Universidad de las ciencias informáticas

Propuesta de metodología para el diseño e implantación de repositorios de activos de software reutilizables.

Trabajo de diploma para optar por el título de
Máster en Gestión de Proyectos Informáticos

Autor: Ing. Iliana Pérez Pupo
Tutor: Msc. Michaell González Jorrín

Ciudad de La Habana, Marzo del 2011

Dedicatoria

A mis padres que son la fuente de mi vida, inspiración de mi voluntad y consuelo de mis deseos.
A mis otras tres madres (Gladys, Consuelo y Bárbara) por su apoyo incondicional, sus consejos, preocupaciones y constante cuidado sobre mí.
A mi abuelita del alma (Titica). A toda mi familia (e Iris) por su amor y regocijo.

Agradecimiento

A mi tutor por su dedicación. Al coordinador de la maestría por su asesoría y constante revisión.A la Universidad de las Ciencias Informáticas por influir en mi formación como profesional.

Resumen

La ingeniería de software basado en la reutilización, es una disciplina que nació de la necesidad del desarrollo de nuevos productos a partir de otros existentes. Para su mejor práctica, fueron definidos varios modelos los cuales definen procesos centrados en la reutilización y establecen la necesidad de un repositorio para almacenar los productos generados para su posterior reúso.

Esta tesis presenta una metodología donde se proponen un grupo de procesos que abarcan las actividades relacionadas con el diseño e implantación de repositorio de activos de software para la reutilización, como soporte a la introducción de modelos de desarrollo de software centrados en la reutilización. Estos procesos se basan fundamentalmente en los conceptos del SEI sobre las Líneas de Productos de Software (LPS), los métodos WATCH de Montilva y TWIN de Samentinger y la gestión de proyectos.

La metodología propuesta cumple con principios como ser proactiva, participativa, sistémica, evolutiva, centrada en la calidad y con involucramiento de la alta gerencia. Como parte de la metodología también se propone un modelo para la especificación de los activos de software, tomando como referencia el modelo de documentación de componentes COTS propuesto por Iribarne. Se definen las entidades involucradas en los procesos de reutilización, delimitando sus responsabilidades; también se definen mecanismos de apoyo que ayudan a esclarecer y fortalecer los procesos para la reutilización definidos en la metodología.

Además, dispone de una guía para su diseño e implantación, tomando como referencia para su análisis de factibilidad, los centros de producción de la red de desarrollo de la Universidad de las Ciencias Informáticas (UCI), donde se exponen los resultados de su implantación en función del grado de publicación y reutilización de los activos de software.

Palabras Claves: repositorio de activos de software, software de reutilización

Abstract

Software engineering based on the reuse is a discipline that arose from the need of developing new products from another existing product. For this major practice were defined several models which define processes aimed to reuse and establish the need for a repository to store the products generated for later reuse.

This thesis posit a methodology which proposes a several processes that covering the activities related to the design and implementation of software active repository for reuse, which support the introduction of software development models that focus on reuse. These processes are mainly based on the concepts of the Software Engineering Institute (SEI) on the Software Product Line (SPL), the WATCH method of Montilva, TWIN (Twin Life Cycle) method of Samentinger and project management.

The proposed methodology has foundations such as to be proactive, participative, systematic, evolutionary, focused on quality and involvement of high management. As part of the methodology also proposes a model for the specification of software active, by reference to the documentation model proposed by Iribarne COST components. It defines the entities involved in the reuse process, which defines their responsibilities; also define support mechanisms that help to clarify and strengthen the reuse processes that were defined in the methodology.

Also has a guide to design and implementation of the methodology, using as a reference for feasibility studies, production centers of the development network of the Informatics Sciences University (ISU), which show the results of its implementation in the degree of publication and reuse of software assets.

Keywords: software active repository, reuse software

ÍNDICE

Índice de Figuras

Índice de tablas

Introducción

La creciente demanda de los clientes en el mercado del software para desarrollar disímiles aplicaciones ha conllevado a la evolución de la disciplina Ingeniería del software. Se requiere de altos niveles de producción para poder satisfacer todas las demandas en este mercado cumpliendo con el tiempo de entrega pactado y con la calidad de los productos.

En este sentido, han cobrado fuerza los conceptos de reutilización como una forma más económica de producir, que permite además producir con mayor rapidez y calidad.

Bajo esta situación, surge la disciplina Ingeniería de Software de reutilización o Ingeniería de software basado en componentes. La definición de esta disciplina ha provocado la definición de nuevos procesos, nuevos modelos y la descripción de nuevos entornos que faciliten la reutilización de las piezas o activos del software.

Se sugiere de esta forma potenciar elevados índices de reutilización tanto de los componentes de software como de los recursos humanos que intervienen en su desarrollo cumpliendo las siguientes características:

- Los componentes finales no deben ser modificados fácilmente por los usuarios para evitar interrupciones significativas a los procesos industriales por mala manipulación de los sistemas.
- Se sugiere que estos componentes sean actualizados solo por los especialistas altamente capacitados y los propios proveedores de los sistemas.
- Aunque su aplicación sea posible en diferentes entornos su funcionamiento básico y la secuencia de operación debe ser la misma para todos los casos creando componentes prácticamente idénticos con mínimos detalles de personalización.

La evolución del desarrollo basado en componentes ha dado lugar a que no solo se reutilicen los componentes de código sino que además cualquier parte útil de las soluciones informáticas. Surge en este sentido el concepto de activo como: “producto de software diseñado expresamente para ser utilizado múltiples veces en el desarrollo de

diferentes sistemas o aplicaciones". [1]

En este escenario, la Universidad de las Ciencias Informáticas (UCI), se presenta con una red de producción a través de centros de desarrollo, agrupados según el campo de aplicación del software que desarrollan, por lo que las caracteríristicas de los software de un mismo centro son similares y los entornos de aplicación no varía mucho. Ésto unido a la considerable demanda software en los centros y el número elevado de clientes, se potencia mucho la necesidad de la reutilización no sólo entre los productos generados en un mismo centro sino también entre diferentes centros.

Sin embargo, en algunas instituciones desarrolladoras de componentes, incluyendo la UCI, no siempre se crean condiciones favorables para un entorno de reutilización, debido fundamentalmente a que no se dispone de un repositorio central para la reutilización, ni una adecuada especificación de los activos de software. La introducción de conceptos de reutilización en este entorno es costosa e incluso mal empleada, puede generar pérdidas, pues una adaptación en un componente implica cambios significativos en el resto de los componentes del sistema para lograr la integración, teniéndose que destinar mucho tiempo y recursos en la reimplementación. [2][80]

Entre las causas fundamentales podemos señalar:

- Pobre diseño de una arquitectura accesible, escalable, flexible, portable, lo suficientemente distribuida, con alta disponibilidad y tolerancia a fallos.
- En la mayoría de los casos, existe incompatibilidad entre los componentes debido a que no se estableció a escala global, una tecnología única para su desarrollo, los estilos de código utilizados no fueron revisados para garantizar que todos cumplieran con las mismas sentencias o formatos de codificación.
- La arquitectura de los componentes no era la misma, la forma de comunicación entre los componentes no estaba parametrizada por lo que había que crear mecanismos alternativos para la interpretación de los mensajes o datos enviados.
- Además, se carecía de una infraestructura para el desarrollo, la documentación y almacenamiento de componentes, que potenciara un proceso de reutilización con calidad.

Dentro de este escenario, se propone el desarrollo de este trabajo que pretende

resolver el siguiente problema de investigación:

Problema científico:

La inexistencia de un repositorio de activos reutilizables en la UCI y de procedimientos para su explotación, está afectando el proceso de introducción de modelos de desarrollo de software centrados en la reutilización, enfocado desde: la poca cantidad de activos de software publicados en la red de centros y la deficiente documentación técnica de los activos existentes.

Objeto de estudio:

Modelo para el desarrollo de software centrado en la reutilización.

Campo de acción:

Repositorios de activos de software reutilizables.

Objetivo General:

Desarrollar una metodología para el diseño e implantación de repositorios de activos de software reutilizables como soporte a la introducción de modelos de desarrollo de software centrados en la reutilización.

Objetivos específicos:

- Realizar estudio del estado del arte de los modelos de desarrollo de software y de software basados en la reutilización.
- Definir una propuesta de una metodología para el diseño e implantación de repositorios de software reutilizables.
- Evaluar la propuesta a partir de la aplicación de la metodología en la Red de Centros de la Universidad de las Ciencias Informáticas y como soporte al proceso de introducción de modelos de desarrollo de software centrados en la reutilización.

Hipótesis:

Si se definiera una metodología para el diseño y la implementación de repositorios de activos que sea proactiva, sistémica, evolutiva y participativa; se logrará potenciar la publicación de activos, aumentar la calidad de la documentación técnica de los activos publicados creando un entorno favorable a la introducción de modelos de desarrollo centrados en la reutilización.

Variables dependientes:

- Proceso de introducción de modelos de desarrollo de software centrados en la reutilización de activos de software.

Variables independientes:

- Metodología para el diseño y explotación de repositorios de activos de software reutilizables.
- Sistema de información para la gestión y explotación de repositorios de activos de software.

Tabla 1: Análisis de la variable dependiente.

Variable dependiente	Indicador	Subindicador	Unidad de medida
Proceso de introducción de modelos de desarrollo de software centrados en la reutilización de activos de software.	Evolutivo	Evoluciona en el tiempo	Sí/No
		Sistema de autoaprendizaje	Sí/No
	Sistémico	Integración de los procesos	Sí/No
	Nivel de respuesta	Nivel de respuesta	Proactivo/Reactivo reactivo
	Participativo	Participación de todos los miembros	Sí/No

Tabla 2: Análisis de las variables independientes.

Variable independiente	Indicador	Subindicador	Unidad de medida
Metodología para el diseño y explotación de repositorios de activos de software reutilziables.	Modelo para documentar activos de software	Calidad	Alta, Media, Baja
	Procesos	Sencillos	Sí/No
		Reutilizables en otros dominios de aplicación	Sí/No
Sistema de información para la gestión de repositorios.	Repositorio de activos de software	Facilidad de uso	Sí/No
	Facilidad de almacenamiento	Compresión de la información.	Sí/No

		Facilidad de actualización.	Sí/No
		Nivel de protección	Sí/No
	Facilidad de búsqueda de información	Capacidad de recuperación	Sí/No

Para el desarrollo se este trabajo se combinan diferentes **Métodos y Técnicas**, los fundamentales son:

A nivel teórico

Métodos de análisis-síntesis e inducción-deducción: Para el estudio de las concepciones y conceptos empleados en el campo de desarrollo de software centrado en reutilización, permitiendo la extracción de los elementos más importantes que se relacionan en este campo.

Análisis histórico-lógico: Para conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a la ingeniería basada en reutilización; además de conceptos, términos y vocabularios propios del campo como componente, reusabilidad, COTS, activos, ingeniería de dominio, ingeniería de aplicación, entre otros.

Método de modelación: Para la caracterización de las funcionalidades propias del desarrollo centrado en reutilización.

A nivel empírico

Revisión de documentos: Para el estudio y análisis, partiendo de documentación científica, de los conceptos relacionados con el modelo de desarrollo centrados en la reutilización de activos de software.

Población: Centros de la red de centros de la UCI.

Muestra: 100% de los centros de la red.

Resultados esperados

Entre los resultados que se esperan de este trabajo de investigación están:

- Metodología para el diseño e implantación de repositorios de activos de software reutilizables.
- Repositorio de activos de software de la red de centros de la UCI.

Organización del trabajo

La presente tesis está organizada de la siguiente manera: en total, se ha estructurado (y en ese orden) de tres capítulos, las conclusiones finales del trabajo, las recomendaciones y un listado de las referencias bibliográficas usadas.

El trabajo consta de una introducción, tres capítulos, las conclusiones, las recomendaciones, la bibliografía y los anexos. Los contenidos de los capítulos en forma abreviada son los siguientes:

Capítulo I. Marco conceptual. Realiza una revisión bibliográfica sobre el tema de investigación para el estudio del estado del arte de los modelos de desarrollo de software centrados en la reutilización.

Capítulo II. Marco organizacional. Se presenta la propuesta de metodología para el diseño e implantación de repositorios de activos de software reutilizables. A modo de conclusión parcial, se analizan sus fortalezas y limitaciones.

Capítulo III. Análisis de factibilidad. Se expone la forma de trabajo de la institución donde se aplicó la metodología. Se muestran gráficas que evalúan el grado de aceptación de la propuesta del trabajo de tesis en los miembros de la institución. Como conclusión parcial se hace una evaluación de las variables dependientes e independientes definidas en la tesis.

1. Estudio del estado de arte de los modelos de desarrollo de sistemas de software para la reutilización.

1.1. Conceptos básicos sobre componentes de software

Para iniciar con el análisis del diseño de software para la reutilización, debemos primero conocer qué es reutilización, qué es un componente, qué es un activo y otros conceptos característicos del campo. Para ellos, nos basaremos en los conceptos definidos por algunos estándares y modelos de autores que han especializado sus estudios en esta rama.

Son muy diversos los conceptos de reutilización, de activos y de componentes, pero casi todos coinciden en que “reutilización de software es el proceso de crear sistemas de software a partir de software existente, en lugar de desarrollarlo desde el comienzo” [3][4]. Montilva afirma que la reutilización ha estado basada en oportunidad, es decir: “Los componentes se almacenan en un repositorio a la espera de una oportunidad de reutilización”. [5][6] Analizando un concepto más detallado, podemos citar a Szyperski quien afirma que un componente es una unidad binaria de composición, carente de estado interno, y que está caracterizado por una serie de interfaces que establecen un contrato entre el componente y su entorno. [7][3][8-11]. Por su parte, Sametinger y Brown definen el componente software como “autocontenido, pieza claramente identificable que describe y realiza funciones específicas”. [12][9][6] [13]

Sin embargo el componente no es la única pieza reutilizable de un software. Exite el término activo de software, el cual abarca todos los artefactos que se generan de un software y que puedan ser reutilizados. Montilva lo clasifica como “un producto de software diseñado expresamente para ser utilizado múltiples veces en el desarrollo de diferentes sistemas o aplicaciones”. [1][14] Un activo de software puede ser:

- Un componente de software
- Una especificación de requisitos
- Un modelo de negocios

- Una especificación de diseño
- Un algoritmo
- 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

Un componente de software reutilizable (CSR) es “una pieza [de software] funcional que es liberada independientemente [de otras] y que proporciona acceso a sus servicios a través de sus interfaces”. [15][7]

Entre las características esenciales de un CSR están:

- *Identificable:* Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación. Debe ser clara y consistente de modo que facilite su catalogación y búsqueda en repositorios de componentes.
- *Auto contenido:* Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado. es conveniente que un componente dependa lo menos posible de otros componentes para cumplir su función de forma tal que pueda ser desarrollado, probado, optimizado, utilizado, entendido y modificado individualmente.
- *Reemplazable por otro componente:* Se puede remplazar por nuevas versiones u otro componente que lo remplace y mejore.
- *Accesible solamente a través de su interfaz:* el componente debe exponer al público únicamente el conjunto de operaciones que lo caracteriza (interfaz) y ocultar sus detalles de implementación. Esta característica permite que un componente sea reemplazado por otro que implemente la misma interfaz.
- *Inmutabilidad de sus servicios:* Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí. Estas variaciones no deben afectar la interfaz.
- *Bien documentado:* Un componente debe estar correctamente documentado,

incluyendo sus servicios, para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.

- *Genérico:* Sus servicios debe servir para varias aplicaciones.
- *Independiente de la plataforma*: Existen diversas plataformas de cómputo de uso frecuente (e.g. Windows/Intel, Solaris/Sparc, OSX/PPC, Linux/Intel) y es deseable que un componente pueda ejecutarse en todas ellas. Asimismo, ya que existe una amplia gama de lenguajes de programación y herramientas de desarrollo, es natural que encontremos componentes escritos empleando lenguajes y herramientas de la preferencia del programador, por lo tanto es deseable que dichas preferencias no limiten el uso de los componentes.[16] [17]

La reusabilidad tiene múltiples ventajas, entre ellas están la reducción del tiempo en el mercado, reducción del desarrollo de los productos, el incremento de la calidad del producto, aumento de la satisfacción de los clientes y lograr una alta escala en la productividad. [85]

El desarrollo basado en componentes (DBC) y la ingeniería de software basada en componentes (ISBC) fomentan y facilitan el uso de los mismos para ser ensamblados y reusados sistemáticamente a través de la definición de modelos de procesos que serán analizados posteriormente.

En este mismo entorno y por necesidad del mercado y la industria de reutilización, surgieron las Líneas de Productos de Software (LPS), definidas por algunos autores [18] como un conjunto de soportes lógicos que poseen unas características comunes y que están orientados a las necesidades específicas de una determinada área de negocio.

El objetivo principal de las LPS es “reducir el tiempo, esfuerzo, costo y complejidad de crear y mantener los productos de la línea”. [19][20][21][22]

Los modelos de reutilización están muy relacionados con el modelo de desarrollo de software que se adopte; se discuten a continuación características esenciales de los principales modelos: modelo en cascada, modelo evolutivo, modelo basado en componentes, modelo RAD con el objetivo de identificar el alcance y los límites para la aplicación de este trabajo. En cada modelo tratado por los diferentes autores también se incluyen otros conceptos dentro del campo de la reutilización.

1.2. Análisis de los modelos de desarrollo de software

1.2.1. Modelo lineal secuencial o cascada

Este es el paradigma más antiguo y más extensamente utilizado en la ingeniería del software, se caracteriza por ejecutar secuencialmente los procesos de análisis, diseño, implementación y prueba de forma tal que nunca se comienza un nuevo proceso hasta que se haya concluido completamente el anterior. [15]

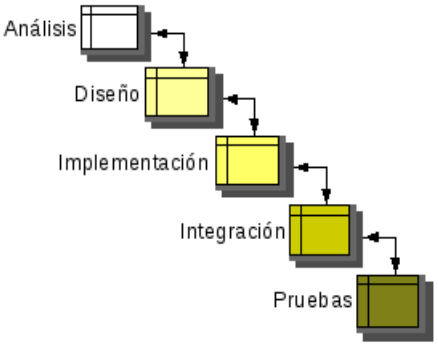


Figura 1: Modelo cascada.

Elementos positivos y limitaciones del modelo cascada:

- Este modelo no promueve fuertemente la reutilización de componentes porque no contempla entre sus fases la selección de posibles componentes para ser usados o la adaptación de soluciones anteriores y componentes al nuevo escenario.
- Exige un orden estrictamente secuencial entre las fases que contrastan con el dinamismo del proceso de desarrollo de software en grandes y medianas soluciones.

1.2.2. Modelo de desarrollo evolutivo

Este tipo de desarrollo permite ir obteniendo gradualmente partes del sistema, ir avanzando en el conocimiento y madurez de los requerimientos e incrementando las funcionalidades del sistema. Dos conceptos básicos de desarrollo de este modelo son los conceptos iterativo e incremental. [23]

Dos formas básicas de implementación de este modelo son el desarrollo basado en prototipos y el desarrollo basado en espiral.

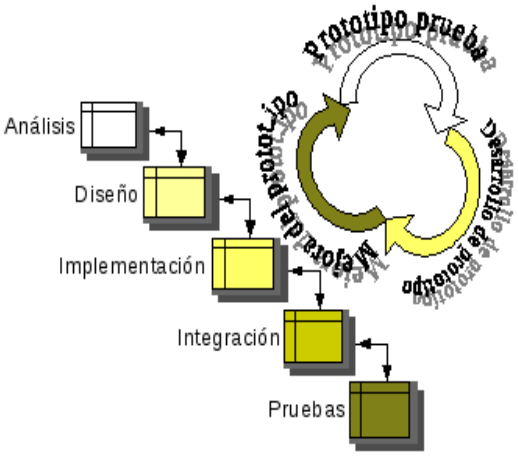


Figura 2: Modelo prototipo.

[24]

Modelo prototipo: Este modelo evolutivo se caracteriza por la creación de prototipos en todas las etapas del desarrollo del software, lo cual favorece a la interacción con el cliente y su aceptación de cómo va quedando el producto. Sin embargo, los continuos cambios durante las intervenciones del cliente, podrían alargar el tiempo de desarrollo, convirtiéndose la ventaja en desventaja. En la figura 2 se muestra un esquema representativo de este tipo de modelo.

Modelo en espiral: Este modelo conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. Durante las primeras iteraciones, las versiones incrementales podrían ser un modelo en papel o prototipos no funcionales. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado. En la siguiente figura se muestra un ejemplo de modelo en espiral donde que contiene seis tareas, características en este tipo de modelo de desarrollo evolutivo de software. [24][15] Ver figura 3.

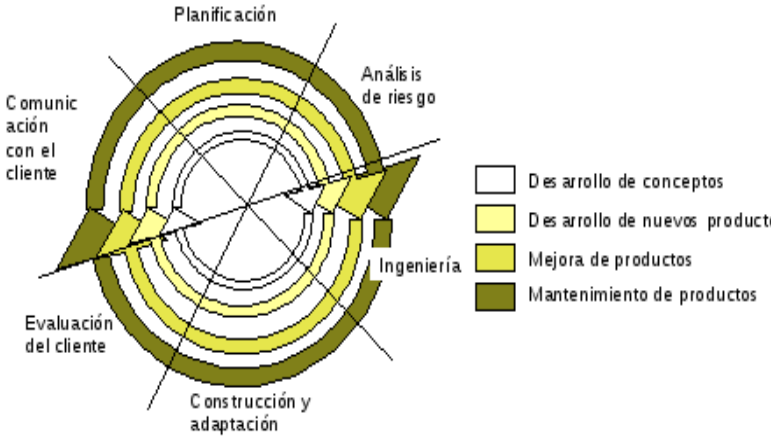


Figura 3: Modelo en espiral.

Elementos positivos y limitaciones del modelo:

- Elemento positivo podemos señalar que permite la detección en etapas tempranas de los problemas más críticos para el desarrollo del software, por lo que disminuye los riesgos.
- La implementación basada en prototipos de este modelo enfoca la reutilización desde la posibilidad de reutilización de los prototipos anteriores en cada una de las subsiguientes etapas.

- Sin embargo la implementación en espiral no se refiere explícitamente al concepto de reutilización. Además, no facilita la integración de aplicaciones que han sido desarrolladas como sistemas independientes.

1.2.3. Modelo basado en herramientas de 4ta generación

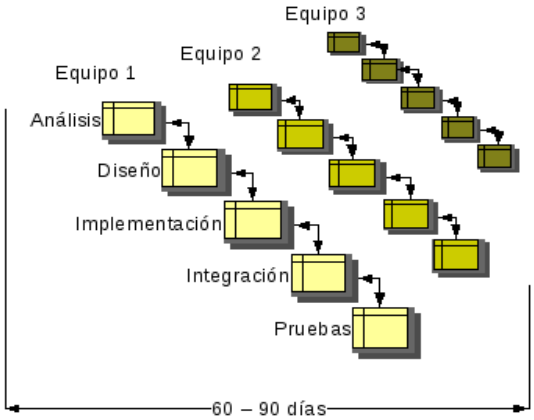


Figura 4: Modelo basado en herramientas de 4ta generación.

El Desarrollo Rápido de Aplicaciones basado en herramientas de 4ta generación (DRA) es una adaptación a alta velocidad del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando una construcción basada en componentes. Es el modelo más próximo al desarrollo de software basado en componentes (DSBC), pues trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o a crear componentes reutilizables (cuando sea necesario).

Elementos positivos y limitaciones del modelo:

En este modelo, la reutilización se concentra en las herramientas de 4ta tecnología que generan soluciones para los usuarios. Básicamente la fuerza de trabajo va dirigida al mejoramiento de esas herramientas, sobre las que se le van adicionando más funcionalidades y mejores servicios, para que las soluciones ofrecidas a los usuarios sean cada vez más óptimas. Y se reutilizan las herramientas en el desarrollo de las soluciones a la medida. [25]

- En este escenario la reutilización está limitada por los escenarios en los que son aplicables las herramientas de cuarta generación de que dispone la organización.
- Por otra parte este modelo de desarrollo requiere un elevado nivel de especialización del personal en el uso y explotación de las herramientas de 4ta generación de que dispone la organización.

