIÓN DE INFORMACIÓN

[14] TOLEDO DÍAZ, Édison Yamir. Elementos de Metodología de la Investigación. La Habana – Cuba. Marzo 2002.

Para recopilar la información de la presente investigación se tendrá en cuenta las siguientes actividades:

- Consulta de bases de datos especializadas
- Consultas a profesores relacionados con el área de investigación, como también a desarrolladores de software expertos de la DSI.
- Consultas en artículos relacionados con el tema, publicados en revistas nacionales e internacionales, libros y tesis que tengan afinidad con el tema de investigación.
- Manuales y documentación en general de las empresas más importantes que proporcionan herramientas, frameworks y componentes para el desarrollo de software de Sun, dado que la DSI desarrolla sus componentes bajo este modelo.
- Tesis de investigación anteriores, relacionadas con el desarrollo de repositorios de componentes.
- Consulta a personal calificado de la DSI, con conocimiento en el proceso de desarrollo y las necesidades que puede suplir el repositorio.

1.5.3. 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 siguiente figura. 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 del repositorio de componentes, diseño e implementación del repositorio de componentes; 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. Estas actividades se pueden observar mejor en la siguiente figura.

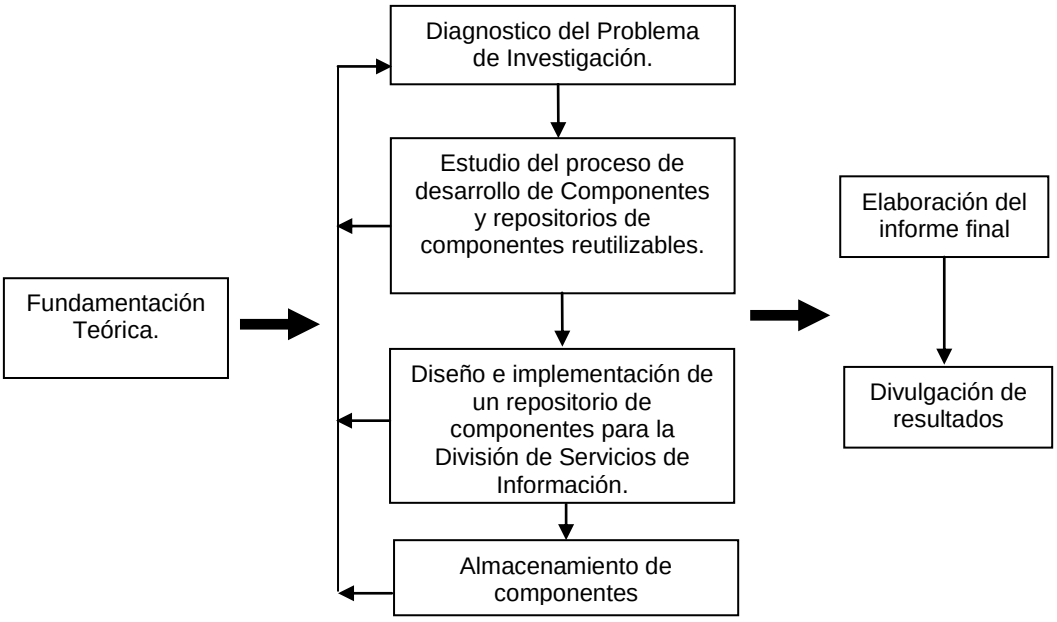

Figura 6. Proceso de Investigación

A continuación se detallan las actividades principales que se deben desarrollar en cada una de las fases de la presente investigación.

FASE 1: FUNDAMENTACIÓN TEÓRICA.

En esta fase se pretende formar el marco teórico de referencia que va ser la guía para la investigación.

Actividades

- Revisión y análisis de la literatura correspondiente al desarrollo de software basado en componentes.
- Revisión y análisis de artículos en bases de datos especializadas y en artículos de revistas a nivel nacional e internacional de investigaciones relacionadas con la ingeniería del software basada en componentes, para poder determinar las preguntas por resolver. (IEEE, ACM, InfoSCI)

- Revisión y análisis de artículos en bases de datos especializadas y en artículos de revistas a nivel nacional e internacional de investigaciones relacionadas con el desarrollo de repositorios de componentes, para poder determinar la estructura, arquitectura y funcionalidades del repositorio de implementar.
- Revisión y estudio de repositorios existentes, con el fin de determinar funcionalidades que puedan ser implementadas o requeridas en la DSI.
- Elaboración del estado del marco conceptual del desarrollo de software basado en componentes y repositorios de componentes.
- Elaboración del estado del arte del desarrollo de software basado en componentes en la DSI.

Resultados Esperados

- Marco Teórico correcto y completo para la investigación.
- Determinación de viabilidad e Impacto de la Investigación.

FASE 2: DIAGNÓSTICO DEL PROBLEMA DE INVESTIGACIÓN

En esta fase se pretende formular el problema de investigación, estableciendo el alcance y los objetivos que se van a llevar a cabo.

Actividades

- Identificación de los problemas del modelo de desarrollo de Software implementado en la DSI por la falta de centralización y documentación de los componentes desarrollados.
- Comunicación con desarrolladores de software para hacer un análisis de los problemas del desarrollo de software existentes y determinar posibles soluciones aplicables mediante la construcción e implementación del repositorio de componentes.
- Formulación del problema de Investigación y de las preguntas a resolver.

Resultados Esperados

- Identificación de la problemática existente en División de Servicios de Información.
- Determinación de los beneficios y ventajas de la utilización del repositorio de componentes en la DSI.

FASE 3: ESTUDIO DEL PROCESO DE DESARROLLO DE COMPONENTES IMPLEMENTADO EN LA DSI.

En esta fase se va a analizar el modelo de componentes software de SUN y el estándar Java EE 5, para así entender y poder especificar detalladamente como el repositorio de componentes puede mejorar el proceso de desarrollo de componentes software reutilizables en la DSI.

Actividades

- Estudio del modelo de componentes Enterprise Java Beans de Sun Microsystems, modelo seguido en la DSI.
- Estudio de las alternativas de arquitectura que se pueden utilizar para la implementación del repositorio de componentes reutilizables.
- Analizar y adaptar un procedimiento para especificación de componentes.
- Estudiar y modificar el procedimiento de desarrollo para reutilización.
- Definir criterios de búsqueda y selección de componentes en el repositorio, basándose en la especificación del componente.
- Determinar las alternativas de distribución de los componentes reutilizables, desde el repositorio de componentes.

Resultados Esperados

- Especificaciones requeridas y definidas de las funcionalidades que debe contemplar el repositorio y las falencias del modelo de desarrollo que va a atacar.

FASE 4: DISEÑO E IMPLEMENTACIÓN DEL REPOSITORIO DE COMPONENTES PARA LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN.

Actividades

- Análisis y diseño del repositorio de componentes que se va a implementar.
- Definición de la arquitectura del repositorio.
- Implementación del repositorio teniendo en cuenta las especificaciones y funcionalidades de la fase anterior.

Resultados Esperados

Un repositorio de componentes de software reutilizables implementado, probado y listo para ser utilizado en el desarrollo de sistemas informáticos en la DSI.

FASE 5: REGISTRO DE COMPONENTES.

En esta fase se va a detallar la forma de cómo se debe almacenar, buscar, seleccionar y observar un componente en el repositorio, con el fin de ilustrar y comprobar el buen funcionamiento del mismo.

Actividades

- Determinar los componentes que deben ser almacenados en el repositorio.
- Realizar la especificación de los componentes que van a ser almacenados.
- Pruebas de funcionalidad del repositorio con los componentes almacenados.

Resultado Esperado

Pruebas satisfactorias de funcionamiento del repositorio.

FASE 6: ELABORACIÓN DEL INFORME FINAL.

Actividades

- Recopilación de los informes preliminares de las etapas anteriores.
- Elaboración del informe final

Resultado Esperado

Documento final, con información acerca del diseño, implementación e implantación del repositorio de componentes, además de las soluciones y mejoras realizadas al proceso de desarrollo de software de la DSI.

FASE 7: DIVULGACIÓN DE RESULTADOS.

Actividades

- Creación y publicación de un artículo para ser divulgado en una revista especializada.

Resultado Esperado

Artículo que informa a la comunidad académica sobre los resultados alcanzados con el desarrollo de la investigación.

2. REPOSITORIO DE COMPONENTES DE LA DIVISION DE SERVICIOS DE INFORMACION.

En esta sección se explicara de forma detallada el trabajo realizado en el diseño, implementación e implantación del repositorio de componentes de la Division de Servicios de Informacion. En la actualidad el modelo de desarrollo por componentes adoptado por la DSI no cuenta con un repositorio de componente que le permita obtener los mejores resultados de este modelo.

2.1. ESPECIFICACIONES Y/O FUNCIONALIDADES.

Para la captura de los requerimientos del sistema repositorio se han seguido varias actividades, estas permitieron la captura de requerimientos por corrección de falencias en el modelo de desarrollo y requerimientos por funcionalidades deseadas dentro del modelo de desarrollo o mas puntualmente del repositorio.

Entre las actividades realizadas para la captura de requerimientos se encuentran:

- **Entrevistas con desarrollados y líderes de área.**

  De estas entrevistas con los desarrolladores o implementadores de los sistemas de información de la Universidad Industrial de Santander se obtuvieron requerimientos como:

  - **Registro de la especificación de componentes**: Los desarrolladores de los sistemas manifestaron la séria dificultad para buscar componentes desarrollados, ya que estos se implementan por las personas que lo requieren pero no existe algún método de divulgación del desarrollo de estos componentes, adicionalmente la descripción de su funcionalidad se encuentra inmersa en su código, y es verdaderamente difícil acceder a este código con el fin de buscar su funcionalidad y así determinar si este componente ofrece las funcionalidades requeridas o deseadas por el desarrollador.
  - **Desarrollo de los componentes por conocedores de la lógica de negocio**: La necesidad de desarrollar y avanzar en la producción de sistemas lleva a los desarrolladores a implementar los componentes que se requieren, sin importar que estos deberían ser desarrollados por otras personas, personas que conocen mejor la estructura de la base de datos, la lógica de negocio, y que pueden proveer mejor las funcionalidades futuras de este componente. Los componentes deben ser desarrollados bajo la supervisión y diseño de los líderes de área de cada área de desarrollo como los son, Area Academica, Recursos Humanos, Financiera, Bienestar entre otras.

    Si bien este requerimiento no tiene su mejora o su optimización mediante la implementación del repositorio de componentes, si

puede corregirse por medio de la inclusión de una política de desarrollo la cual contemple y normalice el proceso de desarrollo de compontes de la DSI.

- **Centralización de componentes desarrollados:** Entre las mayores falencias del modelo de desarrollo, se encuentra la falta de ubicación de los componentes desarrollados. En la actualidad los componentes se desarrollan por personas que los requieren, estos no se agrupan en paquetes que permiten ofrecer sus funcionalidades a otros desarrolladores de forma libre, ya que estos paquetes no se encuentran liberados en un solo sitio, son JAR que empaquetan una serie de componentes y funcionalidades que se encuentran almacenados en cada EJB de cada modulo desarrollado de cada sistema en particular. Para su utilización se hace necesario una búsqueda dispendiosa, esta consta de ir preguntando a cada desarrollador si este componentes existe o no. De encontrarlo, este debe obtenerse llevándose el EJB desarrollado para el sistema en donde se desea integrar.

  La centralización de los componentes facilita la búsqueda y selección del componente requerido, ya que de esta forma se buscará en un solo sitio el componente requerido, y de no encontrarse, se tendrá la plena seguridad de que este no existe o no ha sido implementado. El sitio de centralización es el repositorio de componentes, y cada componente debe registrarse en este para ser utilizado, para ello debe implantarse una política de desarrollo que obligue a utilizar el repositorio de componentes, esta política es de fácil adecuación, ya que la DSI cuenta con un paso de desarrollo en el que consiste la revisión de estandarización, y es allí donde se puede verificar la utilización de componentes registrados en el repositorio.
- **Criterios de búsqueda de componentes:** Los desarrolladores solicitaron o sugirieron las formas que según ellos sería más fácil la búsqueda de los componentes. Se definieron criterios de búsqueda por, palabras clave, funcionalidad, entidades que manipula, tablas

de datos que toca, área de desarrollo, descripción del componente entre otros.

La definición de estos criterios de búsqueda conlleva a la revisión y modificación de la ficha de especificación, esto con el motivo de registrar los nuevos atributos por los cuales el componente podrá ser buscado y seleccionado para implementación en un nuevo sistema.

