R-GAC: Repositorio para la Gestión de Activos basada en Características en el Marco del Proceso Small SPL

Monografía de Trabajo de Grado para Optar el Título de Ingeniero de Sistemas

Diego Javier García Buitrón

Director: PhD. Julio Ariel Hurtado

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Popayán, Octubre de 2015

CONTENIDO

LISTA DE FIGURAS

1. Introducción

1.1 Motivación

Las organizaciones de desarrollo de software de hoy, enfrentan el reto de ser altamente competitivas en términos de productividad y calidad[1], lo que ha hecho que las empresas miren la mejora de sus procesos de software y la innovación técnica en los mismos, como una de las estrategias clave para el éxito[2]. Sin embargo todas estas iniciativas requieren de esfuerzos y costos altos, y más aún en el contexto de las pequeñas empresas de software. Una de estas estrategias consiste en adquirir ventajas sobre sus competidores a través del enfoque de Línea de Productos de Software (SPL). Una SPL es una estrategia donde se combina la mejora de procesos y la innovación en el enfoque productivo, teniendo como centro la reutilización de software en forma planificada[3]. En una línea de productos, un conjunto de productos similares son desarrollados a partir de una misma línea base que se desarrolla a través de un proceso de ingeniería de dominio. En la línea base se tienen activos, procesos de apoyo y estructuras de organización para el desarrollo de productos software y de esta manera lograr los objetivos del negocio. La ventaja es que los productos no se desarrollan desde cero sino a partir de una infraestructura lo que beneficia la calidad de los productos, se reduce el tamaño del código a través de la eliminación de código duplicado, reducción del tiempo de desarrollo del producto mejorando la productividad de los equipos.

1.2 Contexto del Problema

Adoptar el enfoque SPL es una tarea compleja para las organizaciones de software[4], debido al enorme esfuerzo que requiere dedicarse a las actividades de reutilización de software y a la inversión inicial requerida para su adopción. La construcción de la base de activos requiere de un gran esfuerzo inicial para lograr que los artefactos técnicos y de gestión reutilizables, permitan sacar provecho de la similitud entre los productos y, al mismo tiempo, apoyar la variabilidad entre los mismos. Existen algunas barreras que complican la adopción del enfoque SPL, por ejemplo, procesos de desarrollo incompatibles y la falta de conocimientos en la construcción ingeniería de líneas de productos por parte de las organizaciones. Respecto al talento humano se tienen, entre otras, la resistencia cultural, la falta de apoyo a la gestión, las prácticas incompatibles de adquisición, la falta de gestión disciplinada, la falta de una visión de la línea de productos, la falta de un plan de adopción de la línea de productos, y las expectativas poco realistas. En el año 2012 Camacho y Hurtado[5] realizaron un estudio de viabilidad para la adopción del enfoque de líneas de productos en las empresas de la ciudad de Popayán. En dicho estudio, de la población total, el 19,35% de las empresas tienen una gran oportunidad de aprovechar el enfoque debido a que tiene las condiciones de negocio favorables. Un 45,16% no tiene las condiciones del todo favorables, pero tienen la oportunidad de focalizar sus mercados si su realidad lo permite, antes de incursionar en el enfoque de SPL. Para el porcentaje de las empresas

que el enfoque es viable (64,51%), desde el punto de vista técnico resulta poco factible debido en gran parte a las improvisadas técnicas para la reutilización. Por lo anterior, el grupo IDIS ha venido trabajando los últimos años en la adopción del enfoque SPL a través de Small SPL[6], un modelo de proceso de desarrollo de software enfocado en líneas de productos para el manejo de software que puede ser reutilizado por las pequeñas organizaciones. En Small SPL, los activos de software son desarrollados usando un enfoque de análisis y diseño del dominio basado en características (features), los cuales son reutilizados en la construcción de un producto, el cual es caracterizado para su planificación. El proceso de gestión de activos en Small SPL facilita la organización, registro y reutilización de activos tomando como referente los requerimientos y el alcance de un conjunto de sistemas similares. Un común inhibidor a la empresa de reutilización es a menudo la falta de la visibilidad de los activos reutilizables dentro de la comunidad de desarrolladores[18]. Dada la importancia del proceso de gestión de activos en Small SPL, es necesario brindar el soporte conceptual e informático suficiente para garantizar una adecuada administración y reutilización de los activos de software. Los repositorios presentes en empresas que han adoptado en el enfoque SPL son específicos a cada organización, no están disponibles a la comunidad, y si se publican reportes, no se publican los detalles de su implementación y uso, y dada la reciente aparición de Small SPL, es necesario pensar conceptual y tecnológicamente su repositorio con el fin de apoyar su utilización.

1.3 Pregunta de Investigación

Las particularidades del proceso Small SPL requiere de un repositorio que facilite la gestión de activos de una manera simple, sencilla y adecuada para realizar la identificación, desarrollo, documentación, almacenamiento y búsqueda por parte de los diferentes roles definidos en Small SPL[6]. Considerando esta problemática se plantea la siguiente pregunta de investigación: "¿**Cómo soportar conceptual y tecnológicamente las actividades de almacenamiento, búsqueda y recuperación de activos de software a partir del modelo de características, con el fin de hacer aplicables las prácticas de la gestión de activos en proyectos de software que siguen el proceso Small SPL**?".

1.4 Justificación

Si la gestión de activos no cuenta con herramientas disponibles para la producción de software SPL, las empresas deberán construir su propia infraestructura, lo que resulta en una barrera de adopción del enfoque en las pequeñas organizaciones[6]. Los repositorios digitales y las herramientas de gestión de la configuración no consideran la variabilidad planificada (controlada) como un elemento clave para la estructuración y derivación de varios productos de una misma familia, aunque resultan útiles para apoyar la gestión de varios productos individuales y de sus variantes a nivel de versión. La gestión documental es parte de la solución, pero en términos generales no resuelve la gestión de activos en el marco de líneas de productos, a menos que se aborde una solución específica,

particularmente el tema de la variabilidad planificada de activos[1] y menos aún, cuando estos activos no necesariamente son documentos[6]. En nuestra región se encuentran pocos estudios enfocados a pequeñas empresas que ayuden a orientar el camino a seguir para adoptar SPL por lo que se hace urgente apoyar su adopción con el fin de incrementar su competitividad. La presente propuesta ofrece un repositorio para el manejo activos de software en el contexto de Small SPL basada en características de manera sistematizada con el fin de facilitar a la industria de software regional la aplicación de este proceso de desarrollo[5][6]. A nivel investigativo permite fortalecer el modelo Small SPL soportándolo a través de una solución tecnológica para la gestión de activos, lo que permitirá fortalecer los estudios de aplicabilidad del modelo en pequeñas organizaciones. A nivel académico, contar con un repositorio permitirá seguir enseñando y aplicando el modelo Small SPL en cursos, por lo que se convierte en una herramienta de soporte a los procesos pedagógicos de la enseñanza de los tópicos avanzados de la ingeniería de software[6].

1.5 Objetivos de este Trabajo de Grado

1.5.1 Objetivo General

- Definir un repositorio para la gestión de activos de software basada en las características, con el fin de facilitar la identificación, desarrollo, documentación, almacenamiento, búsqueda y uso de activos de software reutilizables en el contexto del modelo de proceso Small SPL.

1.5.2 Objetivos Específicos

- Determinar las características técnicas y de gestión de los repositorios de activos de software más representativos reportados por la literatura.
- Caracterizar desde el punto de vista técnico y de gestión, los repositorios de activos software más representativos reportados por la literatura.
- Implementar y desplegar el repositorio basado en características como aplicación web.
- Evaluar el repositorio en un estudio de caso embebido[2] con tres unidades de análisis en el contexto de un curso avanzado de pregrado en ingeniería de sistemas siguiendo el diseño, ejecución y reporte de casos propuesto por Runeson[92].

[1] Variabilidad de software es la capacidad de un sistema software o artefacto de ser cambiado, personalizado o configurado para uso en un contexto particular.

[2] Un estudio de caso embebido, de acuerdo con Runeson es un estudio de caso que incluye varias unidades de análisis.

2 Marco Teórico

La reutilización de software es una de las estrategias más utilizadas para acelerar la producción de software. Como tal “La reutilización de software es el proceso de implementar o actualizar sistemas de software usando activos de software existentes”[7]. Para Sametinger[8] y Frakes[9] la "reutilización de software es el proceso de crear sistemas de software a partir de software existente, en lugar de desarrollarlo desde el comienzo" incluyendo conocimiento, procesos, metodologías o componentes de software que ya existente para adaptarlo a una nueva necesidad.

Una dificultad de las empresas es la falta de cultura y conocimientos acerca de las prácticas de reutilización. Esto requiere de una aproximación sistemática para reutilizar en una organización, cual debe involucrar los factores clave tales como: la estructura de una organización, prácticas de desarrollo, gestión, educación y promoción[10]. De acuerdo a Frakes[11], se distinguen dos categorías generales de reutilización que son por composición y por generación. En la categoría por composición se orienta a reutilizar productos y exalta la creación de nuevo software a partir de componentes almacenados en bibliotecas o repositorios. La categoría de generalización se orienta a la reutilización del conocimiento para sistematizar la transformación de un artefacto en un nivel de abstracción a otro (o de un dominio a otro). Las tareas de reutilización se orientan desde dos aspectos metodológicos: la visión proveedor y la visión del integrador, y que se conocen como desarrollo para reutilizar y desarrollo con reutilización respectivamente. La visión proveedor, resalta la necesidad de desarrollar componentes con la suficiente calidad para ser incluidos en un repositorio de reutilización. La visión de integrador, destaca la necesidad de utilizar conocimiento existente para satisfacer los requerimientos en la construcción de un nuevo sistema.

De acuerdo a Sametinger[8] y Prieto-Díaz[12], entre las principales ventajas de la reutilización de software se tiene la reducción de los tiempos de desarrollo, el aumento de la calidad de los productos, de la productividad y una mejora substancial de las actividades de mantenimiento y soporte. Por otro lado, Sametinger[8] y Prieto-Díaz[12] resaltan las principales dificultades que presentan al momento de abordar de la reutilización de software: no se realiza en forma planificada, no se le da la prioridad, poco conocimiento en la reutilización, escasa formación de los ingenieros de software en el área, metodologías no adecuadas al enfoque de reutilización y los altos costos de adopción.

2.1 Activos de Software Reutilizables

Un activo (asset en inglés)[13], de software reutilizable (de ahora en adelante activos de software) es un producto diseñado para ser utilizado de forma periódica en el desarrollo de muchos sistemas y aplicaciones, como por ejemplo algoritmos, patrones de diseño,

arquitecturas de software, esquemas de base de datos, especificaciones de requerimientos, elementos de diseño y pruebas, etc. Durante el desarrollo de aplicaciones los activos son reutilizados a través de distintas estrategias. A mayor nivel de abstracción de los activos, es más difícil reutilizarlos, sin embargo, sus beneficios son mucho mayores. Según Withey[14], un activo de software es una descripción de una solución parcial (como un componente o un documento de diseño) o conocimiento (como puede ser una base de datos de requisitos o procedimientos de prueba) que los ingenieros utilizan para construir o modificar productos software.

Los activos de software normalmente comparten características que influyen en su reutilización, que según Cybulsky[15] las cuales son:

- ***Transferibles:*** es posible transferir el activo de software a un entorno o dominio de problema diferente al que en origen fue creado. Esto implica que debe ser lo más independiente posible, con pocas dependencias de implementación, abstracto y bien parametrizado.
- ***Expresivos:*** los activos de software están clasificados en un nivel de abstracción adecuado y expresan una utilidad general que los hace susceptibles de ser reutilizados en diferentes contextos o ser aplicados en una amplia variedad de áreas de aplicación.
- ***Definidos:*** están construidos y documentados con un claro propósito, sus características y limitaciones son fácilmente identificables, sus interfaces, dependencias externas y entornos de operación están especificados, y cualquier otro requisito existente se encuentra perfecta y explícitamente definido.
- ***Formal:*** los activos de software debieran poder ser descritos a algún nivel de abstracción mediante una notación formal o semi-formal, de manera que pudiera existir algún mecanismo para poder verificar su corrección, predecir la violación de integridad de sus restricciones a la hora de integrarlo con otros activos de software o asegurar el nivel de compleción de un producto software construido con activos de software.
- ***Auto-contenidos:*** los activos de software que encierran una única idea son más fáciles de entender, tienen menos dependencias con factores externos (ya sean de entorno o de implementación), tienen unas interfaces fáciles de utilizar, son fáciles de extender, adaptar y mantener.
- ***Independientes del lenguaje:*** los elementos software reutilizables de alto nivel de abstracción deben omitir los detalles de implementación propios de los lenguajes de programación. Esto es, los elementos software reutilizables debieran ser descritos en términos de formalismos de especificación, o de forma que las soluciones de bajo nivel pudieran ser utilizadas por una amplia variedad de lenguajes de programación sobre una plataforma de implementación dada.
- ***Verificables:*** los activos de software deben ser fáciles de probar por los encargados de su mantenimiento y, lo que es más importante, por los usuarios que los utilicen en sus desarrollos con reutilización.

