Propuesta arquitectónica para el núcleo del Gestor de Documentos Administrativos eXcriba 3.0

Trabajo de Diploma para optar por el título de Ingeniero

en Ciencias Informáticas

TRABAJO DE DIPLOMA

Universidad de las Ciencias Informáticas

La Habana, 2013

Propuesta arquitectónica para el núcleo del Gestor de Documentos Administrativos eXcriba 3.0

Trabajo de Diploma para optar por el título de Ingeniero

en Ciencias Informáticas

Autor: Oscar Daniel Torres Hernández.

Tutores: Ing. Pedro Rodriguez Samon.

Ing. Michel David Suárez.

Universidad de las Ciencias Informáticas

La Habana, 2013

Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad.

Albert Einstein (1879–1955).

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informatización Universitaria de la Universidad de las Ciencias Informáticas para que le den el uso que estimen necesario a este trabajo. Para que así conste firmo la presente a los ____ días del mes de _____del año _____.

Pedro Rodriguez Samon

_____

Firma del Tutor

Michel David Suárez

_____

Firma del Tutor

Oscar Daniel Torres Hernández

_____

Firma del Autor

Agradecimientos

A mi familia *por ser mi guía*
Arianne *por todo el apoyo y su ayuda*
Yanet *por su gran ayuda*
Michel *por los sermones y su apoyo*
Johan *por su gran amistad en todo momento*
Yaima Oval *por su gran ayuda*
Marcel *por sus comidas deliciosas y su apoyo*
Reinier *por su ayuda y apoyo*
Pedro *por su ayuda*
Laritza *por su ayuda*

A todos los que de una manera u otra ayudaron a sacar adelante este proyecto.
Gracias.

Dedicatoria

A mi familia...

Resumen

El presente trabajo emprende una investigación centrada en las arquitecturas de *software* para Gestores de Contenido Empresarial. Se aborda un estudio detallado de la disciplina, definiendo un marco teórico que abarca las áreas que comprenden a los Gestores de Contenido, así como sus principales elementos.

Se describe además la propuesta arquitectónica para el núcleo del Gestor de Documentos Administrativos eXcriba mediante la documentación de sus vistas. Esta propuesta tiene como principal objetivo elevar la habilidad de realizar cambios futuros al sistema para dar solución a los problemas que presenta el núcleo del eXcriba actual.

La investigación concluye con la evaluación de la propuesta mediante el método de evaluación MECABIC. Es un trabajo que pretende servir de guía para posteriores propuestas arquitectónicas en Gestores de Contenido Empresarial.

Palabras clave: Gestor de Contenido Empresarial, Arquitectura de Software, Gestor de Documentos Administrativos.

Índice

Índice de Tablas

Índice de Figuras

Introducción

El crecimiento del universo digital, más que un simple factor que afecta individualmente, se ha convertido en una explosión de la información a nivel global. Para las organizaciones este fenómeno ha tenido implicaciones directas en su gestión del contenido (EMC, 2011). De esta manera, los afectados buscan la aplicación de sofisticadas técnicas para el manejo, búsqueda, almacenamiento y seguridad de toda su información. Con los beneficios de los avances tecnológicos, surgen soluciones para atender estos problemas en las organizaciones, sin importar el tamaño de las mismas.

Un Gestor de Contenido Empresarial (ECM por sus siglas en inglés) es el sistema encargado de atender todo el ciclo de vida de los contenidos de una organización, desde su creación hasta su eliminación. Esta solución se crea para mitigar los problemas que se pueden presentar con el crecimiento de la información no gestionada (MUNKVOLD, et al., 2005; NORDHEIM, et al., 2006). Su uso brinda beneficios en la organización, centralización y disponibilidad de la información, permitiendo además la rápida búsqueda y recuperación de los contenidos. Por ese motivo el desarrollo de estos sistemas está enfocado en las crecientes necesidades de sus usuarios.

Los ECM están compuestos por un conjunto de soluciones, cada una enfocada a un campo específico. Sus arquitecturas no solo brindan la posibilidad de establecer un Sistema de Gestión de Contenido Empresarial en una organización; algunos ECM como Alfresco, posibilitan extender sus funcionalidades e integrarse con aplicaciones o sitios *web* existentes. Estas bondades permiten un mayor acercamiento a situaciones específicas del negocio, sin que ello niegue el uso de las distintas herramientas con las que cuenta la empresa.

En la literatura internacional consultada, acerca de los Gestores de Contenido Empresarial se destacan autores como: Potts, J., 2009; Newton, J., 2010; Gartner, 2011; Forrester, 2011; Bergljung, M., 2011; Newton, J. 2012; Gartner, 2012. Estos detallan las principales características de los ECM y especifican cómo se puede proponer una solución mediante el uso o extensión de un Gestor de Contenido Empresarial. Por otro lado, no especifican cómo proponer una solución que utilice solo los elementos que necesite de un ECM ni que plantean una vía para poder renovar sus componentes.

La Universidad de las Ciencias Informáticas, previsora ante los problemas generados por el crecimiento de la información no gestionada; ha desarrollado el Gestor de Documentos Administrativos eXcriba con el propósito de automatizar los procesos de gestión documental en una organización. El sistema utiliza los servicios que provee el ECM Alfresco en su versión estable comunitaria 3.0. Para la integración de eXcriba con el mencionado ECM, se hizo necesario realizar modificaciones en varias de las áreas del código fuente de Alfresco; con el objetivo de cubrir todas las necesidades del GDA.

Los cambios realizados provocaron que el sistema quedara dependiente a la versión 3.0 del ECM, que en estos momentos está desactualizada. Por consiguiente, cuando se han publicado nuevas versiones de Alfresco, ha resultado costoso para el equipo de desarrollo de eXcriba poder renovar los componentes y corregir los errores detectados; lo que ha obligado a aplicar reparaciones complicadas periódicamente sobre el sistema.

Los continuos cambios sobre un sistema pueden dañar la arquitectura, afectando en gran medida su complejidad. Se puede afirmar que todas las reparaciones tienden a destruir la estructura, incrementar la entropía[1] y el trastorno en los sistemas, además de gastar luego más tiempo en arreglar los defectos introducidos en las correcciones anteriores (BROOKS, 1975; BYRON J., et al., 2009).

Sobre la base de las consideraciones anteriores se propone como **problema de investigación:** La dependencia del núcleo del Gestor de Documentos Administrativos eXcriba a la versión 3.0 del Alfresco provoca la obsolescencia del sistema y hace más complejo su mantenimiento.

El **objeto de estudio** en el que está centrada la presente investigación es: Las Arquitecturas de Software para aplicaciones empresariales**.**

Preguntas de investigación

1. ¿Cuáles son las características principales que debe tener una arquitectura de *software* en un ECM?

2. ¿Cuáles son las características que debe poseer la arquitectura del núcleo del GDA eXcriba?

[1] La Entropía se concibe como una “medida del desorden o medida de la incertidumbre”.

3. ¿Cuáles son las pautas necesarias para evaluar una arquitectura mediante el uso de un método de evaluación?

Objetivo:

Diseñar una arquitectura para el núcleo del GDA eXcriba que le permita renovar sus componentes, mediante un nuevo diseño de su Gestor de Contenido Empresarial.

Objetivos Específicos:

1. Caracterizar las áreas que componen a los Gestores de Contenido Empresarial, así como los conceptos teóricos de la arquitectura de *software* en las aplicaciones empresariales, para delimitar el marco teórico de la investigación.

2. Definir la propuesta de la arquitectura para el núcleo del GDA eXcriba que sostenga los elementos teóricos y prácticos, a partir de un prototipo funcional.

3. Validar el prototipo funcional ejecutable de la arquitectura del núcleo del GDA eXcriba, mediante el uso de un método de evaluación de arquitectura.

Campo de acción: Las Arquitecturas de Software en los Gestores de Contenido Empresarial.

Métodos científicos:

De los métodos teóricos (o lógicos) se aplicaron:

1. **El Análisis Histórico-Lógico**, el cual permitió determinar el desarrollo y la evolución hasta el momento de las Arquitecturas de Software en los Gestores de Contenido Empresarial, además de poder determinar las tendencias actuales en el *software***.**
2. **El Analítico-Sintético**, el cual facilita identificar, analizar y seleccionar las definiciones y los conceptos más importantes relacionados con la presente investigación.
3. **El Modelado**, fue utilizado para representar los prototipos del sistema que constituyen posibles soluciones de la arquitectura para el núcleo del GDA eXcriba.
4. **La Inducción-Deducción** que ayudó a analizar las características del comportamiento de los Gestores de Contenido Empresarial y poder luego concluir sobre casos específicos que pueden ser verificados en la práctica.

   De los métodos empíricos se aplicaron:

1. **La Revisión de Documentos** para ampliar los conocimientos relacionados con los Gestores de Contenido Empresarial, así como su funcionamiento.
2. **La Observación** permitió realizar un seguimiento en el comportamiento de las tendencias de los Gestores de Contenido Empresarial.

Justificación de la investigación

Diseñar una arquitectura para el núcleo del Gestor de Documentos Administrativos eXcriba, que permita su actualización con las nuevas versiones que se publiquen de Alfresco y sin que se cree dependencia sobre los componentes del ECM, posibilitará la detección y corrección de errores; así como la incorporación de nuevas funcionalidades en un menor período de tiempo y sin incurrir en elevados costos de mantenimiento.

Estructura de los capítulos

Para cumplir los objetivos específicos se propone la siguiente estructura para el trabajo de tesis de la presente investigación:

Capítulo 1: Fundamentación teórica.

Se exponen las tendencias y el estado de desarrollo actual de las Arquitecturas de Software en los Gestores de Contenido Empresarial, además se describen los componentes y herramientas necesarios para realizar una propuesta arquitectónica para el Gestor de Documentos Administrativos eXcriba.

Capítulo 2: Propuesta de la arquitectura.

Se propone la arquitectura candidata para el núcleo del Gestor de Documentos Administrativos eXcriba mediante la descripción de sus vistas.

Capítulo 3: Validación de la arquitectura del núcleo del Gestor de Documentos Administrativos eXcriba.

Se evalúa la propuesta arquitectónica para el núcleo del Gestor de Documentos Administrativos eXcriba mediante el uso del método MECABIC.

Capítulo 1
FUNDAMENTACIÓN TEÓRICA

El objetivo que se persigue en el presente capítulo es establecer las tendencias y el estado de desarrollo actual de las Arquitecturas de Software en los Gestores de Contenido Empresarial, así como desglosar los componentes y herramientas necesarios para realizar una propuesta arquitectónica para el Gestor de Documentos Administrativos eXcriba.

La mayoría de las empresas generan una elevada cantidad de información que puede ser almacenada en documentos o archivos. Dicha situación provoca una necesidad de gestión coherente de la información digital. Se puede afirmar que los ECM han sido la respuesta a esta necesidad de las organizaciones (MUNKVOLD, et al., 2005; NORDHEIM, et al., 2006).

1.1 Gestor de Contenido Empresarial

Según la Asociación para la Información y Gestión de Imagen (AIIM por sus siglas en inglés) los Gestores de Contenido Empresarial componen las estrategias, métodos y herramientas que se utilizan para capturar, administrar, almacenar, conservar y entregar contenidos y documentos relacionados con procesos organizativos. Las estrategias y herramientas de los ECM permiten gestionar toda la información no estructurada [2]de una organización, en la medida en que esta se desarrolla (ASSOCIATION FOR INFORMATION AND IMAGE MANAGEMENT, 2011).

Los sistemas ECM utilizan un repositorio, diferentes aplicaciones y plataformas de desarrollo de aplicaciones para permitir el control, acceso y distribución de los contenidos. Cualquier información no estructurada, como documentos, páginas *web*, imágenes, videos, registros o archivos simples puede ser manejada. De esta manera, se puede citar a los Gestores de Contenido Empresarial mediante el uso de dos conceptos (GARTNER, 2012):

[2]La información no estructurada se refiere a la información computarizada que no tiene un modelo de datos.

1. Estrategia para lidiar con cualquier tipo de contenido empresarial.
2. El *software* que gestiona todo el ciclo de vida de un contenido.

Estas soluciones informáticas combinan diferentes tecnologías, y para su selección es de vital importancia su modelo de implementación, el conocimiento de las tendencias en el mercado y la competencia.

Según Gartner los principales componentes del núcleo de un ECM son (GARTNER, 2012):

1. Gestor de Documentos (DM por sus siglas en inglés).
2. Aplicaciones de Procesamiento de Imágenes.
3. Flujos de Trabajo y/o Gestor de Procesos de Negocios (BPM por sus siglas en inglés).
4. Administración de Documentos de Archivo (RM por sus siglas en inglés).
5. Gestor de Contenidos *web* (WCM por sus siglas en inglés).
6. Contenido Social.
7. Componentes Extendidos.

Estos componentes generalmente se encuentran organizados en los ECM de la siguiente manera:

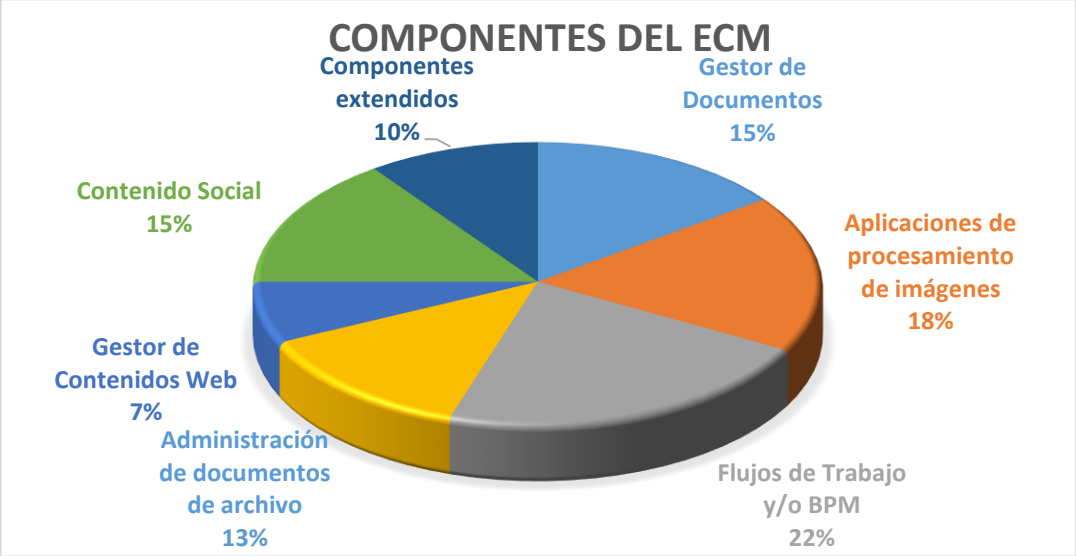

Figura 1: Componentes del ECM

1.1.1 Gestor de Documentos

Según la AIIM, la gestión de documentos o los sistemas de gestión de documentos (DM y DMS por sus siglas en inglés), son el uso de un sistema informático o *software* para almacenar, gestionar

y controlar los documentos e imágenes en formato electrónico que han sido capturados a través del uso de un escáner (ASSOCIATION FOR INFORMATION AND IMAGE MANAGEMENT, 2010). Todo ECM debe centrar gran parte de su atención en poseer un repositorio para la gestión de documentos con los servicios básicos. Esto permite crear una base sólida, sobre la cual podrán descansar luego otros importantes componentes del ECM; se puede tomar como ejemplo la Gestión de Contenidos *web*. Otras de las razones por la cual esta área es importante es la existencia de un enorme mercado para los sistemas que pueden gestionar contenidos no estructurados. Esta necesidad crece en gran medida y su evolución ha despertado preocupaciones, pues la gestión documental es un problema global (POTTS, 2009). Todas las organizaciones que generan documentos pueden ser beneficiadas por el manejo, réplicas, versionado, inclusión de *metadata*[3], seguridad, búsquedas en todo el texto y flujos de trabajo que brinda un gestor de documentos.

Las principales características que deben poseer los sistemas de gestión documental, se pueden desglosar de la siguiente manera (POTTS, 2009):

1. **Escalable y confiable**.

   Estos sistemas necesitan ser capaces de mantener interacción con una gran cantidad de usuarios, además de poder manejar una inmensa cantidad de documentos con un índice de crecimiento diario; propiedad de vital importancia, pues muchos de los contenidos manejados son de gran relevancia y es una prioridad que estén altamente disponibles.

2. **Seguridad.**

   Muchos de los contenidos manejados por la organización pueden ser sensibles y no todo el personal vinculado puede tener contacto con todos los documentos. Por esta razón se debe interactuar con el sistema mediante el uso de mecanismos de autenticación, para que cada miembro maneje solamente los documentos necesarios.

3. **Sencilla integración con otras herramientas de trabajo.**

   Los usuarios necesitan interactuar con el sistema frecuentemente, por lo que el manejo de los documentos hacia dentro y fuera del repositorio debe ser una acción sencilla: se deben abrir, copiar, mover y salvar documentos desde herramientas ofimáticas, exploradores de carpetas y clientes de correo.

[3] Datos que describen a otros datos: autores, títulos, casas editoriales, etc.

4. **Servicios.**

   Esta área está compuesta por un grupo de funcionalidades para el movimiento de la información, su versionado, uso de *metadata* y búsquedas. Estos servicios mejoran el tratamiento de la información con el manejo de un sistema de ficheros en este tipo de repositorio.

Esta área representa la base del GDA eXcriba, ya que es la encargada de almacenar, manejar y controlar todos los documentos administrativos que se desenvuelven en su ámbito. Por esta razón se incluye como primer elemento en la propuesta del núcleo de la presente investigación.

1.1.2 Aplicaciones de procesamiento de imágenes

Las aplicaciones de procesamiento de imágenes dentro de los ECM, son las encargadas de capturar, transformar y manejar información de los documentos en formato físico. Para cumplir su objetivo estas aplicaciones se organizan en dos grupos, siendo el fin de estos:

1. Brindar una adecuada captura de documentos, mediante dispositivos como el escáner y tecnologías para la extracción de la información que incluyan reconocimiento de caracteres.
2. La posibilidad de incluir esta información extraída en el repositorio como otro tipo de contenido dentro de una carpeta y que pueda ser parte de un flujo de trabajo o un proceso de negocio.

