Universidad de las Ciencias Informáticas

Facultad # 3

Título: **Propuesta de implantación de un repositorio para la gestión de componentes reutilizables.**

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: **Samuel Zayas Cuza.**

Tutor: **Ing. Marbys Marante Valdivia.**

La Habana, Junio del 2008

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad # 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Samuel Zayas Cuza.	**Marbys Marante Valdivia.**
Autor.	Tutor.

AGRADECIMIENTOS

Quiero agradecer a todos los que tuvieron que ver de alguna forma con el éxito de mi carrera y de este trabajo. A mis primeros compañeros de aula, a los que les subsiguieron y a aquellos con los que terminé. A mi tutor Marbys, a mis familiares y amigos que me brindaron su apoyo, a Johnny, Omelito, Yandry, Juli, y Shalymar que a pesar de nuestras diferencias siempre hemos sido como hermanos en todo momento. A la Revolución y al Comandante por darme la posibilidad de haberme convertido en un intelectual de esta sociedad. A mi madre por haberme traído al mundo y ser mi punto de apoyo en cualquier circunstancia, pues has sabido también, ser mí amiga. Quiero darte las gracias por ser mi madre. A mi tía Dunia, poseedora de ese don para proteger y cuidar de tus sobrinos, a tí tengo mucho que agradecerte y espero en algún momento poderte retribuir todo ese cariño y amor que nos has dado, por eso quiero agradecerte en nombre de todos nosotros, de tu hijo mayor, gracias.

DEDICATORIA

Dedico este trabajo en primer lugar al hombre que fue mi padre y mi madre durante muchos años, a Pipo mi abuelo y aunque no cumplí mi promesa de ser abogado, soy ingeniero y donde quiera que estés se que te sentirás orgulloso, a quienes me dieron vida, mis padres que han sabido guiar mis pasos para convertirme en lo que soy hoy, ustedes tres son los principales autores de este significativo logro en mi vida. A mi hermano Carlos que sueña con seguir mis pasos, a mi hermano Alieski que nunca me ha fallado y que desde que nos conocemos lo compartimos todo, a mi hermanita Reisel que aunque todavía no sé escribir su nombre ella sabe que la quiero. En general a toda mi familia que me ha brindado su apoyo en los diferentes momentos de mi vida, a la prima que más quiero y me quiere, Haylín y a Maria Alejandra, a mis abuelas Mima y Doris, a mi abuelo Erasmo y mi hermanita Ena, a Bolita, Anaydi, Tania, Roberto y Lázaro. Y a alguien que no puede faltar, y no por ser la última eres la menos importante, al contrario eres muy especial, mi segunda madre, mi tía Dunia que me ha dado su cariño y amor de madre durante toda mi vida y me ha apoyado en todo momento durante estos cinco años.

RESUMEN

En la actualidad la reutilización de activos en la industria del software, ha devenido como uno de los métodos más usados para lograr mayor eficiencia y rapidez en el cumplimiento de los proyectos, a un menor costo y libre de la mayor cantidad de errores posibles, basados no sólo en el producto sino el proceso para construirlo contribuyendo además a mejorar la calidad de los mismos en cuanto a organización y flexibilidad. Los repositorios de componentes de software reutilizables son aquellos donde se guardan estos componentes y su objetivo principal es asegurar la disponibilidad de activos, para apoyar el desarrollo de un producto de software proporcionando constantemente información de cada activo presente en el repositorio.

A pesar de que la reutilización de componentes se ha manifestado de forma aislada, la Universidad de las Ciencias Informáticas, no ha estimulado de forma ordenada la reutilización así como no tiene definida una política al respecto, además que no se cuenta con una repositorio de componentes de software reutilizables con las características adecuadas al proceso productivo de la UCI, que posibilite la utilización de activos elaborados con anterioridad, en un nuevo proyecto de la Universidad para satisfacer una necesidad determinada. Además de que la cultura con relación a la reutilización es muy pobre.

Como propósito en este trabajo se encuentra la selección correcta de estas técnicas para el diseño de un repositorio que posibilite la reutilización entre las diferentes entidades de una factoría de software y que se adapte a las características del proceso productivo y hacerlo extensivo. Para esto se realiza la propuesta de la implantación de un repositorio de componentes de software, a partir de la experiencia adquirida en la Facultad # 3.

PALABRAS CLAVE:

Reutilización, repositorio, componentes, software reutilizable.

INDICE

INDICE DE FIGURAS.

INDICE DE TABLAS.

INTRODUCCIÓN

Desde hace algún tiempo la producción de software ha venido creciendo, debido a la gran demanda existente y el desarrollo de los sistemas computacionales, esto ha traído como consecuencia una mayor complejidad a la hora de construir aplicaciones, además de que los clientes han aumentado su nivel de exigencia en cuanto a calidad y tiempo.

Muchos proyectos hoy en día implementan funcionalidades similares y utilizan componentes de software similares, sin embargo son desarrollados desde cero una y otra vez. Esto conllevaba a repetir esfuerzos empleando tiempo en lo mismo que ya se había desarrollado con anterioridad, e incluso, en ocasiones pudieran repetirse los mismos errores. Desde tiempos muy remotos la necesidad ha obligado al hombre a reutilizar el conocimiento existente y en efecto, al reutilizar segmentos de experiencias, ideas y artefactos, no solo se asegura no cometer las fallas del pasado, sino que se logra construir cosas más grandes, maravillosas y de mayor calidad.

Aplicar estas técnicas y métodos en el mundo del software es un campo al que se le dedica cada vez mayor interés. No sólo se ha venido estableciendo una colección de métodos, técnicas, esquemas y expresiones para describir la arquitectura de un sistema de software, sino que se tiende al desarrollo de entornos que permitan la reutilización efectiva, tanto de los componentes de un sistema, como de la propia arquitectura o configuración del mismo.

La UCI se sitúa en la avanzada en todo lo que se refiere a informática por el por ciento que representa dentro de la Industria Cubana del Software, por la cantidad tecnología de la cual está dotada y por los importantes logros que ha obtenido a través de sus cinco años de creación. Por eso para lograr convertirse en una entidad líder en este campo debe seguir llevando al escenario de la Universidad las prácticas más utilizadas en el mundo de la informática como se ha hecho hasta el momento.

Situación Problémica.

A pesar de la reutilización de componentes se ha manifestado de forma aislada en algunas áreas, no se ha estimulado de forma ordenada. No se ha establecido ninguna política al respecto y no se cuenta con un repositorio de componentes de software reutilizables adecuado a las características del proceso

productivo de la universidad, que permita la utilización de activos elaborados en un proyecto determinado y que sea necesario para el desarrollo de otro u otros. Por esta razón, tampoco existe una cultura de reutilizar componentes de software ya existentes, sino que se comienzan a desarrollar desde cero cada vez.

Por lo que se recomienda hacer una selección de elementos de configuración tales como: documentos, artefactos, segmentos de códigos, librerías, y otros tipos de componentes que son generados a lo largo del ciclo de vida de un software y que puedan ser utilizados nuevamente para agilizar de esta forma el proceso de desarrollo y lograr una mayor eficiencia.

Teniendo en cuenta todo lo descrito anteriormente se puede identificar como **Problema Científico** de la investigación la siguiente interrogante:

¿Cómo combatir la ineficiencia en la gestión de componentes, contribuyendo a solucionar problemas de reutilización en un proyecto productivo?

El **Objeto de Estudio** de este trabajo es el sistema de soporte al proceso de desarrollo de software, en un proyecto productivo.

Abordándose el **Campo de Acción** de Los Repositorios de Componentes Reutilizables.

Planteándose como **Objetivo General** para darle solución a este problema: Desarrollar una propuesta de implantación de un repositorio de componentes reutilizables en un polo productivo.

Se genera como **Hipótesis** que:

Si se logra establecer una biblioteca de componentes con contenido de valor altamente reutilizable se contribuirá a mejorar la calidad y eficiencia de los proyectos productivos.

Entre los principales **Aportes Teórico-Práctico** esperados en esta investigación están:

La propuesta de implantación repositorio es una forma de guiar la atención de los desarrolladores entorno a la reutilización. El repositorio de componentes que se propone como solución a esta problemática posee un alto valor práctico que se manifiesta en su aplicación en los polos de producción posibilitando la reutilización de componentes entre proyectos y es de gran utilidad pues posibilita la gestión de componentes apoyado en técnicas de clasificación y/o recuperación y una mayor productividad en el proceso de desarrollo.

Objetivos Específicos

- Diagnosticar que existen problemas en la producción y falta de organización que impiden la reutilización.
- Seleccionar y describir las tendencias que existe a nivel mundial en cuanto a la reutilización y al Desarrollo de Software Basado en Componentes.
- Desarrollar la propuesta de implantación del Repositorio de Componentes Reutilizables como herramienta de soporte al proceso productivo.
- Demostrar los beneficios del desarrollo y la validación de resultados de la propuesta.

Para darle cumplimiento a los objetivos anteriormente planteados se hace necesario llevar a cabo las siguientes **tareas**:

- Sistematización del estudio del estado del arte.
- Realización de un estudio acerca de los principales métodos, técnicas y herramientas utilizadas en la reutilización.
- Definición de las herramientas que van a ser utilizadas durante el proceso de desarrollo de la aplicación.
- Realización de un estudio sobre las políticas de reutilización de componentes y proponer nuevas políticas.
- Definir las principales funcionalidades del sistema propuesto.
- Ejecución del desarrollo de la propuesta de acuerdo con las características definidas con anterioridad.
- Proposición de la utilización del Repositorio de Componentes Reutilizables.
- Validación mediante la comprobación el cumplimiento de las funcionalidades del repositorio.

Métodos de trabajo científico:

- Métodos Teóricos:

Histórico: Brinda la posibilidad de estudiar todo el proceso y obtener un conocimiento histórico de su desarrollo y comportamiento durante años, así como en los últimos tiempos, es decir cómo está vinculado al conocimiento de las distintas etapas de los objetos en su sucesión cronológica, posibilita conocer los antecedentes y tendencias acerca del tema en el entorno internacional, cubano, empresarial y universitario.

- Métodos Lógicos:

Sistémico: Modela el objeto a través de la determinación de sus componentes y las relaciones que existen entre los mismos, las cuales determinan la estructura del objeto así como su dinámica.

El presente trabajo esta constituido por tres Capítulos que conforman el desarrollo de la investigación. En el primer Capítulo se realiza un análisis de las principales tendencias de la reutilización de componentes de software y su importancia. También se aborda el caso de la UCI y algunas características de herramientas de gestión de proyectos más utilizadas a nivel mundial.

El segundo Capítulo describe las características del Repositorio de Componentes Reutilizables propuesto, sus objetivos, el proceso de desarrollo de la propuesta y se definen políticas de reutilización para apoyar la puesta en ejecución del mismo.

En el tercer Capítulo se valida el desarrollo de la propuesta mediante las normas establecidas en el Modelo de Estimación del Esfuerzo Basado en Casos de Uso, calculándose el esfuerzo total del software, también se exponen los beneficios tangibles e intangibles a través del análisis de los resultados obtenidos.

Capítulo 1: Principales vertientes de reutilización de componentes de software en el mundo.

1.1. __ Introducción.

Este capítulo está enfocado a sentar las bases en lo que se refiere a las tendencias existentes hacia la reutilización, teniendo en cuenta una pequeña reseña de lo que ha sido esta, desde sus inicios, hasta los momentos actuales. Se realiza una breve panorámica de la reutilización de componentes de software y los repositorios que contienen estos elementos en el mundo. Además se mencionan las principales características y ventajas de las herramientas más utilizadas en la gestión de proyectos. Se analizan diversas metodologías que existen para la creación de aplicaciones web a nivel mundial que servirán de guía a este trabajo para alcanzar el objetivo propuesto, así como las herramientas que se utilizaran para el desarrollo de la propuesta.

1.2. __ Surgimiento de la reutilización.

La reutilización de software aparece en el mundo de la informática como una alternativa para desarrollar aplicaciones y sistemas software de una manera más eficiente, productiva y rápida en aras de combatir la mencionada “crisis del software”. El autor de la primera propuesta acerca de que el software fuera ensamblado a partir de componentes previamente desarrollados fue Douglas McIlory, en aquel momento profesor adjunto del Computer Science en el Dartmouth College, New Hampshire, Estados Unidos en el año 1968, en la primera conferencia sobre desarrollo de software patrocinada por el Comité de Ciencia de la OTAN y celebrada en Garmisch, Alemania. A partir de esta propuesta, comienzan a surgir soluciones de reutilización para tareas específicas en diferentes áreas de la computación. A pesar de que el concepto de reutilización de software como tal no estaba aún definido dentro del mencionado proceso, esta experiencia fue el punto de partida para las futuras factorías de software, especialmente en Japón, donde el proceso de reutilización fue definido. La idea es reutilizar elementos y componentes de software en lugar de tener que desarrollarlos desde el principio, persigue como principal objetivo reutilizar lo existente sin tener que volverá rediseñarlo desde el principio.

1.3. __ Principales conceptos relacionados con la reutilización.

Existen varios aspectos relacionados con términos tales como: reutilizar, componentes, reutilización de software, bibliotecas y repositorios, que deben definirse sus conceptos, Para profundizar en lo referente a la reutilización.

¿A que se le llama reutilizar?

"Es utilizar nuevamente algo, con la misma funcionalidad o con otra siempre que las pueda desempeñar." (1)

¿Qué es un componente?

"Unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio." (2)

Conceptos de reutilización de software:

"Es el proceso de creación de sistemas de software a partir de un software existente, en lugar de tener que rediseñarlo desde el principio." (3)

Otros conceptos expuestos son:

"La reutilización es un enfoque de desarrollo de software que trata de maximizar el uso recurrente de componentes de software existentes." (4)

¿Qué es una biblioteca virtual?

"Es una biblioteca en que una proporción significante de los recursos de información se encuentran disponibles en el formato digital (pdf, doc, etc.), accesible por medio de las computadoras. El volumen digital puede sostenerse localmente o puede accederse remotamente vía las redes de la computadora. Una biblioteca digital es un sistema de tratamiento técnico, acceso y transferencia de información digital, estructurado alrededor del ciclo de vida de una colección de documentos digitales, sobre los cuales se ofrecen servicios interactivos de valor añadido para el usuario final." (5)

"Colección o fondo de documentos digitalizados puestos a disposición de los usuarios a través de un sitio Internet." (6)

¿Qué es entonces un repositorio?

"Es un lugar donde se guarda algo." (1)

Basado en lo expresado anteriormente la reutilización puede definirse como un procedimiento que produce o ayuda a producir un sistema mediante el nuevo uso de algún elemento procedente de un esfuerzo de desarrollo anteriormente por tanto un repositorio de componentes de software reutilizables es donde se guardan o conservan todo tipos de estos componentes ya sean documentos, ideas, pedazos de código, librerías de software y todos aquellos componentes que participan en el desarrollo de un software, que puedan ser reutilizados, transformados o consultados.

El proceso de reutilización se fundamenta en la recuperación de activos en un repositorio y su posterior adaptación para el proyecto donde se desean aplicar. Por lo que se debe partir de un repositorio que permita identificar sus componentes, de cara a poder evaluarlos y seleccionar el candidato ideal a formar parte del nuevo proyecto.

1.3.1. __ ¿Hasta que nivel llega la reutilización?

El término reutilización no es un término novedoso hace ya algún tiempo que se viene utilizando y la mayoría de las factorías de software están de acuerdo con él, sin embargo no se explota con frecuencia ni en la medida en que nos puede proporcionar comodidades.

La reutilización de código es la forma más común y extendida de la misma, se desarrolla durante la fase de implementación, donde pueden ser objeto de la reutilización el código fuente, el código objeto así como las bibliotecas estándares entre otras, este proceso realiza mediante editores, inclusión de ficheros, mecanismos de herencia en entornos de programación orientada a objetos y llamadas a las rutinas de una biblioteca. En este nivel está limitada por la dependencia del lenguaje, del sistema operativo y de la aplicación del código. La manera más tradicional de reutilización de código es la que se basa en bibliotecas de funciones, la aplicación llama a las funciones de la biblioteca a través de una capa de interfaz para conseguir los servicios que necesita haciendo alusión a la arquitectura en tres capas.

La reutilización juega un papel clave en varios temas como son: productividad, capacidad de mantenimiento, portabilidad y calidad. Por ello, la reutilización debe aplicarse a cada etapa del ciclo de

vida (levantamiento de requisitos, análisis, diseño, construcción, pruebas y mantenimiento). De lo contrario, no se recogerán todos sus beneficios potenciales.

El aumento de la potencia de reutilización ocurre por la elevación del nivel de abstracción. La abstracción constituye uno de los elementos fundamentales en la reutilización. David Parnas afirma que el desarrollo y catalogación de abstracciones incrementa la capacidad de reutilización del software desarrollado, ya que los desarrollos de una abstracción pueden ser reutilizados para cualquier modelo válido de la abstracción. La reutilización, por lo tanto, tiene que enfocarse hacia la reutilización de requisitos y diseños de alto nivel, de esta forma se elimina la necesidad de volver a inventar arquitecturas y además se posibilita la reutilización de análisis y conjuntos de requisitos, que pueden ser adecuados para las nuevas aplicaciones a desarrollar. (7)

En cuanto al nivel de reutilización de especificaciones, corresponde a la reutilización de las abstracciones de más alto nivel, aquellas que están relacionadas con el conocimiento del dominio. La incorporación de la reutilización sistemática desde la fase de especificación de requisitos, reduce el esfuerzo necesario para trasladar los conceptos iniciales del sistema a su forma final ejecutable. Para que este proceso pueda llevarse a cabo es necesario que la reutilización de requisitos esté asociada a la reutilización o generación automática o semiautomática de los elementos de diseño (esquemas de aplicación) e implementación correspondientes, de forma que permitan obtener un sistema software ejecutable y robusto.

Cuando se realizan desarrollos software que implican reutilización, ya sean desarrollos para reutilización o desarrollos con reutilización, se tiende a reutilizar componentes atómicos dentro de las categorías anteriormente descritas, los cuales se encuentran debidamente almacenados y clasificados dentro de algún repositorio.

1.3.2. __Beneficios del Desarrollo de Software Basado en Componentes (DSBC).

En esencia, un componente *“es una pieza de código pre elaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar, es un ingrediente de las aplicaciones, que se junta a otros y se combinan para llevar a cabo una tarea.”* (8)

Según un estudio hecho por Julio Casal Terreros, Ingeniero de Sistemas Computacionales de la Universidad Católica de Santiago de Guayaquil, el paradigma de ensamblar componentes y escribir código para hacer que estos componentes funcionen se conoce como Desarrollo de Software Basado en Componentes y las principales ventajas que posee su uso son:

Reutilización del software. Nos lleva a alcanzar un mayor nivel de reutilización de software.

Simplifica las pruebas. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.

Simplifica el mantenimiento del sistema. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.

Mayor calidad. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

De la misma manera, el optar por comprar componentes de terceros en lugar de desarrollarlos, posee algunas ventajas:

Ciclos de desarrollo más cortos. La adición de una pieza dada de funcionalidad tomará días en lugar de meses ó años.

Mejor ROI (Rendimiento sobre la inversión). Usando correctamente esta estrategia, el retorno sobre la inversión puede ser más favorable que desarrollando los componentes uno mismo.

Funcionalidad mejorada. Para usar un componente que contenga una pieza de funcionalidad, solo se necesita entender su naturaleza, más no sus detalles internos. Así, una funcionalidad que sería impráctica de implementar en la empresa, se vuelve ahora completamente asequible.

Esta filosofía de “comprar, no construir”, no es una idea recomendable para la UCI sin embargo sí lo es la de reutilizar los componentes construidos en los polos productivos o grupos de trabajo de la Universidad. Esto no quiere decir que si en determinado momento, por requerimientos de un cliente deba comprase un componente específico ya elaborado, no se haga, aunque no debe en paradigma de la Universidad.