- **Selección de componentes:** No solo se requiere la búsqueda de los componentes a través del repositorio, lo más probable es que unos criterios de consulta, generen o retornen una lista de componentes que ofrezcan la mismas funcionalidades, o parecidas. Para facilitar la labor de selección, se requiere la visualización de la ficha de especificación del componente seleccionado, esto para observar su información completa.

  Esta ficha de especificación debe poderse observar para cada componente que retorne de la consulta generada.
- **Prueba de funcionamiento:** Para comprender mejor la funcionalidad del componente seleccionado, se exigió la prueba de funcionalidad, esta consiste en poder ver el componente en funcionamiento cuando sea posible desde el mismo repositorio. Esta funcionalidad permite que el desarrollador comprenda y disipe cualquier duda acerca de la funcionalidad del componente y pueda observar los atributos de entrada del componentes, y los valores o parámetros de salida en el repositorio, este permite observar datos reales, ya que este tomara la información directamente del sistema de desarrollo, servidor que contiene un espejo de la base de datos de producción.
- **Componentes almacenables:** La jefatura de la DSI definió que por política de sostenibilidad y facilidad de mantenimiento, solo se almacenaran en el repositorio de componentes, componentes de software que se desarrollen en la DSI, la División tiene entre sus políticas de desarrollo, la no utilización de generadores de código ni la utilización de cajas negras, entendiendo por cajas negras código

software que no puede ser revisado ni corregido desde su código fuente.

- **Clasificación de los componentes:** Los componentes de la DSI se pueden clasificar de varias formas, pero entre las principales o el principal criterio de clasificación se definió como el tipo de componente desarrollado que puede ser, componente de interface, que es el componente que dentro de su estructura de funcionamiento contiene interfaces propias que le permiten ejecutarse sin necesidad de implementarse dentro de una forma de pantalla propia del sistema donde se esta ensamblando. El otro tipo de componente es el componente de servicio, este retorna datos de algún tipo primitivo, u objetos mapeados posiblemente agrupados en listas. Este tipo de componente requiere indudablemente una interface para poder mostrar la información que retorna, pero esta interface no es propia del componente.
- **Obtención del componente elegido:** Los desarrolladores deben poder obtener el componente elegido, para ello se hace necesario que el desarrollador pueda obtener no solo las funcionalidades del componente, sino también, información acerca de su funcionalidad, parámetros de entrada, forma de sus métodos, código para implementación, tablas o estructura de datos que procesa entre otros. Para ello se dispondrá de algunas funcionalidades que permitirán obtener estos datos de forma independiente.

- **Fundamentación acerca de repositorios de componentes.**
  - **Registro de utilización:** Para el cálculo de grado de reutilización de componentes, se hace necesario el registro de cada desarrollador que decida utilizar un componente del repositorio, para ello se implemento un registro de algunos datos del desarrollador y del sistema en el que se acoplara el componente elegido. Pero el cálculo del grado de reutilización no es la única funcionalidad de este registro, ya que con este registro se puede percibir o medir el impacto que se daría, si se decide modificar o

eliminar la funcionalidad de un componente desarrollado, implementado y en utilización.

- **Documentación genérica de implementación:** Con el fin de implementar algunas funcionalidades extras, se adiciono entre estas, la capacidad de guardar información de documentación, documentación acerca de código requerido para su implementación, esto con el fin de ser utilizado por desarrolladores principiantes, los cueles no tienen experiencia en el ensamble de componentes en el entorno de desarrollo DSI. Documentación de estándares de desarrollo entre otros que se definen por la DSI.

Resumiendo, entre las funcionalidades del repositorio de componentes se tienen: Almacenamiento, búsqueda, selección, identificación, distribución, clasificación, prueba, documentación, registro de utilización.

A continuación se muestran los casos de uso del desarrollador y el repositorio de componente.

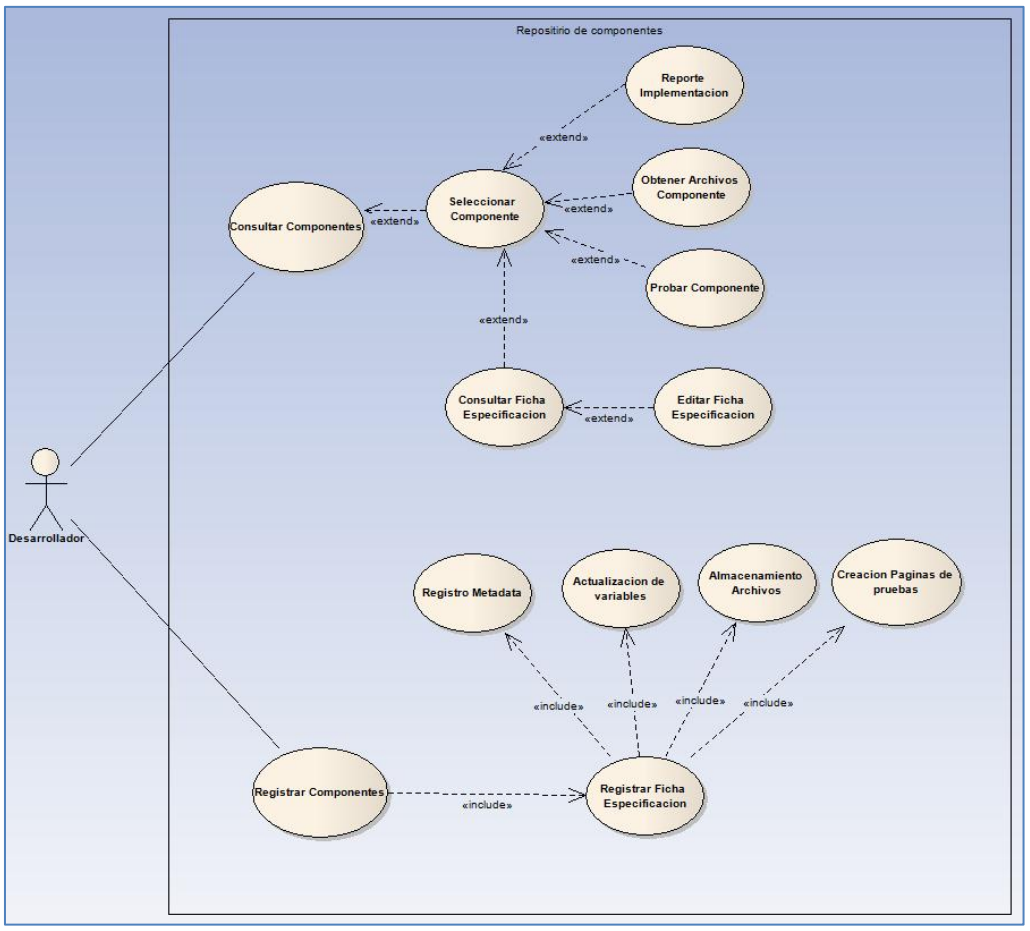

Figura 7. Casos de Uso repositorio de componentes

2.2. ALTERNATIVA DE ARQUITECTURA PARA EL REPOSITORIO DE COMPONENTES.

Una vez definidas las funcionalidades y requerimientos del repositorio de componentes, es necesario seleccionar el estilo arquitectónico que permita su implementación adecuada.

Bruegge y Dutoit[15] 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

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

global, manejo de condiciones de frontera y protocolos de intercomunicación de subsistemas.

Para el presente trabajo de investigación, la arquitectura o estilo arquitectónico define la forma como se interrelacionas los diferentes componentes del repositorio (propios y almacenados) y define el medio de comunicación entre estos y los desarrolladores.

El repositorio de componentes que se pretende desarrollar se debe basar en el modelo arquitectónico Java EE 5 y deben seguir el modelo Java EE 5. La arquitectura básica y general de las aplicaciones empresariales se puede apreciar en la figura.

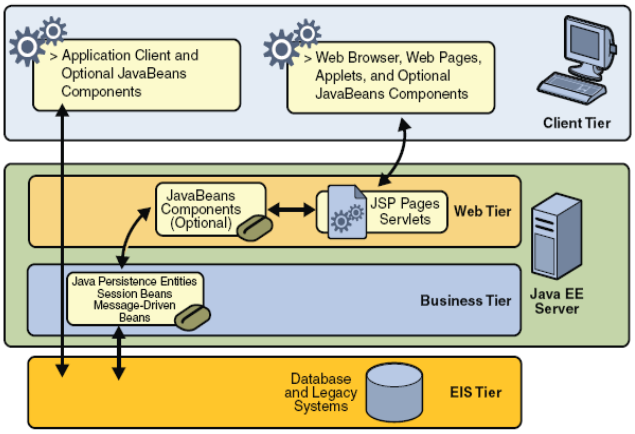

Figura 8. Arquitectura Aplicaciones Java EE 5[7].

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 (SOA).

En la siguiente tabla se muestran las principales características de los estilos arquitectónicos resaltando sus ventajas, desventajas y cuando utilizar cada uno[16].

Tabla 1. Comparación de las Alternativas de arquitectura de los componentes.

COMPARACIÓN DE LAS ALTERNATIVAS DE ARQUITECTURA DE LOS COMPONENTES		
CARACTERÍSTICAS PRINCIPALES		
M.V.C.	**CAPAS**	**SOA**
Modelo: Administra el comportamiento y los datos del dominio de aplicación. **Vista:** Maneja la visualización de la información. **Controlador:** Interpreta las acciones sobre la vista, informando al modelo y/o a la vista para que cambien según resulte apropiado. La vista y el controlador dependen del modelo, el modelo no depende de otros componentes. Esto permite construir y probar el modelo independiente de la representación visual. Mantiene un acoplamiento entre la vista y el modelo, y el controlador y el modelo.	Se puede definir la arquitectura en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la capa inferior. Existe en teoría un bajo acoplamiento entre las capas. Un ejemplo típico de una arquitectura por capas es el modelo OSI.	Un servicio es una entidad de software que encapsula funcionalidad de negocios y proporciona dicha funcionalidad a otras entidades a través de interfaces públicas bien definidas. La arquitectura orientada a servicios pretende la exposición y uso de servicio por parte de los sistemas informáticos. Esta arquitectura se fundamenta principalmente en los servicios web. Un servicio web es un sistema de software diseñado para soportar interacción máquina – a - máquina sobre una red. Posee una interfaz descrita en un formato procesable por máquina (específicamente WSDL). Otros sistemas interactúan con el Web service de una manera prescripta por su descripción utilizando mensajes SOAP, típicamente transportados usando HTTP con una serialización en XML.
VENTAJAS		
M.V.C.	**CAPAS**	**SOA**
- Soporte de vistas múltiples, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente y en diferentes dispositivos. - Adaptación al cambio, dado que el modelo no depende de las vistas,	- Soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.	- Se puede reemplazar un servicio sin tener que preocuparse por la tecnología fundamental; la interface es lo que importa, y está definida en un estándar universal en servicios Web y XML. - Es una arquitectura distribuida, la