- ***Simples:*** Las interfaces pequeñas y sencillas ayudan a la reutilización de los activos de software, así como a su comprensión.
- ***Fáciles de cambiar:*** Ciertos tipos de problemas requieren activos de software que adopten nuevas especificaciones sin que ello provoque una gran cantidad de efectos laterales.
- ***Aditivos:*** esto es, que sea posible integrarlo con otros activos de software para formar elementos reutilizables compuestos, sin tener que realizar excesivas modificaciones y sin sufrir efectos laterales adversos. Esta propiedad lleva a manejar elementos reutilizables de diferente granularidad; así se tienen los elementos reutilizables de ***grano fino***, que son atómicos, es decir no están compuestos por otros, y que serán referenciados a lo largo de este trabajo como activos de software, y los elementos reutilizables de ***grano grueso***, que serán aquellos que están formados por un conjunto de activos de software.
- ***Computacionalmente representables:*** aquellos elementos software reutilizables que pueden ser descritos en términos de los valores de unos determinados atributos computacionales, que pueden ser fácilmente descompuestos en partes representables por un computador, que pueden ser accedidos, analizados, manipulados y posiblemente modificados por procesos basados en computador, tienen un claro potencial para formar parte de una biblioteca flexible de elementos reutilizables. Estos activos de software pueden ser *fácilmente buscados, recuperados, interpretados, modificados y finalmente integrados en sistemas software* de mayor entidad.
- ***Capaces de representar datos y comportamiento:*** Deben ser capaces de encapsular sus estructuras de datos y su lógica hasta un grano fino de detalle, de forma que se aumente la cohesión y se reduzca el acoplamiento por dependencia de datos comunes.

2.2 Repositorios de Activos de software

Un repositorio de software, es un conjunto de activos de software que son mantenidos por una organización para una posible exploración y consulta[17]. Los repositorios de activos de software son utilizados principalmente para tener un acceso rápido a componentes previamente construidos, durante el desarrollo y mantenimiento de productos software, facilitando la ***reutilización de componentes***. Es esencial la existencia de los repositorios de activos de software para permitir a los desarrolladores y a otros ingenieros de software localizar, recuperar comparar y mantener componentes de software reutilizables[18]. Según Jiang Guo y Luqi[19], las operaciones de los repositorios de activos son la localización de activos, el esquema de clasificación y la identificación de activos.

Existen algunos repositorios conocidos de tipo comercial tales como los siguientes:

- **+1Reuse Repository**[20]: Fue desarrollado por +1 Software Engineering Co. en California. En la actualidad se ejecuta en plataformas de estaciones de trabajo Sun. El sistema operativo es Solaris. GUI se basa en OpenWindows, Motif y CDE.

+1Reuse Repository soporta repositorios de reutilización creados y mantenidos por los usuarios, los cuales son "filtrados" bajo estrictos controles de calidad para soportar un enfoque de reutilización selectiva. La reutilización selectiva permite la reutilización o re-ingeniería de cualquier sub-modelo de un proyecto +1Environment existente. En cierto sentido, cada proyecto +1Environment es una biblioteca de reutilización. La reutilización selectiva mejora significativamente la capacidad de un usuario para reutilizar todo el código fuente y la documentación de todos los proyectos anteriores a cualquier nivel de granularidad. El sistema +Reuse soporta la reutilización de: diseño, documentación, código fuente, archivos de cabecera, casos de prueba, scripts de shell de ensayo, resultados de ensayos previstos, e información de modelado. Todo el código fuente diseñado o desarrollado utilizando el +1Environment puede ser reutilizado. +1Reuse aborda aspectos de reutilización tales como la reutilización de código fuente bajo la administración de configuración y nombres de archivo duplicados.

También +1Reuse admite tres formas de reutilización: Biblioteca de Reutilización definida por el Usuario, Biblioteca de Reutilización Filtrada y Reutilización Selectiva. Puesto que la productividad de un programador se puede aumentar mediante la reutilización de código y la documentación existente, +1Reuse ayuda a que todo el código fuente, documentación, archivos de cabecera y los archivos de prueba sea reutilizable bajo el soporte de sub-modelos. Después de que un sub-modelo ha sido seleccionado, +1Reuse copia el sub-modelo y sus archivos asociados al nuevo proyecto y ayuda a resolver una serie de problemas que puedan surgir (por ejemplo, los nombres de los archivos idénticos y los archivos seleccionados en virtud de la gestión de configuración).

- **Sistema de Gestión de Bibliotecas de Activos de Software**[21]: Asset Library Management System **(SALMS)** es un sistema para clasificar, describir y consultar los activos reutilizables. SALMS es un producto de software que proporciona mecanismos de clasificación, descripción y búsqueda de activos de software. Se llena el vacío entre el desarrollo de actividades “para la reutilización” (la construcción, adquisición o reestructuración de los activos reutilizables) y el desarrollo de las actividades “con reutilización” (utilizando activos reutilizables en la creación de nuevos productos de software). Desempeña un papel central en la implementación de un programa de reutilización de la empresa. Además, SALMS proporciona funciones para la actividad de gestión de requisitos, y para la creación y la gestión de la biblioteca técnica de la empresa. SALMS puede trabajar en forma distribuida a través de estaciones de trabajo UNIX del cliente y por lo tanto, dando accesibilidad a todos los desarrolladores dentro de una organización de software, a través de una interface de usuario web. En SALMS, un activo puede ser visto como una colección de artefactos producidos a lo largo del ciclo de vida, como requisitos, modelos de arquitectura, especificaciones de diseño, código fuente, o la prueba de scripts.
- **Automated Software Reuse Repository (ASRR)**[22]: El Repositorio de reutilización del software automático (ASRR) proporciona a los usuarios un

repositorio de datos con información de reutilización. Se compone de dos partes principales, la herramienta de administración y el repositorio de reutilización. En la parte de la administración de la herramienta el usuario realiza funciones administrativas como: la capacidad de agregar, eliminar o cambiar usuarios y sus atributos. Los atributos son los siguientes: los niveles de seguridad, grupo y permisos de seguridad para agregar, editar y eliminar los módulos. El repositorio de reutilización permite al usuario cargar módulos y almacenarlos en un repositorio de búsqueda. El ASRR proporciona las siguientes funciones:

- Programa de control de acceso. Proporciona control de acceso completa para el ASRR.

- Protección. El ASRR puede limitar edición de un usuario, borrar, visualizar, adicionar, cargar y descargar módulos permitidos a través de la porción de administración de la herramienta.

- Seguridad. La herramienta ASRR proporciona seguridad adicional para los usuarios inactivos por el registro fuera de la ASRR después de un período de 30 minutos de inactividad.

- Fácil acceso a reutilizar elementos. La herramienta permite a los usuarios registrados ASRR flexibilidad en la búsqueda de elementos de reutilización en el depósito de reutilización al permitir que los usuarios realicen la búsqueda de cadenas de palabras con " not ", " or", o "and" en la búsqueda.

- Reutilización de información fácilmente disponible para los usuarios.

La información específica está disponible para los elementos del módulo de reutilización incluyendo las plataformas utilizadas, la facilidad de reutilización y la información adicional obtenida de los usuarios.

- **Universal Repository**[23]: El repositorio universal fue desarrollado por Unisy. Está diseñado para ayudar a los clientes avanzar hacia un entorno de desarrollo basado en repositorio. El repositorio universal, se basa en principios orientados a objetos y funciona como la columna vertebral de un grupo de trabajo flexible o en un ambiente de desarrollo empresarial. En el núcleo de este repositorio está el Repositorio de Modelo de Servicios (RSM) - que puede incluir representaciones de todas las herramientas, sistemas de gestión de bases de datos (DBMS), lenguajes de programación, reglas de negocio y los datos.

  Los clientes pueden extender el repositorio universal mediante la adición de sus propios modelos sobre la base de los elementos previstos en el RSM. La suma de todos los modelos definidos en un repositorio se llama el ***modelo de información***. Cada parte del entorno de desarrollo de los clientes, se convierte en una pieza integral de la totalidad, cuando los clientes utilizan los modelos incluidos dentro del modelo de información. Esta visión unificada permite a los desarrolladores y clientes, lograr la integración entre las herramientas.

- **AIRS**[24]: AIRS es un sistema de bibliotecas basado en Inteligencia Artificial para el software de reutilización, que fue desarrollado por E.J. Ostertag, J. A. Hendler, R. Prieto- Díaz, C. Braun. AIRS permite a un desarrollador examinar una biblioteca de software en busca de componentes que mejor se adapten a algún requisito establecido. Un componente es descrito por un conjunto de pares <característica, término>. Una característica representa un criterio de clasificación, y se define por un conjunto de términos relacionados. AIRS también permite la representación de los ***paquetes***, es decir, unidades lógicas que agrupan un conjunto de componentes relacionados. Al igual que con los componentes, los paquetes se describen en términos de características. A diferencia de los componentes, una descripción de paquetes incluye un conjunto de componentes miembro.

  Componentes de reutilización candidatos (y paquetes) se seleccionan a partir de la biblioteca basándose en el ***grado de similitud*** entre sus descripciones y la descripción consultada[25]. La similitud se cuantifica por una magnitud no negativa (llamada distancia) que representa el esfuerzo estimado para obtener el objetivo consultado a partir de un componente candidato en el repositorio. Las distancias se calculan a través de funciones llamadas ***comparadores***. Tres de estas funciones son: ***la premisa menor, la cercanía y comparadores paquete***. El enfoque de clasificación AIRS se basa en una formalización de los conceptos y es similar a la clasificación por facetas[26]. La funcionalidad de una implementación prototipo del sistema AIRS se ilustra a través de la aplicación a dos bibliotecas de software diferentes: un conjunto de paquetes Ada para la manipulación de la estructura de datos y un conjunto de componentes de C para su uso en sistemas de comando, control e Información.

- **Reuse Library Toolset**[24]: EVB Software Engineering, Inc. ha anunciado el lanzamiento comercial de una herramienta para una biblioteca de reuso (RLT). RLT es un sistema para crear y gestionar colecciones de activos reutilizables independientes del lenguaje de programación, el método de diseño, o proceso de desarrollo. Para representar a todos los activos de ciclo de vida RLT utiliza el sistema de clasificación considerando palabras claves controladas, valores de atributo (frames), y la interdependencia de los activos.

  La experiencia ha demostrado que el costo de producción de software se reduce significativamente cuando la reutilización es una parte integral del proceso. RLT apoya todas las tareas orientadas a la reutilización, desde la gestión de la biblioteca de activos a través de análisis de dominio hasta la búsqueda de activos y recuperación. Con su interfaz gráfica de usuario intuitiva, RLT es fácil de aprender a los usuarios comunes, sin embargo, proporciona funcionalidad de gran alcance para los usuarios avanzados con necesidades complejas. RLT ofrece servicios de reutilización y métricas de uso de la biblioteca bajo una arquitectura cliente-servidor, con capacidad de intercambio de información de la biblioteca a través de múltiples

plataformas y bases de datos. La arquitectura abierta de RLT permite una fácil integración con las herramientas de desarrollo, tales como herramientas de modelado, sistemas de control de versiones y sistemas de gestión de la configuración.

También existen otros repositorios de tipo gubernamental tales como los siguientes:

- **Defense Software Repository System (DSRS)**[27]: El DSR es un repositorio automatizado para almacenar y recuperar activos de software reutilizables (ASR). El software DSRS pasa a gestionar los inventarios de activos reutilizables a los siete centros de apoyo a la reutilización de software (SRSCs).

  Los DSRS sirven como punto central de colección para calidad ASRs y facilita la reutilización del software, ofreciendo a los desarrolladores la oportunidad de coincidir con sus requerimientos con productos de software existentes.

  A los empleados del gobierno y personal contratista que actualmente soportan proyectos del Gobierno se les crea cuentas DSRS. El Formulario de Solicitud de Cuenta deberá ser aprobado y firmado por el Gobierno director de proyecto del solicitante antes de su presentación a la SRP. La Oficina de Atención al Cliente (CAO) es el punto de SRP de contacto para información técnica, no técnica y el soporte.

  El Sistema de Repositorio de Software de Defensa (DSRS) soporta la clasificación de activos reutilizables para cumplir con las directrices publicadas (DoD 8020.1-M y TAFIM), soporte de ingeniería de dominio, establecer búsquedas eficaces de activos y aumento de la interoperabilidad. La comunidad del software del Departamento de Defensa está tratando de cambiar su modelo de ingeniería de software de su ciclo actual del software de una manera basada en procesos, de dominio específico, basado en la arquitectura y repositorio para soportar la construcción de software[28]. En este nuevo contexto, los DSRS tiene el mayor potencial para convertirse en el repositorio de reutilización estándar DoD porque es el único, repositorio operativo existente con múltiples ubicaciones interoperables en el Departamento de Defensa. Siete lugares DSRS apoyan casi 1.000 usuarios y la lista de cerca de 9.000 activos reutilizables. La DISA DSRS solo enumera 3.880 activos reutilizables y tiene 400 cuentas de usuario. DSR es adaptable a otros tipos de activos reutilizables y mejores métodos de describirlos (clasificación).

- **Library Interoperability Demonstration (LID)**[29][30]: La Biblioteca Demostración de Interoperabilidad (LID) es un sistema de biblioteca prototípico. Se utiliza para ilustrar cómo las bibliotecas de reutilización monolíticas pueden descomponerse en diferentes capas funcionales, conectadas por interfaces abiertas, como los especificados por el Framework Arquitectura Abierta de Biblioteca de Activos (ALOAF). Se trata de una colaboración entre SAIC y Unisys. La demostración muestra cómo el almacenamiento físico de los activos se puede separar de la

catalogación de los mismos. El Programa STARS desarrollo una especificación de un ALOAF para apoyar un enfoque de "sistemas abiertos" para la construcción de bibliotecas de activos. El ALOAF evolucionó para incorporar las interfaces incorporadas específicamente para la interoperabilidad, que culminó con la liberación de ALOAF Versión 1.2 [31]. LID se basa en las interfaces abiertas proporcionadas por ALOAF, el intercambio de activos asociados Lenguaje (AIL), PCTE, OSF/Motif y POSIX. Como se muestra en el diagrama de arquitectura de software LID, una biblioteca de reutilización se puede dividir en tres capas distintas que están conectadas a través de interfaces abiertas, proporcionando así oportunidades para la interoperabilidad en cada capa. Las tres capas son:

- ***Interfaces de usuario.*** La demostración incluye dos herramientas de interfaz de usuario: un navegador gráfico derivado de (RLF) Unisys y un navegador de texto basado en el modelo de InQuisiX de SPS. Ambas herramientas se basan en enlaces Ada de OSF/ Motif.

