PEGASO: UNA PROPUESTA PARA LA GESTIÓN ACTIVOS DE SOFTWARE

ALEXANDER ALVARO BARÓN SALAZAR

UNIVERSIDAD EAFIT
FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA
MEDELLÍN
2012

PEGASO: UNA PROPUESTA PARA LA GESTIÓN ACTIVOS DE SOFTWARE

ALEXANDER ALVARO BARÓN SALAZAR

Trabajo para optar al título de
Maestro en Ingeniería informática

Asesora
RAQUEL ANAYA DE PÁEZ
Directora Grupo de Investigación en
Desarrollo de Software

UNIVERSIDAD EAFIT
FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA
MEDELLÍN
2012

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

_____
Firma de la Directora

_____
Firma del jurado

_____
Firma del jurado

Medellín, Marzo de 2012

Dedicatoria

Este trabajo es como la vida: un maravilloso viaje por alta mar lleno de aventuras, por eso lo dedico a mis tripulantes, quienes me ayudaron a llegar a feliz puerto.

A Dios, mi guía, El que controla los monstruos marinos, los feroces vientos y las turbulentas aguas. Quien alinea las estrellas que dirigen y dan luz a la travesía.

A mi Padre, mi consejero, quien escribió y me heredó las cartas de navegación que señalan las coordenadas correctas.

A Nancy, mi todo, quien se ocupa de los mapas y de la brújula para no perder el rumbo.

A Karen, mi linda, quien con sus exquisitas recetas de amor, alimenta el espíritu de los viajeros.

A Santiago, mi compañero de pasión, quien me ayuda a mantener firme y en dirección nuestro navío.

A Juan Camilo, mi risa de cada día, quien mantiene funcionando las maquinas que mueven nuestros anhelos.

AGRADECIMIENTOS

A la Doctora Raquel Anaya de Páez, Directora de este trabajo de investigación, por indicarme pacientemente la ruta hacia Dios y enseñarme el camino de la investigación.

Al Ingeniero Jairo Guerrero García, por el sabio consejo siempre oportuno.

Al ingeniero Jesús Insuasti, por su amistad incondicional y por guiarme por el laberinto de los repositorios digitales.

A mi Alma Mater, la Universidad de Nariño, por darme la oportunidad de estudiar, aprender, vivir y soñar.

A la empresa CJT&T y especialmente al Ingeniero Juan Carlos Torres por creer en nuestro proyecto y por permitirnos trabajar a su lado.

A mis compañeros Docentes del Departamento de Sistemas de la Universidad de Nariño, por su aprecio, apoyo y colaboración de todos los días.

A mis Estudiantes, por confiar en mi guía y transitar a mi lado por los senderos de la academia.

CONTENIDO

pág.

RESUMEN **14**

1. INTRODUCCION **15**

1.1 INTRODUCCIÓN 15

1.2 PLANTEAMIENTO DEL PROBLEMA 15

1.2.1 Problemas de gestión y organización 16

1.2.2 Problemas económicos 16

1.2.3 Problemas conceptuales y técnicos 17

1.3 PREGUNTAS DE INVESTIGACIÓN 18

1.4 OBJETIVOS 19

1.4.1 Objetivo general 19

1.4.2 Objetivos específicos 19

1.5 JUSTIFICACIÓN 19

1.6 METODOLOGÍA 21

1.7 ESTRUCTURA DEL DOCUMENTO 23

2. MARCO CONCEPTUAL **25**

2.1 INTRODUCCIÓN 25

2.2 REUTILIZACIÓN: UNA ESTRATEGIA PARA LA MEJORA DEL PROCESO SOFTWARE 25

2.2.1 Generalidades 25

2.2.2 Beneficios de la reutilización 26

2.2.2.1 Mejoramiento de la calidad 26

2.2.2.2 Reducción del esfuerzo 27

2.2.3 La reutilización según la norma IEEE1517 28

2.2.3.1 Referencias normativas y definiciones 29

2.2.3.2 Aplicación del estándar 30

2.2.3.3 Reutilización en ciclo de vida de software 31

2.3 ACTIVO DE CONOCIMIENTO: EL FUNDAMENTO PARA LA DEFINICIÓN DE ACTIVO DE SOFTWARE 33

2.3.1 Activos de conocimiento....34
2.3.2 Activos de conocimiento en las organizaciones de software....34
2.3.3 Un metamodelo para el activo de conocimiento en ingeniería de software 36
2.3.3.1 Tipos de activos de Conocimiento....38
2.3.3.2 Relación entre tipos de activos de Conocimiento....39
2.4 REPOSITORIOS Y CATALOGOS EN INGENIERIA DE SOFTWARE....40
2.4.1 Trabajos relacionados con repositorios y catálogos de activos de software....40
2.5 METADATOS: UN MECANISMO PARA LA DESCRIPCION DE ACTIVOS DE SOFTWARE....42
2.5.1 Dublin Core: un modelo de metadatos para la descripción de recursos digitales....43
2.5.1.1 Elementos Dublin Core relacionados con el contenido....44
2.5.1.2 Elementos Dublin Core relacionados con la propiedad intelectual....44
2.5.1.3 Elementos Dublin Core relacionados con la instanciación....45
2.6 PROPUESTAS DE DESARROLLO DE SOFTWARE Y SU ORIENTACIÓN HACIA LA REUTILIZACIÓN Y EL TRABAJO CON CATÁLOGOS....45
2.6.1 Rational Unified Process (RUP)....46
2.6.1.1 Objetivo de la propuesta RUP....46
2.6.1.2 Modelo de la propuesta RUP....46
2.6.1.3 Activos de la propuesta RUP....46
2.6.1.4 Proceso de la propuesta RUP....47
2.6.1.5 Orientación hacia la reutilización y el trabajo con catálogos en la propuesta RUP....47
2.6.2 Computer Based Software Development (CBSD)....48
2.6.2.1 Objetivo de la propuesta CBSD....48
2.6.2.2 Modelo de la propuesta CBSD....48
2.6.2.3 Activos de la propuesta CBSD....49
2.6.2.4 Proceso de la propuesta CBSD....49
2.6.2.5 Orientación hacia la reutilización y el trabajo con catálogos en la propuesta CBSD....50
2.6.3 Feature Oriented Domain Analysis (FODA)....51
2.6.3.1 Objetivo de la propuesta FODA....51

2.6.3.2 Modelo de la propuesta FODA 51
2.6.3.3 Activos de la propuesta FODA 51
2.6.3.4 Proceso de la propuesta FODA 52
2.6.3.5 Orientación hacia la reutilización y el trabajo con catálogos en la propuesta FODA 52
2.6.4 Agile Unified Process (AUP) 52
2.6.4.1 Objetivo de la propuesta AUP 52
2.6.4.2 Modelo de la propuesta AUP 52
2.6.4.3 Activos de la propuesta AUP 53
2.6.4.4 Proceso de la propuesta AUP 53
2.6.4.5 Orientación hacia la reutilización y el trabajo con catálogos en la propuesta AUP 54

3. CARACTERIZACION DE LOS ACTIVOS DE SOFTWARE 55
3.1 INTRODUCCIÓN 55
3.2 GENERALIDADES 55
3.3 ACTIVOS DE SOFTWARE EN LA PROPUESTA RUP 55
3.4 ACTIVOS DE SOFTWARE EN LA PROPUESTA CBSD 59
3.5 ACTIVOS DE SOFTWARE EN LA PROPUESTA FODA 61
3.6 ACTIVOS DE SOFTWARE EN LA PROPUESTA AUP 62
3.7 ANÁLISIS COMPARATIVO DE ACTIVOS EN PROPUESTAS DE DESARROLLO DE SOFTWARE 66
3.7.1 Identificación de los activos de software que formarán parte del repositorio 68
3.8 METAMODELO DE ACTIVOS DE SOFTWARE DESDE LA PERSPECTIVA DE LA INVESTIGACION 76

4. *PEGASO*: UNA PROPUESTA PARA LA GESTION DE ACTIVOS DE SOFTWARE 79
4.1 INTRODUCCIÓN 79
4.2 ARQUITECTURA DE UN SISTEMA DE GESTION DE ACTIVOS DE SOFTWARE 79
4.2.1 Capa de almacenamiento 81
4.2.2 Capa de lógica de sistema 81

4.2.3 Capa de presentación 83
4.3 DSPACE: LA SOLUCIÓN TECNOLÓGICA DE APOYO A LA GESTIÓN DE LOS ACTIVOS DE SOFTWARE EN *PEGASO* 84
4.3.1 Estudio de herramientas para la gestión de repositorios digitales 84
4.3.1.1 Criterios de análisis 84
4.3.1.2 Resumen del estudio de herramientas para la gestión de repositorios digitales 86
4.3.1.3 DSpace: la herramienta seleccionada 87
4.4 DC-SOFT: UN SISTEMA DE METADATOS PARA LA DESCRIPCIÓN DE ACTIVOS DE SOFTWARE DE *PEGASO* 89
4.4.1 Elementos DC-Soft relacionados con el contenido 89
4.4.2 Elementos DC-Soft relacionados con la propiedad intelectual 90
4.4.3 Elementos DC-Soft relacionados con la instanciación 91
4.4.4 Un ejemplo DC-Soft 92

5. LA GUÍA DE USO DE PORTAL ACTIVO: UN SISTEMA DE GESTION DE ACTIVOS DE SOFTWARE 94
5.1 INTRODUCCION 94
5.2 LOS PROCESOS EN PORTAL ACTIVO 94
5.2.1 Registro de usuarios 99
5.2.2 Administración de usuarios 100
5.2.3 Configuración de los metadatos del repositorio 101
5.2.4 Configuración del proceso software 103
5.2.5 Configuración de la estructura de almacenamiento del repositorio 103
5.2.6 Matricula de activos 108
5.2.7 Revisión y publicación de activos 111
5.2.8 Búsqueda de activos 112
5.2.9 Consultar información sobre un activo 113
5.2.10 Recuperación de activos 116

6. EXPERIENCIA DE IMPLEMENTACION EN UN CONTEXTO REAL 118
6.1 INTRODUCCION 118
6.2 PRESENTACION DE LA EMPRESA 118

6.3 DESCRIPCION DE LA EXPERIENCIA 118
6.3.1 Capacitación 119
6.3.2 Preparación de la herramienta Portal Activo 119
6.3.2.1 Configuración de la estructura de almacenamiento del repositorio 119
6.3.2.2 Definición y construcción de los activos iniciales 120
6.3.3 Aplicación de Portal Activo para el desarrollo de la disciplina requisitos ... 121
6.3.3.1 Búsqueda del activo de software 121
6.3.3.2 Recuperación del activo de software 121
6.3.3.3 Uso del activo de software 121
6.4 RESULTADOS DE LA EXPERIENCIA 123
6.4.1 Aplicación de encuestas 123
6.4.2 Análisis de información 124
6.4.2.1 Encuesta a usuarios finales 124
6.4.2.2 Encuesta a directivos 127
6.4.2.3 Apreciaciones generales 129

7. CONCLUSIONES Y TRABAJOS FUTUROS 130

BIBLIOGRAFÍA Y REFERENCIAS 132

LISTA DE TABLAS

pág.

Tabla 1. Fases y actividades de la Investigación 21
Tabla 2. Actividades y tareas reutilización en el Ciclo de Vida de Software 31
Tabla 3. Elementos del Estándar Dublin Core 43
Tabla 4. Elementos Dublin Core Relacionados con el Contenido 44
Tabla 5. Elementos Dublin Core Relacionados con la Propiedad Intelectual 44
Tabla 6. Elementos Dublin Core Relacionados con la Instanciación 45
Tabla 7. Activos de Software en la Propuesta RUP 56
Tabla 8. Activos de Software en la Propuesta CBSD (PARNAS, 1994) 60
Tabla 9. Activos de Software en la Propuesta FODA 61
Tabla 10. Activos de Software en la Propuesta AUP 62
Tabla 11. Relación de Activos de Software que cumplen objetivos similares 66
Tabla 12. Propuesta de activos de software que hacen parte del catálogo 69
Tabla 13. Siglas de tipos de diagramas y mecanismos de representación 75
Tabla 14. Resumen de estudio de Herramientas 86
Tabla 15. Conjunto de Metadatos DC-Soft 89
Tabla 16. Elementos DC-Soft Relacionados con el Contenido 89
Tabla 17. Elementos DC-Soft relacionados con la propiedad intelectual 90
Tabla 18. Elementos DC-Soft relacionados con la instanciación 91
Tabla 19. Un ejemplo DC-Soft: El modelo de diseño del sistema punto de venta 92
Tabla 20. Usuarios que interactúan con la Herramienta Portal Activo 94
Tabla 21. Procesos en Portal Activo 95
Tabla 22. Esfuerzo de la experiencia 119
Tabla 23. Activos iniciales que hacen parte del repositorio 120
Tabla 24. Opciones de respuesta y valores para ítems tipo escala de Likert 124
Tabla 25. Respuestas de usuarios finales 124
Tabla 26. Respuestas de directivos 127

LISTA DE FIGURAS

pág.

Figura 1. Estructura general del documento de la norma IEEE1517 29
Figura 2. Categorías de reutilización soportadas por el estándar 30
Figura 3. Metamodelo de activo de conocimiento 37
Figura 4. Proceso de la propuesta RUP 47
Figura 5. Ciclo de vida de CBSD integrado con ingeniería de dominio 49
Figura 6. CBSD con reutilización 50
Figura 7. Proceso de la propuesta AUP 53
Figura 8. Metamodelo de Activo de Software extensión de (ANAYA, y otros, 2008) 78
Figura 9. Arquitectura de un sistema de gestión de activos de software 80
Figura 10. Capa de almacenamiento del sistema de gestión de activos 81
Figura 11. Capa de lógica de sistema de gestión de activos 82
Figura 12. Capa de presentación del sistema de gestión de activos 83
Figura 13. Uso de repositorios de libre acceso a nivel mundial 87
Figura 14. Uso de repositorios de libre acceso en Sur América 88
Figura 15. Diagrama de flujo de los Procesos asociados a todos los usuarios 96
Figura 16. Diagrama de flujo de los Procesos asociados al usuario administrador 97
Figura 17. Diagrama de flujo de los Procesos asociados al usuario editor 98
Figura 18. Diagrama de flujo de los Procesos asociados al usuario supervisor 99
Figura 19. Registro de usuario en DSpace UDENAR 100
Figura 20. Administración de usuarios en DSpace UDENAR 101
Figura 21. Configuración de metadatos en interfaz de usuario 102
Figura 22. Configuración de metadatos en base de datos 102
Figura 23. Comandos de reconstrucción y despliegue de DSpace UDENAR 103
Figura 24. Creación de comunidades en DSpace UDENAR 104
Figura 25. Página de inicio de la comunidad en DSpace UDENAR 105
Figura 26. Descripción de colecciones en DSpace UDENAR (1) 106
Figura 27. Descripción de colecciones en DSpace UDENAR (2) 107

Figura 28. Estructura de comunidades y colecciones en DSpace UDENAR 108
Figura 29. Matricula de activos en DSpace paso iniciar 108
Figura 30. Matricula de activos en DSpace paso describir 109
Figura 31. Matricula de activos en DSpace paso subir 109
Figura 32. Matricula de activos en DSpace paso verificar 110
Figura 33. Matricula de activos en DSpace paso licencia 110
Figura 34. Matricula de activos en DSpace paso completo 111
Figura 35. Revisión y publicación de activos en DSpace UDENAR 111
Figura 36. Búsqueda de activos en DSpace UDENAR modo básico 112
Figura 37. Búsqueda de activos en DSpace UDENAR modo avanzado 113
Figura 38. Resultados de búsqueda en DSpace UDENAR 114
Figura 39. Metadatos del activo seleccionado en DSpace UDENAR (1) 115
Figura 40. Metadatos del activo seleccionado en DSpace UDENAR (2) 116
Figura 41. Descarga de archivos en DSpace UDENAR 117
Figura 42. Metadatos del activo de proyecto *Requisitos Plan Fidelización* (1) 122
Figura 43. Metadatos del activo de proyecto *Requisitos Plan Fidelización* (2) 123
Figura 44. Apreciaciones consolidadas por sujeto según encuesta a usuarios ...125
Figura 45. Apreciaciones por ítem según encuesta a usuarios 126
Figura 46. Apreciaciones por sujeto según encuesta a directivos 128
Figura 47. Apreciaciones por ítem según encuesta a directivos 128
Figura 48. Apreciaciones por sujeto consolidado (usuarios - directivos) 129

RESUMEN

Diversos autores y organizaciones han encaminado sus esfuerzos para identificar los problemas que dificultan la reutilización y proponer estrategias que permitan incorporarla de manera sistemática en el desarrollo de software (SOMMERVILLE, 2007)(JHA, y otros, 2006)(FRAKES, y otros, 2005). Estos trabajos han sido analizados para clasificar los problemas identificados en tres principales categorías: problemas de gestión y organización, problemas económicos y problemas conceptuales y técnicos.

Los problemas de gestión y organización se concentran en la falta de procedimientos explícitos y de estándares que regulen las tareas de reutilización en la empresa. Los problemas económicos se refieren a la alta inversión con retorno a mediano y largo plazo. Los problemas conceptuales y técnicos se resumen en: la falta de criterios unificados para definir activo de software reutilizable; la dificultad para crear y gestionar una biblioteca de activos de software, la falta de estándares para clasificación, catalogación, descripción, búsqueda, recuperación y uso (adaptación) de los activos.

Como respuesta a cada uno de los elementos de la problemática identificada, este trabajo de investigación define *PEGASO:* una propuesta para la gestión activos de software. La propuesta es una solución integral para la aplicación de la reutilización e incluye: un metamodelo que define el activo de software, un conjunto de metadatos como mecanismo de descripción de los activos de software, la adopción de una herramienta de gestión de repositorios digitales como solución tecnológica de soporte, una guía de adopción, como mecanismo para asistir los procesos de gestión y la validación de la propuesta por medio de su implementación en un contexto real.

PEGASO permite a las organizaciones de software, construir repositorios lo suficientemente flexibles y prácticos como para incorporar en las tareas del desarrollador la reutilización como práctica sistemática, incrementando los niveles de calidad de los productos software y la productividad del equipo de desarrollo. El repositorio busca que desarrolladores definan los artefactos software potencialmente reutilizables para ser catalogados de manera homogénea y desde donde pueden ser consultados, de manera que facilite su reutilización.

Palabras Claves: software, activo, reutilización, repositorio, catálogo

1. INTRODUCCION

1.1 INTRODUCCIÓN

Este capítulo tiene como objetivo presentar una visión global de los elementos que contextualizan la propuesta. El capítulo inicia con el planteamiento del problema, la sistematización de la investigación, los objetivos establecidos para la investigación, la justificación y la metodología utilizada para el desarrollo de la investigación. Finalmente, describe la organización del documento para ofrecer al lector una visión general y de alto nivel del contenido.

1.2 PLANTEAMIENTO DEL PROBLEMA

Desde sus inicios, la construcción de software ha tenido importantes avances en materia de lenguajes de programación, compiladores, herramientas CASE y entornos integrados de desarrollo, entre otros. Una de las necesidades que se hace evidente con la construcción del software y que cada día cobra más interés para la academia y la industria es la reutilización.

La Reutilización es el uso de un activo para la solución de diferentes problemas. Un activo reutilizable es un elemento tal como diseños, especificaciones, código fuente, documentación, unidades de prueba y manuales de procedimientos, los cuales fueron diseñados para su uso en múltiples contextos (IEEE COMPUTER SOCIETY, 2010). La reutilización tiene dos connotaciones: 1. El grado en el cual un activo puede ser usado en más de un sistema software o pueda ser incorporado dentro de otro activo. 2. En una biblioteca de reutilización, las características de un activo que lo hacen fácil de usar en diferentes contextos, sistemas software o incorporados en otros activos (IEEE COMPUTER SOCIETY, 2010).

En la actualidad existe una buena cantidad de trabajos de investigación que abordan la reutilización como objeto de estudio (JHA, y otros, 2006) (FRAKES, y otros, 2005) (IEEE COMPUTER SOCIETY, 2010) (JASMINE, 2010) (HENNINGER, 1999) (ERDOGMUS, 2002). Igualmente, existe una buena cantidad de catálogos, los cuales se especializan en la gestión de un tipo específico de activo de software orientado a una fase concreta de un proceso de desarrollo o a una tecnología en particular (WINDOWSCLIENT.NET COMUNITY, 2011) (DATA & OBJECT FACTORY, 2001) (INFRAGISTICS, INC. QUINCE, 2009) (WELIE, Martin Van, 2008). Estos catálogos son utilizados por grandes organizaciones, pero no existen soluciones escalables a pequeñas empresas de software.

La ingeniería del software ha generado importantes aportes que contribuyen a facilitar la implementación de prácticas de reutilización en las organizaciones, el uso de activos de software integrados en catálogos o repositorios es un ejemplo.

Sin embargo, a pesar de estos importantes aportes y de los evidentes beneficios que genera, la reutilización no es tan ampliamente practicada como podría suponerse. La forma más común de reutilización es la realizada de manera intuitiva por los desarrolladores a través del cortar y pegar de código que generalmente ellos mismos han desarrollado.

Hay muchos factores que directa o indirectamente dificultan la incorporación de la reutilización en los procesos de desarrollo de las empresas de software. Estos factores pueden ser de tipo conceptual, técnico, administrativo, organizacional, de naturaleza psicológica, económica o jurídica.

Autores como Sommerville (SOMMERVILLE, 2007), Jha (JHA, y otros, 2006), y Frakes y Kang (FRAKES, y otros, 2005) han identificado los problemas que dificultan la incorporación de la reutilización para proponer estrategias de solución. Estos trabajos han sido analizados y clasificados en tres principales categorías (problemas de gestión y organización, problemas económicos y problemas conceptuales y técnicos), los cuales se sintetizan a continuación.

1.2.1 Problemas de gestión y organización

La reutilización no es sólo un problema técnico que debe ser resuelto por los ingenieros de software. El apoyo de la alta gerencia de la empresa y una estructura organizativa adecuada son igualmente importantes. Los problemas referidos a la gestión y organización se concentran en la falta de procedimientos explícitos y a la falta estándares propios o adoptados que regulen las tareas de reutilización en la empresa.

1.2.2 Problemas económicos

Estos problemas dificultan la implementación de la reutilización en las empresas ya que a partir de un equivocado análisis se llega a la conclusión de que es inviable económicamente. Los problemas económicos se refieren principalmente a la alta inversión con retorno a mediano y largo plazo. La reutilización requiere inversiones iniciales en infraestructura, metodología, entrenamiento y herramientas, además de inversiones en la operación del proceso. El retorno de la inversión se da a mediano y largo plazo, tiempo que en ocasiones, la alta dirección no está dispuesta a conceder. Por otro lado, los catálogos de activos de software existentes son utilizados por grandes empresas, pero no existen soluciones escalables a pequeñas empresas de software que tienen menores posibilidades económicas.

1.2.3 Problemas conceptuales y técnicos

Los problemas conceptuales y técnicos más comunes que dificultan la incorporación de la reutilización en las empresas son:

- Falta de criterios unificados sobre la definición de un componente o activo reutilizable.

- Dificultad para crear y gestionar una biblioteca de activos de software. No existen estándares para clasificación, catalogación, búsqueda, recuperación y uso (adaptación) de los activos.

- Especialización de los catálogos en un tipo de activo de software. Los catálogos existentes, se especializan en un tipo específico de activo orientado a una fase concreta de un proceso o a una tecnología en particular. En consecuencia, la infraestructura de gestión se define según las características del tipo de activo, limitando el alcance de aplicación del catálogo.

- Mecanismo de descripción específica para cada tipo de activo. En cada catálogo, la descripción se realiza utilizando un mecanismo exclusivo de acuerdo con las características del tipo de activo que gestiona. Incluso, catálogos que gestionan el mismo tipo de activo, difieren en el mecanismo de descripción.

- Mecanismo de representación específico para cada tipo de activo. Cada catálogo, define el mecanismo de representación (diagramas, plantillas, documentos, etc.), el formato y la herramienta para recuperar el activo. Este hecho limita la expresividad del activo y somete al usuario a la vista que el publicador impone.

- Dificultad para gestionar la diversidad de activos de software. La gran cantidad de tipos de activos de software existentes y la heterogeneidad en su descripción y representación dificultan integrar varios tipos de activos en un mismo catálogo y la compartición de activos entre catálogos. Antes de adicionar un activo es necesario formatear el activo a las condiciones de descripción y representación que impone el catálogo, en este proceso se pone en riesgo la esencia del conocimiento que el activo pretende trasmitir.

- Falta de herramientas que soporten un proceso de desarrollo basado en reutilización que integra un catálogo de activos de software.

La problemática descrita hace que la mayoría de los activos de software no sean conocidos por el equipo de desarrollo, que el desarrollador no pueda entender realmente a los activos de software cuando los utiliza, que no pueda encontrarlos, que al encontrarlos no sepa cómo utilizarlos o que no pueda compartir su experiencia en forma adecuada.

Compartir activos de software es compartir conocimiento. La falta de una propuesta integral para la gestión de activos de software dificulta la transmisión del conocimiento que el activo de software intenta expresar.

La persistencia de estos problemas evidencia la necesidad de implementar mecanismos que les permitan a los desarrolladores aplicar de manera natural la reutilización de activos de software como práctica sistemática, para incrementar su productividad y la calidad de los productos software.

Como una contribución a la solución de esta problemática, este trabajo de investigación define una propuesta para la gestión de activos de software. Esta propuesta integra: un metamodelo que define el activo de software, un conjunto de metadatos como mecanismo de descripción de los activos de software, la adopción de una herramienta de gestión de repositorios digitales como solución tecnológica de soporte, una guía de implementación, como mecanismo para asistir los procesos de gestión y la validación de la propuesta por medio de su implementación en un contexto real.

1.3 PREGUNTAS DE INVESTIGACIÓN

Surgen entonces los siguientes interrogantes:

- ¿De qué manera se pueden caracterizar los activos de software para facilitar su reutilización?
- ¿Cuáles son las buenas prácticas asociadas a la reutilización sistemática?
- ¿Cuál es la solución tecnológica apropiada que apoye la gestión de los activos de software?
- ¿De qué manera se facilita a las organizaciones la adopción de la propuesta de gestión de activos de software?
- ¿Qué beneficios trae a una organización la adopción de la propuesta de gestión de activos de software?

1.4 OBJETIVOS

1.4.1 Objetivo general

Definir una propuesta para la gestión de los activos de software de una organización de tal manera que se promueva la reutilización sistemática

1.4.2 Objetivos específicos

- Identificar las características generales de los activos de software.
- Identificar las buenas prácticas asociadas a la reutilización sistemática.
- Proponer una solución tecnológica que apoye la gestión de los activos de software.
- Definir una guía para la adopción de la propuesta en una organización.
- Validar la propuesta por medio de su implementación en una organización.

1.5 JUSTIFICACIÓN

Se entiende por reutilización sistemática, la incorporación de las estrategias de reutilización de manera explícita y estratégica a lo largo del ciclo de vida de una solución software y va mas allá de la reutilización intuitiva de código que realizan los desarrolladores por iniciativa personal. Algunos de los aspectos que impiden la reutilización sistemática son la falta de procedimientos explícitos de reutilización, la falta de la adopción de estándares que regulen las tareas de reutilización en la empresa, la dificultad para la creación y gestión de nuevos activos de software y la especialización de los existentes (SOMMERVILLE, 2007) (JHA, y otros, 2006) (FRAKES, y otros, 2005)

Esta problemática hace que las organizaciones perciban la reutilización sistemática como una estrategia más costosa y más compleja de lo que realmente es y renuncien a su implementación. Este hecho impide que se aprovechen los evidentes beneficios que la reutilización genera: el incremento de la productividad del equipo de desarrollo y de la calidad del producto software, además de la disminución del esfuerzo y tiempo de desarrollo.

Igualmente, en los casos en los que las organizaciones intentan la implementación de la reutilización, la problemática descrita hace que la mayoría de los activos de software no sean conocidos por el equipo de desarrollo, que el desarrollador no pueda entender realmente a los activos de software cuando los utiliza, que no pueda encontrarlos, que al encontrarlos no sepa cómo utilizarlos o que no pueda compartir su experiencia en forma adecuada.

Esta problemática, genera la necesidad de implementar una estrategia integral que les permitan a los desarrolladores, aplicar de manera natural la reutilización de activos de software como práctica sistemática de manera sencilla y a bajo costo.

Se requiere por lo tanto, que esta estrategia sea una propuesta que integre: un mecanismo estandarizado de definición y descripción de los activos de software, una solución tecnológica que soporte la propuesta integralmente y una guía para la adopción de la propuesta para la gestión y uso de los activos de software.

La propuesta debe integrar el uso de repositorios que gestionen cualquier tipo de activos de software, independiente de la fase y del proceso de desarrollo, así, el repositorio provee la flexibilidad requerida para adaptarse al proceso de la organización y puede soportar todo el proceso de desarrollo. Los catálogos deben permitir compartir entre si activos de software independiente de las características propias del activo, incluida el mecanismo de representación. Igualmente y para garantizar la eficiencia de la propuesta, esta debe ser validada en un contexto real.

Este trabajo de investigación, define *PEGASO:* una propuesta para la gestión activos de software. La propuesta proporciona el marco conceptual y la infraestructura requeridos para la implementación de la reutilización como práctica sistemática en las organizaciones por medio del uso de un repositorio de activos de software.

La aplicación de activos de software reutilizables en las diferentes etapas del ciclo de vida, permite incrementar la calidad del software, disminuir la complejidad del proceso y reducir el costo y el tiempo de trabajo. Los activos de software reutilizables, permiten a los desarrolladores de software agilizar el proceso, incorporando resultados positivos de experiencia anteriores, su uso es muy importante porque permite integrar el conocimiento y la experiencia de muchos desarrolladores. De esta manera se capitalizan las ventajas de la reutilización, agilizando el proceso e incrementando los niveles de calidad del producto.

PEGASO es una propuesta integral para la aplicación de la reutilización que incluye: la extensión de un metamodelo de activo de conocimiento (ANAYA, y otros, 2008) como metamodelo que define el activo de software, un conjunto de metadatos llamado DC-Soft como mecanismo de descripción de los activos de software, la adopción DSPACE (FEDORA, 2011), una herramienta de gestión de repositorios digitales como solución tecnológica de soporte, una guía de adopción, como mecanismo para asistir los procesos de gestión y la validación de la propuesta por medio de su implementación en una empresa de software.

PEGASO permite a las organizaciones de software, construir repositorios lo suficientemente flexibles y prácticos como para incorporar en las tareas del desarrollador la reutilización como práctica sistemática, incrementando los niveles

de calidad de los productos software y la productividad del equipo de desarrollo. El repositorio busca que desarrolladores definan los artefactos software potencialmente reutilizables para ser catalogados de manera homogénea y desde donde pueden ser consultados, de manera que facilite su reutilización.

1.6 METODOLOGÍA

El trabajo de investigación es de tipo descriptivo bajo un enfoque cualitativo, se aplicó estudio transversal y se utilizó el método inductivo.

La investigación es de tipo descriptivo porque tuvo entre sus propósitos reconocer las características generales de los activos de software y las buenas prácticas asociadas a la reutilización como actividad previa a la formulación de una propuesta para la gestión de activos de software.

El enfoque empleado es el cualitativo porque la investigación incluyó entre sus tareas caracterizar, definir y categorizar los activos de software a fin de establecer los procesos de gestión.

Se aplicó un diseño no experimental porque durante la investigación no se alteraron ni manipularon las características de los activos de software. La investigación se concentró en caracterizar los activos de software y categorizarlos para una gestión efectiva.

El estudio es de tipo transversal porque la identificación de las características y la categorización de los activos de software se hicieron en un solo momento de la investigación, con posibilidad de retroalimentación a partir de hallazgos en posteriores etapas del trabajo.

El método es de tipo inductivo porque el trabajo de investigación incluyó el análisis de propuestas de desarrollo para caracterizar los activos de software, el análisis de un estándar de descripción de recursos digitales para definir el mecanismo de descripción de los activos de software y el análisis de las buenas prácticas asociadas a la reutilización para proponer los procesos de gestión de activos de software.

Para el desarrollo del trabajo de investigación se contemplan las fases y actividades que se relacionan en la tablas 1.

Tabla 1. Fases y actividades de la Investigación

Fase / Actividad
1. Caracterización de los Activos de Software
1.1. Leer y analizar documentos referentes
1.2. Identificar las propuestas de desarrollo para analizar

Tabla 1. (Continuación)

Fase / Actividad
1.3. Analizar las propuestas de desarrollo y los activos de software de los cuales se ocupan
1.4. Identificar los activos de software que hacen parte del repositorio
1.5. Identificar y definir las características generales de los activos de software
1.6. Definir activo de software en el marco del trabajo de investigación
2. Definición del repositorio digital que implementa *PEGASO*
2.1. Leer y analizar documentos referentes
2.2. Definir la arquitectura de un sistema de gestión de activos de software
2.3. Identificar las herramientas de gestión de repositorios digitales para estudiar
2.4. Realizar un estudio comparativo de las herramientas identificadas
2.5. Seleccionar la herramienta de gestión de repositorios digitales que soporta la propuesta
2.6. Estudiar el estándar de metadatos Dublin Core
2.7. Definir el conjunto de metadatos orientado a la descripción de activos de software
3. Guía de uso de la reutilización.
3.1. Leer y analizar documentos referentes
3.2. Estudiar el IEEE Standard for Information Technology – System and Software Life Cycle Processes – Reuse Processes, IEEE Standard 1517™ - 2010
3.3. Identificar las buenas prácticas propuestas en el estándar estudiado
3.4. Definir la guía de adopción de la propuesta, incorporando las buenas prácticas identificadas en el estándar estudiado.
3.5. Construir Portal ACTIVO. Portal Activo incorpora el marco conceptual definido para la investigación, implementa el conjunto de metadatos DC-Soft, sigue la guía para la adopción de *PEGASO* y utiliza la solución tecnológica seleccionada.
4. Validación de *la propuesta*
4.1. Lectura de documentos referentes.
4.2. Seleccionar la empresa que provea el contexto real para validar *PEGASO*
4.3. Capacitar al personal de la empresa seleccionada en el marco conceptual y práctico de *PEGASO* y de Portal *Activo.*
4.4. Preparar la herramienta Portal Activo para el desarrollo de la experiencia
4.5. Desarrollar actividades del proceso software de un proyecto en la empresa seleccionada utilizando Portal Activo.
4.6. Diseñar y aplicar encuestas para recolectar la información resultado de la experiencia
4.7. Analizar los resultados obtenidos del proceso de validación de *PEGASO* en la empresa
4.8. Incorporar en *PEGASO,* los aspectos que se consideren pertinentes resultantes del análisis de los resultados obtenidos del proceso de validación.

Tabla 1. (Continuación)

Fase / Actividad
5. Documentación Final
5.1. Elaboración del documento final y de la presentación de *PEGASO*: Una propuesta para la gestión de activos de software.