El GDA eXcriba en estos momentos no contiene servicios de captura, transformación o manejo de información de documentos en formato físico, por lo que esta área no se incluye en la propuesta del núcleo.

1.1.3 Flujos de trabajo o Gestor de procesos de negocios

Los flujos de trabajo o los gestores de procesos de negocios permiten automatizar los procesos de negocios donde se encuentren vinculados los documentos; dejando que el sistema de gestión se encargue de ver a quién entregar qué documento y en qué momento. También son utilizados para el apoyo en los procesos de trabajo que existen en una organización, específicamente para la designación de tareas. Además, es posible realizar un fácil seguimiento del estado y progreso del movimiento del contenido y crear nuevos espacios para las auditorías. Se puede tener un flujo de trabajo sencillo al asignarle la revisión de un documento a una o varias personas y se puede complicar tanto como cualquier proceso del negocio lo necesite.

Por el nivel de importancia que esta área le añade al proceso del manejo y seguimiento de la información y el contenido, es esencial su presencia en la propuesta realizada en esta investigación.

1.1.4 Administración de Documentos de Archivo

Según la Norma ISO 15489 la Administración de Documentos de Archivo es el campo responsable del control sistemático y eficiente de la creación, recepción, mantenimiento, uso y disposición de los documentos de archivo. Incluyendo además los procesos para la captura y mantenimiento de los datos e información sobre las actividades del negocio, además de sus transacciones (ISO, 2005). AIIM amplía esta definición para abarcar a todos los archivos mantenidos en formato electrónico (ASSOCIATION FOR INFORMATION AND IMAGE MANAGEMENT, 2009). Se puede definir como el control centralizado de los activos de información de las organizaciones. Utilizado mayormente para la retención a largo plazo de los contenidos a través de la automatización y las políticas, que garantizan el cumplimiento legal y normativo de la industria.

Al establecer y ejecutar una administración de este tipo se establecen requisitos, responsabilidades y rendiciones de cuenta en la gestión de los activos de información. La necesidad de la aplicación de políticas y rendiciones de cuenta es cada vez más evidente para los ejecutivos y directivos; pues a medida que se genera más información en formato electrónico, crece el riesgo de incumplimiento y pérdida.

El requisito mínimo para un sistema de Administración de Documentos de Archivo es la capacidad de hacer cumplir la retención de documentos críticos del negocio. La calidad se mide por el cumplimiento de las normas de certificación, otorgadas por el Departamento de Defensa de los Estados Unidos de América (DoD por sus siglas en inglés) directiva 5015.2-STD[4], Archivo Nacional (TNA por sus siglas en inglés), Estrategia de Victorian[5] para los archivos electrónicos (VERS por sus siglas en inglés) y Modelos de Requisitos para la Gestión de Documentos Electrónicos (MoReq2).

Es importante para el GDA eXcriba poseer la habilidad de cumplir con los procesos regulatorios o legales y las políticas administrativas que defina el cliente. Por esta razón esta área representa

[4] Criterio estándar para el diseño de la Administración de Documentos de Archivo en aplicaciones de *software.*

[5] Desarrollado por la Oficina de Archivos Públicos de Victoria para ayudar a las agencias del gobierno a manejar, acceder y almacenar sus documentos de archivo.

uno de los puntos de apoyo fundamentales para el núcleo y es tomada en cuenta para la propuesta de la presente investigación.

1.1.5 Gestor de Contenidos web

Un Gestor de Contenidos *web* (WCM por sus siglas en inglés) ayuda a mantener, controlar, modificar y habilitar el contenido de una página *web*. Este contenido se mantiene en gran medida en una base de datos y se habilita utilizando lenguajes flexibles. De esta forma, los usuarios pueden interactuar con el sistema a través de un navegador *web*; posibilitando editar, controlar partes del diseño, dar mantenimiento, añadir páginas *web*, entre otras funcionalidades, sin que sean necesarios conocimientos previos de algún lenguaje de programación o HTML (ASSOCIATION FOR INFORMATION AND IMAGE MANAGEMENT, 2009).

Dentro de las principales funcionalidades que deben poseer los WCM se pueden definir:

1. Plantillas automatizadas de fácil mantenimiento.
2. Contenido de fácil edición.
3. Posibilidad de adicionar componentes de funcionalidades o extensiones.
4. Cumplimiento de los estándares *web*.
5. Actualizaciones regulares.
6. Flujos de trabajos sencillos.
7. Gestión de todo el ciclo de vida de los contenidos *web*.
8. Control de versiones de páginas *web*.
9. Funcionalidades de gestión de cambios y de creación y distribución de contenidos.

Esta área no queda incluida dentro del núcleo propuesto debido a que el GDA eXcriba no necesita una Gestión de Contenidos *web*. No es relevante para la presente investigación que los usuarios puedan controlar parte del diseño o dar mantenimiento al sistema.

1.1.6 Contenido Social

Anteriormente conocido como “colaboración documental”, el contenido social pasa a ser nombrado de esta forma con el fin de reflejar una amplia gama de tipos de contenidos. Utilizado para compartir documentos, colaboración y gestión del conocimiento y el apoyo a los grupos de trabajo. Compuesto mayormente por *Blogs*, *Wikipedia* y otras herramientas de apoyo en línea que se han añadido en los últimos años; el Contenido Social es la categoría que posee el más elevado crecimiento en los nuevos contenidos de la empresa (GARTNER, 2012).

Esta área resulta indispensable para poder brindar diferentes vías de colaboración entre los usuarios del GDA eXcriba. Por otro lado, es importante dejar plasmado que no todas sus características y funcionalidades resultan relevantes para la propuesta de arquitectura de la presente investigación. De esta forma, se incluyen las funcionalidades para la colaboración documental y quedan fuera de los intereses las funcionalidades para la gestión y colaboración del conocimiento, además de las herramientas de apoyo a grupos de trabajo.

1.1.7 Componentes Extendidos

Los componentes extendidos incluyen un conjunto de funcionalidades específicas para enfrentar situaciones especiales en diferentes organizaciones. Es la posibilidad que brinda un ECM de cubrir aquellas necesidades que no están incluidas en el resto de su sistema (BERGLJUNG, 2011). Esta área es utilizada en el eXcriba 2.0 para adicionar varias funcionalidades al núcleo requeridas por los usuarios. Dentro de los componentes extendidos se pueden encontrar:

1. Gestión de recursos digitales (DAM por sus siglas en inglés).
2. Formularios electrónicos.
3. Búsquedas.
4. Indexación de contenidos.
5. Gestión de correo electrónico.
6. Integración de aplicaciones empaquetadas.

Para el GDA eXcriba contar con el indexado de contenido posibilita la obtención de resultados de forma rápida, aún con grandes cantidades de contenido. Las búsquedas representan otro punto importante cuando se trata de brindar una solución para la Gestión de Documentos Administrativos. Lo más importante para la presente investigación es su cualidad de expandir las funcionalidades del sistema y poder cubrir las necesidades de los clientes. Por esa razón se incluye dentro de la propuesta arquitectónica como otra de las áreas relevantes seleccionadas.

1.1.8 Soluciones en el mundo

Con el desarrollo de las tecnologías surgen varios ECM en el mundo para atender el crecimiento no gestionado de la información. Cada uno tiene sus características y puntos de vistas para enfrentar al problema. Se puede afirmar que muchas de estas soluciones poseen elementos de valor que pudieran ser de interés para la propuesta arquitectónica de la presente investigación. Por esta razón se hace necesario contar con un estudio de algunos proveedores y sus soluciones.

1.1.8.1 IBM

Según los estudios de Forrester y Gartner, IBM es uno de los mayores proveedores de ECM en términos de cuota de mercado e ingresos totales (GARTNER, 2011; WEINTRAUB, 2011). Su desarrollo se centra cada vez más en soluciones de alto valor que son reconocidas en todo el mundo. Dicha empresa ha aprovechado con éxito su posición para brindar este tipo de solución con un amplio y riguroso programa, dirigiendo su atención principalmente hacia las transaccionales y a la gestión de casos de uso de contenido social.

Ventajas

- Los productos ECM de IBM tienen gran aceptación por las grandes empresas.
- IBM ha hecho considerables progresos en el Contenido Social, Componentes Extendidos y sus ofertas de Gestión de Contenidos *web* de su ECM (GARTNER, 2012).

Desventajas

- La amplia gama de productos de IBM es compleja, de igual manera lo es su implementación de *software*.
- IBM se enfrenta al reto de reducir sus ofertas actuales para satisfacer las necesidades de las medianas y pequeñas empresas, donde hay una clara oportunidad de crecimiento.
- En el mercado de gestión de contenidos, IBM carece de una estrategia robusta para la nube[6].

1.1.8.2 Microsoft

SharePoint es el ECM propuesto por Microsoft para incrementar la productividad y administrar los contenidos a través de la interfaz familiar de Office (WEINTRAUB, 2011). Es una plataforma que ha sido ampliamente adoptada y aceptada en el mundo (GARTNER, 2011). Más de la mitad de las investigaciones de Gartner sobre ECM tratan o incluyen la discusión de SharePoint, y un tercio de sus clientes lo usan como centro de su estrategia ECM. Sin embargo, los usuarios a menudo tienen que añadir los productos de terceros a SharePoint para lograr sus objetivos. Además, las empresas en el mundo han encontrado complejo trabajar a través de una amplia área geográfica eficazmente. Otros puntos donde están creciendo los motivos de preocupación son la facilidad de administración y de uso general.

Ventajas

[6] La informática en la nube es un paradigma que permite ofrecer servicios de computación a través de Internet.

- La posición de Microsoft como un vendedor con gran atractivo, ha ejercido una gran influencia en los usuarios finales de SharePoint.
- Muchos fabricantes de *software* ofrecen extensiones de diversos fines para SharePoint.
- SharePoint 2010 es utilizado para múltiples implementaciones de casos de uso, más grande en escala que las implementaciones observadas en la versión 2007.
- Al poseer un multifacético intercambio de información y una plataforma para su Contenido Social, SharePoint tiene un lugar importante en muchos entornos empresariales.

Desventajas

- Un conjunto de preocupaciones mencionadas por los usuarios en varias ocasiones es que se esperaba más funcionalidad nativa en áreas como la administración, salvas y restauras, flujo de trabajo, WCM, replicación y soporte móvil.
- La migración desde versiones anteriores de SharePoint 2010 sigue siendo un desafío para muchos, especialmente en el caso de las implementaciones personalizadas, a pesar de que Microsoft y sus socios ofrecen varias herramientas de migración.

1.1.8.3 EMC

Esta solución ha implementado estrategias para centrarse en mejorar la experiencia del usuario, las soluciones de la industria y la gestión de contenidos en la nube (GARTNER, 2012). La adquisición de *Syncplicity*[7] en mayo del 2012 le proporciona compartir el contenido y su gestión a través de dispositivos móviles. Esto la convierte en una plataforma clave para la gestión de contenidos en la nube y como Gestor de Contenido Empresarial se esfuerza por impulsar la adopción de los clientes. La evolución de las estrategias de EMC como ECM, le pueden ayudar a enfrentar los retos claves del mercado (WEINTRAUB, 2011).

Ventajas

- EMC ha vuelto a centrar sus fortalezas en las ciencias que estudian la vida y en los servicios públicos, proponiendo nuevas soluciones, tales como: "*Documentum Quality*" y "*Manufacturing Solution for Life Sciences and Documentum EPFM*".

[7] *Software* para las copias de seguridad y sincronización que la permite a sus usuarios almacenar y sincronizar los servicios en línea entre ordenadores.

- EMC ha puesto un gran énfasis en la gestión de contenidos en la nube, mediante soluciones como *Documentum*[8], *Captiva*[9], algunos productos de *Document Sciences, Documentum D2*, y las dos soluciones para ciencias de la vida mencionados anteriormente sobre la nube. EMC *OnDemand* ofrece un rápido y rentable aprovisionamiento de infraestructura para aplicaciones ECM en la nube.
- La estructura de Gestión de Contenidos que posee EMC es un sólido conjunto de productos complementarios, capaces de gestionar el ciclo de vida del contenido mejor que la mayoría de sus competidores. Sus puntos fuertes son la captura, depósito central, Gestión de Procesos de Negocio, Gestión de Documentos de Archivo y capacidades de composición de documentos. Todos ellos esenciales para soluciones de gestión de contenidos transaccionales.

Desventajas

- EMC sigue teniendo problemas con el aprovechamiento de su poder de mercado, punto vital para impulsar las ventas de su Gestor de Contenidos. Aunque ha realizado importantes inversiones en su negocio de administración de contenido, debe mejorar la comunicación y venta con los clientes en el futuro. Esta empresa debe seguir desarrollando sus canales de ventas y *marketing* para recuperar su fuerte impulso en el mercado (GARTNER, 2012).
- El deterioro en el negocio de *Documentum* es debido a los altos costos y su complejidad, junto con un mercado cada vez más competitivo (GARTNER, 2012).
- El cambio de estrategia en la gestión de contenido social de EMC con *CenterStage* y *eRoom* han dejado a clientes desencantados (GARTNER, 2012).
- Su estrategia de colaboración basada en la integración planificada de D2 con *Syncplicity* es aún incipiente y no probada (GARTNER, 2012).

1.1.8.4 Alfresco

Alfresco es una plataforma de código abierto para la colaboración y gestión de documentos críticos de una empresa, con más de 7 millones de usuarios empresariales de 75 países, más de cuatro mil millones de documentos gestionados y más de dos mil quinientas empresas en 180 países (ALFRESCO, 2011).

[8] Documentum es una plataforma de gestión de contenido empresarial, clave para apoyar los procesos que representan objetivos fundamentales en las organizaciones.
[9] Solución inteligente que permite convertir los documentos en información útil para los procesos de negocio.

Desde el año 2005, ha promovido con el concepto de código abierto; por este motivo ha ganado en fuerza y reconocimiento en todo el mundo. Alfresco es la única oferta de este tipo incluida en los estudios de Gartner, donde se califica a la empresa como visionaria en el mercado de los ECM. El producto se destaca por su fuerte lazo con los estándares de código abiertos, además brinda servicios de contenido, interoperabilidad de gestión (CMIS por sus siglas en inglés), posee una arquitectura moderna y una gran comunidad.

Ventajas

- Gran parte de la fuerza que adquiere Alfresco dentro de este tipo de soluciones, la ha obtenido por ser un sistema de código abierto.
- Soportar estándares como CMIS, y utilización de enfoques arquitectónicos como REST (transferencia de representación de estado por sus siglas en inglés) y servicios *web*.
- Ofrecer integración con *Liferay*, *Drupal*, *Joomla*, *Google Docs*, *Jive*, *Salesforce* e *IBM*.
- Alta aceptación por la versión de la comunidad libre.
- Única solución de gestión de contenido empresarial, que se puede usar en la nube y en los servidores de la empresa.

Desventajas

- Su estrategia ha cambiado mucho a lo largo de los años desde sus raíces como una alternativa de código abierto para Documentum, luego a SharePoint y luego como una plataforma de BPM. Esta vez se ha modificado una vez más por dar marcha atrás a su oferta de Gestión de Contenidos *web*, poner más de un foco en la nube y en la digitalización de documentos (GARTNER, 2012).

1.1.8.5 Cuadrante mágico de Gartner

Gartner es una de las empresas líderes mundialmente en consultoría e investigación sobre tecnologías de investigación. El objetivo de esta empresa es apoyar las decisiones estratégicas de sus clientes, y sus estudios brindan un avanzado punto de vista del mercado, las empresas y sus productos.

En el cuadrante de la figura 2 se pueden apreciar varias de las soluciones que existen en el mundo y se encuentran dentro de los estudios de Gartner. Dentro de estas soluciones se destaca en el

área de visionarios Alfresco, siendo el único *software* de código abierto incluido como uno de los ECM más completos y seguros (GARTNER, 2012):

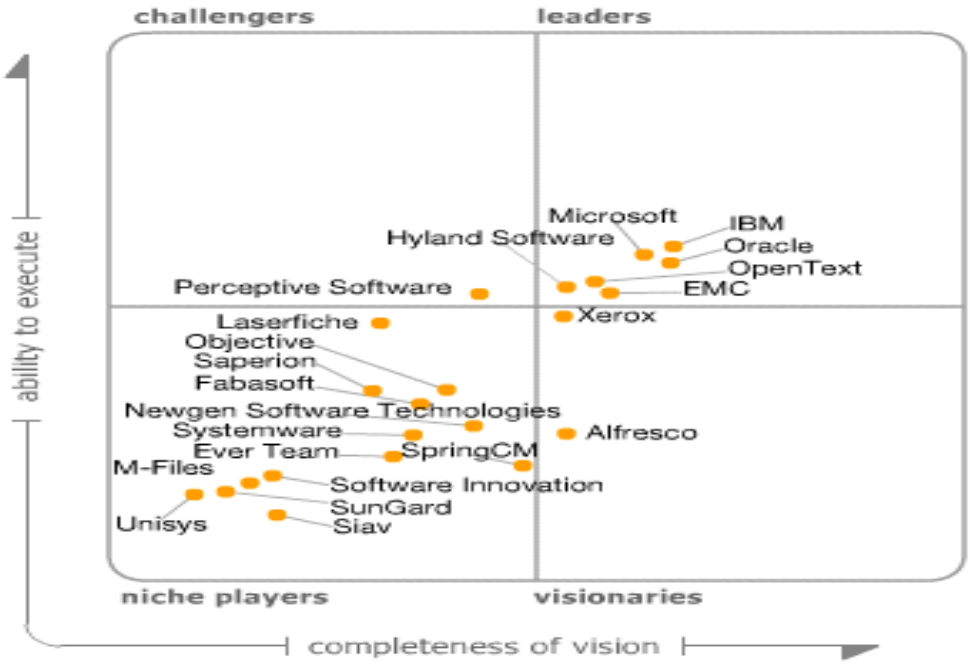

Figura 2: Cuadrante mágico de Gartner de octubre del 2012.

1.2 Metodología de desarrollo de *software*.

El proceso de desarrollo de *software* es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de *software* (JACOBSON, et al., 2000).

Las soluciones informáticas se vuelven más complejas en la medida en que van creciendo. De esta manera, con la necesidad de unos mecanismos de organización para este proceso, nacen las metodologías para el desarrollo de *software*. Estas son las encargadas de definir quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo (JACOBSON, et al., 2000).