1.2.4. Modelo de desarrollo basado en Componentes (DBC)

El desarrollo basado en componentes surgió a partir de la necesidad de reusabilidad, tanto de las experiencias, como de documentaciones, clases, diseños, algoritmos y fragmentos de códigos generados en proyectos anteriores. Y esa necesidad de reusabilidad fue dada por la conveniencia y las ventajas que contribuía a la realización de sistemas software utilizando piezas y experiencias ya probadas, reduciendo el tiempo de producción y los riesgos. Bajo estos principios surge el término de desarrollo de software basado en componentes, del cual varios autores reflejan sus posiciones en este sentido. [26][68][69][71]

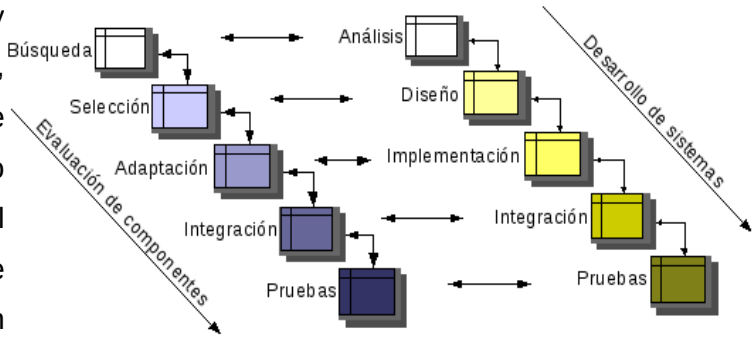


Figura 5: Esquema general de los procesos que caracterizan a los modelos de DBC

1.2.5. Modelo de desarrollo basado en Líneas de Productos de software (LPS)

Una línea de productos de software (LPS) es un modelo de desarrollo de software que se basa en la creación de varios productos similares a partir del «ensamblaje» de componentes pre-fabricados. [5] Otros autores la denominan “portafolio de productos estrechamente relacionados con variaciones en sus características y funciones”. [82]

Las metodologías de desarrollo de líneas de productos de software se enfocan más en la ingeniería de dominio de software de reutilización y del uso de una producción automatizada eliminando virtualmente la ingeniería de aplicación manual de los productos independientes. [67] Constituye un modelo evolucionado que promueve de forma ordanada el desarrollo industrial basado en componentes. [27] [28]

Debido a las LPS, se ha incrementado en número de compañías de alta tecnología en el mercado, fomentando el reuso de todos los artefactos en el ciclo de vida de desarrollo, y así, acortar el tiempo de desarrollo y mantenerse competitivo en el mercado. [84]

Elementos positivos y limitaciones de los modelos basados en reutilización:

- Entrega de productos de software de una manera más rápida, económica y con una mejor calidad.
- Producen mejoras en tiempo de entrega del producto, en costos de ingeniería, en tamaño del portafolio de productos, en la reducción de las tasas de defectos y en la calidad de los productos.
- Requiere que las empresas que lo adopten consideren o conozcan los aspectos conceptuales, tecnológicos, metodológicos, organizativos y gerenciales de las LPS.

1.3. Principales autores que se han referido al desarrollo de software basado en reutilización

Las fases para la reutilización de software son adaptadas e integradas a un modelo de procesos existente como parte de la fase de la ingeniería de software tradicional. [12] Existen muchos trabajos sobre modelos basados en reutilización, tanto de líneas de productos de software como de componentes. En este apartado se analizan algunos enfoques de los principales autores, sus potencialidades y limitantes. En la figura 7 se muestran los procesos que caracterizan a los modelos de desarrollo de software basados en reutilización.

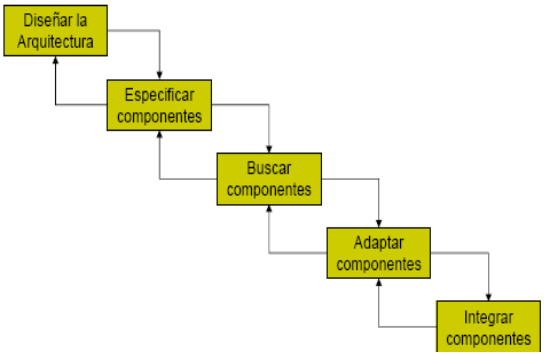


Figura 6: Adaptación de la ingeniería de software para la reutilización.

A continuación se discuten los principales autores del desarrollo basado en componente.

1.3.1. Enfoque Pressman

Según la teoría de Pressman, el modelo de desarrollo basado en componentes es evolutivo por naturaleza, incorporando muchas de las características del modelo en espiral con un enfoque iterativo. Los procesos que define Pressman para el desarrollo de sistemas basados en componentes, se representan en la figura 6, agrupándolos todos en una actividad específica del modelo en espiral. [24]

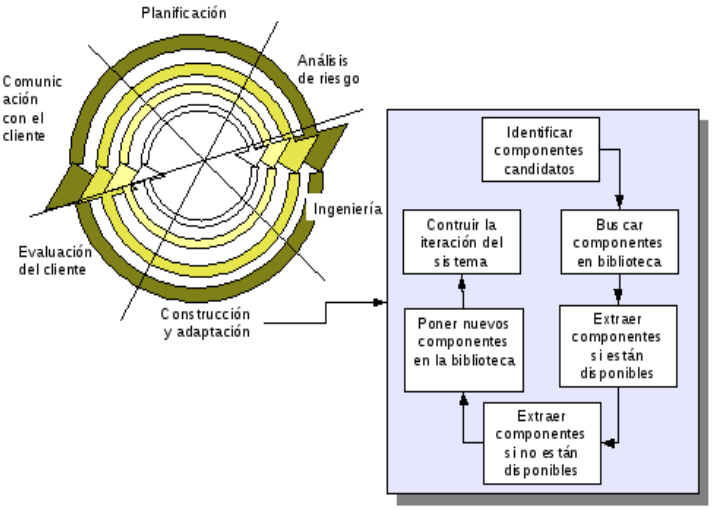


Figura 7: Modelo de software de reutilización de Pressman

Elementos positivos y limitaciones del modelo:

- No incluye explícitamente una etapa de prueba de integración de los componentes en el escenario de su explotación.
- No especifica claramente los elementos a tener en cuenta para caracterizar a los componentes y para garantizar los atributos de calidad requeridos que garanticen su reutilización.

1.3.2. Enfoque Ivica Crnkovic y Magnus Larsson

Otros autores, como Ivica Crnkovic y Magnus Larsson de Inglaterra establecen otro grupo de actividades básicas para el DBC como la búsqueda, selección, adaptación, integración y pruebas del componente, que a su vez, la homologan con las actividades definidas para el desarrollo de sistemas tradicionales en el proceso unificado. En la figura 8 se muestra las actividades homólogas entre el desarrollo de sistemas y desarrollo de sistemas basado en componentes. [29]

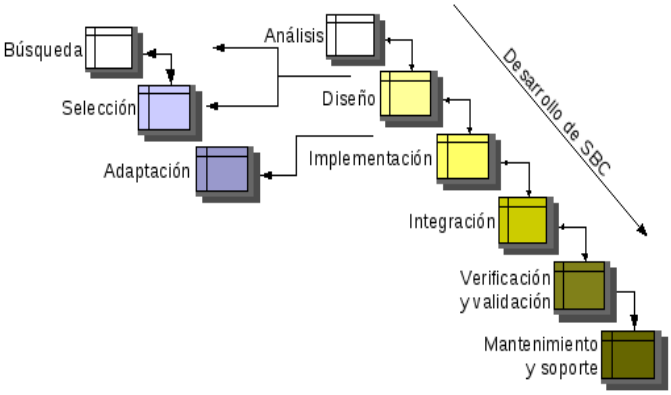


Figura 8: Enfoque de reutilización de Crnkovic y Magnus

Como lo define para el caso de los sistemas basados en componentes (SBC), se inicia con el análisis del sistema, una vez que estén definidos los requerimientos del sistema, se realiza la búsqueda, evaluación y selección de los componentes candidatos según el por ciento de implementación que satisfaga los requisitos definidos para el sistema. Luego se diseña el sistema, adaptando los componentes seleccionados, de forma que estén alineados a la implementación del sistema. Como etapas finales, se integra el sistema, se realizan las pruebas de verificación y validación y una vez que esté el sistema en operación, se le da mantenimiento y soporte.

Elementos positivos y limitaciones del modelo:

- Define como proceso independiente, la búsqueda, selección y adaptación, llegando a detallar cómo realizar estos procesos.
- Incluye repositorio para el almacenamiento de los componentes.
- Establece pautas para la especificación técnica de los componentes.
- No incluye cómo determinar la calidad de los componentes según los criterios que establece.

1.3.3. Enfoque de Iribarne

Iribarne hace un análisis y tratamiento más detallado y profundo de los componentes. No define procesos que intervienen en la reutilización sino que los enriquece con la definición de una estructura para la documentación técnica y un algoritmo de búsqueda (mediador) en un repositorio de componentes. [30]

Elementos positivos y limitaciones del modelo:

- Incluye repositorio para el almacenamiento de los componentes y un algoritmo de búsqueda.
- Establece pautas para la especificación técnica de los componentes.
- No incluye criterios de calidad para el análisis y evaluación de los componentes.

1.3.4. Enfoque de Sommerville

Las principales subactividades dentro del desarrollo basado en componentes software (DBCS) según Sommerville son las que se muestra en el esquema de la figura 9, tomado de su libro sobre ingeniería de software. [31]

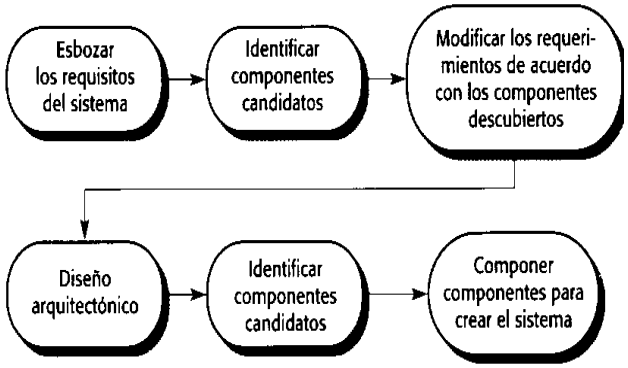


Figura 9: Subactividades dentro del DBCS

Elementos positivos y limitaciones del modelo:

- Usa el concepto “mercado de componentes” para la definición del intercambio entre desarrolladores y clientes de componentes.
- No establece un modelo para la documentación de los componentes, ni identifica que debe existir un repositorio en el mercado de componentes.

1.3.5. Enfoque de Sametinger (modelo TWIN)

El modelo de procesos basado en componentes creado por Sametinger, llamado Ciclo de Vida Paralelo (Twin Life Cycle: TWIN) incorpora explícitamente la reutilización de software en el proceso de desarrollo de aplicaciones. Este modelo considera la

reutilización desde dos perspectivas, tal y como se muestra en la figura 10:

- Desarrollo de software para la reutilización: donde el propósito es producir componentes de software reutilizables. A este proceso se le denomina Ingeniería de Dominio.
- Desarrollo de software con reutilización: cuyo propósito es el desarrollar software reutilizando componentes existentes. Este proceso recibe el nombre de Ingeniería de Aplicación.

La interoperabilidad entre componentes la clasifica en:

- Conexión componente a componente (punto a punto)
- Conexión entre varios componentes (multicast)
- Conexión entre todos los componentes o conexión dinámica (broadcast)

La misma depende del carácter dinámico de los software reusables, es decir, que los componentes puedan ser integrados sin necesitar conocer al resto, permitiendo la composición de los componentes sin ser modificados. [3]

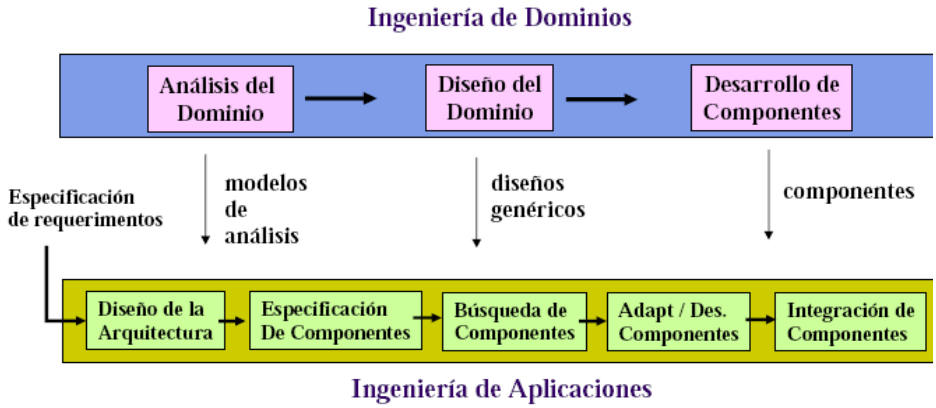


Figura 10: Modelo TWIN

En el primer grupo describe los procesos necesarios para la producción de activos de software reutilizables en un dominio particular. En el segundo describe los procesos necesarios para la producción de software basado en la reutilización.

Del modelo TWIN realizó adaptaciones para agrupar por áreas los procesos según la similitud entre las funcionalidades e incluir un sistema de gestión como proceso rector

entre los definidos tanto en la Ingeniería de Dominio como en la Ingeniería de Aplicación.

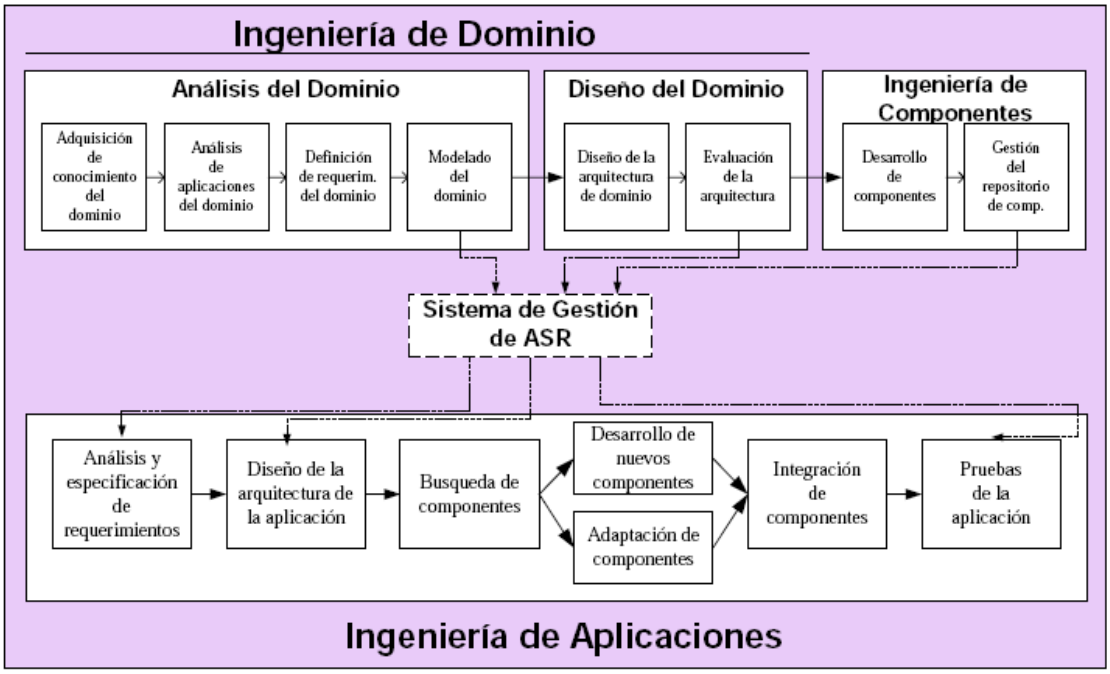


Figura 11: Modelo TWIN extendido

Elementos positivos y limitaciones del modelo

- Incluye nuevos procesos que detalla más el trabajo de reutilización.
- Incluye proceso de Búsqueda de componentes aunque no esté explícito en el modelo una biblioteca o almacén de componentes.
- No define criterios de calidad para la evaluación de los activos ni métodos para calcularlo.
- No garantiza certificación de los componentes.

1.3.6. Enfoque Montilva (Método WATCH)

Según la definición del modelo WATCH, es un método desarrollado expresamente para producir componentes de software reutilizables e integra procesos gerenciales y de desarrollo. [8]

El modelo propuesto en la Universidad de Los Andes (Venezuela) para el desarrollo de aplicaciones empresariales consta de dos componentes metodológicos:

- Ingeniería de Dominio: Desarrollo de Componentes.
- Ingeniería de Aplicaciones: Desarrollo de Aplicaciones Empresariales.

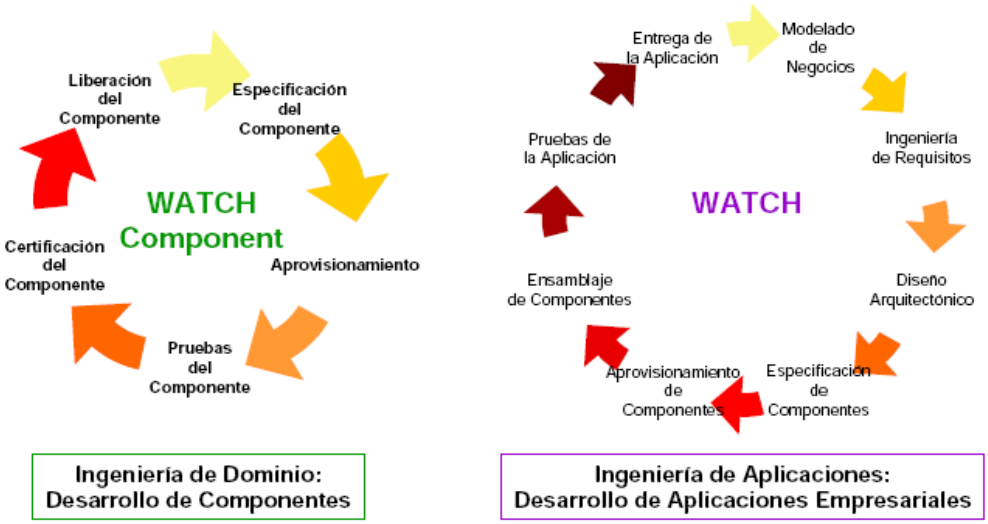


Figura 12: Método WATCH.

A partir de la adaptación de éste, se definen otros dos métodos, el WATCH-Component para la definición de los procesos para el desarrollo de componentes de software reutilizables, y el WATCH-Application como modelo de procesos para el desarrollo de aplicaciones empresariales.

A continuación se realiza un pequeño análisis comparativo de los tres métodos WATCH.

Tabla 3: Comparación entre las adaptaciones del método WATCH.

Método WATCH	Método WATCH-Component	Método WATCH-Application
Incluye los dos ambientes: Ingeniería de dominio e Ingeniería de Aplicación.	Adaptación realizada del método WATCH para aplicarse al desarrollo de componentes reutilizables. Sólo para Ingeniería de dominio.	Adaptación realizada del método WATCH para aplicarse al desarrollo de aplicaciones empresariales. Sólo para Ingeniería de aplicación.
No incluye procesos gerenciales.	Incluye procesos gerenciales.	Incluye procesos gerenciales.
No incluye proceso para la obtención y publicación de componentes de una biblioteca.	No incluye proceso para la publicación de componentes a una biblioteca.	No incluye proceso para la obtención de componentes de una biblioteca.

Elementos positivos y limitaciones del modelo

- Define un repositorio para el almacenamiento y gestión de los componentes.
- No define un proceso para la obtención y publicación de componentes de alguna biblioteca o almacén para su reutilización.
- No describe cuáles criterios podrían tenerse en cuenta para la selección de los componentes.
- No detalla cómo describir los componentes para su documentación.

1.3.7. Enfoque del Instituto del Software Europeo (Modelo PRAISE)

El modelo PRAISE (Product-line Realisation and Assessment in Industrial Settings) definido en el Instituto de Software Europeo (ESI: European Institute Software) en conjunto con Thomson y Bosh propone un esquema parecido al ciclo de vida en cascada añadiéndole una fase de análisis de dominio cuyo resultado es la elaboración de una arquitectura de referencia, y el resultado de la implementación es la producción de componentes reutilizables. [18]

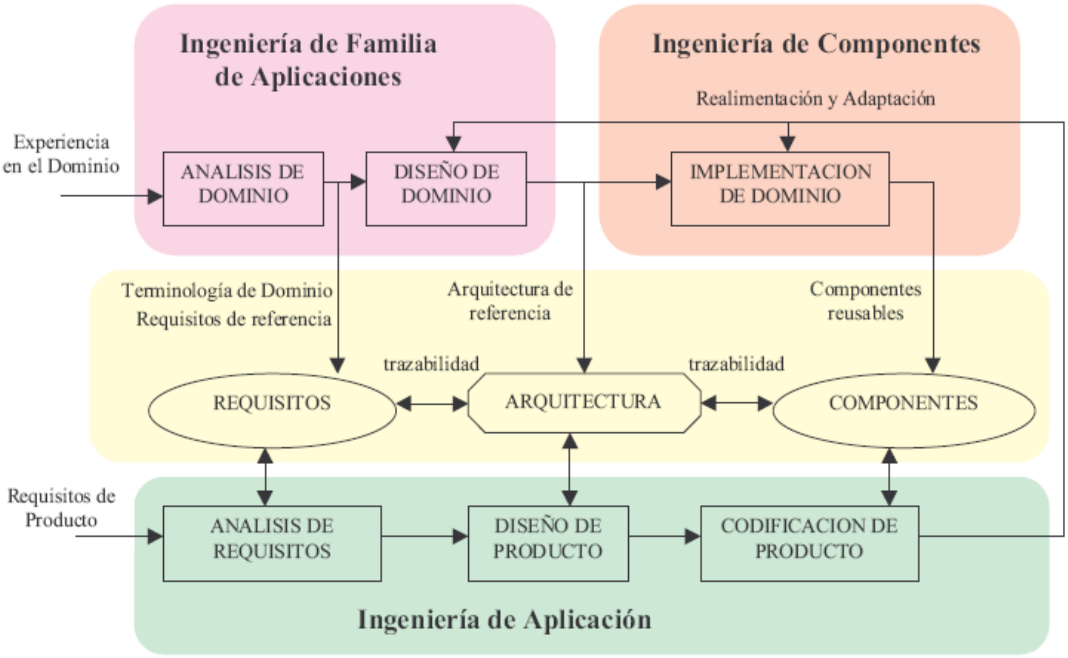


Figura 13: Modelo PRAISE.

Elementos positivos y limitaciones del modelo

- Define procesos para ambos entornos (aplicación y dominio).
- Modelo centrado en la arquitectura.
- No define procesos para la evaluación de los componentes ni del producto.

1.3.8. Enfoque del Instituto de Ingeniería del Software (Modelo SEI)

Una línea de productos de software (LPS) es una forma de crear diferentes productos de software o aplicaciones al reutilizar un conjunto de artefactos centrales de una manera preestablecida. [32] [33]

El Modelo de procesos de LPS desarrollado en el Software Engineering Institute (SEI) describe tres procesos que organizan y engloban toda la actividad que se realiza en virtud de la reutilización, uno para el desarrollo de activos de software para la reutilización, es decir, donde se fabrican las piezas de software que serán reutilizadas en los productos; otro proceso para la integración de los activos para el desarrollo de productos acabado; el tercer y último proceso está dedicado a la gestión de los dos procesos anteriores. [34][35]

Figura 14: Modelo del SEI.

Elementos positivos y limitaciones del modelo

- Tiene bien definidas y limitadas las actividades relacionadas con el desarrollo de activos para la reutilización y el desarrollo de productos basados en la reutilización, especificando entradas y salidas de cada proceso.
- Incluye un proceso de gestión tanto técnica como organizativa para los entornos de ingeniería de dominio y de aplicación.
- La dependencia entre varios activos en una línea de productos podría complicar el ciclo de vida debido a que los múltiples artefactos podrían dificultar el mantenimiento del estado del activo.[36]
- La implantación del paradigma LPS en una empresa es un proceso muy complejo y requiere de una inversión inicial y de la reorganización de la entidad que desea introducirla.[37] [38]

1.3.9. Modelo SPLEP

El Modelo de Evolutionary Software Product Line Engineering Process (SPLEP) permite el desarrollo de las familias de productos de software a partir de varias iteraciones que

se realizan en cada una de las fases del proceso como se muestra en la siguiente figura.

Elementos positivos y limitaciones del modelo

- Fomenta el uso de repositorios para la reutilización.
- Incluye varias iteraciones durante todo el proceso, lo cual facilita elevolución continua del producto.
- No detalla cómo describir los componentes para su documentación ni los criterios de calidad para su evaluación.

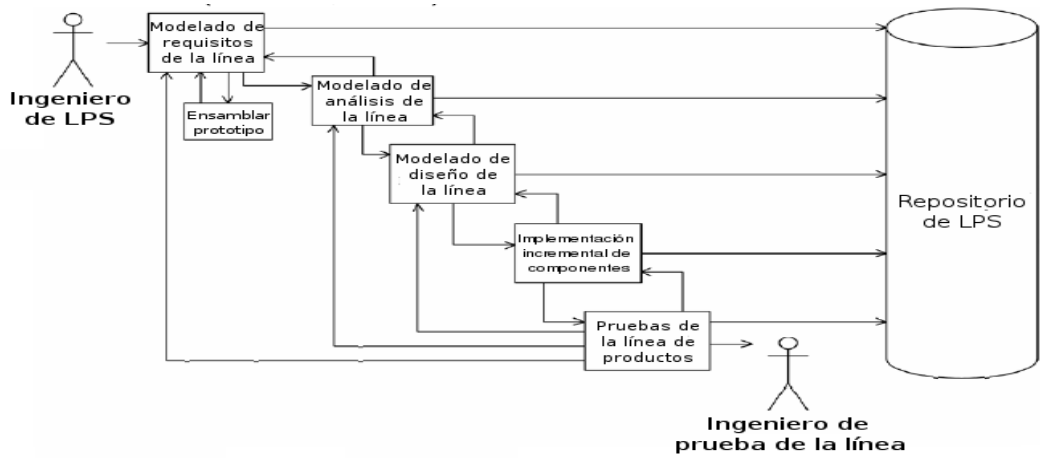


Figura 15: Modelo SPLEP.