1.7 ESTRUCTURA DEL DOCUMENTO

Este documento presenta de manera estructurada el proceso de investigación que se siguió para el logro de los objetivos propuestos.

En el capítulo 2, se presenta el marco conceptual que orienta la investigación: se estudia la reutilización en el marco de la ingeniería de software como una estrategia para la mejora del proceso software. Se estudia también en este capítulo, el concepto de activo de conocimiento como fundamento para caracterizar el activo de software. Mas adelante, en este capítulo, se abordan los catálogos de activos de software como repositorios digitales en el contexto de la ingeniería de software. Igualmente, se analiza el concepto de metadatos como mecanismo para describir recursos digitales. Finalmente se estudian propuestas de desarrollo de software y su orientación hacia la reutilización y el trabajo con catálogos.

En el capítulo 3, se definen y caracterizan los activos de software. Inicialmente, se identifican y describen los activos generados en cada una de las fases de propuestas de desarrollo. Posteriormente se realiza el análisis comparativo para determinar los activos de software que hacen parte del repositorio. Finalmente, se presenta el metamodelo que define y caracteriza el activo de software como objeto central de reutilización.

En el capítulo 4, se presenta *PEGASO:* una propuesta para la gestión de activos de software. Inicialmente, se expone la arquitectura de *PEGASO* y se describen cada uno de sus elementos, sus relaciones y el rol que desempeñan en la gestión de los activos de software. Más adelante, se presenta DSpace, la solución tecnológica de apoyo a la gestión de los activos de software y el estudio de herramientas para la gestión de repositorios digitales que permite identificar a DSpace como la herramienta adecuada para soportar *PEGASO*. Posteriormente, se expone el sistema de metadatos DC-Soft empleado para la descripción de activos de software.

En el capítulo 5, se presenta la guía de uso de Portal Activo: Un sistema de gestión de activos de Software basado en *PEGASO*. La guía define los procesos de gestión de activos de software e implementa el estándar 1517 de la IEEE Computer Society.

En el capítulo 6 se presenta la experiencia de implementación de *PEGASO* en una empresa de software como experiencia en un contexto real. Se presenta la empresa donde se realizó el ejercicio, se describe el proceso de la experiencia y finalmente, se presentan los resultados de la experiencia obtenidos por medio de encuestas aplicadas a los participantes.

En el capítulo 7 se muestran las conclusiones y trabajos futuros.

Finalmente, se relacionan las referencias bibliográficas empleadas para el desarrollo de la investigación.

2. MARCO CONCEPTUAL

2.1 INTRODUCCIÓN

En este capítulo se presenta el marco conceptual que orienta la investigación.

Inicialmente, se realiza un estudio sobre reutilización como estrategia para la mejora del proceso software. Se presentan los aspectos generales de la reutilización, los beneficios de la reutilización y la reutilización en el marco internacional de los estándares. Este estudio permite identificar las buenas prácticas asociadas a reutilización sistemática como base para la definición de una guía para la adopción de la propuesta en una organización.

Más adelante, se realiza un breve estudio de la gestión del conocimiento. Se presentan los aspectos generales, la gestión del conocimiento en proyectos de software y la definición de activos de conocimiento, a efectos de establecer las bases para la definición y tratamiento del activo de software como un activo de conocimiento y su aplicación en la ingeniería del software.

En seguida, se analiza el nivel meta en el ámbito de los datos como mecanismo para describir recursos digitales. Se presenta el estándar Dublin Core, los elementos que integran el modelo y su clasificación. Este análisis permite conocer el estándar y las posibilidades de extensión para describir activos de software.

Finalmente, se presentan experiencias de trabajos sobre catálogos y repositorios digitales en ingeniería del software.

2.2 REUTILIZACIÓN: UNA ESTRATEGIA PARA LA MEJORA DEL PROCESO SOFTWARE

2.2.1 Generalidades

Actualmente, existe una buena cantidad de trabajos de investigación sobre reutilización de software que proponen variadas definiciones al respecto, por ejemplo: Charles W. Krueger en (KRUEGER, 1992), define la reutilización de software como la disciplina encargada de la creación de sistemas de software a partir de software pre-existente. Para Ian Sommerville, la reutilización es el proceso de creación de sistemas de software a partir de partes de software existente en lugar de construir desde cero (SOMMERVILLE, 2007).

La IEEE define la Reutilización como el uso de un activo para la solución de diferentes problemas. La reutilización tiene dos connotaciones: 1. El grado en el cual un activo puede ser usado en más de un sistema software o pueda ser incorporado dentro de otro activo. 2. En una biblioteca, las características de un activo que lo hacen fácil de usar en diferentes contextos, sistemas software, o incorporados en otros activos (IEEE COMPUTER SOCIETY, 2010).

Desde la perspectiva de procesos, la reutilización de software integra dos subprocesos: desarrollo para reutilización y desarrollo con reutilización. El desarrollo para reutilización trata del desarrollo de los activos reutilizables y la definición de las actividades involucradas. Los activos producidos en el desarrollo para reutilización, son catalogados y almacenados en una biblioteca. El desarrollo con reutilización, trata de cómo utilizar los activos en la construcción de aplicaciones, define los mecanismos para la búsqueda y la recuperación por parte de los miembros del equipo (JARZABEK, 2007).

Existen propuestas que se especializan en uno de estos dos subprocesos (WINDOWSCLIENT.NET COMUNITY, 2011) (DATA & OBJECT FACTORY, 2001) (INFRAGISTICS, INC. QUINCE, 2009) (WELIE, Martin Van, 2008). Este trabajo de investigación define una propuesta que puede ser transversal a estos subprocesos pues en primer lugar se orienta a la catalogación de los activos y, en segundo lugar a la recuperación de los mismos.

2.2.2 Beneficios de la reutilización

Los trabajos de investigación que abordan la reutilización sistemática de software (SOMMERVILLE, 2007) (JHA, y otros, 2006) (FRAKES, y otros, 2005), reconocen como los beneficios más relevantes para las organizaciones que incorporan la reutilización, el mejoramiento de la calidad y la reducción del esfuerzo, estos aspectos redundan en el incremento de la productividad del equipo de desarrollo.

2.2.2.1 Mejoramiento de la calidad

La reutilización de software impacta de manera positiva en la calidad del producto, en la productividad, en el rendimiento, en la fiabilidad y en la interoperabilidad.

- Calidad: Los componentes reutilizables de software son probados y validados antes de publicarse en la biblioteca, de tal manera que se puede garantizar su calidad en el proceso de desarrollo de software donde son reutilizados. Sin embargo, esto requiere de la administración y mantenimiento constantes de la biblioteca.

- Productividad: La productividad del equipo de desarrollo se incrementa debido a la menor cantidad de código que debe ser desarrollado. Esto se traduce en menos esfuerzo para realizar pruebas y en ahorro de tiempo de análisis y diseño produciendo un ahorro global del costo del proyecto. Cuando una organización implementa un sistema de gestión de reutilización, al inicio, el incremento de la productividad puede no ser evidente, incluso puede disminuir, debido al esfuerzo que debe invertir el equipo de trabajo en la adaptación y aprendizaje del nuevo proceso de desarrollo. Esta disminución temporal de la productividad debe ser fácilmente compensada por un aumento a largo plazo.

- Rendimiento: En una organización que aplica reutilización extensiva, el esfuerzo invertido en optimizaciones de los componentes reutilizables se capitaliza en el incremento del rendimiento del componente. El rendimiento ganado a partir del esfuerzo invertido en optimizaciones es mayor cuando se aplica en un componente reutilizable que cuando se aplica en un componente que se desarrolla y utiliza una sola vez.

- Confiabilidad: Utilizando componentes reutilizables bien probados aumenta la confiabilidad de un sistema de software. Además, el uso de un componente en varios sistemas hace más riguroso el proceso de pruebas, lo que incrementa la confiabilidad del componente.

- Interoperabilidad: Para que varios sistemas interactúen de manera eficiente, sus interfaces deben implementarse consistentemente. Las interfaces pueden ser componentes reutilizables o sistemas de componentes reutilizables empleados en varios sistemas.

- Estándares de cumplimiento: Algunas normas, como las normas de interfaz de usuario, pueden ser implementadas como un conjunto de componentes estándar reutilizables. Por ejemplo, si los menús en interfaces de usuario se implementan utilizando componentes reutilizables, se garantiza que todas las aplicaciones presentan uniformidad en este aspecto. El uso de interfaces de usuario estándar mejora la confiabilidad de los usuarios ya que interactúan con un familiar y homogéneo aunque se trate de nuevas aplicaciones.

2.2.2.2 Reducción del esfuerzo

La reutilización elimina el trabajo redundante y por lo tanto el tiempo total de construcción del producto software, así como los costos y el tamaño del equipo de trabajo. Esto es especialmente beneficioso para las empresas de software teniendo en cuenta la importancia competitiva de la entrega oportuna.

- El trabajo redundante, el tiempo de desarrollo: El desarrollo de todos los sistemas desde cero implica la construcción redundante de muchas partes del software. Esto se puede evitar cuando estas partes están disponibles como componentes reutilizables y se pueden compartir, lo que resulta en la disminución del tiempo de desarrollo y de los costos asociados.

- Tiempo para estar disponible en el mercado: El éxito o el fracaso de un producto de software, es a menudo determinado por su tiempo de comercialización, es decir el tiempo en estar disponible para el usuario final. El uso de componentes reutilizables permite reducir este tiempo.

- Documentación: A pesar de que la documentación es muy importante para el mantenimiento de un software, es un aspecto que a menudo se descuida. La reutilización de componentes de software reduce la cantidad de documentación para escribir. Sólo la estructura general del software y los nuevos componentes desarrollados deben ser documentados. La documentación de los componentes reutilizables puede ser compartida por varios software.

- Costos de mantenimiento: Se puede esperar menos defectos al emplear componentes reutilizables que ya han sido probados, por lo tanto el software debe ser sometido a menos tareas de mantenimiento. Los componentes reutilizables son mantenidos por un grupo independiente y no por separado en cada software.

- Costos de entrenamiento: Con la práctica y en un tiempo prudente, los ingenieros del software adquieren un buen conocimiento de los componentes reutilizables, en esa medida, los beneficios de la reutilización son capitalizados por la organización.

- Tamaño del equipo de desarrollo: Los grandes equipos de desarrollo sufren de sobrecarga de comunicación. Duplicar el tamaño de un equipo de desarrollo no significa que la productividad se duplique. Si muchos componentes pueden ser reutilizados, entonces los sistemas de software se pueden desarrollar con equipos más pequeños, dando lugar a una mejor comunicación y a una mayor productividad.

- Mejor aprovechamiento del talento humano: En un proceso de desarrollo basado en reutilización los desarrolladores se dedican a la construcción de activos reutilizables, que encapsulan el conocimiento que puede ser aprovechado por otros desarrolladores en diferentes proyectos.

- Reduce el riesgo del proceso: En un desarrollo con reutilización, hay menos incertidumbre en los costos que en un proceso de desarrollo desde ceros. Este es un factor importante para la gestión de proyectos, ya que reduce el margen de error en la estimación de los costos del proyecto. Este beneficio es directamente proporcional al tamaño del elemento reutilizado.

2.2.3 La reutilización según la norma IEEE1517

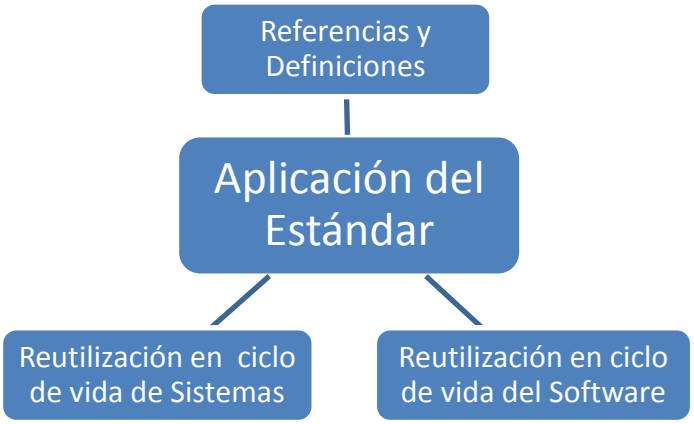
Uno de los puntos fuertes que se destaca en IEEE Computer Society es el diseño e implementación de estándares de industria; en este sentido, la sociedad ha establecido su biblioteca digital (The IEEE Computer Society Digital Library - CSDL) la cual contiene más de 330,000 artículos, 3,500 actas de congresos, y 26 publicaciones de libros seriadas en cooperación con John Wiley and Sons. Para el desarrollo de los estándares, la sociedad involucra a los más prestigiosos

profesionales en las ciencias de la computación y afines, los cuales desarrollan una metodología de revisión internacional la cual eleva los niveles de calidad de la producción escrita.

Uno de los estándares desarrollados por IEEE Computer Society, el cual fue codificado como 1517, presenta los aspectos relevantes a tenerse en cuenta para el desarrollo de los ciclos de vida del software desde un enfoque orientado a la reutilización (IEEE COMPUTER SOCIETY, 2010).

La organización del estándar corresponde con la estructura presentada en la figura 1:

Figura 1. Estructura general del documento de la norma IEEE1517



2.2.3.1 Referencias normativas y definiciones

En este apartado, la norma establece los puntos de partida para la implementación del estándar a través de la declaración de definiciones particulares. La norma define 27 términos a manera de glosario, de los cuales tres son muy pertinentes y son adoptados por la presente investigación. Tales términos son (IEEE COMPUTER SOCIETY, 2010):

- Reutilización: El estándar la define como: 1. El grado en el cual un activo puede ser usado en más de un sistema software o pueda ser incorporado dentro de otro activo. 2. En una biblioteca de reutilización, las características de un activo que lo hacen fácil de usar en diferentes contextos, sistemas software, o incorporados en otros activos.

- Reutilización: El uso de un activo para la solución de diferentes problemas.

- Activo Reutilizable: Un elemento tal como diseños, especificaciones, código fuente, documentación, unidades de prueba, manuales de procedimientos, etc., los cuales fueron diseñados para su uso en múltiples contextos.

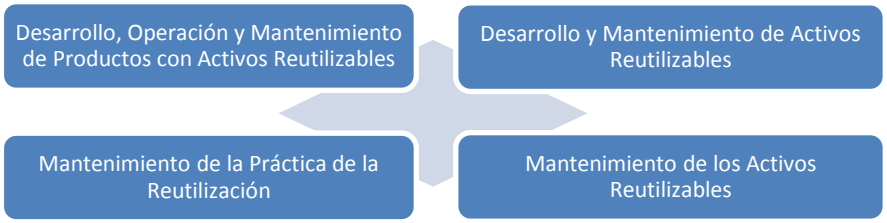
- Reutilización Sistemática: La práctica de la reutilización de acuerdo con procesos repetibles y bien definidos.

2.2.3.2 Aplicación del estándar

Esta cláusula presenta el marco de trabajo de las actividades de reutilización dentro de los ciclos de vida que pueden orientarse hacia el desarrollo, la operación, y el mantenimiento de sistemas y productos software a través de la reutilización de activos de software. En adición, este estándar orienta sus actividades alternativamente hacia el desarrollo y el mantenimiento de activos de software reutilizables.

En términos generales, el estándar maneja cuatro categorías tal como lo muestra la figura 2:

Figura 2. Categorías de reutilización soportadas por el estándar



Cada proceso de ciclo de vida dentro del estándar está dividido en una serie de actividades. Teniendo en cuenta las categorías anteriores, cada una de ellas define los elementos que se deben tener en cuenta para el desarrollo de cada actividad:

- Desarrollo de Productos de Sistemas y Servicios: El estándar recomienda el empleo de modelos de dominio, arquitecturas de dominio, y activos reutilizables para desarrollar y mantener productos y servicios. Los actores que intervienen en estos procesos son: el comprador, el proveedor, el desarrollador y el agente de mantenimiento.

- Desarrollo de Productos de Software: Son tareas relacionadas al desarrollo, operación y mantenimiento de productos de software donde los activos reutilizables han sido integrados. Las actividades empleadas son modelos de dominio, arquitecturas de dominio, activos reutilizables y el mantenimiento de productos de software.

- Administración de la Práctica de Reutilización: El estándar agrupa las actividades de esta tarea; además, se orienta el establecimiento e implementación de las estructuras de reutilización. Por lo general, el Programa de Reutilización Proceso de administración se emplea fuera del ámbito de un proyecto o contrato específico con el fin de gestionar y coordinar la práctica de la reutilización sistemática en varios proyectos o contratos en una organización.

2.2.3.3 Reutilización en ciclo de vida de software

Esta cláusula define los procesos, las actividades y las tareas necesarias para desarrollar, operar y mantener productos independientes de software, servicios o elementos de los sistemas de software. Su enfoque principal es hacia las aplicaciones de software en lugar de los sistemas completos.

La relación de actividades y tareas se describe en la tabla 2 (IEEE COMPUTER SOCIETY, 2010).

Tabla 2. Actividades y tareas reutilización en el Ciclo de Vida de Software

Procesos	Actividades	Tareas
Procesos de Acuerdo	Adquisición	Preparación de Adquisición
		Aceptación de Adquiriente
	Proceso de Suministro	Proveedor de Licitación
		Acuerdo de Contrato
		Ejecución de Contrato
Procesos de Habilitación del Proyecto	Administración del Modelo de Ciclo de Vida	Establecimiento del Proceso
		Procesos de Evaluación
	Administración de Infraestructura	Implementación del Proceso
		Mantenimiento de la Infraestructura
	Administración del Portafolio del Proyecto	Iniciación del Proyecto
		Evaluación de Portafolio
	Gestión de Calidad	Gestión de Calidad
Procesos del Proyecto	Planeación del Proyecto	Planeación del Proyecto
	Evaluación y Control del Proyecto	Monitoreo del Proyecto
		Control del Proyecto
	Gestión de Decisión	Planeación de Decisiones
	Administración de Riesgos	Planeación de Gestión de Riesgos
		Gestión de Perfil de Riesgos
		Análisis de Riesgos

Tabla 2. (Continuación)

Procesos	Actividades	Tareas
Procesos del Proyecto	Gestión de Configuración	Planeación de la Gestión de Configuración
	Gestión de Información	Planeación de la Gestión de Información
		Ejecución de la Gestión de Información
	Medición	Planeación de la Medición
Procesos Técnicos del Sistema	Definición de Requerimientos de Entes Interesados	Identificación de los Entes Interesados
		Identificación de Requerimientos
		Grabación de Requerimientos
	Análisis de Requerimientos de Sistema	Especificación de Requerimientos del sistema
		Evaluación de Requerimientos
	Diseño Arquitectural de Sistema	Establecimiento de Arquitecturas
		Evaluación Arquitectural
	Integración de Sistema	Prueba de Disposición
	Calificación de Pruebas de Sistema	Calificación de Pruebas
Procesos de Implementación de Software	Implementación de Software	Estrategia de Implementación de Software
	Análisis de Requerimientos de Software	Análisis de Requerimientos de Software
	Diseño Arquitectural de Software	Diseño Arquitectural de Software
	Diseño Detallado de Software	Diseño Detallado de Software
	Construcción de Software	Construcción de Software
	Integración de Software	Integración de Software
	Pruebas de Calificación de Software	Pruebas de Calificación de Software
Procesos de Soporte de Software	Gestión Documental de Software	Implementación del Proceso
		Diseño y Desarrollo
	Gestión de Configuración de Software	Implementación del Proceso
	Aseguramiento de Calidad de Software	Implementación del Proceso
	Verificación de Software	Implementación del Proceso
	Validación de Software	Implementación del Proceso
	Revisión de Software	Implementación del Proceso

Tabla 2. (Continuación)

Procesos	Actividades	Tareas
Procesos de Soporte de Software	Auditoria de Software	Implementación del Proceso
	Resolución de Problemas de Software	Implementación del Proceso
	Instalación de Software	Instalación de Software
	Soporte de Aceptación de Software	Soporte de Aceptación de Software
	Operación de Software	Preparación de Operación
	Mantenimiento de Software	Implementación del Proceso
		Problema y Análisis de Modificación
		Migración
	Eliminación de Software	Planificación de Eliminación de Software
		Ejecución de Eliminación de Software

En síntesis, el estándar ofrece una perspectiva general sobre los procesos, las actividades y las tareas de reutilización de software. Estos lineamientos son establecidos dentro del ámbito internacional de tal suerte que sean considerados como punto de partida para el desarrollo profesional de activos de software.

2.3 ACTIVO DE CONOCIMIENTO: EL FUNDAMENTO PARA LA DEFINICIÓN DE ACTIVO DE SOFTWARE

Los enfoques de gestión del conocimiento se han centrado en dos perspectivas distintas: la del conocimiento como un producto y el conocimiento como un proceso (YOUNG, y otros, 2001).

El conocimiento como un producto implica que el conocimiento es un elemento que se puede capturar, distribuir, medir y gestionar como un objeto independiente. Este enfoque se centra principalmente en los productos y artefactos que contienen y que representan el conocimiento (YOUNG, y otros, 2001).

El conocimiento como un proceso, hace énfasis en las formas de promover, motivar, estimular, fomentar y guiar el proceso de conocer. Este punto de vista se refiere a la gestión del conocimiento como un proceso de comunicación social (YOUNG, y otros, 2001).

Este trabajo de investigación se concentra en el tratamiento del conocimiento como un producto y en el activo de conocimiento como fundamento conceptual para la definición y tratamiento del activo de software como un tipo de activo de conocimiento y su aplicación en la ingeniería del software.

2.3.1 Activos de conocimiento

Existen diversas definiciones acerca activo de conocimiento (ANAYA, y otros, 2008) (MENTZAS, y otros, 2003).

En (MENTZAS, y otros, 2003), los activos de conocimiento son los recursos que las organizaciones desean cultivar, pero son diferentes de los recursos de organización.

Para efectos de esta investigación se adopta el concepto de activo de conocimiento propuesto en (ANAYA, y otros, 2008): Activos de conocimiento son el conocimiento sobre mercados, productos, tecnologías y organizaciones, del que una empresa necesita apropiarse y que permiten a sus procesos de negocio generar beneficios. Este conocimiento traducido en activos, se puede utilizar de formas innumerables generando productividad y rentabilidad para la organización. La principal motivación para las organizaciones es que los activos de conocimiento se pueden aprovechar de maneras innovadoras y creativas para generar ventajas competitivas que hacen a la empresa sostenible en el mercado.

Existen estudios que clasifican los activos de conocimiento de acuerdo a diversos criterios (ANAYA, y otros, 2008), (MENTZAS, y otros, 2003) y (DINGSOYR, 2003). En (ANAYA, y otros, 2008), se referencian dos taxonomías de activos de conocimiento: En (MENTZAS, y otros, 2003), sus autores proponen que, los activos de conocimiento son: los activos de conocimiento humano, lo que genera capacidad de organización, los activos estructurales de conocimiento, que generalizan las capacidades humanas, y los activos de conocimiento del mercado, sobre los productos y servicios de la empresa. En (DINGSOYR, 2003), se introducen otros conceptos para la clasificación de activos de conocimiento: capital humano, capital relacional y capital estructural, el capital humano es el conocimiento sobre las capacidades de las personas para trabajar en la organización, sus conocimientos, habilidades y experiencia, este capital se incrementa por la socialización e interiorización. El capital relacional representa la organización del conocimiento relacionado con las relaciones externas de mercado, los clientes de la organización, socios comerciales y competidores, este capital se incrementa por la socialización. El capital estructural, es acerca de las capacidades organizacionales necesarias para alcanzar los requerimientos funcionales.

2.3.2 Activos de conocimiento en las organizaciones de software

El proceso de desarrollo de software es una actividad intensiva en conocimiento (ANAYA, y otros, 2006). Para la organización de software, el éxito de su trabajo depende en gran medida del grado en que el equipo entiende la organización, los procesos, los métodos y los estándares. Este conocimiento puede ser descrito, estructurado y compartido por medio de activos de conocimiento.

En consecuencia, la importancia del activo de conocimiento es evidente porque se constituye factor clave de éxito para la organización de software.

Aunque el conocimiento nace en la mente de la persona debido al aprendizaje individual en el desarrollo de sus tareas, dicho aprendizaje individual no representa un activo de la organización a menos que se provean mecanismos para que estas experiencias individuales puedan ser socializadas, exteriorizadas, combinadas e internalizadas. Es en este punto donde se puede afirmar que existe un aprendizaje organizacional (ANAYA, y otros, 2006).

En el marco del desarrollo de software, el conocimiento puede ser visto en diferentes niveles de refinamiento y que de alguna manera se pueden entender como los niveles de madurez del conocimiento. El primer nivel está representado por los datos puntuales relacionados a un simple proyecto o evento tales como métricas recolectadas de un proyecto y lecciones aprendidas del proyecto. A partir de los datos coleccionados de múltiples proyectos se pueden construir modelos que contienen más información y que son aplicables a nuevos proyectos, este es el segundo nivel. Finalmente, el tercer nivel está representado por el conocimiento suficientemente abstracto y generalizado que ha sido madurado con su aplicación y que puede verse representado como “mejores prácticas” y estándares (ANAYA, y otros, 2006).

Los conocimientos en ingeniería de software se clasifican en las siguientes categorías: conocimiento organizacional, conocimiento de gestión, conocimiento técnico y conocimiento del dominio. Las categorías de organización y gestión representan el conocimiento sobre las políticas y estrategias en torno a la empresa. La mayoría de estos conocimientos se mantienen estables durante la vida de la empresa. El conocimiento técnico y el conocimiento de dominio representan el conocimiento crítico en una organización de software, el conocimiento técnico es acerca de las habilidades para realizar algunas actividades o tareas durante el proceso de software, relacionado con los métodos y las tecnologías apropiadas para la organización en general y para el proyecto en particular. Este conocimiento es dinámico y cambia con frecuencia, pero puede ser descrito explícitamente con estándares y patrones. El conocimiento del dominio se refiere al conocimiento sobre el dominio del que el software trata. Este conocimiento es diferente en cada proyecto y es obtenido de fuentes estáticas y dinámicas durante la elicitación de procesos. Por lo general, este conocimiento está representado por diagramas, modelos, las bibliotecas y frameworks (ANAYA, y otros, 2008).

A pesar de la cantidad de conocimiento ya estructurado y divulgado a nivel mundial en el campo de la ingeniería de software en forma de modelos, estándares, métodos, técnicas, lenguajes, patrones, etc. es importante resaltar que el conocimiento generado en una empresa de software representa un activo significativo para la organización que es urgente gestionar eficientemente.

Aprender de la experiencia previa, demanda cierto ajuste para trasladar el conocimiento a dominios específicos, lo que supone reutilizar aspectos organizacionales y técnicos. Las experiencias generadas en un proyecto se convierten en activos de conocimiento que alimentan la base de conocimiento y que permiten ir enriqueciendo el modelo a partir de la experiencia de su aplicación en los diferentes proyectos de software.

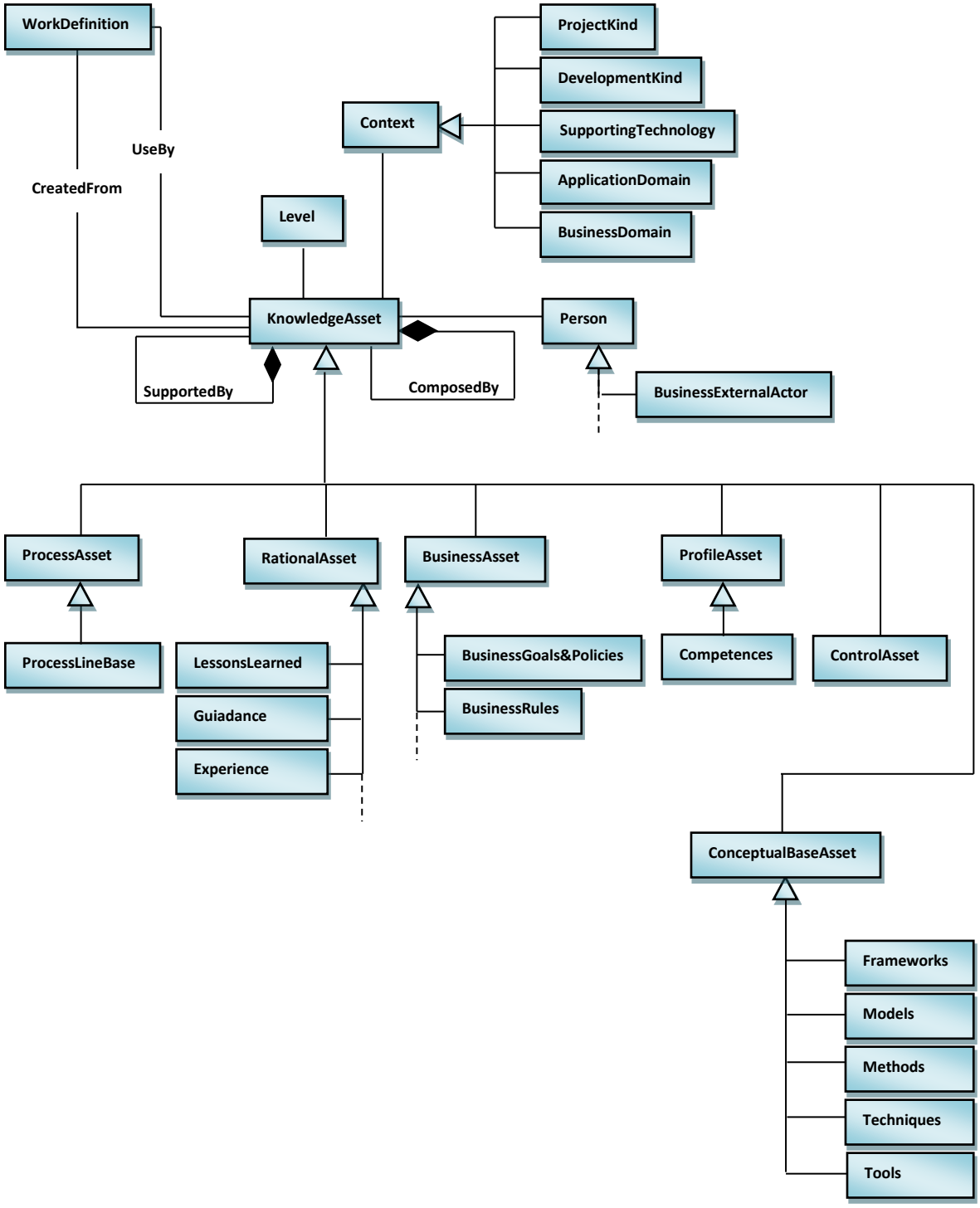
Los proyectos de software suelen involucrar gran cantidad de personas con diferentes antecedentes e intereses y con necesidad de trabajar en forma coordinada y con mecanismos que faciliten la comunicación fluida para el intercambio de información, conocimientos y experiencias entre ellos. Los conocimientos en Ingeniería de Software son diversos y sus proporciones inmensas y creciendo de manera sostenida, y las organizaciones tienen problemas en identificar el contenido, la ubicación y el uso de ese conocimiento.

2.3.3 Un metamodelo para el activo de conocimiento en ingeniería de software

En (ANAYA, y otros, 2008), se propone un metamodelo que permite caracterizar los activos de conocimiento y sus relaciones en una organización de software. En este trabajo de investigación, se parte de este metamodelo propuesto y se extiende con el propósito de caracterizar (especializar) aquellos activos que representan el conocimiento acerca de los productos software en cualquiera de sus fases y que se denominan activo de software.

El metamodelo propuesto en (ANAYA, y otros, 2008), parte de la definición de activo de conocimiento (Ver figura 3): Activos de conocimiento son el conocimiento sobre mercados, productos, tecnologías y organizaciones, del que una empresa necesita apropiarse y que permiten a sus procesos de negocio generar beneficios. Los activos de conocimiento constituyen el capital intelectual de una organización de software y soportan el proceso organizacional en conjunto. Los Activos de conocimiento son generados y usados por unidades de trabajo, y requiere recursos (infraestructura, personas), depende de un contexto (dominio de una aplicación particular, cliente, tecnología de soporte) y es producido, validado y actualizado por personas.

Figura 3. Metamodelo de activo de conocimiento



Fuente: (ANAYA, y otros, 2008)

2.3.3.1 Tipos de activos de Conocimiento

A continuación se relacionan los tipos de activos de conocimiento presentados en (ANAYA, y otros, 2008):

- Activo Rational: Se refiere a la definición y documentación de experiencias, guías, decisiones de diseño, lecciones aprendidas, etc. que permiten aprender de acciones y decisiones anteriores. Un activo rational se categoriza de acuerdo a su nivel de madurez, es decir, de acuerdo al grado de estandarización en la organización y de acuerdo al nivel de análisis que implica. Un activo rational describe de forma explícita el contexto en el que se originó y la justificación de su aplicación.

- Activo de Proceso: Este tipo de activo incluye: definiciones completas de los procesos de ciclo de vida, los elementos del proceso, y las arquitecturas de proceso genérico. Los activos de procesos son gestionados mediante una biblioteca de activos de proceso (Process Active library - PAL). Bajo esta concepción un proceso de desarrollo se define a partir de una línea base de proceso que facilita el uso de procesos ya definidos y disponibles en la biblioteca. Este enfoque promueve la colaboración y el trabajo en equipo al hacer que las actividades, funciones, responsabilidades y dependencias sean visibles a todo el personal.

- Activo de infraestructura: La adaptación de un proceso exige la identificación y definición de los recursos y la infraestructura necesarios para realizar sus actividades. Los activos de infraestructura permiten esta identificación. Un activo de infraestructura puede incorporar el conocimiento sobre la composición, organización, medios de comunicación y perfil del equipo de trabajo. Un activo de infraestructura también puede incorporar el conocimiento referido a la adopción de herramientas externas o internas requeridas para el desarrollo de las actividades de un proceso.

- Activo de negocio: Representa el conocimiento del negocio de la organización cliente. Los desarrolladores deben entender el propósito, las prácticas actuales y la estrategia de TI de la empresa cliente. Para ello, el conocimiento del negocio debe ser documentado explícitamente. Parte de este conocimiento es representado por modelos que aplican prácticas tales como modelado de procesos de negocio, modelado de negocio (modela tanto las entidades estructurales del negocio como las prácticas del negocio), modelado de la arquitectura empresarial. Las prácticas del negocio están regidas por las reglas del negocio. Las reglas de negocio pueden ser: definiciones de términos de negocio, restricciones de integridad de datos, derivaciones matemáticas y funcionales, inferencias lógicas, secuencias de procesamiento y las relaciones entre los hechos del negocio.

- Activo de perfil: Incorpora el conocimiento sobre las capacidades y competencias del recurso humano. Este activo ayuda a mejorar las organizaciones mediante el aumento de la capacidad de los individuos y la alineación de su motivación con la de la organización. Este activo ayuda también, a retener capital humano, es decir, personal con conocimiento y habilidades críticas dentro de la organización. El conocimiento sobre las habilidades individuales y de equipo es un activo estratégicos competitivo que debe ser correctamente modelado, gestionado y almacenado. El conocimiento claro del personal de la organización facilita el reclutamiento, la selección y la asignación de tareas en un proyecto. Los activos de perfil ayuda a la organización a definir políticas de productividad individual y de grupo, entrenamiento y formación.