El Proceso Unificado Racional (RUP por sus siglas en inglés) es una metodología tradicional que establece una serie de fases e hitos que permiten una buena documentación, generación de artefactos y definición de roles. Se apoya en tres principios claves:

1. **Dirigido por casos de uso:** Se utilizan los casos de uso para capturar los requisitos funcionales y definir los objetivos de las iteraciones. En cada una de estas, se identifican y especifican los casos de uso relevantes, luego se crea el diseño utilizando la arquitectura como

guía, se implementa el diseño dividido en componentes y se verifica que estos satisfacen los casos de uso.

2. **Centrado en la arquitectura:** La arquitectura de *software* maneja los aspectos más relevantes del sistema, y actúa como vista del diseño dando una perspectiva completa y describiendo los elementos más importantes. En RUP la arquitectura pasa a manejar un conjunto de factores como los requisitos no funcionales, la plataforma en que se ejecutará el sistema, la existencia de sistemas heredados, el uso de estándares, etc.
3. **Iterativo e incremental:** En RUP se define el marco de desarrollo en cuatro fases (inicio, elaboración construcción, transición), cada una de estas se dividen en una cierta cantidad de iteraciones que brindan como resultado un aumento o una mejora en el producto a construir.

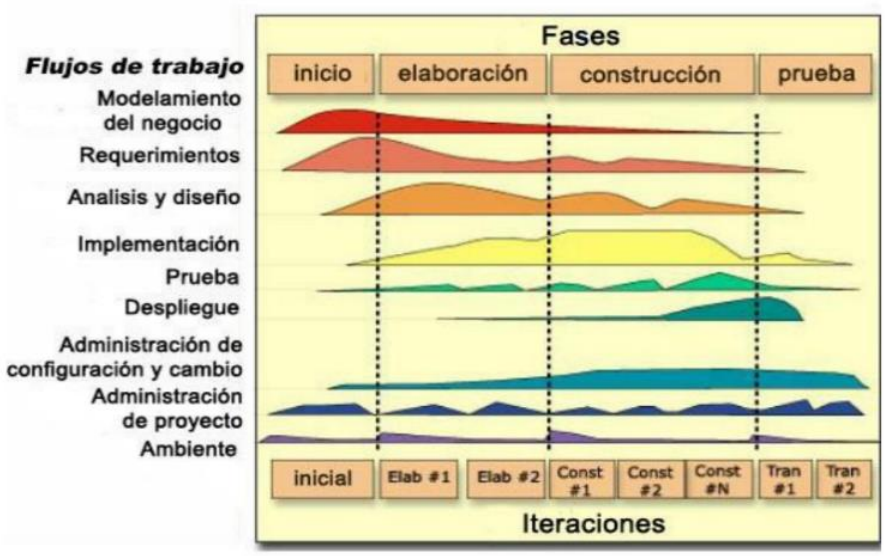

Figura 3: Ciclo de Vida de RUP

RUP define su ciclo de vida con nueve flujos de trabajos que tienen lugar en cuatro etapas (JACOBSON, et al., 2000). En cada una de las etapas se incluye:

1. Planificación.
2. Realización de los flujos de trabajos fundamentales.
3. Evaluación de la etapa.

RUP define roles que atienden la generación de artefactos y documentos en cada flujo y etapa que influye en la calidad del producto final. Es una metodología con años de experiencia, usada mundialmente y su aplicación ha impulsado buenos resultados nacionales e internacionales (JACOBSON, et al., 2000).

Teniendo en cuenta que los involucrados en el desarrollo de eXcriba cuentan con una mayor experiencia sobre la base de esta metodología; se propone su uso para guiar el proceso de desarrollo de *software* cuando la arquitectura esté finalizada.

1.3 Arquitecturas de S*oftware*.

Según el estándar IEEE 1471-2000 la Arquitectura de Software es la organización fundamental de un sistema enmarcada en sus componentes, sus relaciones y la relación con el ambiente, adicionando también los principios que orientan su diseño y evolución. La arquitectura se conoce como uno de los temas más importantes para el control de las interfaces y la integración de todos los componentes en el sistema (PEREIRA, et al., 2004), siendo un vínculo entre los problemas de negocio y las soluciones de las Tecnologías de la Información (IT por sus siglas en inglés) (BRITTON, et al., 2004).

Clements[10] por su parte afirma que es una vista del sistema que incluye los elementos principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta que aporta el más alto nivel de comprensión y la supresión del detalle a la mayor parte de las abstracciones (CLEMENTS, 1996).

Dentro del mundo de las arquitecturas de *software* existe una considerable diferencia entre la percepción académica y la práctica industrial. Para los arquitectos más prácticos le es común ver problemas importantes y complejos que no son considerados como tal por la academia (BOSCH, 2000). De igual manera en ambos se manejan ciertas herramientas, principios esenciales y conceptos que son vitales para el estudio en este campo.

1.3.1 Estilos Arquitectónicos

Los estilos arquitectónicos se pueden identificar como un conjunto de reglas de diseño que identifican las clases de componentes y conectores que se pueden utilizar para componer sistemas o subsistemas, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo (SHAW, et al., 1996). También se han definido como una entidad que se agrupa en cuatro elementos (GARLAN, et al., 1996):

[10] Paul Clements es un alto miembro del personal técnico del Instituto de Ingeniería de Software (SEI por sus siglas en inglés), donde trabaja en arquitectura de *software* e ingeniería de la línea de productos. Es autor de cinco libros y más de tres docenas de documentos sobre estos y otros temas.

- Vocabulario de elementos de diseño (componentes y conectores).
- Reglas de diseño o restricciones que determinan las composiciones permitidas de esos elementos.
- Interpretación semántica que proporciona significados precisos a las composiciones.
- Análisis susceptibles a practicarse sobre los sistemas construidos en un estilo.

A lo largo de los años los estilos los han planteado y organizado de diferentes formas. Bass[11], Clements y Kazman[12] los clasifican en cinco grupos (BASS, et al., 2003).

Flujo de datos	Proceso secuencial por lotes. Red de flujo de datos. Tubería-filtros.
Componentes independientes	Sistemas orientados por eventos. Procesos de comunicación.
Centrados en datos	Repositorio. Pizarra.
Máquina virtual	Intérprete
Llamado y retorno	Programa principal/Subrutinas. Tipos de dato abstracto. Objetos. Cliente-servidor basado en llamadas. Sistemas en capas.

Tabla 1: Estilos Arquitectónicos según Bass, Clements y Kazman.

Buschmann por su parte describe los estilos como "patrones arquitectónicos" de la siguiente manera (BUSCHMANN, et al., 1996):

Del fango a la estructura	Capas Tuberías-filtros. Pizarra.

[11] Len Bass es un miembro superior del personal técnico del SEI. Ha escrito dos libros premiados en la arquitectura de *software*, así como varios libros y numerosos artículos en una amplia variedad de áreas de ciencias de la computación e ingeniería de *software*.

[12] Rick Kazman es un alto miembro del personal técnico en el SEI. También es profesor asociado en la Universidad de Hawaii. Es autor de dos libros, editor de dos más, y ha escrito más de setenta trabajos sobre ingeniería de *software* y otros temas relacionados.

Sistemas distribuidos	Broker
Sistemas interactivos	Modelo-Vista-Controlador. Presentación-Abstracción-Control.
Sistemas adaptables	Reflexión. Microkernel.

Tabla 2: Estilos Arquitectónicos según Buschmann.

Es importante señalar que en la mayoría de las ocasiones es necesario utilizar varios estilos combinados; cada capa o componente puede ser internamente un estilo diferente de los demás (BILLY REYNOSO, 2005).

La arquitectura de la investigación posee un estilo en capas, ya que presenta una organización jerárquica tal que cada capa proporciona servicios a la capa inmediata superior y se sirve de las prestaciones que le brinda la inmediata inferior. Este estilo le permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales, con la posibilidad de aplicar refinamientos, optimizaciones y reutilizaciones futuras (BILLY REYNOSO, 2005). La descripción de cada capa de la arquitectura se detalla en el **Capítulo 2**.

1.3.2 Lenguaje de descripción arquitectónica

Los lenguajes de descripción arquitectónica de *software* (ADL por sus siglas en inglés) son lenguajes (gráficos, textuales, o ambos) para describir un sistema de *software* en términos de sus elementos arquitectónicos y las relaciones entre ellos (CLEMENTS, et al., 2002). Con los ADL se puede modelar una arquitectura antes de que se desarrollen las aplicaciones que la conforman. Su uso le permite al arquitecto observar las características del sistema de forma detallada, y con un nivel de abstracción genérico.

Algunos de los ADL más conocidos se desglosan a continuación (BILLY REYNOSO, 2005):

- ADML
  - Constituye un intento de estandarizar la descripción de la arquitectura sobre la base de XML.
  - Puede ser leído por cualquier *software* que soporte la lectura de XML.
  - Permite definir vínculos con objetos externos a la arquitectura.
- Aesop
  - Se basa en estilos de tuberías y filtros propios de UNIX.

- Manejo de métodos de análisis de tiempo real.

- Acme
  - Capaz de soportar el mapeo de especificaciones arquitectónicas entre diferentes ADL, como un lenguaje de intercambio de arquitectura. Característica que no lo hace un ADL en sentido estricto.
  - Describe con facilidad sistemas relativamente simples.
  - Soporta la definición de cuatro tipos de arquitecturas: la estructura, las propiedades de interés, las restricciones, los tipos y estilos.
- Darwin
  - Puede describir tipos de componentes mediante una interfaz consistente en una colección de servicios provistos (declarados por ese componente) o requeridos (se esperan que ocurran en el entorno).
  - Soporta la descripción de arquitecturas que se reconfiguran dinámicamente.
- Jacal
  - Cuenta con una representación gráfica que permite a simple vista transmitir la arquitectura del sistema, sin necesidad de recurrir a información adicional.
  - Ofrece un nivel de descripción de comportamiento. En este nivel se describen la relación entre las comunicaciones recibidas y enviadas por un componente.

Otra vía utilizada para describir la arquitectura es mediante el Lenguaje Unificado de Modelado que se describe en el siguiente acápite.

1.3.3 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML por sus siglas en inglés) es un lenguaje gráfico que soporta el análisis y diseño orientado a objeto capaz de describir un sistema de *software* (BHARATHI, et al., 2009). UML ofrece un grupo de notaciones gráficas para crear modelos abstractos para sistemas específicos, además de incluir aspectos conceptuales tales como procesos de negocio, funciones del sistema, aspectos concretos de lenguajes de programación y componentes reutilizables. Se puede aplicar en el desarrollo de *software* brindando gran variedad de formas para dar soporte a una metodología como el Proceso Unificado Racional, pero no especifica qué metodología o proceso usar. UML cuenta con varios tipos de diagramas, los cuales

muestran diferentes aspectos de las entidades representadas. El nivel de expresión de una arquitectura con UML es mayor que la de cualquier ADL (BHARATHI, et al., 2009).
Teniendo en cuenta los planteamientos anteriores y la experiencia con la que cuenta el equipo de desarrollo del GDA eXcriba, se propone el uso de UML como lenguaje para la descripción de la arquitectura.

1.3.4 Patrones

Un patrón describe un problema que se presenta repetidamente en un entorno y luego detalla una solución clave para este, de esta manera, se pueden usar estas soluciones en un sin número de oportunidades, sin que esto obligue a hacerlo de la misma manera (FOWLER, et al., 2002). Los patrones expresan un esquema de organización estructural para los sistemas de *software*. Proporcionan un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y lineamientos para organizar la relación entre ellos (BUSCHMANN, et al., 1996).
Existen diversos tipos de patrones entre los cuales se pueden citar a los de análisis, arquitectura, diseño, organización o proceso, programación e idiomas; describiéndose de la siguiente forma (BILLY REYNOSO, 2005):

- Patrones de arquitectura: Relacionados a la interacción de objetos dentro o entre niveles arquitectónicos.
- Patrones de diseño: Manejan conceptos de ciencia de computación en general, independiente de la aplicación.
- Patrones de análisis: Usualmente específicos de aplicación o industria.
- Patrones de organización o proceso: Para el desarrollo o proceso de administración de proyectos, o para técnicas o estructuras de organización.
- Idiomas: Para estándares de codificación y proyecto.

El núcleo de la presente investigación está basado en el ECM Alfresco, el cual utiliza el patrón arquitectónico Capas. Este patrón brinda importantes beneficios para lograr el objetivo propuesto en la arquitectura del sistema. Las ventajas que proporciona se pueden mostrar de la siguiente forma:

- Se pueden especializar equipos de trabajo en cada capa sin la necesidad de que ninguno tenga conocimiento de las restantes.

- Se puede sustituir capas con implementaciones alternativas del mismo servicio.
- Se pueden minimizar las dependencias entre las capas.
- Una vez que se tenga construida una capa esta puede ser utilizada por los servicios de niveles superiores.

En la capa de Dominio (o lógica del negocio) es donde se encuentra el núcleo de la presente investigación. Teniendo en cuenta una arquitectura de *software* basada en patrones la organización de esta capa se puede describir de la siguiente forma:

- Capa de Servicio: Definiendo los límites del núcleo con una capa que establece un grupo de servicios y coordina los componentes responsables de cada uno (FOWLER, et al., 2002).
- Modelo de Dominio: Situado debajo de la Capa de Servicio, es el patrón encargado de crear una aplicación de objetos interconectados. Aconsejado para lógicas del negocio complejas donde las reglas describan muchos comportamientos. Este patrón modela el área del negocio asignándole a cada objeto una responsabilidad (FOWLER, et al., 2002; BUSCHMANN, et al., 2007).
- Repositorio: Patrón ubicado entre el Modelo de Dominio y los datos. Un sistema con un modelo de dominio complejo siempre es beneficiado por una capa que aísle los detalles del acceso a los datos. Cobra mayor importancia cuando la aplicación posee interacciones con grandes volúmenes de información, ya que este patrón actúa como una colección de objetos en memoria. Por esta razón es de vital importancia para abstraer a las capas superiores de las operaciones llevadas a cabo con los datos (FOWLER, et al., 2002; BUSCHMANN, et al., 2007).

1.3.5 Documentación de la arquitectura

La documentación de la arquitectura consiste en documentar las vistas relevantes para luego añadir información aplicable a más de una (CLEMENTS, et al., 2002). Una vista es un subconjunto resultante de practicar una selección o abstracción sobre una realidad, desde un punto de vista determinado (BILLY REYNOSO, 2004). La mayoría de las corrientes que existen en la arquitectura de *software* proponen diferentes vistas, estas se pueden resumir de la siguiente manera (BILLY REYNOSO, 2004):

Kruchten (4+1 1995)	Booch, Rumbaugh y Jacobson (1999)	Arquitectura de Software Orientada por Patrones (POSA 1996)	Microsoft (2002)
Lógica	Diseño	Lógica	Lógica
Proceso	Proceso	Proceso	Conceptual
Física	Implementación	Física	Física
Desarrollo	Despliegue	Desarrollo	
Casos de uso	Casos de uso		

Tabla 3: Vistas utilizadas por diferentes corrientes.

Kruchten presento su modelo de vistas en el año 1995 vinculado con RUP. Este modelo contiene cuatro vistas de la arquitectura de *software* que se pueden describir de la siguiente forma (KRUCHTEN, 1995):

- Vista lógica: agrupando las abstracciones fundamentales del sistema tomados del ámbito del problema.
- Vista de proceso: englobando los procesos de ejecución organizados en las abstracciones de la vista lógica.
- Vista física: incluyendo los elementos de *hardware* y *software* involucrados.
- Vista de desarrollo: agrupando la organización de los módulos en un entorno de desarrollo.
- El quinto componente del modelo planteado considera todas las vistas anteriores en un contexto de casos de uso (BILLY REYNOSO, 2004).

Por otro lado Booch, Rumbaugh y Jacobson formularon un modelo de cinco vistas representando a la arquitectura de *software* como un conjunto de decisiones arquitectónicamente significativas. Sus vistas son descritas del siguiente modo:

- Vista de casos de uso: según el punto de vista de involucrados como los usuarios, analistas y probadores.
- Vista de diseño: agrupando las clases, interfaces y colaboraciones que modelan cómo se le da solución al problema.

- Vista de procesos: representando los procesos en los sistemas que presentan mecanismos de sincronización y concurrencia.
- Vista de implementación: englobando los componentes y archivos organizados en el sistema físico.
- Vista de despliegue: representando los nodos que conforman el *hardware* sobre el que se instaura el sistema.

Por su parte Frank Buschmann encabeza a un grupo que defiende a la arquitectura de *software* orientada por patrones. En su última lista de vistas se inclina por el modelo de cuatro vistas de Kruchten, aunque no incluye el quinto elemento. John Newton [13] afirma que le ha sido útil en varias ocasiones el uso de esta corriente arquitectónica (HART, 2007), además en repetidos momentos ha especificado su uso en el desarrollo de Alfresco (CARUANA, et al., 2010). Este enfoque es el que se toma como referencia para documentar la propuesta arquitectónica del núcleo de eXcriba ya que es la corriente citada en el desarrollo del ECM Alfresco.

En el modelo de Krutchen la vista de proceso se enfoca en requerimientos no funcionales como el rendimiento y la disponibilidad. Esta vista es esencial para resolver problemas sobre concurrencia, distribución y tolerancia a fallos en el sistema, además de representar cómo se organiza la arquitectura de procesos dentro de la vista lógica (KRUCHTEN, 1995). El objetivo de la presente investigación está centrado en los requerimientos no funcionales:

- Habilidad de realizar cambios futuros al sistema.
- Capacidad de someter al sistema a reparaciones y evolución.

Estos requerimientos son los propuestos para guiar a la arquitectura en el cumplimiento del objetivo de la investigación. De esta forma, la vista de proceso del modelo de Krutchen no representa un punto crítico en el manejo de los requerimientos no funcionales propuestos; por lo que en la documentación de la arquitectura esta vista no se tiene en cuenta.