[16] GUIJUN, Wang; CASEY K, Fung. Artículo: Architecture Paradigms and Their Influences and Impacts on Component-Based Software Systems. Proceedings of the 37th Hawaii International Conference on System Sciences – 2004

agregar nuevas opciones de presentación generalmente no afecta al modelo. - Bajo acoplamiento: desacopla las vistas de los modelos y desacopla los modelos de la forma en que se muestran e ingresan los datos. - Alta cohesión: Cada elemento del patrón está altamente especializado en su tarea. - Claridad en el diseño - Alta escalabilidad. - Se puede presentar como una adaptación de la arquitectura por capas.	- Admite muy naturalmente optimizaciones y refinamientos. - Proporciona amplia reutilización, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces. - Facilita la migración. El acoplamiento con el entorno está localizado en las capas inferiores. - Permite trabajar en varios niveles de abstracción. Para implementar los niveles superiores no necesitamos conocer en entorno subyacente, solo las interfaces que proporcionan los niveles inferiores.	carga de las transacciones se pueden dividir en diferentes servidores comunicándose todos por medio del los servicios web. - Los elementos de esta alternativa están débilmente acoplados. - El servicio puede recibir requerimientos de cualquier origen. - La funcionalidad del servicio se puede ampliar o modificar sin rendir cuentas a quienes lo requieran. - Los componentes que requieran un servicio pueden descubrirlo y utilizarlo dinámicamente. - Alta interoperabilidad con otras plataformas.
DESVENTAJAS		
M.V.C.	**CAPAS**	**SOA**
- Complejidad, esta alternativa introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad. También se profundiza la orientación a eventos del código de la interfaz de usuario, que puede llegar a ser difícil de depurar. - Costo de actualizaciones frecuentes. Si el modelo cambia frecuentemente puede desencadenar muchos requerimientos de actualización en las vistas que utilizan el modelo.	- Muchos problemas no admiten un buen mapeo en una estructura jerárquica. Incluso cuando un sistema se puede establecer lógicamente en capas, consideraciones de rendimiento pueden requerir acoplamientos específicos entre capas de alto y bajo nivel. A veces es también extremadamente difícil encontrar el nivel de abstracción correcto. - Al establecer el llamado solo de capas superiores a capas inferiores es posible que el rendimiento no sea bueno.	- La velocidad de intercambio de información es más lenta que una conexión directa entre los elementos. - Su implantación es compleja y se requiere de grandes esfuerzos en su desarrollo. - Los clientes necesitan saber las operaciones y su semántica antes del uso.
CUÁNDO UTILIZAR CADA ALTERNATIVA		
M.V.C.	**CAPAS**	**SOA**
- Se recomienda utilizar este estilo de arquitectura cuando se necesitan muchas vistas o presentaciones del mismo modelo. - También es recomendada utilizar esta alternativa cuando se necesiten elaborar interfaces de usuario complejas y para diferente clase de dispositivos. - Este tipo de arquitectura es el más usado en aplicaciones web.	- Se puede utilizar la arquitectura por capas, cuando se pude estratificar los elementos que conforman el componente en diferentes categorías y no exista un marcado acoplamiento entre estas categorías. - Es útil en sistemas complejos, que manejen muchas transacciones y que puedan cambiar, al dividir por capas es más fácil cambiar una capa sin afectar notablemente las otras. - La arquitectura propuesta por Java para aplicaciones empresariales es por capas, implementa las siguientes capas: nivel de persistencia, nivel de servicios, nivel de aplicación (Web) y el nivel de presentación.	- Se recomienda utilizarla cuando se necesite distribuir los componentes en diferentes maquinas y que funcionen en diferentes plataformas a las cuales fueron implementados. - Es ideal utilizar este estilo arquitectónico cuando el sistema o componente es basado en mensajes, en especial mensajes asíncronos. - Para sistemas multiplataforma es ideal utilizar esta alternativa, debido a que los servicios web se basan en un estándar XML unificado.

Es importante resaltar que en muchos casos, la mejor solución para determinar que arquitectura utilizar en el desarrollo de un sistema informático o un componente es un hibrido de las tres arquitecturas propuestas tomando lo mejor de cada una y haciendo los ajustes necesarios.

Para el caso particular de esta investigación se tomará la arquitectura modelo-vista-controlador. Estilo arquitectónico seguido para el desarrollo de componentes de la DSI. Cabe anotar que el repositorio de componentes se desarrollará bajo los mismos estándares y políticas que direccionan el desarrollo de software de la División de Servicios de Información. Razón por la que el estilo arquitectónico elegido para este desarrollo cuenta con las mismas ventajas y razones que las observadas en el trabajo de investigación desarrollado para la implementación del modelo de desarrollo por componentes para la DSI.

2.3. ARQUITECTURA MODELO–VISTA–CONTROLADOR.

Franky[17] 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 figura 9.

En una parte del modelo estarían las entidades soportadas por el API de persistencia de Java que se encarga de mapear las entidades en java con la base de datos relacional, también se utiliza el framework Jboss Seam.

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

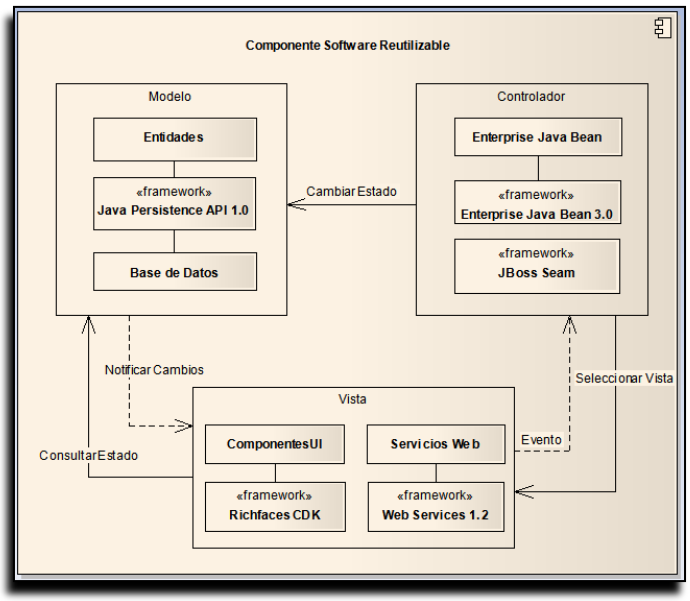

Figura 9. Arquitectura Modelo – Vista – Controlador

En la vista se ubican los componentes o controles para la interfaz de usuario, desarrollados por medio del kit de desarrollo de componentes de Richfaces, si en este repositorio de componentes se permitiera almacenar otros servicios no desarrollados en JAVA, se ofrecerían servicios web, que exponen las funcionalidades del componente de forma remota.

El control seria manejado por los EJBs soportados por los frameworks Enterprise Java Beans 3.0 y Jboss Seam. Los controladores son los encargados de manejar la lógica del negocio del repositorio, se encargan de procesar la información suministrada por los clientes y presentar los resultados.

2.4. DESCRIPCIÓN DEL REPOSITORIO.

El repositorio de componentes de la División de Servicios de Información, permite mejorar el proceso de desarrollo de la DSI, a medida que disminuye la duplicación de componentes, mejora el proceso de búsqueda y selección de componentes, además de estandarizar y unificar el proceso de registro o especificación de componentes.

Las principales actividades que se pueden realizar en el repositorio de componentes son:

- Registro y consulta de archivos los cuales contienen la lógica del negocio de los componentes, estos archivos son llamados dentro del repositorio "*paquetes*".
- Registro y consulta de los componentes desarrollados, para ello se implementaron varias opciones de consulta como palabras clave, funcionalidades entre otras.
- Prueba y ejecución del componente seleccionado, esta opción permite al desarrollador verificar si el componente realiza las funciones requeridas para el nuevo sistema software.
- Revisión e impresión de la ficha de especificación del componente, la cual le permitirá obtener la información necesaria para su implementación o reutilización.
- Registro de reutilización de componente, la cual le permite a los desarrolladores obtener información acerca de los sistemas donde se encuentra reutilizado dicho componente, y a futuro, realizar estadísticas u obtener datos para efectuar proyecciones de afectación de sistemas en el caso de retirar o modificar algún componente utilizado en varios sistemas.

2.5. DESARROLLO DEL REPOSITORIO DE COMPONENTES.

Con los requerimientos funcionales y no funcionales definidos, se inicia el desarrollo del repositorio de componentes, para ello, se crea el proyecto inicial, proyecto con una estructura definida según los estándares de desarrollo de la División de Servicios de Información DSI de la Universidad Industrial de Santander.

También es muy importante tener definidas las herramientas que se van a utilizar para la codificación, prueba y distribución del componente cómo son: servidor de aplicaciones, controlador de versiones, IDE de desarrollo (eclipse, netbeans, jcreator, etc), herramientas para compilación, distribución y pruebas.

El modelo comienza con la implementación de las entidades, actividad que pretende realizar el mapeo objeto – relacional entre las clases java y la base de datos relacional, continua con las pruebas unitarias de este mapeo, con las cuales se pretende garantizar que el mapeo quedó bien realizado. Luego se realiza la implementación de los servicios que va a ofrecer el componente por medio del los frameworks EJB 3.0 y JBoss Seam, después se realizan la pruebas funcionales de estos servicios para verificar que quedaron bien implementados. Posteriormente se implementan los controles personalizados para la interfaz de usuario, los cuales permiten la interacción entre el usuario o desarrollador y el repositorio de componentes. Cada actividad se define con más detalle a continuación.

2.5.1. IMPLEMENTACIÓN DEL REPOSIOTORIO.

La fase de implementación o desarrollo del repositorio tiene varias etapas, estas etapas o actividades se pueden agrupar de la siguiente forma:

- **Implementación de entidades.**

  El objetivo principal de esta fase es convertir el modelo de entidades o modelo del dominio del problema en las clases de entidad implementadas y mapeadas con la respectiva base de datos relacional. El mapeo se realiza en Java versión empresarial 5, utilizando el api de persistencia de Java, el JBoss Seam y el framework EJB 3.0. En la figura 10 se presenta el ejemplo de la implementación de la entidad *"Componente"* en la cual se puede apreciar las principales anotaciones que se utilizan para definir una entidad y realizar su mapeo con la base de datos.

```java
package co.edu.uis.repositorioComponentes.entidades;

import java.io.Serializable;

@Entity
@Table(name = "repositori_comp_j5:componente")                    1
public final class Componente implements Serializable {

    private static final long     serialVersionUID = -8791585824461026539L;
    private Integer               id;
    private String                nombre;


    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)                 2
    @Column(name = "id")
    public Integer getId() {
        return id;
    }

    public void setId(Integer componente) {
        this.id = componente;
    }

    @Column(name = "nombre_componente")                           3
    public String getNombre() {
        return nombre;
    }
```

Figura 10. Implementación de Entidad Componente.

En (1) se define la clase como entidad con la anotación *@Entity*, con la anotación *@Table* se mapea la clase con la tabla *componente* en la base de datos repositorio_comp_j5, y la anotación *@Name* de seam se le da un nombre a la entidad para ser conocida en el contenedor de EJBs y en el servidor de aplicaciones.

En (2) se define la llave primaria de la entidad con la anotación *@Id* y se mapea el atributo de la clase *nombre* con el campo *nombre_componente* de la tabla. De esta forma se realiza para el resto de atributos de la entidad.

En (3) se define la un atributo llamado nombre, mapeado al atributo nombre_componente.

```
@ManyToOne(fetch=FetchType.EAGER)
@JoinColumn(name = "sistema", referencedColumnName="codigo_sistema")
public SistemaInformacion getSistemaInformacion() {
    return sistemaInformacion;
}

public void setSistemaInformacion(SistemaInformacion sistemaInformacion) {
    this.sistemaInformacion = sistemaInformacion;
}
```

Figura 11. Implementación de Entidad Componente - 2.

En la figura 11 aparece un listado que representa una relación de uno a muchos con otra entidad, se define por medio de la anotación *@OneToMany* con los atributos *mappedBy* y *fetch* que indican el atributo de la entidad con el cual se relaciona y el modo de carga del listado que en este caso es *Lazy* (peresoso) esto quiere decir que no realiza la consulta del listado cuando se instancia la entidad sino cuando se hace el llamado al método *get* correspondiente al atributo; En algunas ocasiones se utiliza la anotación *@Orderby,* esta ordena el listado por el atributo deseado.

El anterior es un ejemplo básico de cómo implementar una entidad, la cual consta principalmente de los elementos mencionados con anterioridad: anotaciones de definición de la entidad en la cabecera de la clase, atributos mapeados a los campos de la tabla que representan, llave primaria y relaciones con otras entidades, en ocasiones la llave primaria es una llave compuesta por varios campos, para mapear esta llave primaria se debe definir otra clase, la cual debe ir como atributo de la entidad que estamos mapeando y se debe utilizar la anotación *@EmbeddedId* para indicar la llave primaria. También utilizando anotaciones es posible incluir validaciones de datos, de rangos, de formatos, entre otros. En la tabla siguiente se presentan las anotaciones principales para mapear una entidad. En el libro EJB in Action, capitulo 7 página 217, se

encuentra una guía detallada de cómo mapear las entidades y como implementar el modelo del dominio del problema en Java versión empresarial.
Las entidades a mapear se pueden observar el em modelo de datos de proyecto.

Para el repositorio de componentes se mapearon las entidades encontradas en el modelo de datos, entre las que se encuentran:

- BaseDatos.
- Componente.
- Entidad.
- EntidadComponente.
- EntidadComponenteId.
- EstadoComponente.
- Funcionalidad.
- FuncionalidadComponente.
- FuncionalidadComponenteId.
- PalabraClave.
- Paquete.
- ParametroComponente.
- RegistroUsoComponente.
- TipoComponente.

Como puede observarse, dentro de la entidades mapeadas, se encuentran las entidades *EntidadComponenteId* y *FuncionalidadComponenteId,* Estas entidades nacen de la necesidad de mapear llaves primarias compuestas, es decir llaves primarias que contienen más de un atributo.

Una vez implementadas las entidades es necesario crear el archivo de configuración *orm.xml*, donde se deben declarar las entidades para ser interpretadas por el servidor de aplicaciones. El framework Hibernate tiene herramientas muy útiles que permiten la creación automática de este archivo y la conversión de una base de datos relacional a las entidades mapeadas.