- ***Catálogos de activos.*** La demostración muestra dos catálogos de activos. El primer catálogo se deriva de la colección de componentes de Unisys ASW, y reside en un IBM RISC System/6000 en el Centro de Tecnología STARS. El segundo catálogo se deriva del conjunto de componentes del simulador de vuelo de SAIC, y reside en un RISC System/6000 IBM en las oficinas SAIC en Orlando, FL. La interfaz entre cada uno de los catálogos y las herramientas de la interfaz de usuario se define por la ALOAF.

- ***Almacenamiento de activos.*** En la demostración, el almacenamiento de los activos es proporcionado por la célula de AFS en el Centro de Tecnología STARS. Los catálogos no almacenan los activos en sí, sino que "subcontratan" la función de almacenamiento en el servidor AFS.

- **Multimedia Oriented Repository Environment (MORE)**[32]: Para soportar el programa de ingeniería de software basada en repositorio (RBSE) de la NASA (Administración Nacional de Aeronáutica y del Espacio Aéreo) ha definido los servicios de biblioteca digital y aplicaciones en el marco del proyecto (ELSA). El programa está orientado a la introducción y la aplicación de sistemas comunes y efectivos de diseño, construcción y mantenimiento de sistemas de software, mediante el uso de los activos de software existentes, almacenados en una biblioteca especializada o repositorio.

  Además de operar un repositorio de ciclo de vida de software, el programa RBSE promueve la transferencia tecnológica, académica y apoyo educativo para los programas de reutilización, el uso de estándares comunes de software de ingeniería y prácticas, investigación tecnológica de reutilización de software, e interoperabilidad entre bibliotecas de reutilización / repositorios.

  Durante su ciclo de vida, el proyecto ELSA respondió a las tecnologías emergentes, la creciente sofisticación de su base de clientes y las tendencias de la industria

mediante la mejora de las capacidades de su software de gestión. Por lo tanto, ELSA se erige como un entorno orientado al cliente que emplea un mecanismo de gestión de bibliotecas avanzada, MORE (Multimedia Oriented Repository Environment). ELSA reemplazó a ADAnet el 31 de agosto de 1994, cuando se concedió el primer acceso del público a su nuevo servicio. En un plazo de aproximadamente dos semanas, ELSA evolucionó desde la biblioteca y los metadatos de acompañamiento de un sistema monolítico basado en X-Windows a un sistema basado en MORE.

El repositorio de activos se clasifica utilizando una colección (tema) y la clase (tipo) de paradigma. De acuerdo con su objeto, se incluyen en la colección o colecciones subordinadas que mejor representan la cobertura de dominio. Los activos se clasifican también por los medios de comunicación o el tipo de información a través del enfoque de clase. Así, los usuarios pueden ver la información desde una perspectiva de arriba hacia abajo a través de la jerarquía de colecciones o en las colecciones de la jerarquía de clases.

MORE fue diseñado para apoyar a esta colección y modelo de clases. La navegación se logra a través de la activación de los enlaces de hipertexto de alto nivel que en última instancia conducen a los metadatos o activos a sí mismos. La búsqueda (lenguaje natural o Patrón del partido) se lleva a cabo con la información proporcionada en los metadatos[33][34]. Esta combinación proporciona a los usuarios un sistema de medios seguros y eficientes de acceso a un gran volumen de activos.

Las funciones administrativas están diseñadas específicamente para satisfacer las necesidades de los bibliotecarios. Por ejemplo, los activos se almacenan en el modo de "desarrollo", que proporciona un entorno de sala limpia para el desempeño de las actividades de población y / o certificación. Los activos del desarrollo sólo están disponibles para ser vistos por los bibliotecarios. Tras la finalización de estos procesos, cada activo es promovido a modo de "producción" y por lo tanto es accesible a la población general de usuarios.

- **Asset Source Software Engineering Technology (SAIC/ASSET)**[35]: Ofrece productos y servicios de apoyo a la biblioteca digital, de comercio electrónico y la ingeniería de software con énfasis en la reingeniería y la reutilización. SAIC / ASSET, establecido por la Agencia de Proyectos de Investigación Avanzada (ARPA) como una sub-tarea bajo la tecnología de software para sistemas fiables (STARS) del programa, es la transición a una empresa privada como una división de Science Applications International Corporation (SAIC).

  La misión principal SAIC/ASSET's es proporcionar un sistema de soporte distribuido para la reutilización del software en el Departamento de Defensa (DoD)

y para ayudar a fomentar una industria de la reutilización del software en los Estados Unidos.

El enfoque inicial y actual de SAIC/ASSET's está en las herramientas de desarrollo de software, componentes reutilizables y documentos sobre los métodos de desarrollo de software. SAIC / ASSET está participando en la interoperación con otras bibliotecas de reutilización, tales como:

- Comprehensive Approach for Reusable Defense Software (CARDS)
- Ada and Software Reuse Information Clearinghouse Defense Software Repository System (DSRS)
- Electronic Library Services & Applications Lobby (ELSA).

Los objetivos SAIC / ASSET implican:

- Crear un centro de información de intercambio de reutilización del software.
- Promover el progreso de la tecnología de reutilización de software.
- Proporcionar un mercado electrónico para productos de software reutilizables a la industria nacional.

Para lograr estos objetivos, SAIC / ASSET opera la biblioteca Worldwide Software Reuse Discovery (WSRD). La Biblioteca WSRD se llena con los componentes de software reutilizables de calidad que pueden ser distribuidos a sus suscriptores. WSRD contiene más de 700 activos a disposición de más de 1.500 usuarios en todo el mundo. La biblioteca se especializa en artefactos y documentos escritos específicamente para promover la reutilización de software y el desarrollo del ciclo de vida del software. Los usuarios SAIC/ASSET tienen acceso a otros componentes almacenados en CARDS y DSRS bibliotecas de reutilización. A través de la WSRD, los usuarios pueden buscar, navegar y descargar los catálogos de activos en más de 30 dominios. Páginas Web de SAIC / ASSET 's World, ubicado en http://source.asset.com/, describen los productos y servicios ofrecidos a través de SAIC/ASSET, así como información relativa a la reutilización del software.

- **Public Ada Library (PAL)**[36]: Desde 1984, el Repositorio de Software Ada (ASR) ha sido una fuente principal disponible al público de desarrolladores Ada. Ahora llamado Ada Public Library (PAL), que proporciona más de 100 megabytes de programas, componentes, herramientas, información general y material educativo sobre el Ada. También contiene los materiales en el circuito integrado de muy alta velocidad (VHSIC) lenguaje de descripción de hardware (VHDL), que se basa en Ada. Para los que tienen acceso a Internet, el PAL se puede acceder a través del protocolo de transferencia de archivos (FTP). El PAL se encuentra en el host *wuarchive.wustl.edu* y en los sitios espejo en *ftp.cnam.fr* y *ftp.cdrom.com,* permitiendo a los computadores para compartir archivos a través de una red, y un sistema de consulta de sitios FTP anónimos-, y gopher, a través de servidores Gopher wuarchive.wustl.edu y gopher.wustl.edu.

- **CAPS Software Reusable Component Repository (Computer Aided Prototyping System)**[37]: Es un proyecto de investigación desarrollado por el Grupo de Ingeniería de Software dirigido por el Prof. Luqi en Naval Postgraduate School. La implementación inicial de la base de software CAPS fue explorada por primera vez en 1988. Una implementación de la base del software se llevó a cabo en 1991 mediante el uso de ONTOS, un sistema de gestión de base de datos orientada a objetos que proporciona una interfaz a C++ para la personalización y flexibilidad[38]. La base de programas CAPS se está cambiando a un repositorio de componentes de software desde 1998. El repositorio de componentes CAPS soporta dos funciones críticas, almacenamiento y recuperación de componente. Hasta donde sabemos, el repositorio CAPS es el único que admite la coincidencia del perfil y correspondencia de la firma. Proporciona alta precisión y método de recuperación de memoria en el mismo instante. Se desarrolló un prototipo para verificar el rendimiento de los métodos de recuperación[18].

2.2.1 Servicios de los Repositorios de Activos

Según Jiang Guo y Luqi[39], los servicios de los repositorios de activos son los siguientes:

- **Búsqueda y recuperación de activos:** un repositorio digital puede almacenar miles o millones de activos, para lo cual es necesario proveer a los usuarios mecanismos de búsqueda y recuperación eficientes, ya que los usuarios pueden estar buscando activos específicos con características para lo cual se debe utilizar criterios de búsqueda una y otra vez hasta localizar el activo. Se debe proveer un servicio de búsqueda que permita ingresar algunos términos de búsqueda y retorne un conjunto de activos como resultado.
- **Exploración de activos:** los usuarios pueden acceder a los activos del repositorio mediante la exploración que nos permite obtener un pantallazo general del repositorio.
- **Marco de componentes estándar:** para incluir el propósito, descripción funcional, nivel de certificación, las limitaciones ambientales clave, los resultados históricos de uso y restricciones legales.
- **Esquema de clasificación para cada dominio:** integrar el activo a los ya desarrollados, previas adaptaciones si es necesario, para construir una nueva aplicación software de dominio específico.
- **Sistema de biblioteca:** automatizada con una interfaz gráfica de usuario.
- **Documentación del activo:** los usuarios del repositorio pueden acceder a la documentación de los activos, con el propósito de conocer sus características.
- **Almacenaje y preservación** de activos.

2.2.2 Operaciones de un Repositorio de Activos

Según Jiang Guo y Luqi[39], las operaciones de los repositorios de activos son los siguientes:

- ***Localización de Activos:*** Es sistema debe proporcionar a los usuarios un mecanismo para la localización y la comparación de los activos reutilizables o componentes individuales de la biblioteca.
- ***Esquema de clasificación:*** Se debe tener un sistema efectivo de clasificación de activos para cada dominio, basado en sus características, el uso de especificaciones formales para modelar y recuperar activos reutilizables[40].
- ***Identificación de activos:*** El repositorio debe proporcionar un sistema de identificación de activos para realizar la comparación, evaluación, y recuperación de activos reutilizables similares.

2.3 Línea de Productos de Software

Una línea de productos de software es una de las formas más prometedoras para aplicar la reutilización, ya que su forma sistemática ayuda en la obtención y organización en la arquitectura de dominio, logrando aumento en la productividad, la competitividad en el mercado y la calidad de los productos software final[41]. Según Clements y Northrop[42] una línea de productos software (Software Product Line - SPL) es un conjunto de sistemas de software que comparten un conjunto común y gestionado de activos que satisfacen las necesidades específicas de un segmento de mercado o misión y que son desarrollados a partir de un conjunto común de activos fundamentales [de software] de una manera pre-escrita. Para Krueger[43], una línea de productos software (Software Product Line - SPL) se refieren a técnicas de ingeniería para crear un portafolio de sistemas de software similares, a partir de un conjunto compartido de activos de software, usando un medio común de producción. Según Montilva[44], una línea de productos software SPL consiste en un conjunto de elementos base para producir un conjunto de aplicaciones software que comparten características comunes (llamadas similitudes o commonalities), pero al mismo tiempo mantienen sus características propias (llamada variabilidad). Aunque el enfoque ofrece beneficios de gran impacto para una organización, su adopción no es sencilla y existen algunas dificultades para su adopción práctica: las empresas requieren de una fuerte estructura organizacional para la ejecución exitosa del enfoque SPL.

2.3.1 Características (Features), Puntos de Variación y Variantes

Dentro del contexto de la ingeniería de dominio, cada una de los elementos que permiten caracterizar y diferenciar un producto se define como una característica (feature). Las líneas de productos software, se pueden estructurar en términos de un conjunto de características. Algunas de las características son comunes para todos los productos potenciales de la línea

y otras características, sólo se encontrarán en algunos productos. Una característica es cualquier cosa que los usuarios o programas clientes querrían controlar sobre un concepto[45], también es definida como un aspecto, cualidad o característica de un sistema software o sistemas software que para un usuario es visible y distintivo[46], o un incremento en la funcionalidad[47] o un elemento distinguible de un sistema que es relevante para un participante del sistema[48]. Además de las características, también se encuentra en la literatura el uso de puntos de variación o variantes (variation point / variant). Un punto de variación es una representación de un sujeto variable dentro de un artefacto de dominio enriquecido con información contextual y una variante es una representación de un objeto variable dentro del artefacto del dominio[49]. Las características y los puntos de variación / variantes se utilizan tanto en el espacio del problema como en el espacio de la solución. No hay un consenso en este sentido, Por ejemplo, en [50] y [51], las características se utilizan en el espacio del problema y los puntos de variación/variantes en el espacio de la solución. Sin embargo en [52] es al contrario, las características se utilizan tanto en el espacio del problema como de la solución, mientras los puntos de variación/variantes son sólo útiles en el espacio de la solución [53][54]. La variabilidad de software es la capacidad de un sistema de software o artefacto de ser cambiado, personalizado o configurado para uso en un contexto particular. Un alto grado de variabilidad permite el uso de software en una gama más amplia de contextos, es decir, el software es más reutilizable. La variabilidad es el elemento más característico de las líneas de productos de software y su gestión eficiente es fundamental a la hora de definir la línea de producto satisfactoriamente. La variabilidad de una SPL es definida en el proceso de ingeniería de dominio mediante un modelo de variabilidad que captura la variabilidad del dominio en requisitos, arquitectura, componentes y pruebas, y se explota en la etapa de Ingeniería de Aplicación.