1.3.10. Modelo de LPS en centros de producción

Es un modelo que basa sus principios en las líneas de productos de software, la arquitectura de empresas, la mejora continua y las corrientes más actuales de reutilización y desarrollo basado en componentes. Define seis entidades: Ingeniería de Dominio, Ingeniería de Aplicación, Gestión de Recursos los Humanos, Gestión del Conocimiento y de la Configuración, Gestión de la Calidad y Gestión de la Línea. [39]

Elementos positivos y limitaciones del modelo

- Incluye técnicas para la mejora continua y la calidad total del producto.
- Fomenta el uso de repositorio de activos de software para la reutilización.

- No propone un modelo para la especificación técnica de los activos ni métodos para su evaluación dado un conjunto de criterios de calidad.

1.3 Resumen de los estándares y conclusiones parciales

Parten del análisis de los modelos y métodos definidos para el desarrollo de software centrados en la reutilización, podemos arribar a las siguientes aportaciones y limitaciones generales de los modelos:

Aportaciones de los modelos estudiados

- Basándose en los modelos de la ingeniería de software tradicional, definen nuevos procesos para la reutilización de activos tanto en entornos de aplicación como en entornos de dominio.
- Establecen repositorio o biblioteca para el almacenamiento de los activos de software desarrollados.
- En algunos casos, toman en cuenta la especificación de los activos como uno de los factores más importantes para la reutilización, en específico para la selección y evaluación de los activos.
- Se tienen en cuenta criterios de calidad para la selección y evaluación de los activos de software. Entre los fundamentales están la fiabilidad, reemplazabilidad, genérico, independiente a la plataforma.

1.4 Limitaciones de los modelos estudiados

- En la mayoría de los casos no se especifican procesos relacionados con la gestión del repositorio de activos de software.
- No establecen sistemas de recompensas para impulsar la reutilización.
- No definen mecanismos para la seguridad y control de acceso al repositorio.
- No disponen de un código de ética para la solvencia moral y conciencia de control en el escenario de reutilización.
- En la mayoría de los casos no se brinda mucha información de cómo evaluar los criterios de calidad identificados; ni se identifican otros criterios referidos a la

seguridad y disponibilidad de los componentes.

Tabla 4: Relación de modelos estudiados por regiones.

Modelos	**Lugar**
Sommerville [2005]	Europa, Escocia
Iribarne [2003]	Europa, España
Crnkovic y Magnus [2002]	Europa
Pressman	América, EUA
WATCH [Montilva y Barrios 2002]	América, México
TWIN [Sametinger, et al, 1997]	Europa
SEI [Cohen, et al, 1995]	América, EUA
SPLEP [Gooma, 2004]	América, EUA
H. Pestano [2011]	América, Cuba

2. Propuesta de metodología para el diseño y explotación de repositorios de activos de software reutilizable.

2.1. Caracterización de la metodología

Los elementos que componen la metodología son:

- Principios que la caracterizan.
- Entidades que participan en ella.
- Procesos que intervienen en la reutilización.

Para completar y esclarecer la aplicación de los procesos es necesario definir los siguientes complementos:

- Modelo de especificación para los activos de reutilización.
- Criterios de calidad para la evaluación de los activos de reutilización.
- Almacén físico de los activos reutilizables.

Los principios que sigue la metodología son los siguientes:

1. Centrado en la calidad: Cada involucrado en la reutilización de activos son responsables de la calidad de los datos que se almacenan en el repositorio.
2. Proactivo: En la metodología se definen elementos que se anticipan ante acciones que podrían influir negativamente en alguno de los principios y afectar la correcta ejecución de alguno de los procesos.
3. Sistémico: Todos los procesos están integrados entre sí.
4. Evolutivo: Evoluciona en el tiempo y cuenta con un sistema de auto aprendizaje.
5. Participativo: Exige el involucramiento de todos los miembros que participan en un entorno de reutilización de software y hacen uso de un repositorio de activos.
6. Involucramiento de la alta gerencia: Está comprometida con la metodología y la apoya.

2.2. Propuesta de metodología de diseño e implantación de repositorio de activos reutilizables

Pasos para el diseño de un repositorio de activos:

1. Definición de los procesos.
2. Definir qué tipo de activos, qué características y qué comportamiento tienen.
3. En función de los tipos de activos, revisar la información que se registrará para su especificación.
4. En función de los tipos de activos, revisar los factores de calidad que se tendrán en cuenta para su evaluación y medición del nivel de calidad.
5. Diseñar un almacén físico capaz de:
    - recoger como meta-dato la información que se definió para la documentación de los activos,
    - tener capacidad de compresión,
    - tener cierto nivel de seguridad contra los accesos no autorizados.
6. Diseñar una infraestructura de soporte.
7. Identificar los entes que participarán en la red.
8. Definir mecanismos de apoyo y estimulación.

Pasos para la implantación de un repositorio:

1. Identificar a quiénes se les asignará la responsabilidad de los entes definidos y establecer pautas para garantizar su cumplimiento.
2. Crear una infraestructura de soporte.
3. Implementar un almacén físico de activos.
4. Establecer pautas para una correcta especificación de los activos y evaluación para su certificación en función de los intereses de la organización.
5. Establecer e implementar los mecanismos de apoyo y estimulación establecidos.

2.2.1. Entidades que participan en la metodología

- **Entidad proveedora:** Son quienes desarrollan los activos de software para ser reutilizados y deben garantizar:
    - una adecuada especificación de sus activos,

- que sus productos cumplan con un conjunto de factores de calidad (definidos en este trabajo) en función del activo de software a tratar antes de publicarlos al repositorio,
- evaluación de los activos por una entidad evaluadora o certificadora para que les otorgue un acta de certificación de la calidad,
- llenar el meta-datos pedidos por el repositorio para la publicación de los activos
- una vez de tener publicado algún activo, actualizar el meta-datos cada vez que sufra algún cambio.

- **Entidad cliente:** Son los entes que acceden al repositorio para la reutilización de algún activo y deben informar sobre:
  - los activos que están reutilizando,
  - los bug que presenten los activos que reutilizan,
  - los productos que incluyen algún activo que fuera reutilizado,
- **Entidad evaluadora:** es la responsable de la evaluación y certificación de los activos de software y deben:
  - evaluar los activos haciendo uso de los factores de calidad descritos en la propuesta y según el tipo de activo (si es un componente se usaría la mayoría de los factores de calidad),
  - entregar un acta de certificación de la calidad a los activos evaluados con calidad satisfactoria.
- **Entidad publicadora:** es la responsable de la gestión del repositorio y deben:
  - garantizar un repositorio con total disponibilidad (en tiempo de operación y capacidad de almacenamiento) para la publicación de los activos de software de una institución dada.
  - establecer políticas de seguridad y de acceso al repositorio
  - definir un código de ética para los involucrados del repositorio que deberán firmar como garantía de su cumplimiento, de esta modo formalizar legalmente la disciplina en un entorno de reutilización,
  - garantizar la publicación de todos los activos de software de los que haya

recibido la notificación de publicación, el acta de certificación de la calidad y cuyos proveedores del activo hayan firmado el código de ética,

- monitorear o auditar los activos para verificar las inconsistencias que podrían presentar los meta-datos.

2.2.2. Procesos que intervienen en la metodología

En la siguiente figura se muestran los procesos que intervienen en la explotación de repositorios de activos de software reutilizables, las entidades involucradas y los artefactos para el empaquetado del activo que fueron definidos en la metodología que se presenta en este trabajo. En esta sección se describirán los procesos definidos.

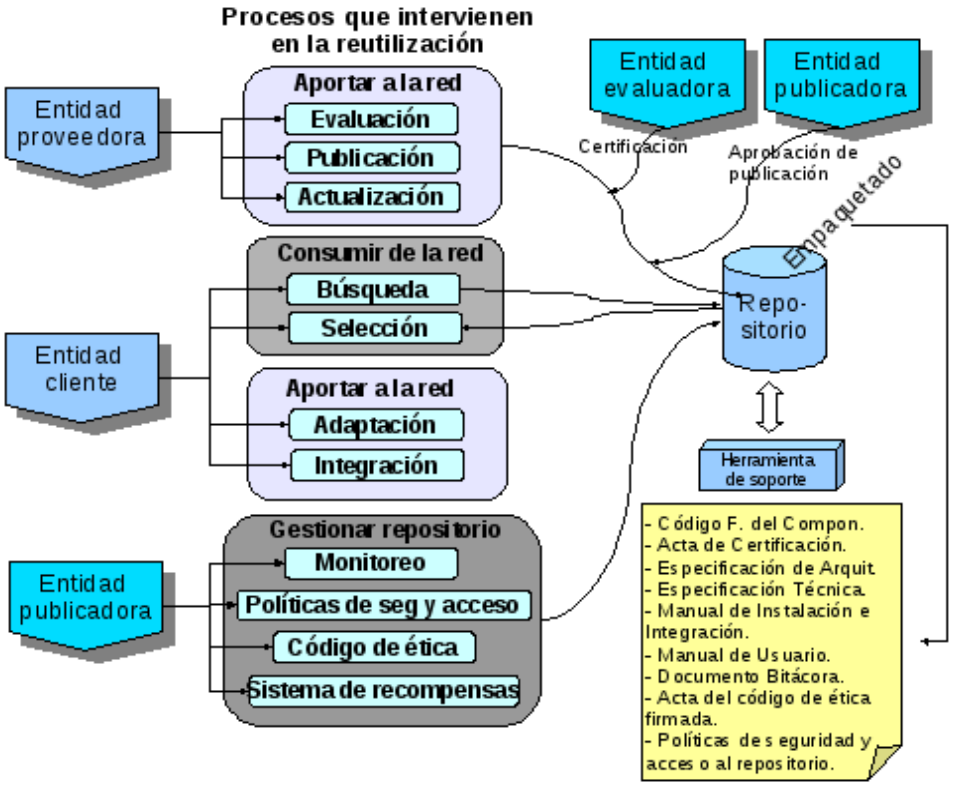


Figura 16: Vista general de los procesos para la reutilización de activos.

2.2.2.1. Proceso 1: Aportar a la red

Proceso mediante el cual el centro de desarrollo y reutilización de componentes comparte a través del repositorio disponible en la red los productos generados, permitiendo la publicación de los activos de software generados para su reutilización. De esta forma se disminuye el esfuerzo de recursos y tiempo en forma de componente reutilizable o servicio, evitando el re-trabajo.

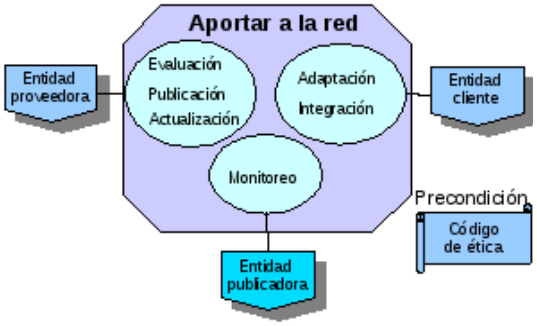


Figura 17: Proceso "aportar a la red".

Como precondición, debe definirse en el entorno de reutilización de software, políticas que regulen la forma de trabajo (código de ética), según los intereses de la institución, y a su vez deben ser conocidas por todas las entidades tanto cliente como proveedora, publicadora y evaluadora.

En este proceso se definen otros subprocesos, tres de ellos son Evaluación, Publicación y Actualización, realizados por la entidad proveedora de activos; también están Adaptación e Integración, que los realiza la entidad cliente y Monitoreo por parte de la entidad publicadora.

Proceso 1.1: Evaluación

Esta actividad es realizada internamente por la entidad desarrolladora como parte de las pruebas que realiza antes de la publicación de un activo de software, según los factores de calidad definidos por la entidad que serán tenidos en cuenta para la evaluación de un activo. [40]

A su vez, es realizado también por algún tercero que se haya contratado como equipo de calidad, de certificación o revisor, encargado de certificar al activo para su publicación y/o venta. La entidad

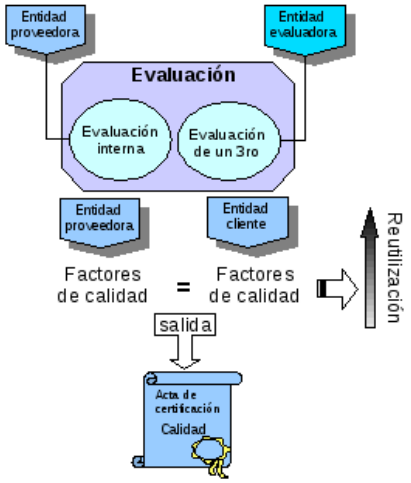


Figura 18: Proceso "Evaluación".

evaluadora, es la responsable de verificar que los datos asociados a la descripción del activo sean reales y que el activo cumple con los factores de calidad.

Si los factores de calidad definidos por el equipo de desarrollo del componente o activo coinciden con los de interés para el cliente, será un elemento más a favor para la reutilización.

Todo este proceso genera como salida un acta de liberación de calidad.

Proceso 1.2: Publicación

Proceso realizado por la entidad desarrolladora que consiste en guardar en algún contenedor o repositorio los activos de software para la reutilización.

Dos elementos importantes que deben garantizar cualquier tipo de contenedor o almacén de información son la fiabilidad y autenticidad de los datos.

Un primer elemento es la fiabilidad de los componentes se pueden garantizar en 3 pasos fundamentales:

- Paso 1: Guardar los componentes en un espacio compartido donde puedan acceder las entidades proveedoras para subir sus activos de reutilización.
- Paso 2: Revisión de los activos por la entidad publicadora para su clasificación en publicables o no publicables, lo cual dependerá de la correctitud y consistencia de la información del meta-datos.
- Paso 3: Publicación de los activos por la entidad publicadora, responsable del repositorio.

Un segundo elemento es la seguridad y protección de los activos, pudiendo ser garantizada a través de algún servicio de autenticación de usuarios, de modo que restrinja el acceso a personas que no estén relacionadas con las entidades tanto publicadora como cliente.

Antes de la publicación de un componente, este debe ser empaquetado junto a un conjunto de documentos que garantice su mejor reutilización. Ellos son:

- Acta de liberación. Documento que valida que el activo fue revisado y evaluado cumpliendo con alguna norma de calidad.
- Especificación de arquitectura del activo o componente. Documento que explica cómo fue diseñada su arquitectura.

- Especificación técnica del activo, que contiene la siguiente información:
    - Descripción
    - meta-datos del componente
    - Funcionalidades y servicios que ofrece
    - Información funcional

Información sintáctica

- Interfaces
- Información semántica
    - Protocolos
    - Patrones
- Información extra-funcional
    - Atributos de calidad
    - Información sobre el ensamblado
    - Información sobre el mercado

- Manual de instalación. Documento que especifica por pasos cuáles métodos se deben invocar de cada interfaz de servicio que ofrece el componente, y cuáles interfaces debe llamar el componente en caso de requerir de alguna interfaz de entrada por dependencias de bibliotecas u otro servicio.
- Manual de integración. Documento que explica los pasos para la integración del componente.
- Manual de usuario. Documentación que genere el equipo de desarrollo para que el usuario puede entender el modo de operación con el componente. Puede incluir conferencias de capacitación (compactado), o un libro electrónico de ayuda o simplemente un documento o pdf de manual de usuario.
- Bitácora de problemas y preguntas más comunes. Este documento es opcional. Es donde se registran las soluciones que el equipo de desarrollo del componente le fue dando a los problemas que se fueron presentando durante la operación con el componente por los diferentes usuarios que lo reutilizan. así mismo, las respuestas a las preguntas más frecuentes.

Antes de ingresar las especificaciones al repositorio, éstas deberán ser revisadas, de forma que se garantice:

- completitud de todos los campos
- corrección de redundancia
- corrección de errores gramaticales y ortográficos
- correspondencia entre la información del documento con la del componente.

Proceso 1.3: Actualización

Es responsabilidad de la entidad publicadora el correcto cumplimiento de este proceso. Es quien debe actualizar los meta datos de los componentes que tenga publicados en el repositorio, es decir, alguna variación en las características técnicas y/o generales de los componentes deberá registrarla, como el versionado, los entornos de sus clientes donde ha sido integrado, las funcionalidades agregadas o eliminadas, entre otros datos. Sólo contando como respaldo la correctitud del meta dato del componente, queda evidenciado su grado de madurez.

Proceso 1.4: Adaptación

Una vez que el cliente seleccione los componentes a reutilizar, procede a la adaptación y extensión que siempre hay que realizar para particularizar el componente, independientemente del producto donde se va a integrar. Por ejemplo, inicializar los valores de los parámetros, especificar pantallas de E/S o formato de los informes, configurar protocolos de comunicación, etc. Incluye además, modificación de la arquitectura en caso de ser necesario para ajustarla a la arquitectura del sistema a integrar. Esta fase tiene implícito las pruebas luego de la adaptación.

Proceso 1.5: Integración

Una vez identificado el o los activos a reutilizar, deben ser examinados, probados operacionalmente e integrados. [41] [42]

- Examinar el activo: revisar por parte del cliente (quien reutilizará el activo) que no existan inconsistencias en los datos del activo.
- Probar operacionalmente el activo: que opere correctamente en tiempo de ejecución y cumpliendo con lo descrito en su especificación.

- Proceso final, donde se desarrolla el código de Integración (Glue Code), que es el código nuevo que hay que desarrollar y probar, necesario para integrar los componentes en un sistema. [43]

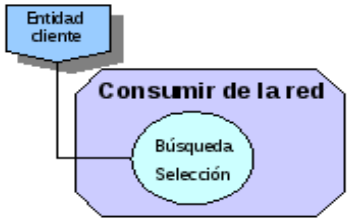


Figura 19: Proceso "aportar a la red".

2.2.2.2. Proceso 2: Consumir de la red

Proceso que permite, después de la búsqueda, el consumo o reutilización de activos del repositorio. Para este se cumple la misma precondición que el proceso de aporte a la red.

El proceso de consumo queda evidenciado en:

- los meta-datos de los proyectos en ejecución, al especificarse el nombre del activo en reutilización,
- las actas de Liberación y Aceptación del software, al hacer mención de los activos que fueron integrados para generar un producto final determinado,
- y en los registros automáticos de los servicios brindados a la red.

En este proceso se definen como subprocesos Búsqueda y Selección, ambos realizados por la entidad cliente.

Proceso 2.1: Búsqueda

La búsqueda es una de las técnicas principales en el desarrollo de software centrados en al reutilización para la localización de activos en el repositorio. [44]

Es realizado por la entidad cliente, la búsqueda de activos consiste en revisar entre los repositorios disponibles para la reutilización, cuál o cuáles se ajustan más a las especificaciones del entorno donde será integrado, es decir, cuál satisface los requisitos impuestos tanto por el cliente como por la arquitectura de la aplicación. Para ello, el cliente se auxilia de los meta-datos registrados por cada activo, donde podrá encontrar un resumen o descripción de las funcionalidades o servicios que ofrece un

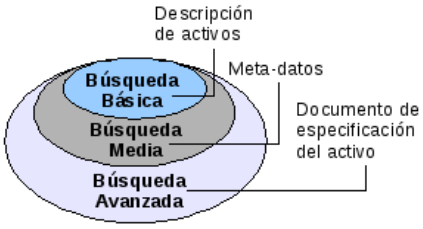


Figura 20: Niveles de búsqueda.

activo, o más específicamente, un componente y la parte más significativa de la información técnica del mismo. Generalmente se hace a través de un mediador (programa de búsqueda de componentes en un repositorio) o algún otro algoritmo de búsqueda que implemente el proveedor de activos para ofrecer como servicio agregado, el de “búsqueda de activos”.

Pueden ser clasificados 3 niveles de búsqueda:

- Básica: se revisa sólo la descripción o síntesis registrada por activos.
- Media: se revisa el meta-dato asociado al activo, además de su descripción.
- Avanzada: se revisa el documento de especificación técnica del activo, donde se incluyen los valores del metadado.

Proceso 2.2: Selección

En esta actividad, el cliente selecciona los activos que se ajustan a un conjunto de especificaciones determinadas. Incluye los siguientes pasos: [40][43]

- Paso 1: Pruebas técnicas. Sirven para verificar el cumplimiento idóneo de las funcionalidades y buena compatibilidad entre bibliotecas de implementación, plataformas de desarrollo y sistemas operativos usados por el proveedor para desarrollar el activo o componente y los que usará el cliente para adaptar e integrar el componente a su sistema. Esto incluye también un análisis de la arquitectura de ambas partes, si la arquitectura definida por el proveedor es muy diferente o difícil de reajustar, el costo de adaptación del activo o un componente en sí será alto.
- Paso 2: Pruebas de calidad. Para conocer el grado de satisfacción del activo con los requisitos de calidad de interés para el cliente. Según el grado de satisfacción de la evaluación, son seleccionados para ser integrados en el sistema cliente.

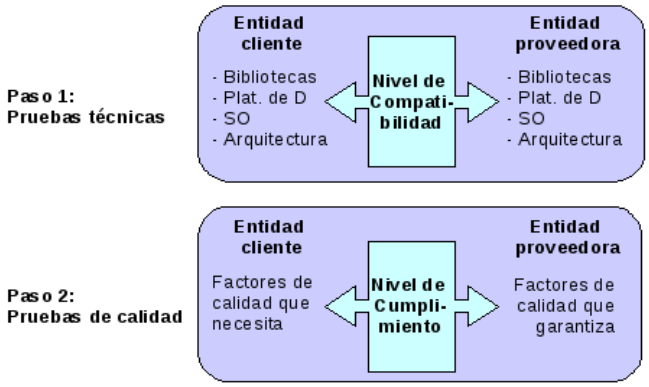


Figura 21: Características del subproceso "selección de componentes".

2.2.2.3. Proceso 3: Gestionar repositorio

El principal objetivo de los sistemas de reutilización es la creación y clasificación de componentes reusables y recuperarlos desde un repositorio. [77-79]

En la metodología de reutilización que se propone, la gestión del repositorio es un proceso que realiza la entidad publicadora para garantizar por un lado el adecuado uso del entorno de reutilización, respaldado en un código de ética y en pautas para la seguridad y el control de acceso al repositorio; y por otro lado, establecer pautas de estimulación o recompensas en virtud de aumentar el grado de reutilización, favoreciendo a los más contribuyentes y agrupando a los menos en la lista de *aislados* del entorno de reutilización.

El término “biblioteca de activos” o “repositorio de activos” se refiere al sistema de información para guardar y gestionar tanto los activos de de reutilización como la información que se almacenen de ellos, tales como: [45-49][76]

- la descripción general de cada activo
- el meta-dato con algunos detalles técnicos
- y el empaquetado que se almacena junto al activo, el cual agrupa un conjunto de documentos que se detalla más adelante.

Proceso 3.1: Definir Políticas de seguridad y acceso al repositorio

La entidad publicadora debe garantizar ciertos niveles de confidencialidad, es decir,

establecer ciertas políticas que disminuyan la intromisión de personal no autorizado o terceros al sistema.

Para ello, es necesario establecer políticas y restricciones que limiten el acceso al repositorio de componentes sólo al personal autorizado. Además, para la seguridad de la información registrada en el repositorio, se debe establecer un sistema de redundancia tanto de datos como de red.

Proceso 3.2: Definir código de ética

Es necesario también establecer pautas para la disciplina y compromiso entre los entes participantes en un entorno de reutilización de activos de software. Para ello, es conveniente definir en un código de ética, los estatutos éticos y morales de la institución, y la firma como constancia del compromiso que tienen las entidades participantes de cumplir con el mismo.

Proceso 3.3: Diseñar sistema de Monitoreo

Este subproceso consiste en revisar la información registrada en los meta-datos para detectar incoherencias, identificar quiénes son las entidades que mayor participación tienen en el entorno de reutilización y quiénes menos, cuál o cuáles son los activos de software más reutilizados. En este proceso se realizan dos acciones principales.

Acción 1:

Dentro del monitoreo, se debe establecer cierta comunicación entre el vendedor y el comprador, o entre la entidad proveedora y la entidad consumidora, comprendido en tres pasos fundamentales:

- Necesidad de reutilización por parte de la entidad cliente.
- Comunicación entre el vendedor y el comprador, o entre la entidad proveedora y la entidad cliente.
- Registrar opiniones positivas o negativas por parte de la entidad cliente sobre el intercambio con la entidad proveedora.

  Esta calificación estará relacionada no a la calidad o características del activo, sino al grado de cooperación del proveedor, es decir, soporte satisfactorio en caso de haberse conveniado, apoyo y capacitación en caso de haberse

solicitado u otro tipo de cooperación.

Según la cantidad de opiniones positivas y negativas que tenga una entidad publicadora de componentes, así será su calificación.

Acción 2:

La segunda acción a realizar es la esencia del proceso de monitoreo, que no es más que mantener un control de los activos reutilizados, de quiénes lo reutilizan y de los entornos donde han sido aplicados. Al final del proceso, se obtendrá como salida de esta acción un conjunto de reportes de ranking que clasifican a las entidades clientes y desarrolladoras de activos reutilizables según su actividad en el entorno de reutilización:

- Componentes más reutilizados
- Entidades que más aportan componentes a la red sin consumir.
  - → Alto índice de aporte y bajo índice de reutilización.
- Entidades que más consumen componentes de la red sin aportar.
  - → Alto índice de reutilización y bajo índice de aporte.
- Entidades que más aportan componentes a la red y a su vez, tienen alto índice de reutilización.
  - → Alto índice de reutilización y alto índice de aporte.
- Entidades que menos aportan componentes a la red y a su vez, tienen más bajo índice de reutilización.
  - → Bajo índice de reutilización y bajo índice de aporte.