Acerca del DSBC Clemens Szyperski, arquitecto experimentado de Microsoft, también emitió un criterio favorable expresando lo siguiente:

"Construir software ensamblando componentes conlleva grandes promesas para la ingeniería de software de la nueva generación. Yo iría más allá y aseguraría que no se puede hablar de ingeniería antes de haber dominado este paso … Si no, se está hablando de hacer las cosas a mano, algo similar a la manufactura temprana previa a la Revolución Industrial. Así que, es claro que deberíamos empezar a construir software basado en componentes." (8)

1.4. __ La reutilización de hoy en día en el campo de la informática.

En la actualidad existe una fuerte tendencia orientada hacia el tema apoyándose en diferentes tipos de componentes como tanto código fuente, como el resto de los artefactos que se obtienen del proceso de desarrollo como son el diseño, especificación de requisitos, manuales de usuario, planes de prueba, procedimientos de instalación, etc.

A la hora de llevar la reutilización a la práctica las tendencias actuales son establecer repositorios que permitan el trabajo conjunto de equipos de desarrollo localizados en diferentes lugares geográficos, utilizando las facilidades que ofrece Internet. La utilización de la flexibilidad y las prestaciones que ofrece la tecnología web puede aprovecharse para crear un entorno que de soporte al desarrollo con reutilización y para la reutilización de forma distribuida a través de Internet. A modo de ejemplo cabe destacar herramientas como MOREplus o Microsoft Repository, así como el incremento del número de bibliotecas de assets disponibles en Internet, siendo ASSET, ELSA, the RueseIC, y CARDS algunos de los más notables repositorios basados en tecnología web disponibles en Internet.

Generalmente las empresas que producen software se especifican en tipos particulares de aplicaciones. Existen muchas partes que son o podrán ser en un momento determinado potencialmente reutilizadas, a las cuales se les llama activos o core assets.

Para tener una mejor noción de lo que es la reutilización se deben conocer sus características, dentro de las que se pueden señalar:

- **Estratégica:** consolida lo común entre los productos, maneja estratégicamente la variación entre ellos y elimina la duplicación de esfuerzos de ingeniería.
- **Predictiva:** la reutilización de activos se da en uno o más productos sobre una línea bien definida y en lugar de reutilizar componentes de manera oportunista, se reutilizan arquitecturas de software.
- **Gestionada:** es sistemática, planificada, institucionalizada y mejorada.

No obstante, la real magnitud de esta reutilización, depende en gran medida de la generalidad y anticipación del cambio que se haya utilizado en el desarrollo de los activos. En este sentido desarrollar activos congruentes, generales, portables, robustos, documentados y probados, es claramente más caro, que desarrollarlos específicamente para una aplicación determinada. Sin embargo, si estos activos son reutilizados un número de veces suficiente, los costos se ven compensados. Se gana además en otros aspectos dentro de los que se puede mencionar: la confiabilidad en la calidad del nuevo producto elaborado, debido a que se integran activos ya probados; la planificación y la administración del desarrollo se tornan más fácil, ya que existen menos partes a desarrollar y se gana también en oportunidad de comercialización, pues desarrollos grandes y complejos pueden hacerse en un plazo de tiempo mucho menor. Por otra parte los clientes ganan en familiaridad con el nuevo software y la curva de aprendizaje se hace más corta, sobre todo si se reutilizan manuales de usuario, diseño de interfaces y mecanismos de interacción.

Un nuevo paradigma en el desarrollo de software son las Líneas de Productos de Software (LPS), que captura esta idea de la reutilización de activos de software a gran escala. Esencialmente no existe reutilización efectiva si la misma no se ha planificado previamente. El desarrollo de líneas de productos de software requiere una estructura organizacional particular, llevada a cabo por personas con roles diferentes que se encarguen de desarrollar los activos (ingeniería de dominio), integrar los productos (ingeniería del producto), o tomar decisiones estratégicas (administración) dentro de la empresa, todo esto de una forma coordinada.

En un estudio realizado por Jonás A. Montilva, miembro de la IEEE(corresponde a las siglas de The Institute of Electrical and Electronics Engineers, en español Instituto de Ingenieros Eléctricos y Electrónicos) plantea una visión muy similar en su investigación *"Desarrollo de Software Basado en*

Líneas de Producto de Software", explica que la idea básica de una LPS es el ensamblaje de partes de software previamente elaboradas, inspirada en los procesos de producción de sistemas físicos, la cual está fundamentada en la reutilización de software y que asume la existencia de una industria por partes. Jonás se basa en las siguientes definiciones de LPS:

- *"Es un conjunto de sistemas de software que comparten un conjunto común y gestionado de aspectos que satisfacen las necesidades específicas de un segmento de mercado o misión y que son desarrollados a partir de un conjunto común de activos fundamentales de una manera prescrita."* (9)
- *"Consiste de una familia de sistemas de software que tienen una funcionalidad común y alguna funcionalidad variable."* (10)

Este concepto esta más relacionado dentro del marco de la UCI al de Polo Productivo, que contienen los procesos de Administración de los Proyectos Específicos y Desarrollo, Mantenimiento y Soporte, realiza sus actividades según los elementos de funcionamiento transmitidos por la Dirección General de Producción, entrega información a esta y los productos que son generados, así como toda la documentación concebida en los proyectos Los Polos Productivos tienen como propósito establecer y llevar a cabo las actividades que posibilitan el cumplimiento de los objetivos de un proyecto en tiempo y costos esperados, así como la realización sistemática de las actividades de análisis, diseño, construcción, integración y pruebas de productos de software nuevos o modificados, cumpliendo con los requerimientos especificados.

En la definición de los polos se agrupan los productos, se trazan las líneas y prioridades claras y se alinean los proyectos. Para la UCI acercarse, al concepto de las líneas de productos, que son los polos productivos como se ha explicado, se hace deben imponer las práctica que caracterizan a este tipo de líneas de productos, el presente trabajo de diploma, propone implantar un repositorio de componentes de software reutilizables para un polo productivo, que garantice que los productos puedan ser construidos a partir de un conjunto de activos, que han sido diseñados, construidos y probados previamente con esta finalidad y que están disponibles en este lugar para quien los necesite.

Se requiere almacenar los activos en repositorios, los cuales son una base de datos especializada que almacena componentes de software y facilita la recuperación y el mantenimiento de los mismos.

Tienen como objetivo asegurar la disponibilidad de componentes para apoyar el desarrollo de un producto de software. Estos repositorios mantienen información relevante de cada activo usado, como son la especificación técnica, historia o registro de uso, clasificación y documentación.

La creciente industria del software ha buscado formas de disminuir eficaz y rápidamente, los tiempos de entrega y los costos de desarrollo de sus productos, al mismo tiempo que pretenden mejorar la calidad de estos. La reutilización es considerada como uno de los mecanismos más realistas de los planteados hasta el momento para lograr los objetivos mencionados anteriormente.

Es importante mencionar que hoy existen en el mercado varias herramientas que apoyan el concepto de reutilización. Dentro de ellas podemos mencionar las de tipo CASE, las cuales buscan automatizar los aspectos claves de todo el proceso de desarrollo de un sistema, desde su comienzo hasta el final. Estas herramientas tienen algunos inconvenientes asociados como son: los elevados costos tanto para adquirirlos como para el entrenamiento del personal en su uso y la falta de adaptación de la herramienta a la arquitectura de la información y a las metodologías de desarrollo utilizadas por la organización.

Grandes compañías de software como ORACLE, IBM, MICROSOFT, entre otras, cuentan con motores propietarios que buscan en cierto modo la reutilización de componentes, suministrando un conjunto de herramientas para el desarrollo de aplicaciones tales como el acceso a datos, la seguridad, la administración de procesos y documentación. El problema que existe con estas herramientas es el alto costo de licenciamiento, lo que le deja sólo a grandes empresas la posibilidad de adquirirlos.

Muchos sistemas empresariales necesitan ser conectados de alguna forma con otros, tanto dentro de una misma compañía como a lo largo de Internet y en muchos casos las nuevas demandas hacen que la integración sea esencial. Es entonces, cuando el desarrollo de componentes se orienta a construir grandes aplicaciones mediante la integración de activos ya existentes. Este enfoque se fundamenta porque ciertas piezas de software deben ser escritas cada vez, mientras que otras que son comunes pueden ser reutilizadas escribiéndose solo una vez. Las aplicaciones basadas en componentes hoy en día son cada vez más complejas, deben ser construidas en tiempo record y cumplir con los estándares más altos de calidad para hacer frente a esto, se concibió y perfeccionó lo

que se conoce como Ingeniería de Software Basada en Componentes (ISBC), la cual se centra en el diseño y construcción de sistemas computacionales que utilizan componentes de software reutilizables.

Microsoft propone el DSBC desde una iniciativa que incorpora la necesidad y proporciona los medios para la manufactura de software, con otras palabras define una Fábrica de Software como un ambiente de desarrollo configurado para soportar el desarrollo acelerado de un tipo específico de aplicación. Este podría convertirse el siguiente paso lógico en la evolución continua de los métodos y prácticas de desarrollo de software, trayendo como consecuencia un cambio en la industria de software introduciendo patrones de industrialización.

Un tema recurrente que se puede observar en la creación de un tipo de herramientas que permitan la automatización del proceso de desarrollo del software, es el hecho de pensar en modelos, más que solamente en objetos. Sin embargo resulta, que aún cuando los modelos juegan un rol importante, no lo son todo. Para llegar a niveles más altos de productividad, se necesita la habilidad de configurar, adaptar y ensamblar rápidamente componentes desarrollados independientemente, auto descriptivos e autónomos de ubicación, para producir así familias de sistemas similares, pero diferentes. Entonces para lograrlo, será necesaria la creación de patrones con dominio específico, frameworks y herramientas que se alineen tanto con la arquitectura del producto, como con el ciclo de vida del producto. Más aún, se deben considerar los procesos usados para analizar requerimientos, desarrollar software y ponerlo en producción. Se precisa disponer de las mejores prácticas, contenido reutilizable y distintos tipos de patrones. La aplicación de todo esto en conjunto, es lo que se conoce como Fábrica de Software. El autor considera que los criterios que se exponen sobre las fábricas de software son acertados, pues para producir con mucha mayor rapidez y calidad, es necesario tener todo lo que explican estos conceptos. Es un nivel de madurez por el que la UCI debe apostar. Estas fábricas son además una línea de producto que configura herramientas de desarrollo extensibles con contenido empaquetado y guías, las cuales son cuidadosamente diseñadas para construir tipos específicos de aplicaciones. Las tres ideas en las que se basa fundamentalmente son:

- **Un esquema de fabricación:** la analogía de esto es una receta, en la que se listan ingredientes como proyectos, código fuente, directorios y archivos de configuración, y explica cómo deberían ser combinados para crear el producto. Detalla qué Lenguajes de Dominio Específico (DSL) pueden ser usados y describe cómo los modelos basados en ellos pueden

transformarse en código y otros artefactos, o en otros modelos. Describe también la arquitectura de la línea de producto, y las relaciones clave entre componentes y frameworks que la componen.

- **Una plantilla de fábrica de software:** provee los patrones, guías, plantillas, frameworks, ejemplos, herramientas personalizadas, hojas de estilos, y otros ingredientes para construir el producto.
- **Un ambiente de desarrollo extensible:** cuando se configura con una plantilla de fábrica de software, se convierte en una fábrica de software para la familia de productos. Se asegura que estas fábricas son posibles hoy en día y que representan el intento de aprender de otras industrias, que enfrentan problemas similares y que aplican además patrones específicos de automatización a tareas de desarrollo que existen manualmente. Estas fábricas vuelven más rápida, barata y fácil la construcción de aplicaciones, concretando de esta forma la visión de la industrialización del software moderno.

Según un estudio realizado por el Dr. José Luis Fernández Sánchez, Profesor Titular de la Cátedra de Proyectos de la Escuela Técnica Superior de Ingenieros Industriales de Madrid, titulado *"Reusabilidad y Desarrollo Orientado a Objetos"*, la experiencia obtenida en la actualidad indica que en la industria existen 3 factores esenciales que llevan al éxito en la reutilización, y que no son tecnológicos sino organizativos. (11) Los factores mencionados son:

- **Soporte de la dirección.** Las decisiones en la utilización de procesos de desarrollo de software comunes, herramientas, y lenguajes, así como las inversiones en formación, construcción de activos reutilizables y adaptación de la organización son inviables sin un apoyo explícito de la alta dirección. Esto se traduce en la UCI, a la existencia de una política que enmarque según la propuesta de modelo para la producción de software en la UCI, el proceso de Gestión de Recursos, dentro del cual está el subproceso de Gestión del Conocimiento, en el que la reutilización componentes de software se hace evidente. Ya la Universidad comenzó a trabajar en este sentido y pero todavía hay que seguir trabajando en ese sentido
- **Cambios en la organización.** La reutilización sistemática conlleva la división del personal de desarrollo en dos grupos principales con funciones distintas. El grupo de Ingeniería de Dominio,

que construye los activos reutilizables y el grupo de Ingeniería de Aplicación, que construye aplicaciones con los activos existentes en la organización. En la UCI se organizan grupos de proyectos según los compromisos de trabajo, agrupándose en Polos Productivos, y todos pueden hacer aportes al repositorio de componentes de software reutilizables, según los resultados que se van obteniendo durante todo el ciclo de vida de los mismos, y a la vez, pueden reutilizar los que están publicados y que se adapten al sistema que están desarrollando.

- **Cambios en el personal.** La formación y los incentivos son útiles para cambiar prácticas habituales en el personal y disminuir la desconfianza en la utilización de código desarrollado por otros como parte del suyo propio. Cuando una organización se decide a adoptar la reutilización, debe ir de las soluciones tecnológicas más simples, como puede ser las bibliotecas de clases, hasta llegar a una mayor complejidad, como son las arquitecturas de referencia. Se concuerda con la idea de que por la falta de cultura que existe en cuanto a reutilización se trata, es importante dentro de este proceso que desde la formación del estudiantado de la UCI, se introduzca la importancia, las ventajas, la esencia de la reutilización de componentes de software.

El autor referencia en su trabajo un modelo de evolución en la adopción de la reutilización por una organización, que conlleva cinco etapas en la evolución de la misma propuesto por Griss.

- **Corta-Pega:** en esta primera fase de la evolución, la organización trata de disminuir los tiempos de desarrollo mediante la clonación de trozos de código. Aunque es una solución a primera vista, los problemas aparecen en el mantenimiento, pues a la hora de realizar cambios hay que ser consciente de todos los sitios en el código donde estos se aplican. Es un problema típico de la gestión de configuración.
- **Caja Negra:** en la reutilización de caja negra se identifican componentes software de uso común en la organización. Se garantiza que existe un original único de ellas. La utilización de bibliotecas y taxonomías de clasificación puede ayudar en esta fase de la evolución de la organización.
- **Activos Reutilizables:** en esta etapa la organización considera otros activos reutilizables distintos del código. Entre estos destacan principalmente los procedimientos de prueba, las

especificaciones, ficheros de ayuda, herramientas y otros. Con esta aproximación se incrementa la reutilización en las diversas fases del ciclo de vida del software, no solo en la fase de implementación como ocurría especialmente en la Caja Negra.

- **Arquitecturas:** en esta fase se generan componentes reutilizables, que podrían ser los anteriores, y una arquitectura software que los aglutina, permitiendo desarrollar aplicaciones en un dominio específico de negocio, sea este la educación, economía, biotecnología, entre otros.
- **Reutilización sistemática:** se basa en la estandarización de los activos reutilizables y los procesos para producirlos, la creación de una infraestructura para su producción y los mecanismos organizativos adecuados para facilitar la reutilización de los mismos. La separación del personal en grupo de ingeniería de dominio, o responsable de crear y mantener los activos reutilizables y el grupo de ingeniería de aplicación, responsable de utilizarlos, es un aspecto fundamental en esta fase.

Haciendo un análisis de lo anteriormente planteado el autor considera que en el caso de la UCI, la evolución de la reutilización de componentes de software debe empezar por una etapa similar al corta y pega que se describe por Griss, pero consideran en una segunda etapa donde se integran las dos posteriores (Caja Negra y Activos Reutilizables), lo cual genera un estudio que debe realizarse para definir con claridad cuáles son específicamente los activos a reutilizar en dependencia del tipo de software. En una fase superior, debe estar la reutilización sistemática, o sea, la creación de un grupo de ingeniería de dominio, cuya función es desarrollar componentes para que sean reutilizados por los proyectos de la Universidad. Lo que elevaría a un nivel superior en la reutilización tanto como la producción.

A criterio del autor, la reutilización como nueva disciplina dentro de la ingeniería de software, es un paso para crear el hábito y la cultura en la reutilización, así como profundizar en su importancia en el proceso de producción de software.

1.5. __ La reutilización de software en la UCI.

Los principales problemas que afronta la producción en la Universidad, y que están dados precisamente por la inmadurez de un proceso productivo que se ha ido desarrollando sobre su propia marcha, se deben a que la Universidad de las Ciencias Informáticas aun se encuentra en una etapa de madurez en cuanto a desarrollo de software se trata, ya que su incorporación a la Industria Cubana consta de alrededor de cinco años y a pesar de que en este corto período ha adquirido buenas experiencias y ha hecho grandes contribuciones, tanto organizativa como económicamente, aún está poniendo orden a su proceso de desarrollo de software, ganando en madurez y estableciendo un grupo de normas para homogeneizar el trabajo de la producción.

Dentro de los problemas que afronta la UCI se encuentran:

- Planificación y estimación de costos imprecisos.
- Poca productividad.
- Baja calidad de los productos.
- Incumplimiento con el tiempo de desarrollo estimado.
- Carencia de una metodología orientada a la reutilización.

Se habla de una carencia de una metodología orientada a la reutilización porque el tema no ha sido de los que más se ha trabajado en la Universidad, se ha visto en algunas áreas, pero no se ha impulsado desde los primeros momentos que se incorporó la UCI a la producción y ahora cuesta un poco más de trabajo adaptarse. Por otra parte no es hay una política sólida al respecto. Existen activos desarrollados en un proyecto que son necesarios en otro, para esto se debe crear un mecanismo o una coordinación entre los líderes de dichos proyectos, para llegar a un acuerdo y que el necesitado obtenga el componente.

Reutilizar resulta en ocasiones más difícil que comenzar el desarrollo del componente desde cero, problema que sería erradicado con la implantación de un repositorio de componentes de software reutilizables. Hablar de reutilización de componentes puede ser algo restrictivo, porque se pueden reutilizar muchas más cosas que un componente, hay que analizar bien entonces cuál es el concepto que se tiene de componente. Teniendo en cuenta esto se debe definir todo lo que es posible reutilizar y

como se hace. En una primera variante puede que no se obtenga todo lo que es posible reutilizar, pero sí acercarse bastante o al menos poner lo que más comúnmente se reutiliza y sobretodo, precisar cómo se documenta, para si se implanta un repositorio de elementos reutilizables, para cada elemento que se incluya algo, exista una explicación sintética, clara, de cuál es la idea y cómo se puede reutilizar.

1.6. __ Componentes que pueden ser reutilizados.

Cuando se habla de reutilizar en el marco de la producción de software, en lo primero que se piensa generalmente es en reutilizar código de programación, líneas de código compiladas en librerías o frameworks, pero el concepto de reutilización como se ha fundamentado en este capítulo, ha evolucionado. Hoy en día se reutiliza todo lo que entra en el ciclo completo de vida del software y esa es la propuesta para la concreción de la reutilización en la UCI. Para que estos componentes o activos sean reutilizables, es indispensable que estén bien documentados, para que todos puedan entenderlos. Sin comprensión plena de los componentes, no hay reutilización posible.