El resultado de esta fase es un archivo .jar que contiene el conjunto de clases de entidad y los archivos necesarios de configuración para ser distribuido en el servidor de aplicaciones.

Como el repositorio de componentes estará funcionando sobre el servidor de producción, y allí se encuentran todos los archivos .jar de las entidades que reconoce el servidor. El repositorio de componentes tiene acceso directo a todas estas entidades, esto es requerido para que el repositorio pueda interpretar todas y cada una de las entidades manipuladas por los componentes, de estar forma determinar que atributos los componen y que tipo de datos poseen.

- **Pruebas unitarias del mapeo de entidades.**

  Teniendo implementadas y mapeadas las entidades es necesario realizar pruebas unitarias a cada entidad para garantizar que el mapeo quedó realizado correctamente y garantizar también la calidad de los objetos del dominio del problema a utilizar en el desarrollo de todo el componente. Para la realización de las pruebas unitarias se puede utilizar el api java llamado JUnit que automatiza las pruebas generando las clases específicas de prueba y los métodos para su ejecución. En la tabla siguiente se detalla una guía para la realización de las pruebas de unidad independientemente si

se utiliza el JUnit. En la tabla se destacan las pruebas principales a realizar, el objetivo de cada prueba y el resultado esperado.

Antes de iniciar las pruebas planteadas es necesario introducir datos de prueba a las tablas de la base de datos relacional sobre la cual se realiza el mapeo. Las sentencias JPQL y los métodos necesarios para realizar las pruebas se pueden implementar utilizando el JUnit o realizando una aplicación web de prueba, en la cual se crea una página xhtml, un EJB y se hacen los métodos respectivos para cada prueba. Con las herramientas del framework JBoss Seam es fácil crear una aplicación web de prueba.

Tabla 2. Pruebas unitarias del mapeo de las entidades.

PRUEBAS UNITARIAS DEL MAPEO DE LAS ENTIDADES		
Descripción Prueba	**Objetivo**	**Resultado Esperado**
Probar mapeo de la entidad con la tabla asociada: Se debe crear una sentencia simple JPQL que consulte los datos de la entidad. Ejemplo: SELECT * FROM Componente.	Verificar que el mapeo de la entidad con la tabla asociada fue hecho correctamente.	Al ejecutar la consulta JPQL se debe obtener como resultado los registros de prueba introducidos. Si se presentó algún error en el mapeo no se devuelve ningún registro.
Probar las relaciones de la entidad con otras entidades: Crear un método que permita realizar la prueba de las relaciones mapeadas en la entidad. Ejemplo: La entidad componente tiene mapeada una relación uno a muchos con la entidad palabras clave. El código de prueba seria: EntityManager em; Componente clase = em.merge(c); List<PalabraClave> u = clase.getPalabraClave(); System.out,println(u);	Verificar que el mapeo de las relaciones de entidades se realizó correctamente.	Al ejecutar el método se deben obtener los registros de las entidades relacionadas. Si se presenta error en el mapeo no se obtiene ningún registro o se evidencia alguna excepción.

Donde c es el componente que se recibe como parámetro en el método.		

Probar la inserción de un registro en la tabla de persistencia asociada: Hacer un método que implemente un persist sobre la entidad. Ejemplo:
 	
EntityManager em;
Componente c = new Componente();
c.setId(1);
c.setNombre("ConsultarProgramas");
em.persist(c);
	Verificar que la inserción de datos o la persistencia automática funcionan correctamente.	Al ejecutar la prueba el registro quedó insertado en la base de datos.

Probar la eliminación de un registro de la tabla de persistencia: Elaborar un método que implemente la eliminación de un registro. Ejemplo: EntityManager em; em.remove(entityManager.merge(c)); Donde c es la palabra clave recibida como atributo en el método.	Verificar que la eliminación de datos de la tabla de persistencia se realiza correctamente.	Al ejecutar la prueba el registro fue eliminado de la base de datos.

Probar la actualización de un registro de la tabla de persistencia: Elaborar un método que implemente la actualización de un registro. Ejemplo:
 	
EntityManager em;
Funcionalidades c = new Funcionalidad();
c.setCodigo(1);
c.setDescripcion("Actualizacion");
em.merge(c);
	Verificar que la actualización de datos en la tabla de persistencia se realiza correctamente.	Al ejecutar la prueba el registro es actualizado en la base de datos.

Una vez finalizadas y aprobadas las pruebas de todas las entidades se puede pasar a la implementación de los servicios que va a prestar el repositorio de componentes de software. Es importante resaltar

que al realizar cambios sobre las entidades es necesario realizar o correr las pruebas descritas en la tabla anterior nuevamente.

- **Implementación de servicios.**

  En esta fase se pretende crear los servicios que va prestar el repositorio de componentes, en primer lugar es importante aclarar la definición de servicio, según Wang y Fung[18] "un servicio es una unidad software que encapsula funcionalidades del negocio y provee las funcionalidades a sus clientes a través de interfaces bien definidas y publicadas". Por lo tanto un servicio puede ser un método de un EJB, que va ser utilizado por sus clientes a través de una interfaz bien definida.

  La forma de implementar los servicios en Java EE 5 es utilizando el framework EJB 3.0 y JBoss Seam. Estos frameworks contienen un conjunto de clases que facilitan la tarea de implementación, distribución y utilización de EJBs. En el libro EJB in Action se encuentra una guía detallada para la implementación y uso de EJBs.

  Dentro de las buenas prácticas planteadas por los autores del libro EJB in Action plantean dos patrones para la implementación de los EJBs: el patrón Facade y el patrón EAO (Entity Access Object) los cuales permiten una mejor distribución e independencia de los elementos que se utilizan dentro de los EJBs reduciendo el acoplamiento y aumentado la cohesión de los servicios implementados. En la siguiente figura se puede apreciar el flujo de datos por medio de estos patrones.

[18] GUIJUN, Wang; CASEY K, Fung. Artículo: Architecture Paradigms and Their Influences and Impacts on Component-Based Software Systems. Proceedings of the 37th Hawaii International Conference on System Sciences – 2004

Figura 12. Patrones Facade y EAO.

El cliente hace peticiones de los servicios invocando métodos del EJB del repositorio que implementa el patrón Facade, y recibe la respuesta a través de este mismo EJB. La clase que implementa el patrón Facade se encarga de desarrollar la lógica del negocio de los servicios que va a prestar el repositorio, servicios de registro, búsqueda y selección entre otros. El EAO se encarga de controlar el acceso a los métodos de manipulación de las entidades, es decir los métodos de persistencia, modificación, eliminación y consulta de información, el EAO divide la lógica del negocio de las operaciones que tienen que ver con la base de datos. Las clases que implementan cada patrón tienen su interfaz bien definida, por lo tanto permite cambiar la implementación de la clase sin afectar la funcionalidad final del sistema.

Para hacer la implementación tanto del patrón Facade como del EAO se deben utilizar EJBs de sesión. En la figura se presenta la implementación del patrón facade, se pueden apreciar la estructura básica de un EJB de sesión con las anotaciones fundamentales que se pueden utilizar.

Sin embargo, para el desarrollo del repositorio de componentes, no es necesaria la implementación del patrón, Ya que las librerías, servicios y métodos desarrollados, no se reutilizaran en otros sistemas de la Division de Servicios de Información, esto debido a que el repositorio no hace parte de la familia de componentes

desarrollados para la creación de los sistemas de software de la Universidad.

A continuación, en la figura 13 se puede observar la estructura utilizada en el desarrollo de los servicios del repositorio de componentes.

En (1) se definen las anotaciones de definición de EJB de sesión, en este caso se utiliza *@Statefull(name="serviciosRepositorio")* indicando un sesión bean sin estado y el name indica el nombre del EJB en el contenedor. La etiquete *@Name* de JBoss Seam, permite definir un nombre para el EJB con el cual es reconocido por JDNI, en el contenedor Seam y en el servidor de aplicaciones EJB.

```
package co.edu.uis.repositorioComponentes.pruebasComponentes;

import java.io.File;

@Stateful                                                    1
@Name("serviciosRepositorio")
public class ServiciosRepositorioEJB implements ServiciosRepositorio{  2

    @PersistenceContext(type = PersistenceContextType.EXTENDED)        3
    private EntityManager entityManager;

    /**
     * Este método se encarga de enviar la petición de verificaión de la existen
     * @param aDireccionJar Dirección donde está ubicado el archivo
     * @param aPaquete Nombre del Archivo.
     * @return true en caso de que exista el archivo, false en caso contrario.
     */
    public boolean isExtenteJar(String aDireccionJar, String aPaquete) {  4
        File fichero = null;
        boolean isExistentePaquete = false;

        fichero = new File(aDireccionJar+"//"+aPaquete+".jar");

        if(fichero.exists()){
            isExistentePaquete = true;
        } else {
            isExistentePaquete = false;
        }

        return isExistentePaquete;
    }
```

Figura 13. Código para la implementación del patrón Facade.

En (2) se implementa la interfaz ServiciosRepositorio, es la interfaz que sirve de especificación de los métodos que tiene el EJB. Todo EJB debe poseer una interfaz bien definida.
En (3) se definen los atributos propios del EJB.
En (4) se encuentra la definición de los métodos utilizados.

Los resultados de esta fase son:

- Archivo *.jar con las clases implementadas.
- Archivo *.ear con los EJBs y archivos de configuración necesarios para utilizar los EJBs.
- Archivo ds.xml, datasource que define la cadena de conexión con la base de datos.

- **Pruebas funcionales de los servicios.**

Esta fase pretende en primer lugar garantizar que los servicios fueron implementados correctamente y que cumplen con la totalidad de los requisitos funcionales planteados para el presente repositorio. En la tabla siguiente se describen las pruebas a realizar sobre los servicios.

Tabla 3. Pruebas funcionales de los servicios.

PRUEBAS FUNCIONALES DE LOS SERVICIOS		
Descripción Prueba	**Objetivo**	**Resultado Esperado**
Prueba Unitaria de servicios: Esta prueba consiste en probar cada uno de los métodos de acceso a entidades implementados en el	Verificar que los métodos de acceso a entidades están implementados correctamente.	Cada método se debe ejecutar sin generar ninguna excepción ni error y los resultados devueltos deben ser los correctos.

EJB.		
Prueba de requisitos: Esta prueba consiste en verificar las funcionalidades implementadas contra los casos de uso definidos en la etapa inicial, para asegurar que el repositorio cumple con la totalidad de los requisitos planteados. Se puede realizar por medio de una lista de chequeo, donde se especifique cada requisito.	Asegurar que el repositorio cumple con todos los requisitos planteados en los casos de uso.	Cumplimiento de la totalidad de los requisitos.

Las pruebas unitarias de servicios se puede realizar utilizando JUnit, el cual permite automatizar las pruebas, también se pueden probar al igual que en el caso de las entidades por medio de una aplicación de prueba donde se implementen y ejecuten los métodos necesarios para realizar dichas pruebas.

- **Implementación de interfaz de usuario.**

  Con la creación de la interfaz de usuario se pretende utilizar la lógica del negocio implementada en el repositorio.

  Si el repositorio de componentes tuviera una lógica de negocio que se debiera reutilizar en otros sistemas, podría desarrollarse por medio de componentes personalizados JSF, para el caso particular del repositorio de componentes, esta lógica no aplica para lógica reutilizable, razón por la cual su implementación no se llevará a cabo como componente UI, las cuales son comúnmente, consultas, donde el usuario selecciona un elemento de un conjunto de datos para

luego aplicar un proceso sobre el elemento seleccionado o manejarle una acción, Por ejemplo, en la universidad, una consulta de programas académicos que se filtre por diferentes criterios para buscar y seleccionar un programa en particular.

Para la implementación de interfaces de usuario, se hace necesario definir los respectivos usuarios del repositorio de componentes. Estos son Ingeniero desarrollador administrador del repositorio. Cada uno de ellos cuenta con un menú de navegación que permite acceder a las utilidades que este requiere del repositorio.

- Rol Administrador: Este rol permite ejecutar todas las acciones posibles dentro del repositorio, entre las cuales se destacan, el mantenimiento y registro de los paquetes de servicios o componentes desarrollados por cada una de las áreas de desarrollo de software de la División de Servicios de Información y el registro de componentes o métodos previamente verificado su funcionamiento.

- Rol Desarrollador: Este rol permite la consulta de los componentes registrados en el sistema, su ejecución si es posible, la revisión de la ficha de especificación del componente, el registro de utilización del componente, entre otros.