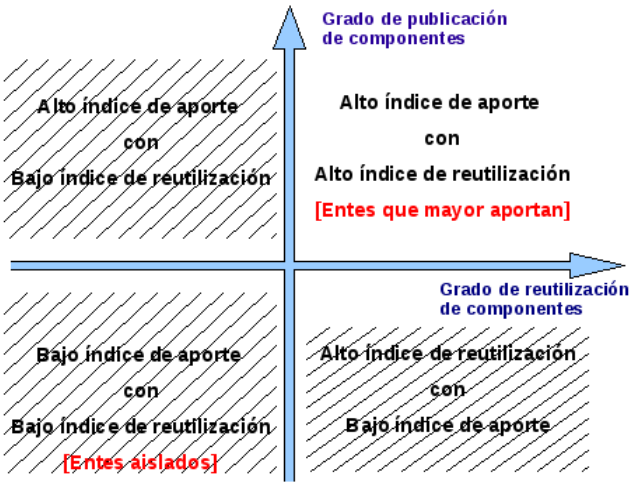


Figura 22: Clasificación según la actividad en el entorno de reutilización.

Para calcular el ranking de activos más reutilizados se definió un escalafón, para el cual se determinaron dos premisas, una nomenclatura y un procedimiento para ser calculado. Se describen a continuación.

Premisas

- Premisa 1: El cálculo supone que en el proceso del cálculo de la calidad de un componente influye la selección que se haya hecho sobre el componente, en específico sobre otros componentes de su mismo tipo simulando un enfrentamiento entre los componentes y su aplicabilidad en diferentes situaciones.
- Premisa 2: Se supone que siempre se reevaluarán los componentes en enfrentamientos dos a dos, y se efectuarán estas evaluaciones o enfrentamientos en el proceso de selección de los componentes que se ejecuta durante el proceso de diseño de la arquitectura de los sistemas y cuando se escojan los componentes a utilizar.

Nomenclatura

- PA (comp) : Puntaje Actual del componente “comp” antes del enfrentamiento.
- Nuevo PA (comp): Nuevo puntaje para el componente “comp” luego del enfrentamiento.
- RE (comp): Resultado del Enfrentamiento para el componente “comp”.

Procedimiento para el cálculo del índice de calidad de un componente:

- Paso 1: Determinación del valor del mérito de vencer en el enfrentamiento “MeritoVencer”
  - → Calcular Diferencia(Comp1, Comp2) = | PA(Comp1) – PA(Comp2) |
  - → Calcular MeritoVencer = Diferencia(Comp1, Comp2)/DiferenciaMaxima
    Donde diferencia máxima se corresponde con el mayor valor PA de todos los componentes del repositorio.
- Paso 2: Informar el resultado del enfrentamiento RE para cada uno de los componentes enfrentados
  - → RE (comp) = 1 si venció ó 0 si perdió.
- Paso 3: Cálculo del nuevo PA
  - → NuevoPA(comp) = PA(comp) + RE(comp)*MeritoVencer

Aquellas entidades proveedoras de activos que resulten tener un alto aporte a la red, serán estimuladas según las políticas de un sistema de recompensas establecido

previamente por la entidad responsable del repositorio de activos de software, en virtud de estimular la publicación de activos en el repositorio y garantizar su intercambio al entorno de reutilización establecido en la organización.

Proceso 3.4: Establecer sistema de recompensas

Esta es otra de las acciones que debe realizar la entidad publicadora dentro del procesos “Gestión del repositorio” para estimular un mejor intercambio en un entorno de reutilización de activos de software, sería apropiado establecer un sistema de recompensas que como consecuencia de la propia estimulación, aumente la publicación de activos y componentes y a su vez fortalece la reutilización.

Este mecanismos actúan como herramientas de control para clasificar los entes participantes en el entorno de reutilización en los diferentes niveles de la red, ya analizados en el puntos anterior, que son:

- entes destacados
- entes de bajo índice de reutilización
- entes de bajo índice de aporte
- entes aislados (ver figura 22)

En la siguiente figura se representa un resumen de los conceptos y forma de trabajo del proceso Gestión de repositorio.

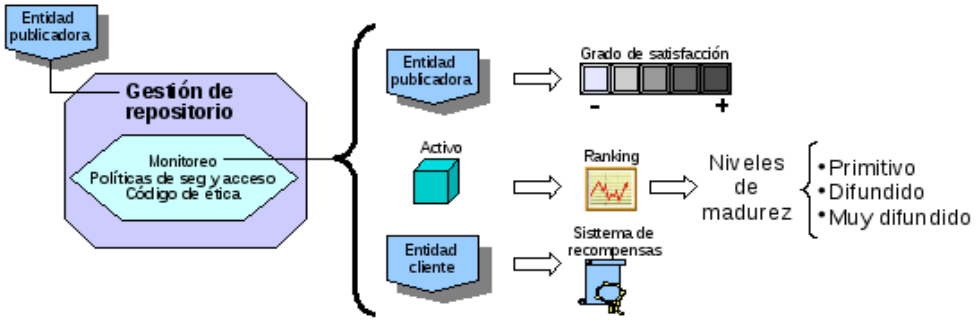


Figura 23: Proceso "Gestión de repositorio de activos".

2.2.3. Especificación de activos de software

La especificación de los activos de software es uno de los elementos más importantes para el desarrollo de software basado en la reutilización. Sin ella, no se podría lograr el

entendimiento entre la entidad proveedora y la cliente que solicita reutilizar un activo o componente; tampoco se podría realizar una correcta selección y evaluación del activo a reutilizar, pues carecería de argumentos.

Iribarne afirma que la especificación de componentes es conocida cuando un actor es capaz de conocer la estructura interna de la especificación (su tipo de información), independientemente de su contenido (la información). [30]

La documentación o especificación de un activo puede ser de diferentes tipos. Según la definición recogida en el libro [50] está la especificación funcional, operacional, de calidad y de diseño; y en la tesis doctoral [45][30] se definen como documentación funcional, extra-funcional, de empaquetamiento y de mercado. En la propuesta de este trabajo se definen sólo dos tipos de especificaciones: la funcional, donde se incluyen documentación sintáctica y semántica; y la extra-funcional, donde se agrupan otros elementos contemplados en las bibliografías analizadas, como información del mercado, el repositorio, del ensamblado y de calidad.

La especificación de los activos ayuda a la persona encargada del ensamblaje o al comprador a conocer cuál de los activos satisface las necesidades para su sistema y hasta qué punto se le podría dificultar o no.

En ella se abarcó toda la información que se puede recoger de un activo de software, incluyendo las de un componente comercial, que es el activo del que mayor cantidad de datos se recoge. Es por esto que la especificación técnica estará más orientada a la documentación de un componente COTS.

2.2.3.1. Descripción funcional

Esta sección está relacionada con todas las características técnicas de implementación que definen el comportamiento y modo de comunicación del activo. La descripción funcional, se clasifica en dos tipos, información sintáctica e información semántica. Los elementos que se tienen en cuenta para la documentación de esta sección son los siguientes:

- Interfaces
  - → nombre
  - → tipo: si es interfaz de:

- entrada (interfaz que provee servicios de otro componente) o de salida (interfaz que brinda el o los servicios que provee el componente)
- editora (si envía eventos) o suscriptora (si recibe eventos)

→ descripción: breve explicación del o los servicios que brinda la interfaz

→ atributos: nombrar los atributos declarados en la interfaz

→ métodos: explicar los funcionalidades que implementa cada método

→ eventos: notificación que realiza la interfaz de un componente para informar cuando sucede algo relevante.

- Patrones
  - → nombre
  - → descripción: que ayude a comprender el tipo de patrón aplicado y qué problema resuelve
  - → clases
    - nombre: especificar los nombres de las clases que intervienen en el patrón
    - descripción: describir el objetivo de cada clase
- Protocolo
  - → nombre: del protocolo de comunicación el componente en caso que sea necesario especificarlo
  - → capa: capa de red del modelo TCP/IP por la que se comunica el componente
- Contracciones
  - → nombre: de la contracción, ya sea precondición o poscondición
  - → descripción: describir la necesidad de la contracción, las condiciones que deben estar pre-establecidas (en caso de referirse a una precondición) y/o las condiciones en la que debe quedar el sistema una vez integrado el componente (en caso de referirse a una poscondición).

2.2.3.2. Descripción extra-funcional

La información extra-funcional es la segunda parte de la documentación de un activo. En ella se incluyen datos que están más relacionados a los activos del tipo componente comercial como información del mercado, [70] que se incluyó con la intención de

abarcar toda la información posible para todos los tipos de activo. Esta sección está relacionada con la calidad del o los servicios ofrecidos por un activo, con la información del mercado del activo, del ensamblado y con los factores de calidad, los cuales ya fueron explicados en el epígrafe anterior.

Esta sección en la documentación de un activo reutilizable es muy importante para su análisis, evaluación y selección. Incluso, es considerado más importante que la información funcional, sobre todo a la hora de elección entre más activos que tengan las mismas características funcionales y se desea elegir cuál se apropia más al entorno.

En estos casos, que son muy frecuentes, es seleccionado el activo que más cumpla con los estándares y normas de calidad, es decir, los que tengan mayor puntuación a favor en cada uno de los atributos de calidad. También los de mayor puntaje en instalaciones positivas son los mejores candidatos para la reutilización.

A continuación se irán explicando por secciones (según el tipo de información) los datos que se recogen para la descripción extra-funcional de un activo.

Información del ensamblado

Esta sección está relacionada con el ensamblado o empaquetado de un activo, dígase componente. Los elementos que se tienen en cuenta para la documentación de esta sección son los siguientes:

- Sistema Operativo (SO): nombre del SO sobre el que se desarrolló, incluyendo la distribución en caso de ser de la familia GNU/Linux y la versión de la distribución.
- Lenguaje: lenguaje usado para la implementación del activo.
- Tipo de código: extensión o extensiones del código, según el lenguaje en el que se implementó el activo.
- Fichero: nombre del fichero contenedor de todo el código fuente.
- Plataforma: nombre del IDE de desarrollo sobre el que se desarrollo.
- Biblioteca: nombre de las bibliotecas usadas ya sean gráficas como de implementación, y la versión.
- Framework: especificar nombre en caso de haberse usado algún framework.
- Gestor de base de datos (BD): nombre y versión del gestor de BD en caso de

haberse utilizados.

- Procesador: nombre del procesador de la máquina sobre la que se desarrolló el activo en caso de ser necesario porque la complejidad lo requiera.
- Runtime: en caso de ser necesario especificar el entorno de ejecución, decir nombre y versión del runtime sobre el que se compiló el activo.

Si queremos integrar dos componentes, ellos deben compartir la misma plataforma de ejecución, o pueden ser diferentes pero la misma plataforma de composición. Esto es importante para que puedan ser compilados al integrarse.

Información de compatibilidad

Es donde se debe registrar las especificaciones y acotaciones tanto de implementación como de ejecución del activo. La información que incluye es:

- SO: nombre del o los SO compatibles, en caso de ser necesario, especificar el mínimo de versiones hasta donde es posible usar.
- Biblioteca: nombre del o las bibliotecas gráficas o de implementación que sean posible usar y no entren en conflicto con el código del activo.
- Gestor BD: nombre y versión del o los gestores compatibles con el que soporta el activo.
- Procesador: nombre del procesador que deben tener las computadoras sobre las que correrán el activo en caso de ser un procesador muy específico debido a la complejidad del activo.
- Runtime: en caso de requerir un runtime específico para la ejecución del activo, especificar su nombre y la versión.

Información de calidad

En ella se especifican los factores de calidad establecidos por la entidad proveedora para la evaluación del activo. Como información especifica:

- Criterios de calidad: se nombran los criterios de calidad que satisfacen el activo. Esas características pueden ser funcionalidad, fiabilidad, usabilidad, mantenibilidad y portabilidad.
  - → Subcriterios: para cada criterio de calidad se corresponden un grupo de

subcriterios.

- Atributo: para cada subcaracterística, se nombra su atributo y el valor que tomó durante la evaluación del activo.

Documentación del mercado

En otras bibliografías, esta sección la denominan documentación de marketing o de mercadotecnia. En ella se registra:

- Descripción: Breve descripción de la posición que tiene el activo en el mercado, el por ciento de clientes interesados y por ciento de los que se les podría ofertar.
- Fabricante: datos necesarios para contactar con el fabricante de un activo como el nombre de la compañía (entidad desarrolladora), dirección física donde laboran, teléfono y dirección del correo electrónico.
- Soporte: datos necesarios para contactar con el personal responsable del soporte del activo, que no necesariamente es el mismo que el fabricante. De él, registrar teléfono y correo.
- Precio: precio que tiene el activo en el mercado.
- Licencia: en caso de que el activo sea dependiente de licencias, nombrarlas. Especificar también la fecha de expiración de la o las licencias.
- Certificado: nombre del certificado asignado al activo como garantía de haber sido certificado por una entidad evaluadora. Se recomienda además, especificar el nombre de la entidad evaluadora o certificadora del activo.
- Expiración: fecha de expiración del uso del activo en caso de haberse especificado en algún contrato.
- URL: en caso de contar con un sitio web, especificar la dirección donde se publica la venta o descarga del activo.
- Notoriedad: grado de éxitos que ha tenido el activo desde la primera vez que fue reutilizado. Esta información incluye cantidad de
  - → instalaciones concretadas y canceladas,
  - → calificaciones positivas, negativas y neutrales.

Información de repositorio

Los datos de esta sección deben ser almacenados por el propio repositorio e incluye:

- Id, fecha, hora, versión almacenada del activo, usuario y correo de quien la registró.

Los repositorios son almacenes o contenedores de algún tipo de información. Son bases de datos especializada que permiten una rápida recuperación y el mantenimiento de los componentes. Su objetivo fundamental es asegurar el almacenamiento y disponibilidad de los componentes para permitir su reutilización.

Estas herramientas, por la naturaleza de su función, deben garantizar la concentración de un gran volumen de datos (en este caso, componentes) y a su vez, la seguridad y protección de los mismos.

Para garantizar una correcta reutilización de componentes en una red, los repositorios deben cumplir con ciertas características:

- El repositorio debería almacenar información a base de feedbacks constantes de los resultados de los procesos de selección, integración y mantenimiento.[51]
- Se debe definir un rol en el equipo de la entidad publicadora responsable de mantener el repositorio.

Para el caso de la que se propone en el trabajo, es una herramienta que permite realizar clasificaciones de activos reutilizables y hacer búsquedas por filtros. Al ser una concentración de activos debe ser asegurado y protegido. La seguridad está garantizada a través del servicio de autenticación agregado a la herramienta que soporta el repositorio; y la fiabilidad, por medio de un código que identifica un componente como único entre los que están registrados en el repositorio.

Nomenclatura:

El nombre de los activos es recomendable que cumplan con una misma nomenclatura, para una fácil identificación. Primero se definirá la fecha de registro o actualización del componente, especificando el año y el mes, ambos con dos caracteres. Luego el nombre del centro seguido por un nombre corto para el componente. La sintaxis sería como sigue:

<aa mm><centro><nombre_comp>

Ejemplo: 10 04 Comp1.

Por el nombre del activo, se infiere que fue almacenado en abril del 2010.

Versiones:

Se recomienda, que antes de guardar algún activo en el repositorio central, éstos se almacenen en algún repositorio local, a modo de ir teniendo en algún repositorio temporal los componentes del centro mientras están en desarrollo. En el repositorio local, los componentes se recomiendan ser almacenados en dos versiones, uno para los componentes inestables, que aún están en fase de reimplementación o de prueba, para los cuales se destinarán un espacio llamado *branches* o rama*;* y otro para las versiones estables, para los casos de los componentes que están en etapa de revisiones y pruebas por calidad central para ser finalmente liberados. Estos últimos se

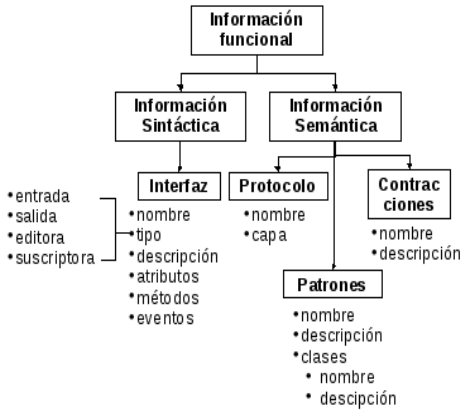


Figura 24: Datos de la información funcional de un activo, en específico de un componente.

almacenarán en *trunk* o tronco.

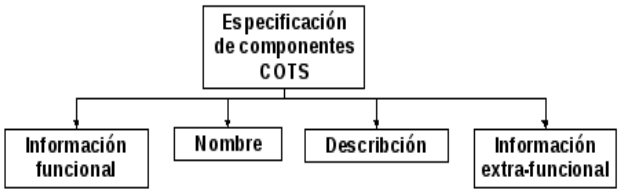


Figura 25: Secciones que incluyen la especificación de un activo de software.

2.2.4. Atributos de calidad de los activos

No existe un criterio común a la hora de definir los atributos o características de calidad que debe cumplir un producto o activo software. Por tal razón, las medidas de calidad presentadas en este trabajo se basó en el modelo propuesto por Manuel F. Bertoa y Antonio Vallecillo, el cual es una adaptación a los estándares ISO/IEC 14598 e ISO/IEC

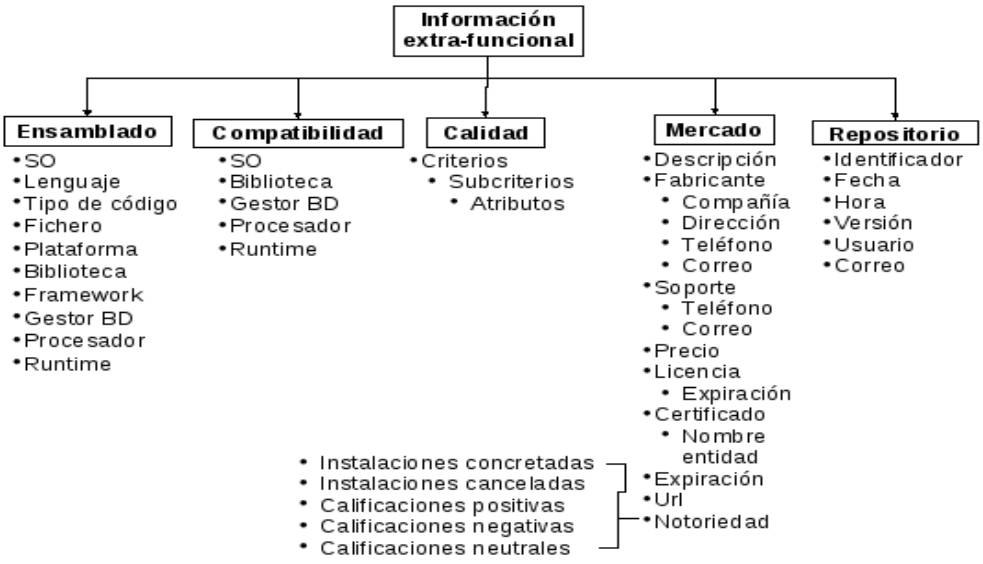


Figura 26: Datos de información funcional de un activo. Incluye los datos para componentes COTS.

9126 (enfocándose más en este último) para hacerlos converger y eliminar algunas incoherencias presentes en ellos; de esta forma, adaptarlos para los dominios de

software de reutilización, específicamente para evaluación de componentes.[52]

Estos autores denominan como atributo a “una propiedad de calidad a la que puede asignársele una métrica, donde una métrica es un procedimiento que examina un componente y produce un dato simple, un símbolo (p.e. Excelente, Sí, No) o un número.” [47][53] A su vez, definen modelo de calidad como el “conjunto de características y sub-características, y cómo éstas se relacionan entre sí. “ [42][83]

Los criterios de calidad podrían tener en determinados casos, el mayor peso para la selección y evaluación de activos de software, sobre todo si la selección es entre dos o más activos; incluso, podría llegar a ser el factor determinante.

Las entidades de nuestro estudio serán los activos software, pero se enfocará más a los componentes por ser los de mayor complejidad y los que más criterios de calidad requerirán para ser evaluados.

La siguiente tabla muestra los criterios de calidad propuestos en la metodología para medir la calidad de un activo, tomando como referencia el modelo de Bertoa y Vallecillo, al cual se le realizaron pequeños cambios para restarle complejidad al análisis de calidad de los activos. La mayoría de los conceptos y formas de cálculo de los atributos de calidad aquí presentados, fueron tomados también del modelo de Bertoa y Vacillo.

Tabla 5: Criterios de calidad.

Criterios	**Subcriterios**	**Atributos**	**Valor**
Funcionalidad	Idoneidad	Cobertura de interfaces	No.
		Exceso de interfaces	No.
		Cobertura de implementación	No.
	Interoperabilidad	Compatibilidad de los datos	Propietario/ Estándar
	Conformidad	Conformidad con estándares	Sí/No
		Certificaciones	Sí/No
Fiabilidad	Madurez	Volatilidad	xps,m,a
		Evolucionalidad	No.
		Fallos eliminados	No.

	Rendimiento	Plataforma en que trabaja	SO
		Transacciones permisibles. Procesamiento.	Tps,m,h
	Disponibilidad	Tiempo medio entre fallas (MTBF)	No.
		Tiempo medio para restaurar (MTTR)	No.
		Disponibilidad	0-1
	Seguridad	Cifrado de datos	Sí/No
		Valor promedio de falso rechazo (FRR)	%
		Valor promedio de falsa aceptación (FAR)	%
		Vulnerabilidad a la Inyección SQL (SIV)	%
	Fiabilidad	Uso promedio de componentes ya probados previamente	%
Usabilidad	Capacidad de aprendizaje	Período para usar correctamente	t(d,s,m,a)
		Período para configurar correctamente	t(d,s,m,a)
		Período para administrar correctamente	t(d,s,m,a)
		Período para dominar correctamente	t(d,s,m,a)
	Comprensibilidad	Documentación de usuario	Sí/No
		Sistema de ayuda	Sí/No
		Documentación computacional	Sí/No
		Formación y capacitación	Sí/No
	Operabilidad	Esfuerzo para operar	Cualit
		Esfuerzo para configurar	Cualit
		Esfuerzo para administrar	Cualit
	Complejidad	Interfaces ofrecidas	No.
		Interfaces externas utilizadas	No.
		Índice de complejidad	Cualit
Mantenibilidad	Adaptabilidad	Índice de modificación	%

Portabilidad	Reemplazabilidad	Compatibilidad hacia atrás	Sí/No
	Capacidad de instalación	Fácilmente instalable	Sí/No
Recuperabilidad	Persistente	Persistente	Sí/No
	Transaccional	Transaccional	Sí/No
	Tratamiento de errores	Detecta errores	Sí/No
		Detectar y avisar errores	Sí/No
		Tratar errores	Sí/No
Eficiencia	Consumo temporal	Tiempo de respuesta	t (ms,s)
		Capacidad en emisión	t (ms,s)
		Capacidad en recepción	t (ms,s)
	Consumo de recursos	Requisitos de memoria	Mb/Kb/b
		Utilización de disco	Mb/Kb/b

Estos factores de calidad fueron definidos pensando en abarcar las características de calidad que debe cumplir un componente por ser el más íntegro y autónomo de los activos. Por esa razón, se hará más alusión a él. Pero estos mismos pueden ser aplicados a cualquier otro activo de software, en dependencia de su complejidad, se utilizará un conjunto de estos factores para medir su calidad.

2.2.4.1. Funcionalidad

La *funcionalidad* expresa la "capacidad para proporcionar las funciones que satisfagan las necesidades establecidas o implícitas cuando se usa bajo las condiciones especificadas." [54]

Dentro de las subcaracterísticas de *funcionalidad* están Idoneidad, interoperabilidad y conformidad.

Idoneidad

Es el conjunto de condiciones necesarias para satisfacer determinado requisito o un conjunto de ellos. Indicador que evalúa el grado de ajuste a los requisitos funcionales que necesita o especifica el cliente del activo.

Por tanto, se deben medir el número de interfaces que se utilizan frente a las que se ofrecen, es decir, que porcentaje se aprovechan; y cuantas interfaces de las que se necesitan no las suministra el componente.

Presenta los atributos:

- Cobertura de interfaces: relaciona el número de interfaces que necesita el usuario del activo con el número de interfaces que ofrece. Puede calcularse según la fórmula 1.
- Exceso de interfaces: relaciona el número de interfaces del activo que se van a utilizar en la aplicación donde se integrará, con las que ofrece. Puede calcularse según la fórmula 2.
- Cobertura de implementación: mide el número de operaciones que se han implementado respecto al núumero total de operaciones que se indican en la especificación del cliente. Puede calcularse según la fórmula 3.

$$100 * \frac{Interfaces\ Necesarias \cap Interfaces\ Ofrecidas}{Interfaces\ Necesarias}$$

Fórmula 1: Para calcular "Cobertura de interfaces".

$$100 * \frac{Interfaces\ Utilizadas \cap Interface}{Interfaces\ Ofrecidas}$$

Fórmula 2: Para calcular "Exceso de interfaces".

$$100 * \frac{Operaciones\ Implementadas}{Operaciones\ Especificadas}$$

Fórmula 3: Para calcular "Cobertura de implementación".