Es importante señalar que en dependencia del tipo software que sea, son diferentes los activos que se generan, aunque se puedan definir algunos componentes generales a reutilizar por todos, no es igual lo que puede ser reutilizado de una multimedia, que de un software de salud, o de inteligencia artificial por ejemplo. Por este motivo, una de las acciones importantes para llevar adelante la reutilización en la UCI, es hacer un estudio para definir de cada uno, qué es posible incluir en el repositorio.

Teniendo en cuenta el concepto definido de componente o activo, algunos de los que se pueden reutilizar de forma general son:

- Una especificación de requisitos.
- Un modelo de negocio.
- Una especificación de diseño.
- Un algoritmo.
- Un segmento de código.

- Un patrón de diseño.
- Una arquitectura de dominio.
- Un esquema de base de datos.
- Una especificación de prueba.
- La documentación de un sistema.
- Un plan.

Para que un componente sea reutilizable resulta imprescindible que esté bien documentado, garantizando así que se comprenda por cualquier persona que lo consulte y de esta manera, defina si cumple con los requisitos que necesita para el sistema que está desarrollando y pueda entonces utilizarlo. Es necesario que se establezcan normas para la documentación que acompaña cada uno de los componentes, de forma tal que sea redactada, teniendo en cuenta los mismos parámetros y requisitos, para una buena comprensión de los activos incluidos en la biblioteca por todos los usuarios. La documentación que acompaña a cada componente debe tener un grupo de elementos que a consideración de los autores no se deben dejar de tener en cuenta:

- Historia o registro de uso.
- Clasificación del activo.
- Descripción del activo, donde se explique de forma tal, que se pueda prescindir para su uso del equipo o persona que lo realizó.
- Mantenimiento y soporte, ayuda en línea.

1.7. __ Principales técnicas de clasificación de componentes de software.

Si se toma como ejemplo una gran biblioteca universitaria su utilización tiene disponibles decenas de millares de libros, periódicos y otras fuentes de información, lo que requiere algún sistema de clasificación para acceder a estos recursos. La forma optima de recorrer este gran volumen de información, los bibliotecarios han definido un esquema de clasificaci6n que incluye un código de clasificación de la

biblioteca, palabras reservadas, nombres de autor y otras entradas de índice. Todas ellas capacitan al usuario para encontrar el recurso necesario de forma rápida y sencilla.

Considerando un gran depósito de componentes donde residen decenas de millares de componentes de software reutilizables. ¿Cómo puede hallar el ingeniero del software el componente que necesita? Para responder a esta pregunta, surge otra pregunta más. ¿Cómo se describen los componentes de software en términos no ambiguos y fácilmente clasificables? Se trata de cuestiones difíciles y todavía no se ha desarrollado una respuesta definitiva. En esta sección, se exploran las tendencias actuales para capacitar a los futuros ingenieros del software acerca de la clasificación de los componentes.

1.7.1. __ Descripción de componentes reutilizables.

La descripción de un componente de software reutilizable se puede describir de varias formas, pero la descripción ideal se centra en el Modelo 3C (concepto, contenido y contexto). Las definiciones de estos conceptos son las siguientes:

- El *concepto* de un componente de software es una co-descripción de lo que hace el componente El concepto debe comunicar la intención del componente. Para describir un componente reutilizable, se tendrá que describir su concepto, contenido y contexto. (12)
- El *contenido* de un componente describe la forma en que se construye el concepto. En esencia, el contenido es una información que queda oculta a los ojos del usuario habitual y que solamente necesita conocer quien intentara modificar ese componente.
- El *contexto* sitúa el componente de software reutilizable en el seno de su dominio de aplicabilidad, esto es, mediante la especificación de características conceptuales, operacionales y de implementación, el contexto capacita al ingeniero del software para hallar el componente adecuado para satisfacer los requisitos de la aplicación.

Para que resulte útil en un sentido práctico, el concepto, el contenido. y el contexto tienen que traducirse en un esquema de especificación concreto. Los métodos de clasificación se pueden descomponer en tres zonas fundamentales: métodos de las ciencias de la documentación y de biblioteconomía, métodos de inteligencia artificial y sistemas de hipertexto.

La gran mayoría de los trabajos realizados sobre este tema sugieren la utilización de métodos propios de biblioteconomía para la clasificación de componentes, sin embargo en un estudio empírico de cada una de las técnicas de clasificación, Frakes y Pole indican que no existe una técnica que sea claramente la mejor y que ningún método es más que moderadamente adecuado en lo tocante a efectividad de búsqueda. (13)

La clasificación de los componentes de software debe conformar el sistema nervioso digital del repositorio, sobre cuya base pueden desarrollarse, con mayor facilidad, las acciones y operaciones que integran metodológicamente la gestión del conocimiento, el aprendizaje organizacional o el capital intelectual, en las organizaciones. Se ofrece, para cada tipo de componente, una explicación sobre su función, la lógica de sus operaciones internas, así como sus ventajas y desventajas.

Existen otras formas de clasificar los assets una de las más sencillas y natural es atendiendo a la fase del ciclo de vida donde fueron generados, así se tendrían tres categorías: *especificación, diseño e implementación*. En la categoría de especificación tendría cabida el modelo de especificación de requisitos, por ejemplo un modelo realizado mediante un lenguaje formal de especificación. En la categoría de diseño entrarían todo tipo de especificaciones y elementos de diseño, por ejemplo un esquema de una base de datos o un patrón. La última categoría se correspondería con el modelo de implementación, por ejemplo clases C++, fuentes Java, componentes visuales.

Se puede añadir una segunda dimensión, ortogonal a la anterior, por la que se pueden clasificar los componentes en *abstractos* y *concretos*. Los componentes abstractos harían referencia a las especificaciones, centrándose en el comportamiento funcional. Los componentes concretos serían las implementaciones, es decir cómo se ofrecen los servicios.

Por último se podría añadir una tercera dimensión, ortogonal a las dos anteriores, que clasificase los componentes en *plantillas* e *instancias*.

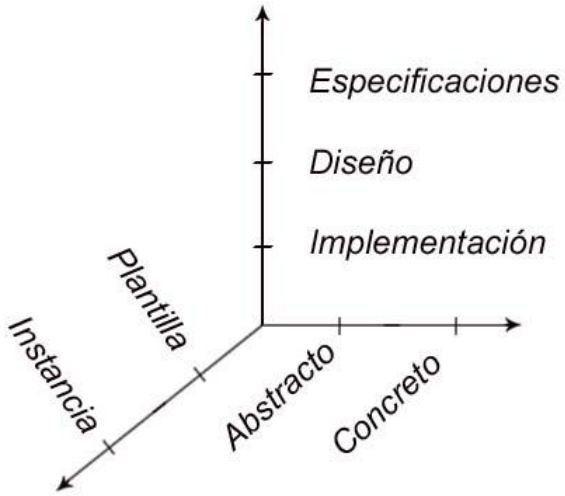


Fig. 1 Dimensiones para la clasificación de los componentes reutilizables.

Los ejes que representan las dimensiones de clasificación, forman un espacio tridimensional, en el que se encuentran los elementos reutilizables. Este espacio tridimensional no puede considerarse de forma discreta, ya que existiría una pérdida semántica al clasificar cualquier elemento simplemente como una 3-tupla compuesta de un elemento de cada dimensión. Por ejemplo, una biblioteca de clases si se considera como un todo, sería un componente reutilizable que vendría caracterizado por la triada <*implementación, concreto, instancia*>, pero si se intentase clasificar cada uno de los componentes de la biblioteca existirían elementos que se acomodarían en diferentes lugares de dicho espacio tridimensional (véase una clase abstracta en C++ <*implementación, abstracta,* instancia> y una clase parametrizada en C++ <*implementación, concreta, plantilla*>). Este problema sugiere la necesidad de elaborar criterios para la medida de la distancia continua entre los componentes en dicho espacio tridimensional, lo cual complementaría el proceso de búsqueda, no obstante dicho criterio queda fuera del ámbito del presente trabajo.

1.8. __ Herramientas para la gestión de proyectos colaborativos.

La industria del software ha dado varios pasos para fomentar la reutilización desde que se comenzó a hablar de este término hasta la actualidad, uno de ellos ha sido el desarrollado de herramientas que contribuyen con esta idea. Algunas de estas contienen elementos ya integrados, que proporcionan utilidades adicionales, contribuyendo de esta forma a desempeñar un mejor desarrollo del software dentro de una comunidad. Dentro de las herramientas más conocidas en este ámbito de la reutilización, se encuentra el Gforge, herramienta de código abierto, para el desarrollo colaborativo.

Gforge es una herramienta basada en plataforma libre para el desarrollo de software en forma comunitaria, que permite organizar y administrar grandes cantidades de proyectos. Proporciona un conjunto integrado de herramientas que facilitan el trabajo en colaboración.

Algunas de las funciones que ofrece esta herramienta son:

- Un servidor Web por proyecto (Web Site).
- Control de versiones.
- Servidor Web para la coordinación del equipo de desarrollo.
- Foros de discusión.
- Tratamiento de incidencias.
- Peticiones de mejora y distribución de parches.
- Comunicación mediante listas de distribución de correo.
- Compartición de la documentación.
- Descargas de archivos.
- Gestión de la planificación de tareas.
- Distribución de noticias significativas para el proyecto.

Dentro de sus características más relevantes se pueden encontrar:

- **Técnicas:** web, abierto, libre, escalable, robusto.

- **Funcionales:** gestión, comunicación, coordinación, centralización (agrupa fuentes de información y hace homogéneo su modo de acceso), control.
- **Utilidad de gestión de proyectos:** tareas, incidencias, listas de correo, discusiones, documentos y repositorio de código y software.

El uso del Gforge ofrece ventajas, pues por ejemplo, permite centralizar y homogeneizar la gestión de proyectos, se tienen herramientas comunes para todo el entorno donde se utilice, es capaz de centralizar los recursos técnicos de un servidor (en lugar de tener que dar soporte a múltiples máquinas por proyecto), y la ventaja esencial es que está basada en plataforma libre.

Otra herramienta con características similares es SharePoint, un conjunto de varias tecnologías que facilitan de algún modo compartir información en la red, las características principales de SharePoint son las siguientes:

- Obtener tecnología de búsqueda adaptada a la información empresarial.
- Satisfacer las necesidades básicas con una única tecnología de búsqueda para la intranet y los sitios de Internet.
- Ampliar fácil y directamente la búsqueda de los repositorios, así como los tipos de archivos normales.
- Compatibilidad del cumplimiento y protección de la propiedad intelectual, con los resultados de la búsqueda garantizados y seguros.
- Mejorar los resultados de la búsqueda mediante la indización y entrega más rápidas.
- Obtener flexibilidad con una estructura muy escalable.
- Responder rápidamente mediante una administración sólida.
- Mejorar las búsquedas mediante interfaces y aplicaciones conocidas, con resultados que se pueden procesar.
- Buscar información fácilmente con la sencilla, eficaz interfaz de usuario, así como la integración de Windows Desktop Search.
- Obtener una solución que crezca al ritmo de las necesidades empresariales.

Esta herramienta en su totalidad es muy completa, brinda una serie de servicios de interés para el desarrollo de software en equipo, como parte de una comunidad, aunque tiene en contra el hecho de

ser una herramienta propietaria. Significando esto que no admite algunas tecnologías como PHP o JSP y teniendo en cuenta además, el dinamismo de Microsoft en cuanto a sus nuevos productos, tendiendo esto a que deje de funcionar en un momento determinado.

1.9. __ Tecnologías y herramientas utilizadas para el desarrollo de aplicaciones web.

El software libre posee muchas ventajas, dentro de ellas que existe una gran comunidad de usuarios y programadores que luchan cada día contra los errores encontrados en los sistemas y por llevar a cabo nuevas soluciones que ayuden a impulsar este movimiento. Las aplicaciones web son productos informáticos que los usuarios utilizan accediendo a un servidor web a través de Internet o de una Intranet, son muy populares debido a lo práctico que resulta el navegador web como cliente ligero y a la habilidad para actualizar y mantener aplicaciones de este tipo sin distribuir e instalar software, en miles de potenciales clientes.

Las aplicaciones web generan dinámicamente una serie de páginas en un formato estándar, soportado por navegadores web comunes como HTML o XHTML. Dichas aplicaciones se clasifican en dos grupos atendiendo a la complejidad y el funcionamiento: las aplicaciones web estáticas, que son mucho más sencillas mientras que las dinámicas son páginas más complejas que dependen de una base de datos que puede ser modificada online.

Para el desarrollo de este tipo de aplicaciones existen productos de gran aceptación dentro de la comunidad como Apache servidor de aplicaciones web de código abierto al igual que PHP, lenguaje de programación de creación de páginas web dinámicas, también podemos encontrar PostgreSQL es uno de los gestores de bases de datos de código abierto más avanzado hoy en día. Ofrece control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario).

Sin embargo para el desarrollo de esta aplicación se ha utilizado Plone un CMS (Content Management System) creado por los mismos desarrolladores del servidor aplicaciones web conocido como Zope, el cual puede correr sobre diferentes plataformas tales como:

- Microsoft Windows.

• Linux (Diferentes distribuciones).

• MacOs

1.9.1. __ ¿Qué es un CMS?

Las siglas CMS provienen del término en inglés "Content Management System" lo que en español significa Sistema de Administración de Contenidos. Un CMS permite la creación, publicación y administración de contenido, su principal uso es en las páginas web. Normalmente los CMS consisten en una base de datos en donde se encuentra todo el contenido, y por medio de plantillas el contenido es desplegado de la manera en que el usuario desee. (14) Hoy en día impulsada por el auge que ha tomado el software libre la utilización de los CMS ha crecido en un número considerable y existen diferentes tipos, por ejemplo para foros, blogs, wikis, entre otros que se clasifican en:

- Entornos para colaboración.
- Bibliotecas digitales.
- Plataformas para desarrollo de gestión de contenidos.
- Plataformas para desarrollo de portales.
- Blogs o bitácoras.
- Aulas virtuales.

1.9.2. __ Razones fundamentales para usar Plone.

Este Sistema de Administración de Contenidos de código abierto permite a los usuarios crear, editar y administrar un sitio web, por lo que resulta muy fácil de utilizar. Desde su instalación se encuentra prácticamente listo para producción e integra todo lo necesario para construir desde la web más sencilla hasta la aplicación más compleja. Plone corre sobre Zope, framework implementado en Python para el desarrollo de páginas web, que permiten que el usuario interactúe de alguna manera con las páginas y la información que se presenta en ella. El contenido que sirven estas páginas puede personalizarse en

función del tipo usuario que las esté navegando, a su vez el usuario puede autentificarse dentro del sistema a través de un nombre de usuario y una contraseña. Utiliza un modelo de seguridad sofisticado, basado en roles, asegura a sus contenidos utilizando una arquitectura que incluso si su sistema estuviera comprometido, los intrusos no tendrán acceso a su servidor o a su red. Ofrece para los desarrolladores una plataforma poderosa para crear aplicaciones orientadas a contenido y para integradores permite fácil personalización y extensibilidad, además que esta dotado de una navegación flexible a través de mapas de sitio dinámicos implementados con estilos que soportan CSS y Javascript comprimidos.

Plone proporciona numerosas ventajas a sus usuarios:

- Producción muy rápida.
- Acento en los contenidos y no en la tecnología.
- Diseño adaptado por el Web.
- Gestión de contenido deslocalizado.
- Edición de las páginas en tiempo real.
- Colaboración fácil.
- Enfoque centrado en el usuario.
- Localización de la interfaz en modo nativo.
- Uso limitado de las imágenes (con la utilización masiva de CSS).
- Acento en la usabilidad.
- Apropiación de los usuarios estimulando la producción de contenidos.
- Gestión histórica y de anulación.
- Plantillas con estándares de la industria.
- Motor de búsqueda completo, indexación en tiempo real.
- Modulable, evolutivo y fácilmente personalizable.
- Motor de workflow integrado.

1.9.3. __ Particularidades de la plataforma Zope.

Zope es un servidor de aplicaciones web de código abierto escrito en Python, un lenguaje script interpretado orientado a objeto. Zope, de acuerdo a sus iniciales "Z Object Publishing Environment" es un entorno para publicar objetos y se utiliza también como una plataforma de desarrollo rápido de aplicaciones que son ricas en contenido y funcionalidad. La licencia de Zope (ZPL) es una licencia de software libre y aunque es incompatible con la GPL ha permitido que miles de desarrolladores en todo el mundo puedan descargarlo, modificarlo, agregarle funcionalidades, productos sin ningún coste.

Este framework es compatible con las plataformas de sistemas operativos más difundidas: GNU/Linux, Windows NT/2000/XP, Solaris, FreeBSD, NetBSD, OpenBSD, y Mac OS X. Permite a desarrolladores crear aplicaciones web con el solo uso de un navegador web, puede ser Mozilla, Internet Explorer, Netscape, Opera, todos compatibles para mostrar y manejar el entorno de desarrollo de Zope.

Zope cuenta con una interfaz web llamada ZMI (Zope Management Interface), que permite la creación y manipulación de objetos de una manera sencilla, rápida y amigable. Se torna muy útil para la realización de tareas de administración, gestión de contenidos, creación de usuarios, asignación de permisos, etc.

1.9.4. __ Arquitectura de Zope.

La flexible arquitectura de Zope permite dar soluciones a varios tipos de problemas. Está basada en una serie de mecanismos de intercambio de datos con el servidor Zope y un conjunto de herramientas de apoyo a ese servidor. Lo cierto es que Zope está compuesto por distintos elementos que trabajan conjuntamente manteniendo una arquitectura cliente/servidor muy completa.

• **ZServer**: Zope tiene incluido un servidor web que provee una alta conectividad. Tiene incluido un servidor FTP, Web, XML-RPC, WebDAV para el intercambio de información. Es posible también que interactúe con otros servidores web a través de los protocolos FastCGI, PCGI y HTTP.

• **Web server** (servidor web): Zope trabaja con otros servidores como es el caso de Apache si no se desea usar el que trae por defecto.

• **Zope Core**: Conocido como el ORB, es el que gestiona las peticiones de objetos e interacciona con las clases, la base de datos y los distintos productos. También está compuesto por un motor de búsqueda y una capa para la seguridad, permisos, etc.

• **Object database** (base de datos de objetos): La base de datos de objetos es la ZODB. Esta soporta transacciones, lo que permite que casi cualquier acción en Zope pueda deshacerse. Otra característica es que los objetos que deban ser persistentes requieren unos cambios mínimos.

• **Relational database** (base de datos relacional): Zope trabaja con otras bases de datos relacionales como Oracle, PostgreSQL, Sybase, MySQL y otras, en caso de que no se quiera usar la ZODB.

• **File system** (sistema de archivos): Zope puede trabajar con documentos y otros archivos almacenados en su sistema de archivos.

• **ZClasses**: Las ZClasses extienden el núcleo de Zope añadiendo nuevos objetos diseñados desde el ZMI. Lo que permite crear productos online de manera sencilla.

• **Zope products**: Los productos de Zope extienden el núcleo de Zope añadiendo nuevos objetos diseñados con Python en su sistema de archivos.