- Activo de Control: Cada elemento de un proyecto debe estar bajo control. Se deben tomar medidas apropiadas para garantizar que los elementos del proyecto sean definidos y controlados. Para lograr este control, se utilizan diversas formas de control como por ejemplo: software de pruebas y medición de atributos de calidad. Siempre que un producto es elaborado, o un proceso definido y ejecutado, los planes afectados, los productos de trabajo y las actividades se ajustan para mantener la coherencia y estar bajo control. Los activos de control debe proporcionar la visibilidad adecuada sobre cómo los elementos deben ser controlados. Esto implica que las diferentes versiones del elemento se conocen (control de versiones) y los cambios se incorporan de una manera controlada (control de cambio). Ejemplos de elementos a ser controlados son estimaciones (tamaño, esfuerzo, tiempo) y atributos calidad de un producto.

- Activo de Base Conceptual: Es el conocimiento adicional requerido para hacer práctico un activo de conocimiento en un contexto. Un activo de base conceptual puede incluir conocimiento sobre métodos, técnicas, prácticas y herramientas. Varios activos de base conceptual pueden ser integrados para elegir la mejor base conceptual para realizar determinada tarea que permita alcanzar la mejor solución.

2.3.3.2 Relación entre tipos de activos de Conocimiento

La estructura de activos de conocimiento propuesta en (ANAYA, y otros, 2008), define un conjunto de relaciones entre activos del conocimiento: Varios activos del conocimiento, que componen las estructuras complejas de forma recursiva, podrían apoyar un activo de conocimiento en particular. Por ejemplo, una técnica de obtención (activo de base conceptual) puede apoyar un proceso de obtención (activos de proceso), una regla de negocio (activos de negocio) pueden apoyar una decisión (activo rational), una herramienta (activo de base conceptual) podrá apoyar una técnica (activo de base conceptual). Varios activos de conocimiento podrían componer un activo de conocimiento en particular. El significado de esta

composición recursiva es que un activo de conocimiento potencialmente puede formar parte de un conjunto de activos de conocimiento de mayor granularidad. Por ejemplo, una lección aprendida (activo rational) pueden necesitar otras lecciones anteriores utilizadas para apoyar una decisión, una técnica (activo de base conceptual) puede necesitar otras técnicas para obtener resultados adecuados.

2.4 REPOSITORIOS Y CATALOGOS EN INGENIERIA DE SOFTWARE

Aunque existen trabajos sobre reutilización que utilizan el concepto repositorio como sinónimo de catálogo, por ejemplo (BERNSTEIN, 1998), se reconocen diferencias conceptuales entre los dos términos:

Un catálogo de reutilización es un conjunto de descripciones de activos con una referencia o apuntador de donde el activo es actualizado, almacenado o hay información acerca de cómo puede ser adquirido (IEEE COMPUTER SOCIETY, 2010). El conjunto de descripciones permite clasificar activos de software potencialmente reutilizables.

Un repositorio digital, es una base de datos con ciertas características adicionales: ubicación y enfoque institucional, centrados en productos de investigación, visibilidad vía Web, disponibilidad de documentos completos (para descargas según políticas de autenticación), información estructurada a partir de metadatos y sostenibilidad y administración garantizada del sistema a largo plazo (WHITEHEAD, 2005).

Por lo anterior se deduce que en ingeniería de software, los términos catálogo y repositorio son complementarios: catálogo como una propuesta para clasificar artefactos de software potencialmente reutilizables, es decir, la vista conceptual del activo y repositorio digital como una solución más general que plantea los mecanismos para almacenar y compartir activos, es decir, la vista lógica del activo.

2.4.1 Trabajos relacionados con repositorios y catálogos de activos de software

En el campo de la ingeniería del software existen trabajos de investigación que se han ocupado de la gestión de artefactos software. Sus autores utilizan de manera indistinta los términos catálogo y repositorio. A continuación se describen brevemente los que se consideran más relevantes:

WindowsClient.Net, es un catálogo fue creado por la empresa Microsoft Corporation, posee patrones de desarrollo para la construcción de aplicaciones en plataformas Windows (WINDOWSCLIENT.NET COMUNITY, 2011).

DoFactory es un framework que comprende un completo repositorio digital de patrones de diseño de software desarrollado en tecnología .NET, específicamente C# y Visual Basic (DATA & OBJECT FACTORY, 2001).

Quince es un catálogo desarrollado por Infragistics, Inc., provee patrones de experiencia e interfaz de usuario, entendiendo por experiencia de usuario como la forma en que un cliente percibe la facilidad de uso y eficiencia de un producto software o sistema (INFRAGISTICS, INC. QUINCE, 2009).

Welie patterns, es un catálogo que provee una gran cantidad de patrones de diseño de interacción (WELIE, Martin Van, 2008).

En (CARRERON, 2008), se presenta una primera aproximación a un procedimiento para la construcción de un catálogo de patrones de requisitos funcionales en el dominio de los sistemas Enterprise Resource Planning (ERP). La construcción de dicho catálogo tiene la finalidad de contribuir a mejorar el proceso de selección de ERP mediante patrones que facilitan la reutilización de requisitos. Para llegar a esta propuesta de procedimiento, se han analizado los temas relacionados llegando a un estado del arte de cada uno de ellos, se ha proporcionado una definición de patrón de requisitos funcional, y se ha realizado la extracción de patrones de un conjunto de libros de requisitos obtenidos de proyectos reales de selección de ERP.

Welicki en (WELICKI, 2006), propone un modelo de meta-especificación y catalogación de patrones y conceptos como respuesta a las necesidades de gestión del conocimiento en éste ámbito de la ingeniería del software. La arquitectura general de la solución propuesta se compone de un lenguaje de meta-especificación para describir a los patrones a un alto nivel de abstracción, un catálogo de patrones creados con ese lenguaje, una infraestructura de catalogación y una herramienta de explotación del catálogo. Para verificar la factibilidad de la solución propuesta, el autor ha creado y evaluado un prototipo respecto a otras soluciones y enfoques existentes para demostrar que el modelo propuesto supera las dificultades recurrentes encontradas en otros enfoques.

En (FUENTES, y otros, 2004), se presenta una propuesta que soporta el diseño e implementación de aplicaciones basadas en componentes y aspectos. Este trabajo describe un entorno integrado de desarrollo (EID) para aplicaciones basadas en componentes y aspectos y una serie de herramientas para dar soporte a dicho entorno. El entorno incluye una plataforma de gestión del repositorio de componentes y aspectos denominada DAOP, cuya característica principal es la composición dinámica de componentes y aspectos. Por otro lado se presenta un lenguaje de definición de arquitecturas (ADL o Architecture Definition Languaje), denominado DAOP-ADL, que sirve para la especificación de la arquitectura de una aplicación y que la plataforma usa para componer componentes y aspectos. El EID incluye además, un repositorio de componentes y aspectos (RCA).

El trabajo presentado por López, Laguna y Marqués en (LÓPEZ, y otros, 2001), propone una aproximación para incorporar el proceso de reutilización desde las etapas iniciales del ciclo de vida del software. La propuesta se basa en generalizar escenarios del problema y compararlos con escenarios genéricos. Los primeros se obtienen con una herramienta automatizada que está en fase de desarrollo. La generalización se realiza mediante reducción de redes de Petri. Los escenarios genéricos se almacenan en un repositorio asociados a estructuras complejas de reutilización denominadas mecanos que enlazan elementos de análisis, diseño e implementación. La comparación se realiza mediante analogía. De este modo, con una estrategia para reutilizar software desde los requisitos funcionales, se ofrece una alternativa para acelerar el desarrollo de soluciones software con reutilización a la vez que se eleva el nivel de abstracción de los elementos reutilizables. La motivación principal del artículo es que los requisitos representan el conocimiento más abstracto del dominio y un alto porcentaje de ellos son reutilizables.

En el desarrollo del proyecto UNC-Corpus Corpus de diagramas UML para la solución de problemas de completitud en ingeniería de software se hace uso de repositorios de diagramas con el fin de optimizar la gestión de los activos de modelado que allí utilizan además de complementarlo con herramientas para la verificación de consistencia y creación de diagramas (ZAPATA, y otros, 2008).

2.5 METADATOS: UN MECANISMO PARA LA DESCRIPCION DE ACTIVOS DE SOFTWARE

Los metadatos son datos que describen otros datos. En general, un conjunto de metadatos se refiere a un grupo de datos, llamado recurso. El concepto de metadatos es análogo al uso de índices para localizar objetos en vez de datos (THE DUBLIN CORE® METADATA INITIATIVE, DCMI, 1995). Un recurso puede ser: una imagen, un documento, un diagrama, un modelo, etc.

La asociación de metadatos descriptivos normalizados a los recursos de un repositorio digital, tiene el potencial para mejorar sustancialmente las capacidades de localización/recuperación, facilitando búsquedas basadas en campos (p. ej. autor, título), permitiendo la indización de objetos no textuales, y facilitando el acceso al contenido sustituido/referenciado que es distinto del acceso al contenido del propio recurso (THE DUBLIN CORE® METADATA INITIATIVE, DCMI, 1995).

Para capitalizar esta oportunidad de mejoramiento, se han desarrollado a nivel internacional, sistemas estandarizados de catalogación para este tipo de información. Uno de los más utilizados es el sistema Dublin Core. El estándar Dublin Core puede ser extendido para crear conjuntos de metadatos para la descripción de cualquier tipo de recurso. En este trabajo se extiende el estándar Dublin core y se genera DC-soft, el conjunto de metadatos para la descripción de activos de software.

2.5.1 Dublin Core: un modelo de metadatos para la descripción de recursos digitales

Dublin Core es un modelo de metadatos elaborado y auspiciado por la DCMI (Dublin Core Metadata Initiative), una organización dedicada a fomentar la adopción extensa de los estándares interoperables de los metadatos y a promover el desarrollo de los vocabularios especializados de metadatos para describir recursos para permitir sistemas más inteligentes del descubrimiento del recurso (THE DUBLIN CORE® METADATA INITIATIVE, DCMI, 1995).

Dublin Core es el conjunto de metadatos más utilizado, este sistema de definiciones fue diseñado específicamente para proporcionar un vocabulario de características base, capaces de proporcionar la información descriptiva básica sobre cualquier recurso, sin que importe el formato de origen, el área de especialización o el origen cultural. Las implementaciones de Dublin Core usan generalmente XML y se basan en el Resource Description Framework. Dublin Core se define por ISO en su norma ISO 15836 del año 2003, y la norma NISO Z39.85-2007 (THE DUBLIN CORE® METADATA INITIATIVE, DCMI, 1995). Dublin Core es un sistema de 15 definiciones semánticas descriptivas que pretenden transmitir un significado semántico a las mismas.

En general, los elementos se clasifican en tres grupos que indican la clase o el ámbito de la información que se guarda en ellos:

- Elementos relacionados con el contenido del recurso.
- Elementos relacionados con el recurso cuando es visto como una propiedad intelectual.
- Elementos relacionados con la instanciación del recurso.

Tabla 3. Elementos del Estándar Dublin Core

Contenido	Propiedad Intelectual	Instanciación
Titulo	Autor	Fecha
Claves	Editor	Tipo
Descripción	Otros colaboradores	Formato
Fuente	Derechos	Identificador
Idioma		
Relación		
Cobertura		

2.5.1.1 Elementos Dublin Core relacionados con el contenido

Tabla 4. Elementos Dublin Core Relacionados con el Contenido

Elemento	Descripción	Etiqueta
Titulo	El nombre dado a un recurso, habitualmente por el autor.	DC.Title
Claves	Los tópicos del recurso. Típicamente, expresa las claves o frases que describen el título o el contenido del recurso. Se fomentará el uso de vocabularios controlados y de sistemas de clasificación formales.	DC.Subject
Descripción	Una descripción textual del recurso. Puede ser un resumen en el caso de un documento o una descripción del contenido en el caso de un documento visual.	DC.Description
Fuente	Secuencia de caracteres usados para identificar unívocamente un trabajo a partir del cual proviene el recurso actual.	DC.Source
Idioma	Idioma o idiomas del contenido intelectual del recurso.	DC.Language
Relación	Es un identificador de un segundo recurso y su relación con el recurso actual. Este elemento permite enlazar los recursos relacionados y las descripciones de los recursos.	DC.Relation
Cobertura	Es la característica de cobertura espacial y/o temporal del contenido intelectual del recurso. La cobertura espacial se refiere a una región física, utilizando por ejemplo coordenadas. La cobertura temporal se refiere al contenido del recurso, no a cuándo fue creado (que ya lo encontramos en el elemento Date).	DC.Coverage

2.5.1.2 Elementos Dublin Core relacionados con la propiedad intelectual

Tabla 5. Elementos Dublin Core Relacionados con la Propiedad Intelectual

Elemento	Descripción	Etiqueta
Autor	La persona u organización responsable de la creación del contenido intelectual del recurso. Por ejemplo, los autores en el caso de documentos escritos; artistas, fotógrafos e ilustradores en el caso de recursos visuales.	DC.Creator
Editor	La entidad responsable de hacer que el recurso se encuentre disponible en la red en su formato actual.	DC.Publisher
Otros Colaboradores	Una persona u organización que haya tenido una contribución intelectual significativa, pero que esta sea secundaria en comparación con las de las personas u organizaciones especificadas en el elemento Creator. (por ejemplo: editor, ilustrador y traductor).	DC.Contributor

Tabla 5. (Continuación)

Elemento	Descripción	Etiqueta
Derechos	Son una referencia (por ejemplo, una URL) para una nota sobre derechos de autor, para un servicio de gestión de derechos o para un servicio que dará información sobre términos y condiciones de acceso a un recurso.	DC.Rights

2.5.1.3 Elementos Dublin Core relacionados con la instanciación

Tabla 6. Elementos Dublin Core Relacionados con la Instanciación

Elemento	Descripción	Etiqueta
Fecha	Fecha en la cual el recurso se puso a disposición en su forma actual. Esta fecha no se tiene que confundir con la que pertenece al elemento Coverage, que estaría asociada con el recurso en la medida que el contenido intelectual está de alguna manera relacionado con aquella fecha.	DC.Date
Tipo	La categoría del recurso. Por ejemplo, página personal, romance, poema, diccionario, etc.	DC.Type
Formato	Es el formato de datos de un recurso, usado para identificar el software y, posiblemente, el hardware que se necesitaría para mostrar el recurso.	DC.Format
Identificador	Secuencia de caracteres utilizados para identificar unívocamente un recurso. Ejemplos para recursos en línea pueden ser URLs o URNs. Para otros recursos pueden ser usados otros formatos de identificadores, como por ejemplo ISBN ("International Standard Book Number").	DC.Identifier

2.6 PROPUESTAS DE DESARROLLO DE SOFTWARE Y SU ORIENTACIÓN HACIA LA REUTILIZACIÓN Y EL TRABAJO CON CATÁLOGOS

Actualmente, existen numerosas propuestas de desarrollo de software que incluyen implícitamente la reutilización y el uso de formas de catálogo, por ejemplo: FODA (FEI, y otros, 2003) y CBSD (CRNKOVIC, y otros, 2002). Otras propuestas de desarrollo de Software, no incluyen implícitamente la reutilización y el uso de catálogos, sin embargo, proporcionan la flexibilidad requerida para implementarlos RUP (AMBLER, 2005) y AUP (AMBLER, 2005).

En este trabajo de investigación, se analizan las propuestas de desarrollo de software RUP (AMBLER, 2005), FODA (FEI, y otros, 2003), CBSD (CRNKOVIC, y otros, 2002) y AUP (AMBLER, 2005), a fin de identificar los tipos de artefactos de software que definen y que son potencialmente reutilizables. Los criterios de análisis empleados son: objetivo, modelo, proceso, activos generados y la orientación hacia el trabajo con catálogos.

2.6.1 Rational Unified Process (RUP)

2.6.1.1 Objetivo de la propuesta RUP

RUP es un método de software probado para la construcción de sistemas computacionales complejos. Basado en el enfoque evolutivo, RUP tiene como objetivo principal la construcción de sistemas complejos de software bajo lineamientos claros de ingeniería, siguiendo fielmente la definición de ciclo de vida, fases y disciplinas (AMBLER, 2005).

Basado en la definición de Krutchen (KRUCHTEN, 2004), RUP es un proceso de desarrollo de sistemas [computacionales] bien definido, con frecuencia orientado hacia el uso de objetos y tecnologías basadas en componentes. Fundamentado en la Ingeniería de Software, RUP se basa en el proceso iterativo/evolutivo, conducido por los requerimientos y centrado en la arquitectura de desarrollo de software.

2.6.1.2 Modelo de la propuesta RUP

RUP es presentado dentro de un modelo marco de trabajo de procesos (Process Framework); este marco de trabajo está claramente descrito por una matriz bidimensional de fases y disciplinas. De acuerdo a Ambler, RUP define una forma estructurada desde la cual pueden ser creados los procesos (AMBLER, 2005). De hecho, RUP insta a adaptar el proceso a realizar para cada proyecto para hacer frente a sus necesidades particulares. Este modelo aboga claramente para que las organizaciones estandaricen los procesos a fin de responder a necesidades particulares.

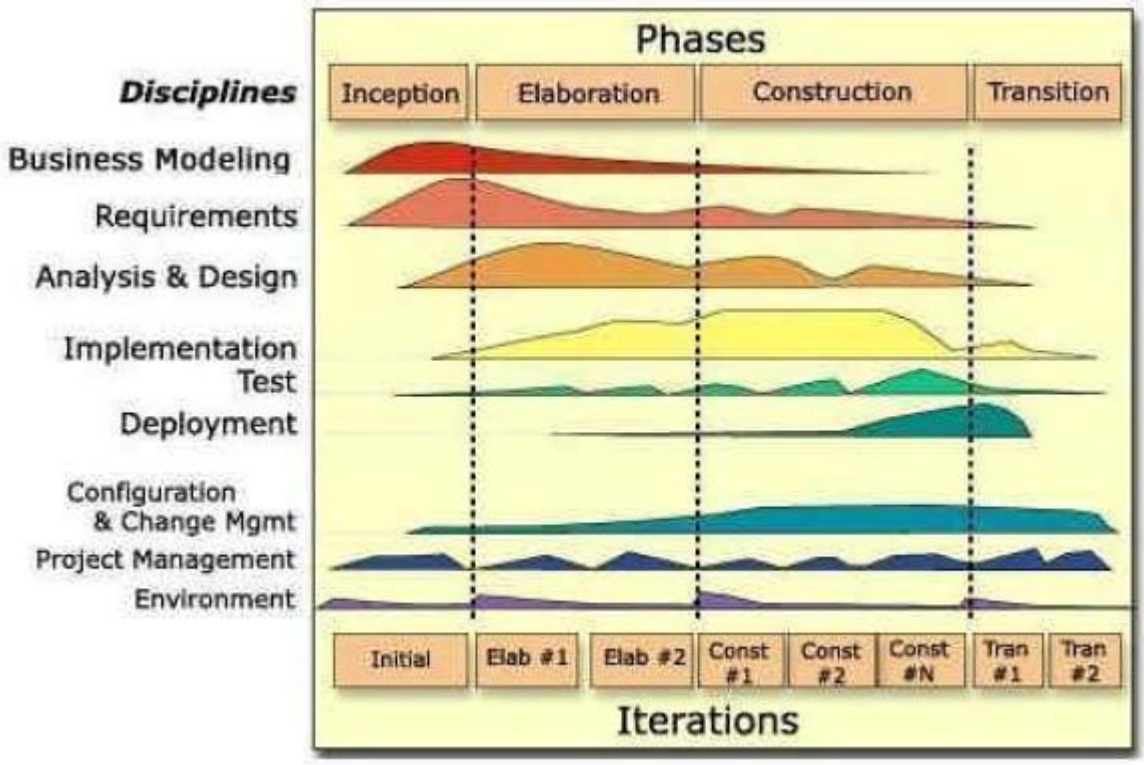
2.6.1.3 Activos de la propuesta RUP

La propuesta RUP está estructurada en dos dimensiones: la primera dimensión hace referencia a las fases, las cuales representan los cuatro grandes escenarios donde el proyecto se ejecuta a través del tiempo (Inicio, Elaboración, Construcción y Transición). La segunda dimensión se refiere a las disciplinas, las cuales representan las actividades lógicas que tienen lugar en la ejecución del proyecto (Modelado de Negocio, Requerimientos, Análisis y Diseño, Implementación, Pruebas, Despliegue, Configuración y Gestión de Cambios, Gestión del Proyecto, y Entorno).

Por consiguiente, cada intersección (entre fases y disciplinas) genera activos de software; estos son: Modelo del Dominio, Modelo de casos de uso del negocio, Glosario, Documento de Visión, Modelo de casos de uso, Especificaciones adicionales, Modelo de diseño, Documento de arquitectura, Modelo de datos, Diagramas de Implementación, Pruebas y Despliegue, y Planes de Transición. Se debe considerar que cada uno de estos activos representa una colección de activos simples que lo conforman.

2.6.1.4 Proceso de la propuesta RUP

Figura 4. Proceso de la propuesta RUP



Fuente: (AMBLER, 2005)

El proceso RUP incluye cuatro fases: inicio, elaboración, construcción y transición. Una fase se puede realizar mediante una o varias iteraciones. Una iteración es la ejecución de un conjunto de disciplinas. Las disciplinas en RUP son: modelado de negocio, requerimientos, análisis y diseño, implementación, pruebas, despliegue, gestión de configuración y cambios, gestión del proyecto, y ambiente. Una iteración incluye la ejecución de varias disciplinas con diferentes niveles de esfuerzo, de acuerdo con la fase que se esté realizando (figura 4).

2.6.1.5 Orientación hacia la reutilización y el trabajo con catálogos en la propuesta RUP

De acuerdo con los autores del modelo RUP de IBM, se ha diseñado un módulo completo que es integrable a IBM® Rational® Method Composer V7.1 para el manejo de RUP en materia de reutilización de activos de software. Dicha propuesta llamada Desarrollo Basado en Activos (Asset-Based Development - ABD) describe un proceso para la gestión de activos reutilizables, la producción de activos reutilizables, y el consumo de activos reutilizables. El plug-in de desarrollo basado en activos también proporciona herramientas asistidas que apoyan

procesos de reutilización. Los activos de gobierno establecen las políticas, las organizaciones, habilita los equipos, establece las herramientas, y determina lo que se va a medir a través de ABD (IBM, 2007).

2.6.2 Computer Based Software Development (CBSD)

2.6.2.1 Objetivo de la propuesta CBSD

El desarrollo de software basado en componentes tiene como objeto producir sistemas escalables de software a través de la reutilización explicita de componentes construidos para tal fin. En este contexto, el desarrollo de software basado en componentes resume sus propósitos bajo ciertos conceptos, principalmente el de componente. Un componente es una unidad reutilizable de despliegue y de composición. Una opinión común es que un componente está estrechamente relacionado con un objeto y que el CBSD es por lo tanto, una extensión del desarrollo orientado a objetos (CRNKOVIC, y otros, 2002).

2.6.2.2 Modelo de la propuesta CBSD

El desarrollo de software basado en componentes forma parte de una sólida propuesta teórica conocida con el nombre de Ingeniería de Componentes. La Ingeniería de Componentes por definición es un enfoque de desarrollo de software basado en reutilización (SAMETINGER, 2010). La reutilización sistemática requiere de una base de componentes de alta calidad con la documentación adecuada. Dichos componentes no pueden ser simplemente extraídos de las aplicaciones existentes; de hecho, la obtención de componentes reutilizables requiere más esfuerzo ya que dichos componentes debieron haber sido diseñados generalmente para necesidades especiales.

Teniendo en cuenta la inclinación marcada hacia la fase de diseño e implementación, el desarrollo de software basado en componentes ha plasmado su filosofía en estándares de la industria. Es por tal razón que hoy por hoy, se hablan de modelos de desarrollo de software basado en componentes a la luz de CORBA (propuesto por OMG), EJB (propuesto por Oracle, antiguamente por Sun Microsystems) y DCOM/COM+ (propuesto por Microsoft), entre otros.

Todos los modelos anteriores se basan en la abstracción general de la propuesta la cual integra los grandes conceptos de componentes, interfaces, contratos, patrones y frameworks.

Esto quiere decir que se requiere una metodología para la construcción de componentes, para su validación, pruebas y control de calidad, y de unos mecanismos de distribución de componentes a través de repositorios de donde dichos elementos puedan ser consumidos. Con esto se evidencia la naturaleza del modelo hacia ambientes de reutilización explicita.

2.6.2.3 Activos de la propuesta CBSD

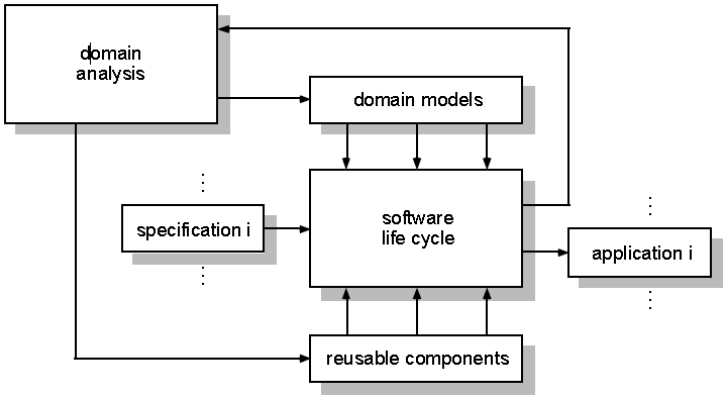
Los activos de la propuesta CBSD son: modelo de requisitos, bibliotecas de componentes, modelo de diseño, modelo de implementación, esquema de datos, sistema integrado y componentes, modelo de despliegue, casos de prueba, documento del entorno de configuración y documentación de soporte y mantenimiento

Teniendo en cuenta que el CBSD profundiza su quehacer en el aspecto de implementación, los activos generados a través de su ciclo son supremamente puntuales en cuanto a nomenclatura. La documentación que generan 4 actores (el diseñador, el arquitecto, el desarrollador y el administrador) son establecidas directamente como activos escritos en UML

2.6.2.4 Proceso de la propuesta CBSD

El desarrollo de software basado en componentes reinterpreta el concepto de ciclo de vida para desarrollar pequeñas iteraciones en cuanto a la secuencialidad de las fases. Como lo muestra Sametinger, los procesos de la propuesta de CBSD van de la mano con los procesos propios de la ingeniería de dominio. Es aquí donde el desarrollo de software basado en componentes se relaciona estrechamente con el análisis de dominio y sus especificaciones:

Figura 5. Ciclo de vida de CBSD integrado con ingeniería de dominio



Fuente: (SAMETINGER, 2010)

A partir de la figura 6, se puede evidenciar que el concepto de componentes reutilizables es generado a través del análisis del dominio. Es aquí donde los activos de software deben ser filtrados a través del escenario de Generalización de Componentes (ajustarlo a lineamientos industriales) a fin de tener éxito en la construcción de aplicación usando componentes específicos.

Finalmente, el desarrollo de software basado en componentes, interactúa con la reutilización en los siguientes niveles de ciclo de vida clásico.

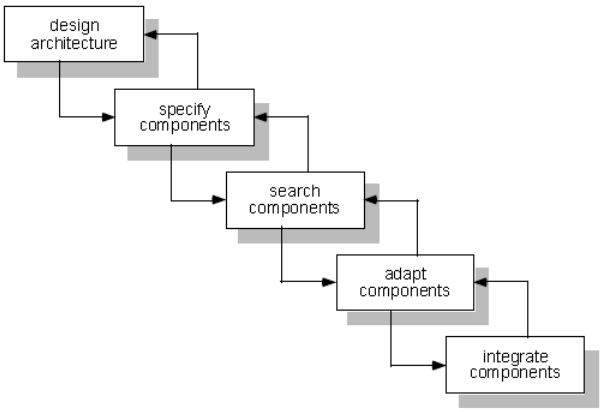
Esto indica que el ciclo de vida de desarrollo necesariamente asocia elementos de evaluación para la obtención de componentes existes. Otra evidencia de que la reutilización es un factor natural de la propuesta.

2.6.2.5 Orientación hacia la reutilización y el trabajo con catálogos en la propuesta CBSD

La reutilización debe ser vista como una parte integral del ciclo de vida del software sobre todo bajo una orientación basada en componentes. En el enfoque tradicional de reutilización, es común buscar los componentes necesarios después de que la mayor parte del trabajo de diseño se ha hecho. En lugar de esperar hasta que el diseño se haga (y luego se busca en vano los componentes a integrar), los productos de software deben ser diseñados en torno a los componentes de software disponibles. En este sentido el Desarrollo de Software basado en componentes tiene una alta inclinación hacia las estrategias de reutilización, especialmente en su fase de implementación.

De acuerdo con (SAMETINGER, 2010), las fases del ciclo de vida del desarrollo de software basado en componentes involucra aspectos de reutilización por cada transición. La figura 6 muestra como dichas transiciones re-toman activos prediseñados, en este caso componentes de software.

Figura 6. CBSD con reutilización



Fuente: (SAMETINGER, 2010)

Según lo anterior, el modelo de la propuesta CBSD se enfoca hacia 5 grandes escenarios de acción:

- Desarrollo de Componentes: Metodologías asociadas a la fase de diseño e implementación de los componentes como tales.

- La Generalización de Componentes: Ajuste a lineamientos de la industria (tales como CORBA, EJB, COM+, entre otros)

- La Certificación de Componentes: Filtro donde se hace una medición de calidad de los componentes a ser reutilizados. Se siguen pruebas prediseñadas para los componentes construidos.

- El uso de Repositorios de Componentes: se establecen lugares donde residen los componentes reutilizables, estos pueden ser: repositorios locales, repositorios de dominio especifico, y repositorios de referencia.

- La Clasificación de Componentes: Varios aspectos son considerados en cuanto a clasificación de componentes. Entre ellos están: Clasificación por función, por objeto, por medio, áreas funcionales, configuraciones, etc.

2.6.3 Feature Oriented Domain Analysis (FODA)

2.6.3.1 Objetivo de la propuesta FODA

FODA tiene como objetivo, identificar y modelar los elementos comunes y variables del sistema en un dominio durante el análisis de dominio, enfocado en las características de los dominios del sistema, donde las características que más se ven influenciadas al aplicar el enfoque FODA son las características de mantenibilidad, entendimiento y reutilización (FEI, y otros, 2003).

FODA se fundamenta en dos conceptos: abstracción y refinamiento. La abstracción es usada para crear productos de dominio de aplicaciones específicas y el refinamiento para mejorar estos productos.

El producto primario de FODA es un framework estructurado de modelos relacionados que catalogan los resultados del análisis de dominio.

2.6.3.2 Modelo de la propuesta FODA

El modelo de la propuesta FODA se basa en la Orientación a Características (Featured-Oriented). Las salidas del modelado de características serán elementos reutilizables para ser usados en la fase de ingeniería de aplicación de productos software (FEI, y otros, 2003). FODA se enfoca en las características de los dominios del sistema. Entonces una característica es, de acuerdo a FODA, un aspecto, calidad o característica de un sistema visible al usuario final (FEI, y otros, 2003).

2.6.3.3 Activos de la propuesta FODA

La propuesta FODA genera los siguientes activos: Modelo de Contexto, Modelo de dominio, Modelo de Características, Modelo de Información, Modelo Funcional u Operacional, Diccionario de dominio, Modelo de Arquitectura, Solución Software y Diagrama de características.

2.6.3.4 Proceso de la propuesta FODA

El proceso FODA incluye tres fases: Análisis de contexto, Modelado de Dominio y Modelado de Arquitectura. La fase de modelado de dominio consta de tres actividades principales: Análisis de características, Análisis de Información y el Análisis Operacional.

2.6.3.5 Orientación hacia la reutilización y el trabajo con catálogos en la propuesta FODA

FODA hace uso de catálogos para gestionar características de dominio que son instanciadas para caracterizar problemas particulares dentro del dominio. Las características del dominio se identifican a partir del análisis de características comunes de sistemas que integran el dominio.

Un ejemplo del uso de FODA orientado al desarrollo de catálogos, es el realizado por el SEI en conjunto con el departamento de defensa de los Estados Unidos llamado “Application of Featured-Orientes Domain Analysis to the Army Movement Control Domain” (COHEN, 1992). Este trabajo presenta un conjunto de catálogos de características del dominio.

El desarrollo del catálogo incluye el análisis de características, su clasificación y posterior modelado. Después de la clasificación de las características y la identificación de los sistemas, se creó el catálogo de características del sistema.

2.6.4 Agile Unified Process (AUP)

2.6.4.1 Objetivo de la propuesta AUP

AUP es un enfoque para el desarrollo de software basado en el Proceso Unificado Rational. El objetivo de AUP es guiar el proceso de desarrollo de aplicaciones software de una manera simple y fácil de entender, usando técnicas ágiles y conceptos que son también válidos en RUP. AUP aplica técnicas ágiles, como el desarrollo dirigido por pruebas, modelado ágil, gestión de cambios ágil, y refactorización de base de datos para mejorar la productividad (AMBLER, 2005).

2.6.4.2 Modelo de la propuesta AUP

El ciclo de vida de Agile UP es serial en lo grande e iterativo en lo pequeño, liberando entregables incrementales en el tiempo. Serial en lo Grande: por medio de fases que son implementadas de una forma serial a lo largo de un proyecto de Agile UP. Iterativo en lo pequeño: por medio de disciplinas que son ejecutadas de forma iterativa para construir, validar y liberar software funcional. Entregas incrementales: por medio de liberaciones de versiones de software al final de cada iteración. Una versión es una parte del software que ha sido sometida a aseguramiento de la calidad, pruebas, y proceso de despliegue (AMBLER, 2005).