Interoperabilidad

La interoperabilidad mide el nivel de intercambio de los datos dentro de un sistema. Como indicador o atributo de esta subcaracterística, está:

- Compatibilidad: indica si maneja los datos en un formato propietario o estándar. Es recomendable especificar además el estándar usado para el manejo de los datos (ej.: XML o alguno similar, o propietario). [52]

Conformidad

La *conformidad* es el grado de cumplimiento con las especificaciones de los estándares y certificaciones del activo. Dentro de él se evalúan:

- Conformidad con estándares: indica si el activo cumple con algún estándar internacional. Se recomienda además, especificar el estándar.
- Certificaciones: indica si el activo está acreditado con algún tipo de certificación, tanto por organización externa o de forma interna. En caso de estar certificado el activo, se recomienda especificar la organización que lo acredita.

2.2.4.2. Fiabilidad

La fiabilidad es definida en algunas bibliografías como “la probabilidad de ejecución sin fallos de un sistema software en un entorno específico para un período de tiempo determinado”, por tanto, determina la productividad operativa del producto. [55-59]

Es el criterio de calidad que sirve para medir la madurez, el rendimiento y la disponibilidad de un software o activo de software. “Las técnicas de fiabilidad de software se enfocan a la reducción o eliminación de las fallas del sistema”.[56]

Madurez

La *madurez* de un activo se mide en función de los cambios que sufren las versiones de los activos o componentes desplegados o integrados a otro sistema, y la velocidad a la que aparecen. Contiene tres parámetros a medir:

- Volatilidad: tiene en cuenta la frecuencia de actualización del activo de software. También podría llamarse, estabilidad, pero en caso de ser así, el parámetro a medir sería el tiempo en que el activo permanece estable. Pero lo importante para este trabajo de investigación es la frecuencia de actualizaciones que tiene el mismo, que podría tener varios valores, en dependencia del esfuerzo del equipo fabricador del componente; podría ser 2 veces por mes (2pm), 2 veces por año (2pa), cada 6 meses (c6m), cada 15 días (c15d).

- → Resumiendo, podría representarse con la siguiente sintaxis.
  x|p|(d|s|m|a) donde 'x' representa un número, 'p' representa la palabra “por” y las letras 'd', 's', "m y 'a' representan “día”, “semana”, “mes” y “año” respectivamente.
- → También podría usarse la sintaxis:
  c|x|(d|s|m|a) donde 'c' representa la palabra “cada” manteniendo el mismo significado para el resto de las variables.

➢ Evolucionalidad: parámetro que indica el número absoluto de versiones del activo (específicamente componente) que han sido desplegadas y/o comercializadas.

➢ Fallos eliminados: indicador que registra el número de fallos que han sido eliminados satisfactoriamente.

Rendimiento

El *rendimiento* mide 3 parámetros:

- Plataforma en que trabaja: indica el o los SO sobre los cuales el componente puede ser utilizado. Para el caso de los SO de GNU/Linux, se especificará también la distribución. Además, si para el buen funcionamiento del componente se requiere de la instalación de determinadas plataformas o bibliotecas de desarrollo, estas también deberán ser especificadas.
- Transacciones permisibles: también conocido como procesamiento, es el número de transacciones procesadas en un período específico de tiempo, ejemplo, transacciones por segundo (tps), transacciones por minuto (tpm), transacciones por hora (tph). El objetivo es el número de transacciones en el tiempo pico; pudiéndose representar con la sintaxis siguiente:
  t|p|(|s|m|h) donde 't' representa la palabra “transacciones”, 'p' representa la palabra “por” y las letras 's', 'm' y 'h' representan “segundo”, “minuto” y “hora” respectivamente.

Disponibilidad

La disponibilidad es la capacidad de un sistema de almacenamiento de acceder a

información aún en caso de producirse algún fallo. Esta falla puede deberse a daños físicos (mal funcionamiento), accidentales u operacionales (persona que operan mal en el componente), lo que produce la pérdida de información almacenada. [60]

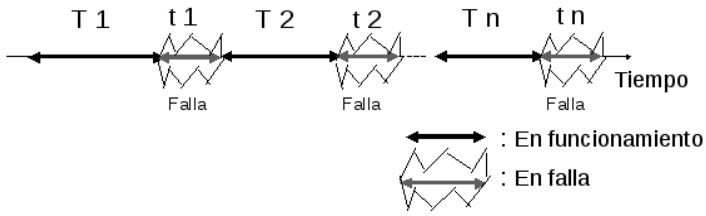
- Tiempo medio entre fallas (MTBF: Medium Time Between Failures): dice qué paradas son las más frecuentes para un proceso.

  Es conocido también como valor promedio del tiempo de operación. El mayor valor es el de mayor período entre fallas o el de mayo período en operación. Por consiguiente, mientras mayor sea este valor, mejor.

  Se puede calcular mediante la fórmula:

  MTBF= (T1 + T2 + • • • + Tn) / n

  Donde 'Tn' es el intervalo de tiempo en operación, sin interrupción por rotura; 'n' es el total de horas de análisis o de prueba. La siguiente figura representa el significado de las variables en una línea de tiempo.



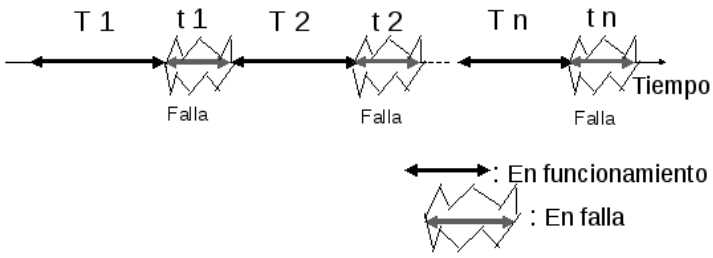
- Tiempo medio para restaurar (MTTR: Medium Time To Repair): dice cuáles fallas son las más graves.

  Conocido también como valor medio de reparación o tiempo de falla, representa los tiempos cuando el dispositivo no está en operación. El valor más pequeño es el período más bajo de reparación. Por consiguiente, mientras más pequeño sea éste, mejor.

  Se puede calcular mediante la fórmula:

  MTTR = (t1 + t2 + • • • + tn) / n

  Donde 'tn' es el intervalo de tiempo de falla. La siguiente figura representa el significado de las variables en una línea de tiempo.



- Disponibilidad: Es el ratio del tiempo en operación con el tiempo total. Los valores obtenidos están entre 0 y 1.

  El valor más cercano a 1, es el de mayor disponibilidad.

  La disponibilidad puede ser calculada con la fórmula: MTBF / (MTBF + MTTR), doinde el valor es un número entre 0 y 1.

$$0 \leqq \frac{MTBF}{MTBF + MTTR} \leqq 1$$

  Si el valor diera 1, significa que el sistema funciona todo el tiempo.

Seguridad

Este atributo de calidad medirá atributos relacionados con veracidad de la información que se registra en las pruebas que se le haga al activo durante su funcionamiento. Es muy objetivo para establecer el correcto funcionamiento del activo. Las variables que se evalúan con este atributo de calidad, son más medibles para en los componentes debido a sus características y complejidad; por tal razón, está más enfocada a los activos del tipo componente:

- Cifrado de datos: Indica si el componente tiene cifrado los datos que son sensibles a la confiabilidad.
- Valor promedio de falso rechazo (FRR): es el valor de la cantidad de rechazos en un sistema cuando debería ser positivo el resultado. Ejemplo, las pruebas que se realizan de autenticación de usuarios con los datos válidos, accesos a recursos o consultas a bases de datos con valores correctos y el sistema rechaza las transacciones cuando deberían

$$\text{No. Pruebas} \rightarrow 100\%$$

$$= \frac{\text{No. Rechazos} * 100\%}{\text{No. Pruebas}} = \text{FRR}$$

$$\text{No. Rechazos} \rightarrow \text{FRR}$$

Fórmula 4: Para calcular "Valor promedio de falso rechazo (FRR)".

ser aceptadas.

La taza de falso rechazo FRR en porcentaje se obtiene al aplicar la siguiente relación: si el número de pruebas en cada umbral es 100%, cuánto será en porcentaje el número de rechazos. A continuación se presenta la regla de tres simple.

- Valor promedio de falsa aceptación (FAR): contrario al FRR, es el valor de la cantidad de aciertos en un sistema cuando debería ser negativo el resultado. Ejemplo de esto es cuando el sistema o componente realiza falsa aceptación ante la autenticación y accesos a recursos o bases de datos con consultas y datos inválidos.
- Vulnerabilidad a la Inyección SQL (SIV): ésta es una vulnerabilidad que presentan las bases de datos en la validación de los valores de entrada que recibe. Una de las causas principales de este problema es el filtrado incorrecto de las variables utilizadas en las partes del programa con código SQL. Cada vez que se inyecta una sentencia SQL y es logrado el objetivo del probador, o del atacante, que no es más que adquirir toda la cantidad de información posible de la base de datos sin necesidades de acceder a través de un usuario válido, es un punto a favor de la vulnerabilidad del componente, o del sistema.

2.2.4.3. Usabilidad

"La usabilidad de un activo de software debe interpretarse como la capacidad para ser utilizado en la construcción de un producto o sistema software." [53] Por esta razón, es la característica principal de medición de calidad de un activo de reutilización. [73][74]

La Organización Internacional para la Estandarización (ISO) dispone como definición de usabilidad: "se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso" [61]

Capacidad de aprendizaje

La aprendibilidad mide el nivel de comprensión y entendimiento del activo, su facilidad de aprendizaje para concebir su reutilización. De ello depende el tiempo que debe transcurrir para que una persona pueda asimilar todo el conocimiento necesario sobre el

activo para poder hacer las adaptaciones necesarias. Para esto, de definen 4 atributos:

- Período para usar correctamente: tiempo medio que necesita un desarrollador de aplicaciones para poder utilizar de forma correcta el activo.
- Período para configurar correctamente: tiempo medio que necesita un desarrollador para poder configurar de forma adecuada los parámetros que el activo permite particularizar, comprendiendo el significado correcto de los valores que se pueden utilizar.
- Período para administrar correctamente: tiempo medio para poder realizar las operaciones de administración que requiera el activo de forma correcta.
- Período para dominar correctamente: tiempo medio para llegar a dominar de forma precisa la gran mayoría de las posibilidades que ofrece el activo.

Este período se puede dar en días, semanas, mes o año, endependencia de la complejidad del activo, independientemente del nivel de especialización del cliente.

Comprensibilidad

Mide las facilidades que brinda el proveedor del activo para que éste sea entendido, cada documentación tanto en el código como manuales técnicos y libros de ayuda son factores que influirán en el nivel de comprensibilidad del activo. A continuación se muestran los atributos que impactan en esta subcaracterística.

- Documentación de usuario: documentos creados con el fin de brindarle al usuario información sobre el funcionamiento y característica generales del activo.
- Sistema de ayuda: describe información acerca de los servicios e interfaces del componente.
- Documentación computacional: descripción modelada del funcionamiento interno del activo.
- Formación y capacitación: comprende toda la documentación generada y referenciada en los cursos preparados para ser impartidos a los usuarios o clientes del activo de software.

Operabilidad

Es definido por la mayoría de las métricas como la capacidad del activo para permitir al desarrollador de sistemas operar con él y controlarlo. Entre las variables que evalúa están:

- Esfuerzo para operar: mide el grado de esfuerzo necesario para poder operar con el activo. Se puede seguir la siguiente clasificación, donde el esfuerzo representa la cantidad de servicios que se requiere ser inicializados para la operación del activo.

Clasificación	Esfuerzo
Lijero	=<3
Medio	3 – 5
Complejo	>=6

- Esfuerzo para configurar: mide el grado de esfuerzo necesario para poder configurar los parámetros del activo. Depende del nivel de acoplamiento que exista entre los servicios e interfaces a la hora de realizar alguna configuración. Para este trabajo se sigue la misma clasificación que el caso de "Esfuerzo para operar", donde la variable "Esfuerzo" sería en este caso la cantidad de servicios de los que dependerá actualizar alguna configuración de algún parámetro del activo.
- Esfuerzo para administrar: mide el grado de esfuerzo necesario para poder realizar operaciones de administración del activo. Para este caso también se usa valores de nivel (valores cualitativos).

Complejidad

Este atributo de calidad da medida de la complejidad del uso de un activo de software y de su integración en un producto o sistema. Medimos el número total de interfaces que tiene y el número medio de operaciones por interfaz:

- Interfaces ofrecidas: cantidad de interfaces que ofrece el activo o componente. Cuanto mayor sea este número, mayor será la complejidad de uso, y en la mayoría de los casos, su complejidad funcional.
- Interfaces externas utilizadas: cantidad de interfaces de otros componentes que

$$\frac{\text{Operaciones en todas las Interfaces Ofrecidas}}{\text{Interfaces Ofrecidas}}$$

Fórmula 5: Para calcular "Índice de complejidad".

se necesita para operar. Indica la complejidad de la integración de este componente en un sistema y su nivel de dependencia con respecto a otros componentes.

- Índice de complejidad: mide el número medio de operaciones por interfaz ofrecida que presenta el componente. Se evalúa según la fórmula 5.

2.2.4.4. Mantenibilidad

La mantenibilidad es una característica que puede ser medida durante el ciclo de vida de un producto. Mide la capacidad de un producto software de ser modificado y entendiendo ante cualquier modificación, corrección, mejora o adaptación. En este sentido se necesitará probar el activo de software antes de incluirlo en su aplicación o cambiar alguno de las parámetros que se pueden particularizar. Por ello, las sub-características Adaptabilidad y Facilidad de Prueba son las que deben ser medidas para los activos. [21][52]

Adaptabilidad

Esta subcaracterística mide la facilidad de cambio del activo, ya sea por corrección, mejora o adaptación.

- Índice de modificación: indicador de la capacidad de modificación de un activo (más aplicado a componentes). Se tienen en cuenta el número de parámetros que tiene un componente y el número de interfaces ofrecidas por el mismo. Así, un componente que ofrece pocas interfaces y muchos parámetros, será muy modificable; mientras que uno con muchas interfaces y pocos parámetros es poco modificable. Se puede calcular según como se describe en la fórmula 6.

$$\frac{Numero\ de\ Parametros}{Numero\ de\ Interfaces\ Ofrecidas}$$

Fórmula 6: Para calcular "Índice de modificación".

2.2.4.5. Portabilidad

Esta característica se define como la capacidad de poder ser reutilizado el activo en distintos entornos, o como lo define también ISO 9126, “capacidad de un sistema software para ser transferido desde una plataforma a otra”. Esa es la esencia misma de los activos de reutilización, que son diseñados y desarrollados específicamente para ser

reutilizados. [20]

Reemplazabilidad

Mide la capacidad del activo de ser reemplazable por otras versiones. Evalúa el atributo:

- Compatibilidad hacia atrás: indica si existe compatibilidad de la versión actual del activo con las versiones anteriores del mismo. Se recomienda además, especificar qué versiones del activo son compatibles con la versión actual.

Capacidad de instalación

Mide la facilidad de instalación del componente en cuanto a que requieran de configuración y documentación.

2.2.4.6. Recuperabilidad

Esta característica se define como la capacidad de poder ser reutilizado el activo en distintos entornos, o como lo define también ISO 9126, “capacidad de un sistema software para ser transferido desde una plataforma a otra”. Esa es la esencia misma de los activos de reutilización, que son diseñados y desarrollados específicamente para ser reutilizados. [20]

Persistente

Este atributo indica si el activo es capaz de almacenar su estado de forma persistente.

Transaccional

Este atributo indica si el activo, específicamente un componente, suministra alguna interfaz que permita que sus operaciones estén sujetas a transacciones.

Tratamiento de errores

Indica si se realiza algún tipo de tratamiento de errores y en su caso el tipo de tratamiento de errores que lleva a cabo el activo de software:

- Detectar errores: se refiere a los activos de software que tienen la capacidad de detectar los errores pero no realiza ninguna acción para su corrección.
- Detectar y avisar errores: activos con la capacidad de detectar y avisar ante la ocurrencia de errores.

- Tratar errores: activo con capacidad de detectar errores e implementa un mecanismo de excepciones.

2.2.4.7. Eficiencia

Esta característica se define como la capacidad de disponer de recursos para conseguir el buen funcionamiento del activo de software. Para nuestro caso, se mide la necesidad de consumo de tiempo y de recursos del activo para realizar su función.

Consumo temporal

Este atributo mide la necesidad de consumo de tiempo para realizar transacciones de recepción y emisión de datos.

- Tiempo de respuesta: mide el tiempo transcurrido desde que el activo recibe la petición hasta que emite una respuesta.
- Capacidad de emisión: mide la cantidad de información que es capaz de transferir el activo en un intervalo de tiempo determinado.
- Capacidad de recepción: mide la cantidad de información que es capaz de admitir y procesar el activo en un intervalo de tiempo determinado.

Consumo de recursos

Este atributo indica en nivel de consumo de recursos que requiere el activo para su ejecución.

- Consumo de memoria: evalúa la cantidad de memoria que necesita un activo para ejecutarse.
- Consumo del disco: evalúa la cantidad de espacio en disco que necesita el activo tanto de forma estática, como la que ocupa de forma dinámica cuando se invoquen sus operaciones.

Estos criterios de calidad se tienen complementados en el documento arquitectura de la UCI. [62]

2.3. Análisis de limitaciones y potencialidades de la metodología propuesta antes y conclusiones del capítulo

2.3.1. Potencialidades

Nivel de abstracción

La metodología propuesta en este trabajo tiene un alto nivel de abstracción debido a que está enfocado a cualquier caso donde exista un ambiente de reutilización, esto posibilita dar cierta flexibilidad a la metodología y libertad a los que la usen para hacer su propia adaptación.

Nivel de abstracción de la documentación de especificación de activos

Por ejemplo, el modelo de descripción para la especificación de los activos de software a reutilizar se diseñó pensando en todas las características que incluye el activo más complejo, es decir, un componente COTS, pues incluye también información de precio en datos del mercado, que se debe tener en cuenta para estos tipos de activos.

Sin embargo, no se detalla qué información tener en cuenta para la documentación de otro tipo de activo que no requiera de tantos datos. Por tal razón, para realizar la especificación de otro tipo de activo menos complejo, habrá que seleccionar un conjunto de datos según los que considere útil la organización para la descripción de sus activos en correspondencia con su complejidad.

Nivel de abstracción de los procesos definidos

Los procesos definidos fueron muy desglosados, pero podría ser eliminado algún proceso o subproceso, o se podrían fusionar algunos de ellos en virtud de disminuir la complejidad. También se podría definir como proceso alguno de los que están como subproceso en virtud de darle mayor peso e importancia.

Proactividad

La metodología tiene un alto nivel de proactividad porque en ella existen elementos que favorecen al anticipo de comportamientos no favorables en un ambiente de reutilización, ejemplo:

- Disminución del tiempo de producción.

- → El propio concepto de reutilización promueve la disminución del tiempo dedicado a generar una solución final, pues las piezas que formarían el producto están ya pre-elaboradas.

- Aumento de la reutilización.
  - → El hecho de poder contar con un repositorio disponible para la publicación de los activos de software, permite poder reutilizar todo lo que se desarrolle en una organización.
  - → Se propone un sistema de recompensas que clasifica a las entidades involucradas en la reutilización según su actividad en la red.
  - → Se define una fórmula para determinar el ranking de los activos más usados. Esto le daría más credibilidad tanto al activo como a la entidad proveedora que lo desarrolló.
  - → Otro elemento que estimula la reutilización es el grado de satisfacción que haya acumulado una entidad proveedora en virtud de las opiniones realizadas por las entidades clientes, basadas en el grado de cooperación e intercambio que haya tenido ésta con su proveedor de activos.
  - → Por último existen niveles de madurez que clasifica a los activos en primitivo, difundido y muy difundido. Mientras más reutilizado es un activo, mayor nivel de madurez le aportará.
- Prevención de la publicación de activos no probados o de poca calidad.
  - → El acta de certificación de la calidad que debe ser asignada por la entidad evaluadora a un activo de software como evidencia de tener un nivel de calidad satisfactorio, es un modo de prevenir la publicación de un activo mal desarrollado o ineficiente, pues este acta deberá ser entregada junto con el activo a la entidad publicadora como pre-condición para poder ponerlo disponible en la red de reutilización.
- Prevención de fraudes y mal uso del repositorio.
  - → Es garantizado a través de un código de ética, que deberá ser firmado previamente por cada entidad que intercambiará con el repositorio de activos ya sea para publicar (entidad proveedora), reutilizar (entidad cliente) o para

evaluar (entidad evaluadora).

- Prevención de accesos no autorizados al repositorio.
  - → Es garantizado en gran medida a través de las políticas de seguridad y control de acceso al repositorio de activos que debe definir cada organización que desee establecer un ambiente de reutilización, como método para garantizar además la fiabilidad de los datos que se almacenen en el repositorio.

Tabla 6: Elementos de la metodología que demuestran su carácter proactivo.

Elementos de la metodología	**Favorecen a:**
La reutilización	Disminución del tiempo de producción
Repositorio de activos de software	Aumento de la reutilización
Sistema de recompensas	
Ranking de activos de software	
Grado de satisfacción de entidad proveedora	
Niveles de madurez de un activo	
Acta de certificación de la calidad	Prevención de la publicación de activos no probados o de poca calidad
Código de ética	Prevención de fraudes y mal uso del repositorio
Políticas de seguridad y control de acceso al repositorio de activos	Prevención de accesos no autorizados al repositorio

Sistémico

Los tres procesos principales que centran su actividad al uso del repositorio definidos en la metodología, se relacionan entre sí. Esto se puede visualizar ya sea porque los subprocesos de uno generan salidas que son entradas de subprocesos de otros, o porque utilizan para su ejecución salidas generadas por otros. La figura 29 muestra el comportamiento de la interacción entre los procesos.

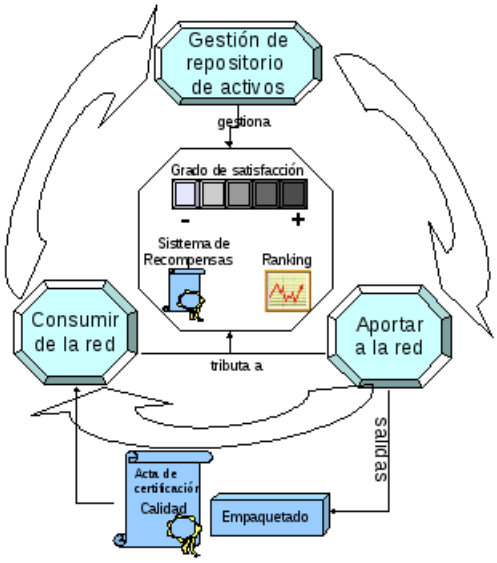


Figura 27: Carácter sistémico. Integración de los procesos.

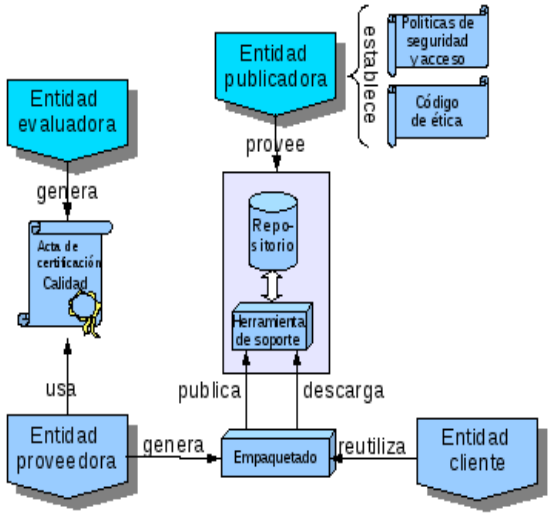


Figura 28: Carácter participativo. Integración de las entidades participantes.

Participativo

Todas las entidades que intervienen en la reutilización están integradas y tienen definidas sus responsabilidades. La figura 30 muestra el comportamiento de la

interacción entre las entidades involucradas en la reutilización.

Evolutivo

La metodología cuenta con un proceso de selección de activos que genera un conocimiento a medida que se avanza en el tiempo. Ese conocimiento acumulado durante todo el ciclo de reutilización de los activos, se expresa en:

- Ranking de activos de software, donde los más reutilizados son los de mayor puntaje.
- Ranking de entidades involucradas, donde las entidades que más activos publiquen y a su vez reutilicen, son las de mayor puntaje.

Este conocimiento acumulado se convierte en una fuente de información importante para el análisis del comportamiento que va teniendo la organización.

Involucramiento de la alta gerencia

La entidad publicadora es a quien se le asigna la máxima responsabilidad de la aplicación de todas las políticas de reutilización en la organización y de velar por que se cumplan.

Sin embargo, para las organizaciones donde la cantidad de entidades tanto proveedoras de activos como entidades clientes que los reutilicen sea un número muy elevado, donde haya un alto nivel de intercambio en la red, no sería factible este modelo porque la actividad de la entidad publicadora para realizar la gestión del repositorio sería muy cargada, y no podría ser llevar paralelo las actividades de dirección.

En ese caso, habría que identificar los responsables de realizar las tareas correspondientes a la alta gerencia.