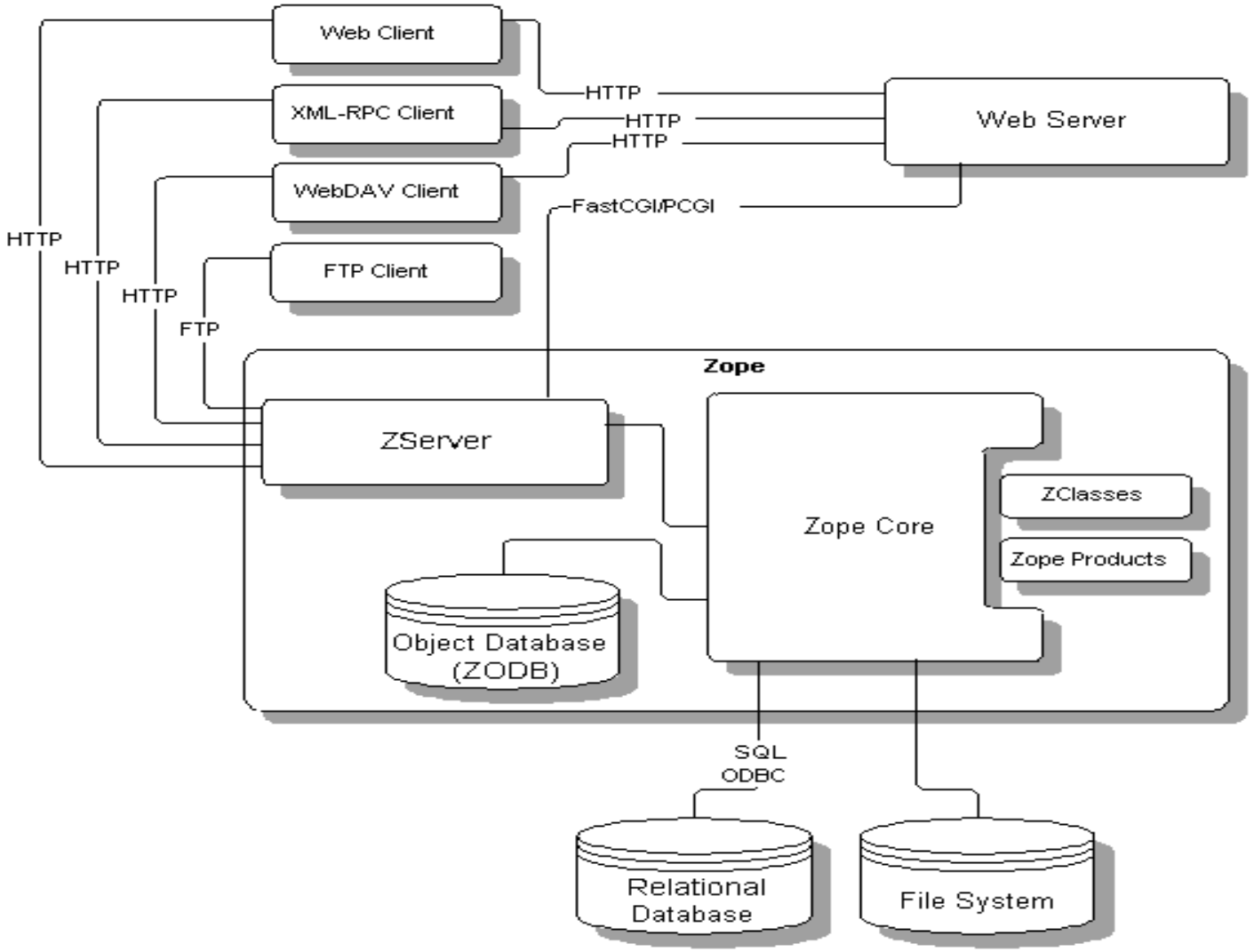


Fig. 2 Arquitectura de Zope.

1.9.5.__ Base de datos.

Una de las características fundamentales de Zope es que en su arquitectura todo son objetos, este presenta una base de datos orientada a objetos denominada Zope Object Database (ZODB) que es capaz de almacenar los objetos ordenados en forma de sistema de archivos, cada objeto con sus propiedades y

métodos, y puede a su vez contener otros objetos. Esto es muy diferente de las bases de datos relacionales normales, no obstante Zope posee múltiples conectores para las bases de datos relacionales.

Puede almacenar contenido, datos personales, plantillas dinámicas HTML, scripts, un motor de búsqueda, y conexiones con bases de datos relacionales. La potencia de este almacenamiento es enorme, sobre todo cuando se modifica dinámicamente y busca datos de los objetos que forma el sitio web.

La base de datos de Zope se usa para almacenar todos los objetos que usted crea. Es rápida y no requiere casi ningún mantenimiento. Como un sistema de archivos, es especialmente bueno en guardar los objetos binarios ligeramente-clasificados según tamaño como los gráficos.

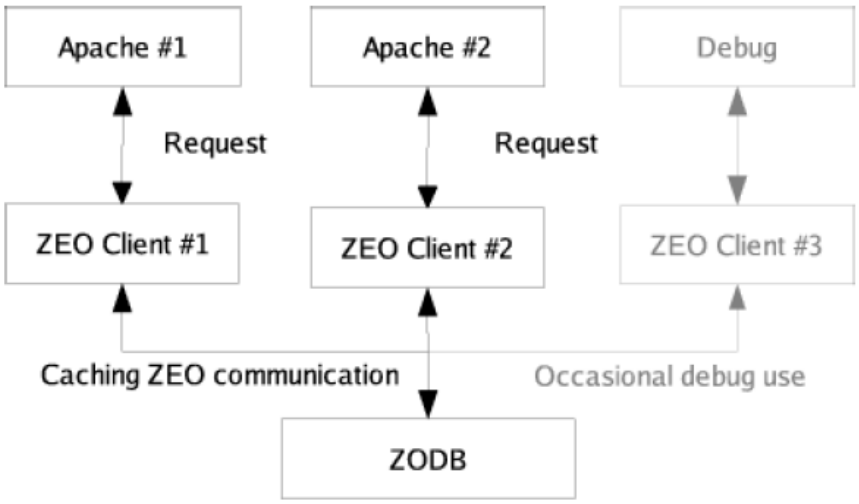


Fig. 3 Estructura de la base de datos de Zope (ZODB).

Álvaro del Castillo en su artículo “Zope, Python y la programación web” señala que Zope Object Data Base se destaca por ser casi transparente para el programador y los objetos que deban ser persistentes requieren unos cambios mínimos (heredar de una clase Python). Da soporte a las transacciones y ello permite que casi cualquier acción que realicemos en Zope pueda ser deshecha. (15)

Destaca también su alto rendimiento, que permite alcanzar un buen rendimiento incluso con bases de datos de gran tamaño. La base de datos proporciona que se pueda interactuar mediante adaptadores con otros manejadores de bases de datos como MySQL, PostgreSQL, Oracle, Sybase, entre otros.

Para el caso especifico de Plone la forma para administrar las bases de datos es por medio de los Zope Enterprise Objects (ZEO) que son los encargados de separar el acceso a Zope Object Database para que se puedan tener varias copias del sitio de Plone accediendo a la vez a la base de datos sin ninguna complicación; esto hace que se puedan reiniciar rápidamente los sitios de Plone puesto que ZEO es muy rápido cargando los objetos.

1.9.6. __ Lenguaje de Programación (Python).

Python es un lenguaje de script interpretado, (lo que significa que no se necesita compilar el código fuente para poder ejecutarlo) y orientado a objetos. Implementa estructuras de datos muy avanzadas (lista, tuplas, diccionarios) que se pueden combinar para crear otras estructuras realmente complejas. Permite además mantener de forma sencilla interacción con el sistema operativo y resulta adecuado para manipular archivos de texto, característica que hace que muchas distribuciones de GNU/Linux lo utilicen para sus herramientas de configuración.

Este lenguaje esta alcanzando un gran auge en el mundo a pesar de ser relativamente joven, debido a que presenta una serie de características que lo hacen muy atractivo. Algunas de las características más relevantes de se destacan a continuación:

• Propósito general.

Se pueden crear todo tipo de programas. No es necesario un lenguaje creado específicamente para la web.

• Multiplataforma.

Es posible utilizar Python en plataformas como Unix, Windows y otros. Puede correr en cualquier sistema siempre y cuando exista un intérprete programado para él

• Interpretado.

No se debe compilar el código antes de su ejecución. La compilación que hace el código, se realiza de manera transparente para el programador.

• Interactivo.

Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias.

• Orientado a Objetos.

La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

• Funciones y librerías.

Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos.

• Sintaxis clara.

Su sintaxis es muy visual, gracias a una notación con márgenes de obligado cumplimiento.

Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

• Libre.

Es un lenguaje gratuito. Actualmente, se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation.

1.10. __ Metodologías.

Actualmente lo más importante en el desarrollo de aplicaciones web han sido las herramientas. La facilidad de crear sitios web ha traído como consecuencia, que para el desarrollo de estas aplicaciones no se utilicen métodos y técnicas formales de análisis y diseño. El desarrollo de Internet y de los medios de comunicación ha originado un creciente interés por el desarrollo de propuestas metodológicas que ofrezcan un marco de referencia idóneo al desarrollar portales web.

"*Durante los últimos años han surgido diferentes metodologías que se mueven en entornos concretos dando más importancia a los aspectos más importantes de los sistemas que tratan*" (16). A raíz de esto varios estudiosos del tema han trabajado y propuesto metodologías que ofrecen modelos y técnicas apropiadas para el desarrollo de aplicaciones web. Dentro de las metodologías que se pueden encontrar es posible citar:

- WSDM (Web Site Design Method)
- WebML (Web Modelling Language)
- UML Based Web Engineering (UWE)
- HDM (Hypermedia Design Method)
- OOHDM (Object-oriented Hypermedia Design Methodology)
- SOHDM (Scenario-based Object-oriented Hypermedia Design Methodology)

De todas las metodologías mencionadas se pueden destacar la tres primeras como las más conocidas para la creación de portales web, además sirven de apoyo al procedimiento que se propone en este trabajo, que será guiado por la metodología RUP (Rational Unified Process). No se tratan las metodologías orientadas a hipermedia debido a que están orientadas sobre todo a la optimización del programa desde el punto de vista informático, pero no tanto a la organización de los contenidos, y esto es algo importante para la tecnología Zope/Plone.

1.10.1. __ Rational Unified Process (RUP).

El Proceso Unificado "es un proceso de desarrollo de software, es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software." (17)

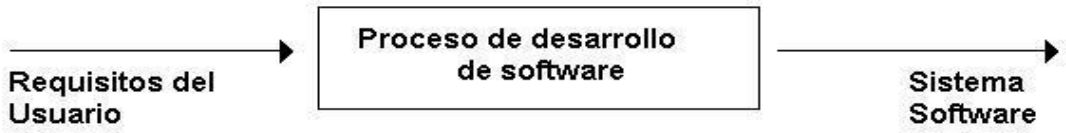


Fig. 4 Proceso de Desarrollo de Software.

RUP o "Proceso Unificado de Desarrollo" es un proceso iterativo e incremental porque propone que cada fase desarrolle en iteraciones; con esto se logra reducir los riesgos del proyecto y tener un subsistema ejecutable tempranamente. El proceso unificado está dirigido por casos de uso, quiere decir que "*el proceso de desarrollo sigue un hilo y avanzan hacia una serie de flujos de trabajo que parten de los casos de uso, estos se especifican, diseñan y los casos finales son la fuente a partir de los cuales se construyen los casos de prueba*". (17) Está centrado en la arquitectura porque muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo. Describe además, los cimientos del sistema que son como base para comprenderlo, desarrollarlo y producirlo económicamente. A medida que los casos de uso se especifican y maduran se conoce algo más acerca de la arquitectura, este proceso continúa hasta que se distinga que la arquitectura es estable.

Para el desarrollo de la propuesta se pone en práctica la propuestas de adaptación de RUP con otras metodologías ágiles, utilizado en conjunto a las técnicas de las metodologías para el desarrollo de aplicaciones web mencionadas anteriormente, con el fin de explotar lo mejor de cada una en dicho, que es de corta duración y planificación de iteraciones. Con la puesta en práctica de esta adaptación de la metodología solo se generan los artefactos que se creen necesarios para el desarrollo de la aplicación.

RUP divide el proceso de desarrollo en ciclos, cada ciclo se divide en cuatro fases: inicio, elaboración, construcción y transición; cada fase concluye con un hito. Los principales elementos del proceso unificado son: trabajadores (quién), actividades (cómo), artefactos (qué) y flujo de actividades (cuándo). En RUP se han agrupado las actividades definiéndose nueve flujos de trabajo, los primeros seis son conocidos como los flujos de ingeniería y los tres últimos como de apoyo. En la figura mostrada a continuación se muestran los nueve flujos de trabajo y las cuatro fases que define RUP.

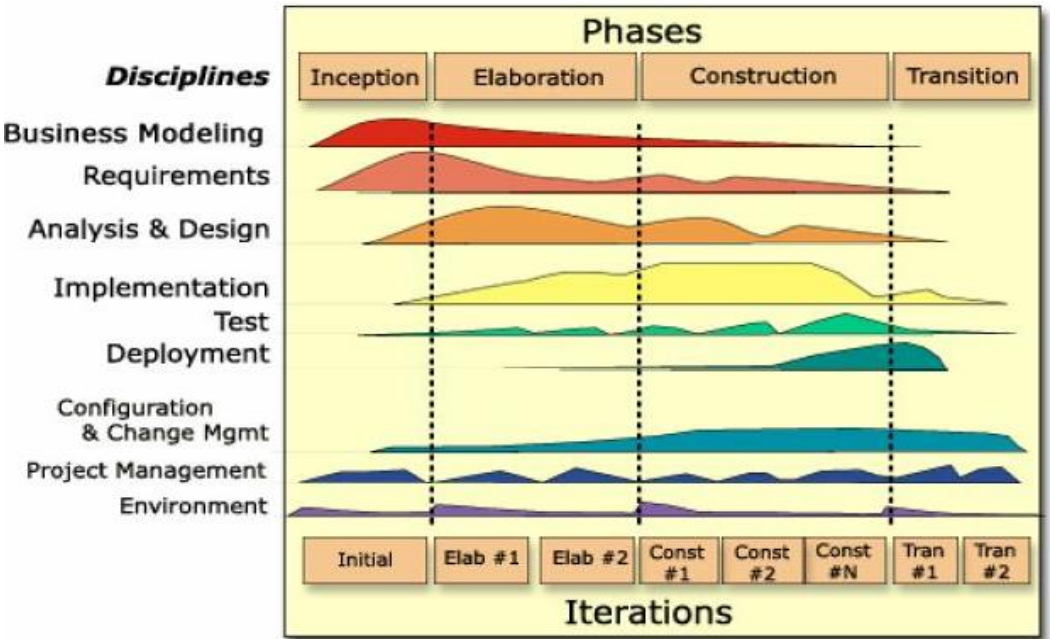


Fig. 5 Fases y flujos de trabajo de RUP

Características de los flujos de trabajo de ingeniería

• Modelamiento del negocio: describe los procesos del negocio indicando quiénes participan y las actividades que requieren automatización.

• Requerimientos: define lo que el sistema debe hacer, para lo que se identifican las funcionalidades requeridas y las restricciones que se imponen.

• Análisis y diseño: describe cómo el sistema será realizado a partir de los requerimientos, indica con precisión lo que se debe programar.

• Implementación: define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes, y la estructura de capas de la aplicación.

• Pruebas: busca los defectos a lo largo del ciclo de vida, las pruebas pueden ser de integración, de sistema, de regresión y de unidad.

• Instalación: produce la publicación del producto y realiza actividades para entregar el software a los usuarios finales.

El Proceso Unificado de Desarrollo de software representa un número de modelos de desarrollo basados en componentes que han sido propuestos en la industria. Utilizando el Lenguaje de Modelado Unificado (UML), el proceso unificado define los componentes que se utilizaran para construir el sistema y las interfaces que conectaran los componentes. *"Utilizando una combinación del desarrollo incremental e iterativo, el proceso unificado define la función del sistema aplicando un enfoque basado en escenarios (desde el punto de vista del usuario). Entonces acopla la función con un marco de trabajo arquitectónico que identifica la forma que tomara el software."* (17)

1.11. __ Conclusiones.

En el capítulo se presentó un resumen del desarrollo de la reutilización desde que se dieron los primeros pasos hasta la actualidad. Se realizó un estudio sobre la situación actual de la reutilización de componentes de software en el mundo, así como específicamente en la UCI, tomando en cuenta la importancia que tiene la implantación de un repositorio de componentes de software reutilizable. Re realizó un estudio de las tendencias actuales y estándares que componen a la tecnología Zope/Plone y que son utilizadas en el desarrollo de la propuesta. De ellas se estudiaron características y funcionalidades con el fin de comprenderlas mejor. También fueron estudiadas las características, funcionalidades y ventajas de Plone para la creación, mantenimiento y actualización de los sitios y de Zope como el servidor de aplicaciones web sobre el cual está montado Plone. Se analizaron además diversas metodologías que guiarán el procedimiento que se realizará en el próximo capítulo. Una vez conocidas las herramientas y los conceptos se puede proceder con el desarrollo de la propuesta.

Capítulo 2: Descripción de las características del Repositorio de Componentes Reutilizables.

2.1__ Introducción.

En el capítulo se define el concepto de repositorio de componentes de software reutilizables que se propone. Se realiza una descripción del mismo teniendo en cuenta el estudio realizado sobre el tema. También se describen sus objetivos, algunos factores que deben tenerse en cuenta para su implantación, los elementos que lo forman, los componentes a reutilizar y cómo debe gestionarse la información en el mismo. Además se realiza la propuesta de implantación de un repositorio de componentes.

2.2 __Características del repositorio de componentes de software reutilizables.

Los repositorios donde se guardan o conservan todos aquellos componentes o activos, que participan en el ciclo de vida de un software, que permitan el trabajo colaborativo de equipos de desarrollo localizados en diferentes lugares geográficos, utilizando las facilidades que ofrecen la Intranet e Internet, de manera tal que los componentes o activos puedan ser aportados, utilizados o consultados son conocidos como repositorios de componentes de software reutilizables. Los componentes almacenados en el repositorio deben tener una representación estándar y estar bien documentados, siendo el encargado de organizar, proteger y gestionar los mencionados componentes. Son propiedades de este repositorio:

- Un sistema de gestión de bases de datos extenso, tanto en capacidad de almacenamiento como en la variedad de componentes a guardar, que permita almacenar, recuperar y consultar, componentes de software, y que sea seguro.
- Un esquema organizativo que actúe como mapa de navegación dentro del repositorio, de manera que los usuarios puedan encontrar lo que necesiten sin grandes complicaciones.
- Una estandarización de los componentes, de forma tal que todos puedan entenderlos correctamente.
- Facilidad para crecer.

Para llevar a la práctica la implantación del repositorio, no solo son importantes los aspectos tecnológicos, o sea, qué tecnologías son fundamentales para desarrollar y mantener activos de software, pues se puede tener una base tecnológica fuerte, lista para echar a andar la reutilización de componentes en la Universidad, y sin embargo, que esta no sea efectiva debido a que los factores organizativos son también esenciales para lograr el éxito. Los tres los factores organizativos que hay que mayormente se deben tener en cuenta son: soporte de la dirección, cambios en la organización y cambios en el personal, mientras que las acciones organizativas que se hacen necesarias para implantar el repositorio son:

Alcanzar la representación estándar de los componentes.

Es esencial que todos los componentes estén escritos de la misma manera y que tengan una documentación de apoyo que los describa, para que todos puedan entender los activos que están en el repositorio. Para esto se deben establecer principios o premisas para el uso de las arquitecturas, plataformas, lenguajes de programación y herramientas para el desarrollo de software que se utilizan en la producción, debido a que este proceso productivo se encuentra un poco desordenado con respecto a esto, pues se utilizan alrededor de 18 lenguajes de programación diferentes y por ejemplo de los proyectos que programan en PHP, algunos lo hacen en PHP 4, otros en PHP 5 e incluso en PHP 6, una muestra de lo explicado se grafica en la figura 4. Se usan 11 entornos integrados de desarrollo, 15 frameworks o plugins, 5 gestores de bases de datos, 6 herramientas de modelado, 6 herramientas de gestión de proyectos y 6 sistemas operativos distintos. (18) Esto evidencia que con esta situación, es imposible agilizar en cuanto a la reutilización. En este sentido ya se han dado algunos pasos pero todavía pero todavía no es suficiente.