2.3.2 Técnicas y Métodos para documentar la Variabilidad

Una primera propuesta para modelar la variabilidad se introdujo como parte del método FODA (Feature-Oriented Domain Analysis) en 1990[55]. A partir de esta propuesta inicial se han construido diferentes métodos como FORM (Feature-Oriented Reuse Method)[56] o Feature RSEB (Feature Reuse-Driven Software Engineering Business)[57], además se han instrumentalizado otros enfoques como la programación generativa[58], la ingeniería de Líneas de Productos Software[59] o PLUSS[60].

Las diferencias entre los diferentes enfoques incluyen, pero no se limitan a los siguientes aspectos: sintaxis, semántica, expresividad, propósito y soporte de automatismo. En lo que se refiere a los **aspectos de sintaxis**, la mayoría de las propuestas son notaciones gráficas, pero también hay lenguajes textuales, como el *Lenguaje Declaraciones de Características* proporcionado por Gears[61].

En lo que respecta a los **aspectos de semántica**, las notaciones gráficas suelen representar diagramas de características por medio de nodos y conexiones. En estos diagramas, en algunas notaciones, la función opcional está vinculada a los nodos (por ejemplo, en el caso de PLUSS[62]), y en otras está vinculada a las conexiones (por ejemplo, en la notación propuesta por M. Riebisch et al.). Sobre los **aspectos de expresividad**, mientras que la mayoría de los lenguajes desarrollados para la documentación de la variabilidad son expresivamente completos, es decir, capaces de representar cualquier serie de prestaciones de combinaciones del mundo real, algunos de ellos no lo son, como por ejemplo, la notación original FODA. En lo que al **propósito** se refiere también

encontramos diferencias; mientras que el *Modelo de Variabilidad Ortogonal* (OVM)[63] está orientado al espacio del problema y de la solución, el *Lenguaje Declaraciones de Características* proporcionado por Gears está orientado exclusivamente al espacio de problema. Esta profusión de lenguajes y notaciones dificulta la comunicación entre los desarrolladores, diseñadores y usuarios y, además, provoca problemas de portabilidad de los diagramas de características entre las herramientas de soporte. Por último, precisamente, está el aspecto de las herramientas de soporte. La automatización y las herramientas es un aspecto clave para el soporte de estas técnicas, métodos y lenguajes para la gestión de la variabilidad.

2.3.3 Ingeniería de Requisitos (IR) en Líneas de Productos

Según Sommerville y Kotonya[64], el término Ingeniería de Requisitos (IR) implica el uso de procedimientos repetibles y sistemáticos para asegurar la obtención de un conjunto de requisitos relevante, completo, consistente y fácilmente comprensible y analizable por parte de los diferentes actores implicados en el desarrollo del sistema. De acuerdo al Standard IEEE 610[65] un requisito es: (a) una condición o capacidad necesitada por un usuario para resolver un problema o alcanzar un objetivo; (b) una condición o capacidad que un sistema o un componente que un sistema debe satisfacer o poseer de acuerdo con un contrato, estándar, especificación u otro documento impuesto formalmente; o bien (c) una representación documentada de una condición o capacidad como en (a) o (b). Los requisitos de una línea de productos definen los productos de dicha línea y sus características comunes y variables[3]. La Ingeniería de Requisitos (IR) para líneas de productos debe gestionar los requisitos de la línea de productos y los requisitos de los productos concretos de la línea. La IR para líneas de productos debe incorporar un mecanismo mediante el cual el conjunto de requisitos para un producto concreto sea producido de manera fácil y rápida a partir de los requisitos de la línea de productos. Siguiendo el enfoque inicial de Greenfield y Short sobre fábricas (factorías) de software (software factories)[66], en las modernas fábricas de software la IR se orienta hacia líneas de productos. El reto fundamental de la IR para líneas de productos es el tratamiento adecuado de la variabilidad de los requisitos[67]. La IR para líneas de productos se relaciona habitualmente con el uso de modelos de características[68], con extensiones para los modelos de casos de uso[69], y más recientemente con modelos de variabilidad independientes[70][71]. En contraste, la IR convencional se suele orientar a la gestión de requisitos textuales preferentemente en lenguaje natural y de documentos de requisitos.

2.3.4 Small SPL

Small SPL es un modelo de proceso de desarrollo de software que busca relacionarse entre las exigencias de producción de líneas de productos y las características de las pequeñas y medianas empresas desarrolladoras de software, este proceso busca ser simple, liviano, adaptable y guiado por la gestión de activos, para que estas entidades puedan aplicarlo de forma más natural[6]. Small SPL sigue un enfoque evolutivo tanto en el desarrollo de los productos como en el desarrollo de los activos reutilizables, y su ciclo de vida incluye tres subprocesos: *Ingeniería de Dominio* - ID e *Ingeniería de Aplicaciones* - IA, engranados por un tercer subproceso denominado *Gestión de Activos basado en Requisitos* – GAR[6]. El proceso de Gestión de Activos basado en Requisitos – GAR: permite comunicar los dos procesos fundamentales en la construcción de una línea de procesos, facilitando la

identificación, desarrollo, documentación, almacenamiento, búsqueda y uso de los activos que serán empleados en el desarrollo de los productos de la línea.

Este proyecto de trabajo de grado propone R-GAC (Repositorio de Gestión de Activos basado en Características), un repositorio para facilitar la gestión de los activos que toma como elemento central el modelo de características para gestionar el flujo de activos entre la ingeniería de dominio y la ingeniería de aplicación. Esto garantiza que cuando se esté desarrollando un proyecto de líneas de producción de software y se requieran activos del repositorio, R-GAC se pueda utilizar para derivar el repositorio base de un producto a partir del repositorio general de la línea de productos.

3 Trabajos Relacionados

3.1 Modelos de recuperación de activos en forma oportunista

Los modelos de recuperación de activos se enfocan hacia la construcción de un gran repositorio en el que la meta-data de los activos es utilizada para facilitar su búsqueda y recuperación. Este tipo de enfoque sigue una estrategia de reutilización oportunista y no planificada como lo requiere el enfoque SPL.

Uno de estos casos es el proyecto **Apache UIMA** (Unstructured Information Management Arquitecture)[72], el cual consiste en aplicaciones software que analizan grandes volúmenes de información no estructurada con el fin de descubrir el conocimiento que sea relevante para el usuario final. Apache UIMA, provee una implementación open source con Licencia Apache, en el sitio es posible acceder a frameworks, componentes e infraestructuras de software. La información no estructurada tal como texto, audio, video, etc. representa la fuente más grande y de mayor crecimiento en la actualidad, tanto para empresas como para organizaciones gubernamentales. La motivación para el desarrollo de tales frameworks fue construir una plataforma común para el análisis no estructurado, fomentar el reuso de componentes para el análisis y reducir la duplicidad del análisis del desarrollo.

Los componentes están escritos en Java o C + +, los datos que fluyen entre los componentes está diseñado para el mapeo eficiente entre estos idiomas. Este repositorio se concentra en almacenar componentes de implementación, dejando por fuera elemento reutilizables a diferentes niveles de abstracción que van más allá del código, tales como diseños, descripciones de requisitos, análisis, planes, entre otros. A diferencia de este repositorio, los activos del repositorio para Small SPL pueden ser código, modelos y requerimientos. Además, existe una relación rica entre los activos de los distintos niveles de abstracción apoyados en los conceptos de reutilización planificada, en lugar de tener activos de software para una reutilización oportunista como en UIMA.

Otro de esos casos se presenta con **SourceCodeOnline**[73] en el que se facilita a través de una web en el cual es posible compartir componentes de software reutilizables, codificados en diferentes lenguajes de programación, artículos, librerías, etc., se puede acceder a los recursos de diferentes maneras, las cuales se encuentran categorizadas por lenguaje de programación, ranking de códigos más accedidos, nuevos componentes, etc.

Además provee una breve descripción, nombre del autor, link y categoría a la cual pertenece el componente dentro de las definidas en el sitio.

Dentro de esta misma línea se encuentra el **Software Artefact Infraestructure Repository (SIR)**[74] es un trabajo en progreso, siendo constantemente mejorado y expandido a través de la incorporación de nuevos temas. Luego de registrarse como usuario, es posible acceder a un menú con opciones tales como, descarga de objetos, descarga de herramientas, objetos archivados, publicaciones y usuarios SIR, etc. Es utilizado por Universidades en sus proyectos de investigación y citado en numerosas publicaciones. Los componentes de este repositorio cuentan con un Handbooks en el cual se describe como fueron creados y proporciona información sobre su uso. Este repositorio tiene la ventaja de manejar las versiones con lo cual se puede controlar un poco la evolución de los activos.

También es el caso de **VCLComponents.com**[75], un repositorio con una interface web que facilita acceder a una gran variedad de componentes, scripts y código fuente implementado en diferentes lenguajes de programación. Por otro lado, los componentes están organizados en un directorio de lenguajes de programación como por ejemplo, ASP, Assembler, Basic, C#, C/C++, Java, Html, Php, Sql entre otros. Dentro de cada categoría de lenguaje es posible identificar otras categorías, por ejemplo, para el lenguaje Java, existen las subcategorías Applets, Calendar Scripts, Clock Scripts, Game software, Jsp and Servlets, Libraries, etc.

De manera similar **NetLib** [76], un repositorio de software libre, documentos y bases de datos para matemáticas, computación científica y otras comunidadesofrece mecanismos de almacenamiento y recuperación. NetLib está bajo el mantenimiento del laboratorio AT&T Bell, la Universidad de Tennessee, el Laboratorio Nacional de Oak Ridge y por sus colegas en todo el mundo. Esta colección de componentes esta replicada en varios sitios de todo el mundo, los cuales están sincronizados automáticamente para proveer un servicio eficiente y fiable a través de la red.

Todos los enfoques anteriores son propuestas generales, centradas en el código como único activo reutilizable y no consideran la gestión de activos en forma planificada.

3.2 Modelos de recuperación de activos en forma planificada

En la recuperación de activos en forma planificada, un repositorio se configura teniendo en cuenta estructuras de organización de líneas de productos, así como el uso de la variabilidad en forma planificada.

Una de estas soluciones es **SPLOT**[77], un repositorio que utiliza el razonamiento basado en la web y el sistema de configuración para Líneas de Producto Software (SPL). El sistema se beneficia de técnicas avanzadas de razonamiento basadas en la lógica, tales como solucionadores SAT y diagramas de decisión binarios para proveer razonamiento eficiente y servicios de configuración interactiva para los investigadores y los profesionales de SPL. Además, el sistema proporciona un repositorio de modelos de entidad que contiene los modelos reales y generados para fomentar el intercambio de conocimientos entre los investigadores en el campo.

Otra solución es **FAMA Framework (FAMA FW)**[78] que propone un enfoque para el análisis automatizado de los modelos de la variabilidad (VM). Su objetivo principal es proporcionar un marco extensible que la investigación actual sobre el análisis automatizado VM puede ser desarrollada y puede integrar fácilmente en un producto final. FAMA FW está construido siguiendo el paradigma SPL apoyar diferentes meta modelos, variabilidad, razonadores o solucionadores, preguntas de análisis y selectores razonador, facilitando la producción de herramientas de análisis de VM personalizados. FAMA FW está escrita en Java y se distribuye bajo licencia LGPL. Con esta herramienta, se pueden realizar diversas operaciones de razonamiento como el cálculo de la cantidad de productos en un SPL, obtener su lista de productos, productos de filtrado de acuerdo con un criterio o detectar y explicar los errores.

Por su parte **BigLever Software**[79] incluye mecanismos para administrar todo un portafolio de línea de producto como un único sistema, su producción automatizada - al igual que una fábrica - que utiliza la característica de perfiles de productos para montar y configurar automáticamente los activos necesarios para diseñar y producir su línea de productos. Este enfoque incluye un único sistema de producción que permite a las organizaciones la transición de prácticas centradas en el producto convencional de ingeniería hacia el enfoque de línea de productos. En este modelo la producción, gestión de la variabilidad y el portafolio de productos están basados en las características (features) de los productos.

Los repositorios de enfoque oportunista se concentran en almacenar componentes de implementación, dejando por fuera elemento reutilizables a diferentes niveles de abstracción que van más allá del código, tales como diseños, descripciones de requisitos, análisis, planes, entre otros. A diferencia de este tipo de repositorios, los activos del repositorio para Small SPL pueden ser código, modelos y requerimientos. Además, existe una relación rica entre los activos de los distintos niveles de abstracción apoyados en los conceptos de reutilización planificada, en lugar de tener activos de software para una reutilización oportunista. FAMA FW que sigue el paradigma de líneas de productos se orienta al análisis de modelos de características, más que soportar las actividades esenciales para la gestión de activos. SPLOT es una solución muy cercana a lo buscado, pero tiene un enfoque experimental, el repositorio está disponible pero no su implementación y por tanto, al momento, no resulta viable su uso en la industria. Mientras el enfoque de Small SPL es centrado en los requisitos dando la posibilidad a la empresa de definirlos como casos de uso, historias de usuario o features, BigLever es una solución comercial centrada en las features (características) y trabaja sobre su propia metodología comercial. La Tabla 1 sintetiza esta comparativa.