Para acceder a estas funcionalidades, el repositorio de componentes dispone de unas interfaces gráficas y un menú de navegación, a continuación se muestran algunas de la interfaces desarrolladas para el repositorio de componentes.

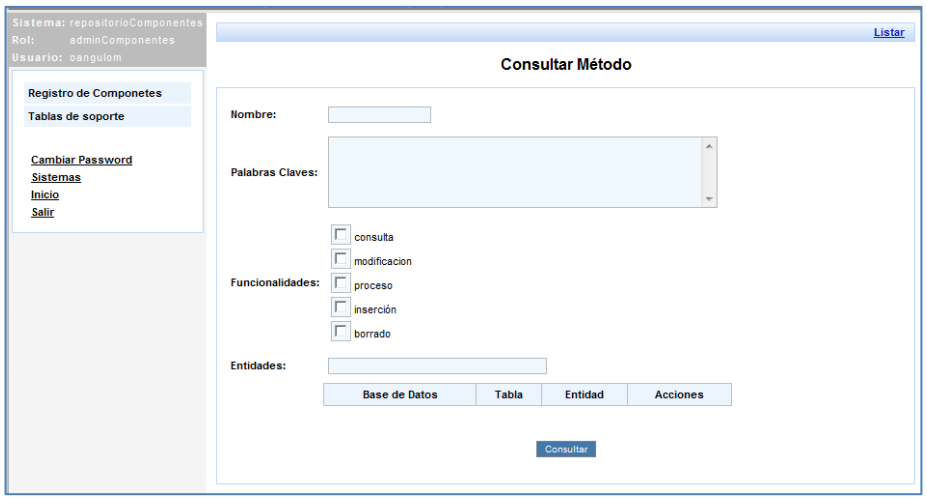

Figura 14. Interface de usuario – Consultar componentes.

En la figura 14, se puede observar en su parte izquierda, el menú de navegación, este permite al usuario, acceder a las opciones del menú, estas agrupadas por módulos. Además se observa, las diferentes formas de cómo se puede consultar los componentes registrados en el repositorio. Entre las opciones de búsqueda están:

Nombre del componente: Esta permite buscar los componentes registrados, que dentro de su nombre contengan el texto allí digitado.
Palabras Clave: Para cada componente, En el momento de registro se debe digitar algunas palabras que permitan identificar de forma rápida, las funcionalidades, tablas, entidades, accesos o atributos que tengan relación directa con el componente registrado.
Funcionalidades: Esta opción de búsqueda permite filtrar desde su búsqueda, los componentes según su actividad básica.
Entidades: Así como en la ficha de especificación del repositorio de componentes, se permiten adicionar palabras clave o funcionalidades, se permite también el registro de las tablas o entidades con las cuales el componente interactúa, por medio de esta opción de búsqueda, el repositorio permite al desarrollador

consultar los componentes que tienen relación con entidades o tablas de la base de datos.

El usuario puede accionar de forma mixta las opciones de consulta, de esta forma podría ejecutar una consulta que filtre por varias opciones.

En la siguiente figura, se observa como el repositorio busca de forma automática las entidades que están relacionadas con el texto digitado en esta caja, este buscador realiza su consulta utilizando los mapeos, y para ellos encuentra buscando por el nombre de la entidad o por el nombre de la tabla mapeada.

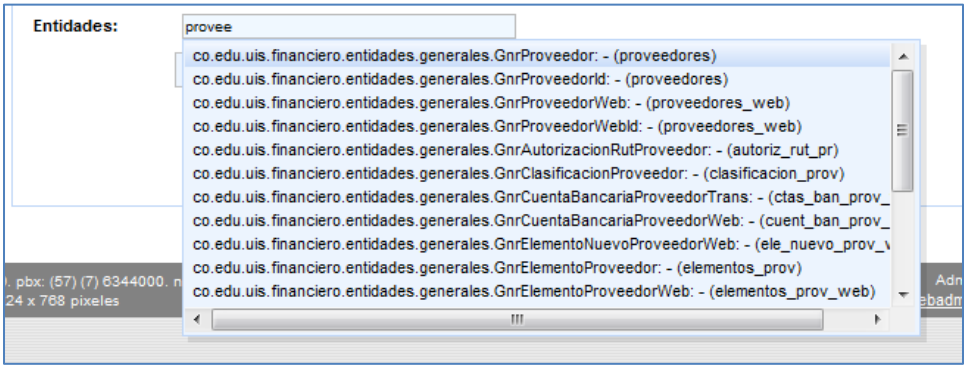

Figura 15. Interface de usuario – Consultar componentes (Entidades-1).

Después del registro de cada tabla o entidad, el repositorio registra cada opción en la grilla que se muestra en la figura siguiente, dentro de cada opción de consulta, el repositorio utiliza Un OR como opción de búsqueda, es así como él puede realizar la consulta buscando componentes que se relacionen con una entidad u otra en la grilla, este principio aplica para las otras opciones de consulta como palabras clave o funcionalidades.

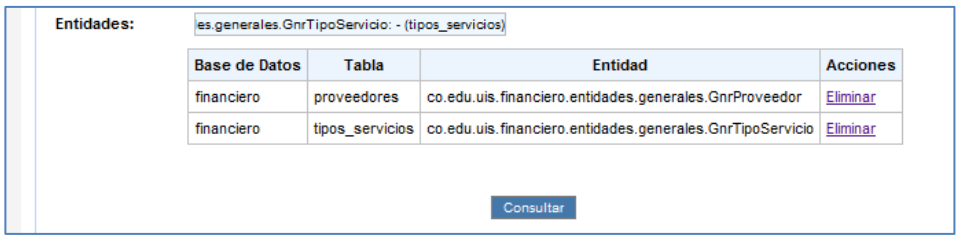

Figura 16. Interface de usuario – Consultar componentes (Entidades-2).

El usuario, después de seleccionar las opciones de consulta, y ejecutar la consulta por medio del botón de consultar, Observa los diferentes componentes registrados, que cumplen con las opciones elegidas.

De forma preliminar se muestran al usuario algunos datos básicos de los componentes como: Nombre, Usuario que registró el componente, Sistema o área de desarrollo y algunas acciones que pueden ejecutarse sobre cada uno de ellos. Entre las acciones a ejecutarse se encuentran, Ver ficha de especificación, Editar Ficha de especificación del componente y registrar uso del componente.

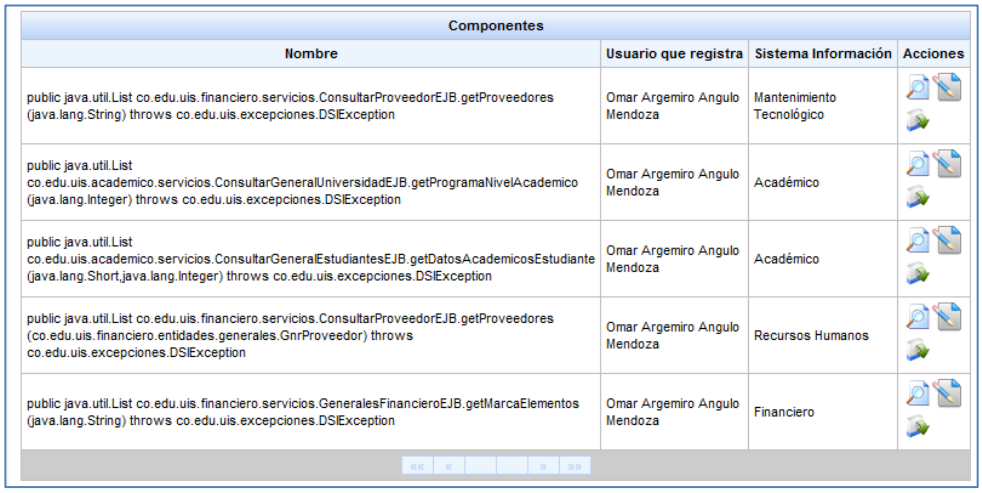

Figura 17. Interface de usuario – Resultado de la Búsqueda.

En la figura 17 se muestran los diferentes atributos de la ficha de especificación, entre los cuales se destacan, los datos básicos del componente, los parámetros de entrada, los parámetros de salida el registro de utilización del componente.

Se puede observar que el repositorio de componentes determina los atributos del objeto que el componente retorna, de esta forma muestra estos atributos, para que el usuario elija o marque los atributos que desea visualizar en la prueba del componente, si así lo desea.

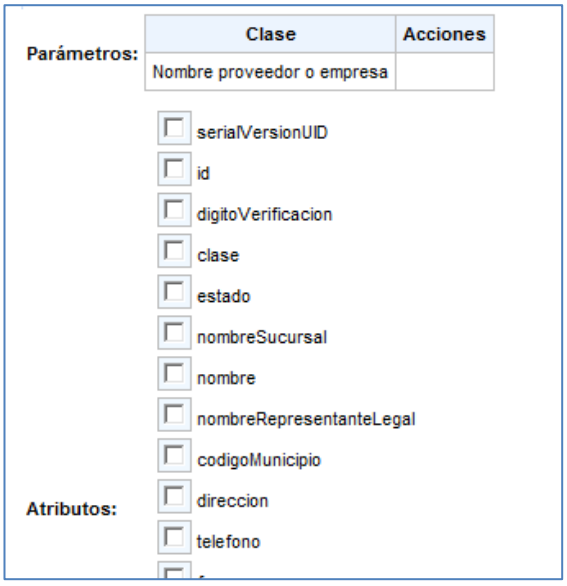

Figura 18. Interface de usuario – Parámetros entrada y salida.

En caso de que el usuario decida probar o ejecutar el componente para verificar sus datos de salida, o su funcionamiento, el repositorio muestra una ventana, donde el podrá observar los datos más básicos del componentes seleccionado y adicionalmente podrá digitar los parámetros de entrada del componente, debidamente nombrados, según el registro realizado. En la figura siguiente se muestra la ejecución del método getProveedores, y para él se distingue los

datos básicos y sus parámetros de entrada como es el nombre del proveedor o de la empresa proveedora.

Figura 19. Interface de usuario – Ejecución de componente.

También se puede ver, que al ejecutar los componentes, este retorna los datos consultados en la base de datos a través del componente, filtrando los atributos por medio de los elegidos por el usuario.

Otras de las acciones que se permiten ejecutar desde el repositorio de componentes es el registro de utilización de componentes. Luego de que el usuario verifico su funcionamiento, verifico que efectivamente, este componente le sirve para la implementación en su sistema, el usuario debe registrar su utilización. Para ello es repositorio exige el registro del sistema en el cual se utilizara, y una descripción donde el deberá especificar con más detalles en el módulo que lo utilizo. Esta interfaz se muestra en la siguiente figura.

Figura 20. Interface de usuario – Registro Utilización de componente.

- **Pruebas de interfaz de usuario.**

  Las pruebas a realizar para las interfaces de usuario se enfocan principalmente en probar que la implementación del repositorio quedó bien realizada y cumple con los requisitos planteados. Las pruebas a realizar se definen el la siguiente tabla.

Tabla 4. Pruebas de Interfaz de usuario.

PRUEBAS DE LAS INTERFACES DE USUARIO		
Descripción Prueba	**Objetivo**	**Resultado Esperado**
Prueba funcional: Con esta prueba se pretende probar todas las funcionalidades implementadas en el repositorio. Se realiza	Verificar que las funcionalidades implementadas en el componente se ejecutan correctamente.	Cada funcionalidad se debe ejecutar sin generar ninguna excepción ni error y los resultados devueltos deben ser los correctos.

suministrando los datos necesarios para probar cada funcionalidad.		
Prueba de eventos: Con esta prueba se quiere detectar posibles fallas en la manipulación de eventos que se hacen en el repositorio. Se deben probar los eventos que puede manejar el repositorio. Se deben crear casos de prueba por evento donde se especifique una acción a realizar y un mensaje a mostrar.	Verificar que el repositorio responde de forma adecuada a los eventos que maneja.	Se ejecuta el evento correctamente y muestra el mensaje suministrado en el caso de prueba.
Prueba de utilización: Con esta prueba se debe garantizar que el repositorio se puede utilizar correctamente con varias instancias en la misma página.	Verificar que el repositorio se puede utilizar correctamente con varias instancias en la misma página.	Que cada instancia del repositorio funcione correctamente, asignando el componente seleccionado en lugar que le corresponde, sin interferir con las demás instancias.

Al registrar un componente en el repositorio de componentes, se debe probar su funcionalidad dentro del repositorio, esto con el fin de no permitir el registro de componentes no funcionales, para asegurar este procedimiento, se implementó dentro del proceso de registro, la ejecución del componente como paso de su almacenamiento.