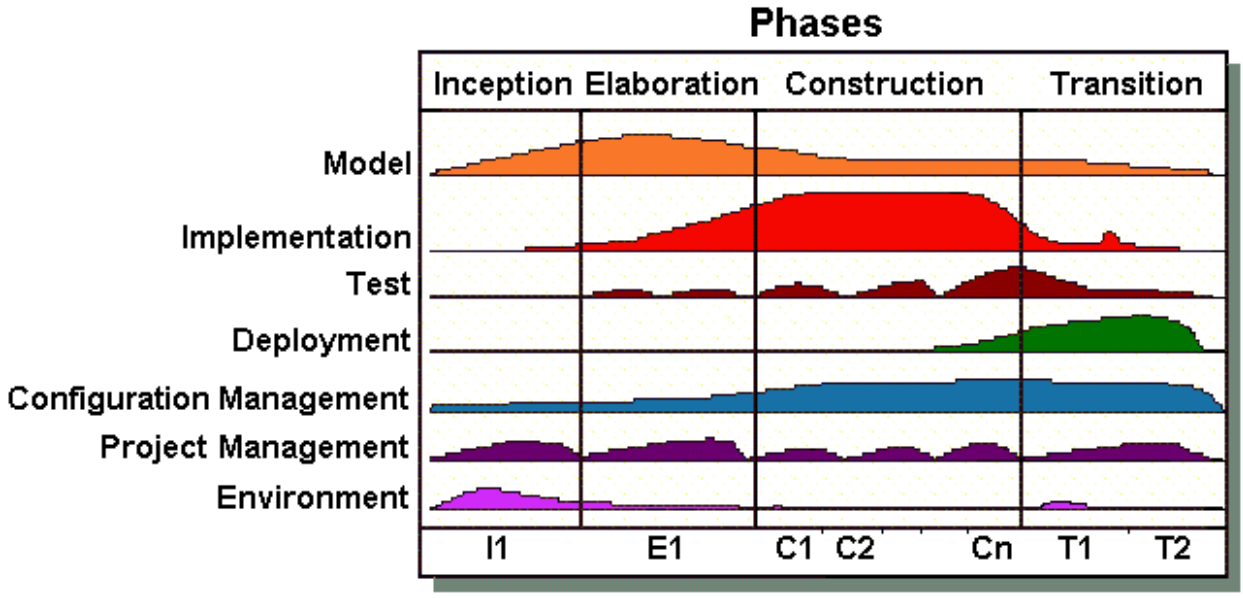
2.6.4.3 Activos de la propuesta AUP

Los activos de software en AUP son variables en cada proyecto, de acuerdo a las particularidades del proyecto, el equipo de desarrollo determina cuales son los activos relevantes.

Los activos de software en AUP son: Resumen del proyecto, modelo de requisitos, prototipo de prueba de concepto, modelo de diseño, documento resumen del sistema, matriz de trazabilidad, suite de pruebas de regresión, código fuente esquemas de datos, plan del proyecto, estrategias de pruebas, modelo de pruebas, registros de revisión, notas de la versión, scripts de instalación, documentación de soporte, documentación de usuario, material de entrenamiento, documentación de operación, entorno de configuración, artefactos del sistema, recursos del proyecto, estimado de trabajo individual, proceso de medición, herramientas de soporte y sistema integral

2.6.4.4 Proceso de la propuesta AUP

Figura 7. Proceso de la propuesta AUP



Fuente: (AMBLER, 2005)

El proceso AUP incluye cuatro fases: inicio, elaboración, construcción y transición. Una fase se realiza mediante iteraciones. Una iteración es la ejecución de un conjunto de disciplinas. Las disciplinas en AUP son: modelado, implementación, pruebas, despliegue, gestión de configuración, gestión del proyecto, y ambiente. Una iteración incluye la ejecución de varias disciplinas con diferentes niveles de esfuerzo, de acuerdo con la fase que se esté realizando (Ver Figura 7).

2.6.4.5 Orientación hacia la reutilización y el trabajo con catálogos en la propuesta AUP

AUP no incluye una disciplina específica para implementar la reutilización. Sin embargo, la flexibilidad de AUP permite integrar la práctica de la reutilización de activos de software en cada una de sus disciplinas incorporando repositorios que son consultados a lo largo de todo el ciclo de vida.

Para los agilitas, es posible tener éxito en la práctica de la reutilización, si la organización ha adoptado un proceso ágil iterativo, solo se debe tener una visión general de la empresa y no una visión particular de proyecto (AMBLER, 2003). AUP es una buena opción.

AUP puede ser extendida por Enterprise Unified Process (EUP) (AMBLER, 2003), que integra en su proceso la disciplina estrategia de reutilización, que define los estándares de reutilización en la organización.

3. CARACTERIZACION DE LOS ACTIVOS DE SOFTWARE

3.1 INTRODUCCIÓN

En este capítulo, se expone el análisis y la comparación de propuestas de desarrollo de software, presentadas en el capitulo anterior, a fin de identificar, describir y definir los activos que hacen parte del repositorio, sus características y mecanismo de representación.

Inicialmente, se identifican y describen los activos generados en cada una de las etapas de las propuestas de desarrollo. Para realizar esta tarea se adopta una definición inicial de activo de software. Posteriormente se realiza el análisis comparativo para determinar los activos de software que hacen parte del repositorio. Finalmente, se presenta el metamodelo que define Activo de software desde la perspectiva de este trabajo de investigación.

3.2 GENERALIDADES

Diversos autores han generado definiciones en torno al concepto de activo de software, cada una desde la perspectiva de sus investigaciones.

El Dr. Staindl Wolfgang de IBM, define activo de software como una colección de artefactos los cuales proveen una solución a los requerimientos o necesidades del negocio o la misión, dentro de uno o varios contextos y con instrucciones de uso (WOLFGANG, 2009).

El Dr. Thomas Pole, propone: Un activo de (ingeniería de) software es algo, en términos computacionales, que se creó en la búsqueda de la construcción de un sistema de software intensivo; en algunos casos, puede ser utilizado de nuevo para crear otros sistemas, en este caso, se trata de un activo reutilizable (POLE, 2007).

La IEEE define activo reutilizable como un elemento tal como diseños, especificaciones, código fuente, documentación, unidades de prueba y manuales de procedimientos, los cuales fueron diseñados para su uso en múltiples contextos (IEEE COMPUTER SOCIETY, 2010).

Este trabajo adopta la definición propuesta por la IEEE Computer society a fin de caracterizar los activos de software.

3.3 ACTIVOS DE SOFTWARE EN LA PROPUESTA RUP

Se debe entender que los activos identificados (Tabla 7), forman parte de grandes cuerpos documentales donde la nomenclatura de descripción es realizada a través de los diagramas UML de las especificaciones de OMG. En este orden de ideas, los activos relacionados a continuación involucran uno o más diagramas a fin de

realizar descripciones de los elementos del activo según corresponda.

Nótese que algunos activos están marcados con (*), para denotar que son activos de software que son comunes a mas de una disciplina debido a que van siendo refinados en diferentes fases del proyecto.

Tabla 7. Activos de Software en la Propuesta RUP

Disciplina	Activo de Software	Descripción
Modelado de Negocio	Modelo de Dominio	Representación de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales, modelos de objetos del dominio y modelos de objetos de análisis (AMBLER, 2005).
	Modelo de casos de uso del negocio	Representación de las funciones de negocio vistas desde la perspectiva de los actores externos. Permite situar al sistema en el contexto organizacional haciendo énfasis en los objetivos en este ámbito. (AMBLER, 2005).
	Resumen del proyecto • Objetivos del proyecto • Evaluación de la organización	Presenta los stakeholders, los objetivos del proyecto, los parámetros iniciales, alcances y delimitaciones con los cuales se pretende construir el sistema.
	Documento de Visión	Defina la vista que tienen las personas involucradas del producto que se va a desarrollar, especificada en términos de las necesidades y características claves de dichas personas. Proporciona la base contractual para los requisitos técnicos más detallados.
Requisitos	Modelo de requisitos • Modelo de procesos de negocio • Oportunidades de automatización • Modelo de interfaz de usuario • Modelo de organización • Especificación de reglas de negocio • Matriz de trazabilidad • Modelo de dominio • Glosario del proyecto • Requisitos técnicos • Pruebas de aceptación	Define el escenario donde la solución computacional se ajusta a ciertas expectativas de quienes están en el lugar donde el sistema se desplegara. Consiste en una amplia documentación donde se puede agrupar perspectivas desde la arquitectura, la estructura, el comportamiento así como aspectos funcionales y no funcionales.

Tabla 7. (Continuación)

Disciplina	Activo de Software	Descripción
Requisitos	Modelo de casos de uso	Presenta las funciones del sistema y los actores que hacen uso de ellas. Integra una descripción detallada utilizando una plantilla, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados, diagramas.
	Documento de Especificaciones adicionales	Muestra los requisitos que no han sido incluidos como parte de los casos de uso y se refieren requisitos no-funcionales globales. Dichos requisitos incluyen: requisitos legales o normas, aplicación de estándares y requisitos de calidad del producto.
Análisis y Diseño	Modelo de Diseño • Modelo de objetos • Modelo de interfaz de usuario (*) • Modelo de hilo de seguridad • Modelo de flujos de trabajo, secuencias y estados • Matriz de trazabilidad (*)	Establece la realización de los casos de uso en clases, pasando desde una representación en términos de análisis hacia una de diseño, de acuerdo al avance del proyecto. Es este activo uno de los de mayor complejidad ya que involucra diferentes vistas de diseño según actores, según funciones, según tiempos, según secuencias, según relaciones, etc. Todos los aspectos que tengan que ver con la fase de diseño esta descrito en activos simples del corpus documental de diseño.
	Modelo de Datos	Describe la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado relacional de datos (LARMAN, 2002).
	Documento de arquitectura	Describe la organización, estructura y comportamiento de los elementos del alto nivel del sistema. RUP sugiere seis vistas de la arquitectura: vista lógica, vista de proceso, vista de despliegue, vista de datos, vista de casos de uso y vista de implementación (LARMAN, 2002).
Implementación	Matriz de trazabilidad (*)	Relaciona los requerimientos con pruebas específicas a ser ejecutadas. Esta herramienta establece un acercamiento entre requisitos y casos de uso definidos.
	Suite de Pruebas de regresión (*)	Técnicas sobre la búsqueda de errores. Las pruebas de regresión buscan carencias de funcionalidad o divergencias funcionales respecto al comportamiento esperado de un sistema computacional.

Tabla 7. (Continuación)

Disciplina	Activo de Software	Descripción
Implementación	Código fuente (*)	Archivos que contienen la escritura de código en un lenguaje de programación específico. El código fuente puede seguir lineamientos de programación como uso de comentarios, identacion, nomenclatura de variables y objetos, etc.
	Sistema	Es el resultado de un proceso de desarrollo; puede ser un paquete completo, un módulo, o un objeto que brinda la solución requerida.
	Esquemas de datos (*)	Es la estructura física organizada que define la forma como la información es manejadla por el software.
Pruebas	Estrategias de pruebas (*)	Presenta los factores que determinan las pruebas del software: los probadores, el cubrimiento, la evaluación, los riesgos y las actividades.
	Modelo de Pruebas • Suite de pruebas de regresión (*) • Reporte de defectos • Matriz de trazabilidad	Establece la valoración del software a través de una serie de casos de pruebas. Siguen fielmente los principios de operatividad, observabilidad y controlabilidad para cada situación.
	Registros de revisión y notas de versión	Describen las producciones liberadas al mercado. Los registros de revisión son los instrumentos que llevan el control de las pruebas efectuadas al software. Las notas de versión describen las especificaciones técnicas y de funcionamiento por cada una de las versiones
Despliegue	Scripts de instalación	Herramientas, manuales y procesos para la instalación, configuración y afinamiento que garanticen el despliegue exitoso del software. Permiten orientar al administrador del sistema en las tareas de despliegue y puesta a punto del software.
	Sistema (*)	Es el resultado de un proceso de desarrollo; puede ser un paquete completo, un módulo, o un objeto que brinda la solución requerida.
	Documentación de soporte (*)	Describe especificaciones técnicas del sistema, pruebas prototipadas, elementos de diseño general y módulos o componentes software de terceras empresas que deban ser requeridos para garantizar el funcionamiento del software.

Tabla 7. (Continuación)

Disciplina	Activo de Software	Descripción
Despliegue	Documentación de usuario (*)	Describe la forma como se debe utilizar el software. Explica la interacción que debe tener el usuario final con el software a fin de convertirse en una guía para su uso y correcto aprovechamiento de las características.
	Material de entrenamiento (*)	Describe un compendio de demostraciones que facilitan la comprensión de las actividades desarrolladas con el software.
	Documentación de operación (*)	Describe características internas del software que ayudan a fortalecer los procesos de extensibilidad. Las operaciones internas pueden ser conocidas de acuerdo con los términos de licenciamiento.
Gestión de la configuración	Artefactos del sistema (*)	Puede ser el software final completo o cada uno de los elementos que lo integran: el código fuente, plataformas de despliegue, bibliotecas de clases, componentes, etc.
Gestión del proyecto	Plan del proyecto • Plan de iteraciones • Cronograma del proyecto • Lista de riesgos • Estimados • Presupuesto	Documento se especifican los objetivos, el alcance, y la descripción detallada del software junto con las estrategias para su consecución. Marca las pautas para servir como guía durante el desarrollo del proyecto.
Ambiente	Herramientas de Soporte	Herramientas que apoyan el funcionamiento y mantenimiento del software. Por ejemplo: programas de instalación, configuración, integración, afinación y despliegue.

3.4 ACTIVOS DE SOFTWARE EN LA PROPUESTA CBSD

Los activos de software relacionados a continuación (tabla 8), forman parte de grandes cuerpos documentales donde la nomenclatura de descripción es realizada por medio de diagramas UML de las especificaciones de OMG. Los activos que se describen a continuación involucran uno o más diagramas a fin de realizar descripciones de los elementos del activo.

Tabla 8. Activos de Software en la Propuesta CBSD (PARNAS, 1994)

Fase	Activo de Software	Descripción
Análisis de Requerimientos y Definición	Modelo de Requisitos	Define el escenario donde el software funcionará y describe las expectativas de los stakeholders: la arquitectura, la estructura, el comportamiento así como aspectos funcionales y no funcionales.
Selección de Componentes y Evaluación	Bibliotecas de Componentes • Componentes • Módulos • Paquetes	Es la infraestructura requerida para la búsqueda de componentes existentes para su reutilización. Los nuevos componentes se integran al repositorio de consumo dinámico. A nivel documental, se presentan activos simples tales como diagramas de componentes, de paquetes y módulos.
Diseño del Sistema	Modelo de Diseño • Clases e Interfaces • Patrones • Diagrama de Componentes	Establece la realización de los casos de uso en clases. Presenta la transición de una representación en términos de análisis a una de diseño. Todos los aspectos que tengan que ver con la fase de diseño están descritos en activos simples del corpus documental de diseño.
Implementación del Sistema	Modelo de Implementación • Código fuente • Diagrama de despliegue	Describe los escenarios de implementación rápida. Los activos son: código fuente, matrices de parametrización de componentes, determinación de contratos y mensajería entre componentes, diagramas de despliegue, etc.
	Esquema de datos	Describe la estructura de organización de la información y la forma como el software la utiliza incluidas las restricciones y alcance.
	Sistema Integrado y Componentes	El sistema integrado hace referencia al software completo resultado de un proceso de desarrollo basado en componentes. Los componentes se refieren a unidades modulares claramente distinguibles en el sistema integrado. Los componentes facilitan la modularidad y escalabilidad del sistema integrado.
Integración del Sistema	Modelo de Despliegue • Scripts de instalación	Describe los aspectos relacionados con integración de componentes en software y del software con el contexto. Permite orientar al administrador del sistema en las tareas de despliegue y puesta a punto del software en un contexto específico.
Verificación y Validación	Casos de prueba	Describe situaciones que el software debe afrontar para emplearlas como escenarios de pruebas. Incluye casos de prueba tanto de unidades como de integración.

Tabla 8. (Continuación)

Fase	Activo de Software	Descripción
Soporte y Mantenimiento	Documento del entorno de configuración	Describe la plataforma de hardware y software requerida para el despliegue del software. Permite también controlar las versiones del sistema integrado y de los componentes por medio de registros de revisión y notas de versión.
	Documentación de Soporte y Mantenimiento	Contempla aspectos como especificaciones técnicas del sistema, herramientas para despliegue y afinamiento, pruebas prototipadas, elementos de diseño general y módulos o componentes software de terceras empresas que deben ser requeridos para garantizar el funcionamiento del software.

3.5 ACTIVOS DE SOFTWARE EN LA PROPUESTA FODA

Por tratarse de una propuesta de análisis de dominio, la propuesta FODA incluye activos a nivel de dominio que son instanciados en la ingeniería de aplicaciones para la construcción de productos software específicos (tabla 9).

Tabla 9. Activos de Software en la Propuesta FODA

Fase	Activo de Software	Descripción
Análisis de Contexto	Modelo de contexto	Define el alcance del análisis de dominio. Se constituye de uno o más diagramas, de bloque, de estructuras de flujo de datos.
Modelado del dominio	Modelo de dominio	Describe el vocabulario de dominio, las propiedades comunes y variables de todos los sistemas del dominio y las dependencias entre estas propiedades (FEI, y otros, 2003).
	Modelo de Características	Describe las características de los objetos del sistema. Sirve como medio de comunicación ente los clientes y los desarrolladores. Ayuda a los analistas de requisitos a definir las especificaciones del sistema (FEI, y otros, 2003).
	Modelo de información	Captura y define el conocimiento del dominio y los requisitos de los datos los cuales son esenciales en la implementación de las aplicaciones (FEI, y otros, 2003).
	Modelo Funcional	Describe el fundamento sobre el cual se inicia el proceso de entender como proveer las características identificadas en el modelado de características (FEI, y otros, 2003).

Tabla 9. (Continuación)

Fase	Activo de Software	Descripción
Modelado de Arquitectura	Modelo de Arquitectura	Describe modelo de referencia de diseño de alto nivel para aplicaciones en un dominio. Este se centra en identificar procesos concurrentes y módulos comunes orientados al dominio (FEI, y otros, 2003).
	Solución Software	Es el resultado de un proceso de desarrollo basado en FODA; puede ser un paquete completo, un módulo, un objeto que brinda la solución requerida (FEI, y otros, 2003).

3.6 ACTIVOS DE SOFTWARE EN LA PROPUESTA AUP

AUP integra un conjunto amplio de activos de software, muchos de ellos heredados de la propuesta RUP. La flexibilidad de AUP, permite a los desarrolladores integrar a sus proyectos aquellos activos que son relevantes en el contexto particular del proyecto. A continuación se relacionan los activos más comunes (tabla 10). Nótese que están marcados con (*), los activos que se han realizado en la disciplina indicada o en otras disciplinas y son retroalimentados en la disciplina indicada.

Tabla 10. Activos de Software en la Propuesta AUP

Disciplina	Activo de Software	Descripción
Modelado	Resumen del proyecto • Visión del proyecto • Objetivos del proyecto • Evaluación de la organización	Sigue una plantilla predefinida por la organización que incluye: la visión del proyecto desde la perspectiva de la organización cliente, los objetivos del proyecto y la evaluación de la organización que genera el diagnostico organizacional. Establece los parámetros iniciales para la construcción del sistema, alcances y delimitaciones.
	Modelo de requisitos • Modelo de casos de uso • Modelo de procesos de negocio • Oportunidades de automatización • Modelo de interfaz de usuario • Modelo de organización • Especificación de reglas de negocio • Modelo de dominio • Glosario del proyecto	Define el escenario de operación del software, basado en las expectativas de los stakeholders. Puede ser tan amplio como la complejidad y tamaños del sistema. Las perspectivas presentadas integran la arquitectura, la estructura, el comportamiento así como aspectos funcionales y no funcionales del sistema.

Tabla 10. (Continuación)

Disciplina	Activo de Software	Descripción
Modelado	Modelo de Diseño • Modelo de objetos • Modelo físico de datos • Modelo de interfaz de usuario (*) • Modelo de hilo de seguridad • Modelo de despliegue • Matriz de trazabilidad (*) • Documento de resumen del sistema	Define la realización de los casos de uso identificados en el modelo de requisitos. Permite la transición del análisis al diseño en el proceso de construcción. Esta a su vez, integrado por varios modelos que describen diversas vistas del sistema.
Implementación	Modelo de diseño (*)	Define la realización de los casos de uso identificados en el modelo de requisitos. Permite la transición del análisis al diseño en el proceso de construcción. Esta a su vez, integrado por varios modelos que describen diversas vistas del sistema.
	Matriz de trazabilidad (*)	Permite relacionar los artefactos de las diferentes fases del proceso de desarrollo y realizar el seguimiento a la evolución de los casos de uso desde el modelado hasta el despliegue.
	Suite de Pruebas de regresión (*)	Se refiere a la documentación de las técnicas empleadas para la realización de pruebas que permitan identificar errores en la producción del software. Las pruebas de regresión buscan deficiencias de funcionalidad e incoherencias con el comportamiento esperado del sistema.
	Código fuente (*)	Hace referencia a los programas que encapsulan el código escrito en un lenguaje de programación. En las organizaciones de software, normalmente, se definen y documentan estándares de programación como uso de comentarios, identacion, nomenclatura de variables y objetos.
	Sistema	Puede ser un paquete completo, un módulo o un objeto que brinda una solución requerida. Se refiere al resultado funcional de una iteración o proceso de desarrollo.
	Esquemas de datos (*)	Constituye una estructura organizada que describe como el sistema maneja la información. La información puede ser manejada en bases de datos o en archivos.

Tabla 10. (Continuación)

Disciplina	Activo de Software	Descripción
Pruebas	Plan del proyecto (*)	Especifica los objetivos, el alcance y la descripción detallada del sistema. Define las directrices que guían el desarrollo y hace parte de la documentación técnica y empresarial del proyecto.
	Estrategias de pruebas (*)	Definen las actividades y momentos relacionados con la forma como se realizaran las pruebas del sistema. Las estrategias de pruebas deben ser bien documentadas indicando: los probadores, el cubrimiento, la evaluación, los riesgos y las actividades.
	Modelo de Pruebas • Suite de pruebas de regresión • Reporte de defectos • Matriz de trazabilidad	Se refiere al marco general de trabajo para el desarrollo de las pruebas del sistema. Es adaptable a las particularidades de los proyectos. Siguen los principios de operatividad, observabilidad, y controlabilidad para cada caso de prueba.
	Registros de revisión y notas de versión	El control de versiones ayuda a garantizar la sostenibilidad del producto en el mercado a través del tiempo. Permite a la organización de software controlar las configuraciones para determinar la versión de producto más adecuada para un conjunto de requisitos. Los registros de revisión son los instrumentos que llevan el control de las pruebas que permiten identificar oportunidades de mejoramiento para versiones futuras. Las notas de versión permiten describir las especificaciones técnicas y de funcionamiento por cada una de las versiones producidas.
Despliegue	Plan de proyecto (*)	Especifica los objetivos, el alcance y la descripción detallada del sistema. Define las directrices que guían el desarrollo y hace parte de la documentación técnica y empresarial del proyecto.
	Scripts de instalación	Permite orientar al administrador del sistema y usuario finales en las tareas instalación, despliegue y puesta a punto del sistema software.
	Sistema (*)	Puede ser un paquete completo, un módulo o un objeto que brinda una solución requerida. Se refiere al resultado funcional de una iteración o de un proceso.

Tabla 10. (Continuación)

Disciplina	Activo de Software	Descripción
Despliegue	Documentación de soporte (*)	Contempla todos aquellos aspectos que deban ser requeridos para garantizar la adecuada operación y el correcto mantenimiento de la solución software.
	Documentación de usuario (*)	Describe de forma explícita como se debe utilizar el sistema software a fin de convertirse en una guía para el correcto aprovechamiento de las funcionalidades que provee.
	Material de entrenamiento (*)	Permite al usuario realizar ejercicios de operación del software. Puede integrar demostraciones que facilitan la comprensión de las funcionalidades que provee el sistema. El material de entrenamiento puede formar parte de los activos de documentación del usuario.
	Documentación de operación (*)	Posee características adicionales a la documentación de usuario que ayudan a fortalecer los procesos de extensibilidad del software. Las operaciones internas del software pueden ser conocidas si así lo establecen los términos de licenciamiento.
Gestión de la configuración	Artefactos del sistema (*)	El software como producto final, se constituye en un activo que puede estar compuesto por múltiples artefactos, por ejemplo: el código fuente, el código objeto/binario/ejecutable, las bibliotecas de enlace dinámico del sistema operativo, las plataformas de despliegue de frameworks y/o máquinas virtuales, las bibliotecas de clases, los compiladores e intérpretes, los componentes para navegadores y los componentes para reproductores.
Gestión del proyecto	Plan del proyecto • Hardware • Software • Financiamiento • Instalaciones • Plan de iteraciones • Cronograma del proyecto • Lista de riesgos • Estimados • Presupuesto	Especifica los objetivos, el alcance y la descripción detallada del sistema. Define las directrices que guían el desarrollo y hace parte de la documentación técnica y empresarial del proyecto.

Tabla 10. (Continuación)

Disciplina	Activo de Software	Descripción
Ambiente	Herramientas de Soporte	Son herramientas que facilitan tareas de instalación, configuración, integración, afinación y despliegue del software. En algunos escenarios, las herramientas de soporte son proveídas por terceras empresas.

3.7 ANÁLISIS COMPARATIVO DE ACTIVOS EN PROPUESTAS DE DESARROLLO DE SOFTWARE

El propósito de este análisis comparativo es identificar los activos de software que van a hacer parte del repositorio y sus características, para posteriormente proponer un metamodelo que los define y caracteriza.

Inicialmente, se identifican los activos de software que cumplen objetivos similares en las diferentes aproximaciones. Esta tarea se realiza con base en los activos de software identificados y descritos en cada una de las propuestas de desarrollo. Nótese que en algunos casos (marcados con (*)), un solo activo cumple objetivos que en otras aproximaciones son asignados a varios activos de software (Tabla 11).

Tabla 11. Relación de Activos de Software que cumplen objetivos similares

RUP	CBSD	FODA	AUP
✓ Modelo de Dominio		✓ Modelo de contexto ✓ Modelo de Dominio (*)	✓ Resumen del proyecto (*)
✓ Modelo de casos de uso del negocio		✓ Modelo de Características (*) ✓ Modelo Funcional (*)	✓ Resumen del proyecto (*)
✓ Resumen del proyecto			✓ Resumen del proyecto (*)
✓ Documento de Visión		✓ Modelo de Dominio (*)	✓ Resumen del proyecto (*)
✓ Modelo de requisitos	✓ Modelo de Requisitos (*)	✓ Diccionario de dominio	✓ Modelo de requisitos (*)
✓ Modelo de casos de uso		✓ Modelo Funcional (*)	✓ Modelo de requisitos (*)
✓ Especificaciones adicionales	✓ Modelo de Requisitos (*)	✓ Modelo de Características (*)	✓ Modelo de requisitos (*)

Tabla 11. (Continuación)

RUP	CBSD	FODA	AUP
✓ Modelo de Diseño	✓ Modelo de Diseño	✓ Modelo de Dominio (*)	✓ Modelo de Diseño (*)
✓ Modelo de Datos	✓ Modelo de Diseño	✓ Modelo de Información	✓ Modelo de Diseño (*)
✓ Documento de arquitectura	✓ Modelo de Diseño centrado en arquitectura	✓ Documento de Arquitectura	✓ Modelo de Diseño (*)
✓ Matriz de trazabilidad			✓ Matriz de trazabilidad
✓ Suite de Pruebas de regresión			✓ Suite de Pruebas de regresión
✓ Código fuente	✓ Código fuente		✓ Código fuente
✓ Sistema	✓ Modelo de Implementación ✓ Sistema Integrado y Componentes (solución software)	✓ Solución Software	✓ Sistema
✓ Esquemas de datos	✓ Esquema de datos		✓ Esquemas de datos
✓ Estrategias de pruebas	✓ Modelo de Evaluación de Componentes		✓ Estrategias de pruebas
✓ Modelo de Pruebas	✓ Casos de prueba		✓ Modelo de Pruebas
✓ Registros de revisión y notas de versión			✓ Registros de revisión y notas de versión
✓ Scripts de instalación	✓ Modelo de Despliegue (*)		✓ Scripts de instalación
✓ Documentación de soporte	✓ Modelo de Despliegue (*)		✓ Documentación de soporte
✓ Documentación de usuario	✓ Modelo de Despliegue (*)		✓ Documentación de usuario
✓ Material de entrenamiento	✓ Modelo de Despliegue (*)		✓ Material de entrenamiento
✓ Documentación de operación	✓ Modelo de Despliegue (*)		✓ Documentación de operación
✓ Artefactos del sistema	✓ Componente		✓ Artefactos del sistema

Tabla 11. (Continuación)

RUP	CBSD	FODA	AUP
✓ Plan del proyecto			✓ Plan del proyecto
✓ Herramientas de Soporte	✓ Entorno de configuración ✓ Soporte y Mantenimiento		✓ Herramientas de Soporte

3.7.1 Identificación de los activos de software que formarán parte del repositorio

Para facilitar a las organizaciones la tarea de identificar y catalogar los activos de software potencialmente reutilizables, se propone un conjunto de activos de software que tienen un propósito particular y que representan información relevante del producto software en diferentes fases y desde diferentes perspectivas. La identificación de activos se basa en el análisis comparativo de activos en las propuestas de desarrollo y en la jerarquía de procesos, actividades y tareas propuestas en el estándar 1517 desarrollado por la IEEE Computer Society (IEEE COMPUTER SOCIETY, 2010). (Ver tabla 12).

Nótese que a cada activo identificado se le determina la jerarquía de la norma IEEE-1517 a la que pertenece, se le asigna el nombre más representativo o más utilizado en las diferentes aproximaciones e indica: el propósito general del activo y una sigla del tipo de diagrama o mecanismo que se utiliza en cada aproximación para representar el activo. Se puede observar además que buena parte de la notación sugeridas para los diferentes modelos está basada en el lenguaje UML por ser la notación mas generalizada en este momento (Las siglas empleadas se relacionan en la tabla 13).

Tabla 12. Propuesta de activos de software que hacen parte del catálogo

Jerarquía norma IEEE-1517			Clase de Activo de Software	Propósitos del Activo	RUP	CBSD	FODA	AUP
Proceso	Actividad	Tarea						
Procesos de Acuerdo	Adquisición	Aceptación de Adquiriente	Documento de visión	✓ Definir la vista que tienen los stakeholders del producto que se va a desarrollar. ✓ Proporcionar la base contractual para los requisitos técnicos más detallados.	PL		PL	PL
Procesos del proyecto	Planeación del proyecto	Planeación del proyecto	Plan de proyecto	✓ Especificar los objetivos, el alcance y la descripción detallada del sistema. ✓ Definir las directrices que guían el desarrollo del proyecto.	PL	PL		PL

Tabla 12. (Continuación)

Jerarquía norma IEEE-1517			Clase de Activo de Software	Propósitos del Activo	RUP	CBSD	FODA	AUP
Proceso	Actividad	Tarea						
Procesos técnicos del sistema	Análisis de Requerimientos del sistema	Especificación de Requerimientos	Modelo del Dominio	✓ Describir el dominio de interés por medio de la representación de las clases conceptuales, las asociaciones y los atributos. ✓ Facilitar la comunicación ente los clientes y los desarrolladores. ✓ Identificar los aspectos comunes y variantes del dominio para el desarrollo de activos de software reutilizables.	PL DC.		PL DB DET DFD DC DA.	PL DC
Procesos técnicos del sistema	Análisis de Requerimientos del sistema	Especificación de Requerimientos	Modelo de Casos de Uso del Negocio	✓ Describir las funciones, procesos y el flujo de la información de negocio vistas desde la perspectiva de los actores externos. ✓ Situar al sistema en el contexto organizacional.	DCU PL	DCU	DCR DA DE	DCU PL

Tabla 12. (Continuación)

Jerarquía norma IEEE-1517			Clase de Activo de Software	Propósitos del Activo	RUP	CBSD	FODA	AUP
Proceso	Actividad	Tarea						
Procesos de implementación de software	Análisis de requerimientos de software	Análisis de requerimientos de software	Modelo de casos de Uso del sistema	✓ Presentar las funciones del sistema y los actores que hacen uso de ellas.	DCU	DCU		DCU
Procesos de Implementación de Software	Análisis de Requerimientos de Software	Análisis de Requerimientos de Software	Modelo de Requisitos	✓ Definir el escenario donde funcionará el software. ✓ Agrupar las perspectivas desde la arquitectura, la estructura, el comportamiento, los aspectos funcionales y no funcionales. ✓ Definir los términos relevantes y sus definiciones en el contexto del dominio del problema. ✓ Relacionar los requerimientos con pruebas específicas a ser ejecutadas	PL	PL	PL	PL

Tabla 12. (Continuación)

Jerarquía norma IEEE-1517			Clase de Activo de Software	Propósitos del Activo	RUP	CBSD	FODA	AUP
Proceso	Actividad	Tarea						
Procesos de Implementación de Software	Diseño Detallado de Software	Diseño Detallado de Software	Modelo de diseño	✓ Definir la realización de los casos de uso en clases, pasando desde una representación en términos de análisis hacia una de diseño.	DS DC DE	DS DC DE DCP		DS DC DE
Procesos de Implementación de Software	Diseño Detallado de Software	Diseño Detallado de Software	Modelo de datos	✓ Describir la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado de datos. ✓ Capturar y definir el conocimiento del dominio y los requisitos de los datos. ✓ Describir en forma precisa como la información es manejada.	DER DC	DER DC	DER DC	DER DC

Tabla 12. (Continuación)

Jerarquía norma IEEE-1517			Clase de Activo de Software	Propósitos del Activo	RUP	CBSD	FODA	AUP
Proceso	Actividad	Tarea						
Procesos de Implementación de Software	Diseño Arquitectural de Software	Diseño Arquitectural de Software	Documento de Arquitectura	✓ Describir la organización, estructura y comportamiento de los elementos de alto nivel del sistema. El comportamiento del sistema, se describe especialmente en función de responsabilidades de gran escala de los subsistemas y sus colaboraciones. ✓ Identificar procesos concurrentes y módulos comunes orientados al dominio.	DP DS	DP DS DCP	DP DS	DP DS

Tabla 12. (Continuación)

Jerarquía norma IEEE-1517			Clase de Activo de Software	Propósitos del Activo	RUP	CBSD	FODA	AUP
Proceso	Actividad	Tarea						
Procesos de Implementación de Software	Construcción de Software	Construcción de Software	Sistema	✓ Proveer una funcionalidad requerida genérica o concreta, puede ser un componente, un paquete completo, un módulo, un diagrama o un objeto. La solución software integra varios activos software que lo definen integralmente.	DP DCP	DP DCP	PL	DP DCP
Procesos de Implementación de Software	Pruebas de Calificación de Software	Pruebas de Calificación de Software	Modelo de Pruebas	✓ Valorar el software a través de una serie de casos de pruebas. ✓ Definir las actividades y momentos relacionados con la forma como se realizaran las pruebas del sistema. ✓ Identificar carencias de funcionalidad o divergencias funcionales respecto al comportamiento esperado de un sistema computacional.	PL	PL		PL

Tabla 12. (Continuación)

Jerarquía norma IEEE-1517			Clase de Activo de Software	Propósitos del Activo	RUP	CBSD	FODA	AUP
Proceso	Actividad	Tarea						
Procesos de Implementación de Software	Integración de Software	Integración de Software	Modelo de despliegue	✓ Orientar al administrador del sistema y usuario finales en las tareas instalación, despliegue, puesta a punto, uso y mantenimiento del sistema software.	DD	DD		DD

Tabla 13. Siglas de tipos de diagramas y mecanismos de representación

Sigla: Descripción
DC: Diagrama de Clases
DB: Diagrama de Bloques
DE: Diagrama de Estructuras
DA: Diagrama de Actividades
DS: Diagrama de Secuencia
DP: Diagrama de Paquetes
DED: Diagrama de Estructura de Dominio
DFD: Diagrama de Flujo de Datos
DCT: Diagrama de Contexto
DCU: Diagrama de Casos de Uso
DCR: Diagrama de Características
DET: Diagrama de Estados
DER: Diagrama Entidad Relación
DCP: Diagrama de componentes
DD: Diagrama de despliegue
PL: Plantilla

3.8 METAMODELO DE ACTIVOS DE SOFTWARE DESDE LA PERSPECTIVA DE LA INVESTIGACION

El metamodelo que define y caracteriza el activo de software, es una extensión del metamodelo de activo de conocimiento propuesto en (ANAYA, y otros, 2008), el cual fue adaptado para contemplar los resultados obtenidos de la identificación y descripción de activos de los activos que hacen parte del repositorio. El metamodelo que define y caracteriza el activo de software se muestra en la figura 8 y se describe a continuación:

Un activo de software es un artefacto o una colección de artefactos que se crea y retroalimenta a lo largo del proceso software. Cuando se utiliza para crear nuevos sistemas, se trata de un activo reutilizable.