1.3.6 Evaluación de la arquitectura

El objetivo de evaluar una arquitectura de *software* es conocer si el sistema a construir va a cumplir con las necesidades requeridas (CARRASCOSO, et al., 2010). Esta evaluación es una tarea

[13] Presidente y director técnico de Alfresco, además cofundador de Documentum en 1990. Ha aportado una gama de conceptos ampliamente utilizados en la industria de los ECM en la actualidad.

compleja, puesto que se pretende medir propiedades del sistema sobre la base de especificaciones abstractas. Por esa razón, la intensión es evaluar el potencial de la arquitectura diseñada para alcanzar los atributos de calidad requeridos. Las mediciones que se realizan sobre una arquitectura de *software* pueden tener distintos objetivos, dependiendo de la situación en la que se encuentre el arquitecto y la aplicabilidad de las técnicas que emplea. Algunos de estos objetivos son: cualitativos, cuantitativos y máximos y mínimos teóricos (CAMACHO, et al., 2004).

- La medición cualitativa se aplica para la comparación entre arquitecturas candidatas y tiene relación con la intención de saber la opción que se adapta mejor a cierto atributo de calidad. Este tipo de medición brinda respuestas afirmativas o negativas, sin mayor nivel de detalle.
- La medición cuantitativa busca la obtención de valores que permitan tomar decisiones en cuanto a los atributos de calidad de una arquitectura de *software*. El esquema general es la comparación con márgenes establecidos, como lo es el caso de los requerimientos de desempeño, para establecer el grado de cumplimiento de una arquitectura candidata, o tomar decisiones sobre ella. Este enfoque permite establecer comparaciones, pero se ve limitado en tanto no se conozcan los valores teóricos máximos y mínimos de las mediciones con las que se realiza la comparación.
- La medición de máximo y mínimo teórico contempla los valores teóricos para efectos de la comparación de la medición con los atributos de calidad especificados. El conocimiento de los valores máximos o mínimos permite el establecimiento claro del grado de cumplimiento de los atributos de calidad.

En ocasiones, la evaluación de una arquitectura de *software* no produce valores numéricos que permiten la toma de decisiones de manera directa. Ante la posibilidad de efectuar evaluaciones a cualquier nivel del proceso de diseño, con distintos niveles de especificación, Kazman propone un esquema general en relación con la evaluación de una arquitectura con respecto a sus atributos de calidad. En este sentido, afirma que de la evaluación de una arquitectura no se obtienen respuestas del tipo “sí - no”, “bueno - malo” o “6.75 de 10”, sino que explica cuáles son los puntos de riesgo del diseño evaluado (CAMACHO, et al., 2004).

1.3.6.1 Método de evaluación de Arquitecturas

- ATAM (*Architecture Trade-off Analysis Method*): está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM

(*Software Architecture Analysis Method*). El nombre del método ATAM surge del hecho de que revela la forma en que una arquitectura específica satisface ciertos atributos de calidad, y provee una visión de cómo los atributos de calidad interactúan con otros. El mismo se concentra en la identificación de los estilos arquitectónicos o enfoques arquitectónicos utilizados. Estos elementos representan los medios empleados por la arquitectura para alcanzar los atributos de calidad, así como también permiten describir la forma en la que el sistema puede crecer, responder a cambios, e integrarse con otros sistemas (CARRASCOSO, et al., 2010). El método de evaluación ATAM comprende nueve pasos, agrupados en cuatro fases.

- Bosch (2000). Plantea que: "El proceso de evaluación debe ser visto como una actividad iterativa, que forma parte del proceso de diseño, también iterativo. Una vez que la arquitectura es evaluada, pasa a una fase de transformación, asumiendo que no satisface todos los requerimientos. Luego, la arquitectura transformada es evaluada de nuevo" (CAMACHO, et al., 2004). Este método consta de cinco pasos divididos en dos etapas.
- ADR (*Active Design Review*). "ADR es utilizado para la evaluación de diseños detallados de unidades del *software* como los componentes o módulos. Las preguntas giran en torno a la calidad y completitud de la documentación y la suficiencia, el ajuste y la conveniencia de los servicios que provee el diseño propuesto" (CAMACHO, et al., 2004).
- ARID (*Active Reviews for Intermediate Design*). ARID es un método de bajo costo y gran beneficio, el mismo es conveniente para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo (CARRASCOSO, et al., 2010). ARID es un híbrido entre ADR y ATAM. Se fundamenta básicamente en ensamblar el diseño de los *stakeholders* para articular los escenarios de usos importantes o significativos, y probar el diseño para ver si satisface los escenarios. Como resultado de la aplicación de dicho procedimiento se obtiene un diseño de alta fidelidad acompañado de una alta familiarización con el diseño de los *stakeholders*. Este método consta de nueve pasos agrupados en dos fases.
- Losavio (2003): Es un método para evaluar y comparar arquitecturas de *software* candidatas, que hace uso del modelo de especificación de atributos de calidad adaptado del modelo ISO/IEC 9126. La especificación de los atributos de calidad haciendo uso de un modelo basado en estándares internacionales ofrece una vista amplia y global de los

atributos de calidad, tanto a usuarios como arquitectos del sistema, para efectos de la evaluación (LOSAVIO, et al., 2003). El método contempla siete actividades.

- El MECABIC está inspirado en ATAM, aunque con el objetivo de facilitar su aplicación sobre Arquitecturas de Software Basadas en Componentes (ASBC) se incluyó en algunos de sus pasos un enfoque de integración de elementos de diseño, inspirado en la construcción de partes arquitectónicas adoptado por el Diseño Basado en la Arquitectura (ABD por sus siglas en inglés) (GONZÁLEZ, et al., 2005). Además, se propone un árbol de utilidad inicial basado en el modelo de calidad ISO-9126 instanciado para Arquitectura de Software propuesto por Losavio[14]. La adopción de este modelo por parte del MECABIC permite concentrarse en características que dependen exclusivamente de la arquitectura, y al ser un estándar facilita la correspondencia con características de calidad consideradas por los métodos anteriores. Los escenarios incluidos en este árbol son específicos para aplicaciones basadas en componentes.

1.4 Herramientas

Sobre la base del objetivo planteado se hace necesario el estudio de herramientas que faciliten el desarrollo del GDA eXcriba. La principal premisa es brindar la posibilidad de incorporar el código necesario para las funciones del eXcriba al núcleo sin que esto modifique a este último.

1.4.1 Lenguaje de programación Java

Java es un lenguaje de programación orientado a objeto[15] que deriva de lenguajes como C y C++. Como diferencia presenta un modelo de objeto de menor complejidad y elimina algunas de las herramientas de bajo nivel. Presenta mejoras en el uso de la memoria ya que esta es gestionada por el lenguaje y no por el programador. Se propone el uso de Java ya que se utiliza en la mayor parte del código fuente de Alfresco.

[14] Fundadora y Coordinadora del Centro ISYS (Investigación en Ingeniería de Software Y Sistemas), fundadora y Responsable de LaTecS (Laboratorio de Tecnología del Software), responsable del laboratorio MoST (Modelos, Software y Tecnología).

[15] La programación orientada a objetos es un paradigma de programación que usa objetos en sus interacciones, para diseñar aplicaciones y programas informáticos. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos.

1.4.2 Ant

Apache Ant es una herramienta de construcción de *software* que es usada en la programación para la realización de tareas mecánicas y repetitivas, normalmente, durante la fase de construcción y en la mayoría de los casos en proyectos de desarrollo de Java (APACHE, 2013). Es un proyecto de código abierto de la Fundación Apache Software y una de las primeras ayudas que brindó fue solucionar los problemas de portabilidad de la compilación. Su misión consiste en guiar los procesos descritos en un archivo de construcción para lograr los objetivos trazados. Esta herramienta provee una gran cantidad de funcionalidades para cubrir las necesidades que puedan ocurrir en la construcción de un sistema, posibilitando además ser utilizada en cualquier plataforma.

Alfresco utiliza esta herramienta para su proceso de construcción y brinda los ficheros para la composición de cada uno de sus componentes. De igual forma el equipo de desarrollo del eXcriba cuenta con experiencia en el uso de Ant, lo que facilita la comprensión de los ficheros de construcción de Alfresco. Sobre la base de los planteamientos anteriores se propone el uso de esta herramienta de construcción para cumplir con el objetivo de la investigación.

1.4.3 Eclipse

Eclipse es un Ambiente de Desarrollo Integrado (IDE por sus siglas en inglés) de código abierto y multiplataforma. Está compuesto por un grupo de herramientas enfocadas a agilizar el trabajo del programador brindando una inmensa cantidad de comodidades para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de *software*, aplicaciones *web*, etc.

Dentro de sus principales características se pueden encontrar:

1. Editor de texto.
2. Resaltado de sintaxis.
3. Compilación.
4. Pruebas unitarias con Junit.
5. Integración con varias herramientas de construcción, entre ellas Ant.
6. Asistentes: para creación de proyectos, clases, etc.
7. Refactorización: Eclipse también puede manejar componentes, los cuales pueden brindar funcionalidades adicionales como:
    - Control de versiones con Subversion.

- Integración con Hibernate.

Por todas sus características, Eclipse se ha ganado un gran espacio en el mundo, tanto así que es el más utilizado para desarrollar algunos ECM como Alfresco (POTTS, 2009) y Nuxeo (NUXEO, 2010). Por este motivo es común obtener el código fuente de algunos ECM como Alfresco en forma de proyectos realizados en este IDE. Sobre la base de los planteamientos anteriores se propone a Eclipse como IDE de desarrollo para el núcleo de eXcriba.

1.4.4 Servidor de Aplicaciones

Apache Tomcat es un servidor *web* multiplataforma que se desarrolla bajo el proyecto denominado Jakarta perteneciente a la Fundación Apache Software. Implementa las especificaciones para servlet y Páginas Servidor de Java (JSP por sus siglas en inglés). Está desarrollado en el lenguaje Java, lo cual le permite ser multiplataforma. Es utilizado además por la empresa Alfresco para desarrollar y desplegar su solución. De igual forma el núcleo propuesto en esta investigación propone su uso como servidor de aplicaciones *web*.

1.4.5 Control de Versiones

Subversion es un sistema de control de versiones totalmente libre desarrollado bajo una licencia de tipo Apache/BSD. Presenta como característica diferente a otros sistemas de la misma función, la propiedad de organizar su repositorio mediante un único número de versión; lo que identifica un estado común de todos los archivos en el tiempo. Brinda el acceso a su contenido a través de la red, permitiendo en muchos casos ser utilizado como herramienta de colaboración específicamente en entornos de desarrollo de *software*.

1.5 Caracterización del estado actual del núcleo del GDA eXcriba

Desde los inicios del desarrollo del GDA eXcriba se optó por el uso del ECM Alfresco como Sistema de Gestión del Contenido (CMS por sus siglas en inglés). De esta forma eXcriba pudo disponer de una sólida base con una amplia gama de funcionalidades para poder brindar sus servicios. Como afirmara Jeff Potts[16] en una frase bien conocida por todos los miembros del desarrollo del eXcriba "La libertad del repositorio de Alfresco, particularmente su disponibilidad de ser expuesto fácilmente como un conjunto de servicios hacen de Alfresco una plataforma ideal para las aplicaciones centradas en contenidos" (POTTS, 2009; DAVID, 2011).

[16] Director de la comunidad de Alfresco con 18 años de experiencia en gestión de contenidos y colaboración.

Este ECM está compuesto por un conjunto de proyectos, un grupo desarrollado por la propia empresa y el resto son herramientas implementadas por terceras partes[17]. Esta fusión se puede entender como un desarrollo de *software* que no pretende reinventar la rueda, sino que utiliza lo existente y desde este punto comienza a desarrollar la solución.

Con las nuevas versiones del Alfresco cada uno de sus proyectos fue evolucionando de manera acelerada. Cada liberación adicionaba valor agregado al sistema y corregía errores de versiones anteriores. Por otro lado, la versión 2.0 del GDA eXcriba sufría transformaciones en algunos de sus proyectos. Estos cambios fueron realizados directamente en el código fuente y tenían el objetivo de lograr satisfacer ciertas necesidades del GDA. De esta manera, todos los proyectos modificados incluían elementos necesarios por el eXcriba, lo cual impedía actualizarlo de sus versiones superiores.

Ante la imposibilidad de actualizar el ECM, el equipo de desarrollo del GDA toma la decisión de corregir manualmente los errores reportados en cada versión. Decisión que conllevó a realizar continuos cambios sobre el sistema, alterando la organización y complejidad del mismo. De esta forma, la versión de Alfresco que es utilizada actualmente por eXcriba ha adquirido un nivel de complejidad que dificulta el entendimiento del código y por ende complica la adición de nuevas funcionalidades y corrección de errores. El mantenimiento sobre el ECM utilizado cada vez es menos sostenible, además no es viable, debido a que duplica esfuerzos en realizar cambios en vez de obtener las actualizaciones directamente.

Estas limitaciones corroboran la necesidad de seguir buscando vías para lograr un mantenimiento estable del núcleo del GDA eXcriba. De esta forma, la respuesta que se ofrece en la presente investigación es la elaboración de una propuesta arquitectónica que es formalizada en el siguiente capítulo.

[17] De esta forma se le hacen llamar a las herramientas que son desarrolladas por otras organizaciones.

1.6 Conclusiones

Luego de finalizado el capítulo se concluye que:

- Se logró una caracterización de los ECM que permitió seleccionar cinco de las siete áreas que los componen para conformar la propuesta arquitectónica de la investigación. Estas áreas fueron: el gestor de documentos, el gestor de procesos de negocios, la administración de documentos de archivo, el contenido social y los componentes extendidos.
- Se llevó a cabo un estudio de las arquitecturas de *software* que permitió seleccionar los estilos arquitectónicos, patrones y vistas para conformar la propuesta realizada en el siguiente capítulo.
- El estudio realizado a los ECM posibilitó detectar a Apache Ant como herramienta de construcción utilizada por Alfresco. De esta forma, se selecciona para llevar a cabo el proceso de construcción del núcleo de la investigación.

Capítulo 2

Propuesta de la arquitectura

En este capítulo se propone una arquitectura candidata para el núcleo del Gestor de Documentos Administrativos eXcriba.

Hasta las pequeñas arquitecturas de *software* deben ser reestructuradas, cuando la dificultad de hacer los cambios se vuelve desproporcionalmente grande con respecto a su tamaño (LINDVALL, et al., 2002). Se puede afirmar además, que afectar la complejidad de un sistema reduce su entendimiento y con ello la posibilidad de su mantenimiento (BANKER, et al., 1993). De esta forma, el presente capítulo tiene el objetivo de lograr una arquitectura que permita renovar los componentes del núcleo del GDA eXcriba sin que esto afecte la complejidad del sistema.

2.1. Funcionalidades

Las funcionalidades son la habilidad que posee un sistema para hacer el trabajo por el cual fue concebido (BASS, et al., 2003). Una tarea requiere que un número de elementos del sistema trabajen de manera coordinada para completar un trabajo. Por esta razón es importante definir qué elementos son los encargados de asumir qué responsabilidad para que el sistema pueda ofrecer la funcionalidad requerida.

Para la propuesta arquitectónica de la presente investigación las funcionalidades se han dividido según las áreas seleccionadas en el **acápite 1.1**. De esta manera, se pueden desglosar las áreas y sus funcionalidades:

Gestor de documentos y Administración de documentos de archivos	Almacenar contenido. Crear contenido. Eliminar contenido. Modificar contenido. Recepción del contenido.

	Controlar el mantenimiento del contenido. Controlar el uso del contenido. Controlar la disposición del contenido.
Flujos de trabajo o gestor de procesos de negocios.	Automatizar procesos de negocios.
Contenido social	Compartir contenido.
Componentes extendidos	Buscar contenidos. Indexar contenido. Integrar aplicaciones empaquetadas. Gestión de recursos digitales.

Tabla 4: Áreas y funcionalidades de la propuesta.

2.2. Requerimientos no funcionales

Los requerimientos no funcionales son aquellas cualidades o propiedades que el *software* debe tener y además describen las características del sistema a desarrollar. En la presente investigación se proponen sobre la base de la Arquitectura de Software del GDA eXcriba definida por el Departamento de Gestión Documental.

2.2.1. Requerimientos de hardware

Hardware del servidor del sistema:

- Periférico de comunicación para permitir la comunicación del servidor del sistema con las estaciones de trabajo clientes y el servidor de base de datos.

Hardware del servidor de base de datos:

- Periféricos de comunicación como la tarjeta de red para permitir la comunicación entre el servidor de base de datos y el servidor del sistema.

2.2.2. Requerimientos de software

Servidor del sistema:

- Sistema operativo Ubuntu 12.4, Nova 3, Debian Squeeze y Windows Server 2003.
- Máquina virtual de Java 7.0.170 (32-bit).
- Servidor de aplicaciones *web* Apache Tomcat 7.0.

2.2.3. Requerimientos de seguridad

Servidor del sistema y de base de datos:

- Debe poseer un backup (cada servidor en caso de ser más de uno) para asegurar la mayor disponibilidad de la información y evitar la pérdida de datos.
- Los usuarios podrán acceder a su información una vez que se hayan autenticado en el sistema.
- Se registran todas las operaciones que se realicen en el sistema.
- Los niveles de acceso a la información se manejarán mediante roles.

2.3. Atributos de calidad

Con el objetivo de proporcionar una base sólida para tomar decisiones objetivas sobre el diseño y permitir realizar predicciones con una precisión razonable sobre el sistema, se propone el uso de los siguientes atributos de calidad de *software* para guiar la arquitectura:

- Habilidad del sistema para realizar el trabajo para el cual fue concebido (Funcionalidad).
- Habilidad de realizar cambios futuros al sistema (del inglés Modificability).
- La capacidad de someter al sistema a reparaciones y evolución (del inglés Mantenibility).
- Habilidad que posee el sistema para ser ejecutado en diferentes ambientes de *hardware*, *software* o una combinación de los dos (Portabilidad).

El uso de estos atributos de calidad en la presente investigación brinda la posibilidad de guiar y evaluar el sistema para llegar a una mejor propuesta arquitectónica.

2.4. Vistas arquitectónicas

Las vistas se pueden considerar como el concepto más importante en la documentación de la arquitectura de *software* (BASS, et al., 2003). Son la representación coherente de un grupo de elementos arquitectónicos, creados y utilizados por todos los involucrados dentro de la arquitectura. Una sola vista no define una arquitectura, sino que es el conjunto de todas las vistas las que conducen una arquitectura de *software*. La selección de las vistas de la arquitectura depende de la necesidad de su uso por los involucrados (CLEMENTS, et al., 2002).