Incluir la reutilización de componentes de software como una nueva disciplina.

Si se quiere lograr una cultura de reutilización, que se comprendan en realidad las ventajas que su aplicación aporta al proceso productivo de software, debe comenzarse a introducir el tema desde la formación de los estudiantes, como parte esencial de la pirámide que forman la producción, la investigación y la propia formación, en la búsqueda del éxito de la naciente industria en la que se aspira a convertir la UCI. Pressman aborda en su libro “Ingeniería del Software. Un enfoque práctico” (19), el tema de la reutilización de componentes de software como Ingeniería de Software basada en componentes (ISBC), sin embargo no se profundiza sobre la misma durante le proceso de aprendizaje, tomándola como

un concepto más sin darle la importancia que tiene. La formación del personal es muy importante, cuando se requiere cambiar prácticas que son habituales y disminuir la desconfianza que causa el hecho de reutilizar componentes elaborados por otras personas.

2.2.1__ Políticas de reutilización de componentes.

El proceso reutilización de componentes debe regirse por las políticas aprobadas con anterioridad. Para lograr que el proceso de reutilización se realice de forma organizada se proponen las siguientes políticas de reutilización:

- Utilizar el repositorio de componentes designado por la dirección de la universidad.
- Para subir un componente al repositorio es necesario la autorización de los directivos que se encuentran al frente de la aplicación.
- La publicación de un componente requiere de la anterior revisión por parte del grupo de calidad, que asegure la fiabilidad y la calidad del asset.
- Los componentes almacenados en el repositorio deben tener una representación estándar y estar bien documentados.
- Los componentes deben tener un nivel de abstracción determinado que permita una amplia reutilización y no una reutilización remota.
- Posibilitar la interrelación entre componentes que permita la conformación de nuevos componentes favoreciendo el proceso de construcción de software haciendo uso de los componentes más generales hasta los más simples según las necesidades.
- La responsabilidad de organizar, proteger y gestionar dichos componentes es del sistema gestor de componentes.
- Establecer un estándar de codificación que permita la comunicación ente los programadores, con otras palabras es lograr que la comunidad a la hora de utilizar un componente no dependa de quien lo programó para su entendimiento.

- Descripción del activo, donde se explique de forma tal, que se pueda prescindir para su uso del equipo o persona que lo realizó.

Estas políticas una vez aprobadas deberán ser de estricto cumplimiento si se desea lograr el éxito de la reutilización de software y convertir la UCI en la gran factoría de software con la que se sueña.

2.2.2 __ Objetivos del Repositorio.

Una vez conocido los principales problemas que afronta la producción en la Universidad mencionados anteriormente, basado en estos se traza una vía para hacer estas dificultades menores, es la implantación de un repositorio de componentes de software reutilizables, que dentro de sus objetivos esenciales tiene:

- No repetir esfuerzos en la producción.
- Contribuir a reducir el tiempo de entrega de los proyectos.
- Contar en todo momento, con componentes de calidad para utilizarlos en cualquier etapa del ciclo de vida del software, pues se estarían mejorando constantemente.
- Contribuir a obtener productos más acabados y mejores, teniendo en cuenta que todos los componentes que se incluyan en la biblioteca deben estar certificados por calidad.

Permitir la socialización del conocimiento, lo cual se inscribe en el estilo de trabajo colaborativo que le interesa potenciar a la máxima dirección de la producción de software en la UCI.

2.2.3 __ Condiciones que propiciarán el éxito.

Hace algún tiempo se ha venido estimulando el desarrollo colaborativo dentro de la Universidad, así como la migración hacia software libre para el desarrollo de aplicaciones en forma comunitaria y basándose en esa política de la UCI se propone la implantación de esta aplicación. Un repositorio de componentes de software reutilizables permite organizar y administrar grandes cantidades de activos y proporciona un conjunto facilidades para el trabajo colaborativo desde diferentes partes de la universidad.

Para el funcionamiento fructífero del repositorio se deben crear las condiciones necesarias, por ejemplo:

- La herramienta debe ser utilizada dándole el uso para el cual fue confeccionada.
- Los componentes requieren de una actualización periódica para lograr una versión más cercana a la perfección de los mismos.
- Los componentes que se modifiquen por los usuarios deben publicarse siempre que cumplan con las políticas de reutilización para que la comunidad pueda disfrutar de sus nuevas funcionalidades.
- Crear un grupo que se dedique solamente a desarrollar este tipo de activos, lográndose así una especialización por parte de estos desarrolladores en esta materia, lo que contribuye a agilizar este proceso.
- Se necesita desarrollar una cultura de la información, así como propiciar los procesos de comunicación e información entre las personas y grupos para profundizar en la gestión del conocimiento.

Esta propuesta trae consigo algunas ventajas importantes para el buen desarrollo de la reutilización, la herramienta está basada en plataforma libre y se propone comenzar utilizar la tecnología, disponible para todos dentro de la Universidad, el repositorio de componentes reutilizables que solo requiere para su pleno uso de una buena organización y del cumplimiento de las política propuestas en el capitulo anterior así como las establecidas por la Dirección de la Universidad.

2.3__ Propuesta para el desarrollo de un repositorio de componentes de software reutilizables.

Después de haber realizado análisis la decisión que se ha tomado como la solución más ajustada para la implantación de un repositorio de componentes de software reutilizables, es el desarrollo de un repositorio que se adapte a las características del proceso productivo de la Universidad y utilizando recursos tanto tecnológicos como humanos de la UCI. De esta forma, se favorece y facilita la reutilización de componentes de software, logrando un producto que se ajusta a las necesidades de reusabilidad de la Universidad.

2.3.1 __ Descripción del problema de dominio.

Para la definición de los requisitos de la aplicación fue necesario llevar a cabo un proceso minucioso para capturar los requisitos que debería tener el sistema. El Repositorio de componentes Reutilizables para la gestión de componentes debe permitir disponer de un lugar común donde usuarios y grupos de desarrollo almacenes este tipo de componentes aprovechando las posibilidades que nos brinda el entorno intranet de la universidad. Dicho sistema debe presentar en su interfaz principal un mensaje de bienvenida que describa los servicios que se le ofrecen al usuario además de la opción de registrarse o autenticarse si ya se registrado con anterioridad. El acceso a la aplicación se realizara a través de la inserción de un nombre de usuario y una contraseña previamente escogidos y que deben validarse. Una vez registrado el usuario y después de haber validado su acceso puede seleccionar según el rango que le corresponde las siguientes actividades:

Adicionar Componente.

Permite a los usuarios ingresar componentes dentro del sistema para una posterior revisión.

Descargar Componente.

Permite a los usuarios descargar componente publicado por el sistema.

Actualizar Componente.

Permite a los usuarios ingresar una nueva versión de algún componente dentro del sistema.

Buscar Componentes.

Asiste a los usuarios de una búsqueda de componentes que puede restringirse especificando una serie de criterios con los que deben cumplir los objetos que desea encontrar.

Revisar Componente.

Permite al grupo responsable de calidad, la revisión de un componente para su aprobación o rechazo.

Publicar Componente.

Permite al grupo responsable de calidad, la aceptación de un componente para que pueda ser accedido por todos.

Rechazar o Eliminar Componente.

Permite al grupo responsable de calidad rechazar un componente dado que no cumple con los parámetros esenciales para su publicación.

Solicitar Componente.

Permite al usuario realizar una solicitud de un componente o versión de alguno con nueva funcionalidad.

Otras funcionalidades.

Debe presentar facilidades para su manejo dotada de un manual de usuario que explica en detalles como utilizar la herramienta.

2.3.2 __ Requisitos funcionales.

Para que el sistema se desempeñe a cabalidad debe ser capaz de realizar un grupo de acciones básicas a las cuales se les denomina Requerimientos Funcionales. De forma general para obtener estos requisitos fue necesario hacer un análisis del modelo de dominio. Dentro de los requisitos funcionales que el autor considera que el repositorio debe tener se encuentran:

RF1- Autenticar Sección.

RF2- Adicionar Contenido.

RF3- Descargar Contenido.

RF4- Actualizar Contenido.

RF5-Buscar Contenido.

RF6- Revisar Contenido.

RF7- Publicar Contenido.

RF8- Rechazar o Eliminar Contenido.

RF9- Adicionar Sección.

RF10- Modificar Sección.

RF11- Eliminar Sección.

RF12-Buscar Sección

RF13-Realizar Solicitud.

RF14-Buscar Usuario

2.3.3__ Requisitos no funcionales del producto.

Otros requerimientos que dotan al producto de funcionalidades son los no funcionales, ya que le dan la posibilidad hacerlo más amigable o sea fácil de usar, seguro e interactivo. Forman una parte significativa de la especificación. Son propiedades o cualidades que el producto debe tener, las cuales hacen que el mismo sea atractivo, usable, rápido o confiable.

Apariencia o interfaz externa:

Debe ser sencilla y fácil de manejar.

Usabilidad:

El sistema podrá ser usado por cualquier usuario que tenga conocimientos básicos de computación, pero está pensado y orientado a las personas que trabajan directamente con la aplicación.

Rendimiento:

Los tiempos de respuestas del sistema y los de procesamiento de la información deben ser rápidos. También debe ser rápido el tiempo de respuesta en accesos concurrentes debido a que existirán varios usuarios conectados al mismo tiempo al sitio.

Soporte:

Plone brinda la posibilidad de mejorar el sistema continuamente por lo que se puede garantizar una extensión progresiva de sus funcionalidades, esto se logra mediante la incorporación de productos.

Portabilidad:

Es capaz de ejecutarse en distintos sistemas operativos sin hacer modificaciones en el código fuente, esto se debe a que Plone resulta compatible con distintos sistemas operativos existentes en la actualidad entiéndase GNU/Linux, Windows, Mac OS X, Solaris y BSD.

Seguridad:

Para la entrada al sistema resulta obligatorio autenticarse con el objetivo de mostrar a cada usuario los servicios que le corresponde ver de acuerdo al rol que desempeñe dentro del sitio. Las contraseñas de los usuarios de la aplicación se almacenarán en paquete que viajarán por la red encriptados.

Disponibilidad:

El sistema debe estar disponible en todo momento ofreciendo sus servicios. Ha sido creado de forma que los cambios en la aplicación y nuevas versiones de los contenidos no requieran que el sistema se detenga ni ocurra ningún daño en los servicios.

Hardware:

La máquina que se utilice como servidor para el buen funcionamiento del sistema requiere como mínimo una Pentium IV, 512 de RAM, además de disponer de conectividad y requiere de gran capacidad de almacenamiento.

Navegación:

Según la página del sitio en que se encuentre el usuario siempre habrá un identificador que le indique donde se encuentra.

Legalidad:

Todas las herramientas que se utilicen en el desarrollo del sistema deben estar respaldadas por licencias bajo las condiciones de software libre.

2.3.4 __ Modelo de Dominio.

Teniendo en cuenta que la estructura y los mecanismos en el proceso de administración y actualización del repositorio no cuentan con una estructura compleja, además de que se tienen solamente definidos un grupo de conceptos acerca de lo que es un repositorio y su funcionamiento, se puede decir que por estas características se utilizará un modelo de dominio que permitirá contribuir a la comprensión del sistema y de los requerimientos que se desprenden de este contexto.

Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema. El objetivo del modelo de domino es comprender y describir las clases más significativas dentro del sistema, o sea que permitirá la comprensión del problema que el sistema resuelve en su contexto, utilizando un vocabulario común para todas las personas que se interesen, mostrándole de manera visual a los usuarios los principales conceptos que se tratan en él.

Una premisa fundamental para construir un sistema con las características esenciales es tener un amplio conocimiento del funcionamiento del proceso en cuestión, partiendo de una captura de requisitos correcta. Como primer paso se identificarán todos los conceptos que se utilizarán en el diagrama, mediante un glosario de términos sobre los nombres.

Glosario de Términos.

Contenido: es una generalización del tipo de componentes que se va a gestionar dentro del sistema, que debe permitir almacenar archivos con diferentes tipos de extensiones.

Usuario: es la persona que interactúa con la aplicación desempeñando un roll determinado.

Jefe de Calidad: es el usuario con la responsabilidad y potestad de mantener solo dentro del sistema componentes de calidad tomando las decisiones más convenientes sobre los mismos.

Administrador: es un usuario con poderes ilimitados para manejar todas las acciones que se lleven a cabo dentro del sistema, incluso para revertirlas.

Sección: es el espacio propio que le corresponde a cada usuario en el sistema que se manifestará según el roll que tenga asignado dicho usuario.

Solicitud: es una petición echa por el usuario que le permitirá solicitar un componente.

Repositorio: es el espacio que agrupara las secciones, contenidos y solicitudes echas por el usuario.

Todos estos conceptos junto a sus relaciones son representados en el modelo de objetos del dominio siguiente.

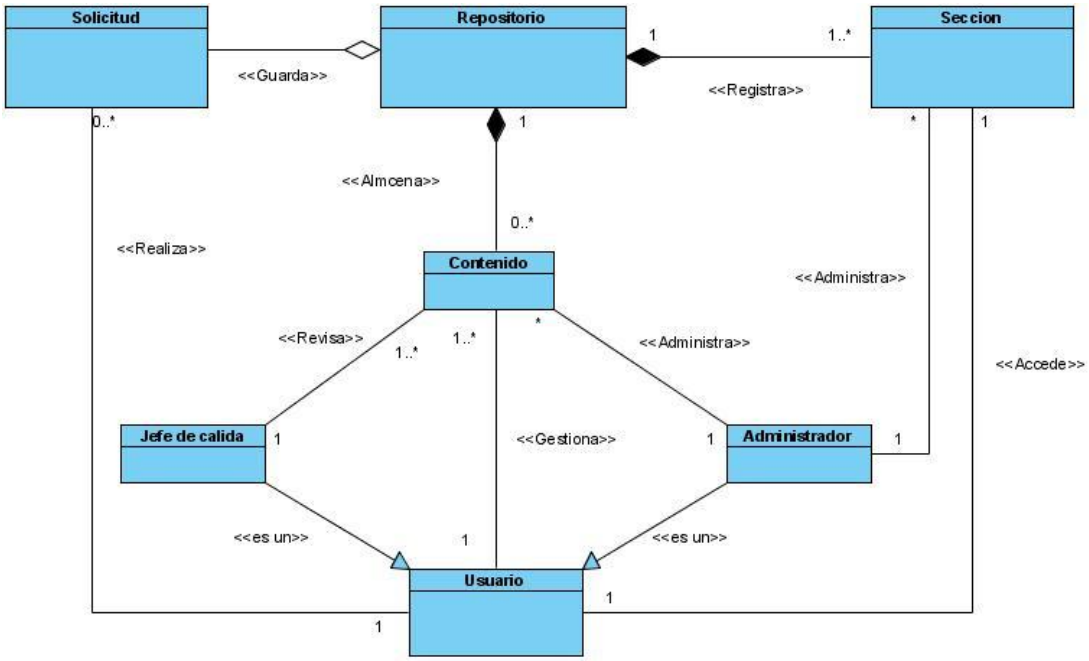


Fig. 6 Modelo de Objetos del Dominio.

2.3.5__ Actores del sistema.

Actores	**Justificación**
Usuario	Es la persona que tiene permiso a la manipulación de los activos.
Responsable de Calidad	Es la persona que se encarga de garantizar la calidad de los activos y su publicación.
Administrador	Es la persona encargada de administrar todas las actividades que se realizan en el sistema.

2.3.6__ Definición de los Casos de Uso del Sistema.

Antes de definir los casos de usos sería bueno tener en cuenta, que es un caso de uso y para que se utiliza.

¿Cómo se hiso para decidir cuales serían los casos de uso de este sistema?

Para esto se formularon varias preguntas acerca de que exactamente es lo que deseaba hacer, preguntas como:

¿Que capacidades o condiciones el sistema debe cumplir?

¿Que propiedades o cualidades debe tener el producto?

Así como cuales son las características del sistema como tal. Todo el mundo se apoya en un procedimiento semejante a la hora de tomar alguna decisión ante situaciones como esta, apoyándose en un tipo de análisis del caso que se le presenta. Lo importante es saber cuáles son esos requerimientos para obtener de modo algo que cumpla con nuestras necesidades.

Entonces se puede definir cono un caso de uso:

Una descripción de las acciones de un sistema desde el punto de vista del usuario que ayuda a los analistas a trabajar y a determinar la forma en que se usara un sistema.

Después de conocer cuál es el concepto de casos de uso y cuál es su función principal, se pueden definir los casos de uso del sistema que se propone.

Casos de Uso del Sistema

CU1	Gestionar Sección.
Actor	Administrador
Descripción	El caso de uso comienza cuando el actor solicita adicionar, eliminar o modificar una sección de usuario.
Referencia	RF1

CU2	Gestionar Contenido
Actor	Usuario
Descripción	El caso de uso comienza cuando el actor solicita adicionar, actualizar, buscar, o descargar un contenido.
Referencia	RF2

CU3	Gestionar Calidad.
Actor	Responsable de Calidad.
Descripción	El caso de uso comienza cuando un actor solicita revisar, publicar o eliminar un contenido.
Referencia	RF3

CU4	Solicitar Contenido.
Actor	Usuario
Descripción	El caso de uso comienza cuando un actor solicita la publicación de un contenido con nuevas funcionalidades.
Referencia	RF4

CU5	Buscar Usuario.

Actor	Usuario.
Descripción	El caso de uso comienza cuando un actor solicita Buscar Usuario que este registrado dentro del sistema.
Referencia	RF5

CU6	Autenticar Usuario.
Actor	Usuario
Descripción	El caso de uso comienza cuando el actor se autentifica para acceder al repositorio.
Referencia	RF6

2.3.7 __ Modelo de Casos de Uso del Sistema.

La representación de los casos de usos del sistema trae como beneficio, permitir el análisis de los casos de uso y mostrar los confines entre el sistema y el mundo exterior. Generalmente, los actores se encuentran fuera del sistema, mientras que los casos de uso están dentro de él. La manera de representar los casos de uso en un modelo se le denomina, modelo de caso de uso.