Con la realización de las pruebas se garantiza que el repositorio se contenga u ofrezca las funcionalidades para las que fue concebido.

Dentro de las funcionalidades a probar se encuentran las de búsqueda de componentes, para ello se hace necesario registrar la información de la especificación del componente, este información se

encuentra en la ficha de especificación, de la cual se hablara a continuación.

2.5.2. FICHA ESPECIFICACION DEL COMPONENTE.

Un componente tiene que mostrar información suficiente para poder ser implementado y usado. La especificación del componente se centra en:

- Descripción del componente.
- La especificación de su interfaz: muestra todo lo que el cliente debe saber para su uso en tiempo de ejecución.
- La especificación de la implementación: Define la información necesaria del componente en tiempo de diseño, para ser apropiadamente implementado, distribuido o instalado.
- Especificación de las características funcionales del componente: Especificar de los casos de uso que maneja.
- Especificación de las características no funcionales, como son los recursos requeridos, el rendimiento, la fiabilidad, la concurrencia, etc.
- Especificación de las características más importantes del diseño del componente.
- Especificación de las pruebas realizadas al componente.

La especificación de los componentes software, desarrollados por la DSI, no se encuentra registrada en ningún repositorio de componentes, este procedimiento de especificación se basa en el registro de una información básica del componente en un formato impreso, este formato no se encuentra diligenciado en la actualidad para ningún componente de software de la DSI. Esto se debe a que

dicho procedimiento no contempla la existencia de un repositorio de componente que permita el registro y la consulta de su ficha de especificación, ficha que puede diluir o aclarar cualquier duda acerca del funcionamiento del componente, y así permitir de forma clara y precisa la identificación del componente requerido.

El procedimiento de especificación de componentes de la DSI, fue modificado, este procedimiento se encuentra ahora inmerso en el proceso de registro del componente en el repositorio, esta modificación permite estandarizar la ficha de especificación, llenar con carácter obligatorio la información requerida en dicha ficha, centralizar el sitio de almacenamiento de las fichas de especificación de componentes y facilita la consulta y visualización de las fichas de especificación de los componentes de software registrados en el repositorio de componentes de software reutilizables.

Para la implementación de este procedimiento de especificación de componentes, se hizo necesario modificar el procedimiento de desarrollo de sistemas de la DSI, ya que se adicionaron políticas nuevas en el proceso, políticas como la utilización obligatoria de solo componentes registrados en el repositorio, validación de integración de componentes registrados en el paso de verificación y revisión de estándares y adición de algunos pasos obligatorios en el proceso de desarrollo de componentes, pasos como la sobre-escritura del método toString() de cada entidad, esto con el fin de utilizarlo en la visualización de la descripción del componente en la búsqueda dentro del repositorio.

A continuación se muestra el procedimiento de especificación de componentes de la DSI.

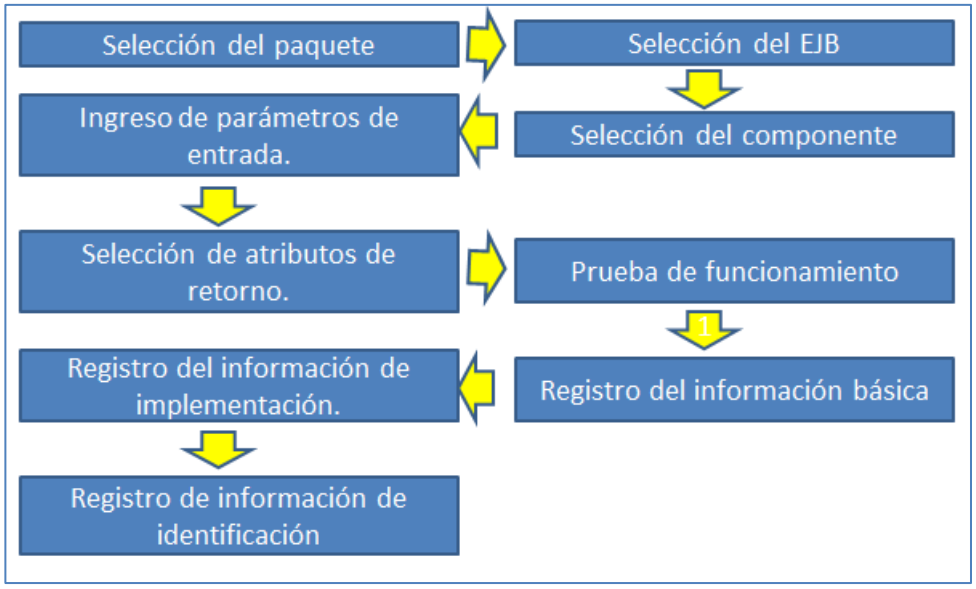

Figura 21. Procedimiento de Especificación de Componentes

La ficha de especificación de componentes de la División de Servicios de Información se revisó, de este proceso de revisiones, se hicieron algunas modificaciones a la ficha actual de especificaciones, entre las principales modificaciones se encuentran:

- La inclusión de un nuevo bloque de agrupación de atributos llamado especificación de identificación.
- La redistribución de algunos atributos.
- Eliminación del atributo de tecnología, ya que este no es relevante para la DSI, ya que los componentes registrados en el repositorio, serán solo componentes desarrollados por la DSI y bajo la misma tecnología.
- Aparición de un nuevo atributo, atributo que permite identificar el tipo de componente registrado (interfaz o servicio).
- Desaparición de atributos informativos, información que es importante para la mezcla de diferentes estructuras de componentes y lenguajes de programación, situación que no se presenta en la DSI.

- Inclusión de atributos básicos para las funciones definidas por la DSI, funciones que permitirán identificar el componente dentro del repositorio.

A continuación se muestra la nueva ficha de especificación de componentes implementada en la DSI.

FICHA DE ESPECIFICACIÓN DEL COMPONENTE					
NOMBRE:	Nombre del componente.	**FECHA:**	Fecha de creación	**VERSIÓN:**	1.0
DESCRIPCIÓN:	Descripción breve del componente, que hace, las funcionalidades que maneja, domino de aplicación al cual pertenece.				
ESPECIFICACIÓN DE IMPLEMENTACIÓN					
TIPO VISUALIZACION	Tecnología en la cual fue implementado el componente, junto con sus versiones.				
INTERFAZ:	Nombre de la interfaz que define los servicios del componente.				
SERVICIO:	Define o registra el paquete de ubicación del componente o servicio.				
PRECONDICIONES:	Definen los requisitos que se deben cumplir antes de entrar a utilizar el componente.				
POSTCONDICIONES:	Definen los requisitos que se deben cumplir después de utilizar el componente.				
ESPECIFICACIÓN NO FUNCIONAL					
GRADO DE REUTILIZACIÓN:	Es el puntaje obtenido por el componente en el análisis de reutilización realizado en el modelo de selección de componentes.				
DISTRIBUCIÓN:	Define la forma como se debe instalar y utilizar el componente en las plataformas soportadas.				
ESPECIFICACIÓN DE IDENTIFICACIÓN					
TABLAS:	Define las estructuras de datos que están involucradas en el proceso.				
ENTIDADES:	Define las entidades relacionadas directamente con la lógica del componente.				
PALABRAS CLAVE:	Define o relaciona las palabras que identificaran el componente con los criterios de búsqueda.				
SISTEMA:	Sistema bajo el cual el componente fue desarrollado, y por lo cual pertenece su lógica de negocio.				
FUNCIONALIDAD:	Define las funcionalidades realizadas por el componente (INSERT, UPDATE..).				
AUTOR:	Nombre de la persona que desarrollo de componente.				

2.5.3. DISTRIBUCION DEL COMPONENTE.

Para poder implementar el componente dentro de algún nuevo sistema donde sea requerido, no es necesario que el desarrollador obtenga diferentes archivos ni dll's del repositorio, esto se debe a que el repositorio de componentes se encuentra implementado sobre la misma plataforma de los sistemas de información y utiliza las misma librerías de desarrollo de los sistemas de información. Lo realmente importante para el desarrollador es conocer de la existencia de dicho componente, su paquete, su EJB y las funcionalidades que este le presta. Todo esto lo obtiene de repositorio de componentes, gracias a su ficha de especificación diligenciada al momento del registro. Si por algún motivo, el desarrollador no cuenta con el componente dentro de sus librerías locales de desarrollo, basta con actualizar dichas librerías desde el servidor de versiones, para ello cuenta con una cuenta de usuario que le permite realizar esta acción. En el repositorio de componentes, se puede obtener un documento PDF, el cual le permite obtener información acerca de cómo implementar cualquier componente en el sistema, ya sea de tipo interfaz o de servicio.

2.5.4. PRUEBAS

Además de las pruebas unitarias e independientes realizadas a los mapeos de las entidades, interfaces y demás, se deben realizar pruebas adicionales de las funcionalidades del repositorio ya ensamblado, funcional y puesto en marcha en el servidor de preproducción de la Universidad. Para ello se hace necesario, reemplazar las librerías utilizadas localmente para el desarrollo, por las librerías actuales y liberadas del servidor de versiones, esto con el fin de asegurar la funcionalidad correcta en el momento de subir el repositorio de componentes a los servidores de producción de la Institución.

Para realizar esta prueba de componentes, se llevaron a cabo varias actividades que además de realizar la prueba de la funcionalidad, permitiera cumplir con uno de los objetivos planteados. Este es el objetivo de registro de algunos componentes desarrollados para el sistema académico de pregrado UIS. Para determinar cuáles y cuantos servicios se registraran, se realizo un conteo detallados de los servicios, o métodos registrados en los paquetes o librerías

ofrecidas para el uso de los desarrolladores, en la creación de los sistemas de información institucionales. Para la prueba de la funcionalidad en el repositorio, se creó un paquete de servicios temporal llamado “academico-services”, este paquete, cuenta con seis servicios o componentes desarrollados para el sistema académico, el objetivo de este paquete radica en verificar la funcionalidad del repositorio y permitir observar las falencias en la implementación del componente, es decir fallas, que deben ser corregidas en el código del servicio, además se evidenciaron falencias en la estandarización y policitas de los componentes o mapeo, entre las que se destaca, la falta de redefinición de el método toString(), el cual permite mostrar el nombre de cada método o entidad de forma entendible y abreviada.

2.5.4.1. Prueba de registro de paquetes.

Para realizar el registro de cualquier componente, se debe primero realizar el registro del paquete donde este contenido, para ello se dispuso en el repositorio de componentes la opción de registro de paquetes. Antes de realizar el registro del paquete, este paquete debe estar ubicado en la carpeta Lib del servidor Jboss. Como prueba de esta acción, se realizara el registro del paquete *“académico-services”*, paquete implementado para tal fin.

Como lo muestra la figura 23, para el registro del paquete se deben diligenciar algunos atributos, entre ellos, el nombre del paquete, este nombre se digita tal cual aparece en las librerías del servidor, una descripción del paquete, un identificador de la conexión del entity manger utilizada por este paquete, esto con el fin de poder utilizar los permisos asignados al login de implementación de este component, sin exigir privilegios a todas las entidades para el login de funcionamiento del repositorio de componentes. Este nombre se encuentra en el archivo persistence.xml ubicado en el archivo.jar, dentro de la carpeta META-INF.

A continuación se muestra el contenido de este archivo.

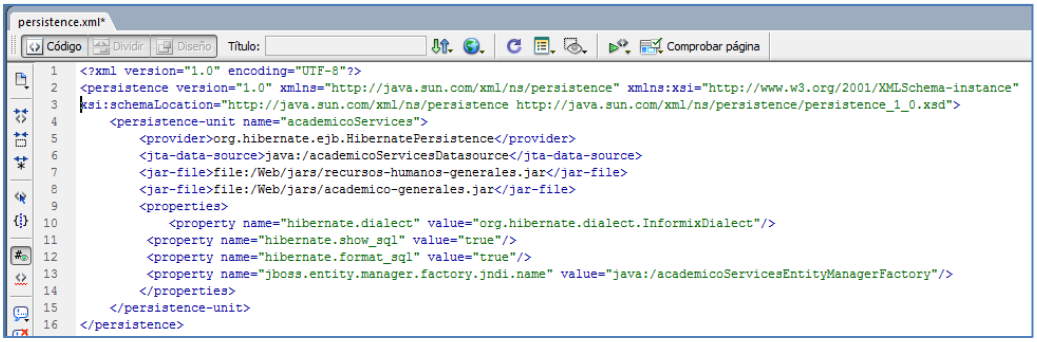

Figura 22. Contenido persistence.xml.

Para el registro del paquete, el sistema dispone de la siguiente forma de pantalla, la cual le permite el registro de los campos anteriormente descritos.