El proceso software se compone de un conjunto de actividades, cada una de las cuales a su vez, puede integrar otras actividades configurando una jerarquía de actividades de proceso. El estándar 1517 de la IEEE, por ejemplo, define la jerarquía: tareas integradas en actividades, actividades que hacen parte de procesos y procesos que hacen parte del proceso software de la organización. Un activo de software se inscribe en la jerarquía del proceso software.

El activo de software es una especialización de activo de conocimiento. Por lo tanto, hereda sus características, al igual que el activo de conocimiento, el activo de software es generado y usado por unidades de trabajo, requiere recursos (infraestructura, personas), depende de un contexto (dominio de una aplicación particular, cliente, tecnología de soporte), es propiedad de una organización que determina las condiciones de uso por parte de terceros, es producido, validado y actualizado por personas y soportado o compuesto por otros activos.

Los activos de software se clasifican en tipos y clases. Hay tres tipos de activos de software: Los activos de ingeniería, los activos de negocio y los activos de proyecto. Entre los activos de ingeniería y de negocio se establece una relación de trazabilidad: los activos de ingeniería implementan los activos de negocio.

Los activos de negocio son activos de software que se encuentran en el espacio del problema. Estos activos representan el conocimiento del negocio de la organización cliente. Este conocimiento es necesario para la construcción del software que soporta los procesos que definen la organización y es representado por medio de modelos, por ejemplo: modelo de negocio, modelo de procesos de negocio y modelo de arquitectura empresarial.

Los activos de ingeniería son activos de software que se encuentran en el espacio de la solución. Estos activos son el resultado de la aplicación de tareas de ingeniería que transforman artefactos del espacio del problema en artefactos reutilizables del espacio de la solución.

Los activos de proyecto son el resultado de la aplicación de los activos de ingeniería en la solución de un problema puntual dentro de contexto concreto o de un proyecto especifico.

Las clases de activos de software en este trabajo de investigación, se identificaron a partir del análisis de propuestas de desarrollo de software (tabla 12). Pueden ser clases de activos de software: modelos de dominio, modelos de requisitos, documentos de arquitectura, etc.

Un activo de software puede integrar activos Rational que definen y documentan las experiencias, guías, decisiones de diseño, lecciones aprendidas, etc. que permiten aprender de acciones y decisiones anteriores.

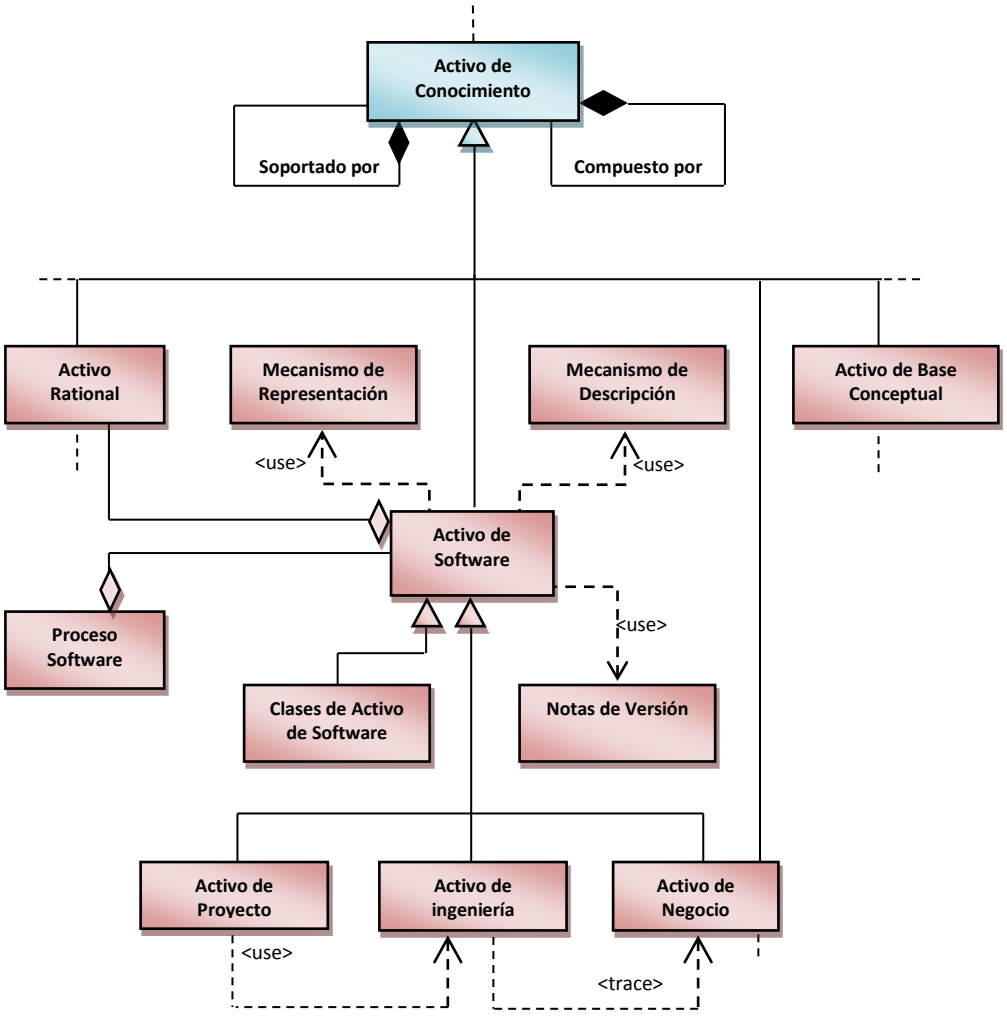
Los activos de base conceptual en este trabajo de investigación, son complementarios y permiten gestionar el marco teórico conceptual del repositorio. Estos activos son empleados para publicar aspectos conceptuales que soportan el repositorio y el proceso software y además, son relevantes para la organización.

Los activos de software son representados utilizando mecanismos de representación como por ejemplo: diagramas, modelos, documentos, plantillas y código fuente.

Los activos de software son descritos por medio de mecanismos de representación, para tal fin, este trabajo de investigación propone el uso sistemas de metadatos.

Un activo de software puede ser ofertado en varias versiones, las características específicas de una versión y las diferencias con otras versiones se documentan por medio de notas de versión.

Figura 8. Metamodelo de Activo de Software extensión de (ANAYA, y otros, 2008)



4. *PEGASO*: UNA PROPUESTA PARA LA GESTION DE ACTIVOS DE SOFTWARE

4.1 INTRODUCCIÓN

En este capítulo se describe *PEGASO* como propuesta para la gestión de activos de software y cada uno de los elementos que la componen.

Inicialmente, se describe la arquitectura propuesta para el sistema de gestión de activos de software, la arquitectura se fundamenta en el marco conceptual y la caracterización de los activos de software. Más adelante, se presenta la solución tecnológica de apoyo a la gestión de activos, seleccionada por medio del estudio comparativo de herramientas para la gestión de repositorios digitales. Finalmente, en este capítulo, se presenta DC-Soft, el sistema de metadatos para la descripción de los activos de software.

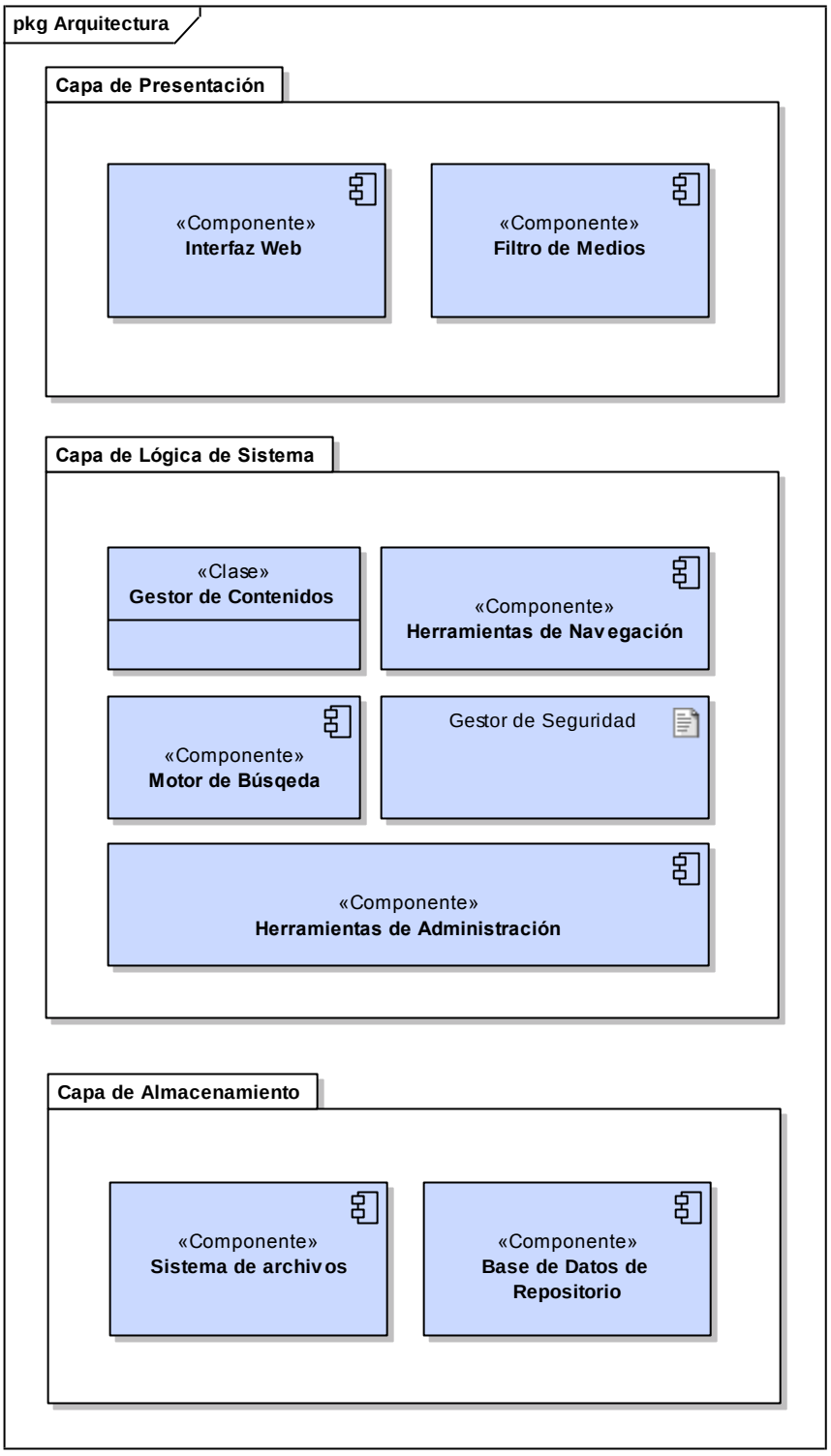
4.2 ARQUITECTURA DE UN SISTEMA DE GESTION DE ACTIVOS DE SOFTWARE

La arquitectura del sistema de gestión de activos de software es una adaptación de la arquitectura de herramientas para la gestión de repositorios digitales (FEDORA, 2011), la cual ha sido orientada al propósito particular de gestionar activos de software.

Un sistema de gestión de activos de software se compone en su arquitectura, de tres capas: *la capa de almacenamiento*, *la capa de lógica de sistema* y *la capa de presentación* (ver figura 9).

La capa de almacenamiento se compone de elementos que en conjunto permiten salvaguardar y hacer persistente la información completa de los activos de software y los activos de base conceptual*, la capa de lógica de sistema* integra elementos que permiten la gestión del repositorio*, la capa de presentación* contiene componentes que administran la comunicación con los usuarios del sistema, permiten presentar los activos de software, los activos de base conceptual y toda la información y herramientas relacionadas.

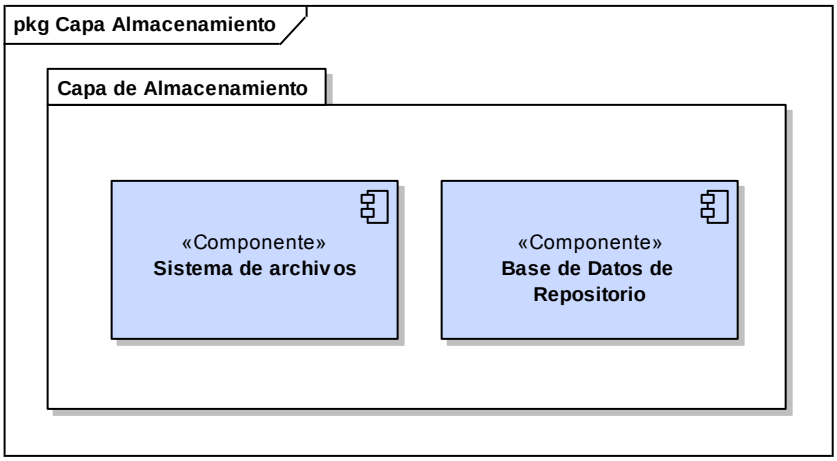
Figura 9. Arquitectura de un sistema de gestión de activos de software



4.2.1 Capa de almacenamiento

El paquete capa de almacenamiento, contiene dos tipos de almacenamiento que se describen gráficamente en la figura 10:

Figura 10. Capa de almacenamiento del sistema de gestión de activos



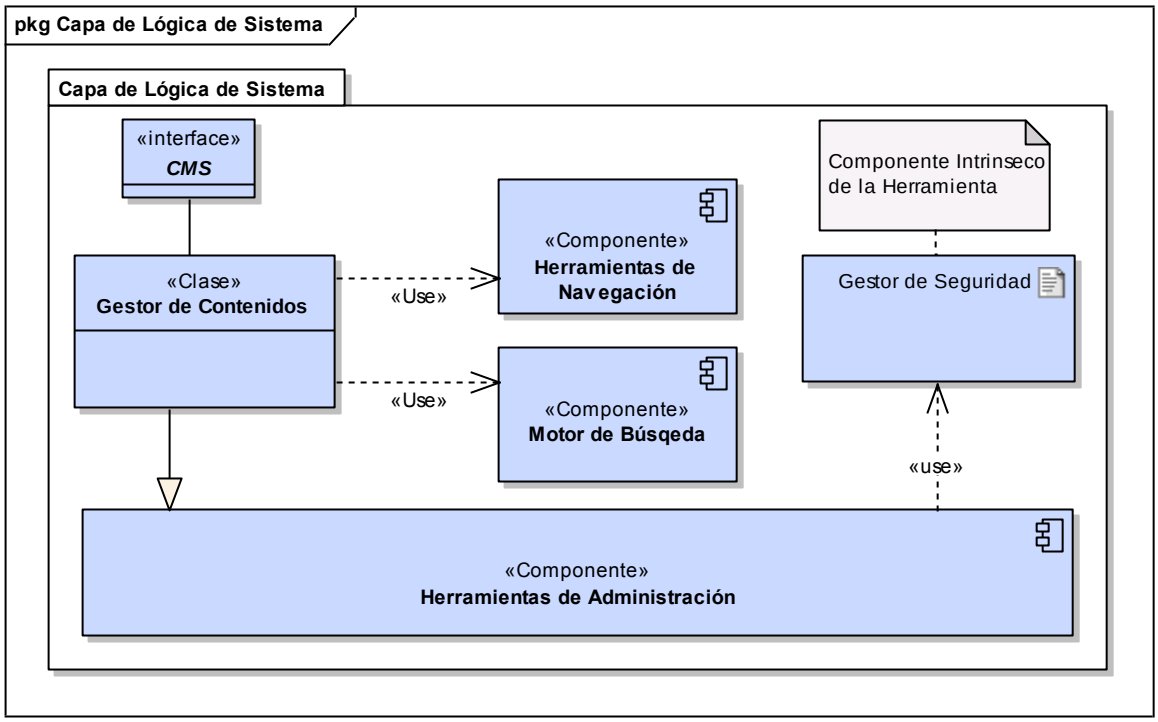
- *El sistema de archivos de repositorio*, es un componente que almacena los archivos adjuntos (mínimo 1 archivo) que representan físicamente el activo de software o el activo de base conceptual. Estos archivos pueden ser desde un documento de procesador de texto, una imagen, un diagrama, código fuente, ensamblados binarios o código objeto, código ejecutable, producción multimedia, entre otros.

- *La base de datos de repositorio,* es un componente que almacena los datos textuales que describen los activos de software mediante un conjunto de metadatos.

4.2.2 Capa de lógica de sistema

El paquete capa de lógica de sistema (figura 11), está compuesto por los siguientes elementos:

Figura 11. Capa de lógica de sistema de gestión de activos



- *El gestor de contenidos de repositorio*, es una clase que administra los archivos que contienen físicamente los activos de software, los activos de base conceptual y los metadatos que los describen. Esta clase se realiza a través del componente *administración,* implementa las características fundamentales de un CMS (Content Management System) y usa dos componentes principales: el *motor de búsqueda* y las *herramientas de navegación*.

- *Las herramientas de navegación de repositorio*, es un componente que permite recorrer el repositorio por la estructura jerárquica en la que están organizados los activos de software y los activos de base conceptual.

- *El motor de búsqueda de repositorio*, es un componente que utiliza como criterios el conjunto de metadatos orientado a la descripción de los activos de software y los activos de base conceptual. La búsqueda puede ser básica o avanzada dependiendo de los metadatos utilizados.

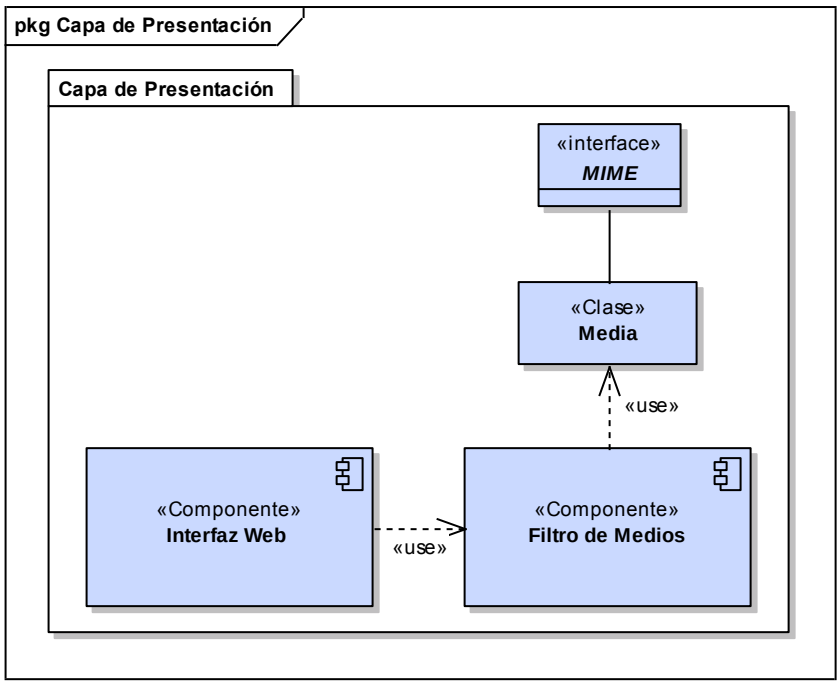
- *El gestor de seguridad de repositorio,* administra la seguridad del repositorio y es un componente intrínseco de la herramienta de gestión de repositorios digitales.

- *Las herramientas de administración de repositorio* forman un componente que permite controlar el uso del repositorio digital.

4.2.3 Capa de presentación

El paquete capa de presentación (figura 12), contiene los siguientes elementos:

Figura 12. Capa de presentación del sistema de gestión de activos



- *La interfaz web de repositorio,* es un componente que sirve como medio de interacción del usuario con el repositorio digital para las operaciones de navegación, alimentación, búsqueda y recuperación de los activos de software y los activos de base conceptual que hacen parte del repositorio.

- *El filtro de medios de repositorio,* es un componente que permite identificar el formato de los archivos que contienen físicamente los activos y las aplicaciones requeridas para recuperarlos independiente de la plataforma que se esté utilizando. Con el fin de establecer esta asociación (archivo - aplicación), se utiliza una clase de medios (MEDIA) que implementa la interface genérica MIME. Dicha interfaz define los servicios para asociar a cada archivo la aplicación respectiva con la cual puede ser manipulado.

4.3 DSPACE: LA SOLUCIÓN TECNOLÓGICA DE APOYO A LA GESTIÓN DE LOS ACTIVOS DE SOFTWARE EN *PEGASO*

Los repositorios digitales permiten a las organizaciones gestionar de mejor manera el conocimiento requerido y generado por sus procesos. El auge de la gestión del conocimiento en las organizaciones ha generado el desarrollo de diversas herramientas para la gestión de repositorios digitales. Estas herramientas emplean diversas tecnologías y técnicas de gestión de conocimiento.

4.3.1 Estudio de herramientas para la gestión de repositorios digitales

A fin de identificar la herramienta tecnológica más apropiada para apoyar la gestión de los activos de software, se realizó un estudio comparativo de tres de las herramientas de gestión de repositorios digitales más reconocidas por la comunidad usuaria: DSpace (FEDORA, 2011) E-PRINTS (UNIVERSITY OF SOUTHAMPTON, 2011) y FEDORA COMMONS (FEDORA, 2005).

A continuación se exponen los criterios de análisis, el resumen del estudio y se presenta la herramienta seleccionada. El estudio completo se presenta en el anexo A.

4.3.1.1 Criterios de análisis

- **Descripción General:** Es un resumen de la presentación de la herramienta. Presenta aspectos relacionados con la empresa propietaria, tipo de software, sus desarrolladores y una breve reseña histórica. Este criterio provee al lector una idea general de la herramienta objeto de análisis.

- **Requisitos de software:** Describe los requisitos de software mínimos necesarios para el correcto funcionamiento de la herramienta. Este criterio permite determinar el nivel de exigencia de la herramienta en aspectos de potencia de hardware y software de plataforma.

- **Facilidad de Personalización:** Describe las facilidades que provee la herramienta para personalizar y adaptar las funcionalidades a los requisitos particulares de un repositorio digital específico accediendo al código, utilizando plantillas o herramientas de parametrización. Este criterio permite determinar la flexibilidad de la herramienta para adaptarla a las condiciones específicas de un repositorio digital.

- **Documentación disponible:** Hace referencia a las fuentes de información existentes y de fácil acceso para soportar la configuración, adaptación, uso y mantenimiento del repositorio digital. Este criterio permite determinar el nivel de respaldo que tiene la herramienta cuando el usuario requiere de ayuda para la implementación de cualquier tarea o funcionalidad.

- **Comunicación con el Usuario:** Describe las funcionalidades que provee la herramienta para adaptar la interfaz de usuario tanto de entrada como de salida a las características particulares del tipo de recurso digital. Este criterio permite establecer el nivel de flexibilidad de adaptación de las interfaces.

- **Mecanismos de Descripción de recursos digitales:** Describe el mecanismo empleado por la herramienta para la descripción de los recursos digitales del repositorio. Este criterio permite establecer la flexibilidad de la herramienta para configurar los metadatos a las características de un recurso digital específico.

- **Estadísticas:** Describe las funcionalidades que provee la herramienta para proporcionar información que permita el análisis estadístico sobre diferentes aspectos de los recursos digitales del repositorio.

- **Gestión de Usuarios:** Describe las funcionalidades que provee la herramienta para la gestión de usuarios, el mecanismo de control y la asignación de permisos y niveles de acceso. Este criterio permite establecer la seguridad con que la herramienta permite gestionar los recursos digitales del repositorio.

- **Gestión de los recursos digitales (funcionalidades básicas):** Describe los mecanismos para el desarrollo de las funcionalidades básicas que operan sobre los recursos del repositorio digital tales como: adición, eliminación, modificación.

- **Mecanismos de búsqueda:** Describe el mecanismo empleado para ubicar un recurso digital específico en el repositorio a partir de criterios de búsqueda pre-establecidos. Este criterio permite establecer la flexibilidad de la herramienta para determinar criterios de búsqueda acordes con las características y mecanismo de descripción de un tipo de recurso digital.

- **Mecanismos de Recuperación:** Describe la forma como la herramienta muestra y deja a disposición del usuario un recurso digital que previamente ha sido buscado. Este criterio permite establecer la flexibilidad de la herramienta para adaptar el mecanismo de visualización del recurso digital de acuerdo con sus características particulares y el tipo de archivo que lo contiene.

- **Gestión de experiencias:** Describe las funcionalidades que provee la herramienta para gestionar las experiencias de uso de los recursos digitales del repositorio. Este criterio permite determinar la eficiencia de la herramienta para gestionar las impresiones de los usuarios acerca del recurso digital, información que puede ser consultada por otros usuarios.

- **Mecanismos de almacenamiento:** Describe el proceso y la estructura de datos empleada por la herramienta para almacenar los recursos digitales del repositorio. Este criterio permite determinar la flexibilidad de la herramienta para adaptar el repositorio a las características particulares del recurso digital gestionado y el tipo de repositorio que se pretende construir.

4.3.1.2 Resumen del estudio de herramientas para la gestión de repositorios digitales

A continuación se muestra un resumen del estudio comparativo de herramientas para la gestión de repositorios digitales (tabla 14), se presentan los criterios de comparación más relevantes y las calificaciones asociadas. Las calificaciones fueron asignadas de acuerdo con la experiencia de estudio y prueba de las herramientas comparadas.

Tabla 14. Resumen de estudio de Herramientas

Criterio	Herramientas		
	DSpace	**E-Prints**	**Fedora Commons**
Facilidad de personalización	Provee potentes herramientas para un nivel de programación básico y avanzado	Provee herramientas para un nivel avanzado de programación	Provee herramientas para un nivel avanzado de programación
Calificación	9,0	9,0	8,0
Comunica-ción con el Usuario	Provee variedad de herramientas de configuración de interfaces para nivel básico y avanzado de programación	Provee herramientas de configuración de interfaces para nivel básico y avanzado de programación	Provee herramientas de configuración de interfaces para nivel básico y avanzado de programación
Calificación	9,0	8,0	8,0
Documentación	Provee varias opciones de fácil acceso para aprendizaje, configuración y uso	Provee documentos de poco detalle. Existen opciones que tienen costo para el usuario	Provee documentos con detalle insuficiente para aprendizaje y configuración
Calificación	9,0	8,0	8,0
Mecanismos de búsqueda	Provee potentes herramientas para configuración de búsqueda básica y avanzada para un nivel básico de programación	Provee herramientas para configuración de búsqueda básica y avanzada para un nivel básico y avanzado de programación	Provee herramientas para configuración de búsqueda básica y avanzada para un nivel avanzado de programación
Calificación	9,0	8,5	8,0

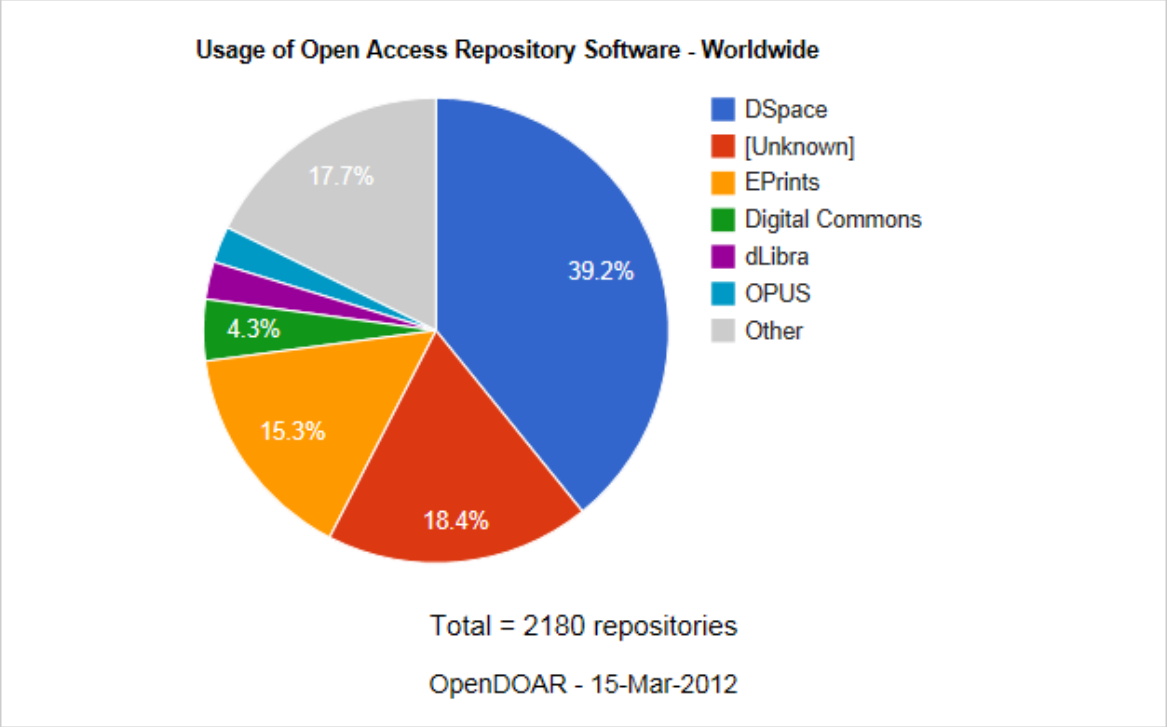
Tabla 14. (Continuación)

Criterio	Herramientas		
	DSpace	E-Prints	Fedora Commons
Mecanismos de almacenamiento	Provee un gestor robusto y sofisticado para la persistencia de recursos de forma ilimitada	Provee sofisticados servicios de gestión de persistencia de recursos digitales	Provee un sistema básico de almacenamiento, con opciones de versión de recursos
Calificación	9,0	9,0	8,5
Promedio por Herramienta	**9,0**	**8,5**	**8,1**

4.3.1.3 DSpace: la herramienta seleccionada

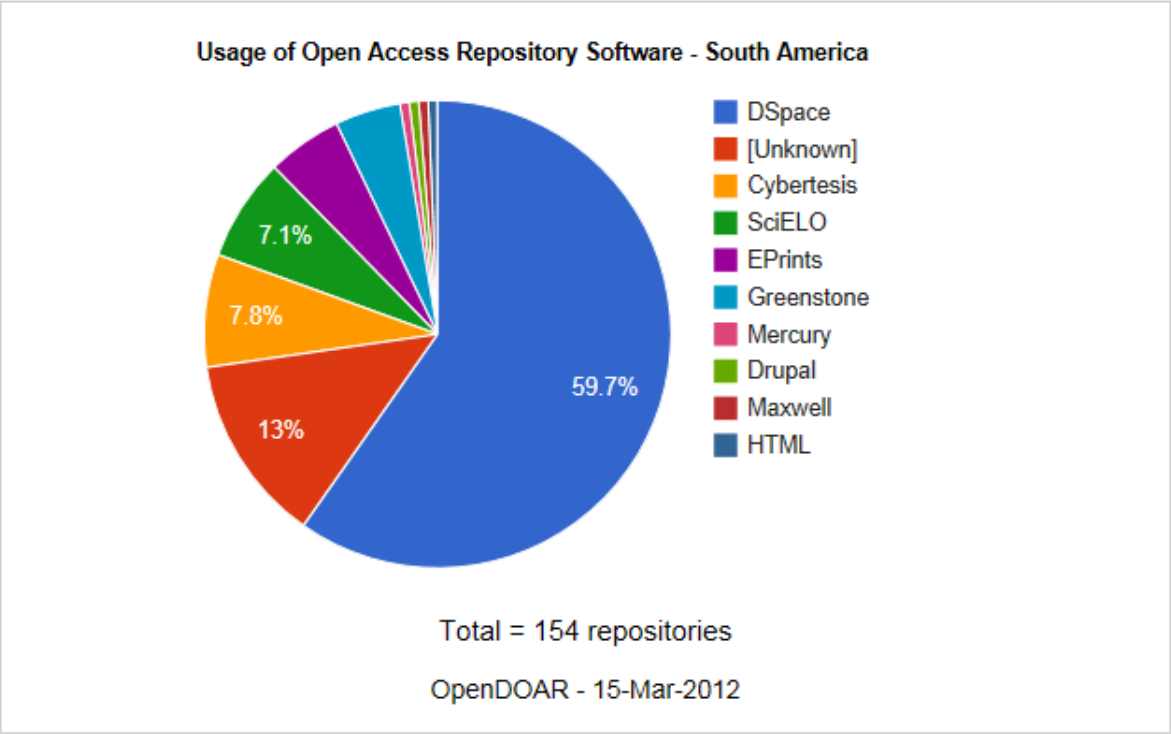
Como se observa en las figuras 13 y 14, OpenDOAR (Directorio autoritativo de repositorios académicos de libre acceso), presenta una serie de estudios estadísticos que concluyen que DSpace es una de las herramientas de gestión de recursos digitales más utilizadas a nivel mundial (UNIVERSITY OF NOTTINGHAM, 2006).

Figura 13. Uso de repositorios de libre acceso a nivel mundial



Fuente: (UNIVERSITY OF NOTTINGHAM, 2006)

Figura 14. Uso de repositorios de libre acceso en Sur América



Fuente: (UNIVERSITY OF NOTTINGHAM, 2006)

DSpace es utilizada por instituciones de investigación a nivel mundial para satisfacer necesidades de archivo digital como: repositorios institucionales, repositorios de objetos de aprendizaje, tesis Electrónicas, administración de registros electrónicos, preservación digital y publicación, entre otras.

Teniendo en cuenta los resultados del estudio comparativo (tabla 14) y los datos estadísticos presentados por OpenDOAR, se ha optado por utilizar DSpace como la herramienta para la implementación de un sistema de gestión de activos de software basado en *PEGASO*.

DSpace proporciona las funcionalidades y herramientas requeridas para implementar el catálogo de activos de software aplicando la propuesta de este trabajo de investigación: facilidades de personalización y comunicación con el usuario, herramientas para la gestión, búsqueda y recuperación de activos, herramientas para la gestión de experiencias y facilidades para la persistencia de los activos y del catálogo.