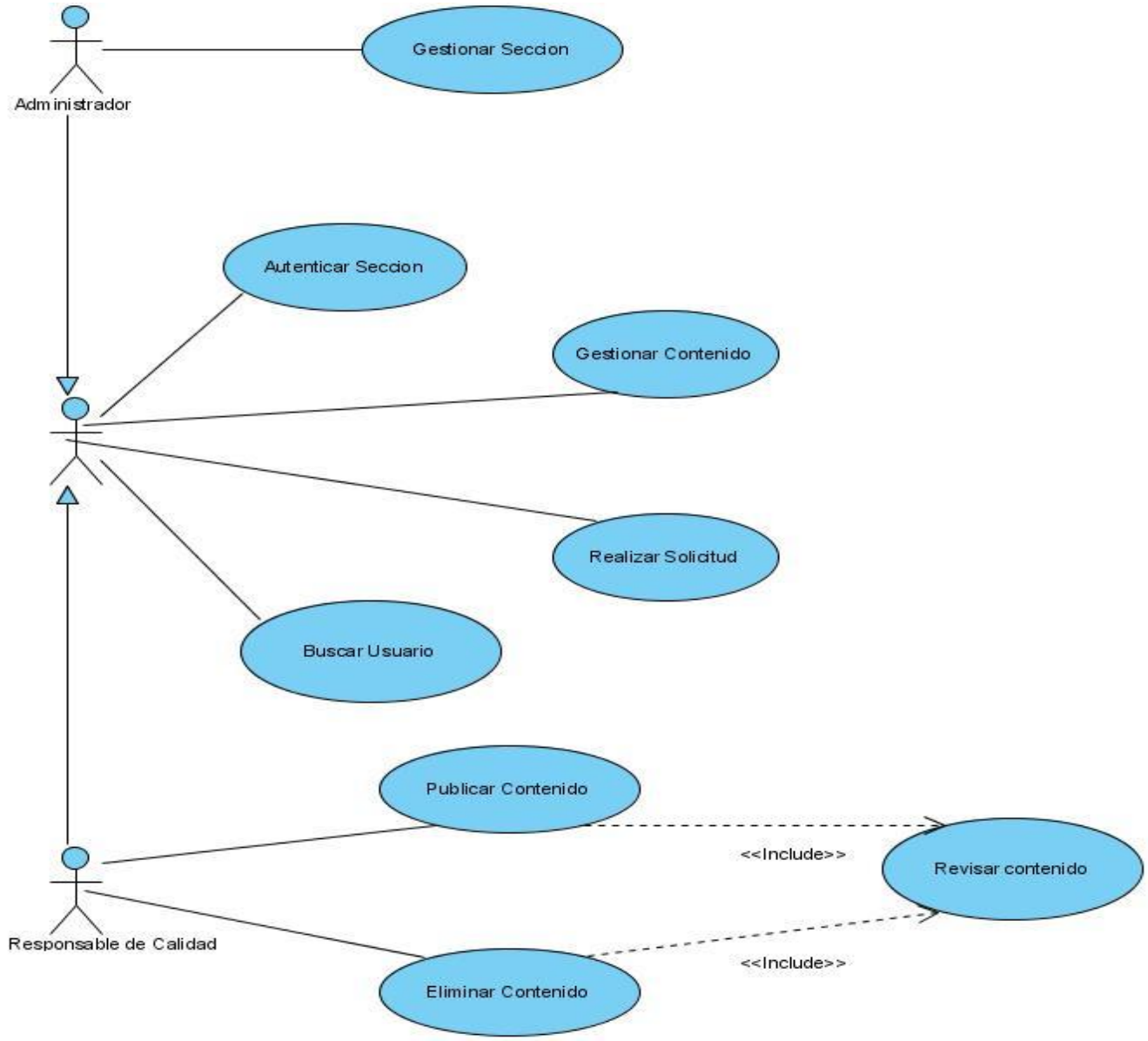


Fig. 7 Modelo de Casos de Uso del Sistema.

2.3.8 __ Descripción de los Casos de Uso.

Con el fin de obtener una mejor visión de cada uno de los escenarios que pueden manifestarse en los de casos de uso se realiza una descripción de los mismos. En las tablas que se muestran a continuación se describen los casos de usos que fueron definidos y mostrados en el modelo presentado anteriormente.

Casos de uso extendidos.

Caso de uso	
CU 1	Autenticar Usuario
Proposito	Permitir a los usuarios del sistema la realización de las acciones que tenga habilitadas de acuerdo a su rol.
Actor: Usuario	
Resumen: Este caso de uso se inicia cuando el actor, que puede tener cualquier rol, solicita autenticarse, para poder desempeñar ese roll, requiere nombre y contraseña.	
Referancia	RF1
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. Introduce sus credenciales (usuario y contraseña) y selecciona “Entrar”.	2. Comprueba las credenciales, autentica el actor y automáticamente cambia el rol del usuario convirtiéndolo en: Periodista, Editor o Revisor, de acuerdo al rol que tome así serán los permisos que le asigne en el sistema.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	2. En caso de no ser válidas las credenciales el sistema devuelve un error y brinda la posibilidad de autenticarse de nuevo.

<table>
<tr><td colspan="2">Caso de uso</td></tr>
<tr><td>CU 2</td><td>Gestionar Contenido</td></tr>
<tr><td>Propósito</td><td>Permitir adicionar, descargar, o actualizar un componente dentro del sistema.</td></tr>
<tr><td colspan="2">Actor: Usuario</td></tr>
<tr><td colspan="2">Resumen: Este caso de uso se inicia cuando el actor desea adicionar, descargar, o actualizar un contenido.</td></tr>
<tr><td>Referancia</td><td>RF2</td></tr>
<tr><td colspan="2">Precondicion: El usuario esta autenticado.</td></tr>
<tr><td colspan="2">Poscondicion:</td></tr>
<tr><td colspan="2">Curso normal de los eventos</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td>El actor envía la solicitud de:
• Adicionar Contenido.
• Descargar Contenido.
• Actualizar Contenido.
• Buscar Contenido</td><td>2. Muestra la interfaz de visión global de contenidos.
-Si el actor decide adicionar un contenido, ir a la sección “adicionar contenido”.
-Si el actor decide actualizar un contenido, ir a la sección “actualizar contenido”.
-Si el actor decide descargar un contenido ir a la sección “descargar contenido”.
-Si el actor decide descargar un contenido ir a la sección “buscar contenido”.</td></tr>
<tr><td colspan="2">Sección Adicionar Contenido</td></tr>
<tr><td>3. Introduce los datos del contenido y acepta.</td><td>4. Si los datos son válidos adiciona el</td></tr>
</table>

	contenido satisfactoriamente.
Sección Descargar Contenido	
	2. Muestra una interfaz para que el actor realice la búsqueda del contenido que desea descargar.
3. Selecciona un criterio de búsqueda para realizarla.	4. Realiza la búsqueda y muestra los resultados.
5. Selecciona el contenido y completa la solicitud.	6. Descarga el componente satisfactoriamente.
Sección Actualizar Contenido	
	2. Muestra una interfaz para que el actor realice la búsqueda del contenido que desea actualizar.
3. Selecciona un criterio de búsqueda para realizarla.	4. Realiza la búsqueda y muestra los resultados.
5. Selecciona el contenido y completa la solicitud.	6. Actualiza el componente satisfactoriamente.
Sección Buscar Contenido	
	2. Muestra una interfaz para que el actor escoja el criterio según el que desea buscar.
3. Introduce los datos.	4. Si los datos son válidos mostrara un listado con los contenidos que se corresponden con ese criterio en caso de que exista alguno.

Flujo Alternativo 1	
Sección Actualizar Contenido	
	3. Si los datos no son válidos muestra un mensaje de error y brinda la posibilidad de insertarlos de nuevo.
Flujo Alternativo 2	
Sección Actualizar Contenido	
	7. Si no existe ningún componente con esas características o los datos no son validos muestra un mensaje de error y brinda la posibilidad de insertarlos de nuevo.
Sección Buscar Contenido	
	7. Si no existe ningún componente con esas características o los datos no son validos muestra un mensaje de error y brinda la posibilidad de insertarlos de nuevo.

Caso de uso	
CU 3	Gestionar Calidad
Propósito	Permitir al grupo responsable de calidad revisar, eliminar, o publicar un componente dentro del sistema.
Actor: Responsable de calidad.	

Resumen: Este caso de uso se inicia cuando el actor solicita calidad revisar, eliminar, o publicar un componente.	
Referancia	RF3
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Sección Revisar Contenido.	
	2. Muestra una interfaz un listado con los contenidos que no han sido revisado.
3. Selecciona el contenido deseado.	
4. Revisa el contenido y emite su criterio.	5. Guarda los cambios
Sección Publicar Contenido.	
1. Selecciona la opción publicar contenido y envía la solicitud.	2. Muestra una interfaz para que el actor realice la búsqueda de los contenidos que no han sido publicados.
3. Selecciona el contenido y completa la solicitud.	4. Publica el contenido.
Sección Eliminar Contenido.	
1. Selecciona la opción rechazar o eliminar contenido y envía la solicitud.	2. Muestra una interfaz para que el actor realice la búsqueda.
3. Introduce los datos.	4. Muestra una interfaz con el contenido en caso de que exista.
5. Selecciona el contenido y completa la solicitud	6. Elimina el contenido si es un contenido publicado con anterioridad o lo rechaza si no ha sido publicado aun.
Flujo Alternativo 1	
Sección Revisar Contenido.	

	6. Si no existe ningún contenido que no ha sido revisado muestra un mensaje mostrando esta información.
Sección Publicar Contenido.	
	5. Si no existe ningún contenido que no ha sido publicado muestra un mensaje mostrando esta información.
Sección Eliminar Contenido.	
	7. Si no fue seleccionado ningún contenido muestra un cartel de información solicitando la selección de alguno y ofrece nuevamente la posibilidad.
Puntos de Extension: Ver caso de uso 6 (CU 6- Buscar Usuario)	

Caso de uso	
CU 4	Gestionar Sección.
Proposito	Adicionar, Modificar o Eliminar una Sección.
Actor: Administrador	
Resumen: El caso de uso se inicia cuando el actor desea hacer alguna de las operaciones que le están permitidas, las cuales son: insertar, modificar o eliminar alguna sección.	
Referancia	RF4
Curso normal de los eventos	

Sección: Adicionar Sección	
Acción del actor	**Respuesta del sistema**
El actor envía la solicitud de: • Adicionar Sección. • Descargar Sección. • Actualizar Sección.	. Muestra la interfaz de visión global de contenidos. -Si el actor decide adicionar una sección, ir a la sección “adicionar sección”. -Si el actor decide actualizar una sección, ir a la sección “modificar sección”. -Si el actor decide descargar una sección ir a la sección “eliminar sección”.
Sección: Adicionar Sección	
Acción del actor	**Respuesta del sistema**
	2. Muestra la planilla que el actor debe llenar para adicionar la sección.
3. Llena los campos de la planilla y completa la solicitud.	4. Actualiza la información satisfactoriamente
Sección: Modificar Sección	
	2. Muestra una interfaz para que el actor realice la búsqueda de la sección que desea modificar.
3. Selecciona un criterio de búsqueda para realizarla.	4. Realiza la búsqueda y muestra los resultados.

5. Selecciona la sección y completa la solicitud.	6. Modifica la sección satisfactoriamente.
Sección: Eliminar Sección	
	2. Muestra una interfaz para que el actor realice la búsqueda de la sección que desea eliminar.
3. Introduce un criterio de búsqueda para realizarla.	4. Realiza la búsqueda y muestra los resultados.
5. Selecciona la sección y completa la solicitud.	6. Modifica la sección satisfactoriamente.
Flujo alternativo 1	
Acción del actor	**Respuesta del sistema**
	5. Si los datos no son válidos muestra un mensaje de error y brinda la posibilidad de insertarlos de nuevo.
Flujo alternativo 2	
Acción del actor	**Respuesta del sistema**
	7. Si no existe ninguna sección con esas características o los datos no son válidos muestra un mensaje de error y brinda la posibilidad de insertarlos de nuevo.

Caso de uso	
CU 5	Realizar Solicitud.

Proposito	Permitir a los usuarios del sistema la realizar la solicitud de componente o de una versión del mismo que no estén publicados en el sistema y sean de necesidad del usuario.
Actor: Usuario	
Resumen: Este caso de uso se inicia cuando el actor solicita un nuevo contenido o una versión de alguno.	
Referancia	RF5
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción Solicitar contenido.	2. Muestra una interfaz para que el usuario introduzca los datos.
3. Introduce los datos.	4. Almacena la solicitud.
Flujo alternativo 1	
Acción del actor	**Respuesta del sistema**
	5. Si los datos no son válidos muestra un mensaje de error y brinda la posibilidad de insertarlos de nuevo.

Caso de uso	
CU 6	Buscar Usuario.
Proposito	Permitir a los usuarios del sistema buscar a otros usuarios registrados dentro del sistema.
Actor: Usuario	
Resumen: Este caso de uso se inicia cuando el actor solicita buscar un usuario.	
Referancia	RF 6

Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción Buscar Usuario.	2. Muestra una interfaz para que el usuario introduzca los datos.
3. Introduce los datos.	4. Muestra el usuario encontrado.
Flujo alternativo 1	
Acción del actor	**Respuesta del sistema**
	5. Si los datos no son válidos muestra un mensaje de error y brinda la posibilidad de insertarlos de nuevo.

2.4 __ Modelo de Análisis.

El Modelo de análisis las contiene clases del análisis y sus objetos organizados en paquetes que colaboran. Las clases del análisis: Se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos.

2.4.1__ Diagramas de clases del análisis.

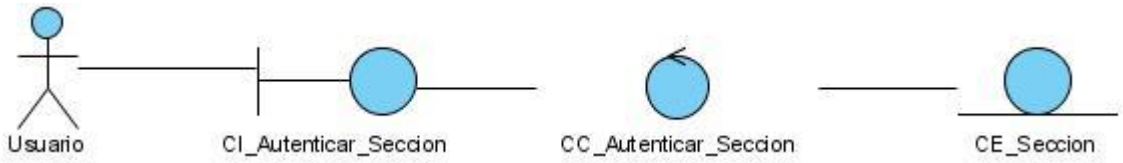


Fig. 8 Diagrama de Clases de Análisis Autenticar Sección.

Fig. 9 10 Diagrama de Clases de Análisis Buscar Usuario.

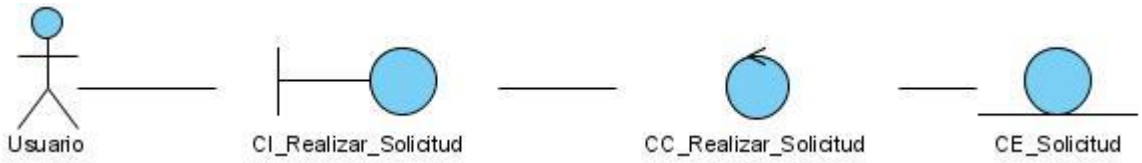


Fig. 11 Diagrama de Clases de Análisis Realizar Solicitud.

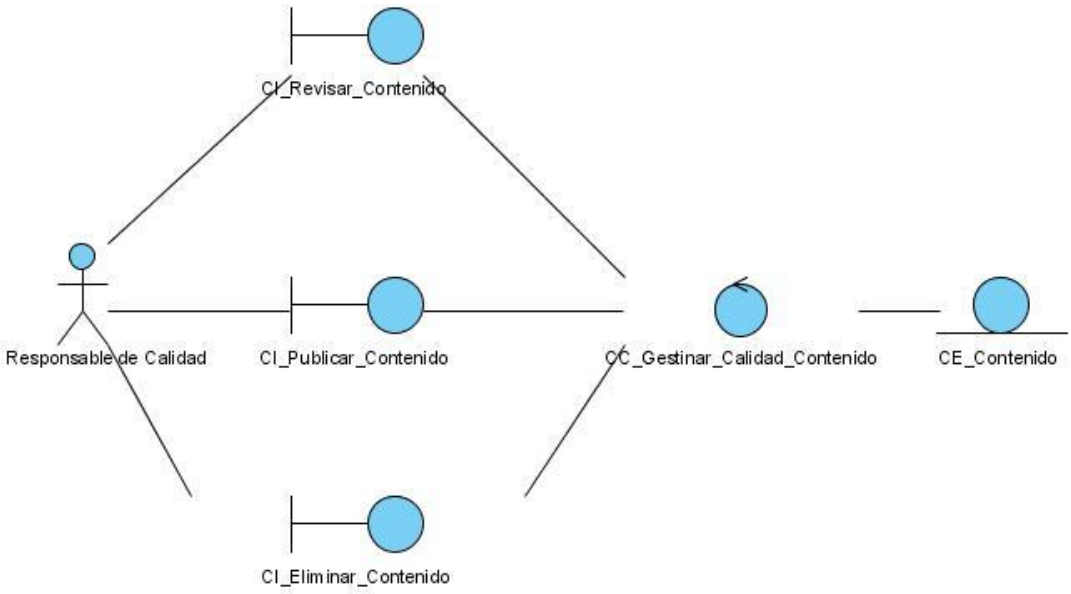


Fig. 12 Diagrama de Clases de Análisis Gestionar Calidad.

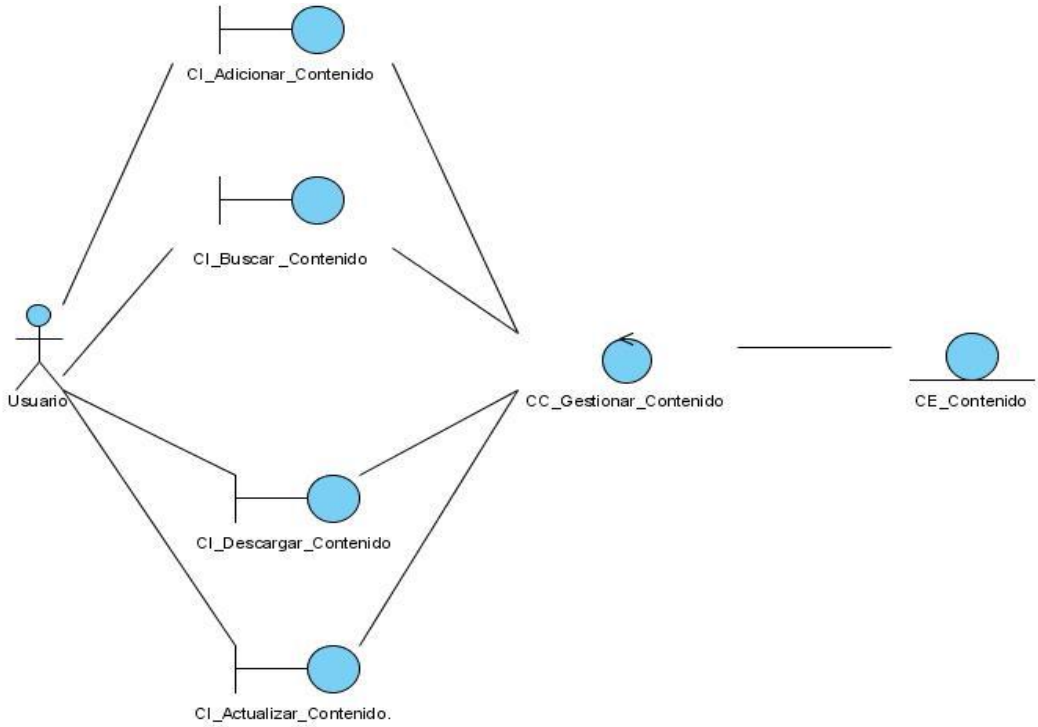


Fig. 13 Diagrama de Clases de Análisis Gestionar Contenido.

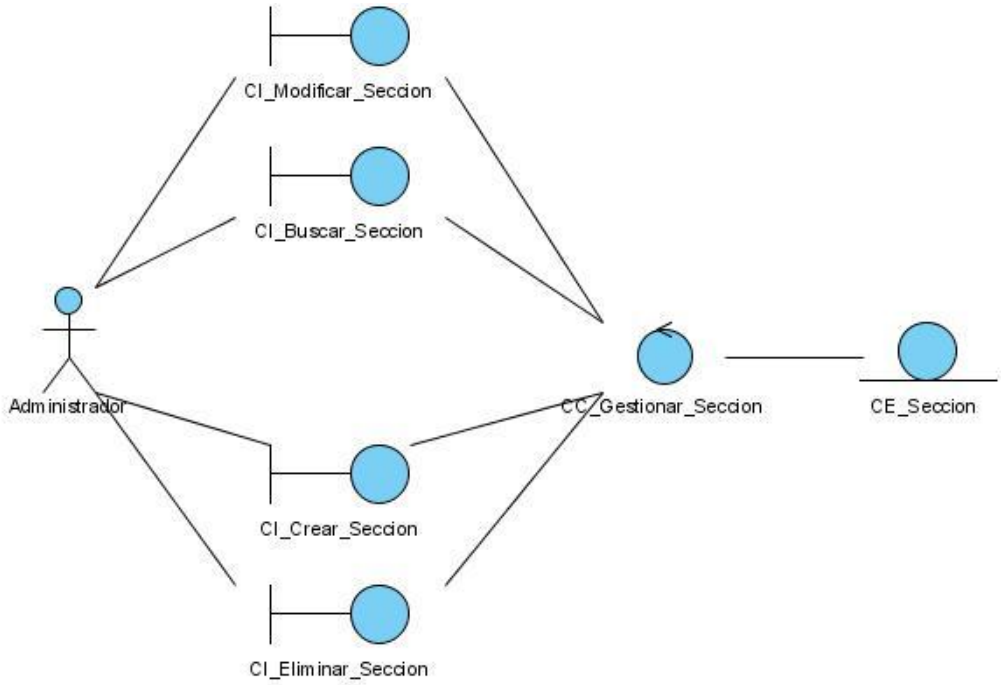


Fig. 14 Diagrama de Clases de Análisis Gestionar Sección.