2.4.1. Vista lógica

La arquitectura lógica soporta los requisitos funcionales: lo que el sistema debe proveer en términos de servicios a sus usuarios. Luego el sistema es descompuesto en grupos claves y

abstractos, tomados generalmente del ámbito del problema, con forma de objetos o clases de objetos (KRUCHTEN, 1995). Esta descomposición no es solamente con el objetivo del análisis funcional, sino además sirve para identificar mecanismos comunes y elementos del diseño alrededor de varias partes del sistema.

En la siguiente figura se puede apreciar la descomposición del sistema en sus grupos claves mediante el uso del patrón Capas (Layers en inglés).

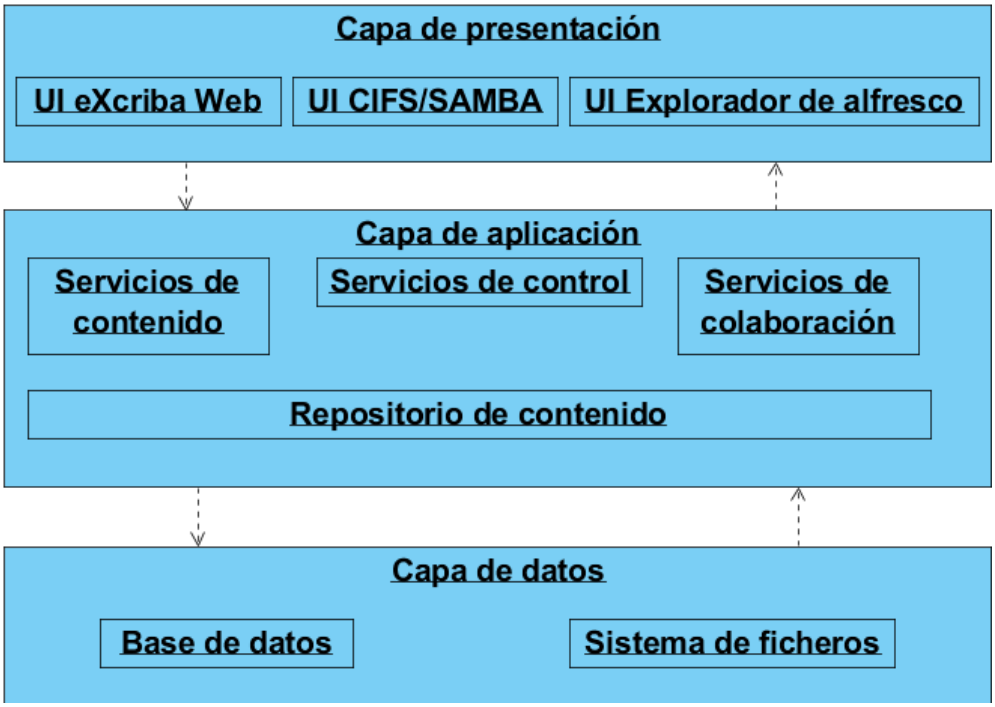

Figura 4: Vista lógica del sistema.

En la **capa de presentación** se brindan dos clientes, el cliente *web* eXcriba y el explorador de Alfresco:

- El cliente *web* de eXcriba es un Gestor de Documentos Administrativos cuyo objetivo es brindar servicios que cumplan con los requisitos de la gestión documental, apoyando a las actividades de los procesos documentales.
- El explorador de Alfresco es incluido en el núcleo como una herramienta de administración de apoyo, ya que expone todas las funcionalidades del ECM utilizado y es requerido para llevar a cabo acciones más avanzadas como el mantenimiento de las reglas de negocio y la gestión de grupos y usuarios.

- Otra forma de interactuar con el núcleo es mediante un explorador de archivo de un sistema operativo, donde los usuarios pueden compartir documentos a través de la red mediante el uso de protocolos como: Protocolo de Transferencia de Archivos (FTP por sus siglas en inglés), Sistema de Fichero Común para Internet (CIFS por sus siglas en inglés) y la Edición y Versionado Distribuidos sobre la *web* (WEBDAV por sus siglas en inglés).

La **capa de aplicación (o capa de dominio)** abarca el repositorio de contenido y los servicios que este brinda, dividiendo así la capa en dos: la capa de modelo de dominio y la capa de servicio. Se usan dos estándares para definir el repositorio de contenidos: Servicios de Interoperabilidad de Gestión del Contenido (CMIS por sus siglas en inglés) y Repositorio de Contenido de Java (JCR por sus siglas en inglés). Estos estándares poseen las especificaciones para la definición, almacenamiento, recuperación, versionado y gestión de permisos de los contenidos. Por ese motivo su uso brinda una implementación segura, escalable y eficiente (ALFRESCO, 2011).

La capa de aplicación provee las siguientes categorías de servicios incluidas dentro del repositorio de contenido:

- Servicios de contenido: brinda servicios avanzados para la gestión del contenido.
- Servicios de control: encapsula los servicios para los flujos de contenido.
- Servicios de colaboración: brinda servicios para la integración de contenidos con las redes sociales.

En la capa inferior del diagrama se encuentra la **capa de datos**, agrupando al sistema de ficheros y la base de datos. El sistema de ficheros está compuesto a la vez por los ficheros del contenido y los índices de búsqueda. En esta capa se guarda toda la información y el contenido que es gestionado por el núcleo cuando realiza sus funciones.

Alfresco 4.2 es la versión que se propone como base para conformar el núcleo de la presente investigación. Este ECM está compuesto por veinticinco proyectos de los cuales solo doce (en lo adelante proyectos principales) conforman el núcleo propuesto en este capítulo. Los restantes proyectos (en lo adelante proyectos secundarios) brindan funcionalidades que van más allá de las necesidades del GDA eXcriba, por ese motivo no se incluyen en la propuesta. La siguiente figura muestra cómo está compuesto el núcleo y diferencia los proyectos principales (de color verde) de los que tienen un valor adicional (de color azul).

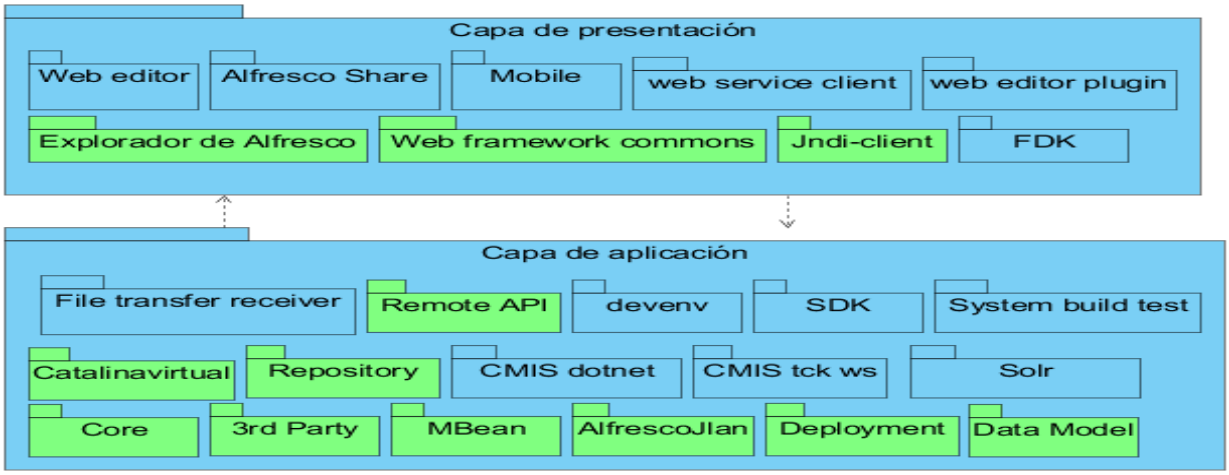

Figura 5: Vista lógica de las capas de presentación y aplicación.

Los proyectos principales se pueden describir de la siguiente forma:

- **3rd-party:** Contiene todas las funcionalidades desarrolladas por terceras partes que a la vez son utilizadas por el núcleo.
- **Core:** Agrupa clases de utilidades y ayudas genéricas así como configuraciones y gestión de errores del sistema.
- **MBean:** El servidor de Mbean actúa como un agente en el cual se puede publicar información relacionada con el sistema: gestión de memoria, hilos, clases cargadas.
- **Deployment:** Es el receptor del sistema de ficheros de Alfresco. Encargado de definir nuevos protocolos de comunicación con el sistema de ficheros.
- **Alfresco Jlan:** Contiene las implementaciones de los protocolos CIFS, FTP y WEBDAV para Alfresco.
- **Repository:** Engloba toda la implementación del repositorio de contenidos de Alfresco. Dentro de sus principales características se pueden nombrar:
    - Gestión de contenidos, correo electrónico y contenidos *web*.
    - Implementación de flujos de trabajo.
    - Búsquedas no indexadas e indexadas mediante el uso de Lucene[18] y Solr[19].
    - Implementación de CMIS y JCR para el repositorio.

[18] API para la recuperación de información originalmente implementada en Java. Utilizada en aplicaciones que requieran indexado y búsquedas a texto completo.
[19] Motor de búsquedas de código abierto implementado en Java por el proyecto Lucene.

    - Implementación y configuración de las listas de control de acceso (por sus siglas en inglés ACL).
    - Implementación de auditorías.
- **Remote API:** Contiene el programa de aplicación de interfaz (API por sus siglas en inglés) de los servicios *web* de Alfresco.
- **Data model:** Agrupa las implementaciones para el trabajo sobre *metadata*, aspectos y asociaciones de los contenidos. De igual modo define todos los modelos de contenido del sistema.
- **Explorador de Alfresco:** Herramienta para la administración del sistema, que expone todas las funcionalidades del ECM utilizado.
- **Jndi client, web framework commun:** Contienen implementaciones que son usadas por el Explorador de Alfresco.
- **Catalina virtual:** Contiene implementaciones usadas por el Repositorio.

Es importante aclarar que los proyectos que no se incluyen dentro de la propuesta pueden en algún momento ser necesidades del GDA eXcriba. Por esa razón es una prioridad en la arquitectura brindar los mecanismos para incluirlos. Estos detalles se pueden encontrar en el **acápite 2.2.3**.

2.4.2. Vista física

La vista física permite establecer la distribución física del sistema mediante el uso de nodos, los componentes que los conforman y las vías de conexión que se establecen entre ellos. El diagrama de despliegue que se muestra a continuación muestra la información que contiene esta vista.

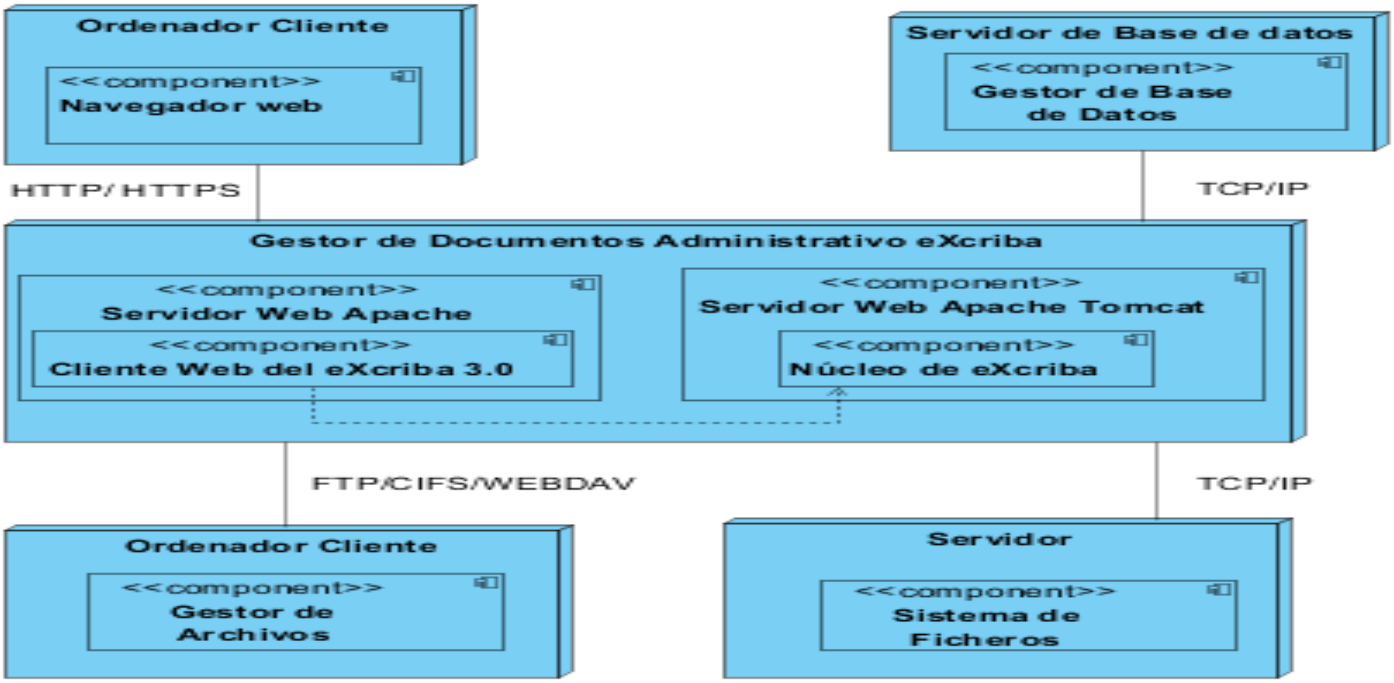

Figura 6: Diagrama de despliegue

Los **ordenadores clientes** son las estaciones de trabajo utilizadas por los usuarios para interactuar con el núcleo de eXcriba. Como se muestra en el diagrama anterior, para la comunicación es necesario un navegador *web* o un gestor de archivos (del inglés file manager). Esta comunicación entre los ordenadores clientes y el núcleo del eXcriba se realiza mediante los protocolos FTP, CIFS y WEBDAV con el uso de un gestor de archivos y el protocolo de transferencia de hipertexto (HTTP por sus siglas en inglés) y el protocolo seguro de transferencia de hipertexto (HTTPS por sus siglas en inglés) mediante el uso de un navegador *web*.

El **servidor de base de datos** está encargado de la centralización de la información recopilada por el sistema. Alfresco permite el uso de cualquiera de los sistemas de gestión de base de datos (SGBD) mostrados a continuación:

- Oracle
- MySQL
- Microsoft SQL Server
- PostgreSQL

Esta posibilidad contribuye a un mayor cubrimiento de las necesidades que puedan presentar los clientes. Del mismo modo, los requerimientos de *hardware* en este servidor varían según estas necesidades.

El nodo que representa **el sistema de ficheros** no es más que un servidor dedicado a guardar todos los contenidos gestionados por el núcleo. Alfresco brinda la posibilidad de distribuir esta parte de su arquitectura con el objetivo de separar los contenidos a un lugar dedicado al almacenamiento.

El **gestor de documentos administrativos eXcriba** identifica al servidor donde radica el núcleo y el cliente *web* del GDA eXcriba. Dentro del nodo residen dos servidores *web*:

- Servidor Apache con el cliente *web* del GDA eXcriba.
- Apache Tomcat con el núcleo del GDA eXcriba.

En este servidor se pueden ubicar además, el gestor de base de datos y el sistema de ficheros, lo que evitaría el uso de dos servidores adicionales. No obstante, se diagrama de esta forma para lograr una mejor visión de cómo pueden distribuirse los componentes que usa el núcleo de eXcriba; lo cual permite generalizar la mayoría de las posibles necesidades que puedan presentar los clientes en este sentido.

2.4.3. Vista de desarrollo

La arquitectura de desarrollo se centra en la organización de los módulos del *software* en su entorno de desarrollo. De esta manera, se puede entender que el sistema es empaquetado en pequeños subsistemas o librerías de programas que son implementados por los programadores. La arquitectura de implementación de un sistema es generalmente representada por los diagramas de módulos y subsistemas (KRUCHTEN, 1995).

A continuación se propone la organización de las carpetas donde se lleva a cabo la implementación del GDA eXcriba 3.0:

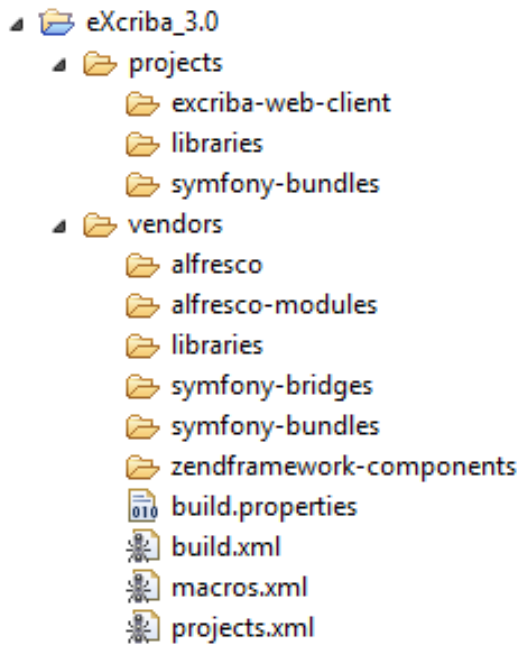

Figura 7: Organización de las carpetas para la implementación del GDA eXcriba.

Dentro de la carpeta ***projects*** se encuentra organizado el código fuente del cliente *web* de eXcriba. Los detalles de este cliente se encuentran fuera del alcance de la presente investigación.

En la carpeta ***vendors*** se localizan todas las soluciones de los proveedores que son utilizadas para la construcción del GDA eXcriba. En esta vista, la presente investigación se centra en las carpetas alfresco y alfresco-modules, adicionando los ficheros build.xml, macros.xml, projects.xml y build.properties dado que estos elementos conforman el núcleo del eXcriba. El directorio ***alfresco*** reúne todos los proyectos y ***alfresco-modules*** los módulos utilizados en la construcción del núcleo. En la Vista Lógica del sistema se mostró de forma detallada los proyectos principales que conforman el sistema y los secundarios, localizándose ambos en el directorio ***alfresco***. Por otro lado, la carpeta ***alfresco-modules*** contendrá todos los módulos utilizados por el sistema.