Centrado en la calidad

Existe como precondición para la publicación en algún activo de software, la presentación del acta de certificación de la calidad junto con el empaquetado del activo. De esta forma se valida que el activo fue probado y certificado antes de ser publicado.

2.3.2. Limitaciones del modelo

Grado de aporte directo a la productividad

La metodología no se compromete con la reutilización efectiva de los activos, queda fuera de su alcance y es parte de la responsabilidad de la alta gerencia.

Facilidad de integración continúa

La metodología no contempla elementos ni sugiere herramientas hasta la versión actual para la integración continua de los activos publicados en el repositorio.

Nivel de actualización automática de los activos

La metodología hasta la versión actual no identifica métodos para la actualización automática de componentes.

Inclusión de sistema para la explotación del repositorio (mediador de repositorio)

Hasta la versión actual de la metodología, no se establece un mediador para la búsqueda automática de los activos.

Nivel de integración de activos de software formando ecosistemas

Quedó fuera del alcance de la metodología, la propuesta de mecanismos para la creación de colecciones de proyectos de software formando ecosistemas (colección de proyectos software que se desarrollan y evolucionan en un mismo entorno) y de un súper-repositorio explícito para los repositorios de control de versiones de los disímiles proyectos software de un entorno de reutilización. [63]

3. Aplicación y validación de la propuesta

En el presente capítulo se reflejan los resultados obtenidos durante la implantación de la propuesta del ambiente de reutilización de componentes en la UCI. En el desarrollo del mismo se contó con la ayuda de la Dirección Técnica de la universidad para establecer como política la forma de trabajo para la colaboración en el entorno de reutilización de componentes.

Aplicación de la propuesta:

En la Universidad de las Ciencias Informáticas (UCI), centro de estudios basado en el concepto de universidad productiva, convergen las necesidades de la formación y de la producción de soluciones informáticas hacia un único objetivo: formar recursos humanos para el desarrollo cubano en virtud de informatización e ingreso de divisas por exportación.

Para cumplir con este objetivo, fue necesario en la universidad una modificación de su plan de estudio y del modelo de producción simultáneamente, potenciando la sinergia entre las áreas de producción, formación, investigación y postgrado.

El centro de estas modificaciones vista desde la producción se enfoca en: la modificación del modelo de desarrollo tecnológico potenciando la productividad, reordenamiento de las fuerzas, incorporación de herramientas que informaticen la gestión de la información, el control y seguimiento. En este sentido se reconceptualiza la infraestructura productiva de la universidad y se crea una red de centros de producción de la universidad.

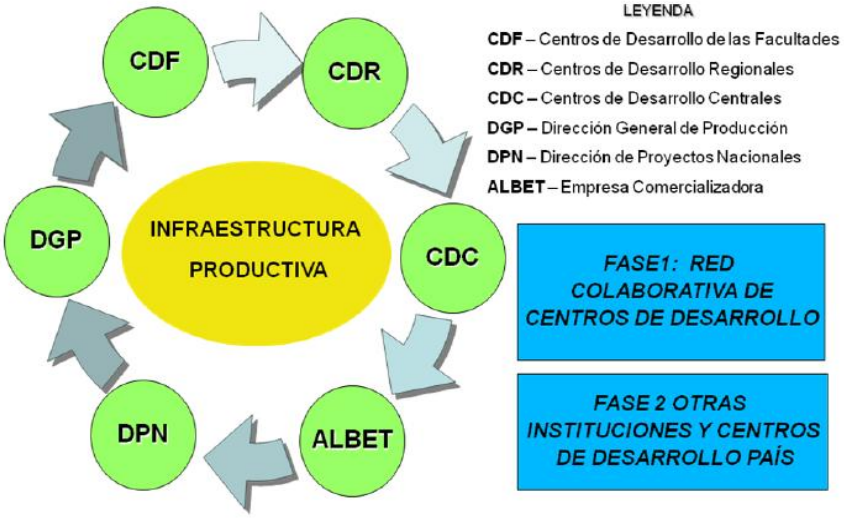


Figura 29: Centro de desarrollo de software en la UCI.

Entre los principios del trabajo de esta red de centros se encuentran:

- La red de centros de la UCI debe estar completamente en consonancia con los lineamientos económicos del partido especialmente con los relacionados con la eficiencia productiva a partir de la reutilización y la auto-sustentabilidad económica, científica y tecnológica.
- Todos los centros de la RED colaboran de forma sistémica para alcanzar los objetivos de la RED relacionados fundamentalmente con el ingreso de divisas por concepto de exportación de soluciones de software y servicios informáticos y la constante búsqueda de nuevas formas de producción que disminuyan los costos asociados al proceso productivo.
- La colaboración entre los entes de la Red debe fomentar la creación de Ecosistemas de Aplicaciones Informáticas potenciando la comercialización en las áreas temáticas en que la red tiene experiencias demostradas de saber hacer. Se define en este sentido ecosistema de aplicaciones informáticas como:
  - → *Ecosistema de aplicaciones informáticas:* El conjunto de entidades de negocios que actúan como una unidad e interactúan para compartir un

mercado común de aplicaciones informáticas y servicios bajo un sistema ordenado de interrelaciones entre ellas. El sistema de relaciones entre las entidades está soportado en la simbiosis entre arquitecturas y plataformas comunes que facilitan la integración de soluciones y componentes, el intercambio de información, recursos, artefactos y activos en general.“[65][66]

- El desarrollo de la red debe soportado por un robusto sistema para la protección legal de los resultados y el establecimiento de reglas de intercambio que favorezcan la comercialización y el desarrollo de activos de la base tecnológica de la red de centros.
- Los centros de la red deben implantar modelos de producción industrial novedosos centrados en la reutilización y el intercambio de conocimientos basados en los ecosistemas de software y la producción basada en líneas de productos de software. Ver en la sección Anexos, la figura 31 que muestra el Modelo que representa en una única vista relacionada con el desarrollo tecnológico las interrelaciones entre las entidades.

En este entorno se hace necesaria la creación de un repositorio de componentes que garantice el almacenamiento y de los componentes y las soluciones desarrolladas por los centros de la Red. Se define a la Dirección Técnica de la Producción (DT) como:

- Área responsable del establecimiento de las políticas para el desarrollo de la producción y del modelo de producción que potencie la reutilización.
- Área responsable del establecimiento y del repositorio de activos reutilizables a la Dirección Técnica de la Producción (DT).
- Área responsable de la estandarización de las herramientas que soporte el modelo de producción y la gestión de los proyectos en el mismo.

Se decide establecer que el sistema Paquete de Gestión de Proyectos GESPRO v1.0[1] [64] constituya una plataforma integrada para potenciar la dirección integrada de los proyectos de la universidad y la implantación del nuevo modelo de desarrollo tecnológico centrado en la reutilización.

[1] (No Registro CENDA Cuba paquete GESPRO v1.0: 1540-2010.)

Este paquete está formado por cinco grupos principales de herramientas:

- Herramientas para la dirección integrada de proyectos.
- Herramientas para la gestión documental y el control de versiones.
- Herramientas para el monitoreo, la administración y la recuperación ante fallos.
- Herramientas para el trabajo colaborativo y la reutilización
- Herramientas para la ayuda a la toma de decisiones.

En la figura 32 de la sección Anexos se muestran los ppaquetes que contiene la herramienta GESPRO.

Es precisamente en el grupo de herramientas para el trabajo colaborativo de GESPRO que se decide incorporar el "Repositorio de Activos Reutilizables de la UCI" y aplicar para su diseño, implantación y explotación la metodología propuesta en la tesis.

Se define entonces que se aplique la metodología propuesta en la tesis para el beneficio del 100% de los centros de la red de centros de la universidad para crear un repositorio que satisfaga los siguientes tres objetivos fundamentales:

1. Disminuir el re-trabajo en la organización, evitar la duplicidad de esfuerzos en pos de una misma funcionalidad o conjunto de estas cuando pueden ser reutilizadas.
2. Incrementar el nivel de la actividad o de integración de los centros de desarrollo en la red de centros de la Universidad a partir del aporte de activos reutilizables y servicios, así como el consumo de elementos de este tipo desde la red. Permitiendo detectar los parásitos de la red.
3. Aumentar la eficiencia a partir de la reutilización de diseños, análisis de requisitos, código fuente, componentes y componentes COTS entre otros activos, aumentando la fortaleza y productividad de la red de centros.

3.1. Aplicación de la metodología para el diseño de un repositorio de activos.

Es éste acápite se realiza una descripción detallada de los pasos seguidos para el diseño e implantación de un repositorio de activos según la forma en que fue adaptado y aplicado a la institución donde se hizo la validación de esta propuesta. En esta

adaptación, se hace más énfasis a los componentes por ser el activo de software más reutilizado.

3.1.1. Paso 1 detallado: Definición de los procesos.

El trabajo con el repositorio de activos de software y los propios componentes dentro de su ciclo de vida descansan en 6 procesos fundamentales en el entorno de reutilización aplicado en la institución. Todos trabajan para impulsar el trabajo con el repositorio pero independientes entre sí para evitar los retrasos y cuellos de botella de los procesos secuenciales:

1. Aportar a la red
2. Evaluación de componente
3. Consumir de la red
4. Actualización de la información
5. Auditoría a componentes
6. Gestionar repositorio

La siguiente figura muestra las adaptaciones realizadas a la metodología donde se exponen los elementos que apoyan la reutilización en la red de centros de producción, los procesos o actividades que aportan a la integración de la red de reutilización y las formas de medición de la integración de los centros para la reutilización.

En esta sección se detallarán los procesos aplicados en la institución a partir de la adaptación realizada de los definidos en la metodología, de los que debido a su importancia e impacto, fueron priorizados y movidos a un nivel más global del que se encuentra definido en la metodología.

Los elementos de apoyo a la reutilización y las formas de medición serán explicados en pasos más adelante.

Una precondición latente en cada proceso, es que todas las entidades involucradas en la red de reutilización deben conocer cómo es el trabajo con el repositorio de activos de software; es decir, conocer los procesos y responsabilidades.

Para ello, la institución estableció un código de ética que deberá ser firmado por cada miembro antes de acceder al repositorio. Sólo después de la firma del acta de responsabilidad, le serán activados a los usuarios, los privilegios de acceso al

repositorio de activos de software. En la sección Anexos se puede ver imágenes referentes al acta de responsabilidad del código de ética en "Acta del código de ética".

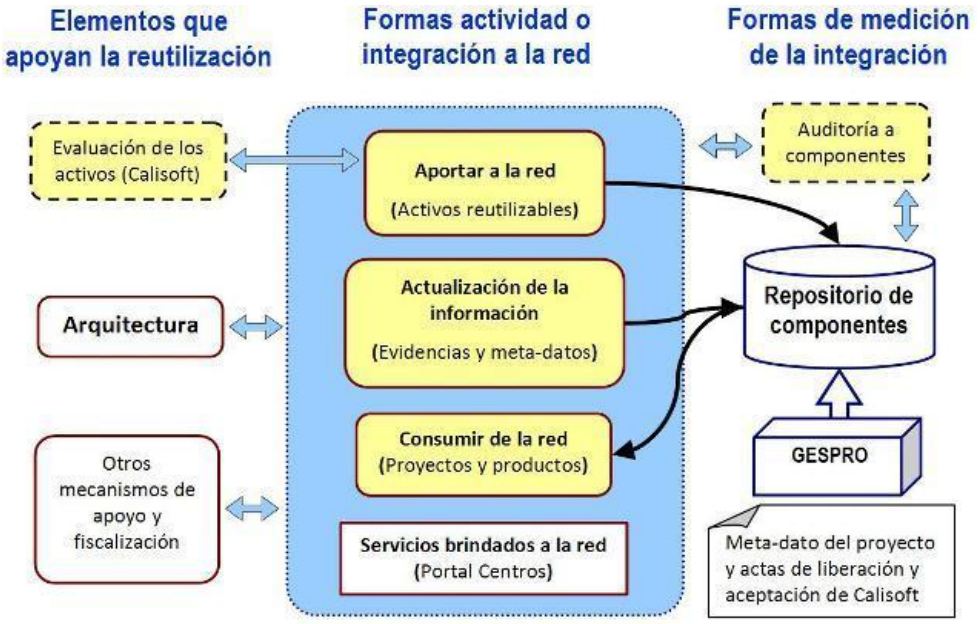


Figura 30: Entorno de reutilización aplicado en la UCI.

Los procesos centrados en la reutilización aplicados a la UCI están descritos de forma detallada en la sección Anexos, en "Procesos de reutilización aplicados en la UCI".
En la siguiente tabla se muestra la adaptación de los mismos tomando como referencia los procesos definidos en la metodología propuesta.

Tabla 7: Adaptación de los procesos.

Procesos definidos en la Metodología		**Procesos adaptados de la Metodología**	
Procesos	**Subprocesos**	**Procesos**	**Subprocesos**
Aportar a la red	Evaluación	Evaluación de los activos	
	Publicación		
	Adaptación	Aportar a la red	
	Integración		

	Actualización	Actualización de la información	
Consumir de la red	Búsqueda	Consumir y buscar de la red	
	Selección		
Gestionar repositorio	Monitoreo Políticas de seguridad y acceso Código de ética Sistema de recompensas	Gestionar repositorio	Auditoría de activos Políticas de seguridad y acceso Código de ética Sistema de recompensas

Las adaptaciones realizadas a la metodología en este paso se corresponden con la necesidad de priorizar algunos subprocesos dentro del caso de la UCI y dada la importancia de los mismos y el incipiente desarrollo de la organización en cuanto a la reutilización.

3.1.2. Paso 2 detallado: Definir qué tipo de activos, qué características y qué comportamiento tienen.

- Documentación
- Componentes no COTS
- Componente COTS

3.1.3. Paso 3 detallado: En función de los tipos de activos, revisar la información que se registrará para su especificación.

Los formatos establecidos para la documentación a publicar tanto dentro del empaquetado como adjunto al activo son:

- PDF
- .RTF
- .DOC
- .ODT

Se definió que cada activo debidamente empaquetado y compactado (utilizando solo formato zip) contiene un conjunto de documentos tanto obligatorios como opcionales, los cuales de describen en la sección “Empaquetado de los activos” de los Anexos.

3.1.4. Paso 4 detallado: En función de los tipos de activos, revisar los factores de calidad que se tendrán en cuenta para su evaluación y medición del nivel de calidad.

Los criterios de calidad están documentados en el expediente de proyecto definido en la UCI. [3]

3.1.5. Paso 5 detallado: Diseñar un almacén físico capaz de recoger como meta-dato la información que se definió para la documentación de los activos, tener capacidad de compresión, tener cierto nivel de seguridad contra los accesos no autorizados.

La información que se registra en el almacén físico o repositorio al ser publicado un activo se puede ver dentro de los Anexos en la sección “Información de meta datos asociado a los activos”.

Para la restricción del acceso al almacén físico, se adicionó un módulo de autenticación que autoriza o deniega según los datos que captura, si son correctos y pertenecen a algún usuario registrado, autoriza, de lo contrario, rechaza el acceso.

Para garantizar un adecuado uso del repositorio y realización de los procesos que se relacionan con él, se exige para cada usuario que necesite de publicación o descarga de algún activo, la firma del Acta de Responsabilidad, donde se incluye el código de ética definido por la entidad publicadora (Dirección Técnica) con apoyo de la Alta Dirección.

3.1.6. Paso 6 detallado: Diseñar una infraestructura de soporte.

Para la infraestructura de soporte para la red de reutilización en la UCI fue definido:

- Una herramienta para realizar todas las actividades de ingeniería, de gestión y de soporte de los proyectos.
- Un almacén físico de activos.
- Un generador de reportes para facilitar el análisis de las actividades de reutilización en la red de centros.
- Un gestor de BD para el almacenamiento de todos los activos y la documentación del empaquetado de los activos de los centros.
- Un modelo de calidad para la evaluación y certificación de todos los productos desarrollados en los centros.

- Departamento que gestione y den soporte a las no conformidades.

3.1.7. Paso 7 detallado: Identificar los entes que participarán en la red.

Fueron identificados como entes que participarán en la red, los mismos que fueron definidos en la metodología:

- Entidad proveedora
- Entidad cliente
- Entidad publicadora
- Entidad evaluadora

3.1.8. Paso 8 detallado: Definir mecanismos de apoyo y estimulación.

Los mecanismos de apoyo y estimulación actúan como herramientas de control a los diferentes niveles de la red: a nivel general, a nivel de centro y a nivel de activos de software. También actúan sobre el proceso de desarrollo al establecer políticas que favorecen la reutilización, quedando como mecanismos:

- El sistema de recompensas tanto individuales como a nivel de centro.
- Emitir informe de fortaleza de la red de centros.
- Emitir informe de nivel de actividad o de integración de los centros en la red.
- Emitir informe de resultados de la auditoría a los activos.
- Emitir políticas que favorezcan la utilización de los activos publicados.
  - → Documento de arquitectura
  - → Restringir los valores de los campos en el GESPRO.
- Control y seguimiento de la reutilización en los consejos técnicos.

3.2. Aplicación de la metodología para la implantación del repositorio de activos en el entorno de la red de centros de la UCI

Descripción detallada de los pasos seguidos para la implantación de un repositorio:

3.2.1. Paso 1 detallado: Identificar a quiénes se les asignará la responsabilidad de los entes definidos y establecer pautas para garantizar su cumplimiento.

La siguiente tabla muestra los departamentos a quienes se les asignó su

responsabilidad según las tareas que realiza.

Tabla 8: Relación de departamentos por rol

Roles del Modelo Propuesto	Departamento por roles
Entidad proveedora	Ente/Centros
Entidad cliente	Ente/Centros
Entidad publicadora	Dirección Técnica
Entidad evaluadora	Calisoft

3.2.2. Paso 2 detallado: Crear una infraestructura de soporte.

Herramienta para realizar todas las actividades de ingeniería, de gestión y de soporte de los proyectos.	Fue creado y desplegado en todos los centros de l UCI el GESPRO.
Almacén físico de activos.	Se creó un plug-in para integrar el almacén físico de activos al GESPRO.
Generador de reportes para facilitar el análisis de las actividades de reutilización en la red de centros.	Se creó un plug-in para integrar un gestor de reportes al GESPRO.
Gestor de BD basado en postgre para el almacenamiento de todos los activos y la documentación del empaquetado de los activos de los centros.	El GESPRO es soportado por un gestor de BD cubano, creado enla UCI basado en postgre que permite gestionar todos los servicios ofrecidos por la universidad.
Modelo de calidad para la evaluación y certificación de todos los productos desarrollados en los centros.	Se aplica en todos los centros, el modelo de calidad definido por Calisoft basado en una adaptación del CMMI, donde cada proyecto debe llenar un “Expediente de Proyecto” donde se registren todas las actividades y artefactos del producto, que incluye hasta la fase de soporte.
Laboratorios para el soporte de los servicios ofrecidos por la Dirección Técnica.	La DT cuenta con 2 laboratorios de 10 computadoras cada uno para el soporte de los servicios que ofrece.
Departamento que gestione y den soporte a las no conformidades.	La DT cuenta con personal capacitado para dar soporte a las no conformidades detectadas con el repositorio.

El GESPRO genera los reportes necesarios para el análisis del estado de la reutilización de la red como:

- Activos más reutilizados
- Activos menos reutilizados
- Centros que más publican activos
- Centros que menos publican activos
- Centros que más reutilizan
- Centros que menos reutilizan
- Centros que más publican y reutilizan activos
- Centros que menos publican y reutilizan activos (parásitos de la red)

El GESPRO es una herramienta de integración a la cual los centros pueden acceder no sólo al almacén físico, sino también al entorno de trabajo creado para cada proyecto dentro de cada centro y a la sección de publicación de los artefactos generados en las actividades de gestión (Alfresco[2]), que contiene lo que registra cada proyecto en el Expediente de Proyecto.

3.2.3. Paso 3 detallado: Implementar un almacén físico de activos.

Fue creado un repositorio adicionado como plug-in a la herramienta GESPRO, donde se almacenan un conjunto de informaciones como meta-dato, la misma se muestra en la sección "Información registrada en el repositorio de activos (meta-dato)" en los Anexos.

3.2.4. Paso 4 detallado: Establecer pautas para una correcta especificación de los activos y evaluación para su certificación en función de los intereses de la organización.

Se estableció:

- Como línea base la documentación de la arquitectura de los activos de software, la cual se encuentra bien detallada en el expediente de arquitectura.
- Como precondición para poder participar en la red de reutilización, la firma del acta de responsabilidad del código de ética establecido por la universidad.
- Como precondición para poder publicar activos al repositorio, presentar el acta de certificación otorgado por Calisoft luego de la revisión del activo.

3.2.5. Paso 5 detallado: Establecer e implementar los mecanismos de apoyo y

[2] Alfresco: gestor de documentación.

estimulación establecidos.

Se estableció un sistema de recompensas como parte de los mecanismos de apoyo y fiscalización para fomentar la reutilización en la UCI, el cual cuenta con varias dimensiones como se muestra en la sección "Sistema de recompensas" de los Anexos.

3.3. Resultados finales de la aplicación

- Se instauró para toda la red de centros de la universidad, un repositorio de activos de software para fomentar la reutilización, en conjunto con un código de ética para la legalización de las responsabilidades de cada centro.
- Se definió como meta-dato de cada activo de software un conjunto de informaciones técnicas que detallan las funcionalidades y características internas, de mercado y de empaquetado del activo.
- Fueron definidos criterios de calidad para la evaluación de los activos. Los mismos fueron incluidos en el expediente de arquitectura de cada activo para garantizar una arquitectura centrada en la calidad y así establecer durante todas las fases de desarrollo del activo, pautas para la calidad total.

Tabla 9: Resumen de análisis de la variable dependiente.

Variable dependiente	Indicador	Subindicador	Unidad de medida
Calidad del proceso de reutilización de activos de software en la red de centros.	Evolutivo	Evoluciona en el tiempo	Sí
		Sistema de autoaprendizaje	Sí
	Sistémico	Integración de los procesos	Sí
	Nivel de respuesta	Nivel de respuesta	Proactivo
	Participativo	Participación de todos los miembros	Sí

Tabla 10: Resumen de análisis de las variables independientes.

Variables	Indicador	Subindicador	Resultados

independientes			**finales de evaluación de las variables**
Metodología para el diseño y explotación de repositorios de activos de software reutilizables.	Modelo para documentar activos de software	Calidad	Alta
	Procesos	Sencillos	Sí
		Reutilizables en otros dominios de aplicación	Sí
Sistema de información para la gestión de repositorios.	Repositorio de activos de software	Facilidad de uso	Sí
	Facilidad de almacenamiento	Compresión de la información.	Sí
		Facilidad de actualización.	Sí
		Nivel de protección	Sí
	Facilidad de búsqueda de información	Capacidad de recuperación	Sí

De esta forma se demuestra que los objetivos de la investigación fueron cumplidos satisfactoriamente.

Conclusiones Finales

- Teniendo como referencia el estudio del estado del arte realizado a los modelos y métodos definidos para la ingeniería y desarrollo de software basado en reutilización, se observó que la mayoría de los modelos analizados definen procesos para la reutilización de activos tanto en entornos de aplicación como en entorno de dominio, establecen la necesidad de contar con un repositorio o biblioteca para el almacenamiento de los activos o componentes desarrollados y en algunos casos se proponen algunos criterios de calidad. Sin embargo, en la mayoría de los casos presentan las siguientes limitantes:
  - → no especifican los procesos relacionados con la gestión del repositorio de activos de software;
  - → no establecen sistemas de recompensas para impulsar la reutilización;
  - → no definen mecanismos para la seguridad y control de acceso al repositorio;
  - → no disponen de un código de ética para la solvencia moral y conciencia de control en el escenario de reutilización;
  - → ni definen principios por los que se debe regir la metodología.
- Se propuso una nueva metodología para el diseño e implantación de un repositorio de activos de software para la reutilización que cumple como principio, ser proactiva, sistémica, participativa, centrada en la calidad y con involucramiento de la alta gerencia.
- Como parte de la metodología se proponen un grupo de procesos que abarcan las actividades relacionadas con el diseño e implantación de repositorio de activos para la reutilización. Estos procesos se basan fundamentalmente en los conceptos del SEI sobre las LPS, los métodos WATCH de Montilva y TWIN de Samentinger y la gestión de proyectos.
- Como parte de la metodología también se propone un modelo para la especificación de los activos basado en el modelo de documentación de componentes COTS propuesto por Iribarne.
- Como parte de la metodología se sugiere la definición de 4 entidades

involucradas en la reutilización: entidad cliente, proveedora, evaluadora y publicadora.

- También se propone el empleo de una herramienta de soporte para el repositorio de activos y toda la actividad de gestión.
- Además se definen otros elementos en la metodología usados como mecanismos de apoyo a la reutilización como:
  - → Acta de certificación que garantice la certificación de los activos antes de su publicación en el repositorio.
  - → Código de Ética profesional que establezca normas generales de conductas para la reutilización en la red, como garantía de una solvencia moral y fortalecimiento de conciencia de control en el uso del repositorio.
  - → Sistema de recompensas para el control de la reutilización, como herramienta para la clasificación de activos más y menos reutilizados y entes que más y menos aportan al entorno de reutilización.
- El modelo propuesto se aplicó en la red de centros de producción de la Universidad de las Ciencias Informáticas (UCI) demostrándose que:
  - → la metodología logró potenciar la publicación de activos, y con ello su reutilización en la red colaborativa de la UCI;
  - → cumple con los principios proactividad, sistémica, evolutiva, participativa, centrada en la calidad y con un alto involucramiento de la alta gerencia;
  - → contribuyó a aumentar la calidad de la especificación de los activos publicados, creando un entorno favorable a la aplicación de modelos de reutilización.