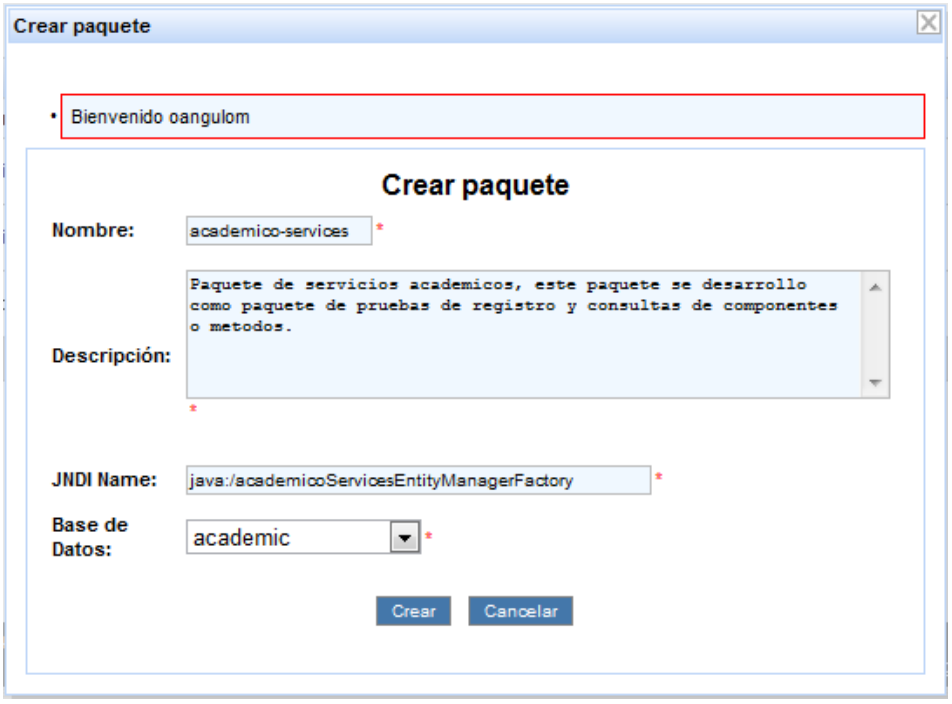

Figura 23. Registro paquete academico-services.

Después de este registro, por la opción de registro de componentes, se pueden registrar los métodos contenidos en este paquete.

2.5.4.2. Prueba de registro de componentes.

Para la prueba del registro de componentes, se hace necesario, registrar inicialmente el nuevo paquete con los componentes o servicios ofrecidos. Luego de ellos, se procede a registrar todos y cada uno de los componentes o servicios implementados dentro del paquete. Para es necesario, seguir los pasos del registro de componentes detallados en el capítulo correspondiente a registro de componentes.

Para evidenciar la prueba del registro correcto de los servicios getNotasEstudiante(2), getCreditosMatriculados, getProgramasAcademicos, getCondicionalidades y getEscuelasUniversidad, contenidos en el EJB denominado ConsultarGeneralAcademico, se pueden realizar consultas en el repositorio de componentes, realizando el proceso de búsqueda de componentes.

En la figura siguiente, se muestran los primeros pasos que deben realizarse en el momento de registrar un componente de tipo servicios, Este registro corresponde a la identificación del paquete y EJB que contiene los servicios o métodos a registrar.

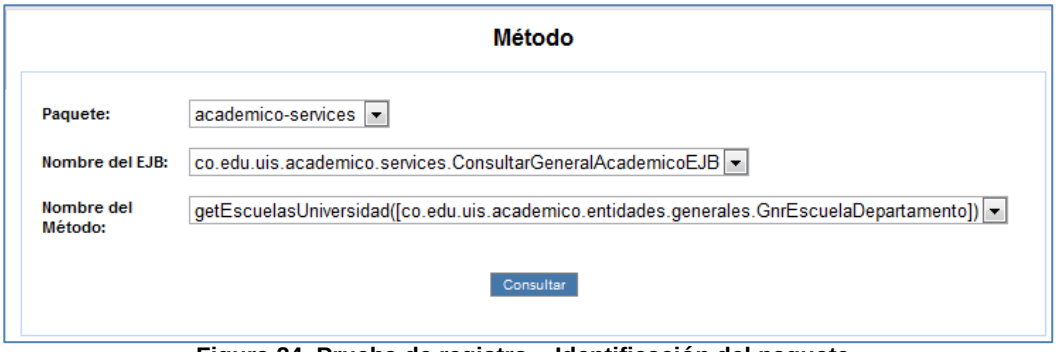

Figura 24. Prueba de registro – Identificación del paquete.

Luego de identificar plenamente el componente o servicio que desea registrarse, el repositorio de componentes procede a realizar una identificación plena del componente, es decir identificar sus atributos de entrada y salida, su interfaz o firma y nombre, además permite el registro de algunos datos que el sistema requiere para realizar su ejecución, esto con el fin de validar su funcionalidad antes del permitir el registro.

En la figura 23, se muestran algunos de los atributos que el repositorio exige en el momento de la ejecución de prueba, para el componente denominado getEscuelasUniversidad.

Figura 25. Prueba de registro – Validación de funcionalidad.

En la figura también se puede observar que el repositorio identifica los objetos de entrada y salida, para el objeto de salida, el repositorio muestra los atributos de dicho objeto con el fin de que el desarrollador elija los atributos que desea observar, y de esta forma no mostrar la totalidad de atributos del objeto de salida. Para el caso de los objetos de entrada (parámetros), el repositorio permite, por medio del botón *"ir"*, inicializar un objeto del tipo requerido, para pasarlo al componente.

En la figura a continuación, se muestra la pantalla que el repositorio despliega para la inicialización del objeto de entrada, se puede apreciar que el sistema reconoce los atributos necesarios para la inicialización o identificación de un objeto del tipo requerido y los exige en el momento de inicialización.

Figura 26. Prueba de registro – Iniciar parámetro de entrada.

Luego de ejecutar el componente con el fin de validar su funcionamiento, el repositorio permite diligenciar la ficha de especificación del componente, en la figura 25 se muestran algunos datos de la ficha de especificación del componente, luego de su respectiva validación.

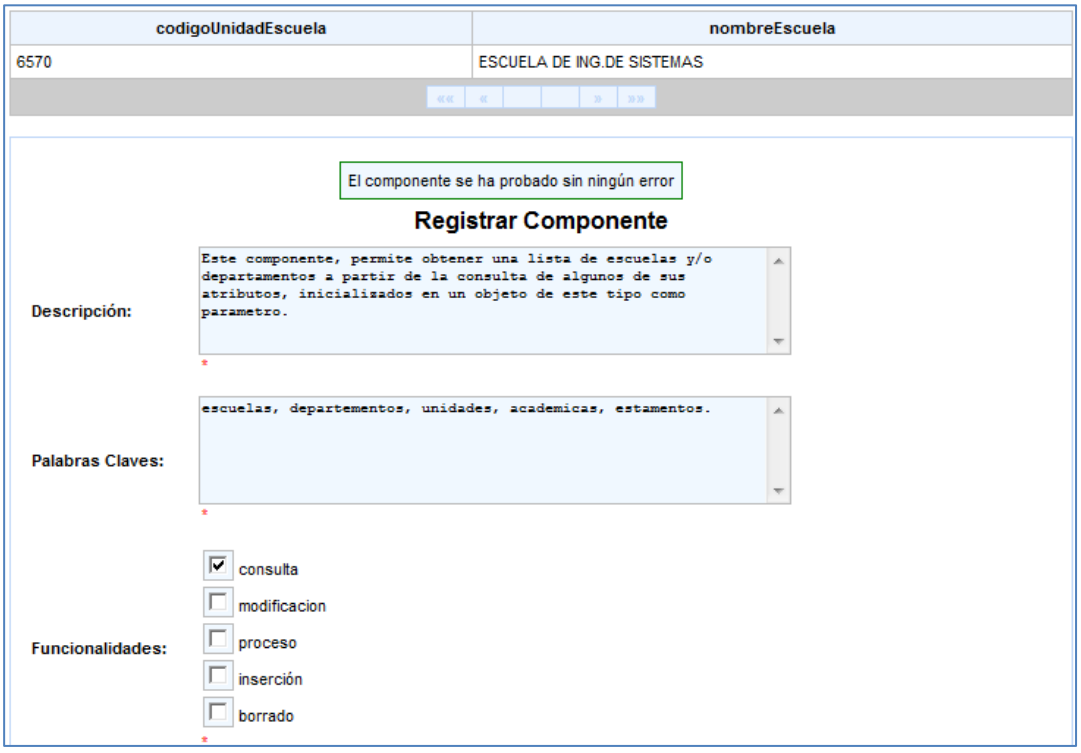

Figura 27. Prueba de registro – Especificación - 1.

El repositorio de componentes, luego de identificar los parámetros de entrada del componente que se va a registrar, los muestra, esto para exigir una descripción

clara del atributo, esta se mostrara al desarrollador en el momento de la consulta y no se mostrara el nombre del atributo en la tabla o entidad como ocurre en el momento del registro. Esta observación se puede apreciar en la figura 26, junto con otra información adicional de la ficha, información como la entidades o tablas con las cuales el componente está relacionado, para el caso de prueba, se puede ver que el repositorio identifica de manera automática la entidad GnrEscuelaDepartamento o Tabla escuelas_deptos, esto puesto que el tipo de datos que retorna es de este tipo.

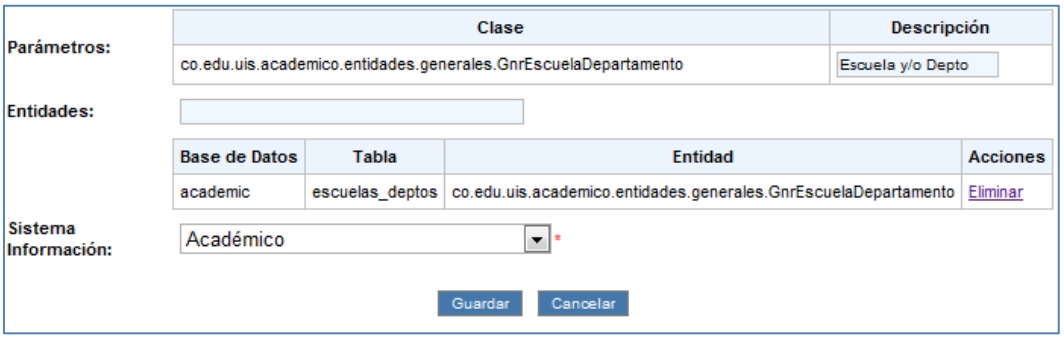

Figura 28. Prueba de registro- Especificación - 2.

Luego de realizar correctamente los pasos sugeridos por el repositorio, el sistema muestra una pantalla avisando del registro exitoso del componente, como se muestra en la figura siguiente.

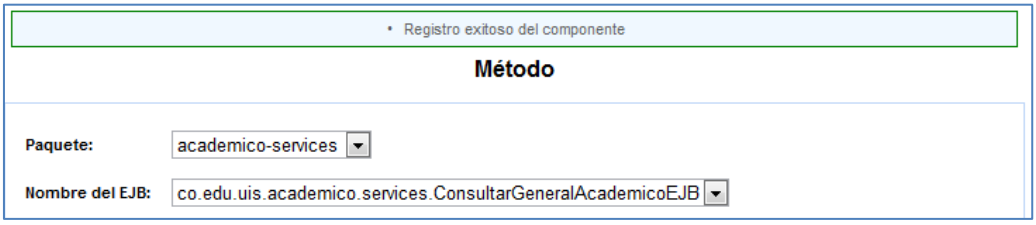

Figura 29. Prueba de registro - Exitoso.

Después de este proceso los desarrolladores pueden realizar los respectivos procesos de consulta de los cuales se detallara su respectiva prueba a continuación.

2.5.4.3. Prueba de búsqueda o consulta de componentes registrados.

Para verificar las funcionalidades de consulta, se pueden utilizar los componentes registrados en el paso anterior, para el caso de los componentes registrados, a todos se les coloco funcionalidad de consulta, pero diferentes clases de palabras clave de búsqueda.

A continuación se muestra algunos de los criterios de consulta en el repositorio, y sus resultados.

Figura 30. Prueba de consulta de componente.

En la figura anterior se muestra los criterios de consulta y resultados de esta para dichos criterios de búsqueda.

Se puede ver que para este caso en particular, el retorno de la consulta fue el componente de búsqueda de condicionalidades, ya que se coloco un criterio único para este componente. Para el siguiente caso, se tomo como base la misma consulta.