Como se definió en el capítulo anterior se utiliza en esta arquitectura la herramienta Ant para la construcción del núcleo. Los ficheros build.xml, macros.xml, projects.xml y build.properties son los elementos en los que se define cómo llevar a cabo esta construcción. Estos ficheros se pueden describir de la siguiente forma:

- build.xml: Contiene los objetivos para la construcción del núcleo y el apoyo al entorno de desarrollo.

- projects.xml: Engloba los objetivos para la construcción de cada uno de los proyectos principales y de valor añadido del núcleo propuesto en este capítulo.
- macros.xml: Contiene los objetivos generales para las operaciones de construcción.
- build.properties: Agrupa todas las variables que son utilizadas a lo largo del proceso de construcción del núcleo.

En Ant un objetivo representa un paso a llevar a cabo en el proceso de construcción de un sistema. De igual manera un objetivo puede estar compuesto por uno o varios objetivos, entendiéndose de manera separada toda la secuencia de pasos necesarios de comienzo a fin. Finalmente, se conforma la estructura del núcleo como un objetivo compuesto por los proyectos principales mostrados en la Vista Lógica:

```
<target name="eXcriba-core"
        depends="incremental-core,
                 incremental-datamodel,
                 incremental-jlan,
                 incremental-mbeans,
                 incremental-deployment,
                 incremental-repository,
                 incremental-remoteapi,
                 incremental-webframeworkcommons,
                 incremental-jndi,
                 incremental-catalinavirtual,
                 package-webclient-war">
</target>
```

Figura 8: Objetivo para la construcción del núcleo.

La figura muestra el objetivo llamado “eXcriba-core” compuesto por los proyectos seleccionados para conformar la arquitectura de la investigación. Como Ant representa un estándar para Java, cada IDE realiza una implementación en esta herramienta para llevar a cabo la construcción de sus proyectos. De esta forma, se puede añadir cualquier proyecto escrito en Java al núcleo con solo especificar su ruta en el fichero build.properties e incluir su objetivo de construcción en el objetivo “eXcriba-core”.

Construir de manera separada cada uno de los proyectos involucrados dentro del núcleo permite definir los intereses específicos del GDA eXcriba sin modificar el código de los proyectos del

Alfresco. Por esta vía se consigue lograr una independencia del ECM sin que esto afecte la posibilidad de renovar los componentes de versiones superiores. De esta forma, se cumple con una parte del objetivo de la presente investigación.

Para renovar un componente es necesario conocer sus funciones y dependencias. De este modo es posible tener una percepción del impacto que puede ocasionar su ausencia en el sistema. En las siguientes figuras se muestran los componentes que conforman las capas de aplicación y presentación con sus respectivas dependencias.

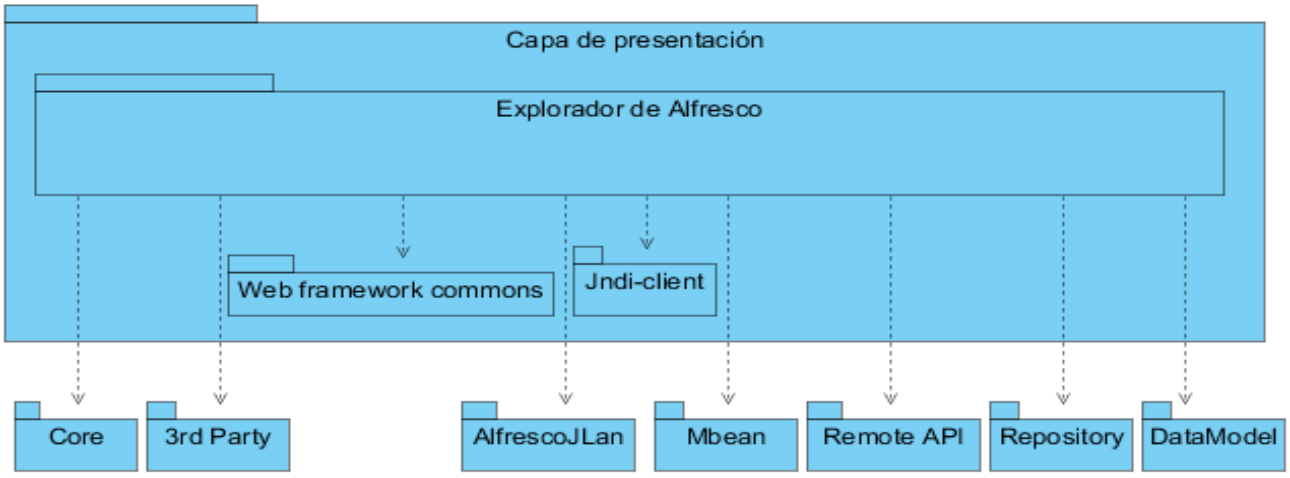

Figura 9: Vista de implementación de la capa de presentación.

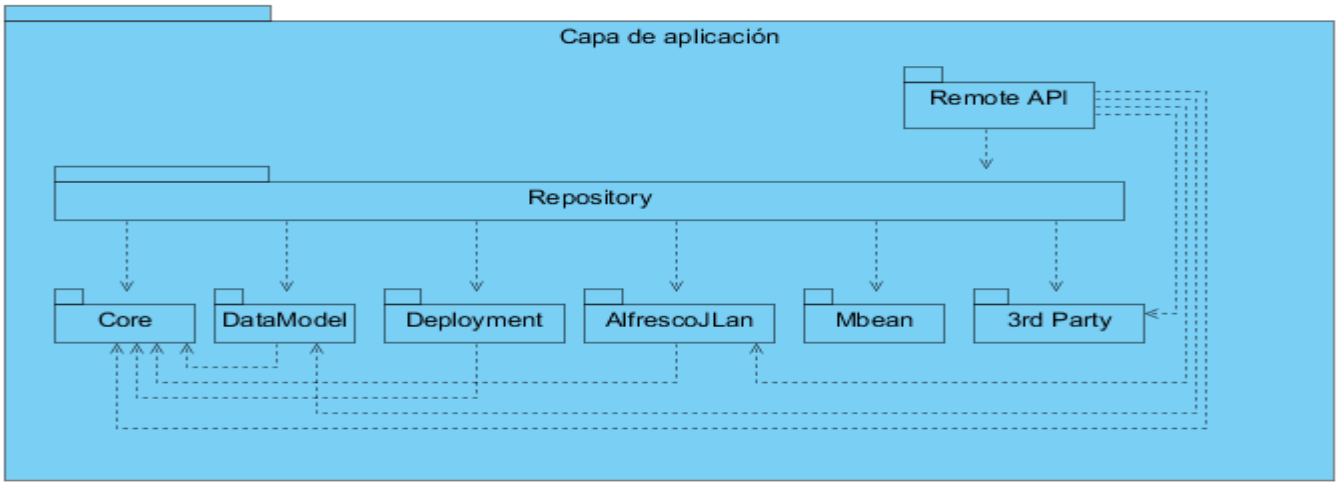

Figura 10: Vista de Implementación de la capa de aplicación.

Para evitar algunos tipos de modificaciones en el código fuente del núcleo, por las necesidades que puedan ocurrir en su desarrollo, se propone añadir cada cambio de forma individual mediante módulos que posean sus propias implementaciones e interfaces. Para este proceso Alfresco

brinda el uso de los Módulos Empaquetados de Alfresco (AMP). Un fichero AMP se puede mezclar con la aplicación *web* del núcleo propuesto mediante el uso de la Herramienta de Gestión de Módulos (MMT por sus siglas en inglés). De esta manera, se evita que el sistema vuelva a ser modificado internamente y con esto complicar la posibilidad de renovar cualquiera de sus componentes.

Los ficheros AMP están compuestos por una estructura de carpetas que es organizada luego por el MMT en la aplicación *web* (fichero de extensión war). Esta estructura se describe en la siguiente figura:

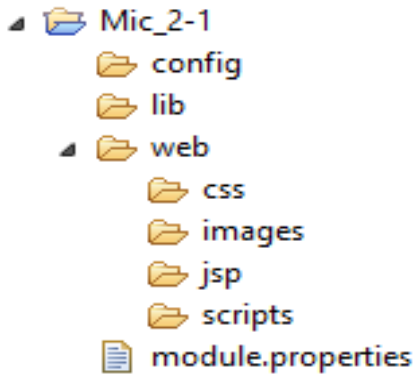

Figura 11: Estructura básica para un AMP.

Los directorios mostrados anteriormente son organizados dentro de la aplicación *web* de la siguiente forma:

- Directorio **config** se organiza en /WEB-INF/clases.
- Directorio lib se organiza en /WEB-INF/lib.
- Directorio web/jsp se organiza en /jsp.
- Directorio web/css se organiza en /css.
- Directorio web/images se organiza en /images.
- Directorio web/scripts se organiza en /scripts.

Finalmente, en el fichero **module.properties** se definen algunas propiedades del módulo como el ID y la versión.

2.5. Entorno de desarrollo

Cuando se realiza un desarrollo de *software* es importante planear e implementar un entorno de desarrollo desde el comienzo de todo el proyecto. De esta manera, todos los involucrados

conocerán cómo está organizado el sistema, cómo construirlo, desplegarlo y probar que esté en buen estado (BERGLJUNG, 2011).

2.5.1. Solución de integración continua

La implementación de una solución de integración continua (CI por sus siglas en inglés) en el desarrollo de un proyecto de *software*, le brinda la posibilidad a este último de capturar los problemas desde etapas tempranas (BERGLJUNG, 2011). Por ese motivo las extensiones con las que se trabaja pueden ser continuamente construidas, manejadas para realizarles pruebas y revisiones de cambios.

El diagrama que se muestra a continuación detalla el establecimiento de un entorno de integración continua para el desarrollo del GDA eXcriba.

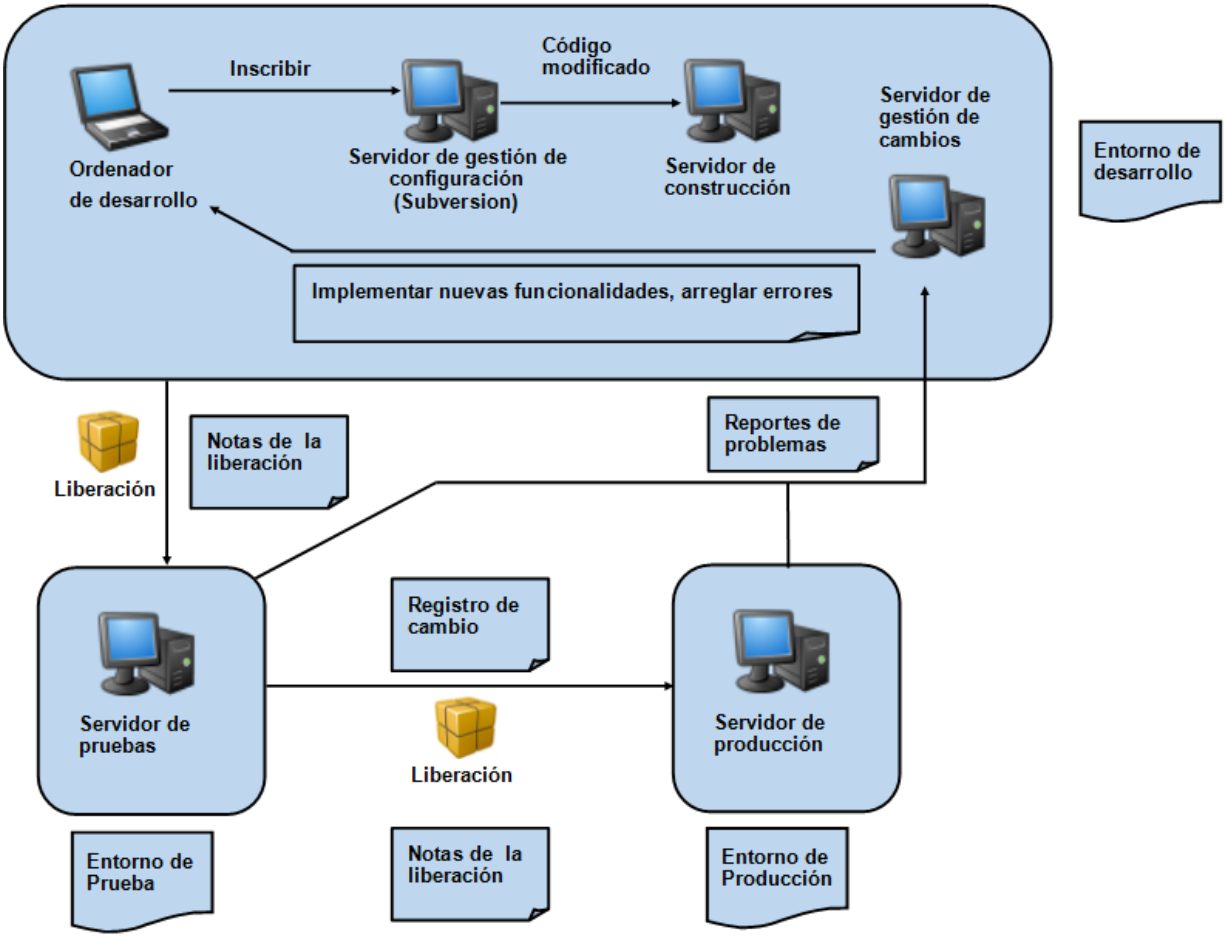

Figura 12: Entorno de desarrollo integrado para el núcleo del GDA eXcriba.

El diagrama anterior se interpreta de la siguiente forma:

1. Los programadores implementan nuevas funcionalidades o arreglan errores, luego construyen su núcleo localmente y realizan sus pruebas unitarias. Al comprobar que todo está en orden se actualizan los cambios en el servidor de gestión de configuración.
2. El servidor de gestión de configuración le avisa al servidor de construcción, el cual realiza las construcciones y pruebas automáticas al nuevo núcleo.
3. Los programadores despliegan el sistema eXcriba 3.0 de su ordenador y prueban que el listado de los requisitos funcionales fundamentales no presenta problemas (pruebas de humo (BERGLJUNG, 2011)). Si todo está en orden redactan la nota de la liberación que acompaña al módulo (ver Anexo1 y 2).
4. Se etiqueta la nueva liberación en el servidor de gestión de configuración. De esta manera, el código fuente del módulo estará disponible.
5. Los programadores despliegan el módulo en el entorno de prueba y entregan la nota de la liberación a los integradores y probadores del sistema.
6. Los probadores realizan todas las pruebas necesarias para que la liberación sea aceptada. En caso de ser detectado algún error, este es reportado mediante el uso del servidor de gestión de cambio y se vuelve a comenzar desde el paso 1. Por otro lado, si la liberación pasa de manera exitosa todas las pruebas, se continúa con el paso siguiente.
7. El arquitecto despliega la nueva versión del sistema en el entorno de producción y actualiza el registro de cambio (ver Anexo 3).
8. Cualquier involucrado con el sistema puede reportar cualquier error o no conformidad en el servidor de gestión de cambio.

2.6. Conclusiones

- La descomposición del ECM Alfresco permitió identificar cada uno de sus proyectos y qué función cumplían. De esta forma, fue posible realizar una selección de los elementos necesarios para conformar el núcleo propuesto en este capítulo.
- La implementación de un mecanismo de construcción mediante la herramienta Apache Ant garantizó la construcción de los componentes de Alfresco y eXcriba de manera separada en una misma aplicación web. De esta manera, se pueden renovar los componentes utilizados de Alfresco sin la pérdida de las implementaciones de eXcriba; logrando de esta forma el objetivo de la presente investigación.
- La implementación de un sistema de integración continua posibilitó reducir los procesos repetitivos realizados manualmente y los riesgos en el proceso de desarrollo del núcleo, además de brindar una mejor visibilidad del proyecto.

Capítulo 3
Validación de la arquitectura del núcleo del GDA eXcriba

En este capítulo se realiza la evaluación de la propuesta arquitectónica para el núcleo del GDA eXcriba.

Una de las realidades más importantes de la arquitectura de un sistema es que al conocerla se pueden mostrar las propiedades que va a poseer el *software*, incluso aunque este no exista. Del mismo modo, las decisiones de diseño que toman los arquitectos tienen efectos en el sistema que estos están construyendo, pero estos efectos pueden ser conocidos y predecibles (BASS, et al., 2003). Por ese motivo las estrategias y los patrones a usar sobre una arquitectura le proporcionan propiedades conocidas al sistema y pueden ser analizables. De esta manera, se puede afirmar que al obtener una arquitectura se pueden deducir los aspectos del sistema, aunque este no se haya construido.

Evaluar una arquitectura candidata antes de que se convierta en un proyecto aceptado es la vía más económica de evitar desastres. Con el uso de métodos estructurados de evaluación, se le brinda a la arquitectura bajos costos en la capacidad de mitigación de riesgos (BASS, et al., 2003). De esta manera, también se puede asegurar que la arquitectura es la correcta y no guiar las decisiones por “el buen sentido”.

3.1. Método de evaluación MECABIC

Para evaluar la propuesta arquitectónica de la presente investigación se propone el método de evaluación MECABIC. Este método está basado en ATAM y presenta como objetivo principal analizar y evaluar la calidad de una arquitectura de *software*, facilitando su aplicación sobre Arquitecturas de Software Basadas en Componentes (ASBC) (CARRASCOSO, et al., 2010). MECABIC adiciona en algunos de sus pasos un enfoque de integración de elementos del diseño,

seleccionados de la construcción de partes arquitectónicas propuesto por la Arquitectura Basada en Diseño (ABD por sus siglas en inglés).

Este método propone el uso de un árbol de utilidad inicial basado en el modelo de calidad ISO-9126 enmarcado para arquitecturas de *software* propuesto por Losavio. Los escenarios incluidos en este árbol son específicos para aplicaciones basadas en componentes. Esta adopción le permite a MECABIC concentrarse en características que dependen exclusivamente de la arquitectura.

3.1.1.Equipo de evaluación

Los involucrados en el equipo de evaluación se pueden agrupar en tres grupos. Estos se pueden mostrar de la siguiente manera (GONZÁLEZ, et al., 2005):

Equipo	Definición
Arquitectos	Responsables de generar y documentar una arquitectura de *software* para el sistema estudiado.
Evaluador	Integrado por personas especialistas en asuntos de calidad quienes guiarán el proceso de evaluación de la arquitectura.
Relacionados	Son las personas involucradas de alguna manera con el sistema: programadores, usuarios, gerentes, entre otros.