4.4 DC-SOFT: UN SISTEMA DE METADATOS PARA LA DESCRIPCIÓN DE ACTIVOS DE SOFTWARE DE *PEGASO*

DC-Soft es el conjunto de metadatos que describe los activos de software en *PEGASO*, extiende el estándar Dublin Core, hereda su estructura general y su sistema de definiciones. En general, los metadatos describen las características de los activos de software identificadas en este trabajo de investigación.

DC-Soft integra metadatos heredados del estándar Dublin Core y metadatos que permiten implementar el metamodelo que define y caracteriza el activo de software (Tabla 15). Nótese que los elementos heredados de Dublin Core se indican con (*) y los Elementos de Extensión se indican con (**).

Tabla 15. Conjunto de Metadatos DC-Soft

Conjunto de Metadatos DC-Soft		
Contenido	**Propiedad Intelectual**	**Instanciación**
Titulo*****	Autor*****	Fecha*****
Claves*****	Editor*****	Tipo*****
Descripción*****	Otros colaboradores*****	Clase**
Fuente*****	Derechos*****	Propuesta**
Idioma*		ActividadN1**
Relación*****		ActividadN2**
Cobertura*****		ActividadN3**
Versión**		Formato*****
Nota de Versión**		

4.4.1 Elementos DC-Soft relacionados con el contenido

Tabla 16. Elementos DC-Soft Relacionados con el Contenido

Elemento	Descripción	Etiqueta
Titulo	El nombre dado al activo. El nombre del activo debe expresar el conocimiento que el activo trasmite. Puede ser un nombre compuesto que incluya por ejemplo el activo de ingeniería y el activo de negocio que implementa. El administrador del repositorio debe definir una práctica estándar para nombrar los activos de software.	DC-Soft.Title
Claves	Palabras o frases clave que describen el activo. Permiten identificar las temáticas con los cuales el activo tiene relación. Se pueden incluir aspectos como: dominio de aplicación, herramientas empleadas, diagramas presentados, etc.	DC-Soft.Subject

Tabla 16. (Continuación)

Elemento	Descripción	Etiqueta
Descripción	Una descripción textual del activo de software. Brevemente orienta al usuario sobre el contenido intelectual del activo. Se puede incluir breve comentario sobre el problema o requisito que resuelve y los aspectos más relevantes de la solución.	DC-Soft.Description
Fuente	Identifica el proyecto que generó el activo o el repositorio externo que lo contiene. El proyecto puede ser académico o industrial. Complementa el contexto del activo de software con el elemento de cobertura.	DC-Soft.Source
Idioma	Describe el mecanismo de representación utilizado para exponer el activo de software como por ejemplo: diagramas, modelos, documentos, plantillas y código fuente. Depende de la notación y de la propuesta de desarrollo de software.	DC-Soft.Language
Relación	Metadato que integra dos elementos: El activo con el cual tiene relación el activo actual y la relación existente. La relación se lee del activo actual al activo identificado en este metadato. En el caso de un activo de ingeniería, este metadato se refiere al activo de negocio que implementa y la relación es de trazabilidad. En el caso de un activo de proyecto, este metadato se refiere al activo de ingeniería utilizado. Pueden darse otras relaciones entre activos de software como por ejemplo: uso, soporte y composición.	DC-Soft.Relation
Cobertura	Es la característica de cobertura intelectual del activo. Se refiere al contexto del activo (tipo de proyecto, tipo de desarrollo, dominio de aplicación, cliente, tecnología de soporte, etc.).	DC-Soft.Coverage
Versión	Identifica la versión del activo. El administrador del repositorio define el estándar para codificar la versión de un activo.	DC-Soft.Version
Nota de Versión	Permite describir las características específicas de una versión y las diferencias con otras versiones.	DC-Soft.Note

4.4.2 Elementos DC-Soft relacionados con la propiedad intelectual

Tabla 17. Elementos DC-Soft relacionados con la propiedad intelectual

Elemento	Descripción	Etiqueta
Autor	La persona y la organización responsable de la creación del activo. Si el autor es una persona de la organización, se registra la unidad de trabajo a la que pertenece. Si el autor es una persona externa, se registra la organización de la que procede.	DC-Soft.Creator

Tabla 17. (Continuación)

Elemento	Descripción	Etiqueta
Editor	La entidad o persona responsable de hacer que el activo se encuentre disponible (publicado) en su formato actual.	DC-Soft.Publisher
Otros colaboradores	Una persona u organización que haya tenido una contribución intelectual significativa.	DC-Soft.Contributor
Derechos	Se refiere a la organización propietaria del activo. Puede utilizarse una referencia (por ejemplo, una URL) para una nota sobre derechos de autor, para un servicio de gestión de derechos o para un servicio que dará información sobre términos y condiciones de acceso a un activo de software.	DC-Soft.Rights

4.4.3 Elementos DC-Soft relacionados con la instanciación

Tabla 18. Elementos DC-Soft relacionados con la instanciación

Elemento	Descripción	Etiqueta
Fecha	Fecha en la cual el activo de software se puso a disposición del usuario (se publicó) en su forma actual.	DC-Soft.Date
Tipo	La categoría del activo de software. Define si se refiere a un activo de ingeniería, a un activo de proyecto, a un activo de negocio o a un activo de base conceptual.	DC-Soft.Type
Clase	Se refiere a la clase de activo, de acuerdo con la propuesta de activos de software que hacen parte del repositorio (ver tabla 12). Puede ser por ejemplo: modelo de diseño, modelo de requisitos, modelo de negocio, etc.	DC-Soft.Class
Propuesta	Se refiere al nombre de la propuesta de desarrollo o el nombre dado al proceso software que se aplica para el desarrollo del proyecto que genera el activo de software.	DC-Soft.Proposal
ActividadN1	En conjunto con los metadatos ActividadN2 y ActividadN3, definen la jerarquía de actividades de la propuesta o proceso de software que genera el Activo. Este metadato es el nivel 1 de la jerarquía. En el caso del estándar 1517 de la IEEE, se refiere al proceso.	DC-Soft.ActivityL1
ActividadN2	Es el nivel 2 de la jerarquía de actividades de la propuesta de desarrollo o del proceso de software. En el caso de aplicar el estándar 1517 de la IEEE, se refiere a la actividad.	DC-Soft.ActivityL2
ActividadN3	Es el nivel 3 de la jerarquía de actividades de la propuesta de desarrollo o del proceso de software. En el caso de aplicar el estándar 1517 de la IEEE, se refiere a la tarea	DC-Soft.ActivityL3

Tabla18. (Continuación)

Elemento	Descripción	Etiqueta
Formato	Ofrece al usuario información sobre el formato, el software y posiblemente el hardware que se necesitaría para recuperar el activo de software.	DC-Soft.Format

4.4.4 Un ejemplo DC-Soft

Tabla 19. Un ejemplo DC-Soft: El modelo de diseño del sistema punto de venta

Etiqueta		Ejemplo
Contenido	DC.Soft.Title	Sistema punto de venta
	DC.Soft.Subject	Modelo de diseño, Punto de Venta, UML, RUP, Sys Ltda.
	DC.Soft.Description	Presenta varios diagramas de la notación UML y un documento descriptivo. Las clases conceptuales describen el diseño detallado del caso de aplicación del punto de venta. El modelo aborda aspectos como: descuentos e impuestos y la interacción con otros sistemas como contabilidad e inventario. Hace parte del proyecto del Sistema de información comercial de la Empresa Sys Ltda. El proyecto sigue la metodología RUP, el modelo de diseño es un activo de ingeniería que se crea durante la disciplina de análisis y diseño.
	DC.Soft.Source	Proyecto Sistema de Información Comercial Sys Ltda.
	DC.Soft.Language	Plantilla, Documento, UML
	DC.Soft.Relation	Descuentos, trazabilidad
	DC.Soft.Coverage	Dominio Comercio, Dominio Punto de Venta
	DC.Soft.Version	V 1.0
	DC.Soft.Note	Versión que sigue el formato de presentación de documentos técnicos del sistema de gestión de calidad de la empresa vigente hasta el 10 de noviembre de 2011.
Propiedad Intelectual	DC.Soft.Creator	Alexander Barón Salazar, Grupo de investigación en Ingeniería del Software.
	DC.Soft.Publisher	Universidad Eafit, Universidad de Nariño
	DC.Soft.Contributor	Raquel Anaya de Páez
	DC.Soft.Rights	Alexander Barón Salazar, Universidad Eafit, Universidad de Nariño

Tabla 19. (Continuación)

Etiqueta		Ejemplo
Instanciación	DC.Soft.Date	22/11/2011
	DC.Soft.Type	Activo de ingeniería
	DC.Soft.Class	Modelo de diseño
	DC.Soft.Proposal	RUP
	DC-Soft.ActivityL1	Procesos de implementación del software
	DC-Soft.ActivityL2	Diseño detallado del software
	DC-Soft.ActivityL3	Diseño detallado del software
	DC.Soft.Format	Microsoft Word

5. LA GUÍA DE USO DE PORTAL ACTIVO: UN SISTEMA DE GESTION DE ACTIVOS DE SOFTWARE

5.1 INTRODUCCION

PORTAL ACTIVO es una solución computacional en plataforma web como respuesta tecnológica a la propuesta *PEGASO* que promueve la reutilización de activos de software a través de catálogos inmersos dentro de un repositorio digital.

PORTAL ACTIVO hace uso del repositorio DSpace UDENAR, siendo éste una adaptación de la propuesta original DSpace. DSpace UDENAR implementa el conjunto de metadatos DC-Soft propuesto en este trabajo de investigación para describir activos de software.

En este capítulo se describe la forma como Portal Activo debe ser utilizado y las funcionalidades principales a manera de procesos. Inicialmente se muestra la clasificación de los usuarios que interactúan con la herramienta de acuerdo al rol que desempeñan, más adelante se presenta un resumen de los procesos que gestiona Portal Activo y finalmente, se describen de manera detallada los procesos más relevantes.

5.2 LOS PROCESOS EN PORTAL ACTIVO

Los roles que pueden desempeñar los usuarios en la herramienta Portal Activo se relacionan y describen en la tabla 20. Estos roles se definen haciendo uso de las facilidades que provee DSpace, de acuerdo a un nivel de privilegios que son otorgados a los usuarios por el sistema de autorización. En Portal Activo, los usuarios pueden ser integrados en grupos y los privilegios se asignan individualmente o a nivel de grupo.

Tabla 20. Usuarios que interactúan con la Herramienta Portal Activo

Usuario	Descripción
Administrador	Usuario con privilegios de acceso y control total sobre el repositorio
Supervisor	Usuario con privilegios para revisar y publicar activos
Editor	Usuario con privilegios para matricular activos
Observador registrado	Usuario con privilegios para buscar y recuperar cualquier activo del repositorio.
Observador anónimo	Usuario con privilegios únicamente para buscar y recuperar ciertos activos.

Los procesos en Portal Activo y los usuarios involucrados se relacionan en la tabla 21 y se describen mas adelante.

Tabla 21. Procesos en Portal Activo

Número	Proceso	Usuario Responsable
1	Registro de usuarios	Todos Usuarios
2	Administración de usuarios	Administrador
3	Configuración de los metadatos del repositorio	Administrador
4	Configuración del proceso software	Administrador
5	Configuración de la estructura de almacenamiento del repositorio	Administrador
6	Matricula de activos	Editor
7	Revisión y Publicación de activos	Supervisor
8	Búsqueda de activos	Todos usuarios
9	Consultar información sobre un activo	Todos usuarios
10	Selección y recuperación de activos	Todos usuarios

De la figura 15 a la 18, se presenta por usuario responsable y a manera de procesos el conjunto de funcionalidades que provee Portal Activo.

Figura 15. Diagrama de flujo de los Procesos asociados a todos los usuarios

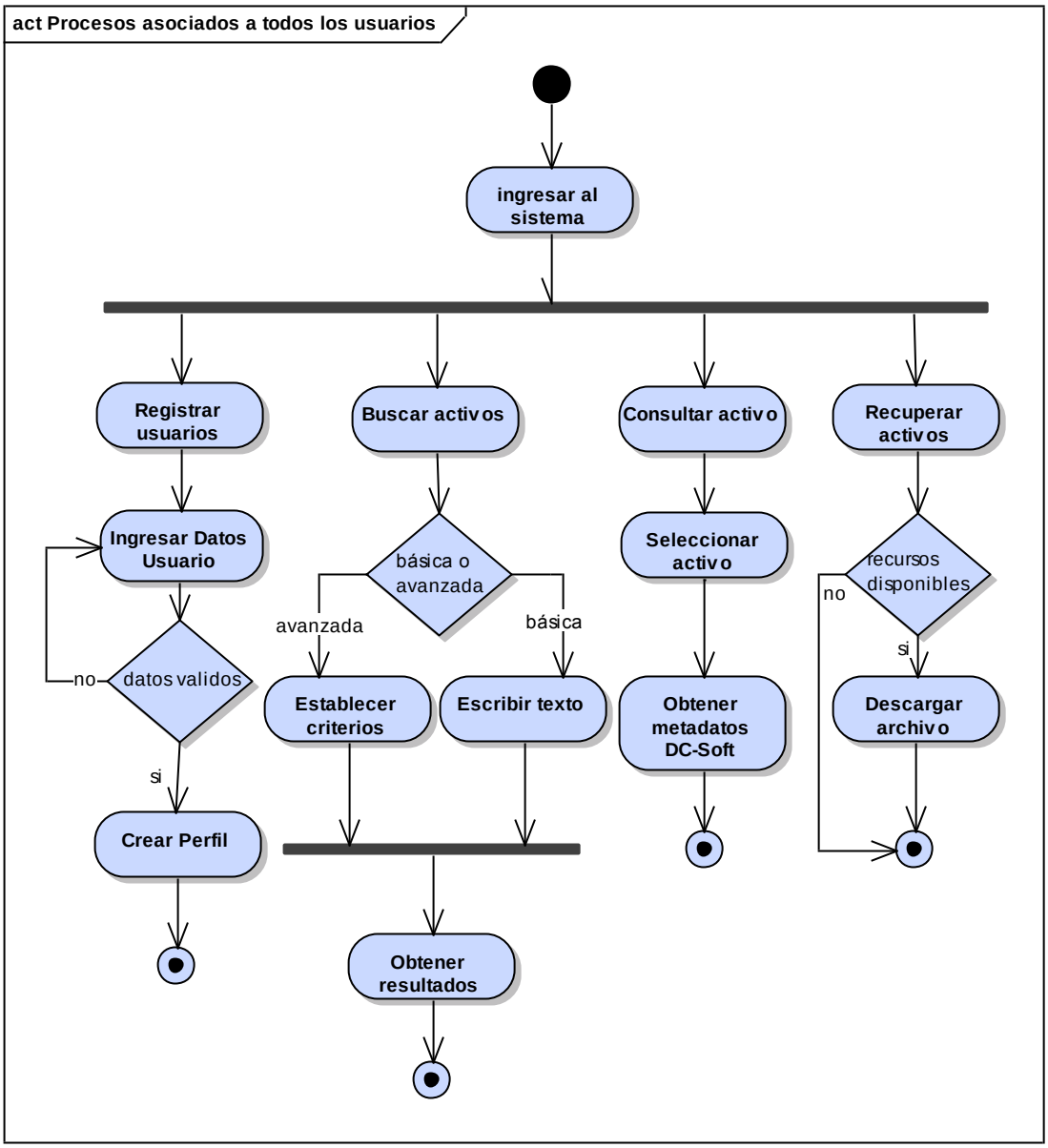


Figura 16. Diagrama de flujo de los Procesos asociados al usuario administrador

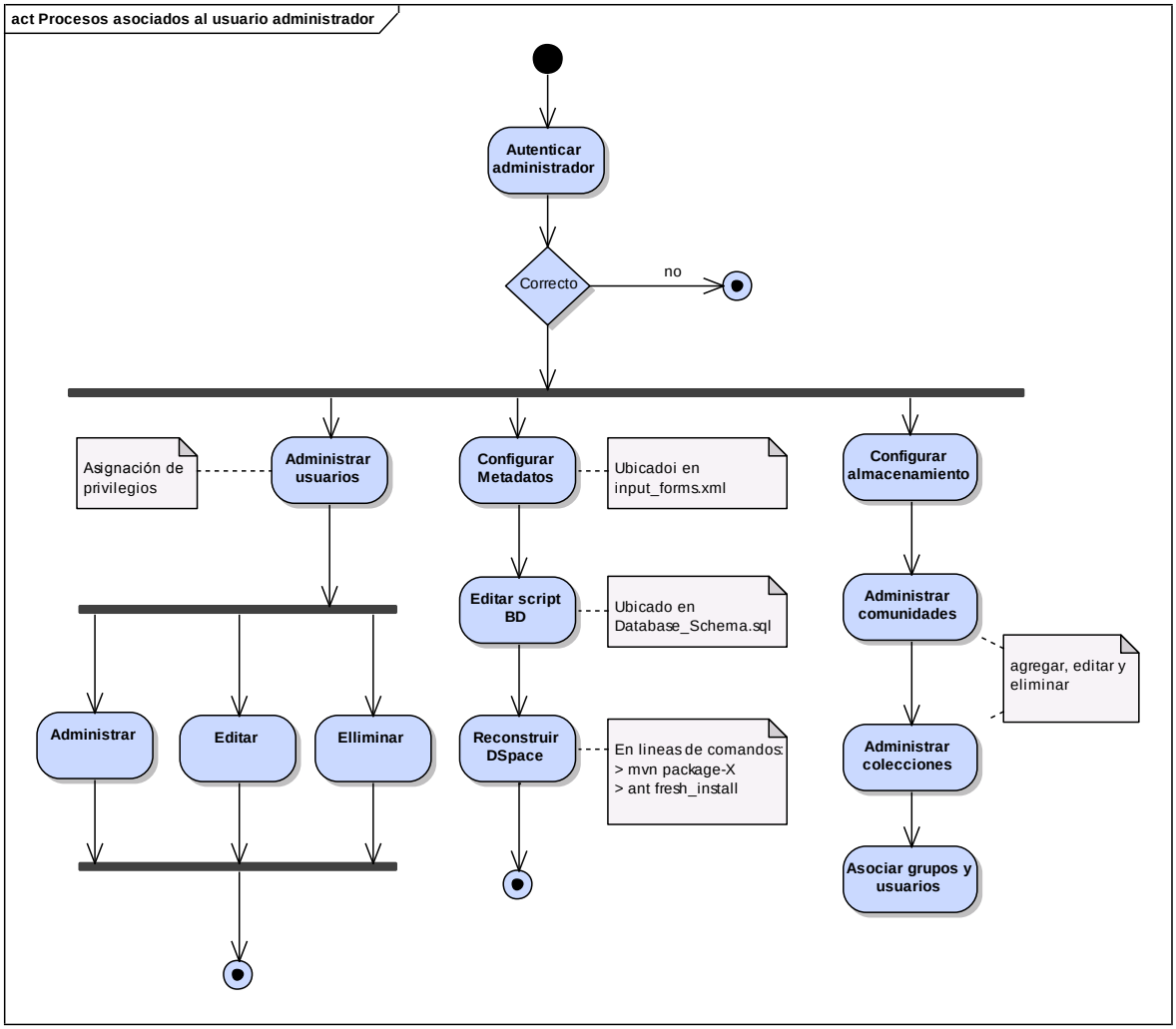


Figura 17. Diagrama de flujo de los Procesos asociados al usuario editor

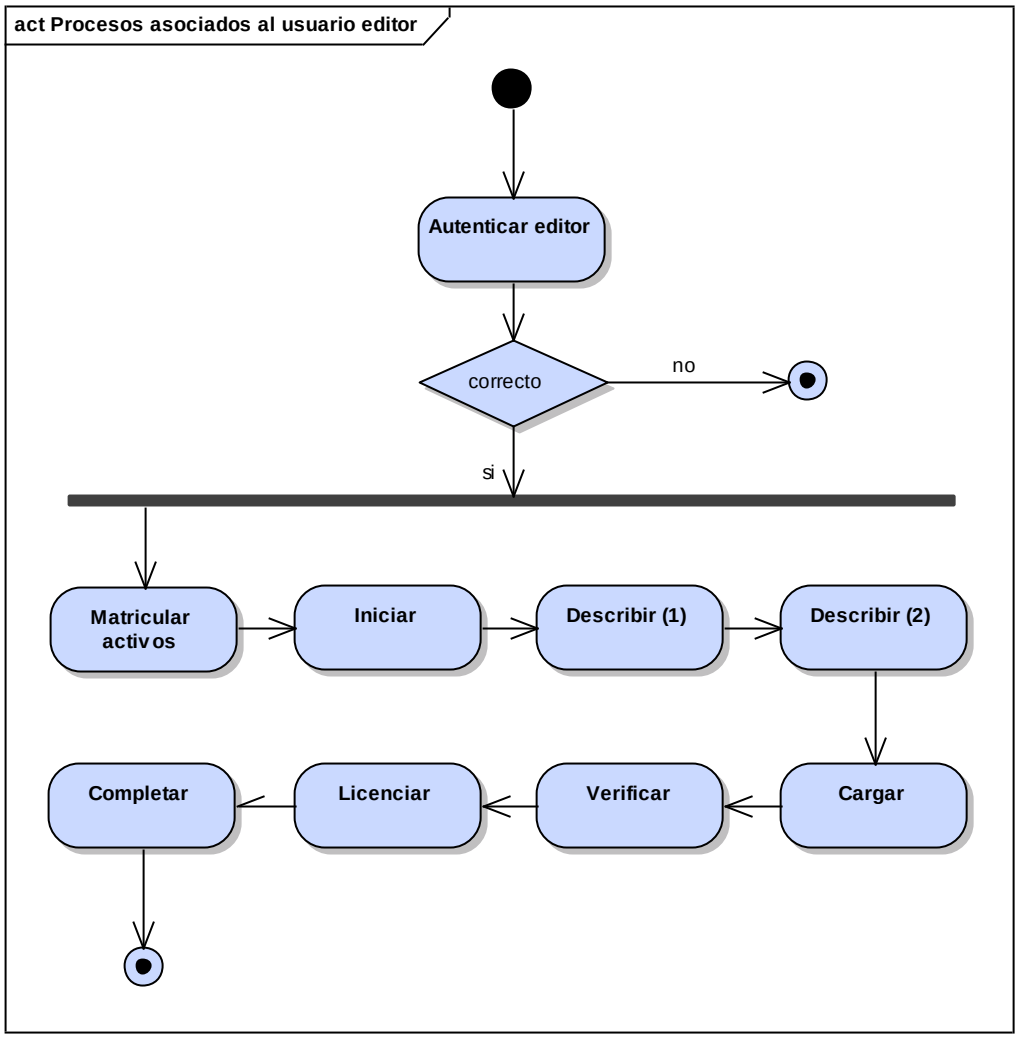
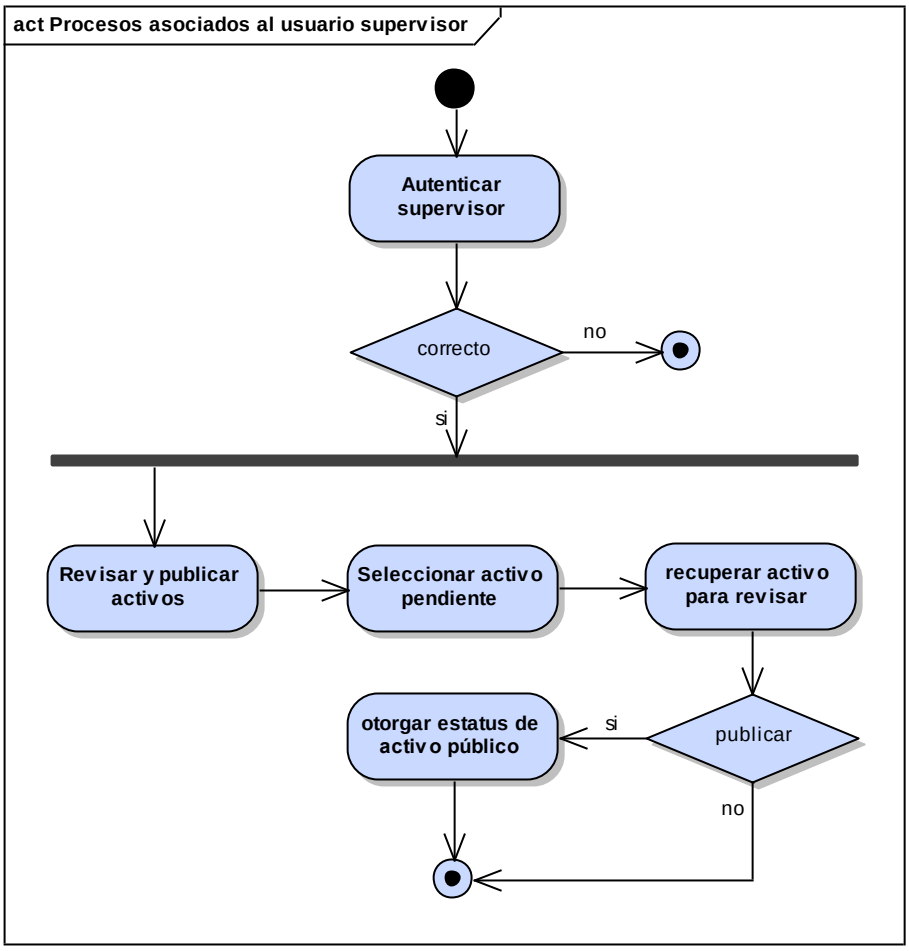


Figura 18. Diagrama de flujo de los Procesos asociados al usuario supervisor



5.2.1 Registro de usuarios

Todos los usuarios de Portal Activo deberán suministrar los datos de correo electrónico, nombres, apellidos, teléfono, lenguaje y las opciones de uso de registro y certificado, como se muestra en la figura 19. Una vez validada la información se procede a realizar la creación de perfil de tal suerte que el usuario tenga acceso a la herramienta.

Figura 19. Registro de usuario en DSpace UDENAR

Fuente: (BARÓN, y otros, 2011)

5.2.2 Administración de usuarios

En DSpace UDENAR, una vez ingresa el administrador del repositorio digital se puede efectuar la gestión de usuarios con la posibilidad de crearlos, editarlos o eliminarlos (figura 20).

Figura 20. Administración de usuarios en DSpace UDENAR

Fuente: (BARÓN, y otros, 2011)

5.2.3 Configuración de los metadatos del repositorio

Para configurar DC-Soft, el conjunto de metadatos orientado a la descripción de activos de software que es una extensión de Dublin Core, se deben intervenir archivos de configuración a nivel de interfaz de usuario (figura 21), script de base de datos (figura 22), y finalmente se debe reconstruir DSpace UDENAR para su despliegue (figura 23).

Figura 21. Configuración de metadatos en interfaz de usuario

input-forms.xml

```
<form-definitions>

  <form name="traditional">
    <page number="1">
      <field>
        <dc-schema>dc</dc-schema>
        <dc-element>contributor</dc-element>
        <dc-qualifier>author</dc-qualifier>
        <repeatable>true</repeatable>
        <label>Authors</label>
        <input-type>name</input-type>
        <hint>Enter the names of the authors of this item below.</hint>
        <required></required>
      </field>
```

Fuente: (BARÓN, y otros, 2011)

Figura 22. Configuración de metadatos en base de datos

database_schema.sql

```
CREATE SEQUENCE harvested_collection_seq;
CREATE SEQUENCE harvested_item_seq;

-------------------------------------------------------
-- IEEE1517_Process
-------------------------------------------------------
CREATE TABLE IEEE1517_Process
(
    id_process       INTEGER PRIMARY KEY,
    name_process     VARCHAR(256)
);

-------------------------------------------------------
-- IEEE1517_Activity
-------------------------------------------------------
CREATE TABLE IEEE1517_Activity
(
    id_activity      INTEGER PRIMARY KEY,
    id_process       INTEGER REFERENCES IEEE1517_Process(id_process),
    name_activity    VARCHAR(256)
);

-------------------------------------------------------
-- IEEE1517_Task
-------------------------------------------------------
CREATE TABLE IEEE1517_Task
(
```

Fuente: (BARÓN, y otros, 2011)

Figura 23. Comandos de reconstrucción y despliegue de DSpace UDENAR

```
cd [dspace-source]/dspace
mvn package
cd [dspace-source]/dspace/target/dspace-<version>-build.dir
ant fresh_install
cp -r [dspace]/webapps/* [tomcat]/webapps
/etc/init.d/tomcat start
[dspace]/bin/dspace create-administrator
```

Fuente: (BARÓN, y otros, 2011)

5.2.4 Configuración del proceso software

Este trabajo de investigación implementa como proceso software por defecto el propuesto por la norma IEEE1517. Sin embargo, es posible configurar otros procesos software modificando los datos de los metadatos ActivityL1, ActivityL2 y ActivityL3. Dicha información reside en la base de datos del repositorio y se modifica mediante el script mostrado en la figura 22; no se requiere para este caso, la reconstrucción de DSpace.

5.2.5 Configuración de la estructura de almacenamiento del repositorio

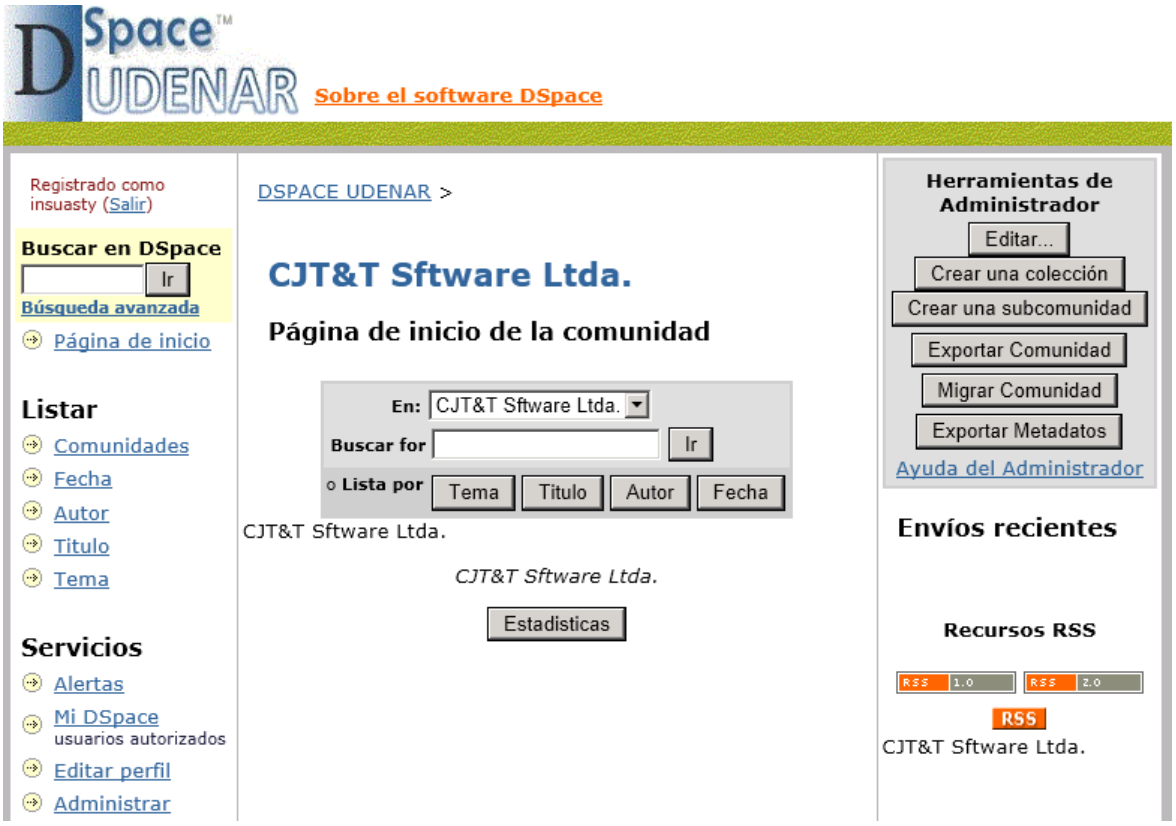
Siendo la configuración del repositorio digital una de las tareas más largas, diferentes aspectos deben tenerse en cuenta para hacer que DSpace UDENAR trabaje en forma adecuada. La estructura jerárquica de DSpace inicia con comunidades y colecciones; estos elementos pueden ser agregados, editados y eliminados. La forma de crear una comunidad se muestra en la figura 24.

Figura 24. Creación de comunidades en DSpace UDENAR

Fuente: (BARÓN, y otros, 2011)

Se debe tener en cuenta que las comunidades integran catálogos (cuyo término en DSpace es colección). En la figura 25 se puede apreciar la página de inicio de la comunidad y las herramientas del administrador, que incluye la opción para crear colecciones.

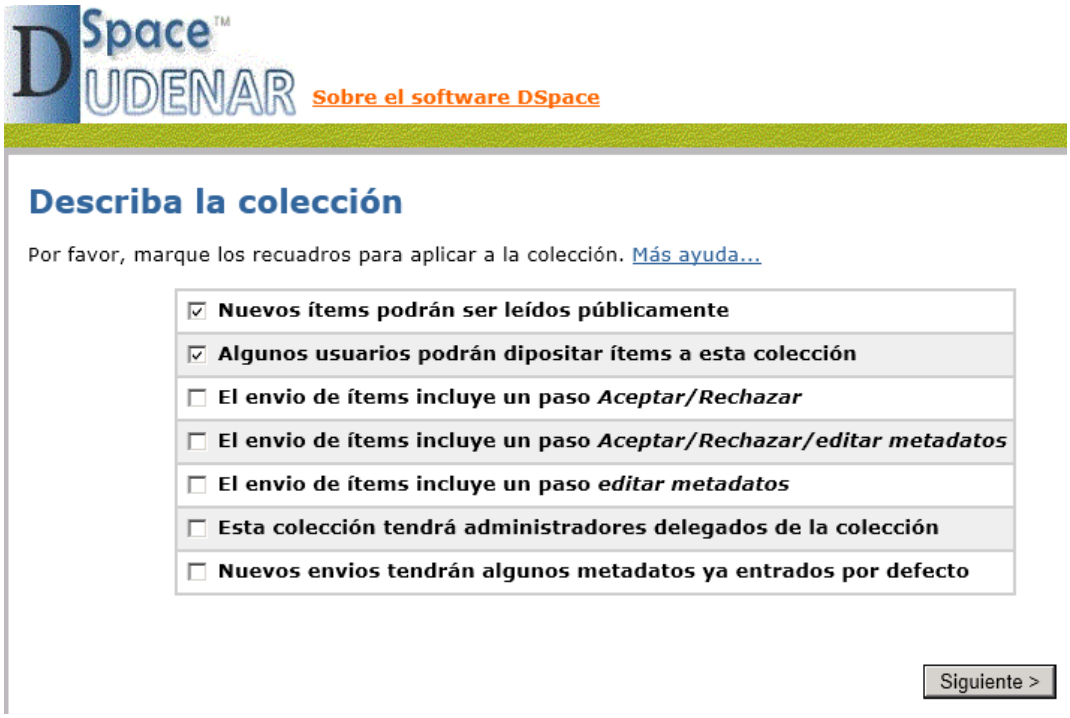
Figura 25. Página de inicio de la comunidad en DSpace UDENAR.