2.4.2 __ Diagramas de Interacción.

Los diagramas de interacción ilustran el flujo de interacciones entre las clases y subsistemas participantes. Estos diagramas de interacción son normalmente diagramas de secuencia y diagramas de colaboración, que permiten expresar fácilmente el comportamiento de los casos de uso en función de objetos que colaboran entre sí para realizar una operación. Estos diagramas contienen objetos, enlaces, mensajes y también pueden contener notas y restricciones.

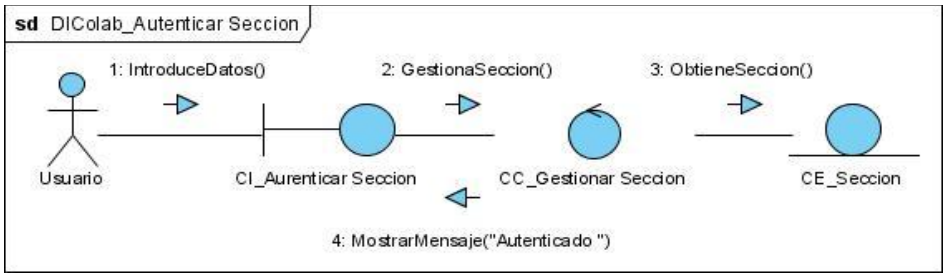


Fig. 15 Diagrama de Colaboración Autenticar Sección.

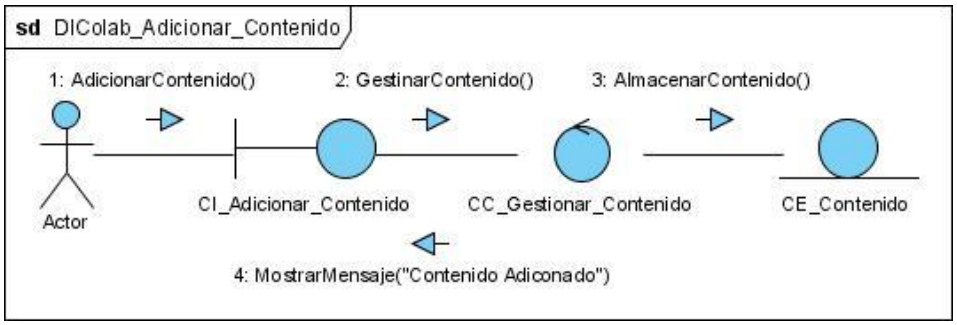


Fig. 16 Diagrama de Colaboración Adicionar Contenido.

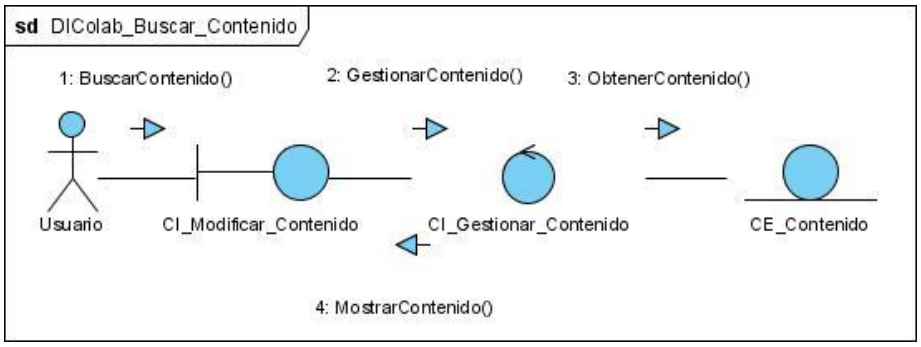


Fig. 17 Diagrama de Colaboración Buscar Contenido.

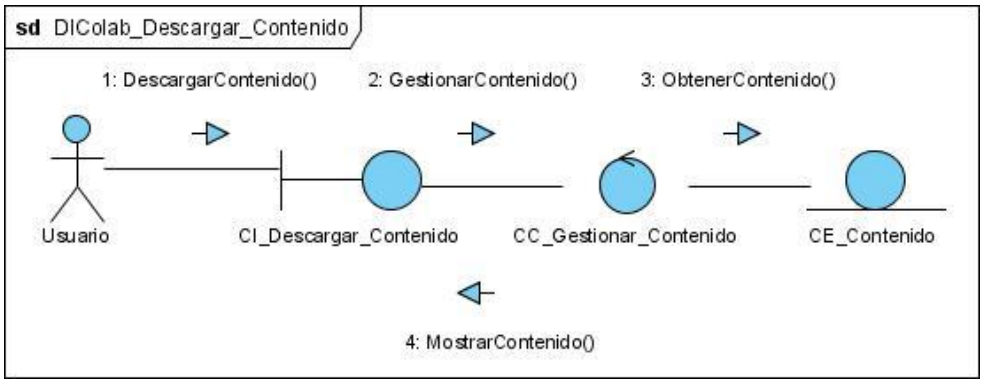


Fig. 18 Diagrama de Colaboración Descargar Contenido.

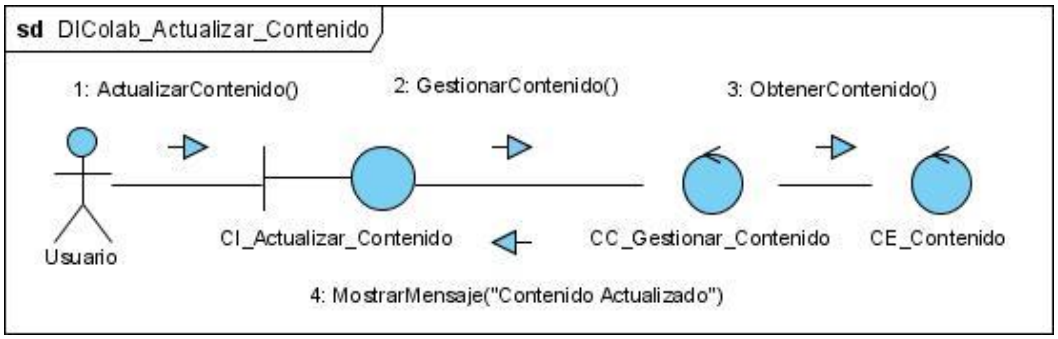


Fig. 19 Diagrama de Colaboración Actualizar Contenido.

2.4.3 __ Prototipo de Interfaz de usuario

El diseño del Repositorio de Componente Reutilizables se apoya fundamentalmente en el diseño predeterminado del CMS. Una vista de la interfaz para la gestión de los componentes se muestra a continuación.

Fig. 20 Prototipo de interfaz de usuario

Para el diseño de la interfaz de usuario de la aplicación web también se utiliza el que posee Plone que facilita el proceso de gestión documental y es sencillo y fácil de manipular. La interfaz interna muestra los servicios que ofrece el Repositorio.:

Fig. 21 Prototipo de interfaz interna.

2.5__ Conclusiones.

En el capítulo se definieron conceptos importantes para comenzar a impulsar la reutilización de componentes en la UCI, como son el de repositorio de componentes reutilizables y el de componente o

activo de software. Se describieron además las características generales del repositorio que se propone desarrollar en la Universidad, así como sus objetivos. También se mencionaron factores organizativos que son importantes para la implantación de la biblioteca. Se definieron elementos básicos que la forman y los componentes a incluir para su reutilización. Se obtuvo una visión general y detallada del de los requerimientos funcionales y no funcionales de nuestro sistema que nos permite conocer las acciones que realiza el mismo así como los actores que intervienen e interactúan con el proceso. Este paso es el principal para el desarrollo de la ingeniería del sistema, pues una buena y exhaustiva búsqueda de requisitos y una mejor selección de los casos de uso son las pautas que guían el proceso de desarrollo de la aplicación.

Capítulo 3: Beneficios y validación de resultados.

3.1__ Introducción.

La planificación juega un papel significativo en el proceso de desarrollo de un proyecto, su importancia radica, en poder estimar sus resultados y valores de costo, tiempo y recursos y determinando de esta manera si el objetivo que se persigue, es costeable o no, evitando las pérdidas de recursos que afectan la economía de la entidad o empresa donde se lleva a cabo.

En este capítulo se valida el desarrollo de la propuesta mediante las normas establecidas en el Modelo de Estimación del Esfuerzo Basado en Casos de Uso, calculándose el esfuerzo total del software. También se exponen los beneficios tangibles e intangibles a través del análisis de los resultados obtenidos

Todo proceso, ya sea de forma consciente o no, obliga en algún momento a determinar la posibilidad que tiene de ser viable. Para poder hacerlo existe un grupo de métricas que estuvieron presentes durante el desarrollo del mismo, que sirvieron de guía y apoyo a la ejecución y el control del proyecto y sus actividades.

3.2__ Planificación basada en Casos de Uso.

El primer método utilizado es el de estimación del esfuerzo de desarrollo de un producto de software a partir de los Casos de Uso y algunos factores de complejidad técnica y ambiente que influyen en el desarrollo. Para cumplir este objetivo esta basado en cálculo de los puntos de casos de uso (PCU) y cálculo del factor de peso de los actores (FPA).

Calcular los puntos de casos de uso (PCU).

(**PCU = FPA + FPCU**).

PCU = Puntos de casos de uso.

FPA = Factor de peso de los actores.

Método de estimación del esfuerzo de desarrollo de un producto de software a partir de los Casos de Uso y algunos factores de complejidad técnica y ambiente que influyen en el desarrollo.

Calcular el factor de peso de los actores (FPA)

Nombre del actor	Complejidad
Administrador.	**Persona que interactúa con el sistema a través de una interfaz gráfica. Complejidad 3.**
Responsable de calidad.	**Persona que interactúa con el sistema a través de una interfaz gráfica. Complejidad 3.**
Usuario.	**Persona que interactúa con el sistema a través de una interfaz gráfica. Complejidad 3.**

Tabla 1 Actores que interactúan con el sistema a través de una interfaz grafica.

Se tienen tres actores todos con complejidad 3 puesto que todos interactúan con el sistema mediante una interfaz gráfica.

FPA = 3 x 3 = 9.

Factor de peso de los Casos de Uso (FPCU).

Este valor se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción se entiende como una secuencia de actividades atómica

Caso de uso	Peso
Autenticar Sección.	**5**
Gestionar Contenido.	**10**
Gestionar Sección.	**10**
Revisar Contenido.	**5**
Publicar Contenido.	**5**
Eliminar Contenido.	**5**
Buscar Usuario	**5**
Realizar Solicitud	**5**

Tabla 2 Peso de los casos de uso.

Se tienen seis casos de uso con complejidad simple y dos de complejidad media.

FPCU = 6 x 5 + 2 x 10= 50

Calculando los puntos de casos de uso.

PCU = FPA + FPCU

PCU = 9 + 50 = 59

3.3 __ Ajuste de los puntos de Casos de Uso (PCUA).

Después de calculados los Puntos de Casos de Uso (PCU) estos se deben ajustar teniendo en cuenta un grupo de factores técnicos y ambientales.

PCUA = PCU X FCT X FA

PCUA = Puntos de Casos de Usos Ajustados.

FCT = Factor de Complejidad Técnica.

FA = Factor de Ambiente.

Calcular el Factor de Complejidad Técnica (FCT).

El Factor de Complejidad Técnica (FCT) Se estima mediante la cuantificación del peso de un grupo de factores que determinan la complejidad técnica del software. A cada factor se le asigna un valor entre 0 y 5 de acuerdo con la relevancia.

FCT = 0.6 + 0.01 x Σ (Peso i x Valor i)

Factor	Descripción	Valor	Peso
T1	Sistema distribuido.	0	2
T2	Objetivos de rendimiento o tiempo de respuesta.	2	1
T3	Eficiencia del usuario final.	4	1
T4	Procesamiento interno complejo	2	1
T5	El código debe ser reutilizable	4	1
T6	Facilidad de instalación	2	1
T7	Facilidad de uso	4	0.5
T8	Portabilidad	4	2
T9	Facilidad de cambio	1	0.5

T10	Concurrencia	3	1
T11	Incluye objetivos especiales de seguridad	2	1
T12	Provee acceso directo a terceras partes	1	1
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	0.5

Tabla 3 Factor de complejidad técnica.

FCT = 0.6 + 0.01 x (0 x 2 + 2 x 1 + 4 x 1 + 2 x 1 + 4 x 1 + 2 x 1 + 4 x 0.5 + 4 x 2 + 1 x 0.5 + 3 x 1 + 2 x 1 + 1 x 1 + 1 x 0.5)

FCT = 0.91

Factor de Complejidad Ambiente (FA).

Para calcular el Factor de Complejidad Ambiente se consideran las habilidades, entrenamientos y experiencias del grupo de desarrollo, se estima de forma similar al FCT.

FA = 1.4 - 0.03 x Σ (Peso i x Valor i)

Factor	Descripción	Valor	Peso
E1	Familiaridad con el modelo de proyecto utilizado	3	0.5
E2	Experiencia en la aplicación	1	1
E3	Experiencia en orientación a objetos	3	1

E4	Capacidad del analista líder	3	0.5
E5	Motivación	5	1
E6	Estabilidad de los requerimientos	4	2
E7	Personal a tiempo compartido	3	1
E8	Dificultad del lenguaje de programación	3	0

Tabla 4 Factor de complejidad ambiente.

FA = 21.4 - 0.03 x Σ (3 x 0.5 + 1 x 1 + 3 x 1 + 3 x 0.5 + 5 x 1 + 4 x 2 + 3 x 1 + 3 x 0)

FA = 0.71

Con los Puntos de Casos de Uso PCU (sin ajustar) y los Factores de Complejidad Técnica (FCT) y de Ambiente (FA) calculados:

PCUA = PCU X FCT X FA

PCUA = 59 x 0.91 x 0.71 = 38.12

3.4 __ Esfuerzo (E).

Para convertir los Puntos de Casos de Uso Ajustados a Esfuerzo de desarrollo.

E = PCUA x FC con **FC: Factor de Conversión**

Para cada punto de caso de uso ajustado, Gustav Karner creador de ese sistema sugirió 20 H/h, pudiendo ser calibrado entre 15 y 30 H/h en dependencia del FA. (22)

E = 38.12 x 20 H/h = 762.4 H/h

3.5 __Tiempo de desarrollo.

El esfuerzo total es equivalente al tiempo de desarrollo de la aplicación teniendo en cuenta esto se pueden estimar la duración del proyecto mediante la obtención de la cantidad de horas por hombre.

Esfuerzo ≠ Tiempo

TDES (total) = E (total) / CH (total)

TDES: Tiempo de Desarrollo **CH**: Cantidad de Hombres

Hombres – Mes x 192	Hombres - Hora
Hombres – Mes x 24	Hombres - Día
Hombres – Mes / 12	Hombres - Año

Tabla 5 Conversiones.

TDES (total) = 762.4 H/h / 1 h

TDES (total) = 762.4 H

Tiempo estimado para desarrollar las actividades del proyecto es de 762.4 horas equivalente a 32 días.

3.6 __ Costo total.

El costo total se puede calcular a través del producto entre el esfuerzo total y el costo de cada hombre por hora.

C (total) = E (total en HH) x CHH

CHH: Costo de Hombre por Hora.

CHH = K x THP

K: Coeficiente que tiene en cuenta los costos indirectos

(1,5 y 2,0)

THP: Tarifa Horaria Promedio

El THP es se obtiene mediante el calculo del promedio entre el salario de las personas que trabajan en el proyecto divididos entre la cantidad de horas al mes.

CHM = 24 x 8 H

CHM = 192 H

C (total) = E (total en HH) x K x THP

En este caso se toma un salario medio de $225.00

THP = $225.00 / 192

THP = $1.17

Suponiendo que el coeficiente de los costos indirectos es 1.5.

C (total) = 762.4 x 1.5 x $1.17

C (total) = $ 1338.00

El costo total del proyecto es de $ 1338.00

3.7 __ Beneficios tangibles e intangibles.

El sistema trae consigo una gran cantidad de beneficios, la mayor parte de los mismos son intangibles. Está realizado fundamentalmente para que los desarrolladores de un proyecto estén dotados de una herramienta que le permita el trabajo colaborativo a distancia y poder hacerlo de forma dinámica y estandarizada.

El uso de esta herramienta contribuirá a que el proceso de desarrollo quede marcado por la reducción del tiempo y costos de producción pues brindará una serie de posibilidades que actualmente no existen dentro de la UCI de manera estándar, como por ejemplo la facilidad de manejar componentes, la

búsqueda optimizada de contenidos y usuarios dentro del sistema, gran ventaja en cuanto a la notificación de los cambios en la información, pues, al permitir el acceso a la misma desde cualquier lugar.

La aplicación presente ventajas en cuanto a servicios prestados, procesamiento de datos, tiempo de respuesta, gestión unificada de grandes volúmenes de información; todo esto traducido en orígenes de información, centralización y organización de la misma. Estableciendo una relación más estrecha entre desarrolladores, posibilitando la socialización del conocimiento.

3.8 __ Análisis de costos y beneficios

Para analizar los costos y beneficios se tuvieron en cuenta una serie de aspectos como la plataforma y el software utilizado, la adquisición de equipamiento nuevo, así como los beneficios que arroja la puesta en marcha del sistema. A continuación se amplían estos aspectos.

Los software utilizados en para el desarrollo y puesta en práctica del producto son de código abierto y en el caso de Plone gratuito por lo que nos es necesario hacer gastos de licencia en su implementación, ni para su implantación.

Para el desarrollo de la aplicación no fue necesario adquirir equipamiento nuevo ya que todos están disponibles, se cuenta con los recursos técnicos necesarios de tipo tecnológico (hardware o software). En cuanto a la utilización de recursos humanos con la participación de una persona se pudo llevar a cabo todo el proceso de desarrollo puesto que la aplicación a pesar de constituir una herramienta de valioso alcance está clasificada dentro de las aplicaciones de tamaño pequeño.

Se tienen en cuenta los beneficios que se obtendrán al poner en marcha el sistema, en cuanto a calidad en la producción, ahorro de recursos tanto tecnológicos como humanos, tiempo y esfuerzo, el costo de desarrollar la aplicación no es grande.

Luego de realizado el análisis de los beneficios tangibles e intangibles que reportará el sistema desde el punto de vista del usuario y teniendo en cuenta los costos del mismo se decide a proceder con la puesta en marcha del mismo. De modo que el usuario pueda contar con una herramienta poderosa en el manejo de contenidos, que disminuirá los gastos en tiempo y esfuerzo. Concluyendo de esta manera que ha sido factible su desarrollo y puesta en marcha.

3.9 __ Conclusiones.