Tabla 5: Grupos participantes en el método MECABIC.

3.1.2.Instrumentos y técnicas

Se hace uso de un árbol de utilidad para realizar la especificación de la calidad, este posee como nodo raíz la bondad o utilidad del sistema. En el segundo nivel del árbol se localizan los atributos de calidad. Cada hoja del mismo es un escenario, que representa los mecanismos mediante los cuales extensas y ambiguas declaraciones de cualidades son posibles de evaluar. La generación del árbol de calidad incluye un paso que permite establecer prioridades. Para el análisis de la arquitectura se utiliza la técnica de escenarios, soportada por cuestionarios, para identificar lo que es señalado por los involucrados.

3.1.3.Fases

MECABIC está compuesto por cuatro fases las cuales se pueden desglosar de la siguiente forma:

1. La presentación es la primera fase, es donde se da a conocer el método, el sistema y su organización entre todos los grupos. Finalmente, se presenta cuál es la arquitectura que debe ser evaluada.
2. La investigación y análisis es la segunda fase, en ella se determina de qué manera se va a estudiar la arquitectura. Se le pide a los involucrados dentro de la evaluación, que expresen de una manera precisa qué escenarios de calidad se desean y se analiza si la arquitectura es apropiada o se requiere modificaciones para que lo sea.
3. La tercera fase es de prueba, en ella se realiza la revisión de la segunda fase. Por esa razón es necesaria para esta fase la participación de todos los grupos.
4. La presentación de los resultados se lleva a cabo en la cuarta y última fase. Al igual que la fase anterior, participan todos los grupos.

Una vez detalladas las fases se muestra la evaluación de la arquitectura propuesta:

3.1.3.1. Presentación

Paso 1. Presentación de MECABIC: El director de evaluación presenta el método a todos los participantes. Luego se explica el proceso que debe cumplir cada participante, se responde preguntas y se establecen las expectativas sobre las tareas restantes.

Paso 2. Presentación de las directrices del negocio: Se detallan las metas del negocio que motivan el esfuerzo y se aclara que se persiguen objetivos de tipo arquitectónico. De esta forma, se presentan como principales funcionalidades del sistema:

1. Almacenar contenido.
2. Crear contenido.
3. Eliminar contenido.
4. Modificar contenido.
5. Recepción del contenido.
6. Controlar el mantenimiento del contenido.
7. Controlar el uso del contenido.
8. Controlar la disposición del contenido.
9. Automatizar procesos de negocios.
10. Compartir documentos.
11. Buscar contenidos.
12. Indexar contenido.
13. Integración de aplicaciones empaquetadas.
14. Gestión de recursos digitales.

Tabla 6: Principales funcionalidades del sistema

Se propone como meta de los atributos de calidad que guían la arquitectura:

- Lograr la habilidad del sistema para cumplir el trabajo para el que fue previsto.
- Lograr la habilidad del sistema a conservarse activo a lo largo del tiempo.
- Lograr que el grado con el que el sistema cumpla sus funciones designadas, dentro de ciertas restricciones como velocidad, exactitud o uso de memoria, sea satisfactorio.
- Lograr la capacidad de someter al sistema a reparaciones y que se desarrollen de manera rápida y a bajo costo.
- Lograr la habilidad de efectuar cambios futuros al sistema sin grandes afectaciones a lo que se posee.
- Lograr la habilidad del sistema para ser ejecutado en diferentes ambientes computacionales.

Paso 3. Presentación de la arquitectura: El arquitecto explica a los involucrados la arquitectura a evaluar centrándose en cómo satisface los objetivos del negocio. En este paso se expone el contenido del capítulo anterior.

3.1.3.2. Investigación y análisis

Paso 1. Identificación de elementos de diseño: Los arquitectos identifican los enfoques arquitectónicos. La arquitectura debe ser evaluada completamente desde el sistema con sus componentes de mayor granularidad, hasta el mínimo nivel de granularidad que corresponde a los componentes. Describir las relaciones y dependencias que existen entre componentes, así como definir los conjuntos de decisiones que tengan influencia considerable sobre otros elementos de diseño. Estos pasos se encuentran detallados en los capítulos anteriores.

Paso 2. Generación del árbol de utilidad: Se obtienen los atributos de calidad que engloban la “utilidad” del sistema especificados en forma de escenarios. Se plasman los estímulos y respuestas, estableciendo la prioridad entre ellos. MECABIC brinda un árbol de utilidad inicial del cual se seleccionan un conjunto de escenarios de interés (GONZÁLEZ, et al., 2005):

Característica	Subcaracterística	Escenario
Funcionalidad	Interoperabilidad	El sistema posee componentes capaces de leer datos provenientes de otros sistemas.
		El sistema posee componentes capaces de producir datos para otros sistemas.
	Precisión	Los resultados ofrecidos por los componentes del sistema son exactos.
		La comunicación entre los componentes no altera la exactitud de los datos.
	Seguridad	El sistema detecta la actuación de un intruso e impide el acceso a los componentes que manejen información sensible.
		El sistema asegura que los componentes no pierden datos ante un ataque (interno o externo).
	Obediencia (*Compliance*)	Los componentes respetan un estándar de fiabilidad.
		La comunicación entre los componentes no viola los estándares de fiabilidad.
Fiabilidad	Madurez	Los componentes del sistema manejan entradas de datos incorrectas.
	Tolerancia a fallas	Todas las operaciones ejecutadas por los componentes se realizan correctamente bajo condiciones adversas.
	Capacidad de restablecimiento o recuperación	Ante problemas con el ambiente un subconjunto determinado de componentes puede continuar prestando sus servicios.

		Los componentes del sistema no fallan bajo ciertas condiciones especificadas.
Eficiencia	Tiempo de comportamiento	El sistema debe recibir los servicios de sus componentes en el transcurso de un tiempo indicado.
	Recursos utilizados	Los componentes deben compartir recursos adecuadamente.
		El sistema controla que ningún componente se quede sin recursos cuando lo necesita.
Mantenibilidad	Habilidad de cambio, estabilidad, prueba.	Es posible verificar el estado de los componentes del sistema.
		El sistema brinda facilidad para adaptar un componente.
		El sistema debe facilitar la sustitución/adaptación de un componente.
Portabilidad	Adaptabilidad	El sistema debe continuar funcionando correctamente aun cuando los servicios de los componentes provistos por el ambiente varíen.
	Capacidad de instalación	Los componentes pueden instalarse fácilmente en todos los ambientes donde debe funcionar.
	Co-existencia	Los componentes manejan adecuadamente los recursos compartidos del sistema

Tabla 7: Subconjunto del árbol de utilidad inicial propuesto por el método.

Paso 3. Seleccionar los escenarios iniciales a considerar: Luego de ser estudiada esta propuesta inicial, el grupo evaluador muestra los diferentes escenarios al resto de los involucrados y se decide cuáles son los elementos a incluir en el árbol de utilidad de la presente arquitectura.

Paso 4. Proposición de escenarios no contemplados: Se analiza la posibilidad de adicionar escenarios no contemplados dentro de esta propuesta inicial. En caso de existir alguno que cuente con un alto consenso de los involucrados, este se incluye directamente dentro del árbol de utilidad. Debido a que se evalúa un conjunto de características de interés no deberían estar presentes escenarios de calidad para todas las características de calidad conocidas.

Paso 5. Priorizar los escenarios de calidad: Una vez que se decide qué escenarios se incluyen en el árbol de utilidad se procede a definir la prioridad de cada uno como un par (X, Y), donde la X define el esfuerzo para satisfacer al escenario y la Y representa el riesgo que se corre al excluirlo. Los posibles valores que pueden tomar X y Y se dividen en Alto (A), Medio (M), Bajo (B). El árbol generado se toma como plan para el resto de la evaluación que realiza el método.

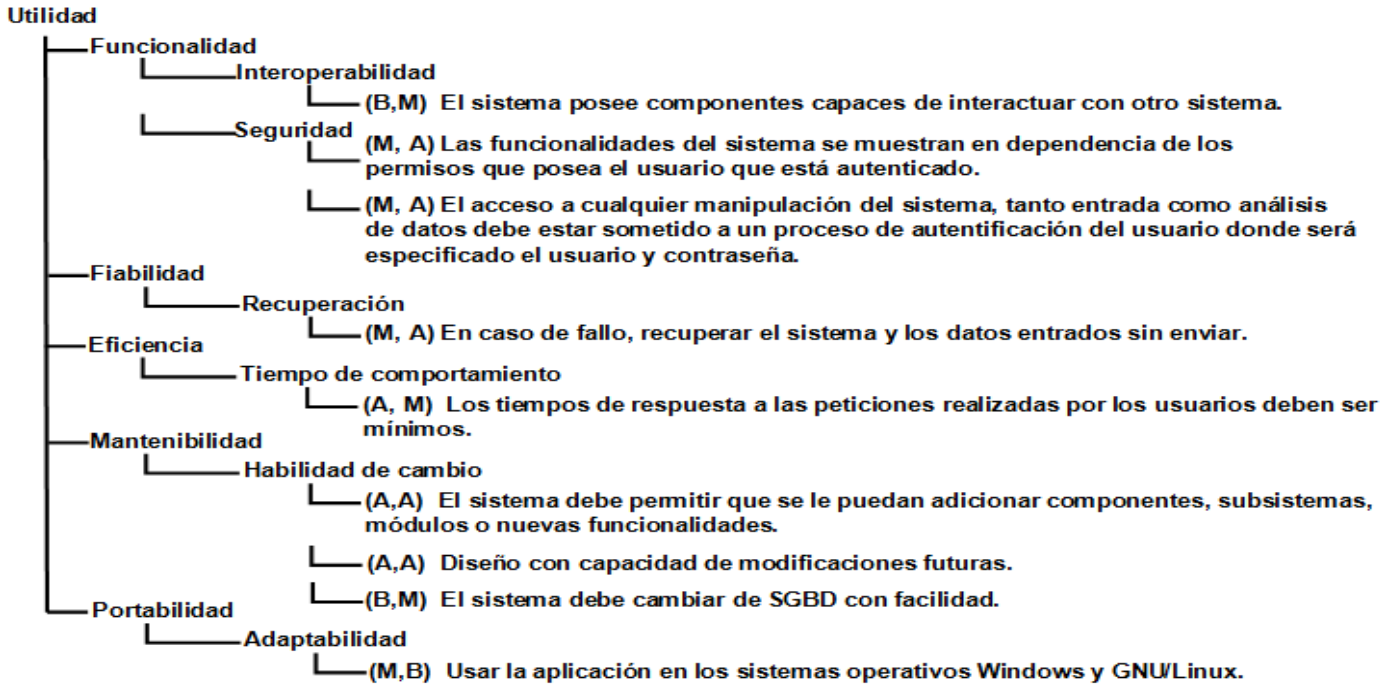

Figura 13: Árbol de utilidad

Paso 6. Análisis de elementos de diseño para ASBC: Se analizan los elementos de diseño identificados en el paso 4, para determinar cómo influyen sobre la realización de los escenarios de calidad presentes en el árbol de utilidad generado en el paso 5.

Paso 7. Análisis de los elementos de diseño estudiados en el paso 4: Los escenarios de calidad deben ser aplicados a la arquitectura que ha sido generada. Se debe preguntar si las decisiones tomadas permiten alcanzar los escenarios de calidad planteados. El equipo de evaluación se

orienta con las preguntas de análisis propuestas por el método para determinar decisiones sobre la arquitectura (GONZÁLEZ, et al., 2005):

Características	Sub-características	Preguntas del análisis
Funcionalidad	Precisión	¿Puede la comunicación entre los componentes introducir imprecisiones en los servicios ofrecidos por los componentes?
	Interoperabilidad	¿Dónde se encuentran los componentes que permiten al sistema interactuar con otros sistemas?
Fiabilidad	Madurez	¿Existen decisiones para minimizar el manejo incorrecto de datos en la comunicación entre componentes?
	Tolerancia a fallas	¿Cómo se detecta el funcionamiento incorrecto de un componente?
Eficiencia	Tiempo de comportamiento	¿Cómo es la relación entre el número de componentes de las diferentes partes de la arquitectura?
Mantenibilidad	Habilidad de cambio, estabilidad, prueba	¿Cómo se verifica el funcionamiento correcto de un componente?
		¿Cómo se verifica el estado de una comunicación entre componentes?
		¿Cómo se facilita el reemplazo de un componente?

Tabla 8: Tabla de preguntas de análisis propuestas por el método.

Preguntas del análisis

- **¿Dónde se encuentran los componentes que permiten al sistema interactuar con otros sistemas?**

El núcleo del eXcriba cuenta con un subsistema de comunicación para la capa de presentación llamado Remote-API. Este es el proyecto donde se debe dar soporte para que interactúe con otros sistemas.

- **¿Cómo se detecta el funcionamiento incorrecto de un componente?**

Se detecta mediante el uso de pruebas unitarias cuando se desarrolla el componente. Solamente se procede a integrar un componente al núcleo cuando su funcionamiento es correcto.

- **¿Cómo se facilita el reemplazo de un componente?**

El sistema permite el reemplazo de componentes mediante el uso de la herramienta ANT. En los archivos de construcción del sistema están definidos los objetivos de construcción de todos los componentes incluyendo al núcleo. De esta forma, atendiendo a las dependencias de los componentes se pueden reemplazar, eliminar o añadir componentes al núcleo.

- **¿Existe un mecanismo para determinar si todos los componentes del sistema se encuentran correctamente instalados?**

Mediante la prueba de humo (BERGLJUNG, 2011) se realizan pruebas unitarias a cada componente del núcleo. De esta manera, se puede comprobar que cada componente funciona correctamente antes de que el sistema se ponga en funcionamiento.

- **¿Existe un mecanismo para la gestión de dependencias de los componentes?**

No existe en estos momentos.

La segunda y cuarta pregunta detectó debilidades en la arquitectura del sistema, lo que conllevó a realizar una solución de integración continua. Esta solución se encuentra detallada en el **acápite 2.3.1** y da respuesta a las dos preguntas planteadas. Por otro lado, la quinta pregunta se mantiene como una debilidad de la arquitectura, puesto que la herramienta de construcción Apache Ant no posee un mecanismo para la gestión de dependencia. Esta debilidad queda reflejada como una recomendación para futuras investigaciones.

3.1.3.3. Prueba

Paso 1. Revisión del árbol de utilidad: Los escenarios del árbol que no hayan sido considerados, son colocados por los involucrados en el paquete de la tormenta de ideas, para volverlos a revisar. Los escenarios generados se comparan con la lista considerada inicialmente, llegando a la conclusión de que los escenarios evaluados son adecuados para el grupo de interesados en el sistema.

Paso 2. Revisión de los elementos de diseño definidos: El equipo evaluador estudia la forma en que los elementos del diseño promueven los escenarios propuestos por los involucrados.

3.1.3.4. Generación de una arquitectura final y reporte

Paso 1. Generación de acuerdos: En este paso los involucrados conocen los beneficios que pueden ser obtenidos mediante la alternativa que se ha estudiado. Por esa razón se genera una negociación crítica sobre la aceptación de la arquitectura. Si no es aceptada conlleva a la necesidad de replantear el núcleo propuesto.

Paso 2. Presentación de los resultados: Para finalizar la aplicación del método se presenta un resumen de su aplicación, de forma oral, donde se incluyen los documentos a partir de los cuales se inició la evaluación.

3.2. Conclusiones

- La evaluación propuesta posibilitó detectar debilidades en la arquitectura, permitiendo replantear su diseño e incluir elementos claves como un sistema de integración continua.
- Se evaluaron los escenarios propuestos por el método MECABIC, permitiendo formular un conjunto de preguntas del análisis que responden satisfactoriamente a la aceptación de la arquitectura propuesta.

Conclusiones Generales

Al finalizar la presente investigación se puede concluir que:

- Se logró una caracterización de los ECM que permitió seleccionar cinco de las siete áreas que los componen para conformar la propuesta arquitectónica de la investigación. Estas áreas fueron: el gestor de documentos, el gestor de procesos de negocios, la administración de documentos de archivo, el contenido social y los componentes extendidos.
- La implementación de un mecanismo de construcción mediante la herramienta Apache Ant garantizó la construcción de los componentes de Alfresco y eXcriba de manera separada en una misma aplicación *web*. De esta manera, se pueden renovar los componentes utilizados del Alfresco sin la pérdida de las implementaciones del eXcriba; logrando de esta forma el objetivo de la presente investigación.
- La evaluación propuesta posibilitó detectar debilidades en la arquitectura, permitiendo replantear su diseño e incluir un sistema de integración continua.

Recomendaciones

- Se recomienda incorporar a la arquitectura un diseño que centre mayor atención en atributos de calidad como el rendimiento y la disponibilidad.
- Se recomienda definir un mecanismo de gestión de dependencias ligado al sistema de integración continua propuesto.
- Con el objetivo de fortalecer la arquitectura se recomienda aplicar otros métodos de evaluación.
- Se recomienda instaurar el entorno de producción en servidores con prestaciones semejantes al entorno final que utilizará el cliente.

Glosario de Términos

ADL: Del inglés Architecture Description Language. Lenguaje de Descripción de la Arquitectura.

API: Del inglés Application Programming Interface. Programa de aplicación de interfaz.

CIFS: Del inglés Common Internet File System. Sistema de Fichero Común para Internet.

CMIS: Del inglés Content Management Interoperability Services. Servicios de Interoperabilidad de Gestión del Contenido.

ECM: Del inglés Enterprise Content Management. Gestión de Contenido Empresarial.

Entropía: Se concibe como una “medida del desorden o medida de la incertidumbre”.

FTP: Del inglés File Transfer Protocol. Protocolo de Transferencia de Archivos.

GB: Del Inglés gigabyte. Gigaocteto es un múltiplo del byte u octeto que equivale a 10 a la 9 bytes.

GDA: Gestor de Documentos Administrativos.

IDE: Del inglés Integrated Development Environment. Entorno de desarrollo integrado, se conoce como un *software* compuesto por un conjunto de herramientas de programación y provee un marco de trabajo amigable para la mayoría de los lenguajes de programación.