Recomendaciones

La metodología propuesta puede ser mejorada a partir de:

- La definición de mecanismos para demostrar el grado de aporte directo a la productividad;
- Contar con un método rápido y efectivo para la integración continua de los activos publicados en el repositorio.
- Poder disponer de herramientas para la automatización de:
  - → actualización de los activos publicados en el repositorio,
  - → las pruebas de calidad que se le realicen a los activos y
  - → la búsqueda de activos en el repositorio (mediador)
- Definir mecanismos para la colección de los activos de software publicados en el repositorio para fomentar la creación de ecosistemas.

Bibliografía

[1] J.A. Montilva, “Desarrollo de Software Basado en Componentes,” Santa Cruz, Bolivia.: XIII Asamblea General del ISTEC, 2003, p. 36.

[2] F.A. Fernández, “Consideraciones sobre el desarrollo basado en componentes en la Red Cubana de Ciencia,” *CITMATEL*, 2008, p. 20.

[3] J. Sametinger, *Software Engineering with Reusable Components*, Berlin Heidelberg New York London Paris Tokyo Hong Kong Barcelona Budapest: Springer-Verlag Berlin Heidelberg, 1997.

[4] P. Niranjan and G. Rao, “A mock-up tool for software component reuse repository,” *IJSEA*, vol. Vol. 1, No. 2, Abril. 2010, p. 12.

[5] J.A. Montilva, “Desarrollo de Software Basado en Líneas de Productos de Software,” Universidad de Los Andes. Facultad de Ingeniería. Departamento de Computación. Mérida – Venezuela.: IEEE Computer Society Región 9, 2006, p. 67.

[6] M.R. Danish and S.A. Khan, “Component Repository Browser,” Tesis de grado, Mälardalen University, School of Innovation, Design and Engineering, 2010.

[7] C. Canal, “Un Lenguaje para la Especificación y Validación de Arquitecturas de Software,” Tesis Doctoral, Universidad Málaga, del Área de Lenguajes y Sistemas Informáticos, 2000.

[8] J.A. Montilva, J. Barrios, and Vanessa Hamar, “Aspectos metodológicos del desarrollo de componentes de software reutilizable,” Guadalajara, México: CEISOFT, 2004, p. 60.

[9] J. Li, “Process Improvement and Risk Management in Off-The-Shelf Component-Based Development.,” *NTNU Trykk, Trondheim,* Jun. 2006.

[10] J. Li, R. Conradi, P. Mohagheghi, O.A. Sæhle, Ø. Wang, E. Naalsund, and O.A. Walseth, “A Study of Developer Attitude to Component Reuse in Three IT Companies,” 2006, p. 15.

[11] J. Li, F.O. Bjørnson, R. Conradi, and V.B. Kampenes, “An Empirical Study of Variations in COTS-based Software Development Processes in the Norwegian IT Industry,” 2006, p. 31.

[12] J. Sametinger, “Component Interoperation,” *Johannes Kepler University*, 1997, p. 5.

[13] A. Brown and K.C. Wallnau, “Engineering of Component-Based Systems,” IEEE Computer Society Press, 1996.

[14] F.J.G. Peñalvo, J.M.M. Corral, and J.M.M. Raedo, “Mecano: Una propuesta de componente de software reutilizable,” *II Jornadas de Ingeniería del Software, Donostia-San Sebastián,* Sep. 1997, p. 13.

[15] N. Méndez and M. Elvia, “Modelo de evaluación de metodologías para el desarrollo

de software," Tesis de grado, Universidad de Caracas, 2006.

[16] William Anderson, Ed Morris, Dennis Smith, and Mary Catherine Ward, *COTS and Reusable Software Management Planning: A Template for Life-Cycle Management,* 2007.

[17] M.A. Rojas and J.C.M. García, "Introducción y principios básicos del desarrollo de Software basado en componentes," Sep. 2004, p. 12.

[18] J.M. Torío, "Estrategia de prueba de líneas de productos de sistemas de tiempo real especificados con diagramas de estados jerárquicos," Tesis Doctoral, UNIVERSIDAD POLITÉCNICA DE MADRID, 2004.

[19] C.W. Krueger, "Towards a Taxonomy for Software Product Lines," 2003, p. 11.

[20] C.W. Krueger, "New Methods in Software Product Line Development," *IEEE*, 2006, p. 99.

[21] C.W. Krueger, "The 3-Tiered Methodology: Pragmatic Insights from New Generation Software Product LinesThe 3-Tiered Methodology: Pragmatic Insights from New Generation Software Product Lines. ," 2007, p. 106.

[22] C.W. Krueger, "Catalysts and Inhibitors for Momentum in the Software Product Line Industry," *12th International Software Product Line Conference*, 2008.

[23] O. Hummel, "Semantic Component Retrieval in Software Engineering," Tesis Doctoral, Universität Mannheim, 2008.

[24] R.S. Pressman, *Ingeniería del Software. Un enfoque práctico,* McGraw-Hill, 2002.

[25] P.Y. Piñero, "Introducción a los modelos de desarrollo de software," Jul. 2008.

[26] C. Albert and L. Brownsword, "Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview. Key Elements in Building, Fielding, and Supporting Commercial-off-the-Shelf (COTS) Based Solutions," *CMU/SEI-2002-TR-009*, Jul. 2002, p. 64.

[27] L. Bass, P. Clements, S. Cohen, L. Northrop, and J. Withey, "Product Line Practice Workshop Report," *CMU/SEI-97-TR-003*, Jun. 1997, p. 46.

[28] J. Bergey, G. Campbell, P. Clements, S. Cohen, L. Jones, R. Krut, L. Northrop, and D. Smith, "Second DoD Product Line Practice Workshop Report," *CMU/SEI-99-TR-015*, Oct. 1999, p. 79.

[29] I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems*, Artech House, 2002.

[30] L.F. Iribarne, "Un Modelo de Mediación para el Desarrollo de Software basado en Componentes COTS," Tesis Doctoral, Departamento de Lenguajes y Computación Universidad de Almería, 2003.

[31] I. Sommerville, *Ingeniería de Software*, Madrid, España: Addison Wesley, 2005.

[32] O. Hernández, J. Octavio, and C. Verdin, "Documentando Arquitecturas Orientadas

a Aspectos para Líneas de Productos de Software," 2008, p. 6.

[33] M. Vázquez, "Definición de una arquitectura de referencia para una línea de productos de software," Tesis de Maestría, Universidad de las Ciencias Informáticas (UCI), 2011.

[34] T. Käkölä and J.C. Dueñas, *Software Product Lines. Research Issues in Engineering and Management,* Jyväskylä, Finlandia: 2006.

[35] P.C. Clements, L.G. Jones, L.M. Northrop, and J.D. McGregor, "Project Management in a Software Product Line Organization," *Published by the IEEE Computer Society*, Oct. 2005, p. 9.

[36] S.A. Ajila, "Change Management: Modeling Software Product Lines Evolution," 2005, p. 6.

[37] K. Pohl, G. Böckle, and F.V.D. Linden, *Software Product Line Engineering. Foundations, Principles, and Techniques,* Printed in Germany: Springer-Verlag Berlin Heidelberg, 2005.

[38] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns,* Addison-wesley Professional, 2001.

[39] H. Pestano, "Propuesta de modelo de desarrollo para líneas de productos de software en centros de producción," Tesis de Maestría, Universidad de las Ciencias Informáticas (UCI), 2011.

[40] J. Li, M. Torchiano, R. Conradi, P.N. Slyngstad, and C. Bunse, "A State-of-the-Practice Survey of Off-the-Shelf Component-Based Development Processes," 2006, p. 17.

[41] G.J. Chastek, P. Donohoe, and J.D. McGregor, "Formulation of a Production Strategy for a Software Product Line," *SEI: Software Engineering Institute*, Aug. 2009, p. 39.

[42] O.S.G. Gómez, "Integración continua en componentes EJB," vol. Versión 1.0, Enero. 2008, p. 17.

[43] J. Li, R. Conradi, P.N. Slyngstad, C. Bunse, and U. Khan, "Validation of New Theses on Off-The-Shelf Component Based Development," 2006, p. 17.

[44] Yunwen Ye and Gerhard Fischer, "Context-Aware Browsing of Large Component Repositories," *IEEE ASE'01*, Nov. 2001, p. 8.

[45] M. Mecella, B. Pernici, M. Rossi, and A. Testi, "A Repository of Workflow Components for Cooperative e-Applications," 2003, p. 20.

[46] M. Gaedke, J. Rehse, and G. Graef, "A Repository to facilitate Reuse in Component-Based Web Engineering," *8th International World-Wide Web Conference (WWW8)*, May. 1999, p. 7.

[47] Y. Crespo, M.Á. Laguna, and F.J. Pérez, “Integrando un modelo de reutilización en la producción de software: entorno distribuido para el desarrollo basado en reutilización,” 2004, p. 15.

[48] S. Spring, “Introduction to repository management.,” *Sonatype*, 2009, p. 9.

[49] N. Rodrigues and L. Barbosa, “On the Specification of a Component Repository,” 2004, p. 9.

[50] A. Egyed, N. Medvidovic, and C. Gacek, “A Component-Based Perspective on Software Mismatch Detection and Resolution,” 2004, p. 28.

[51] F. Navarrete, P. Botella, and X. Franch, “Análisis de los Métodos de Selección de Componentes COTS desde una Perspectiva Ágil,” 2005, p. 9.

[52] M.F. Bertoa and A. Vallecillo, “Atributos de Calidad para Componentes COTS,” 2006, p. 12.

[53] M.F. Bertoa and A. Vallecillo, “Medidas de Usabilidad de Componentes Software,” *IEEE Latin American Transactions*, vol. VOL. 4, Abril. 2006, p. 8.

[54] C. Becker and A. Rauber, *Improving component selection and monitoring with controlled experimentation and automated measurements* , Austria: 2010.

[55] M. Palviainen, A. Evesti, and E. Ovaska, “The Reliability Estimation, Prediction and Measuring of Component-Based Software,” Enero. 2011.

[56] R. Roshandel, S. Banerjee, L. Cheung, N. Medvidovic, and L. Golubchik, “Estimating Software Component Reliability by Leveraging Architectural Models,” *ICSE'06*, May. 2006, p. 4.

[57] L. Iribarne, J.M. Troya, and A. Vallecillo, “Trading for COTS components in Open Environments,” 2001, p. 9.

[58] C. Rahmani, A. Azadmanesh, and L. Najjar, “A Comparative Analysis of Open Source Software Reliability” *JOURNAL OF SOFTWARE*, vol. VOL. 5, NO. 12, Diciembre. 2010.

[59] V. Goswami and Y.B.Acharya, “Method for Reliability Estimation of COTS Components based Software Systems”, 2009, p. 4.

[60] P.Y. Piñero, “Diseño de Fiabilidad para sistemas software,” 2007.

[61] D. Zubrow and G. Chastek, “Measures for Software Product Lines. Software Engineering Measurement and Analysis Initiative,” *CMU/SEI-2003-TN-031*, Oct. 2003, p. 35.

[62] R. Lazo, M. Jorrín, and P.Y. Piñero, “Expediente de arquitectura de referencia para sistemas de información,” 2011.

[63] M.F. Lungu, “Reverse Engineering Software Ecosystems,” Tesis Doctoral, Faculty of Informatics of the University of Lugano, 2009.

[64] *Paquete de Gestión de Proyectos*, Dirección Técnica. Universidad de las Ciencias

Informáticas: Universidad de las Ciencias Informáticas (UCI).

[65] Jan Bosch, “From Software Product Lines to Software Ecosystems,” 2009, p. 10

[66] J.D. McGregor, “Ecosystems, continued,” ETH Zurich, Chair of Software Engineering, 2009, p. 15.

[67] Charles W. Krueger, “New Methods in Software Product Line Development,” *IEEE*, 2006, p. 5.

[68] Gerald Goh Guan and Wong Kim Yen, “Software component reuse in information systems development: a review of challenges and strategies,” *Journal of Han Chiang College*, vol. vol. 3, 2005, p. 13.

[69] J.C. Calvo, “Reutilización, parametrización y patrones de diseño,” 2001, p. 15.

[70] A. M. Zaremski and J. M. Wing, “Specification matching of software components,” 3ra ACM SIGSOFT Symposium on the Fundations on Software Engineering, 1995.

[71] H. Mili, “Reusing software: Issues and research directions,” IEEE Transactions on Software Engineering, 1995.

[72] J.Q. Ning, “Component-based software engineering: CBSE enabling technologies,” 4th. Conference on Software Reuse. IEEE Computer Society, 1996.

[73] A. Schappert, P. Sommerlad, and W. Pree, “Automated suppord for software development with frameworks,” ACM CIGSOFT Symposium on Software Reusability, 1995.

[74] R.T. Monrroe and D. Garlan, “Style-based reuse for software architectures,” 4th International Conference on Software Reuse, 1996.

[75] C.W. Krueger, “Software reuse,” *ACM Computing Surveys*, Jun. 1992.

[76] P. Shireesha and V.N. Sharma, “Building Reusable Software Component for Optimization Check in ABAP Coding .,”*International Journal of Software Engineering & Applications (IJSEA)*, vol. Vol.1, No.3, Jul. 2010, p. 9.

[77] B.H. Cheng and J. Jeng, “Formal Methods Applied to Reuse,” *Department of Computer Science Michigan State University*, 2001.

[78] J. Jeng and B.H.C. Cheng, “Using Formal Methods to Construct a Software Component Library,” *ESEC '93 Proceedings of the 4th European Software Engineering Conference on Software Engineering* , Sep. 1993.

[79] J. Jeng and B.H.C. Cheng, “Using Formal Methods to Construct a Software Component Library,” Springer-Verlag London, 1993.

[80] C. Lung, G.T. Mackulak, and J.E. Urban, “Software Reuse and Knowledge Transfer through Analogy and Design Patterns,” 2003, p. 11.

[81] N. Maiden and A. Sutcliffe, “Exploiting reusable specifications through analogy,” *Magazine*, vol. 35, Abril. 1992.

[82] C.W. Krueger, “New methods in Software Product Line practice. Examining the

benefits of next-generation SPL methods.," *Communications of the ACM*, vol. Vol. 49 No. 12, Diciembre. 2006, p. 40.

[83] H. Koziolek, "Performance evaluation of component-based software systems: A survey ," *Elsevier*, vol. 67, Augusto. 2009, pp. 634-658.

[84] I. Groher, C.W. Krueger, and C. Schwanninger, "A Tool-Based Approach to Managing Crosscutting Feature Implementations," *AOSD '08*, Abril. 2008.

[85] S.A. Ajila, "Change Management: Modeling Software Product Lines Evolution," *Department of Systems & Computer Engineering, Carleton University*, 2005, p. 6.

Anexos

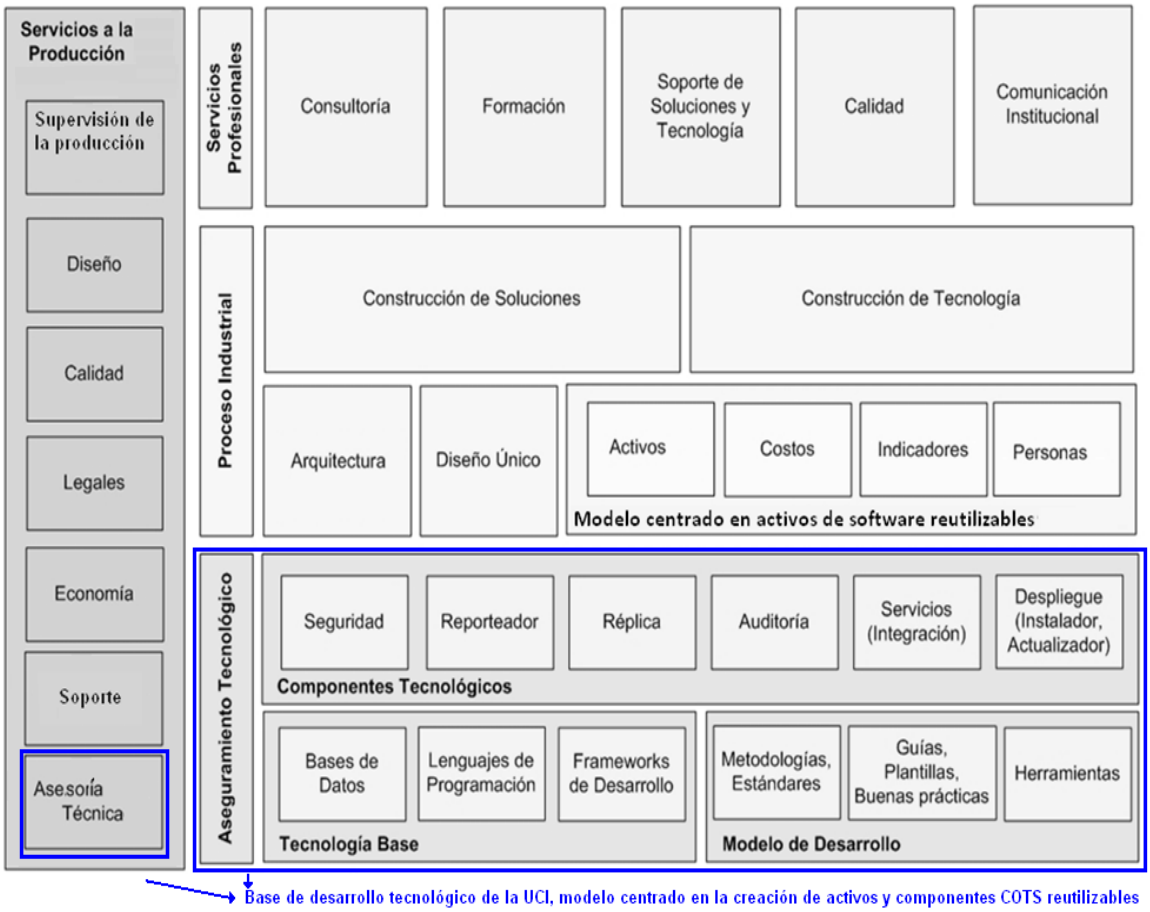


Figura 31: Modelo que representa en una única vista relacionada con el desarrollo tecnológico las interrelaciones entre las entidades.

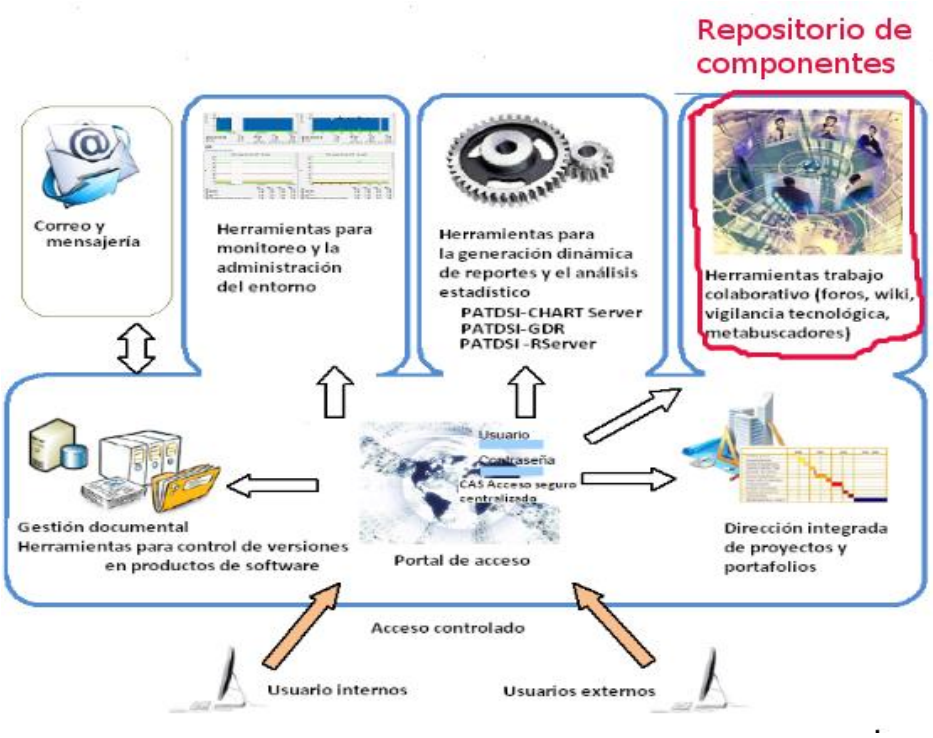


Figura 32: Paquetes de Gestión de Proyectos GESPRO.

Procesos de reutilización aplicados en la UCI

Proceso 1: Aportar a la red:

Proceso mediante el cual el centro comparte, a través del repositorio, el resultado de una inversión de recursos y tiempo en forma de componente reutilizable o servicio evitando el re-trabajo.

Los centros pueden estar aportando a la red: incorporando los componentes haciendo

uso del servicio de la Dirección Técnica (DT), sin que para ello medie la evaluación del componente, este proceso realizado por Calisoft solo variará los meta-datos cambiando el nivel de madurez del componente a Liberado por Calisoft, después de la prueba de aceptación del producto por el cliente Calisoft puede alcanzar nivel de desplegado y muy difundido cuando el uso sobrepasa los 10 clientes.

El proceso “Aportar a la red” incluye cuatro subprocesos:

- 1.1. Identificar el componente a aportar, desacoplarlo y hacer un empaquetado.
- 1.2. Desacoplar la documentación del componente e integrarla al empaquetado.
- 1.3. Recolectar la información esencial que describe al componente
- 1.4. Acceder al servicio de la DT y llenar los meta datos asociados con el componente. Para conocer la información que se recoge en los meta datos asociados al componente, ver la sección Anexos.

Proceso 2: Evaluación de componente:

Este proceso realizado por Calisoft es previo al registro del componente de software o activo en general dentro del Repositorio de Activos Reutilizables. Como evidencia de este proceso el designado por cada ente a interactuar con el repositorio adjuntará dentro de la documentación del activo el acta de liberación de Calisoft (donde refleje claramente el uso del activo en el producto) sin este documento no podrá registrarse en el repositorio.

Después de las pruebas de aceptación del producto por el cliente y Calisoft puede alcanzar nivel de desplegado y muy difundido cuando el uso sobrepasa los 10 clientes. Pero debe actualizar el activo registrado adjuntando las actas de aceptación correspondientes donde refleje claramente el uso del activo en el despliegue.

Describe 5 subprocesos fundamentales:

- 2.1. Revisar la documentación asociada al componentes
- 2.2. Establecer los atributos de calidad a ser evaluados
- 2.3. Aplicar las pruebas según el paso anterior por parte del equipo de desarrollo
- 2.4. Emitir evaluación de calidad por parte de calisoft
- 2.5. Actualizar meta-datos

Proceso 3: Consumir de la red:

Proceso que permite, después de la búsqueda, realizar el consumo de componentes del repositorio o servicios de la red de centros, evidenciado en los meta-datos del proyecto y las actas de liberación y aceptación de Calisoft.

El proceso de consumo se inicia con la búsqueda e identificación de los componentes necesarios para el desarrollo. Es independiente al resto de los procesos. Radica en que los meta-datos brindan los elementos necesarios para la toma de decisión, donde un elemento de peso es el nivel de madurez del componente. El proceso de consumo basa está evidenciado en:

- los meta-datos de los proyectos en ejecución,
- las actas de Liberación y Aceptación del software y
- en los registros automáticos de los servicios brindados a la red.

Describe 6 subprocesos fundamentales:

- 3.1. Buscar (nivel ágil) activos candidatos en el reporte de activos.
- 3.2. Refinar la búsqueda (avanzada) de los activos buscando en los meta-datos.
- 3.3. Seleccionar el activo a utilizar revisando la documentación asociada.
- 3.4. Actualizar los meta-datos del proyecto con la decisión tomada.
- 3.5. Declarar en la solicitud de liberación que se utiliza este activo.
- 3.6. Revisar que en el acta de liberación y aceptación esté declarado el activo utilizado.

  El subproceso Búsqueda de activos permite realizar su búsqueda en el repositorio estableciendo tres niveles:

  - *Ágil*: Basado solo en la descripción. (con reporte asociado)
  - *Avanzado*: Basado en los meta-datos asociados a los activos.
  - *Decisión técnica*: Además estudia la documentación asociada al activo.

Como subproceso importante y derivado de la baja reutilización de los activos, se denomina también **Parasitar la red:** Proceso que identifica a los centros que se limiten al consumo de la red, no intercambien componentes o servicios, o los niveles de intercambios estén por debajo de las políticas de la organización y por lo tanto no contribuyen a la fortaleza de la red de centros por su bajo nivel de actividad o integración.

Proceso 4: Actualización de la información:

Es responsabilidad del centro autor del activo mantener actualizado su activo, según las variaciones en las versiones del mismo, definición del repositorio, actualizar los meta-datos respaldados con evidencias, por ejemplo usando solo el identificador del acta de Liberación o Aceptación del software emitido por Calisoft.