Fuente: (BARÓN, y otros, 2011)

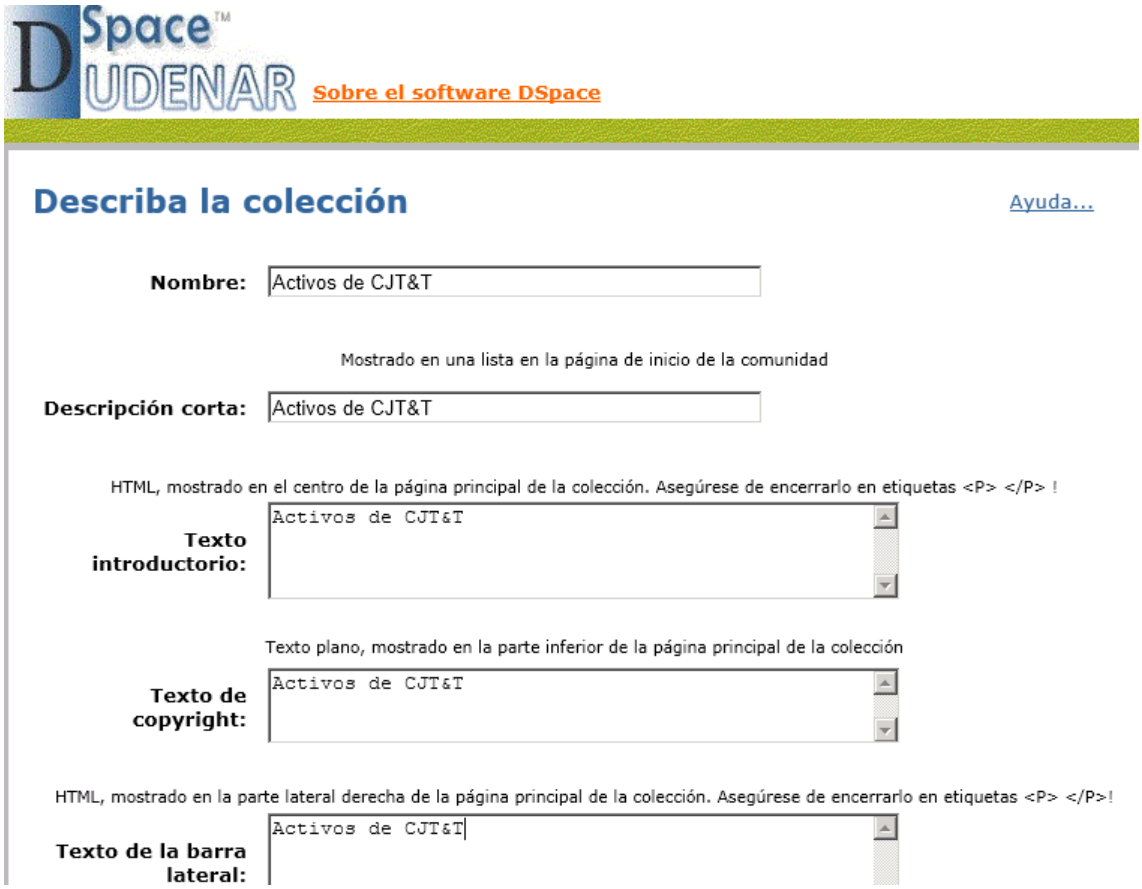
Cuando el administrador elige la opción crear una colección, la herramienta activa un asistente de dos pasos que permite describir la nueva colección. Las figuras 26 y 24 muestran dicho asistente.

Figura 26. Descripción de colecciones en DSpace UDENAR (1)



Fuente: (BARÓN, y otros, 2011)

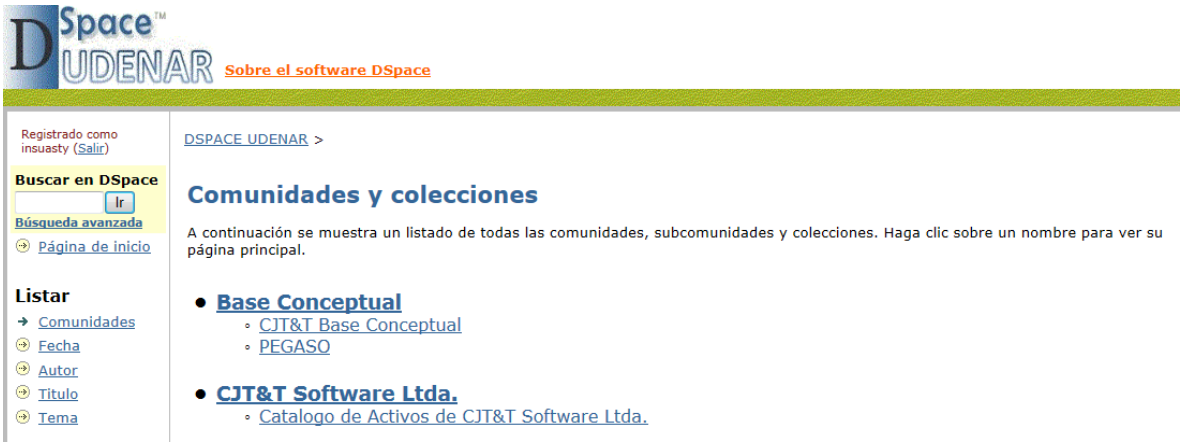
Figura 27. Descripción de colecciones en DSpace UDENAR (2)



Fuente: (BARÓN, y otros, 2011)

El resultado de las tareas de configuración de la estructura de almacenamiento del repositorio en DSpace UDENAR, se muestra en la figura 28.

Figura 28. Estructura de comunidades y colecciones en DSpace UDENAR



Fuente: (BARÓN, y otros, 2011)

5.2.6 Matricula de activos

Se debe tener presente que solo usuarios con privilegios para agregar activos de software (administrador y editor), podrán realizar esta tarea por lo cual el proceso requiere de un paso preliminar de autenticación.

DSpace UDENAR provee un asistente que guía el proceso de matrícula de activos que DSpace define como ítems (figuras 29 a 34).

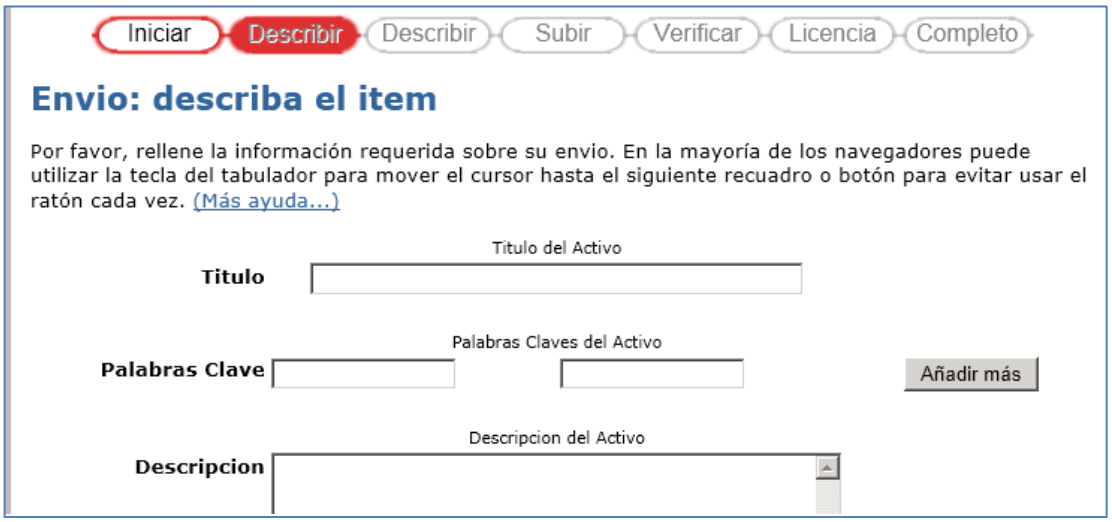
En el paso Iniciar, se pregunta si el activo tiene más de un titulo, si ha sido publicado con anterioridad o si contiene más de un archivo asociado (figura 26).

Figura 29. Matricula de activos en DSpace paso iniciar

Fuente: (BARÓN, y otros, 2011)

En los paso2 Describir (1ra parte) y Describir (2da parte), se requiere información sobre título, una descripción general, así como el resto de metadatos según DC-Soft (figura 30).

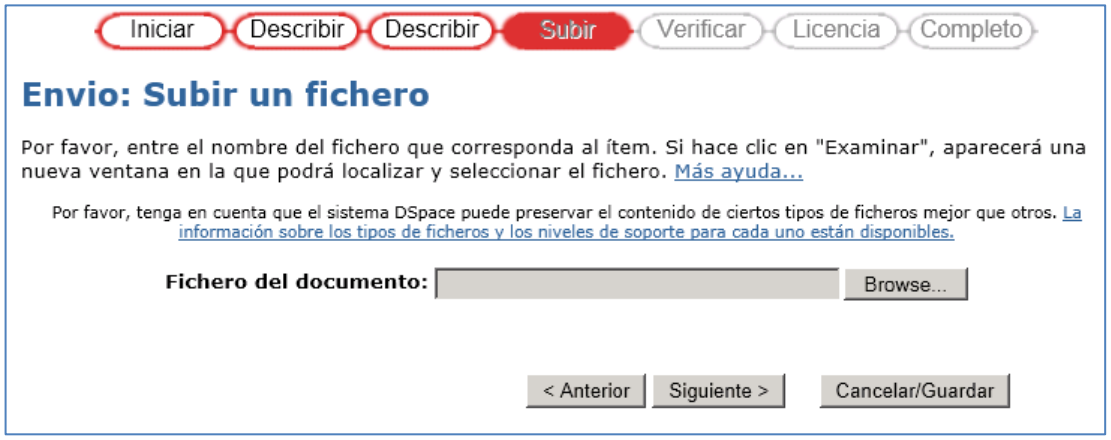
Figura 30. Matricula de activos en DSpace paso describir



Fuente: (BARÓN, y otros, 2011)

En el paso Cargar, Subir en DSpace, los archivos pueden ser cargados al servidor (figura 31).

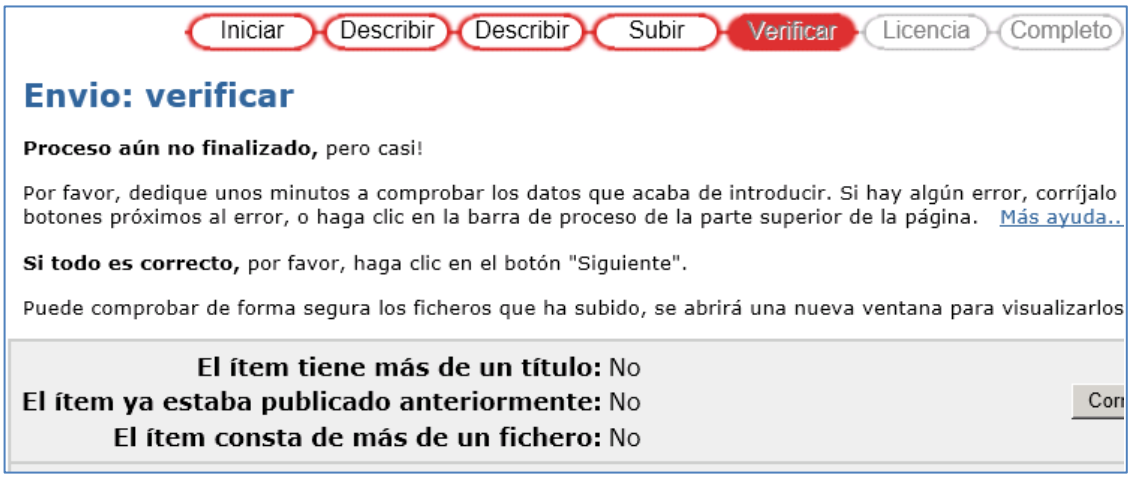
Figura 31. Matricula de activos en DSpace paso subir



Fuente: (BARÓN, y otros, 2011)

En el paso Verificar, se revisa los datos suministrados acerca del activo de software (figura 32).

Figura 32. Matricula de activos en DSpace paso verificar



Fuente: (BARÓN, y otros, 2011)

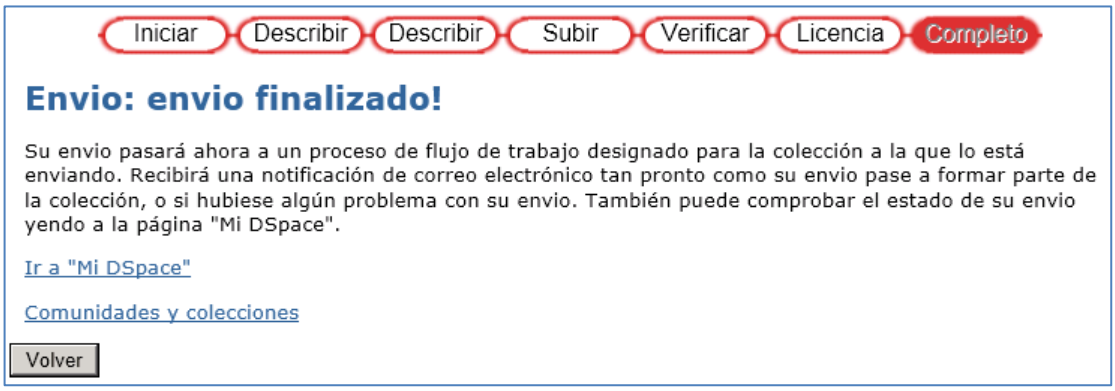
En el paso Licencia, se aceptan los términos de la licencia de publicación de activos de software (figura 33).

Figura 33. Matricula de activos en DSpace paso licencia

Fuente: (BARÓN, y otros, 2011)

En el paso Completo, se confirma el envío del activo (figura 34).

Figura 34. Matricula de activos en DSpace paso completo



Fuente: (BARÓN, y otros, 2011)

5.2.7 Revisión y publicación de activos

Los usuarios con roles de Administrador y Supervisor, ingresan al sistema y tiene asignadas tareas de revisión; posterior a revisar los activos, los usuarios determinan si se debe o no publicar dicho activo a fin de ser consumido por los usuarios del Portal. Al igual que el proceso anterior, solo usuarios con privilegios para revisión podrán realizar esta tarea por lo cual el proceso requiere de un paso preliminar de autenticación. La interface correspondiente al proceso de revisión y publicación de activos, se muestra en la figura 35.

Figura 35. Revisión y publicación de activos en DSpace UDENAR

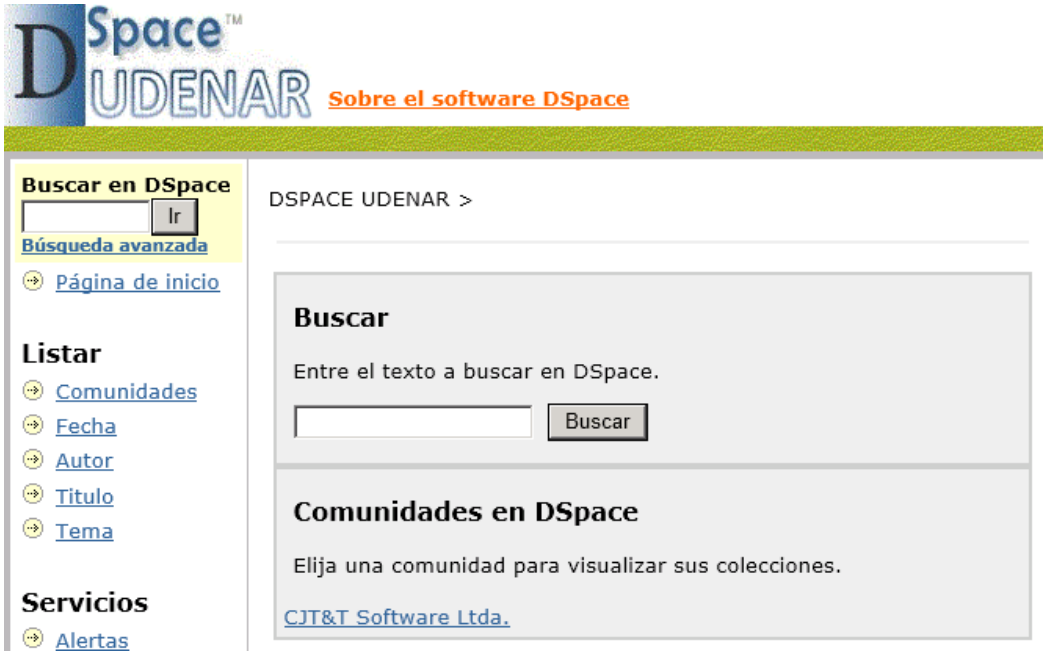
Fuente: (BARÓN, y otros, 2011)

5.2.8 Búsqueda de activos

Para realizar una búsqueda el usuario debe elegir el modo de búsqueda; básico o avanzado y dar la información requerida, la herramienta despliega posteriormente los resultados obtenidos.

El modo básico realiza la búsqueda por las coincidencias de una cadena en los campos de publicación (metadatos), el modo básico de búsqueda se muestra en la figura 36.

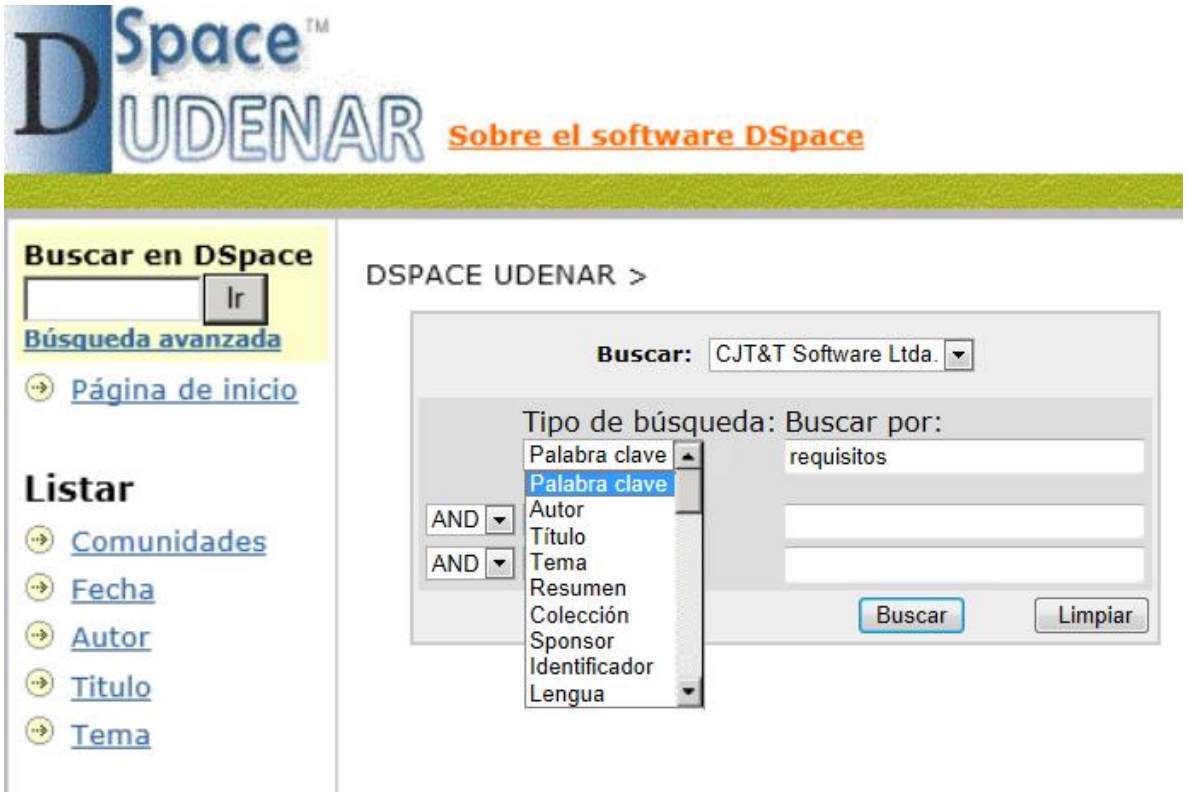
Figura 36. Búsqueda de activos en DSpace UDENAR modo básico



Fuente: (BARÓN, y otros, 2011)

La búsqueda avanzada se realiza por medio de la combinación booleana de criterios según la lista de metadatos (figura 37).

Figura 37. Búsqueda de activos en DSpace UDENAR modo avanzado



Fuente: (BARÓN, y otros, 2011)

5.2.9 Consultar información sobre un activo

Si la búsqueda tuvo éxito, se procede a seleccionar un ítem del conjunto de activos que cumplieron con los criterios requeridos (figura 38). El sistema despliega los metadatos según DC-SOFT para el usuario, adjuntando además los recursos (archivos, diagramas, formularios, código, etc.) si están disponibles (figuras 39 y 40).

Figura 38. Resultados de búsqueda en DSpace UDENAR

DSPACE UDENAR >

Resultados de búsqueda

Buscar: Todo DSpace
por requisitos Ir

Resultados 1-2 de 2.

Resultados por Pagina 10 | Ordenar por Relevancia Orden Descendente Et al - ilimitado Ilimitado Actualizar Exportar Metadatos

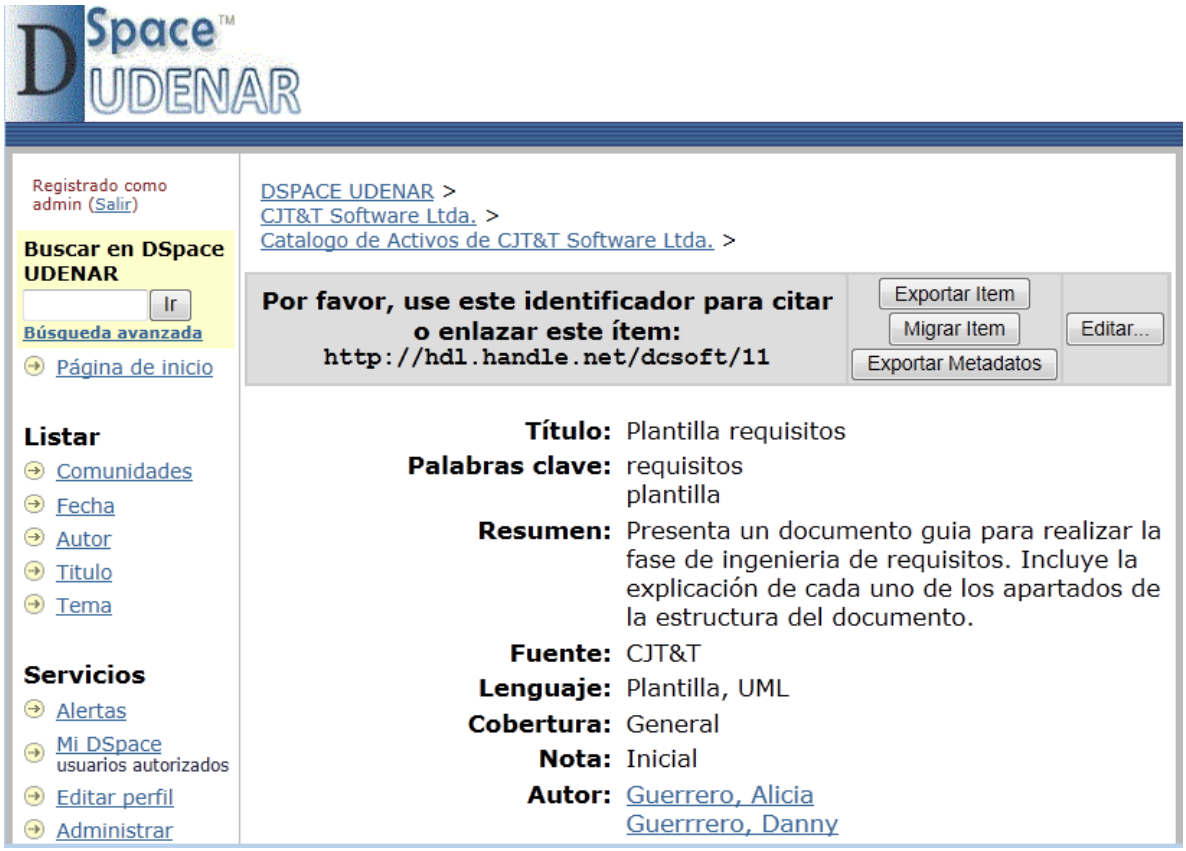
Hits de ítem:

Fecha de publicación	Título	Personal(Editor; Autor(es); Colaborador)
15-Mar-2012	Plantilla requisitos	*Alexander Barón; Guerrero, Alicia; Guerrrero, Danny; Universidad Eafit, Universidad de Nariño*
15-Mar-2012	Documento requisitos biblioteca Colegio Artemio Mendoza	*Alexander Barón; Cuchala, Wilmer; Chiran, Paulo; CJT&T, Universidad de Nariño, Universidad Eafit*

1

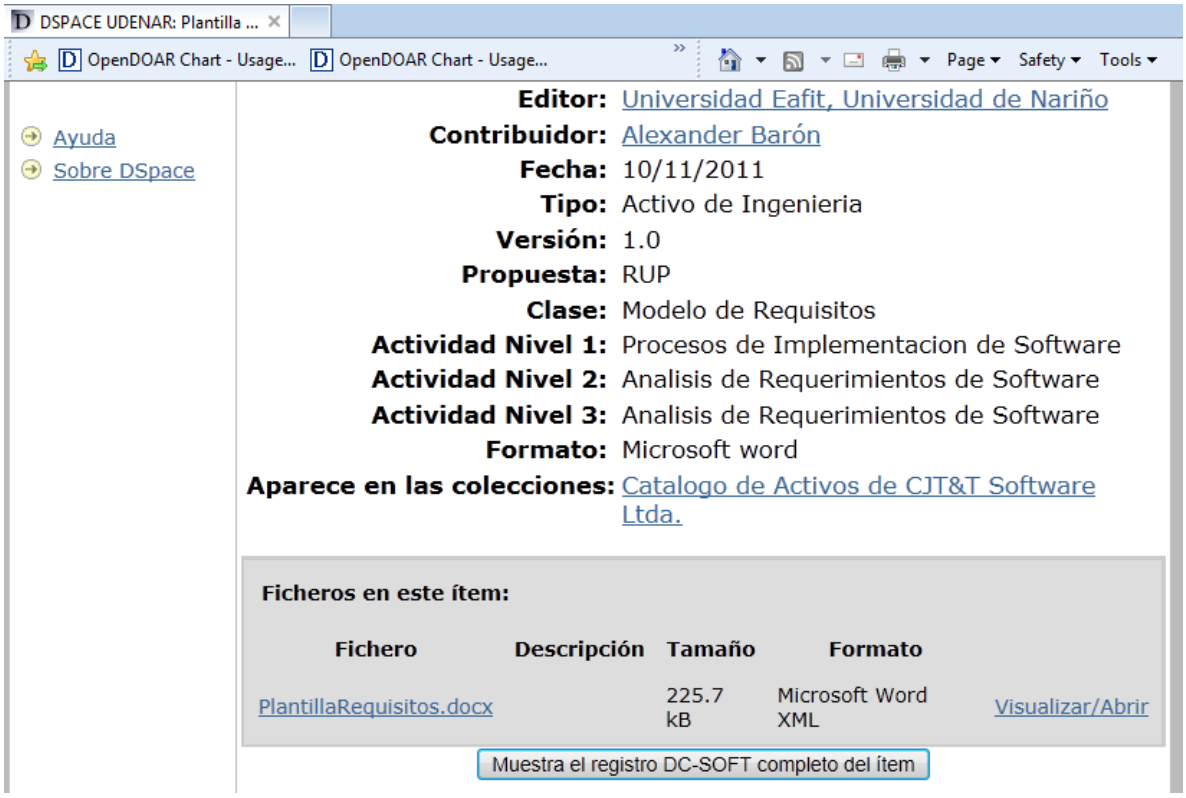
Fuente: (BARÓN, y otros, 2011)

Figura 39. Metadatos del activo seleccionado en DSpace UDENAR (1)



Fuente: (BARÓN, y otros, 2011)

Figura 40. Metadatos del activo seleccionado en DSpace UDENAR (2)



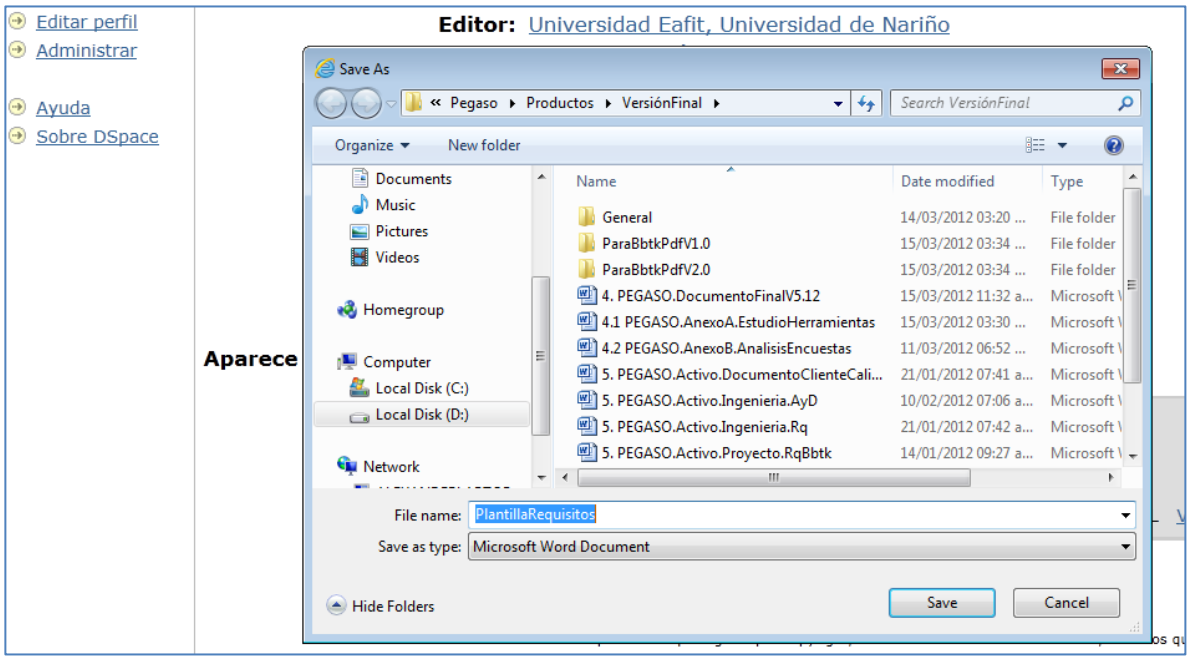
Fuente: (BARÓN, y otros, 2011)

5.2.10 Recuperación de activos

El repositorio digital esta diseñado para almacenar grandes volúmenes de archivos administrados por datos en un contenedor especifico. De estar disponibles los recursos (posterior al proceso de búsqueda), los archivos pueden ser descargados.

Como se puede apreciar en la parte figura 40, al finalizar el despliegue de los metadatos que describen el activo, se muestra el apartado ficheros en este ítem, de los cuales el usuario puede elegir el que desea descargar, posterior a esta acción el usuario debe indicar la ruta donde desea que el archivo sea guardado, como lo muestra la figura 41.

Figura 41. Descarga de archivos en DSpace UDENAR



Fuente: (BARÓN, y otros, 2011)

6. EXPERIENCIA DE IMPLEMENTACION EN UN CONTEXTO REAL

6.1 INTRODUCCION

A fin de validar la propuesta por medio de su implementación en un contexto real, se ha realizado una alianza con la empresa de software CJT&T. En este capitulo se documenta este ejercicio. Inicialmente, se hace una breve presentación de la empresa, posteriormente se describe la experiencia de uso del portal y finalmente, se exponen los resultados de la experiencia obtenidos a partir de dos encuestas aplicadas a los participantes.

6.2 PRESENTACION DE LA EMPRESA

CJT&T es una empresa dedicada al desarrollo e implantación de proyectos informáticos. Actualmente, la compañía cuenta con tres líneas de negocio: la principal y en la que centra su mayor esfuerzo, la construcción de soluciones informáticas a la medida para todo tipo de organizaciones; el licenciamiento de software y el suministro de servidores y computadores (CJT&T Ingeniería de Software, 2007).

El plan estratégico organizacional identifica como retos principales: Incrementar la utilidad financiera, elevar los niveles de productividad, mejorar la calidad de los productos y disminuir los tiempos de los proyectos de construcción de software. CJT&T plantea el cumplimiento de estos objetivos por medio de la implementación de un sistema integrado de gestión de calidad que incluya entre otros elementos un proceso estandarizado debidamente documentado, socializado y certificado, que adopte metodologías, técnicas y estrategias de vanguardia en el campo de la ingeniería de software.

6.3 DESCRIPCION DE LA EXPERIENCIA

Como experiencia de implementación, la propuesta es aplicada al proceso de requisitos de un proyecto de software y se realizó siguiendo las siguientes actividades: capacitación, preparación de la herramienta Portal Activo y aplicación de Portal Activo. El proceso de capacitación abordó dos frentes: la base conceptual de la propuesta y el uso de la herramienta Portal Activo que la implementa. La preparación de la herramienta Portal Activo incluye la configuración de la estructura de almacenamiento del repositorio y la definición y construcción de activos iniciales. La aplicación de Portal Activo integra la búsqueda, la recuperación y el uso del activo de software.

A lo largo de toda la experiencia el equipo de trabajo estuvo integrado por 9 personas (6 de la empresa CJT&T Limitada, 2 de la Universidad de Nariño y el autor de este trabajo de investigación). Por las características propias de las actividades, en algunas de ellas, solo una parte del equipo tuvo participación.

El esfuerzo invertido en el desarrollo de la experiencia se muestra en la tabla 22.

Tabla 22. Esfuerzo de la experiencia

Actividad		Número de horas	Número de participantes
Capacitación	Base Conceptual	8	9
	Portal Activo	12	9
Preparación Portal Activo	Configuración estructura almacenamiento	2	9
	Definición y construcción de activos	48	4
Aplicación Portal Activo	Búsqueda del activo	16	3
	Recuperación del activo		3
	Uso del activo		3
Total		**86**	

6.3.1 Capacitación

La capacitación correspondiente a la base conceptual se realizó en 4 sesiones de 2 horas cada una. En lo referente a la capacitación de la herramienta Portal Activo se realizaron 6 sesiones de 2 horas cada una y el acompañamiento permanente durante su uso, para un total de 20 horas de capacitación.