Trabajos Relacionados	***Enfoque de Reutilización***	***Tipo de Activos***	***Tipo de Repositorio***	***Para VSE[3]***	***Orientación***

[3] Para VSE - El criterio utilizado para este punto, es si bien el propósito de su construcción se refiere a una pequeña organización o bien existe una evidencia publicada a la fecha de su uso en pequeñas organizaciones.

Apache UIMA	Reutilización Oportunista	Código	Comunidad	No	Código
SourceCode Online	Reutilización Oportunista	Código, Artículos, Librerías	Comunidad	No	Documentos
Software Artefact Infraestructure Repository (SIR)	Reutilización Oportunista	Código, objetos archivados, Publicaciones	Comunidad Científica	No	Documentos
VCLComponents.com	Reutilización Oportunista	Scripts y código fuente	Comunidad	No	Documentos
NetLib	Reutilización Oportunista	Código, documentación	Comunidad Científica	No	Documentos
SPLOT	Líneas de Productos	Modelos	Comunidad Científica	No	Modelos
FAMA FW	Líneas de Productos	Modelos	Comunidad	No	Features
BigLever Software	Líneas de Productos	Modelos	Comercial	No	Features
GAC	Líneas de Productos	Modelos, Artefactos y Mecanos	Comunidad	Si	Features

En este trabajo de grado se busca lograr algo similar pero a nivel abierto y para el soporte dentro del proceso Small SPL. En el presente trabajo haremos uso de la gestión de activos basada en las ***características*** como elementos de referencia. El modelo de características fue el artefacto que mayor uso e impacto tuvo en los estudios de caso con Small SPL[6]. En el contexto de este proyecto, una ***característica*** es una propiedad, dentro del dominio, que, tanto desde la perspectiva del espacio de la solución, como del espacio del problema representa un aspecto relevante para cualquier participante. Para llevar a cabo una representación de las ***características***, se utilizan los modelos de características[82] , que visualmente simplifican la comprensión del sistema.

4. RGAC: Repositorio para la Gestión de Activos basada en Características en el Marco del Proceso Small SPL

Es posible que aunque no haya sido diseñada para tal fin, ni se reporten experiencias el enfoque pueda resultar de utilidad para una pequeña organización.

4.1 Introducción

RGAC es un repositorio web como soporte a la propuesta realizada por la metodología Small SPL que promueve la reutilización de activos de software basada en características a través de líneas de productos de software inmersos dentro de un repositorio. Si bien existen varios repositorios, aquellos que están disponibles o no están orientados a las líneas de productos o se orientan, pero no bajo el enfoque centrado en las características del software. RGAC fue desarrollado en la plataforma Java EE y utilizando Mysql como versión inicial de investigación para el manejo de las líneas de productos de software gestionando activos en pequeños equipos de desarrollo. La gestión de activos en el contexto de una línea de productos es una de las actividades claves para facilitar la adopción del enfoque, en este capítulo se presenta el contexto, los principios, las características funcionales, el modelo conceptual, el modelo de diseño, la implementación de dicho repositorio y la forma de utilizarlo. Toda esta caracterización se basa en el resultado de la comparativa lograda en el estudio del estado del arte y en las experiencias previas con el proceso Small SPL, para el cual este repositorio brinda soporte.

4.2 Contexto de RGAC

La perspectiva de la producción de líneas de productos software se basa en la reutilización, convirtiéndose en un desafío para las pequeñas empresas productoras de software, modificando las prácticas existentes de reutilización desde un enfoque reactivo hacia prácticas simples e iníciales de desarrollo para la reutilización.

En la Figura 1 se visualiza cómo la gestión de activos engrana la producción, almacenamiento, búsqueda y uso de los activos producidos en la línea. Particularmente, la interacción entre el repositorio y los procesos de Ingeniería de Dominio (ID) e Ingeniería de Aplicación (IA) por medio de eventos significativos al negocio durante el desarrollo de la línea de productos. Así, la gestión de las características puede inicializar por un evento generado en cualquiera de los otros dos subprocesos, permitiendo que sucedan en forma paralela.

A medida que sea necesario se hace el cambio entre un proceso y otro, teniendo en cuenta las condiciones del activo que se desarrolla. Por ejemplo si éste es reutilizable y según la variabilidad exhibida, debería guiarse por el proceso de la ingeniería de dominio. Sin embargo, si se trata de un activo específico a un producto se seguirá la ingeniería de aplicaciones de acuerdo al activo marco determinado en la ingeniería de dominio. Si este activo marco no existe, entonces de todas formas se dispara el proceso de ingeniería de dominio, puesto que nada puede ocurrir de forma no planificada. Esto busca simplificar y orientar hacia el valor el desarrollo de líneas de productos a las pequeñas y medianas entidades productoras de software.

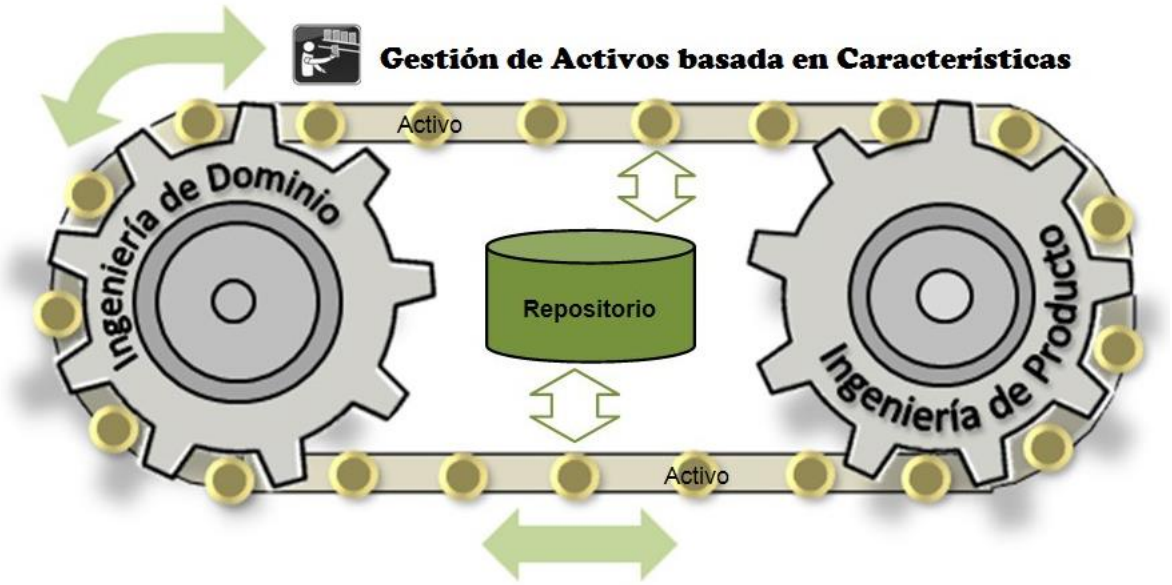


Figura 1. Perspectiva de la Gestión de activos basada en Características.

La gestión de activos basada en características determina el flujo de activos entre la ingeniería de dominio y la ingeniería de aplicación. Lo que significa que frente al diseño e implementación de una característica se dispara un proceso o tarea en el respectivo subproceso. Esto garantiza que cuando se esté desarrollando una aplicación particular y se requieran elementos de dominio no desarrollados se disparará el flujo de trabajo de los procesos de ingeniería de dominio.

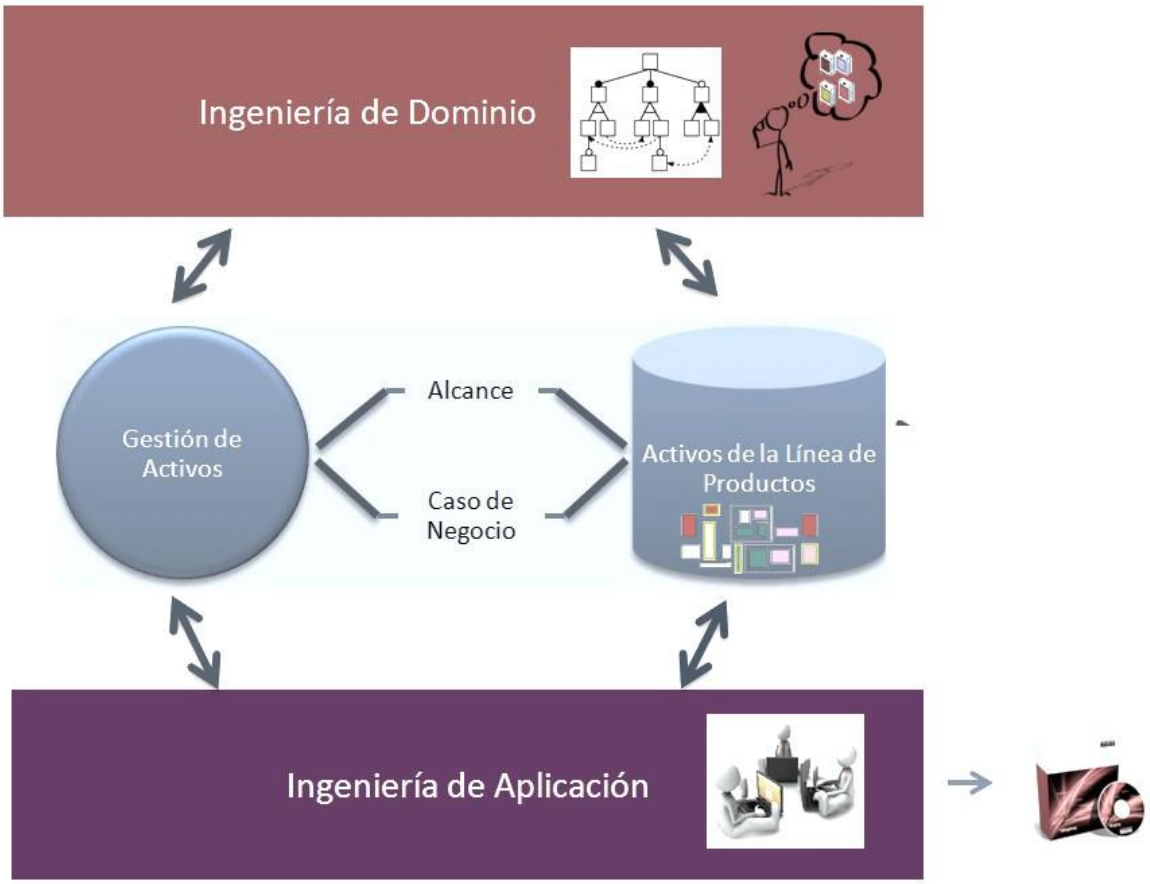


Figura 2. Subproceso Gestión de activos basada en característica.

El Repositorio RGAC está enfocado a pequeñas y medianas empresas de software, las cuales muchas veces no poseen el tiempo ni los recursos necesarios para dedicarse exclusivamente a la ingeniería de dominio, dadas algunas experiencias reportadas de desarrollo de una línea de productos a partir de un producto base[80][81]. El Repositorio RGAC ha sido definido para permitir la implementación y de desarrollo del proceso Small SPL.

La gestión de activos es por tanto una actividad clave, permite a los desarrolladores gestionar la pila de necesidades de la línea y del producto, de tal manera que determinará si un producto hace o no parte de la línea, y si existe un caso de negocio que lo amerite, permitirá extender el alcance de la línea, esto puede darse en el desarrollo de la ingeniería de dominio, pero también en la producción de una aplicación específica, por lo tanto, RGAC debe permitir la gestión de los activos, las características y las líneas de producto.

Las actividades del subproceso RGAC son tres: preparación, almacenamiento y consumo y actualización como se puede observar en la siguiente figura:

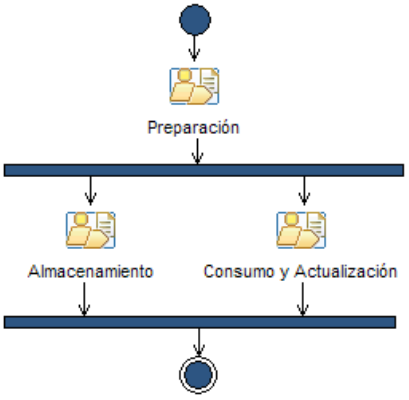


Figura 3. Actividades del subproceso RGAC.

Actividad Preparación

El objetivo de esta actividad es preparar las condiciones necesarias para realizar el desarrollo para la reutilización, elementos organizacionales como pautas a seguir y condiciones técnicas. Las tareas de esta fase en la figura.

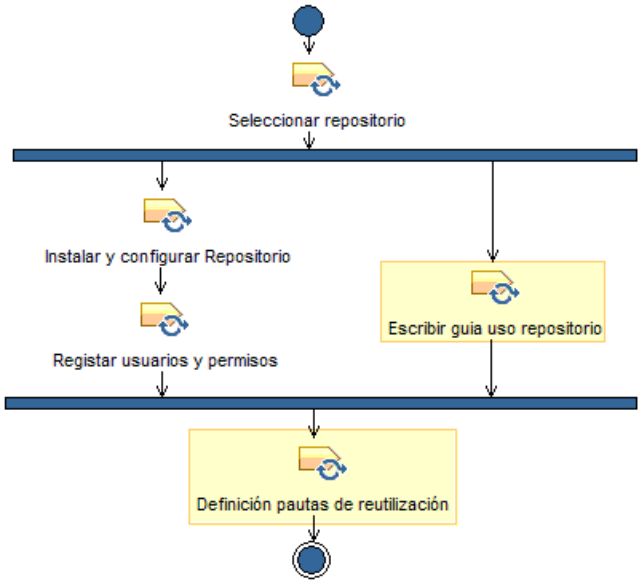


Figura 4. Tareas de la actividad preparación del subproceso RGAC.