Estos documentos sirven de evidencia en dos sentidos, uno para declarar que un producto usa el activo A: en ese caso se coloca en el activo A el acta de liberación del o los productos que lo usan; en otro sentido, para evidenciar el nivel de madurez del activo se coloca en su acta de liberación para poder colocar el nivel Liberado por Calisoft y así con el resto de los niveles.

Describe 3 subprocesos fundamentales:

- 3.1. Revisar actualización y las variaciones en la definición del repositorio.
- 3.2. Revisar la información publicada.
- 3.3. Actualizar el activo.
  - → 3.3.1. Colocar nueva versión del activo.
    - Comunicarse con la dirección técnica.
    - Tomar una decisión respecto a la versión anterior (ciclo de vida).
  - → 3.3.2. Actualizar meta-datos.
  - → 3.3.3. Colocar la evidencia y actualizar el meta-dato.

Proceso 5: Auditoría a activos:

Proceso que permite detectar incoherencias en los meta-datos, mejorar los procesos relacionados con el repositorio y los servicios, identificar los parásitos de la red y emitir una medida de actividad o integración de los centros.

Busca como metas: detectar incoherencias en los meta-datos respecto a la definición actual del repositorio, encontrar elementos faltantes o desactualizados en el empaquetado. Evaluar la adherencia y emitir acciones correctivas encaminadas a mejorar los procesos relacionados con el repositorio.

Describe 5 subprocesos fundamentales:

- 5.1. Organizar la auditoría y convocar a los participantes
- 5.2. Detectar incoherencias y faltantes en los meta-datos

- 5.3. Evaluar la adherencia a los procesos revisando las evidencias
- 5.4. Emitir informe de acciones correctivas para la mejora de los procesos
- 5.5. Emitir informe de resultados de la auditoría a los activos

Proceso 6: Gestionar repositorio:

Proceso mediante el cual el centro comparte, a través del repositorio, el resultado de una inversión de recursos y tiempo en forma de activo reutilizable o servicio evitando el re-trabajo.

La conformación del repositorio de activos de reutilizables en la red de centros de la UCI es responsabilidad de la DT; por tanto, la interacción con el mismo se establece a través de servicios brindados desde esta dirección con el empleo de la herramienta GESPRO como base. Esta actividad se describe en 5 pasos:

- Paso 1: Cada ente de la red, designa un responsable (o varios) para consumir de la red y aportar a la red.
- Paso 2: Por su importancia desde el punto de vista ético, económico y de seguridad referente a los designados se debe:
  - → Paso 2.1: Enviar comunicación por correo a la DT de los designados en orden de prioridad del ente (Responsabilidad de cada director).
  - → Paso 2.2: Registrar en el portal de la DT cada uno de los designados (Responsabilidad individual de los designados).
  - → Paso 2.3: Ser capacitados y certificados (Responsabilidad de la DT).
  - → Paso 2.4: Responsabilizarse formalmente cada designado y director con la información que van a manejar. Firmar el Acta de Responsabilidad.
    - Paso 2.4.1: Esta acta firmada debe ser entregada a la DT en copia dura.
    - Paso 2.4.2: En caso (baja, cambio de funciones, etc.) que a un designado se le retire por parte del ente este privilegio: es responsabilidad del director del ente comunicarlo oportunamente a la DT y verificar su desactivación
- Paso 3: La DT activa a los usuarios registrados una vez que se tenga física el acta debidamente firmada. También en dependencia de las políticas que se definan en este sentido se limitará el número de designado de un ente de la red.

- Paso 4: Acceder al portal de la DT a través de sus servicios o utilizando el vínculo *Componentes* en el proyecto *Repositorio de Activos Reutilizables.*
- Paso 5: Se debe seguir el procedimiento “Aportar a la red” para publicar un activo para su reutilización. Para ello acceder al proyecto “Repositorio de Activos Reutilizables” de la DT.
  - → Paso 5.1: En la opción “Nuevo activo” se llena la información correspondiente al meta-datos relacionada con el activo como se especifica en el procedimiento “Aportar a la red”.

Información registrada en el repositorio de activos (meta-dato)

- Categoría de Activo
  - → Documentación Usuario/cliente: Especificaciones de requisitos, Especificaciones de pruebas Aceptación.
  - → Documentación Técnica: Especificación de diseño, Algoritmo, Patrón, Arquitectura, Diseño BD, Especificaciones de pruebas (Concepto, Unitarias, Integración, Sistema y otras pruebas internas)
  - → Componentes de Software: Código fuente y especificación técnica, instalador más código fuente y especificación técnica, Componentes COTS
- Arquitectura, establece la ubicación del componente dentro de la vista de tecnología representa el nivel principal de la arquitectura con que se relaciona, si tuviese otras relaciones se colocan en la descripción.
  - → Sistema Operativo
  - → Gestor BD
  - → Tecnología Mapeo Objeto-Relacional (ORM)
  - → Tecnología Réplica de Datos
  - → Tecnología Lenguaje de Programación Nivel Negocio
  - → Tecnología Lenguaje de Programación Bajo Nivel
  - → Tecnología Ambiente Integrado de Desarrollo (IDE)
  - → Tecnología Marco de Trabajo para el Desarrollo (Framework)
  - → Tecnología Marco de Trabajo para Interfaz Usuario (Framework)

- → Tecnología Modelado
- → Tecnología Versionado
- → Tecnología Reportes
- → Tecnología Servidores de Aplicaciones
- → Tecnología Autenticación
- → Tecnología para la protección del código
- → Identificación de requisitos de seguridad del producto
- → Seguridad Capa de presentación
- → Seguridad Nivel de la capa de negocio
- → Seguridad Capa de datos
- → Seguridad del Servidor de aplicación
- → Monitoreo de la seguridad
- → Seguridad en el proceso de desarrollo

- Título, debe dar idea de la utilidad y función del componente.
- Descripción, especificar el domino del componente, detalles del soporte, principales funcionalidades que encierra, forma de utilización del componente. Establece la relación entre el componente y su ubicación dentro de la vista de tecnología. Otros componentes con que se relaciona estrechamente (No. De registro). Nivel de madurez del activo. Línea de producción del ente que respalda su desarrollo y soporte. En caso excepcional de que los valores de los campos no satisfagan su necesidad colocar otros y especificar en la descripción.
- Autores, en el mismo orden que se usará en el registro del CENDA. Se asume el primero como el principal y el resto en orden de aportación.
- Fecha actualización, cuando fue colocado la última evidencia o actualizado un meta-dato.
- Ciudad, es un meta-dato del ente de la red que lo produjo.
- País, es un meta-dato del ente de la red que lo produjo.
- Ente publicador, Centro o ente de la red que hace la publicación y respalda su desarrollo.
- Versión, el número estructurado según lo establecido por la Dirección Técnica.

- Número de registro, Único y otorgado por la Dirección Técnica (DT) para su identificación inequívoca. Es un campo que no debe ser variado a no ser por la DT.
- Palabras claves, Palabras que faciliten su ubicación e identificación
- URL, Vínculo a la página principal del centro o ente de la red que hace la publicación y respalda su desarrollo. También se acepta la página del componente si la tiene.
- Dirección de los autores, en el mismo orden que estos fueron puestos en el campo de autores. Se asume el primero como el principal y el resto en orden de aportación.
- Lenguaje, el principal de los lenguajes utilizados para desarrollar el componente.
- Marca o Nombre corto, Si no tiene definida y autorizada una marca puede usar un nombre corto.
- Público objetivo, para quien está dirigido el componente.
- Principales clientes, proyectos, productos, entidades, centros o estructuras que lo utilizan.
- Framework de Desarrollo (versión), plataforma que se utilizó en el desarrollo del componente, importante especificar la versión.
- COTS, especifica si es o no un componente de grano grueso que puede constituir un producto en sí mismo y por lo tanto comercializable en algún esquema de negocio.
- Gestor de Bases de Datos (versión), especifica gestor de bases de datos que utiliza el componente. En caso de ser varios el principal y especificar, importante especificar la versión.
- Portabilidad (versión): Especificaciones sobre la plataforma sobre la que corre el componente, importante especificar la versión.
- Ficheros: Adjuntar evidencias del uso, registro, certificación y pruebas.
- S.O (versión/distribución): Especificar los detalles sobre el sistema operativo en que corre el componente, importante especificar la versión, distribución, etc.
- Costo asociado: Estimación del costo, que incluya los gastos de tiempo de

máquina (electricidad estimada), consto del empleo de profesionales y estudiantes.

- Extensión de ficheros fuente: Que tipos de ficheros son los utilizados para almacenar el código fuentes.
- Versión de librerías que usa: Especificar detalles sobre las librerías utilizadas.
- Tipo de licencia: Especificar, a partir los elementos utilizados para su desarrollo, bajo que tipo de licencia se pueden comercializar los productos que se deriven él.
- Expiración del soporte: Fecha que representa hasta cuando se extiende el compromiso del ente de la red de centros en darle soporte.
- Notoriedad en productos: Cantidad de productos diferentes que usan este activo.
- Notoriedad en despliegues: Cantidad de clientes diferentes que usan este activo.
- Última actualización base: Última actualización liberada por el suministrador de los elementos usados como base para su desarrollo. Puede ser actualizada durante la auditoría y se consideraría una incongruencia.
- Frecuencia de actualización: Frecuencia con que se considera se actualizará el repositorio con una nueva versión.

Empaquetado de los activos

Como Documentos obligatorios:

- Evidencias de que se llevaron a cabo las pruebas según lo establecido.
  - → *DOCUMENTO 1*: Acta de liberación de Calisoft
- Debe ser presentado completo el documento de especificaciones funcionales, de la arquitectura e incluye el código fuente estable.
  - → *COMPACTADO DE FICHEROS DEL CÓDIGO FUENTE:* Todos los ficheros de código fuente compactados en una solo fichero zip.
  - → *DOCUMENTO 2:* Declaraciones de funcionalidad, los requisitos en el formato que la metodología usada exija.
  - → *DOCUMENTO 3:* La especificación de la arquitectura del activo.
- Presentar documentación de soporte con las necesidades de personal para el

montaje, instalación y prueba de la solución especificado dentro de los manuales. Las competencias del personal involucrado en los diferentes niveles de la aplicación, todos los tipos de usuarios, y del personal de mantenimiento.

→ *DOCUMENTO 4:* Manual de instalación.

→ *DOCUMENTO 5:* Manual de usuario.

Como Documentos opcionales:

- Evidencias de respuestas a preguntas y soluciones a problemas más frecuentes.
  → *DOCUMENTO 6:* Bitácora de preguntas y problemas más frecuentes.

Para el D*OCUMENTO 2*, debe apoyarse en la documentación establecida en el Expediente de Proyecto3 vigente y utilizar al menos el documento para la especificación de requisitos, de forma tal que quede debidamente explicitado que hace el activo y de que forma lo hace.

El *DOCUMENTO 3* o especificación de la arquitectura del activo, contiene la guía y plantilla para la elaboración de la Guía Base de la Arquitectura de Software de los productos y soluciones de la Universidad de las Ciencias Informáticas. Contiene además como anexos al documento principal:

- Estándar de codificación en lenguaje PHP
- Estándar de codificación en lenguaje C++
- Estándar de codificación en lenguaje C#
- Estándar de codificación en lenguaje Python
- Estándar de codificación en lenguaje Java
- Estándar de codificación en lenguaje SQL para Bases de Datos
- Documento guía base para la arquitectura de información de los Sistemas de Información Transaccionales

El *DOCUMENTO 4 o* Manual de instalación es un documento que especifica por pasos cuáles métodos se deben invocar de cada interfaz de servicio que ofrece el activo o componente, y cuáles interfaces debe llamar el activo en caso de requerir de alguna interfaz de entrada por dependencias de bibliotecas u otros servicios. Este documento también debe ser usado para explicar cómo el activo se integra a otros escenarios.

---

[3] Expediente de Proyecto: Libro de registro de todas las actividades de ingeniería, de gestión de proyecto y de soporte que se realizan en el proyecto y entidades que se generan en él.

El *DOCUMENTO 5 o* Manual de usuario es la documentación que genera el equipo de desarrollo para que el usuario pueda entender el modo de operación con el activo. Puede incluir conferencias de capacitación (compactado), o un libro electrónico de ayuda o simplemente un documento o pdf de manual de usuario.

El *DOCUMENTO 6* o Bitácora es donde se registran las soluciones que el equipo de desarrollo del activo le fue dando a los problemas que se fueron presentando durante la operación con el activo por los diferentes usuarios que lo reutilizan. Así mismo, las respuestas a las preguntas más frecuentes.

Tabla 11: Relación de la información técnica definida en la metodología y la gestionada en el repositorio de activos de software de la UCI.

Información técnica propuesta en la metodología		**Información técnica que se gestiona en el repositorio de activos de la UCI.**	**Ubicación**
Información sintáctica		Interfaces	Documento arquitectura: Vista de Procesos
Información semántica	Protocolos		Documento arquitectura: Vista de Procesos
	Contracciones	No se especifican	No están documentadas
	Patrones	Patrones	Documento arquitectura: Vista de Integración Vista de Sistema, Vista de Seguridad y Vista de Datos
Información extra-funcional	Factores de calidad	Frecuencia de actualización:	Meta-dato
		Portabilidad	
		El resto de los factores de calidad	Documento arquitectura: En todas las vistas
Información del mercado	Fabricante	Autores	Meta-dato y Documento arquitectura: Vista de Despliegue
	Precio	Costo asociado	
	licencia	Tipo de licencia	
	Certificado	Acta de certificación de calidad	

	Expiración	Expiración del soporte	
	Notoriedad	Notoriedad en productos Notoriedad en despliegues	
	URL	URL	
Información del repositorio	Identificador	Número de registro	Meta-dato almacenado de forma automática en el repositorio
	Fecha	Fecha actualización	
	Versión	Versión	
	Usuario	Ente publicador	
	Correo	Correo	
	No tiene	Palabras claves	
Información del ensamblado	S.O	S.O	Meta-dato y Documento arquitectura: Vista de Entorno de desarrollo tecnológico
	Procesador	No tiene	
	runtime	No tiene	
	Ficheros	Ficheros	
	Gestor de Bases de Datos	Gestor de Bases de Datos	
	Lenguaje	Lenguaje	
	Framework	Framework de Desarrollo	
	Bibliotecas	Versión de librerías que usa	
	Extensión del Código	Extensión del Código	
	Plataforma (IDE)	IDE de desarrollo	

Sistema de recompensas

- Dimensión nivel personal
  - → Los programas de maestría evaluarán la publicación de activos como parte del sistema de créditos no lectivos. (Ya incluido en la Maestría en Gestión de Proyectos Informáticos).
  - → Los programas de maestría evaluarán la prestación de servicios de

consultoría a la red de centros como parte del sistema de créditos no lectivos avalado por la dirección técnica de la producción. (Ya incluido en la Maestría en Gestión de Proyectos Informáticos).

- → La dirección de información evaluará la emisión de certificados de publicación por la publicación de activos en el repositorio de activos de la UCI como base de conocimiento interna de la UCI.
- → El sistema de formación pregraduada de la universidad evaluará como evidencias para el proceso de formación del estudiante la publicación de activos en el repositorio de activos de software de la UCI.

- Dimensión nivel de los centros.
  - → La dirección de la producción incluirá en la evaluación de los centros su aporte a la red de centros a partir de la publicación de activos en el repositorio de activos de la UCI. (Ya implementado en el seguimiento a la RED de centros).
  - → La dirección de investigaciones incluirá en el análisis de ciencia y técnica de la universidad la introducción de resultados a partir de la publicación de los mismos en el repositorio de activos reutilizables de la UCI y la reutilización de los mismos por la red de centros.
  - → La universidad en la organización del Forum de Ciencia y Técnica reconocerá la publicación de activos en el repositorio de componentes de la UCI como parte de los avales de aplicación y generalización de los resultados.

Acta del código de ética

A continuación se muestran imágenes del código de ética establecido en la UCI para conceder permiso o autorizo de acceso al repositorio de activos reutilizables.

ACCESO AL REPOSITORIO DE ACTIVOS REUTILIZABLES

FORMULARIO DE SOLICITUD DE SERVICIOS DE LA DIRECCIÓN TÉCNICA
UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Día	Mes	Año

Fecha de Solicitud

Siglas ente/centro

Datos Personales *(Usar letra clara y legible. Inhabilitar las casillas que no utilizará)*

1. Primer Apellido	Segundo Apellido	Nombre(s)	Número Carné de Identidad	Firma

Datos del Servicio

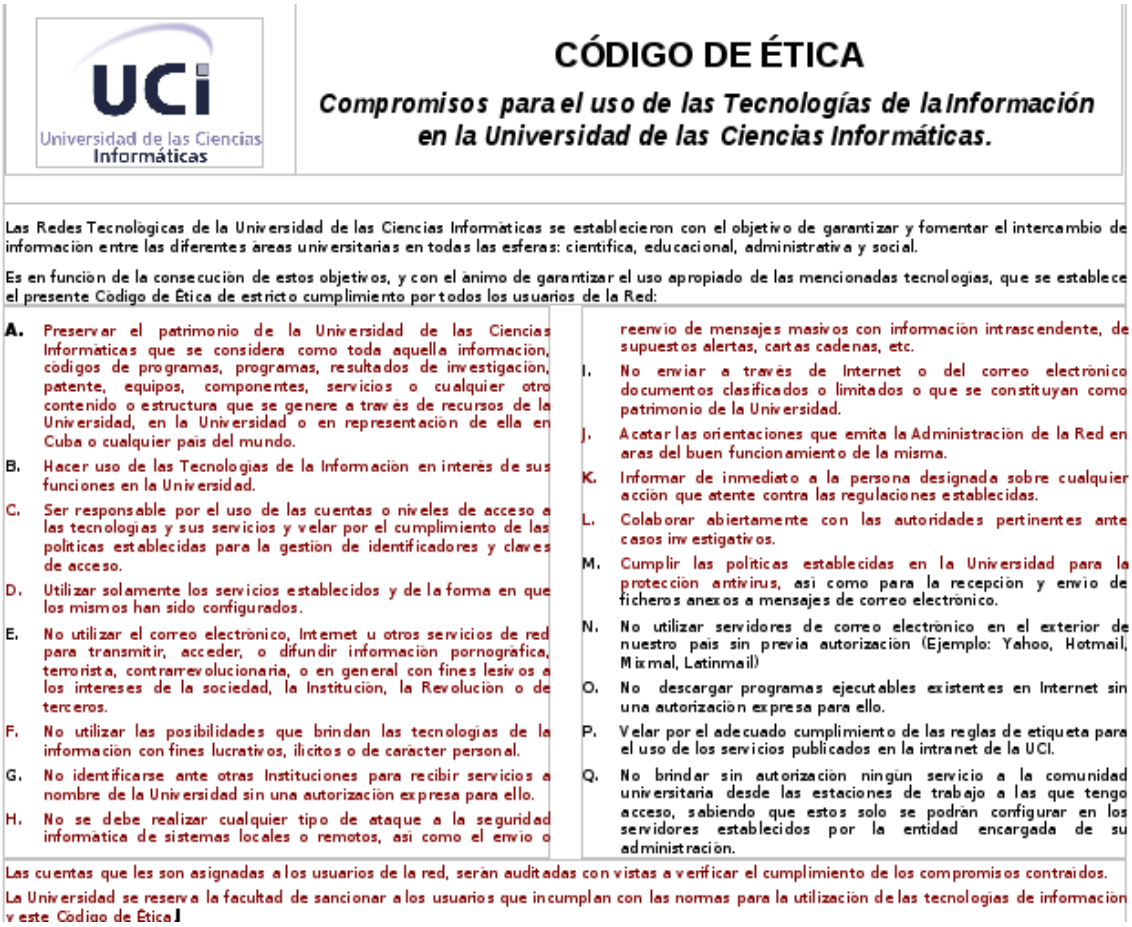
Acciones que pueden desarrollar una ver activados sus privilegios:
• **Búsqueda de componentes**: Para la búsqueda de los componentes en el repositorio se establecen diferentes niveles de búsqueda (**Ágil**: Basado solo en la descripción. **Avanzado**: Basado en los meta-datos asociados a los componentes. **Decisión técnica**: Además estudia la documentación asociada al componente.)
• **Aportar a la red**: Proceso mediante el cual el centro comparte, a través del repositorio, el resultado de una inversión de recursos y tiempo en forma de componente reutilizable o servicio evitando el re-trabajo.
• **Consumir de la red**: Proceso que permite, después de la búsqueda, el consumo de componentes del repositorio o servicios de la red de centros evidenciado en los meta-datos del proyecto y las actas de liberación y aceptación de Calisoft.
• **Parasitar la red**: Proceso que identifica a los centros que se limiten al consumo de la red, no intercambien componentes o servicios, o los niveles de intercambios estén por debajo de las políticas de la organización y por lo tanto no contribuyen a la fortaleza de la red de centros por su bajo nivel de actividad o integración.
• **Auditoría a componentes**: Proceso que permite detectar incoherencias en los meta-datos, mejorar los procesos relacionados con el repositorio y los servicios, identificar los parásitos de la red y emitir una medida de actividad o integración de los centros.

Manifiesto que los datos expuestos en el formulario se ajustan a la realidad. Doy fe de que todos hemos leído y entendido el formulario y el Código de Ética al dorso de éste formulario y que por lo tanto acataremos todo lo expresado por la letra de dicho documento así como también las disposiciones que estipule la Dirección Técnica al respecto. El contenido del formulario regirá en lo adelante el comportamiento del ente como usuario del Repositorio de Activos Reutilizables.

Nos comprometemos además a no divulgar o usar a través de la RED hacia el exterior de la Universidad ningún activo o información confidencial, clasificada, de importancia estratégica o que pueda constituirse patrimonio de la Universidad. Como lo expresan los incisos del A. al L. del código al dorso.

Responsable del ente que autoriza	
Nombre y Apellidos	
Cargo y nombre del ente	
Firma y Cuño	
Director Técnico (IP)	**Ejecutado por**
Firma	Firma

Figura 33: Autorizo para acceso al repositorio de activos.



CÓDIGO DE ÉTICA

Compromisos para el uso de las Tecnologías de la Información en la Universidad de las Ciencias Informáticas.

Las Redes Tecnológicas de la Universidad de las Ciencias Informáticas se establecieron con el objetivo de garantizar y fomentar el intercambio de información entre las diferentes áreas universitarias en todas las esferas: científica, educacional, administrativa y social.

Es en función de la consecución de estos objetivos, y con el ánimo de garantizar el uso apropiado de las mencionadas tecnologías, que se establece el presente Código de Ética de estricto cumplimiento por todos los usuarios de la Red:

A. Preservar el patrimonio de la Universidad de las Ciencias Informáticas que se considera como toda aquella información, códigos de programas, programas, resultados de investigación, patente, equipos, componentes, servicios o cualquier otro contenido o estructura que se genere a través de recursos de la Universidad, en la Universidad o en representación de ella en Cuba o cualquier país del mundo.

B. Hacer uso de las Tecnologías de la Información en interés de sus funciones en la Universidad.

C. Ser responsable por el uso de las cuentas o niveles de acceso a las tecnologías y sus servicios y velar por el cumplimiento de las políticas establecidas para la gestión de identificadores y claves de acceso.

D. Utilizar solamente los servicios establecidos y de la forma en que los mismos han sido configurados.

E. No utilizar el correo electrónico, Internet u otros servicios de red para transmitir, acceder, o difundir información pornográfica, terrorista, contrarrevolucionaria, o en general con fines lesivos a los intereses de la sociedad, la Institución, la Revolución o de terceros.

F. No utilizar las posibilidades que brindan las tecnologías de la información con fines lucrativos, ilícitos o de carácter personal.

G. No identificarse ante otras Instituciones para recibir servicios a nombre de la Universidad sin una autorización expresa para ello.

H. No se debe realizar cualquier tipo de ataque a la seguridad informática de sistemas locales o remotos, así como el envío o reenvío de mensajes masivos con información intrascendente, de supuestos alertas, cartas cadenas, etc.

I. No enviar a través de Internet o del correo electrónico documentos clasificados o limitados o que se constituyan como patrimonio de la Universidad.

J. Acatar las orientaciones que emita la Administración de la Red en aras del buen funcionamiento de la misma.

K. Informar de inmediato a la persona designada sobre cualquier acción que atente contra las regulaciones establecidas.

L. Colaborar abiertamente con las autoridades pertinentes ante casos investigativos.

M. Cumplir las políticas establecidas en la Universidad para la protección antivirus, así como para la recepción y envío de ficheros anexos a mensajes de correo electrónico.

N. No utilizar servidores de correo electrónico en el exterior de nuestro país sin previa autorización (Ejemplo: Yahoo, Hotmail, Mixmal, Latinmail)

O. No descargar programas ejecutables existentes en Internet sin una autorización expresa para ello.

P. Velar por el adecuado cumplimiento de las reglas de etiqueta para el uso de los servicios publicados en la intranet de la UCI.

Q. No brindar sin autorización ningún servicio a la comunidad universitaria desde las estaciones de trabajo a las que tengo acceso, sabiendo que estos solo se podrán configurar en los servidores establecidos por la entidad encargada de su administración.

Las cuentas que les son asignadas a los usuarios de la red, serán auditadas con vistas a verificar el cumplimiento de los compromisos contraídos.

La Universidad se reserva la facultad de sancionar a los usuarios que incumplan con las normas para la utilización de las tecnologías de información y este Código de Ética.

Figura 34: Código de ética.