En el proceso de capacitación participó todo equipo de trabajo: De la empresa CJT&T Limitada 6 personas, 2 directivos y 4 ingenieros auxiliares, por la Universidad de Nariño 2 personas como equipo de soporte, un docente investigador y una estudiante de pregrado. Este equipo estuvo liderado por el autor de este trabajo de investigación.

6.3.2 Preparación de la herramienta Portal Activo

La preparación de la herramienta *Portal Activo* consiste en la configuración de la estructura de almacenamiento del repositorio digital y la definición y construcción de los activos iniciales que hacen parte del repositorio.

6.3.2.1 Configuración de la estructura de almacenamiento del repositorio

La estructura de comunidades y colecciones definida por CJT&T Limitada incluye: las comunidades: *Base Conceptual* que integra las colecciones *CJT&T Base Conceptual* y *PEGASO* y la comunidad CJT&T Software Ltda. Que integra la colección Catalogo de Activos CJT&T Software Ltda.

La colección *Catálogo de Activos CJT&T Software Ltda.,* permite gestionar los activos de software*,* la colección *PEGASO* permite gestionar los activos de base conceptual de la propuesta y la colección *CJT&T Base Conceptual* permite gestionar los activos de base conceptual generados por la organización.

Todas las personas que participaron en el proyecto leyeron previamente los activos de base conceptual almacenados en la colección *PEGASO* de la comunidad *Base Conceptual*.

6.3.2.2 Definición y construcción de los activos iniciales

Teniendo en cuenta que se trata de un repositorio digital incipiente y que la organización usuaria plantea como una de sus principales falencias la falta de procedimientos y documentación estandarizados, en consenso con las directivas de CJT&T, se determina implementar activos de ingeniería que permitan unificar procedimientos y documentación.

En tal sentido y de acuerdo con las clases de activos de software que propone este trabajo de investigación, se construyen los activos de software que se relacionan en las tabla 23. Estos activos son de uso obligatorio para los funcionarios de CJT&T y contienen instrucciones precisas sobre su aplicación que asisten a los desarrolladores en su construcción, igualmente, tienen activos de proyecto relacionados como experiencias de aplicación.

Tabla 23. Activos iniciales que hacen parte del repositorio

Nombre	Descripción
Plantilla Documento Visión	Define la vista que tienen las personas involucradas del producto que se va a desarrollar, especificada en términos de las necesidades y características claves de dichas personas. Proporciona la base contractual para los requisitos técnicos más detallados.
Plantilla Plan Proyecto	Especifica los objetivos, el alcance y la descripción detallada del sistema. Define las directrices que guían el desarrollo y hace parte de la documentación técnica y empresarial del proyecto.
Plantilla Requisitos	Define el escenario donde la solución computacional se ajusta a ciertas expectativas de quienes están en el lugar donde el sistema se desplegara. Consiste en una amplia documentación donde se puede agrupar perspectivas desde la arquitectura, la estructura, el comportamiento así como aspectos funcionales y no funcionales
Plantilla Análisis y Diseño	Establece la realización de los casos de uso en clases, pasando desde una representación en términos de análisis hacia una de diseño, de acuerdo al avance del proyecto. Es este activo uno de los de mayor complejidad ya que involucra diferentes vistas de diseño según actores, según funciones, según tiempos, según secuencias, según relaciones, etc. Todos los aspectos que tengan que ver con la fase de diseño esta descrito en activos simples del corpus documental de diseño.

Tabla 23. (Continuación)

Nombre	Descripción
Requisitos Biblioteca	Es una experiencia de uso del activo: *Plantilla Requisitos*
Visión Planes Fidelización	Es una experiencia de uso del activo: *Plantilla Documento Visión*

Para la construcción de los activos de software, se analizaron proyectos anteriores desarrollados por CJT&T, este ejercicio permitió generar los activos: *Plantilla Requisitos* y *Plantilla Análisis y Diseño*.

Los demás activos, fueron el resultado de una experiencia académica al interior de la asignatura *ingeniería de software aplicada* del programa de ingeniería de sistemas de la Universidad de Nariño. Los estudiantes integrados en equipos de trabajo realizaron los activos de software que posteriormente fueron validados. En este caso, algunos de los activos propuestos no fueron aprobados, por no cumplir con características de calidad y de facilidad de reutilización.

6.3.3 Aplicación de Portal Activo para el desarrollo de la disciplina requisitos

6.3.3.1 Búsqueda del activo de software

Inicialmente, el usuario final, hace uso de la búsqueda básica que provee la herramienta (figura 36).

Posteriormente, la herramienta presenta el resultado de la búsqueda (figura 38). En este punto, el usuario elige el activo de software *Plantilla Requisitos.*

6.3.3.2 Recuperación del activo de software

El sistema muestra los metadatos principales que describen el activo seleccionado *Plantilla Requisitos*. En seguida, se descarga el archivo físico que contiene el activo (figuras 39 y 40).

6.3.3.3 Uso del activo de software

En este caso, el activo de software, es una plantilla editable en formato Microsoft Word. Esta plantilla presenta la estructura estandarizada para la presentación de la ingeniería de requisitos en CJT&T que incluye su imagen corporativa. Cada apartado del documento tiene la explicación correspondiente que guía al usuario. Este activo, tiene un activo de proyecto relacionado, que presenta una experiencia de aplicación de la plantilla que el usuario puede emplear como referente.

El resultado final de la aplicación del activo *Plantilla Requisitos* genera el activo de *Requisitos Plan Fidelización* que alimenta el repositorio de la herramienta Portal Activo (BARÓN, y otros, 2011), los metadatos que describen este nuevo activo se presentan en la figuras 42 y 43.

Figura 42. Metadatos del activo de proyecto *Requisitos Plan Fidelización* (1)

Fuente: (BARÓN, y otros, 2011)

Figura 43. Metadatos del activo de proyecto *Requisitos Plan Fidelización* (2)

Fuente: (BARÓN, y otros, 2011)

6.4 RESULTADOS DE LA EXPERIENCIA

Durante la capacitación se realizaron ejercicios que permitieron explorar el proceso de implementación de la práctica de reutilización en la empresa y las funcionalidades que provee Portal Activo. Este trabajo permitió obtener apreciaciones informales que retroalimentaron tareas posteriores de mejoramiento. Sin embargo, al terminar la experiencia se hace necesario conocer las impresiones que tienen usuarios y directivos. A continuación se describen los resultados obtenidos de la experiencia.

6.4.1 Aplicación de encuestas

A fin de conocer las impresiones de los participantes en la experiencia de implementación y con el ánimo de retroalimentar la propuesta, se diseñaron y aplicaron dos encuestas: la primera, dirigida a los directivos participantes de la empresa CJT&T, la segunda, dirigida a usuarios finales. En ambos casos las preguntas se diseñaron de acuerdo a la problemática que motiva esta

investigación: aspectos de gestión y organización, aspectos económicos y aspectos conceptuales y técnicos.

Las encuestas se diseñaron utilizando ítems tipo escala de Likert (BLANCO, y otros, 2003) y preguntas abiertas. Para los ítems, las opciones de respuesta y los valores asignados para efectos de tabulación y análisis se muestran en la tabla 24. Los formatos de estas encuestas, su aplicación y el análisis de resultados completo, se presentan en el anexo B.

Tabla 24. Opciones de respuesta y valores para ítems tipo escala de Likert

Opción	Valor
Totalmente en desacuerdo	1
En desacuerdo	2
Indiferente	3
De acuerdo	4
Totalmente de acuerdo	5

6.4.2 Análisis de información

El análisis de la información recolectada se realiza para cada una de las encuestas debido a que responden a aspectos diferentes de la propuesta.

6.4.2.1 Encuesta a usuarios finales

Este cuestionario se aplicó a los cuatro ingenieros auxiliares, de la empresa CJT&T y a los dos participantes de la Universidad de Nariño. El cuestionario incluye 16 ítems tipo escala de Likert y 2 preguntas abiertas. Las respuestas dadas a los ítems se muestran en la tabla 25 (Se utilizan las siglas I. A. para Ingeniero Auxiliar, E.P. para Estudiante de pregrado y D.I para Docente Investigador).

Tabla 25. Respuestas de usuarios finales

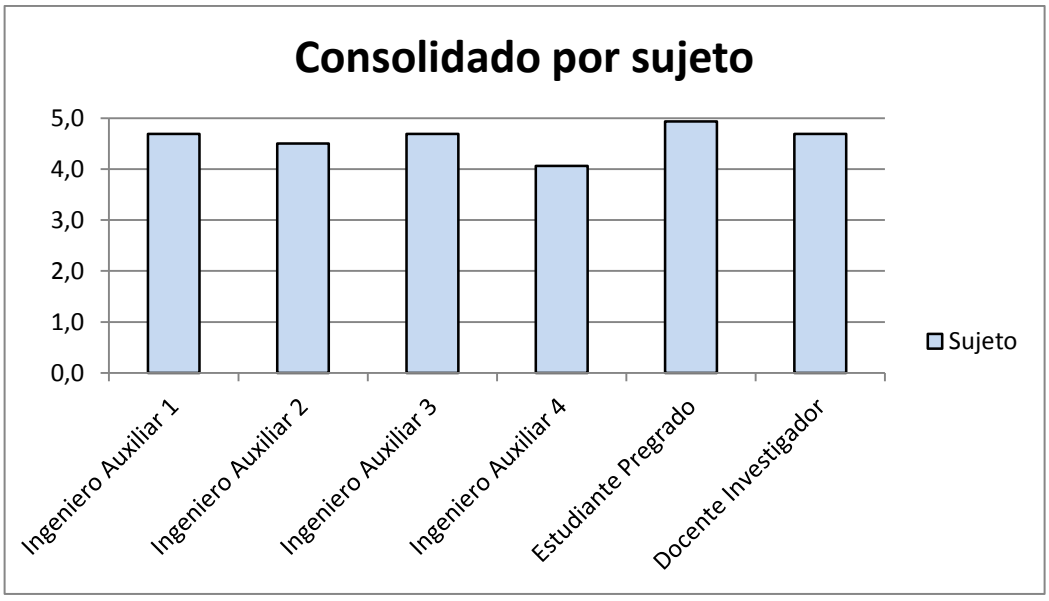
#	Ítem referente a:	I. A. 1	I. A. 2	I. A. 3	I. A. 4	E. P.	D. I.	Promedio por Ítem
1	Apoyo de la alta gerencia	5	4	5	4	5	4	4,5
2	Procedimientos de reutilización	4	4	4	4	5	4	4,2
3	Estándares que regulan la reutilización	4	4	4	4	5	4	4,2
4	Aplicación de la reutilización	4	4	4	4	4	5	4,2
5	Facilidad de Integración de reutilización	4	4	4	4	5	5	4,3
6	Definición unificada de activo de software	4	4	4	4	5	4	4,2

Tabla 25. (Continuación)

#	Ítem referente a:	I. A. 1	I. A. 2	I. A. 3	I. A. 4	E. P.	D. I.	Promedio por Ítem
7	Eficiencia de metadatos DC-Soft	5	5	5	4	5	5	4,8
8	Esfuerzo de aprendizaje de Portal Activo	5	5	5	4	5	4	4,7
9	Esfuerzo de configuración Portal Activo	5	5	5	4	5	5	4,8
10	Uso de Portal Activo en proyectos futuros	5	5	5	4	5	5	4,8
11	Facilidad de integración de Portal Activo	5	5	5	5	5	5	5,0
12	Flexibilidad de Portal Activo	5	5	5	4	5	5	4,8
13	Sugerir Portal Activo a colegas	5	5	5	4	5	5	4,8
14	Gestión de todo tipo de activo	5	5	5	3	5	5	4,7
15	Información para selección de activo	5	4	5	5	5	5	4,8
16	Facilidades de búsqueda	5	4	5	4	5	5	4,7
Promedio por Sujeto		4,7	4,5	4,7	4,1	4,9	4,7	***4,6***

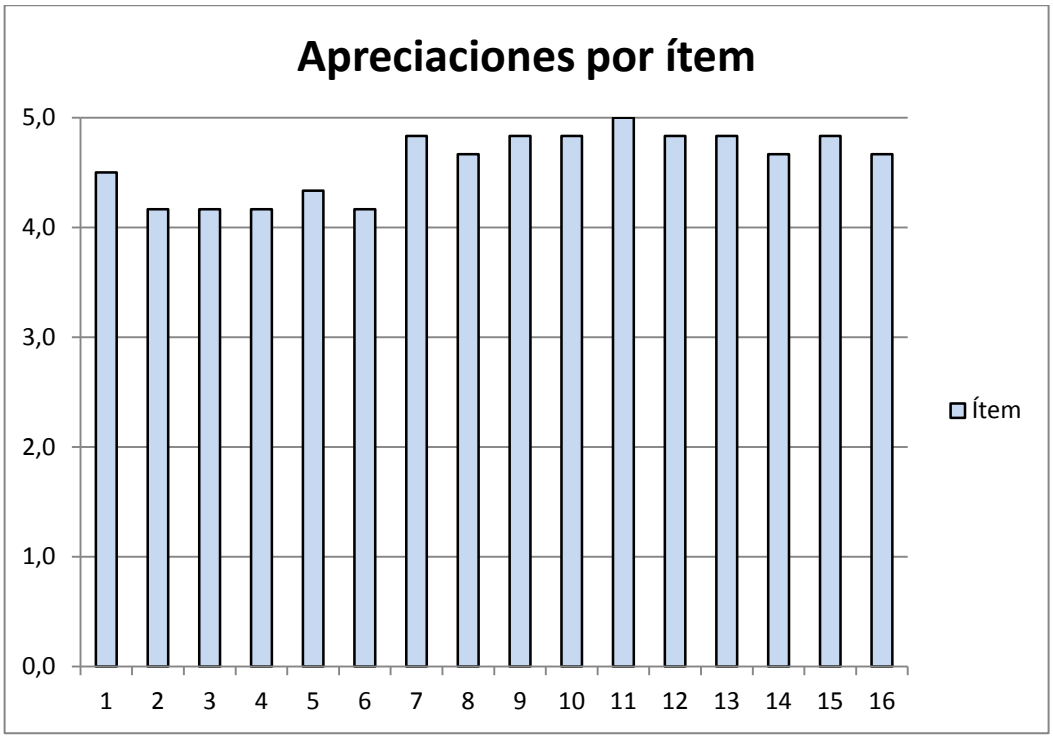
La tabla 25 ha sido construida de tal manera que permita analizar la información obtenida utilizando los criterios *ítem* y *sujeto*. En esta tabla se puede apreciar que las respuestas son homogéneas es decir, no existen valores extremos que desvíen radicalmente el conjunto de apreciaciones.

Figura 44. Apreciaciones consolidadas por sujeto según encuesta a usuarios



En la figura 44, se puede observar que el 100% de los usuarios encuestados tienen una apreciación favorable del proceso de implementación de la práctica de reutilización y de la herramienta Portal Activo. Este aspecto se evidencia también en la tabla 25 que muestra un promedio general de apreciación igual a 4,6 este valor corresponde a una calificación cualitativa entre *de acuerdo* y *totalmente de acuerdo*.

Figura 45. Apreciaciones por ítem según encuesta a usuarios



En la figura 45, se observa que el 100% de los aspectos evaluados reciben una apreciación favorable, esto se evidencia en que todos los ítems obtienen un promedio superior a 4,0. En términos cualitativos este valor se sitúa entre *de acuerdo* y *totalmente de acuerdo*.

En cuanto a la pregunta abierta numerada como 17, se obtiene que: para los usuarios finales, los beneficios que podría traer para su trabajo diario, la adopción de la herramienta portal activo son los siguientes: disminución de tiempo y esfuerzo invertidos en las tareas de desarrollo, incremento de la calidad del producto, incremento de la productividad del equipo de desarrollo, mejora de la gestión de activos de software, mejora del clima organizacional, mejora del proceso software, posible certificación de la organización y aprendizaje de nuevas tendencias de construcción de software.

Igualmente, en cuanto la pregunta abierta numerada como 18, los usuarios finales, sugieren las siguientes mejoras a la herramienta Portal Activo para facilitar su uso por parte de los desarrolladores: ofrecer vistas previas de los activos de software, ofrecer consultas vía dispositivos móviles.

6.4.2.2 Encuesta a directivos

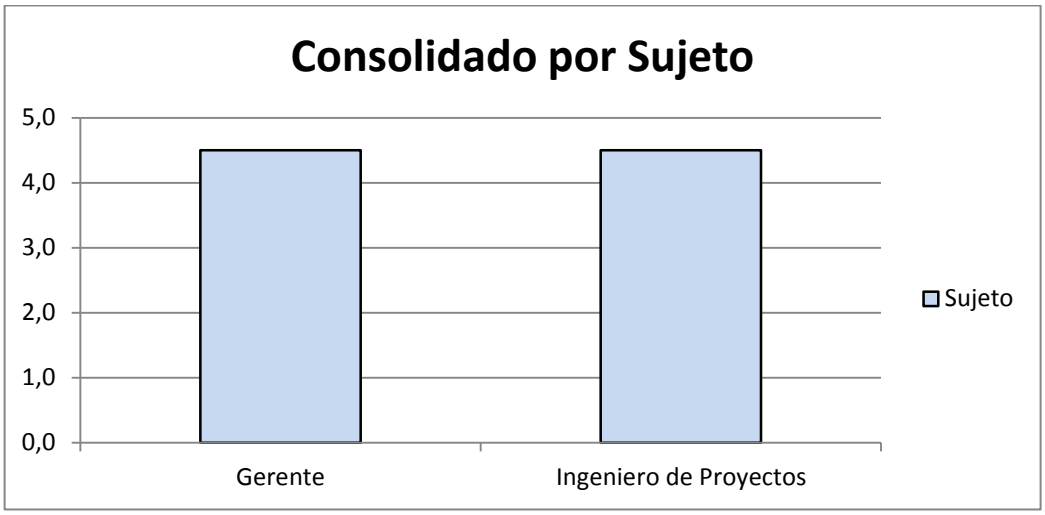
Este cuestionario se aplicó al Gerente y al director de proyectos de la empresa CJT&T. El cuestionario incluye 10 ítems tipo escala de Likert y 1 pregunta abierta. Las respuestas dadas a los ítems se muestran en la tabla 26.

Tabla 26. Respuestas de directivos

#	**Ítem referente a:**	Gerente	Ingeniero de proyectos	**Promedio por Ítem**
1	Apoyo de la alta gerencia	5	5	4,5
2	Estructura organizativa de la empresa	4	5	4,0
3	Inversión y beneficios de la reutilización	4	4	4,0
4	Procedimientos de reutilización	4	4	4,0
5	Estándares que regulan la reutilización	4	4	4,0
6	Aplicación de la reutilización	4	4	4,0
7	Facilidad de Integración de reutilización y Portal Activo con el proceso	5	4	4,5
8	Portal Activo como estrategia organizacional	5	5	5,0
9	Aceptación de Portal Activo	5	5	5,0
10	Uso de Portal Activo en proyectos futuros	5	5	5,0
Promedio por Sujeto		4,5	4,5	***4,5***

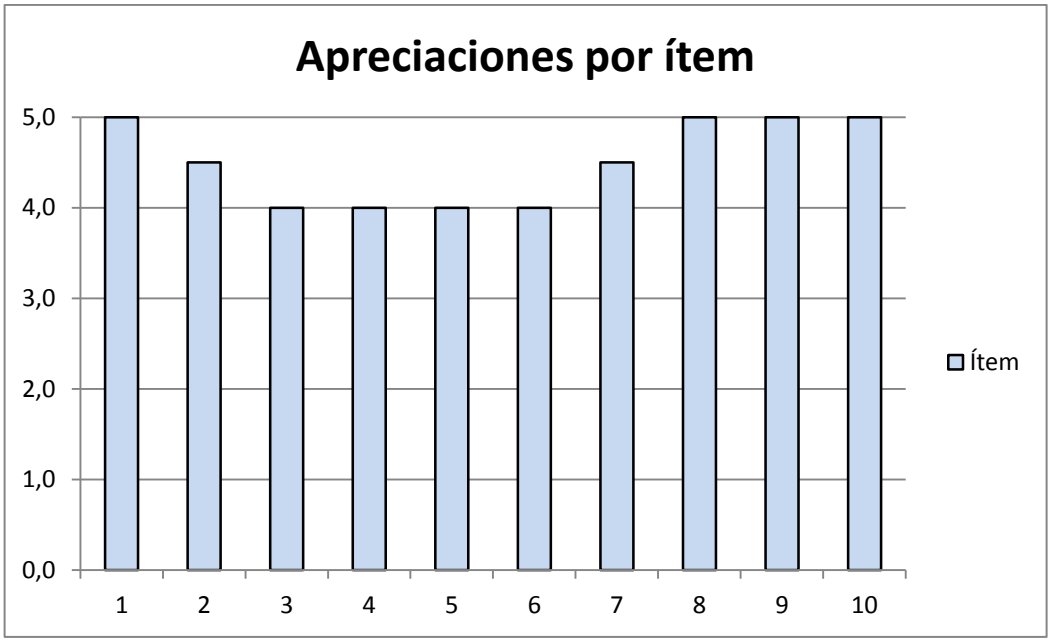
La tabla 26 ha sido construida de tal manera que permita analizar la información obtenida utilizando los criterios *ítem* y *sujeto*. En esta tabla se puede apreciar que las respuestas son homogéneas es decir, no existen valores extremos que desvíen radicalmente el conjunto de apreciaciones.

Figura 46. Apreciaciones por sujeto según encuesta a directivos



En la figura 46, se puede observar que el 100% de los directivos encuestados tienen una apreciación favorable del proceso de implementación de la práctica de reutilización y de la herramienta Portal Activo. Este aspecto se evidencia también en la tabla 26 que muestra un promedio general de apreciación igual a 4,5 este valor corresponde a una calificación cualitativa entre *de acuerdo* y *totalmente de acuerdo*.

Figura 47. Apreciaciones por ítem según encuesta a directivos



En la figura 47, se observa que el 100% de los 10 ítems evaluados reciben una apreciación favorable, esto se evidencia en que todos los ítems obtienen un promedio igual o superior a 4,0. En términos cualitativos este valor se sitúa entre *de acuerdo* y *totalmente de acuerdo*.

En cuanto a la pregunta abierta numerada como 11, se obtiene que: para los directivos, los beneficios que esperaría de adoptar la práctica de la reutilización y de la herramienta Portal Activo, en la organización son los siguientes: disminución de tiempo y esfuerzo invertidos en las tareas de desarrollo, incremento de la calidad del producto, incremento de la productividad del equipo de desarrollo, mejora de la gestión de activos de software, mejora del clima organizacional, mejora del proceso software, incrementar la rentabilidad de los proyectos y mejora en la estimación de proyectos.

6.4.2.3 Apreciaciones generales

En la figura 48, se integran las apreciaciones obtenidas en las dos encuestas y se aprecia que el resultado es favorable en un 100%, todos los participantes del proceso de implementación de la practica de reutilización y del uso de la herramienta Portal Activo, coinciden en asignar calificaciones que superan el 4.0, este valor corresponde a una calificación cualitativa entre *de acuerdo* y *totalmente de acuerdo*.

Figura 48. Apreciaciones por sujeto consolidado (usuarios - directivos)

7. CONCLUSIONES Y TRABAJOS FUTUROS

Se ha presentado en este trabajo una propuesta para la gestión de activos de software denominada *PEGASO*, la cual hace uso del principio de reutilización sistemática en las fases del proceso software independientemente de la metodología utilizada.

Se ha realizado una caracterización conceptual de los activos de software a partir de la visión general de activo de conocimiento, dado que el activo de software representa el capital intelectual de mayor valor en una empresa software. Asimismo de ha definido un catálogo de tipos de activos de software potencialmente reutilizables que pueden ser matriculados en la biblioteca de activos, tomando como referencia los artefactos definidos en algunas de las aproximaciones de desarrollo de software.

El soporte de la tecnología de repositorios digitales fue una excelente alternativa cómo mecanismo para gestionar activos de software. Para lograr esta articulación fue necesario estudiar, probar y comparar herramientas de gestión de repositorios a fin de seleccionar la mejor alternativa. La herramienta elegida fue adatada para gestionar activos de software. Este proceso representó un alto esfuerzo en este trabajo.

Se propuso una extensión del estándar Dublin Core denominado DC-Soft como mecanismo de descripción de los activos de software. DC-Soft describe cualquier tipo de activo de software, permite adoptar cualquier proceso software, incluido el propuesto por el estándar internacional IEEE1517 y es configurable en herramientas de gestión de repositorios. Su uso garantiza la portabilidad de la información gracias a la herencia de la estructura del estándar internacional de metadatos Dublin Core.

Se propuso la guía de adopción Portal Activo (herramienta que implementa la propuesta), como mecanismo para asistir los procesos de gestión de activos de software. La guía facilita la implementación de la reutilización sistemática en las organizaciones presentando a manera de procesos las funcionalidades que provee Portal Activo

La propuesta fue validada en una compañía de software. La experiencia permitió demostrar la facilidad de adopción de la propuesta por medio del uso de la herramienta Portal Activo. Se realizaron los procesos de Capacitación del equipo de trabajo y la configuración y aplicación de la herramienta. Aunque no se mostró el potencial real de la herramienta por las particularidades de la empresa y del proyecto, el nivel de aceptación fue adecuado, de acuerdo con el análisis de la información recolectada por medio de encuestas aplicadas al equipo participante.

Como posibles trabajos futuros, se encuentran los siguientes:

Realizar un estudio empírico, tanto de tipo cualitativo como cuantitativo, de manera que permita evaluar formalmente los beneficios y retos del uso de la herramienta en otras organizaciones, considerando escenarios de heterogeneidad cultural y con diferentes metodologías asociadas al proceso software.

Construcción de un módulo que permita la integración de este sistema con un servidor de gestión de configuración de manera que los artefactos de software existentes puedan ser matriculados en el repositorio de manera masiva.

Construcción de un modulo interface que permita la integración de activos distribuidos en la web en diversos directorios, aprovechando la potencialidad inherente a los repositorios digitales.

Construcción de un asistente para configuración procesos software de tal manera que la herramienta permita fácilmente incorporar el proceso software de la organización en la catalogación de los activos. Actualmente es posible realizar esta operación modificando el script de la base de datos, lo que dificulta su realización.

Realización de búsqueda de conocimiento en la base de datos de bitácora de uso de PORTAL ACTIVO, con el fin de detectar patrones de comportamiento en materia de construcción de software basado en reutilización. Con técnicas de minería de datos, es posible descubrir tales patrones de uso a fin de identificar nuevos requisitos para incorporarlos en futuras versiones de la propuesta.

BIBLIOGRAFÍA Y REFERENCIAS

AMBLER, Scott. 2005. *A Manager's Introduction to The Rational Unified Process (RUP), Ambysoft.* Toronto : s.n., 2005.

—. 2005. The Agile Unified Process (AUP). [En línea] 2005. [Citado el: 4 de Noviembre de 2011.] http://www.ambysoft.com/unifiedprocess/agileUP.html.

—. 2003. The Enterprise Unified Process (EUP). [En línea] 2003. [Citado el: 4 de Noviembre de 2011.] http://www.enterpriseunifiedprocess.com.

ANAYA, Raquel, CECHICH, Alejandra y HENAO, Mónica. 2008. *A model to Classify Knowledge Assets of a Process-Oriented Development.* Medellín : Departamento de Informatica y Sistemas, Universidad EAFIT, 2008.

ANAYA, Raquel, y otros. 2006. *Enfoque Integrado de la Gestion del Conocimiento en el modelo de procesos de Competisoft.* 2006.

BARÓN, Alexander y ANAYA, Raquel. 2011. Portal Activo. [En línea] 01 de 08 de 2011. [Citado el: 01 de 02 de 2012.] http://activo.udenar.edu.co/.

BLANCO, Angeles, UROZA, Belén y MORALES, Pedro. 2003. *Construcciones de escalas de actitudes tipo Likert.* Madrid : La Muralla, 2003.

CARRERON, Maria. 2008. *Construcción de un catálogo de patrones de requisitos funcionales para ERP.* s.l. : Universitat Politécnica de Cataluña, 2008.

CJT&T Ingeniería de Software. 2007. CJT&T. [En línea] 2007. [Citado el: 5 de Septiembre de 2011.] http://www.cjtytsoftware.com/.

COHEN, Sholom. 1992. *Application of Feature-Oriented Domain Analysis to the Army Movement Control Domain.* Pittsburgh, P.A : s.n., 1992.

CRNKOVIC, Ivica. y LARSSON, Magnus . 2002. *Building reliable component-based software systems.* Norwood (MA) : Artech House Computing Library, 2002.

DATA & OBJECT FACTORY. 2001. dofactory#1 in design patterns. *dofactory.* [En línea] 2001. [Citado el: 14 de Septiembre de 2011.] http://www.dofactory.com.

DINGSOYR, Torgeir. 2003. *Knowledge Management in Medium-Sized Software Consulting Company, An Investigation of Intranet-based Knowledge management Tools for Knowledge Cartography and Knowledge Repositories for Learning Software Organizations.* Norway : Norwegian University of Science and Technology, 2003.

ERDOGMUS, Hakan. 2002. *A Real Options Perspective of Software Reuse.* Austin Texas : International Workshop on Reuse Economics, 2002.

FEDORA. 2011. Islandora and Discovery Garden, Dspace. [En línea] 2011. [Citado el: 18 de Septiembre de 2011.] https://wiki.duraspace.org/display/DSPACE/Home.

—. 2005. Islandora and Discovery Garden, Fedora Common. [En línea] 2005. [Citado el: 18 de Septiembre de 2011.] http://fedora-commons.org.

FEI, Cao, y otros. 2003. *Automating feature-Oriented Domain Analysis.* 2003.

FRAKES, William y KANG, Kyo. 2005. *Software Reuse Research: Status and Future,.* s.l. : Ieee transactions on software engineering, 2005. Vol. 31.

FUENTES, Lidia., JIMENEZ, Daniel. y PINTO, Monica. 2004. *Hacia un entorno de desarrollo integrado basado en Componentes y Aspectos.* Malaga : Departamento de Lenguajes y Ciencias para la Computacion , ETSI Informática Campus de teatinos s/n., 2004.

HENNINGER, Scott. 1999. *An Evolutionary Approach to Constructing Effective Software Reuse Repositories.* 1999.

IBM. 2007. RUP for Asset-Based Development V3.0 and Asset-Based Development Governance Plug-in V1.0, IBM Developer Works. [En línea] 2007. [Citado el: 23 de Octubre de 2011.] http://www.ibm.com/developerworks/rational/downloads/07/rup_abd_gov.

IEEE COMPUTER SOCIETY. 2010. *IEEE Standard for Information Technology – System and Software Life Cycle Processes – Reuse Processes, IEEE Std 1517™.* New York, USA : Society IEEE Computer, 2010.

INFRAGISTICS, INC. QUINCE. 2009. Quince. *quince.* [En línea] Quince, 2009. [Citado el: 14 de Septiembre de 2011.] http://quince.infragistics.com.

JARZABEK, Stanislaw. 2007. *Effective Software Maintenance and Evolution - A Reuse - Based Approach.* s.l. : Auerbach Publications, 2007.

JASMINE, K. 2010. *A New Capability Maturity Model For Reuse Based Software Development process.* 2010. Vol. 2.

JHA, Meena, O´BRIEN, Liam y MAHESHWARI, Piyush. 2006. *Identify Issues and Concerns in Software Reuse.* 2006.

KRUCHTEN, Philippe. 2004. *The Rational Unified Process 3rd Edition: An Introduction. Reading.* MA : Addison-Wesley Longman, Inc., 2004.

KRUEGER, Charles W. 1992. *Software Reuse.* s.l. : ACM Association for Computing Machinery, 1992. Vol. 24.

LARMAN, Craig. 2002. *UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Prentice Hall, 2002.

LÓPEZ, Oscar, LAGUNA, Miguel Angel y MARQUÉS, José Manuel. 2001. *Reutilización del Software a partir de Requisitos Funcionales en el Modelo de Mecano: Comparación de Escenarios.* s.l. : Universidad de Valladolid, 2001.

MENTZAS, Gregoris, y otros. 2003. *Knowledge Asset Management: Beyond the Process-centred and Product-centred Approaches.* s.l. : Springer, 2003.

PARNAS, David. 1994. *Software Reuse and Component Based Software Engineering.* Italy : s.n., 1994. págs. 1-22.

POLE, Thomas. 2007. Software-Engineering Asset Management, Microsoft Developer Network (Skycrapr). [En línea] 2007. http://msdn.microsoft.com/en-us/library/bb421530.aspx.

SAMETINGER. 2010. *Software Engineering with Reusable Components.* Berlin : Springer-Verlag, 2010.

SOMMERVILLE, Ian. 2007. *Software Engineering.* 8th. United Kingdom : Pearsons Education, 2007.

THE DUBLIN CORE® METADATA INITIATIVE, DCMI. 1995. Dublin Core. [En línea] 1995. [Citado el: 17 de Octubre de 2011.] http://www.dublincore.org.

UNIVERSITY OF NOTTINGHAM, U.K. 2006. OPENDOAR. [En línea] 2006. [Citado el: 15 de Marzo de 2012.] http://www.opendoar.org/index.html.

UNIVERSITY OF SOUTHAMPTON, U.K. 2011. E-prints. [En línea] 2011. [Citado el: 18 de Septiembre de 2011.] http://www.eprints.org.

WELICKI, Leon. 2006. *Meta-Especificacion y Catalogacion de Patrones de Software con Lenguajes de Dominio Especifico y Modelos de Objetos Adaptativos: Una via para la Gestion del Conocimiento en la Ingenieria del Software.Tesis Doctoral.* s.l. : Universidad Pontificia de Salamanca, campus Madrid., 2006.

WELIE, Martin Van. 2008. Welie.com-Patterns in interaction Desing. [En línea] 2008. [Citado el: 14 de Septiembre de 2011.] http://www.welie.com.

WHITEHEAD, Derek. 2005. *Repositories: What is the Target? An Arrow Perspective', New Review of Information Networking, .* 2005. págs. 123-134.

WINDOWSCLIENT.NET COMUNITY. 2011. Microsoft WindowsClient.NET-windows forms-Windows Presentation Foundation. [En línea] 2011. [Citado el: 10 de Marzo de 2011.] http://www.windowsclient.net.

WOLFGANG, Steindl. 2009. 'Jazz and Beyond', IBM Rational Group – Rational Software, IBM Corporation,. [En línea] 2009. http://www-05.ibm.com/at/events/software_experience/pdf/24062009/jazz_and_beyond.pdf.

YOUNG, Ronald y MENTZAS, Gregoris. 2001. *Strategy, processes and systems for leveraging corporate knowledge.* 2001. Vol. 4.

ZAPATA, Carlos, HERNÁNDEZ, Juan y ZULUAGA, Raúl. 2008. *UNC-Corpus Corpus de diagramas UML para la solución de problemas de completitud en Ingeniería de Software.* s.l. : Universidad EAFIT, 2008. Vol. 44.