La tarea selección del repositorio tiene como objetivo la búsqueda, adquisición, e instalación del repositorio, se busca que sea el más adecuado para la empresa, el equipo de desarrollo, y el proyecto de líneas de productos.

La tarea ***Definición de pautas de reutilización*** tiene como objetivo que el equipo identifique, describa, y acuerde pautas para la reutilización planificada, de tal forma que todo el equipo siga los lineamientos acordados, y facilite el proceso de reutilización.

Actividad de Almacenamiento

La actividad de ***almacenamiento*** tiene como objetivo almacenar, clasificar y organizar los activos generados en el desarrollo de la línea de productos, de tal forma que estén fácilmente disponibles para su reutilización. Las tareas de esta actividad ocurren como respuesta a un evento generado en el subproceso de ingeniería de dominio, o en el subproceso ingeniería de aplicaciones, en cualquiera de los tres posibles escenarios. De acuerdo al evento se iniciara una de tres tareas: ***Adicionar características***, ***Adicionar activos*** y ***Catalogar elementos anteriores***, como puede visualizarse en la figura.

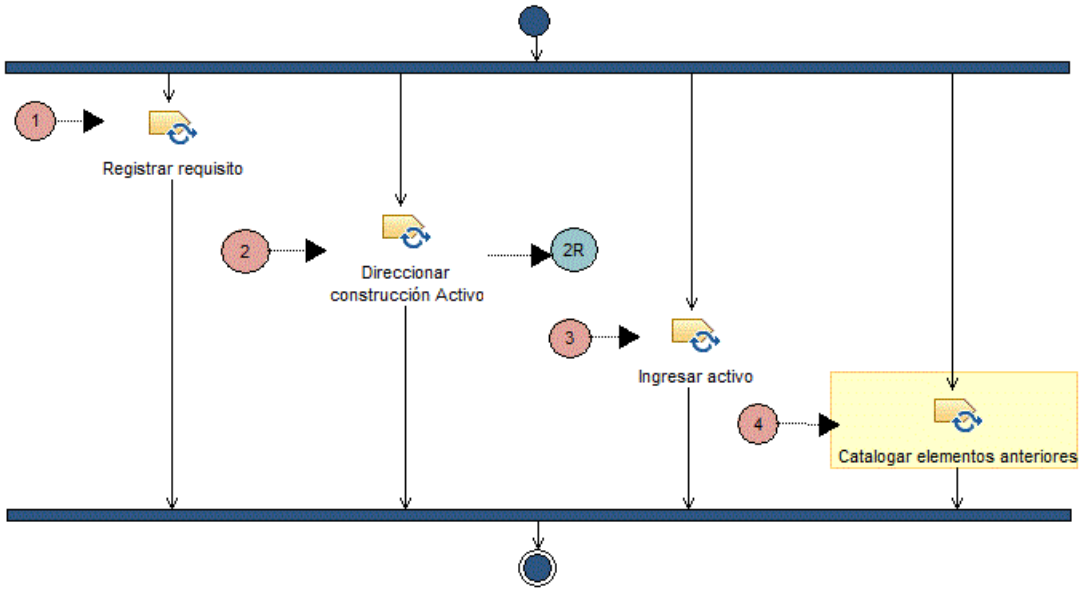


Figura 5. Actividad Almacenamiento de RGAC.

La tarea ***adicionar característica*** tiene como objetivo la inscripción de una característica en un catálogo de características, posteriormente son clasificados y se establece el estado en que se encuentra, esta actividad permite la trazabilidad del historial de una característica.

La tarea ***adicionar activos*** permite el almacenamiento y la clasificación de los activos que se van generando en el desarrollo de la línea de productos, y se realiza una relación de correspondencia entre características y activos.

Actividad Consumo y Actualización

El objetivo de la actividad de ***consumo y actualización*** es la búsqueda, recuperación, actualización y trazabilidad de los activos, para su reutilización en el desarrollo de los

productos de la línea, las tareas de esta fase son Gestión de Línea de Productos, Gestión de Características, Gestión de Activos y Gestión de Usuarios, cuales son mostradas en los Anexos.

4.3 Modelo Conceptual de R-GAC

Las características de la Línea de Productos constituyen los activos núcleo más importantes y tangibles[42]. La gestión de activos basada en características tiene aspectos particulares para una línea de productos debido a que las características de una línea de productos abarcan varios activos para generar el producto. Existe una dependencia natural entre las características en la ingeniería de dominio y las características en la ingeniería de aplicaciones. El repositorio RGAC permite gestionar las modificaciones y evoluciones que se producen en los activos durante el desarrollo del dominio y del producto en forma relacionada a las características.

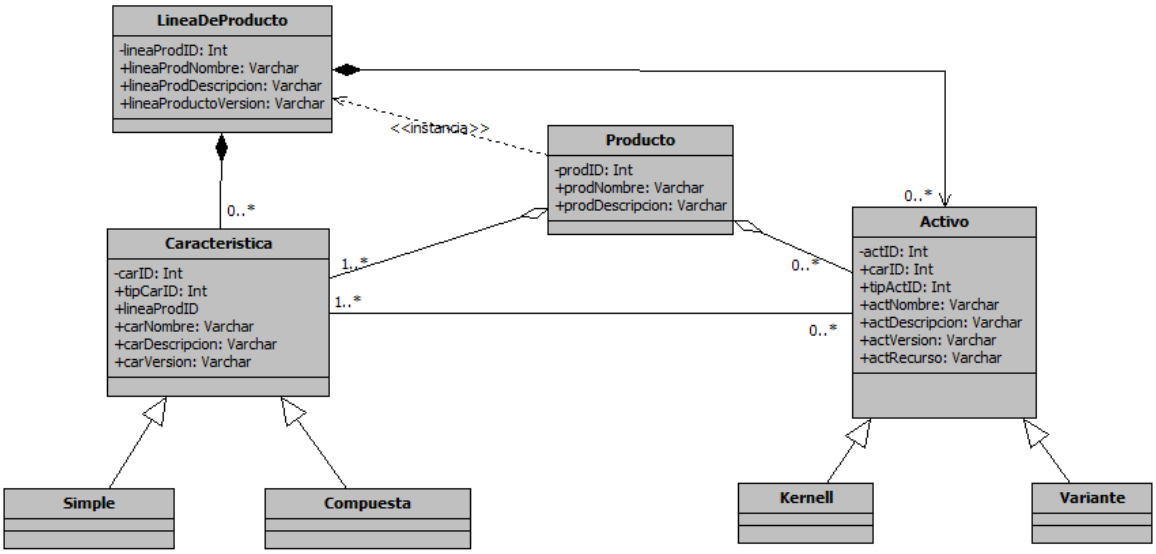


Figura 6. Modelo Conceptual de RGAC.

Una línea de productos es la composición planificada de un conjunto de activos de software reutilizables. A su vez es expresada como un conjunto de características que permiten configurar uno de los posibles productos. Los activos son entidades que expresan una parte reutilizable del sistema en un nivel de especificación, dada la clasificación de las partes en una línea de productos, un activo puede ser Kernell, si corresponde a una parte común en todos los productos derivable. Los Kernell incluyen puntos de variación para manejar las variantes, las cuales son activos de software reutilizables pero variables de producto a producto.

A continuación, se describe cada concepto del modelo del Repositorio RGAC:

Línea Producto

Superclase: Ninguna
Descripción: “Línea Producto” es una especificación concreta que representa el concepto de línea de productos y es el contenedor principal de un grupo características que la describen y de los activos de software que le pertenecen.
Atributos:

- “Nombre: String” Es un atributo que permite definir el nombre de la Línea de Productos Software.
- “Descripción: Text” En esta propiedad se permite especificar la descripción de la Línea de Productos Software.
- “Versión: String” Este campo permite definir la versión de la Línea de Productos Software.

Propiedades Asociadas:

- “myFeatures: Características” Línea de Productos almacena un grupo de elementos los cuales son utilizadas para agrupar un conjunto de características de los proyectos las cuales son semánticamente relacionadas. Esta asociación de composición representa que cada Característica está contenida en una Línea de Productos.
- “myAssets: Activos” Línea de Productos tiene un grupo de elementos los cuales son utilizados para agrupar un conjunto de activos de los proyectos los cuales son semánticamente relacionados, Esta asociación de composición representa que cada Activo está contenido en una Línea de Productos.

Semántica: Línea Producto provee la propiedad de relacionar la clase Característica y la clase Activo a esta clase. Línea Producto representa una estructura que contiene las características que la describen y los activos de software que la componen, con ello permite configurar productos particulares que le pertenecen a la familia de productos que la línea de productos es capaz de generar.

Característica

Superclase: Ninguna
Descripción: “Característica” es una especificación concreta que representa una propiedad implementable del sistema, que puede recursivamente ser expresada en términos de sub-características y que tienen asociados algunos activos de proceso.
Atributos:

- “Nombre: String” Es un atributo que permite definir el nombre de la Característica.
- “Descripción: Text” Es un atributo que permite especificar la descripción de la Característica.
- “Versión: String” Este campo permite definir la versión de la Característica.

Propiedades Asociadas:

- “imply: Característica” Está asociación de composición representa que cada Característica puede estar asociada a cero o muchas Características.
- “myAssets: Activo” Está asociación de composición representa que cada Característica puede estar asociada a cero o muchos Activos.
- “myFeatures: CaracterísticaCompuesta” Característica Compuesta almacena un grupo de elementos los cuales son utilizadas para agrupar un conjunto de características de los proyectos las cuales son semánticamente relacionadas. Esta

asociación de composición representa que algunas características están contenidas en una Característica Compuesta.

Semántica: Característica provee la propiedad de relacionar la clase Línea de Producto y la clase Activo a esta clase. Característica representa una estructura que permiten poseer campos para gestionar información basada en los requerimientos y la identificación de las características de la familia de productos. Las características siguen el enfoque de modelado de Feature Models[82] y por tanto se estructuran como un árbol que facilita su definición recursiva. Cada Característica permitirá gestionar todos los activos reutilizables de software asociados que hagan parte de la línea de productos a la cual también debe pertenecer la característica.

Característica Simple

Superclase: "Característica"
Descripción: "Característica Simple" es una generalización abstracta de una característica y que representa un tipo de característica de un grupo de los activos existentes.
Semántica: Característica Simple provee la propiedad de relacionar la clase Característica a estas subclases. Característica Simple permite gestionar información para la identificación de las características básicas de los activos.

Característica Compuesta

Superclase: "Característica"
Descripción: "Característica Compuesta" es una generalización abstracta de varias características y que representa un tipo de características de un grupo de los activos existentes.
Semántica: Característica Compuesta provee la propiedad de relacionar la clase Característica a estas subclases. Característica Compuesta permite poseer campos para gestionar información, en la identificación de las características compuestas de los activos.

Alternativo

Superclase: "Característica Compuesta"
Descripción: "Alternativo" es una generalización abstracto de características que representa un tipo de características compuestas de un grupo de los activos existentes.
Semántica: Alternativo provee la propiedad de relacionar la clase Característica Compuesta a estas subclases. Alternativo permite gestionar información para la identificación de las características compuestas de los activos.

Alternativo XOR

Superclase: "Característica Compuesta"
Descripción: "Alternativo XOR" es un tipo abstracto de características que representa un prototipo de características compuestas de un grupo de los activos existentes.
Semántica: Alternativo XOR provee la propiedad de relacionar la clase Característica Compuesta a estas subclases. Alternativo XOR permite gestionar información para la identificación de las características compuestas de los activos.

Activo

Superclase: “Característica”
Descripción: “Activo” es una representación concreta de los activos o funciones reutilizables.
Atributos:

- “Nombre: String” Es un atributo que permite definir el nombre del Activo.
- “Descripción: Text” Es un atributo que permite especificar la descripción del Activo.
- “Versión: String” Es un atributo que permite definir la versión del Activo.
- “Recurso: String” Es un atributo que permite determinar la ubicación del activo o función.

Propiedades Asociadas:

- “realizedFeatures” Esta asociación representa que cada activo está asociado a una o muchas Características.
- “myFeatures: ActivoCompuesto” Activo Compuesto almacena un grupo de elementos los cuales son utilizadas para agrupar un conjunto de activos de los proyectos los cuales son semánticamente relacionadas. Esta asociación de composición representa que algunos activos están contenidos en una Activo Compuesto.

Semántica: Activo provee la propiedad de relacionar la clase Línea de Producto y la clase Característica a esta clase. Un activo representa una estructura que permiten poseer campos para gestionar información basada en las funciones pertenecientes a las características de la familia de productos. Los activos siguen el enfoque de modelado de Feature Models[82] y por tanto se estructuran como un árbol que facilita su definición recursiva. Cada activo permitirá gestionar todos los productos de software creados y que están asociados que a la línea de productos.

Kernel

Superclase: “Activo”
Descripción: “Kernel” es una representación abstracta de un tipo de los activos o funciones reutilizables.
Propiedades Asociadas:

- “myPoints:Puntos de Variación” Esta asociación de composición representa que los Puntos de Variación están relacionados con un núcleo.

Semántica: Kernel provee la propiedad de relacionar la clase Activo a estas subclases. Kernel representa una estructura de almacenamiento que permite gestionar la información de los Activos.

Variante

Superclase: “Activo”
Descripción: “Kernel” es una representación abstracta de un tipo de los activos o funciones reutilizables.

Semántica: Variante provee la propiedad de relacionar la clase Activo a estas subclases. Variante representa una estructura de almacenamiento que permite gestionar la información de los Activos.