Este ha sido un paso imprescindible en el desarrollo de una propuesta que satisfaga las necesidades y comodidades de los usuarios y clientes, pues ha permitido demostrar la factibilidad de la ejecución del proceso, a través de la información basada en los cálculos. Estos cálculos aunque no son exactos permitieron ser utilizados a lo largo del desarrollo e incluso también tendrán utilidad para futuros desarrolladores. El estudio ha demostrado la factibilidad de la ejecución del proyecto Repositorio de Componentes Reutilizables, que aunque no muestra beneficios tangibles si muestra beneficios en cuanto a lo que representa esta herramienta para la facilitación del manejo y gestión de componentes. El sistema es de alto valor para los encargados de llevar a cabo el proceso del ciclo de vida de un proyecto por las ventajas que proporciona la reutilización y la utilización de los repositorios de componentes reutilizables.

CONCLUSIONES GENERALES.

Con la realización de esta investigación se arriba a las siguientes conclusiones:

- Existe en la actualidad tendencia a nivel mundial en cuanto al Desarrollo de Software Basado en Componentes, permitiendo agilizar este proceso y de esta forma cumplir con las expectativas del cliente en cuanto a tiempo de entrega y calidad del producto
- Los repositorios de componentes reutilizables constituyen un instrumento eficaz para el enriquecimiento de las capacidades de solución a problemas de reutilización y obtención de productividad
- El Repositorio de Componentes Reutilizables aumenta el valor de los componentes que en él se publican, contribuyendo a disminuir los costos de producción y a elevar la calidad del proceso de desarrollo.

RECOMENDACIONES

- Que se motive a los desarrolladores a utilizar la herramienta para la gestión de componentes.
- Que sea utilizado el sistema propuesto como apoyo al proceso productivo de la universidad.
- Que se continúe desarrollando el sistema propuesto a partir de los nuevos requisitos que puedan surgir como resultado de su explotación.
- Que se creen un conjunto de componentes reutilizables de calidad que permitan la reutilización entre los distintos Polos Productivos de la Universidad.

BIBLIOGRAFIA

1. **España, Real Academia de.** [en linea.] s.l. : http://buscon.rae.es/draeI/, 11/2/2007.

2. **Reinoso Vigoa, Yordamy.** Diseño e implementación de componentes reutilizables para la capa de interfaz del proyecto Telebanca 2.0. La Habana, Cuba. : s.n., junio-2007.

3. **KRUEGER, Charles W.** *Software Reuse, ACM Computing Surveys. Volumen 24 no. 2.* Junio,1992.

4. **MEYER, B.** *Construcción de Software Orientado a Objetos. Segunda Edición. .* Madrid, : s.n., 1999.

5. **Wikipedia, La Enciclopedia Libre.** [en línea] s.l. : http://es.wikipedia.org/wiki/Biblioteca_digital, 16/2/2008.

6. *Manual de terminología.* [en linea] s.l. : http://www.translationbureau.gc.ca/index.php?lang=francais&cont=s_700. 22/2/2008.

7. **Peñalvo, Francisco José y otros autores.** *Mecano: Una Propuesta De Componente Software Reutilizable.* [en linea] s.l. : http://www.giro.infor.uva.es/oldsite/docpub/jis97.html, 22/2/2008.

8. **Terreros, Julio Casal.** *Desarrollo de Software basado en Componentes .* [en linea] s.l. : http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3985/default.aspx#M2, 24/2/2008.

9. **CLEMENTS, P. and NORTHROP, L.** *Software Products Line: Practices and Patterns.* s.l. : Addison Wesley, 2002.

10. **SODHI, J. and SODHI, P.** *Software reuse: Domain analysis and design processes.* New York : s.n., 1999.

11. **GRISS, M.L. WOSER, M.** *Making Reuse Work at Hewlett-Packard.* s.l. : IEEE Software, enero 1995.

12. **Whittle, B.** *Models and Languages for Component Description and Reuse, ACM Software engineering Notes, vol. 20.* 4/1995.

13. **Frakes, W.B., y T.P. Pole.** *An Empirical Study of Representation Methods for Reusable Software Components.* s.l. : IEEE Trans. Software Engineering, 8/1994.

14. **Lotze, T and Theune, C.** *Content Management with Plone: A Handbook for Authors and Editors.* 10/ 2006.

15. **Del Castillo, Alvaro.** *Zope, Python y la programación web.* [en linea] 25/4/ 2008.

16. **Escalona, M.J. & Koch, N.** *Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.*

17. **Jacobson, I.,Booch G., Rumbaugh, J.** *UML: El Proceso Unificado De Desarrollo de SoftWare.* La Habana : Félix Varela, 2004.

18. *La producción en la UCI, "Mi proyecto, EL PROYECTO FUTURO".* **Machado, Alejandro Gabriel.** Abril 2007.

19. **Pressman, Roger.** *Ingenieria del Software: Un Enfoque Prcictico.* ver el año.

20. **Peralta, Mario.** *ESTIMACIÓN DEL ESFUERZO BASADA EN CASOS DE USO.* 2004.

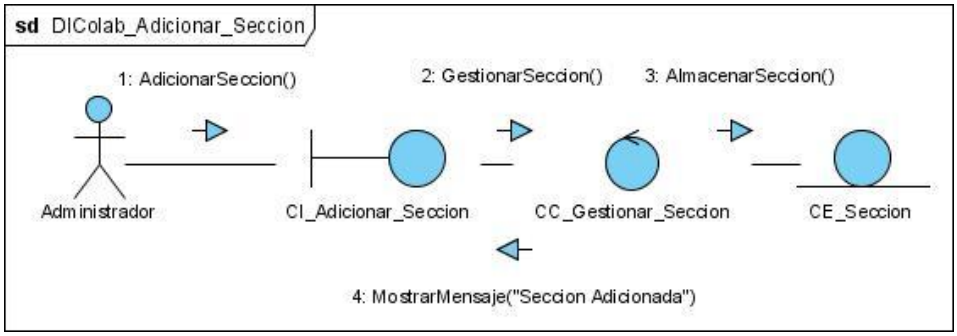
21. **Ceri, Stefano y otros.** *Web Modeling Language (WebML): a modeling language for designing Web sites.* 6/2000.

22. **Appelmans, Thomas.** *Web Globalization and WSDM Methodology of Web Design.* 2004.

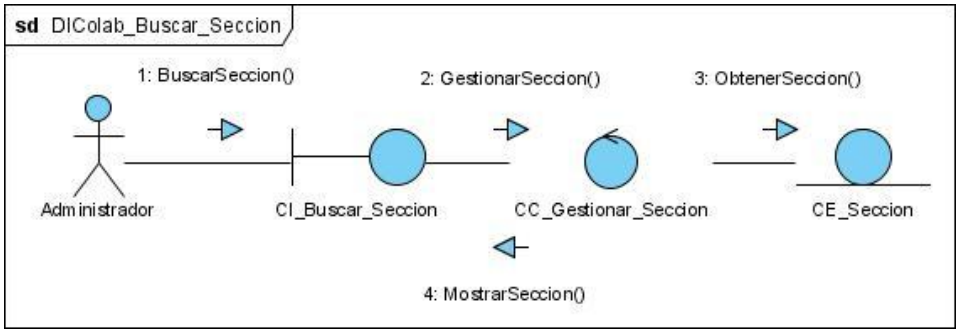
ANEXOS

1. __ Diagramas de Interacción por escenario.

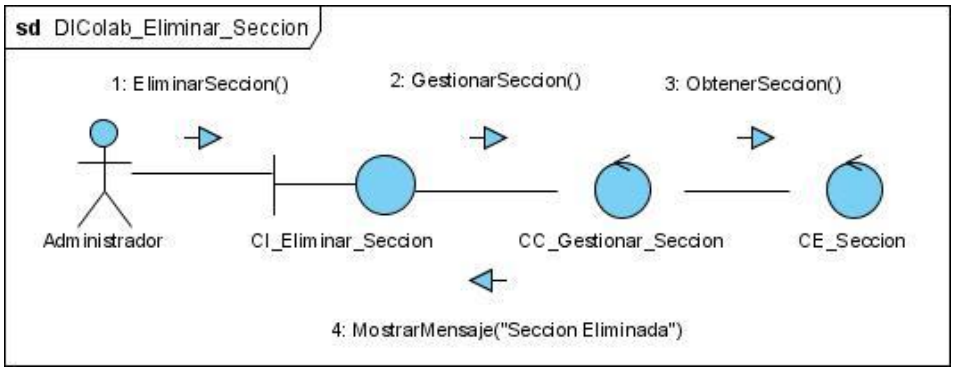
Caso de Uso Gestionar Sección.



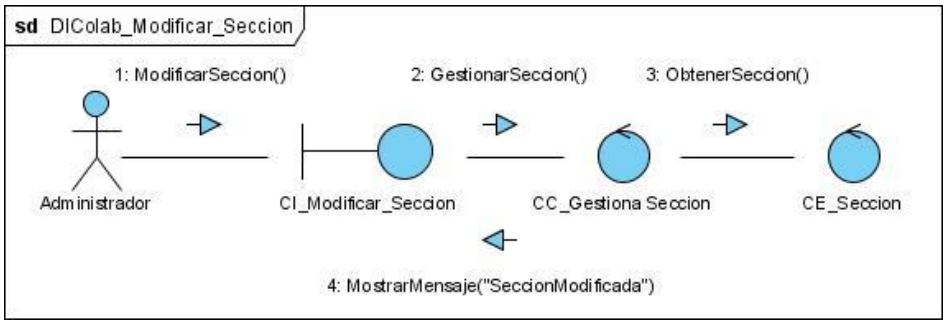
Anexo 1 Diagrama de Colaboración Adicionar Sección.



Anexo 2 Diagrama de Colaboración Buscar Sección.

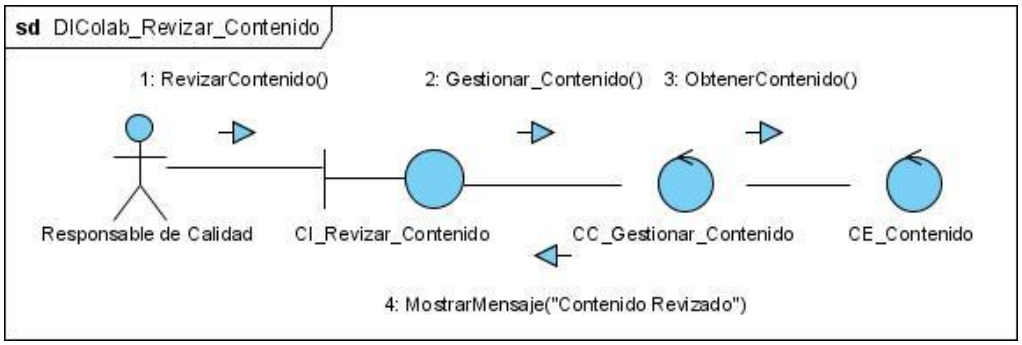


Anexo 3 Diagrama de Colaboración Eliminar Sección.

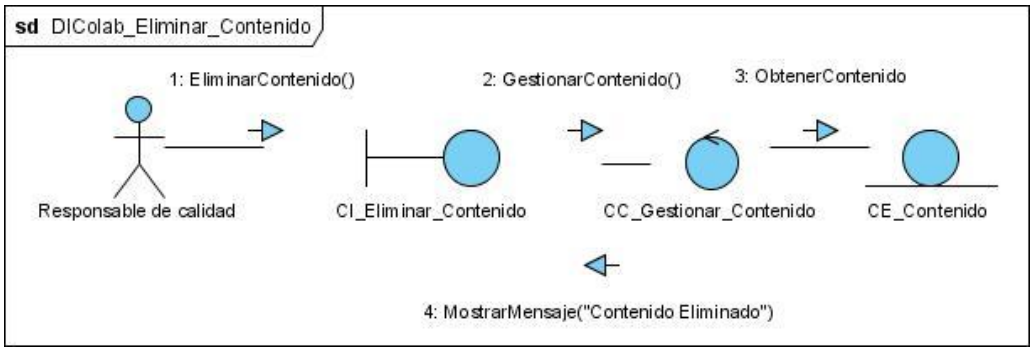


Anexo 4 Diagrama de Colaboración Modificar Sección.

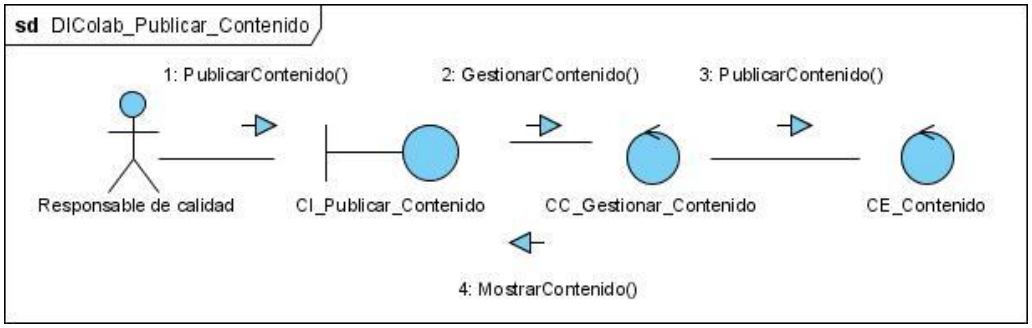
Caso de Uso Gestionar Contenido.



Anexo 5 Diagrama de Colaboración Revisar Contenido

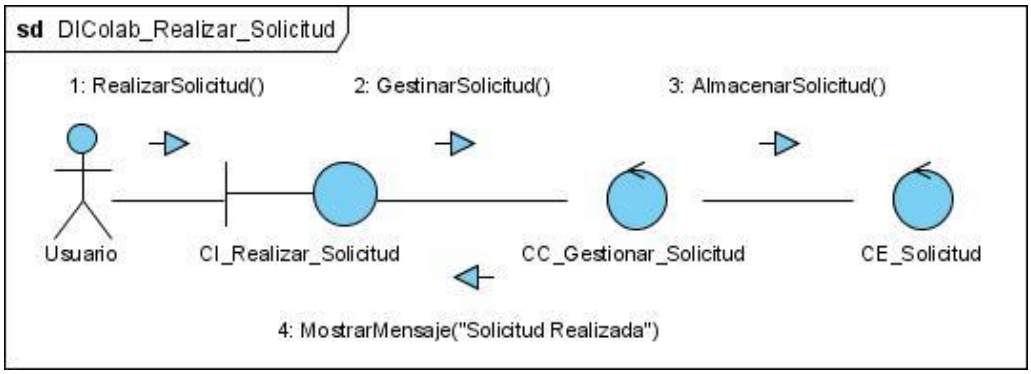


Anexo 6 Diagrama de Colaboración Eliminar Contenido.



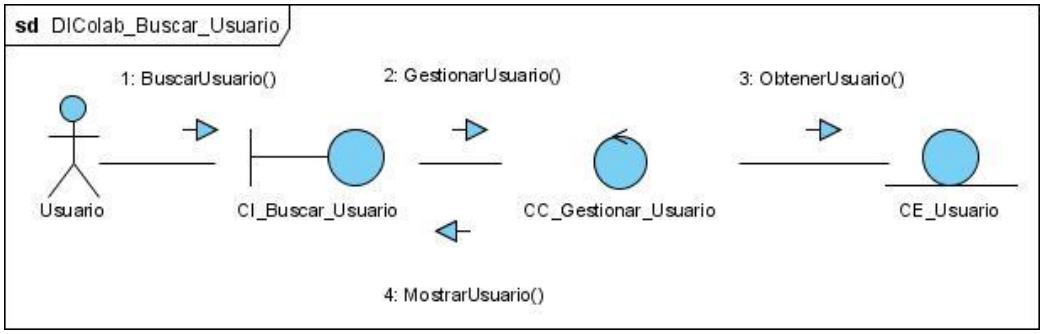
Anexo 7 Diagrama de Colaboración Publicar Contenido.

Caso de Uso Realizar Solicitud.



Anexo 8 Diagrama de Colaboración Realizar Solicitud.

Caso de Uso Buscar Usuario.



Anexo 9 Diagrama de Colaboración Buscar Usuario.

2. __ Lista de Chequeo aprobada por el Grupo de Arquitectura de la Facultad # 3 para incluir componentes en el repositorio.

ASPECTOS A TENER EN CUENTA		**PUNTUACIÓN (Cant/Calidad)**	
1. Características del componente			
a) Lenguaje desarrollo			

b) Dependencias			
c) Descripción del Componente			
d) Versión del componente			
2. Requisitos que debe cumplir			
a) Reutilización			
b) Documentación del componente			
3. Datos adicionales			
a) Autor del componente			
b) Dirección de correo del autor			
Para que un componente pueda ser incluido en el repositorio debe cumplir con todos requisitos que se exponen en este documento			

LISTA DE CHEQUEO

Características del componente

Lenguaje de desarrollo

Se debe definir en qué lenguaje está desarrollado el componente para lograr una mejor organización dentro del repositorio.

- C#
- C++
- Java
- PHP
- Plone
- Python

Dependencias

Es necesario definir con claridad las dependencias necesarias para poder utilizar el componente, como las que se definen a continuación:

- Framework en que fue desarrollado
- Clases de las que depende, o hereda, interfaces que implementa.
- Otras (Especificar cuál)

Descripción del componente

La descripción del componente permite conocer al desarrollador si es factible utilizar el componente además de conocer sus funcionalidades y donde se aplica

- Componentes visuales
- Controladores de conexión a Base de datos, tráfico de datos, y componentes para el trabajo con Base de datos.
- Componentes para sitios web, servicios web
- Otros (especificar)

Versión del componente

Si existen más de una versión del componente debe especificarse que versión es la que se va a agregar al repositorio.

Requisitos que debe cumplir

Reutilización

Es imprescindible que el componente que se vaya a agregar al repositorio sea reutilizable, ya que no cumple objetivo tener almacenados componentes que son específicos de un proyecto y que no se pueden utilizar en otros proyectos.

Documentación

La documentación debe cumplir con los siguientes requisitos

- Explicar bien el funcionamiento del componente
- Contar con la estructura definida por el equipo de arquitectura para documentar los componentes que se encuentra en el sitio de la facultad.

Datos adicionales

Datos del autor

Se debe conocer los siguientes datos del autor

- Nombre y apellidos.
- Grupo
- Proyecto al que pertenece (en caso de no estar en proyecto dejar en blanco)

Dirección de correo del autor

Esta dirección es para poder contactar al autor en caso de que el componente tenga problemas o se necesite realizar alguna consulta sobre el funcionamiento del mismo.

GLOSARIO

ROI: (Return On Investment) o Rendimiento sobre la inversión es la relación existente entre el coste de la publicidad y los beneficios obtenidos de las conversiones (por ejemplo, ventas o clientes potenciales). El ROI indica el valor que obtiene su empresa como resultado del coste que invierte en su campaña publicitaria.

Crisis del software: Término informático acuñado en 1968, en la primera conferencia organizada por la OTAN sobre desarrollo de software, de la cual nació formalmente la rama de la Ingeniería de Software. El término se adjudica a F. L. Bauer, aunque previamente había sido utilizado por Edsger Dijkstra en su obra The Humble Programmer.

IEEE: Corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, científicos de la computación, ingenieros en informática e ingenieros en telecomunicación.

CASE: Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Lenguajes de Dominio Específico: (Domain-Specific Languages DSL) sirven para encapsular el conocimiento sobre un tipo de sistemas, de manera que desarrolladores "no expertos" puedan utilizar ese conocimiento. Debido a que los DSLs están ideados para un tipo específico de sistemas, la generación es más sencilla y puede "embeber" las "mejores prácticas" que conocen los expertos.

Gestión del Conocimiento: del inglés Knowledge Management, es un concepto aplicado en las organizaciones, que pretende transferir el conocimiento y experiencia existente entre sus miembros, de modo que pueda ser utilizado como un recurso disponible para otros en la organización. Usualmente el proceso requiere técnicas para capturar, organizar, almacenar el conocimiento de los trabajadores, para transformarlo en un activo intelectual que aporte beneficios y se pueda compartir.