Al elegir este componente, se puede revisar los datos de la especificación de componentes, y si se desea, se puede ejecutar dicho componentes, esto con el objetivo de ver la funcionalidad del componentes y los posibles atributos que puedo obtener de el.

2.5.4.4. Prueba de selección y ejecución de componentes dentro del repositorio.

Para probar la ejecución de un componente, es necesario haber realizado un proceso de consulta exitoso, para esta prueba se pueden utilizar también los componentes registrados para la realización de pruebas de registro. Después de haber realizado la consulta, se debe elegir un componente o método a ejecutar, antes debe visualizarse la ficha de especificación, esta muestra datos acerca de su información básica, información de los servicios como parámetros de entrada, objetos de salida, y dentro de estos objetos, cuales atributos se pueden obtener. Esta información puede observarse en la siguiente figura, Al final de esta selección, se encuentra un botón que permite la ejecución del componente.

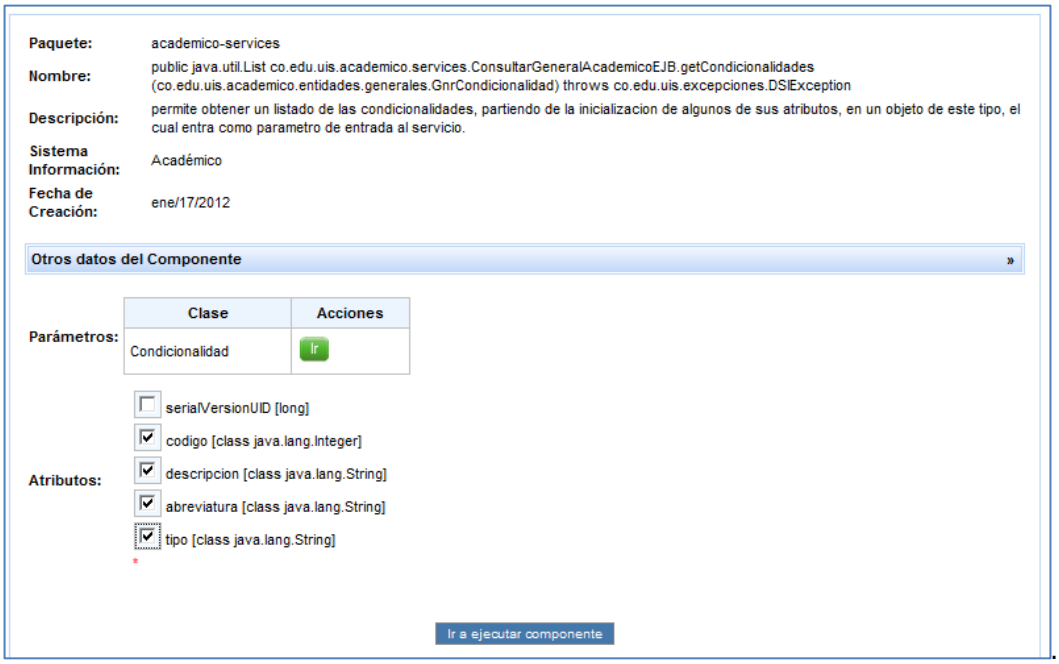

Figura 31. Prueba de ejecución de componente-1.

A continuación, se muestra en la figura, la pantalla mostrada al ejecutar un componente, se puede ver los parámetros que el componente requiere para su ejecución. Y la descripción que se coloco para el atributo de entrada en el momento del registro.

Figura 32. Prueba de ejecución de componente-2.

2.5.4.5. Prueba del registro de utilización del componente.

Para la prueba de esta funcionalidad, se debe hacer click sobre el icono de registro de uso en la consulta de componentes, los atributos que se exigen son el sistema donde se utilizara este componente, y alguna descripción más detallada, en la figura a continuación, se muestra el resultado de haber realizado un registro de uso.

Figura 33. Prueba de registro de uso de componente.

2.6. LECCIONES APRENDIDAS

2.6.1. APRENDIDO

Durante el desarrollo del trabajo de investigación se aprendió acerca del modelo de desarrollo de componentes, sus beneficios y sus principales características. Además de cómo hacer una revisión de un proceso implantado en una casa de software, entre lo que se destaca la identificación de falencias o necesidades del proceso. De esta forma, proponer posibles soluciones e identificar la más adecuada para solucionar un problema identificado.

También se aprendió acerca del proceso de desarrollo por componentes, proceso en el cual se deben llevar a cabo varias actividades, actividades de adquisición de requerimientos, definición y construcción de los casos de uso, como documentación, creación de prototipos no funcionales como método de socialización con los clientes de los casos de uso a implementar, identificación de componentes. Y finalmente el desarrollo de un repositorio de componentes, parte en la cual se obtuvo valiosa experiencia, ya que el proceso de adaptación o implantación de un sistema a un caso particular, conlleva a realizar una serie de adaptaciones, adiciones y eliminaciones de algunas características del sistema original, entre ellas se pueden recordar, la modificación de la ficha de especificación, ficha a la cual se adicionaron, modificaron y eliminaron diversos atributos que de una u otra forma no contribuían al proceso de desarrollo propio de la Universidad. Y cambios en el modelo de desarrollo, como en el proceso de especificación de componentes.

2.6.2. MEJORAS FUTURAS.

Entre las posibles mejoras al proceso de desarrollo de la DSI se identificaron las siguientes:

- Se recomienda la utilización de componentes que se encuentren debidamente registrados en el repositorio, esto como control de componentes desarrollados, centralización de los mismos y obligatoriedad del registro de la ficha de especificación.
- También se debería adoptar como política de desarrollo, la creación de los componentes o métodos solo por la persona o el área con el conocimiento de la lógica de negocio, ya que de la forma actual se está haciendo por una

persona sin el conocimiento suficiente de la lógica, lo que conlleva a que estos no se encuentren desarrollados de la mejor manera, o tengan algunas falencias de optimización por el no conocimiento de la estructura de datos.

- Realizar una depuración actual de los componentes desarrollados, esto, para determinar que componentes se encuentran actualmente duplicados, o contienen funcionalidades similares y podrían ser unificados en un solo componente.
- Definición de políticas que permitan controlar los cambios realizados en los componentes y que afecten el funcionamiento de los sistemas que actualmente los implementan, esta política debería también adaptarse a los cambios en las estructuras de datos y entidades de uso general.
- Estudio de porcentaje o grado de reutilización de los componentes desarrollados vs cantidad de código implementado, esto como practica de mejoramiento continuo y seguimiento al proceso de desarrollo de la División de Servicios de Información.

3. CRONOGRAMA

Actividad	Periodo de Tiempo (Meses)											
	1	2	3	4	5	6	7	8	9	10	11	12
FASE 1: FUNDAMENTACIÓN TEÓRICA												
Revisión y análisis bibliográfico												
Revisión y análisis de artículos especializados.												
Revisión y estudio de repositorios existentes												
Elaboración del estado del arte y marco conceptual.												
FASE 2: DIAGNÓSTICO DEL PROBLEMA DE INVESTIGACIÓN												
Comunicación con desarrolladores de software para hacer un análisis de los problemas del desarrollo de software existentes y determinar posibles soluciones												
Identificación de los problemas del modelo de desarrollo de software.												
Formulación del problema de Investigación y de las preguntas a resolver												
FASE 3: ESTUDIO DEL PROCESO DE DESARROLLO DE COMPONENTES IMPLEMENTADO EN LA DSI.												
Modelo de componentes EJB V. 3.0.												
Estudio de las alternativas de arquitectura												
especificación de componentes												
Estudiar y modificar el procedimiento de desarrollo												
Definir criterios de búsqueda y selección de componentes												
Determinar las alternativas de distribución												
FASE 4: DISEÑO E IMPLEMENTACIÓN DE UN REPOSITORIO DE COMPONENTES PARA LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN												
Análisis y diseño del repositorio de												

componentes												
Definición de la arquitectura del repositorio.												
Implementación del repositorio												
FASE 5: REGISTRO DE COMPONENTES												
Identificación de los componentes implementados por el grupo de desarrollo académico para determinar la cantidad de componentes a registrar.												
Elección de componentes a almacenar.												
Especificación de los componentes seleccionados.												
Pruebas de funcionalidad del repositorio.												
FASE 6: ELABORACIÓN DEL INFORME FINAL												
Recopilación de los informes preliminares de las etapas anteriores												
Elaboración de la propuesta final.												
FASE 7: DIVULGACIÓN DE RESULTADOS												
Creación del artículo.												
Publicación del artículo.												

4. PRESUPUESTO

4.1. PRESUPUESTO GLOBAL DE LA PROPUESTA

Rubro	Valor
Gastos de personal	$ 55.780.000
Gastos en equipos	$ 2.800.000
Gastos en materiales	$ 400.000
Software[19]	$ 0
TOTAL	**$ 58.980.000**

4.2. GASTOS DE PERSONAL

Investigador	Función	Dedicación	Total
Msc. Fernando Rojas Morales	Director	3 hr/semana	$ 24.960.000
Msc. Fredy Humberto Vera.	Codirector	1 hr/semana	$ 8.320.000
Ing. Robinson Delgado Rojas.	Autor del proyecto	6 hr/día	$ 22.500.000
TOTAL			**$ 55.780.000**

4.3. GASTOS EN EQUIPOS

Recurso	Justificación	Ubicación	Total
Computador	Realización de los documentos, informes, implementación del repositorio de componentes.	DSI	$ 2.500.000
Impresora Láser(uso y tinta)	Para imprimir documentos e informes	DSI	$ 350.000
TOTAL			**$ 2.800.000**

4.4. GASTOS EN MATERIALES

Material	Total
Papelería	$ 50.000

[19] Se utilizaran herramientas libres para el desarrollo de la investigación y las herramientas que tiene licenciadas la División de Servicios de Información.

Adquisición de libros y artículos técnicos	$ 200.000
Fotocopias	$ 50.000
Cds y Dvds.	$ 100.000
TOTAL	**$ 400.000**

5. BIBLIOGRAFIA

ARIZA ROJAS Maribel, MOLINA GARCIA, Juan Carlos. Introducción y principios básicos del desarrollo de software basado en componentes. Universidad Javeriana.

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

Brown, A.W., y K.C. Wallnau, «Engineering of Component Based Systems», Component-Based Software Engineering, IEEE Computer Society Press, 1996,

CHO, Hyung y McGREGOR, John D. Artículo: Component Specification for Enterprise Software Development on Web Services Environment. En: 2005 Third ACIS Int'l Conference on Software Engineering Research, Management and Applications – 2005 IEEE.

CRNKOVIC Ivica, CHAUDRON Michel, LARSSON Stig. Component-based Development Process and Component Lifecycle. Mälardalen University, Department of Computer Science and Electronics, Sweden.

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. Los Castros Santander – España.

EUN CHA, Jung y JUNG YANG, Young. Design and implementation of component repository. Software Engineering Department, Electronic and Telecomunications Research Institute. KOREA.

GUO Jiang, Research Associate US National Research Council,USA. And Luqi, Department of Computer Science US Naval Postgraduate School Monterey,USA.

GUO, Jiang and luqi. survey of software Reuse Repositories Department of Computer Science.

HERNANDEZ DIEZ, Carmen y LAGUNA SERRANO Miguel. Biblioteca de Reutilizacion GIRO.

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

JAMES Petro and Michael E. Fotta, Model-Based Reuse Repositories - Concepts and Experience Electronic Warfare Associates.

JIHYUN Lee, Jinsam Kim, and Gyu-Sang Shin. Facilitating Reuse of Software Components using Repository Technology. Software Engineering Research Team, Embedded S/W Technology Center.
Electronics and Telecommunications Research Institute, Republic of Korea.

MONTILVA C, Jonas, ARAPE Nelson, COLMENARES Juan. Desarrollo de software basado en componentes, Universidad de los Andes, Venezuela.

M.R.J. Qureshi, S.A. Hussain. A reusable software component-based development process model, Department of Computer Science, COMSATS Institute of Information Technology, Lahore, Pakistan. elsevier.com/locate/advengsoft

Object-Oriented Analysis and Design Using UML. Copyright 2003 Sun Microsystems

PADMAL Vitharana, Fatemeh ªMariamº Zahedi, and Hemant Jain, Knowledge-Based Repository Scheme for Storing and Retrieving Business Components: A Theoretical Design and an Empirical Analysis. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 29, NO. 7, JULY 2003.

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

PRIETO-Díaz, R., «Domain Analysis for Reusability », *Proc. COMPSAC* '87, Tokyo, Octubre de 1987, paginas 23 – 29.

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