Puntos variación

Superclase: Ninguna
Descripción: “Puntos Variación” es una representación abstracta de diferentes tipos de los activos o funciones reutilizables.
Propiedades Asociadas:

- “myPoints:Puntos de Variación” Esta asociación de composición representa que los Puntos de Variación están relacionados con una o muchas Variantes.

Semántica: Puntos Variación provee la propiedad de relacionar la clase Kernel y la clase Variante a estas subclases. Puntos Variación representa una estructura de almacenamiento que permite gestionar la información de los Activos.

Producto

Superclase: “Activo”
Descripción: “Producto” es una representación abstracta de los productos creados en el repositorio.
Atributos:

- “Nombre: String” Es un atributo que permite definir el nombre del Producto.
- “Descripción: Text” Es un atributo que permite especificar la descripción del Producto.

Propiedades Asociadas:

- “myFeatures: Características” Producto almacena un grupo de elementos los cuales son utilizadas para agrupar un conjunto de características de los proyectos las cuales son semánticamente relacionadas. Esta asociación de composición representa que algunas características hacen parte de un Producto.
- “myAssets: Activos” Producto tiene un grupo de elementos los cuales son utilizados para agrupar un conjunto de activos de los proyectos los cuales son semánticamente relacionados, Esta asociación de composición representa que algunos activos hacen parte de un Producto.

Semántica: Producto provee la propiedad de relacionar la clase Característica y la clase Activo a estas clases. Producto representa una estructura de almacenamiento que permite poseer los campos para el manejo de la información de los nuevos Productos creados.

4.4 Modelo Funcional de R-GAC

En Small SPL permite la interface de los otros dos subprocesos interdependientes: la ingeniería de dominio y la ingeniería de aplicaciones, para un correcto y útil engranaje productivo. En este proyecto, RGAC, la gestión de activos es basada en características y se define como el subproceso conector (Features).

4.4.1 Metáfora Funcional

Feature-Oriented (árbol de features, la selección y la derivación)

Las líneas de productos software, como se mencionó con anterioridad, se gestionan en términos de un conjunto de características. Algunas de las características serán comunes para todos los productos potenciales de la línea y otras características sólo se encontrarán en algunos productos.

El repositorio RGAC se basa conceptualmente en el **árbol de características** corresponde a una vista gráfica de un modelo de características. Las características del sistema están estructuradas de forma jerárquica. Las características pertenecen a los siguientes tipos:

- **Obligatoria**: la característica debe de estar en el producto.
- **Alternativa**: sólo una característica puede ser seleccionada.
- **Opcional**: características que pueden o no estar en el producto final.
- **O**: una o más características pueden ser seleccionadas. Esta relación puede llevar asociada una cardinalidad del tipo n: m, que indica que un mínimo de n características deben de seleccionarse, y como mucho se pueden seleccionar m.

Con las características alternativas y opcionales se representa parte de la variabilidad del sistema. Las características alternativas permiten elegir una característica de un conjunto y las características opcionales pueden ser seleccionadas o no. Podemos encontrar variaciones en cuanto a la capacidad expresiva de los modelos de características.

La variabilidad es la capacidad de un sistema, un activo reutilizable o un entorno de desarrollo, de producir un conjunto de artefactos que difieren entre ellos en la forma que previamente se ha planificado[42].

También por variabilidad entendemos la capacidad o tendencia a cambiar de un artefacto, para ser extendido, personalizado o configurado, con el objetivo de ser usado en un contexto particular[83].

Uno de los puntos claves en el desarrollo de características de productos es identificar cuáles son los aspectos comunes y variables de una misma característica. Cada activo es un miembro de una característica, y esta pertenece a la línea de producto.

Utilizamos el árbol de características para un mejor manejo visual de las características de una línea de productos y de esta manera poder seleccionar los activos para la creación del producto.

4.4.2 Modelo Funcional (Casos de Uso)

RGAC es una aplicación de software, en la cual se han identificado tres roles de usuarios principales, ellos son: ingeniero de dominio, administrador de activos, y el ingeniero de producto.

Actor	Descripción
Ingeniero de Dominio	Usuario encargado de la Gestión de Líneas de Productos, Características y Usuarios.
Administrador de Activos	Usuario encargado de la Gestión de Activos.
Ingeniero de Productos	Usuario encargado de la Gestión de Productos.

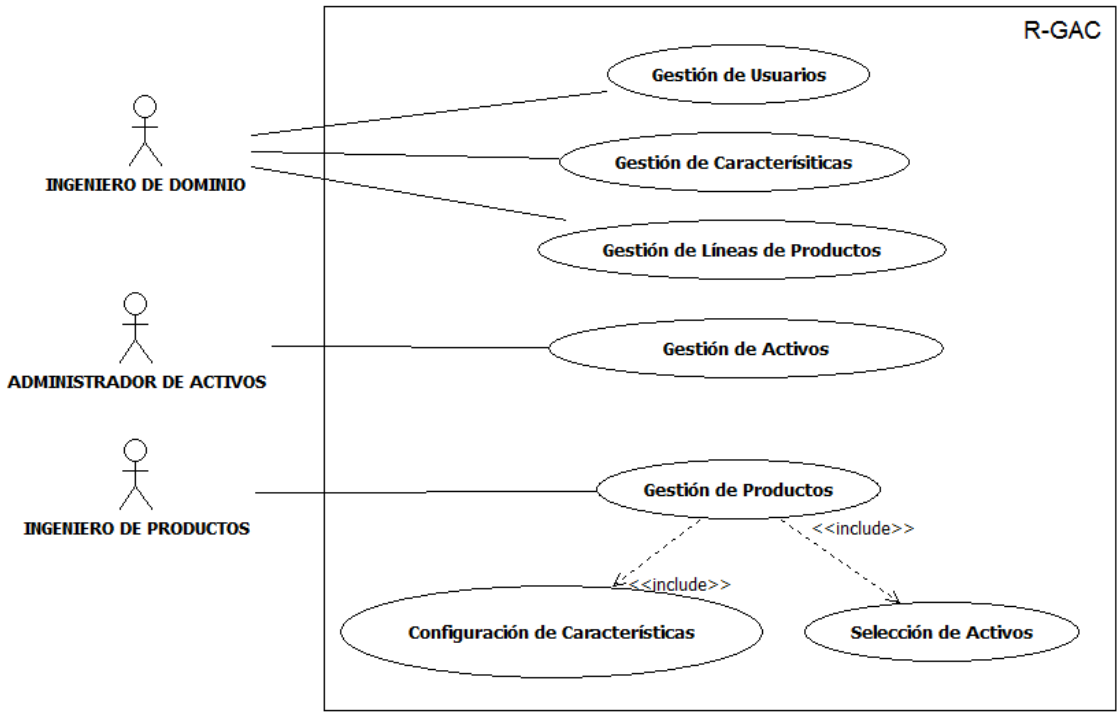


Figura 7. Modelo Funcional de RGAC.

Modelo de Casos de Uso

Los casos de uso nos ayudan a describir un uso del sistema y cómo éste interactúa con el usuario. “Un Caso de Uso es una secuencia de acciones realizadas por el sistema, que producen un resultado valioso para un actor en particular”[84].

A continuación se describe en forma resumida los casos de uso que representan los usos del sistema:

Gestión de Líneas de Productos

Los siguientes casos de uso muestran las operaciones relacionadas con la Gestión de Líneas de Productos.

Figura 8. Gestión de Líneas de Producto.

Caso de uso:	**Adicionar Línea de Producto**
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor adicionar la Línea de Productos.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para crear una nueva Línea de Productos.	2. El sistema muestra una interfaz para introducir nombre, descripción y versión para introducir los datos de la Línea de Producto.
3. El actor introduce los datos solicitados.	4. El sistema verifica los datos y los envía a la base de datos.
	5. El sistema confirma la grabación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	**Modificar Línea de Producto**
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor modificar la Línea de Productos.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para modificar una Línea de Productos.	2. El sistema muestra una interfaz con una lista para elegir la Línea de Producto a modificar.
3. El actor señala en la lista la Línea de Productos elegida.	4. El sistema muestra una interfaz con los datos de la Línea de Producto.
5. El actor selecciona el campo o los campos y los modifica.	6. El sistema verifica los datos y los envía a la base de datos.

Caso de uso:	Modificar Línea de Producto
	7. El sistema confirma la grabación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	Eliminar Línea de Producto
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor eliminar la Línea de Productos.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para eliminar una Línea de Productos.	2. El sistema muestra una interfaz con una lista para elegir la Línea de Producto a eliminar.
3. El actor señala en la lista la Línea de Productos elegida.	4. El sistema pide la confirmación de la eliminación.
5. El actor pulsa sobre el botón para confirmar la eliminación.	7. El sistema elimina los datos de la base de datos.
	8. El sistema confirma la eliminación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	Consultar Línea de Producto
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor consultar los datos de la Línea de Productos.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para consultar los datos de una Línea de Productos.	2. El sistema muestra una interfaz con los datos de la Línea de Producto.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Gestión de Características

Los siguientes casos de uso muestran las operaciones relacionadas con la Gestión de Características.

Figura 9. Gestión de Características.

Caso de uso:	**Adicionar Característica**
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor adicionar la Característica.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para crear una nueva Característica.	2. El sistema muestra una interfaz para introducir nombre, descripción, versión, línea de producto, tipo, y Característica padre para introducir los datos de la Característica.
3. El actor introduce los datos solicitados.	4. El sistema verifica los datos y los envía a la base de datos.
	5. El sistema confirma la grabación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	**Modificar Característica**
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor modificar la Característica.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para modificar una Característica.	2. El sistema muestra una interfaz con una lista para elegir la Característica a modificar.

Caso de uso:	**Modificar Característica**
3. El actor señala en la lista la Característica elegida.	4. El sistema muestra una interfaz con los datos almacenados para realizar los cambios de la Característica.
5. El actor selecciona el campo o los campos y los modifica.	6. El sistema verifica los datos y los envía a la base de datos.
	7. El sistema confirma la grabación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	**Eliminar Característica**
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor eliminar la Característica.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para eliminar una Característica.	2. El sistema muestra una interfaz con una lista para elegir la Característica a eliminar.
3. El actor señala en la lista la Característica elegida.	4. El sistema pide la confirmación de la eliminación.
5. El actor pulsa sobre el botón para confirmar la eliminación.	7. El sistema elimina los datos de la base de datos.
	8. El sistema confirma la eliminación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	**Consultar Característica**
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor consultar los datos de las Características.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para consultar los datos de una Característica.	2. El sistema muestra una interfaz con los datos de la Característica.

Caso de uso:	Consultar Característica
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Gestión de Activos

Los siguientes casos de uso muestran las operaciones relacionadas con la Gestión de Activos.

Figura 10. Gestión de Activos.

Caso de uso:	Adicionar Activo
Actor:	Administrador de Activo
Descripción:	Este caso de uso permite al actor adicionar el Activo.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para crear un nuevo Activo.	2. El sistema muestra una interfaz para introducir nombre, descripción, versión, línea de producto, tipo, y Característica padre para introducir los datos del Activo.
3. El actor introduce los datos solicitados.	4. El sistema verifica los datos y los envía a la base de datos.
	5. El sistema confirma la grabación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	Modificar Activo
Actor:	Administrador de Activos
Descripción:	Este caso de uso permite al actor modificar el Activo.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	

Caso de uso:	**Modificar Activo**
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para modificar un Activo.	2. El sistema muestra una interfaz con una lista para elegir el Activo a modificar.
3. El actor señala en la lista el Activo elegido.	4. El sistema muestra una interfaz con los datos almacenados para realizar los cambios del Activo.
5. El actor selecciona el campo o los campos y los modifica.	6. El sistema verifica los datos y los envía a la base de datos.
	7. El sistema confirma la grabación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	**Eliminar Activo**
Actor:	Administrador de Activos
Descripción:	Este caso de uso permite al actor eliminar el Activo.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para eliminar un Activo.	2. El sistema muestra una interfaz con una lista para elegir el Activo a eliminar.
3. El actor señala en la lista el Activo elegido.	4. El sistema pide la confirmación de la eliminación.
5. El actor pulsa sobre el botón para confirmar la eliminación.	7. El sistema elimina los datos de la base de datos.
	8. El sistema confirma la eliminación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	**Consultar Activo**
Actor:	Administrador de Activos
Descripción:	Este caso de uso permite al actor consultar los datos del Activo.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para consultar los datos de un Activo.	2. El sistema muestra una interfaz con los datos del Activo.

Caso de uso:	Consultar Activo
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Gestión de Productos

Los siguientes casos de uso muestran las operaciones relacionadas con la Gestión de Productos.

Figura 11. Gestión de Productos.

Caso de uso:	Adicionar Producto
Actor:	Ingeniero de Productos
Descripción:	Este caso de uso permite al actor adicionar el Producto.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para crear un nuevo Producto.	2. El sistema muestra una interfaz con un árbol para elegir los Activos que pueden hacer parte del nuevo Producto.
3. El actor selecciona los activos deseados.	4. El sistema verifica los activos y los envía a la base de datos.
	5. El sistema confirma la grabación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	Eliminar Producto
Actor:	Ingeniero de Productos
Descripción:	Este caso de uso permite al actor eliminar el Producto.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**

Caso de uso:	**Eliminar Producto**
1. El actor pulsa sobre el botón para eliminar un Producto.	2. El sistema muestra una interfaz con una lista para elegir el Producto a eliminar.
3. El actor señala en la lista el Producto elegido.	4. El sistema pide la confirmación de la eliminación.
5. El actor pulsa sobre el botón para confirmar la eliminación.	7. El sistema elimina los datos de la base de datos.
	8. El sistema confirma la eliminación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	**Consultar Producto**
Actor:	Ingeniero de Productos
Descripción:	Este caso de uso permite al actor consultar los datos de los Productos.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para consultar los datos de un Producto.	2. El sistema muestra una interfaz con los datos del Producto.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Gestión de Usuarios

Los siguientes casos de uso muestran las operaciones relacionadas con la Gestión de Usuarios.