KB: Del inglés Kilobyte. Kiloocteto es un múltiplo del byte u octeto que equivale a 10 a la 3 bytes.

MB: Del inglés megabyte. Megaocteto es un múltiplo del byte u octeto que equivale a 10 a la 6 bytes.

Repositorio: Del lat. Repositorĭum, lugar donde se guarda algo.

RUP: Del inglés Rational Unified Process. Proceso Unificado Racional. Metodología de desarrollo de *software*.

SEI: Del inglés Software Engineering Institute. Instituto federal estadounidense de investigación y desarrollo, fundado por el congreso estadounidense en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de *software*. De esta manera, dar respuesta a los problemas de la programación e integración de los subsistemas de *software* en la construcción de complejos sistemas.

UCI: Universidad de las Ciencias Informáticas.

UML: Del inglés Unified Modeling Language. Lenguaje Unificado de Modelado. Es uno de los lenguajes de modelado más utilizados en la actualidad.

Bibliografía Referenciada

ALFRESCO. 2011. Arquitectura. *Ayuda de Alfresco.* [Online] Alfresco, 2011. [Cited: Enero 2, 2013.] http://docs.alfresco.com/4.2/index.jsp.

—. 2011. Is your content business-critical? It's time you took a look at Alfresco. *Alfresco Web site.* [En línea] Alfresco, 2011. [Citado el: enero 22, 2013.] http://www.alfresco.com/.

APACHE. 2013. Apache Ant - Welcome. *Apache Ant site.* [En línea] Apache Software Fundation, 5 21, 2013. [Citado el: 5 31, 2013.] http://ant.apache.org/.

ASSOCIATION FOR INFORMATION AND IMAGE MANAGEMENT. 2010. What is Document Management (DMS)? *aiim.* [En línea] AIIM, 2010. [Citado el: Enero 27, 2013.] http://www.aiim.org.

—. 2009. What is Electronic Records Management (ERM)? *aiim.* [En línea] AIIM, 2009. [Citado el: enero 10, 2013.] http://www.aiim.org.

—. 2011. What is Enterprise Content Management (ECM)? *aiim.* [En línea] AIIM, 2011. [Citado el: Enero 29, 2013.] http://www.aiim.org.

—. 2009. What is Web CMS (o WCM)? *aiim.* [En línea] AIIM, 2009. [Citado el: Enero 28, 2013.] http://www.aiim.org.

BANKER, R.D., et al. 1993. *Software complexity and maintenance costs, communications of the ACM.* s.l. : ACM, 1993.

BASS, Len, Clements, Paul and Kazman, Rick. 2003. *Software Architecture in Practice, Second Edition.* s.l. : Addison Wesley, 2003. 0-321-15495-9.

BERGLJUNG, Martin. 2011. *Alfresco 3 Business Solutions.* Birmingham : Packt, 2011. ISBN/978-1-849513-34-0.

BHARATHI, B. and Sridharan, D. 2009. *UML as an Architecture Description Language.* Chennai : International Journal of Recent Trends in Engineering, 2009. Vol. 1.

BILLY REYNOSO, Carlos. 2005. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : Microsoft, 2005.

—. 2004. *Introducción a la Arquitectura de Software.* Buenos Aires : Microsoft, 2004.

BOSCH, Jan. 2000. *Design and use of Software Architecture.* s.l. : Addison-Wesley, 2000.

BRITTON, C. and Bye, P. 2004. *IT Architectures and Middleware Strategies for Building Large Integrated Systems, Second edition.* Boston : Addison-Wesley, 2004.

BROOKS, F. 1975. *The Mythical Man–Month.* s.l. : Addison-Wesley, 1975.

BUSCHMANN, Frank, et al. 1996. *Pattern-oriented software architecture – A system of patterns.* s.l. : John Wiley & Sons, 1996.

BUSCHMANN, Frank, Henney, Kevlin and C. Schmidt, Douglas. 2007. *PATTERN-ORIENTED SOFTWARE ARCHITECTURE.* West Sussex : WILEY, 2007. 978-0-470-05902-9.

BYRON J., Williams and Jeffrey C., Carver. 2009. *Characterizing software architecture changes.* Ámsterdam : Elsevier, 2009.

CAMACHO, Erika, Cardeso, Fabio and Nuñez, Gabriel. 2004. *Arquitecturas de Software.* 2004.

CARRASCOSO, Yoan Arlet and Vega, Anisleydi. 2010. Procedimiento para la evaluación de arquitecturas de software basadas en componentes. *Gestiopolis Web site.* [Online] 2010. [Cited: octubre 21, 2012.] http://www.gestiopolis.com/administracion-estrategia/archivo/procedimiento-para-la-evolucion-de-las-arquitecturas.

CARUANA, David, et al. 2010. *Professional Alfresco.* Indianapolis : Wiley, 2010. 978-0-470-57104-0.

CLEMENTS, Paul. 1996. *A Survey of Architecture Description Languages.* Alemania : Proceedings of the International Workshop on Software Specification and Desing, 1996.

CLEMENTS, Paul, et al. 2002. *Documenting Software Architectures: Views and Beyond.* s.l. : Addison Wesley, 2002. ISBN/0-201-70372-6.

DAVID, Michel. 2011. *Propuesta arquitectónica para el desarrollo del Gestor de Documentos.* Tesis (Ingeniero en Ciencias Informáticas). Ciudad de la Habana, Cuba: Universidad de las Ciencias Informáticas : s.n., 2011. p. 33.

EMC. 2011. Universo Digital. Estudio patrocinado por EMC. *EMC Web site.* [En línea] 2011. [Citado el: diciembre 15, 2012.] http://www.emc.com.

FOWLER, Martin, et al. 2002. *Patterns of Enterprise Application Architecture.* s.l. : Addison Wesley, 2002. ISBN/0-321-12742-0.

GARLAN, David, et al. 1996. Architectural Style: An Object-Oriented Approach. [Online] febrero 1996. [Cited: octubre 28, 2012.] http://www.cs.cmu.edu/afs/cs/project/able/www/able/papers_bib.html.

GARTNER. 2012. Magic Quadrant for Enterprise Content Management. *Gartner Web site.* [En línea] octubre 18, 2012. [Citado el: enero 27, 2013.] http://www.gartner.com/. G00237781.

—. 2011. Market Share Analysis: Enterprise Content Management Software. *Gartner Web site.* [En línea] 2011. [Citado el: enero 20, 2013.] http://www.gartner.com/.

GONZÁLEZ, Aleksander, et al. 2005. *Método de Evaluación de Arquitecturas de Software Basadas en Componentes (MECABIC).* Colimia, Mexico : s.n., 2005.

HART, Laurence. 2007. Review: Patterns of Enterprise Application Architecture. *Word of Pie site.* [Online] Word of Pie, julio 19, 2007. [Cited: febrero 19, 2013.] http://wordofpie.com/2007/07/19/review-service-oriented-architecture-soa-compass/.

ISO. 2005. Norma ISO 15489, vol I. *ISO Web site.* [En línea] Organización Internacional de Normalización, 2005. [Citado el: febrero 2, 2013.] http://www.iso.org/.

JACOBSON, I., Booch, G. and Rumbaugh, J. 2000. *El PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE.* Madrid : ADDISON WESLEY, 2000. 84-7829-036-2.

KRUCHTEN, Philippe. 1995. Architectural Blueprints-The “4+1” View Model of Software Architecture. 1995, Vol. 12, 6.

LINDVALL, M., Tesoriero, R. and Costa, P. 2002. *Avoiding architectural degeneration: an evaluation process for software architecture.* s.l. : Proceedings of the Eighth IEEE Symposium on Software Metrics, 2002.

LOSAVIO, Francisca, et al. 2003. *Quality Characteristics for Software Architecture.* Zurich : ETH, 2003.

MUNKVOLD, B.E. and Päivärinta, T. 2005. *Enterprise Content Management: An Integrated Perspective on Information Management.* s.l. : Proc HICSS, 2005.

NORDHEIM, S. and Päivärinta, T. 2006. *Implementing enterprise content management: from evolution through strategy to contradictions out-of-the-box.* 2006.

NUXEO. 2010. Installing Nuxeo IDE. *Nuxeo Web site.* [En línea] Nuxeo, 2010. [Citado el: febrero 27, 2013.] http://doc.nuxeo.com.

PEREIRA, C.M. and Sousa, P. 2004. *A method to define an Enterprise Architecture using the Zachman Framework.* s.l. : Proc. SAC, 2004.

POTTS, Jeff. 2009. *Alfresco Developer Guide.* s.l. : Packt Publishing, 2009. 978-1-847193-11-7.

SHAW, Mary and Clements, Paul. 1996. *A field guide to Boxology: Preliminary classification of architectural styles for software systems.* Carnegie Mellon University : Computer Science Department and Software Engineering Institute, 1996.

WEINTRAUB, Alan. 2011. *The Forrester Wave: Enterprise Content Management, Q4 2011.* Cambridge : Forrester, 2011.

Bibliografía Consultada

ALFRESCO. 2011. Arquitectura. *Ayuda de Alfresco.* [Online] Alfresco, 2011. [Cited: Enero 2, 2013.] http://docs.alfresco.com/4.2/index.jsp.

—. 2011. Is your content business-critical? It's time you took a look at Alfresco. *Alfresco Web site.* [En línea] Alfresco, 2011. [Citado el: enero 22, 2013.] http://www.alfresco.com/.

APACHE. 2013. Apache Ant - Welcome. *Apache Ant site.* [En línea] Apache Software Fundation, 5 21, 2013. [Citado el: 5 31, 2013.] http://ant.apache.org/.

ASSOCIATION FOR INFORMATION AND IMAGE MANAGEMENT. 2010. What is Document Management (DMS)? *aiim.* [En línea] AIIM, 2010. [Citado el: Enero 27, 2013.] http://www.aiim.org.

—. 2009. What is Electronic Records Management (ERM)? *aiim.* [En línea] AIIM, 2009. [Citado el: enero 10, 2013.] http://www.aiim.org.

—. 2011. What is Enterprise Content Management (ECM)? *aiim.* [En línea] AIIM, 2011. [Citado el: Enero 29, 2013.] http://www.aiim.org.

—. 2009. What is Web CMS (o WCM)? *aiim.* [En línea] AIIM, 2009. [Citado el: Enero 28, 2013.] http://www.aiim.org.

BANKER, R.D., et al. 1993. *Software complexity and maintenance costs, communications of the ACM.* s.l. : ACM, 1993.

BASS, Len, Clements, Paul and Kazman, Rick. 2003. *Software Architecture in Practice, Second Edition.* s.l. : Addison Wesley, 2003. 0-321-15495-9.

BERGLJUNG, Martin. 2011. *Alfresco 3 Business Solutions.* Birmingham : Packt, 2011. ISBN/978-1-849513-34-0.

BHARATHI, B. and Sridharan, D. 2009. *UML as an Architecture Description Language.* Chennai : International Journal of Recent Trends in Engineering, 2009. Vol. 1.

BILLY REYNOSO, Carlos. 2005. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : Microsoft, 2005.

—. 2004. *Introducción a la Arquitectura de Software.* Buenos Aires : Microsoft, 2004.

BOSCH, Jan. 2000. *Design and use of Software Architecture.* s.l. : Addison-Wesley, 2000.

BRITTON, C. and Bye, P. 2004. *IT Architectures and Middleware Strategies for Building Large Integrated Systems, Second edition.* Boston : Addison-Wesley, 2004.

BROOKS, F. 1975. *The Mythical Man–Month.* s.l. : Addison-Wesley, 1975.

BUSCHMANN, Frank, et al. 1996. *Pattern-oriented software architecture – A system of patterns.* s.l. : John Wiley & Sons, 1996.

BUSCHMANN, Frank, Henney, Kevlin and C. Schmidt, Douglas. 2007. *PATTERN-ORIENTED SOFTWARE ARCHITECTURE.* West Sussex : WILEY, 2007. 978-0-470-05902-9.

BYRON J., Williams and Jeffrey C., Carver. 2009. *Characterizing software architecture changes.* Ámsterdam : Elsevier, 2009.

CAMACHO, Erika, Cardeso, Fabio and Nuñez, Gabriel. 2004. *Arquitecturas de Software.* 2004.

CARRASCOSO, Yoan Arlet and Vega, Anisleydi. 2010. Procedimiento para la evaluación de arquitecturas de software basadas en componentes. *Gestiopolis Web site.* [Online] 2010. [Cited: octubre 21, 2012.] http://www.gestiopolis.com/administracion-estrategia/archivo/procedimiento-para-la-evolucion-de-las-arquitecturas.

CARUANA, David, et al. 2010. *Professional Alfresco.* Indianapolis : Wiley, 2010. 978-0-470-57104-0.

CLEMENTS, Paul. 1996. *A Survey of Architecture Description Languages.* Alemania : Proceedings of the International Workshop on Software Specification and Desing, 1996.

CLEMENTS, Paul, et al. 2002. *Documenting Software Architectures: Views and Beyond.* s.l. : Addison Wesley, 2002. ISBN/0-201-70372-6.

DAVID, Michel. 2011. *Propuesta arquitectónica para el desarrollo del Gestor de Documentos.* Tesis (Ingeniero en Ciencias Informáticas). Ciudad de la Habana, Cuba: Universidad de las Ciencias Informáticas : s.n., 2011. p. 33.

EMC. 2011. Universo Digital. Estudio patrocinado por EMC. *EMC Web site.* [En línea] 2011. [Citado el: diciembre 15, 2012.] http://www.emc.com.

FOWLER, Martin, et al. 2002. *Patterns of Enterprise Application Architecture.* s.l. : Addison Wesley, 2002. ISBN/0-321-12742-0.

GARLAN, David, et al. 1996. Architectural Style: An Object-Oriented Approach. [Online] febrero 1996. [Cited: octubre 28, 2012.] http://www.cs.cmu.edu/afs/cs/project/able/www/able/papers_bib.html.

GARTNER. 2012. Magic Quadrant for Enterprise Content Management. *Gartner Web site.* [En línea] octubre 18, 2012. [Citado el: enero 27, 2013.] http://www.gartner.com/. G00237781.

—. 2011. Market Share Analysis: Enterprise Content Management Software. *Gartner Web site.* [En línea] 2011. [Citado el: enero 20, 2013.] http://www.gartner.com/.

GONZÁLEZ, Aleksander, et al. 2005. *Método de Evaluación de Arquitecturas de Software Basadas en Componentes (MECABIC).* Colimia, Mexico : s.n., 2005.

HART, Laurence. 2007. Review: Patterns of Enterprise Application Architecture. *Word of Pie site.* [Online] Word of Pie, julio 19, 2007. [Cited: febrero 19, 2013.] http://wordofpie.com/2007/07/19/review-service-oriented-architecture-soa-compass/.

ISO. 2005. Norma ISO 15489, vol I. *ISO Web site.* [En línea] Organización Internacional de Normalización, 2005. [Citado el: febrero 2, 2013.] http://www.iso.org/.

JACOBSON, I., Booch, G. and Rumbaugh, J. 2000. *El PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE.* Madrid : ADDISON WESLEY, 2000. 84-7829-036-2.

KRUCHTEN, Philippe. 1995. Architectural Blueprints-The “4+1” View Model of Software Architecture. 1995, Vol. 12, 6.

LINDVALL, M., Tesoriero, R. and Costa, P. 2002. *Avoiding architectural degeneration: an evaluation process for software architecture.* s.l. : Proceedings of the Eighth IEEE Symposium on Software Metrics, 2002.

LOSAVIO, Francisca, et al. 2003. *Quality Characteristics for Software Architecture.* Zurich : ETH, 2003.

MUNKVOLD, B.E. and Päivärinta, T. 2005. *Enterprise Content Management: An Integrated Perspective on Information Management.* s.l. : Proc HICSS, 2005.

NORDHEIM, S. and Päivärinta, T. 2006. *Implementing enterprise content management: from evolution through strategy to contradictions out-of-the-box.* 2006.

NUXEO. 2010. Installing Nuxeo IDE. *Nuxeo Web site.* [En línea] Nuxeo, 2010. [Citado el: febrero 27, 2013.] http://doc.nuxeo.com.

PEREIRA, C.M. and Sousa, P. 2004. *A method to define an Enterprise Architecture using the Zachman Framework.* s.l. : Proc. SAC, 2004.

POTTS, Jeff. 2009. *Alfresco Developer Guide.* s.l. : Packt Publishing, 2009. 978-1-847193-11-7.

SHAW, Mary and Clements, Paul. 1996. *A field guide to Boxology: Preliminary classification of architectural styles for software systems.* Carnegie Mellon University : Computer Science Department and Software Engineering Institute, 1996.

WEINTRAUB, Alan. 2011. *The Forrester Wave: Enterprise Content Management, Q4 2011.* Cambridge : Forrester, 2011.

Anexos

Anexo 1: Notas de la liberación

Nota de liberación	
Nombre de la liberación:	
Versión:	
Autor:	
Fecha:	
Cliente	
Descripción de la liberación:	
Errores corregidos:	
Ficheros:	
Fecha de llegada al entorno de prueba:	
Fecha de llegada al entorno de producción:	

Anexo 2: Ejemplo práctico de una nota de la liberación

Nota de liberación	
Nombre de la liberación:	Personalización del núcleo para el MIC
Versión:	2.1
Autor:	Oscar Daniel Torres
Fecha:	5/16/2013
Cliente:	MIC
Descripción de la liberación: **Actualización del Modelo de Contenido adicionando el tipo de contenido Planilla de Reunión del PCC. Es posible de esta forma realizar búsquedas para este tipo de contenido.**	
Errores corregidos:	**-**
Ficheros:	Mic_2-1.amp
Fecha de llegada al entorno de prueba:	5/17/2013
Fecha de llegada al entorno de producción:	5/20/2013

Anexo 3: Registro de cambio

Registro de cambio <Nombre>		
Fecha	**Descripción**	**Autor**

Anexo 4: Ejemplo práctico de un registro de cambio

Registro de cambio para personalización en el MIC		
Fecha	**Descripción**	**Autor**
5/10/2013	Actualización de los modelos de contenidos	Oscar Daniel Torres
5/14/2013	Corrección de errores 34,35	Michel David