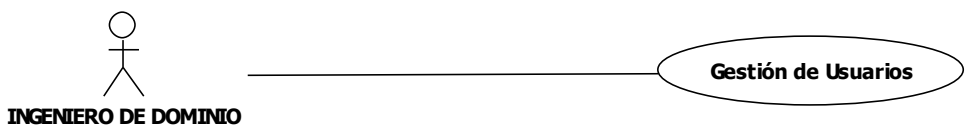


Figura 12. Gestión de Usuarios.

Caso de uso:	**Adicionar Usuario**
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor adicionar el Usuario.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para crear un nuevo Usuario.	2. El sistema muestra una interfaz para introducir usuario, contraseña y tipo para introducir los datos del Usuario.
3. El actor introduce los datos solicitados.	4. El sistema verifica los datos y los envía a la base de datos.
	5. El sistema confirma la grabación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	**Modificar Usuario**
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor modificar el Usuario.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para modificar un Usuario.	2. El sistema muestra una interfaz con una lista para elegir el Usuario a modificar.
3. El actor señala en la lista el Usuario elegido.	4. El sistema muestra una interfaz con los datos almacenados para realizar los cambios del Usuario.
5. El actor selecciona el campo o los campos y los modifica.	6. El sistema verifica los datos y los envía a la base de datos.
	7. El sistema confirma la grabación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	**Eliminar Usuario**
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor eliminar el Usuario.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	

Caso de uso:	Eliminar Usuario
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para eliminar un Usuario.	2. El sistema muestra una interfaz con una lista para elegir el Usuario a eliminar.
3. El actor señala en la lista el Usuario elegido.	4. El sistema pide la confirmación de la eliminación.
5. El actor pulsa sobre el botón para confirmar la eliminación.	7. El sistema elimina los datos de la base de datos.
	8. El sistema confirma la eliminación de los datos.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

Caso de uso:	Consultar Usuario
Actor:	Ingeniero de Dominio
Descripción:	Este caso de uso permite al actor consultar los datos del Usuario.
Pre-Condición:	El usuario debe estar autenticado al sistema.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El actor pulsa sobre el botón para consultar los datos de un Usuario.	2. El sistema muestra una interfaz con los datos del Usuario.
Post-Condiciones	El sistema que use este servicio, será quien decida qué hacer con los datos obtenidos en la ejecución exitosa de este servicio.

4.5 Modelo del Diseño de R-GAC

A continuación, se describe el Modelo del Diseño de R-GAC con los Criterios de Diseño y las Vistas Arquitectónicas del sistema:

4.5.1 Criterios de Diseño

Como lenguaje de programación para la aplicación se utiliza Java con JSF (Java Server Faces) para la creación de las interfaces del sistema, JPA (Java Persistent Applications) para el manejo de datos relacionados y el servidor de aplicación utilizado es el GlassFish Server 4. Para la gestión de la base de datos se utiliza el MySQL.

4.5.2 Vistas Arquitectónicas

A continuación veremos el modelo lógico, el modelo de comportamiento y el modelo de datos del sistema.

4.5.2.1 Modelo Lógico del Aplicativo

En el modelo lógico del sistema veremos el modelo de paquetes y también el modelo de componentes y los conectores de la aplicación.

4.5.2.1.1 Modelo de Paquetes

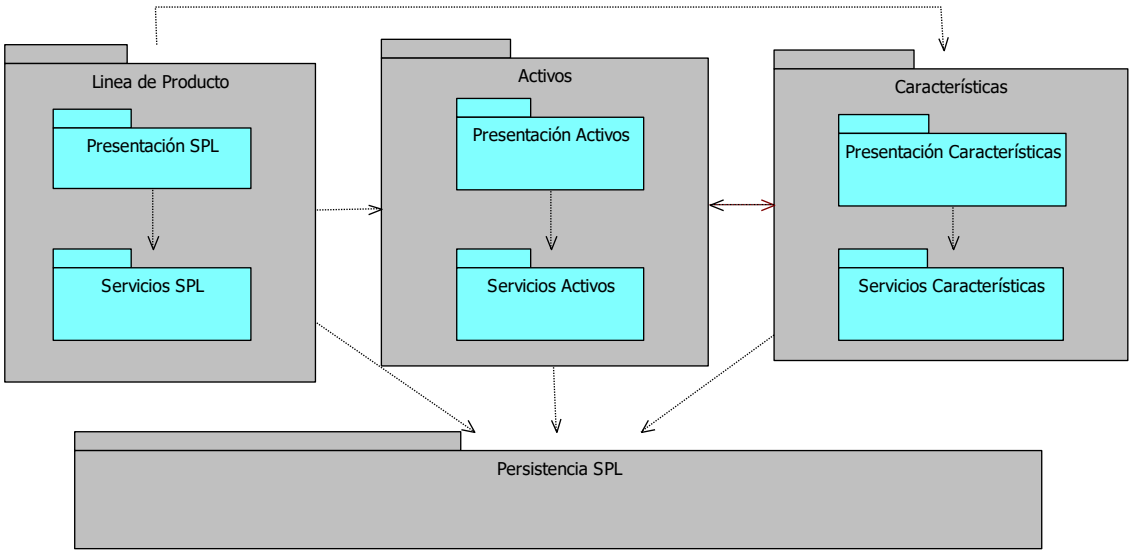


Figura 13. Modelo de Paquetes del Sistema RGAC.

- **Capa de Línea de Productos:** Esta capa representa la Gestión de las Líneas de Productos utilizadas en RGAC para la funcionalidad necesaria en el manejo del sistema. Se comunica con la capa de los Activos y la capa de Características para la realización de la Gestión de Productos, así como con la capa de Persistencia SPL para el manejo de la información en la base de datos. Posee los elementos de la Presentación SPL, así como la de Servicios SPL que hacen parte de su manejo.

- **Capa de Activos:** Esta capa tiene la responsabilidad de administrar los Activos que hacen parte de la Gestión de Productos. Se comunica con la capa de Línea de Productos y la capa de Características, cuales los clasifican de acuerdo a su funcionalidad, así como con la capa de Persistencia SPL para el manejo de los datos del sistema. Esta capa tiene como elementos la Presentación de Activos y Servicios de Activos para obtener la funcionalidad necesaria del sistema.

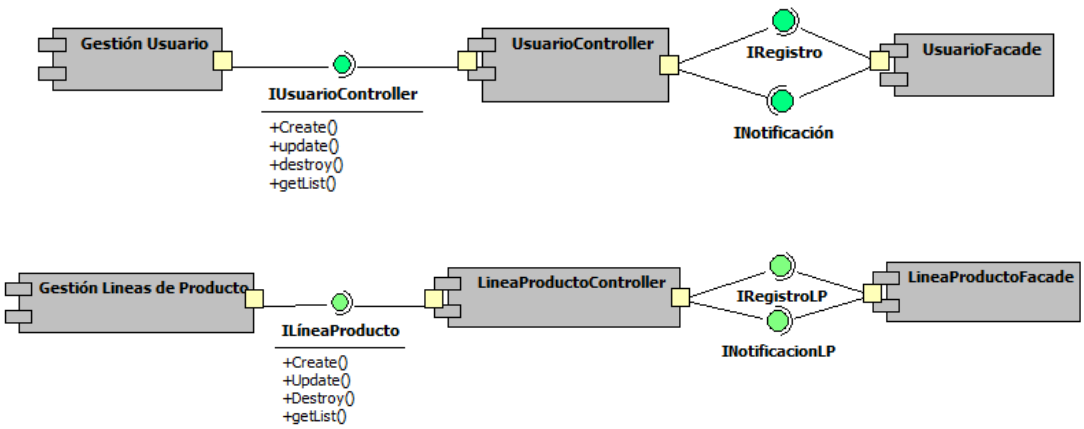
- **Capa de Característica:** Esta capa encapsula las funciones correspondientes a la Gestión de las Características, para la realización de las funciones utilizadas para la Gestión de Productos que se realizan en el sistema. La Capa de Características está asociada con las Líneas de Productos en la cual hace parte, así como en los Activos que hacen parte de estas Características. También se asocia con la capa de Persistencia SPL para la gestión en la base de datos. Para realizar las funciones del sistema posee los elementos de Presentación de Características y Servicios de Características.

- **Capa de Persistencia SPL:** Esta capa posee las funciones del manejo de los datos del sistema, para la realización de la Gestión de Productos. Se conecta con las demás capas para efectuar la ejecución de instrucciones SQL en la base de datos.

En general, cada capa posee diferentes niveles de funcionalidad que son desarrollados, empaquetados e instalados en distintos módulos, componentes y nodos, cuales mostraremos en los siguientes capítulos.

4.5.2.1.2 Modelo de Componentes y Conectores (Tiers)

A continuación veremos la representación del modelo de componentes y conectores del sistema.



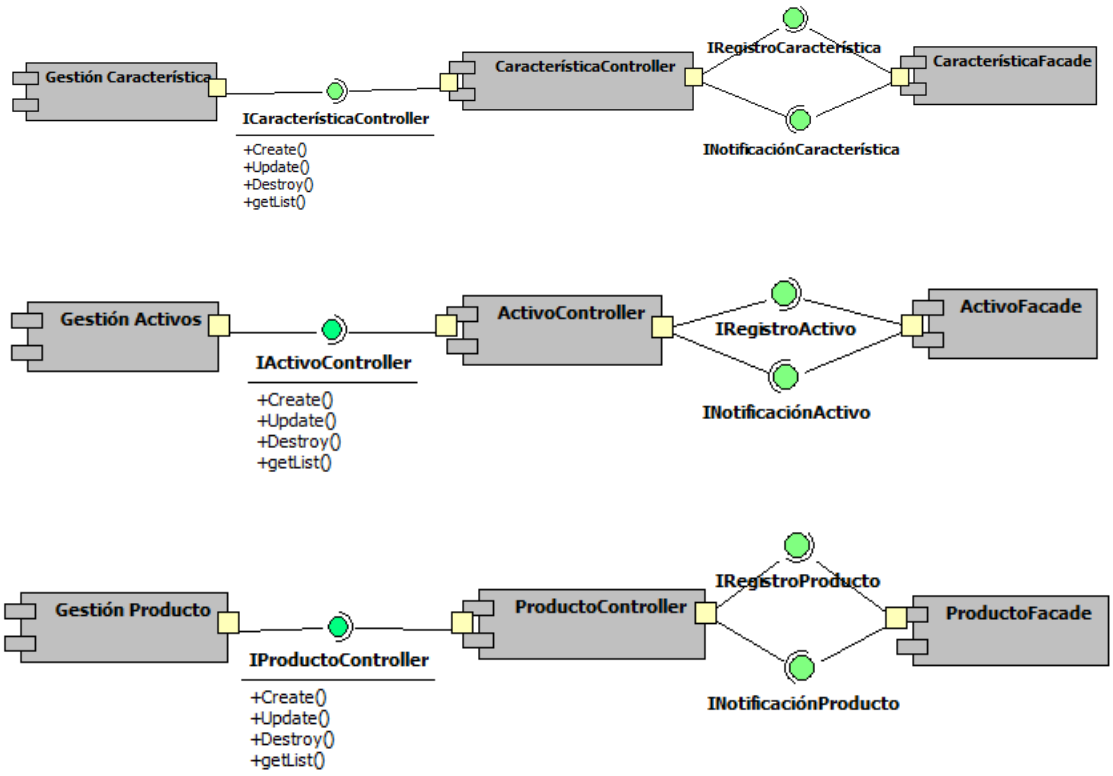


Figura 14. Modelos de Componentes y Conectores de RGAC.

4.5.2.2 Modelo de Datos

A continuación veremos la representación de los datos del sistema en forma general, así como la relación entre ellos.

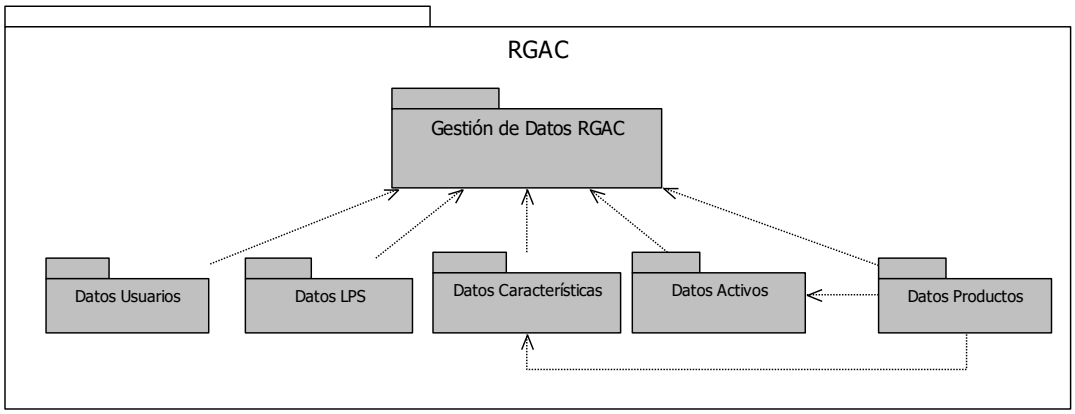


Figura 15. Representación de los datos de RGAC.

Ahora mostraremos en la Figura siguiente la presentación del modelo relacional del repositorio R-GAC:

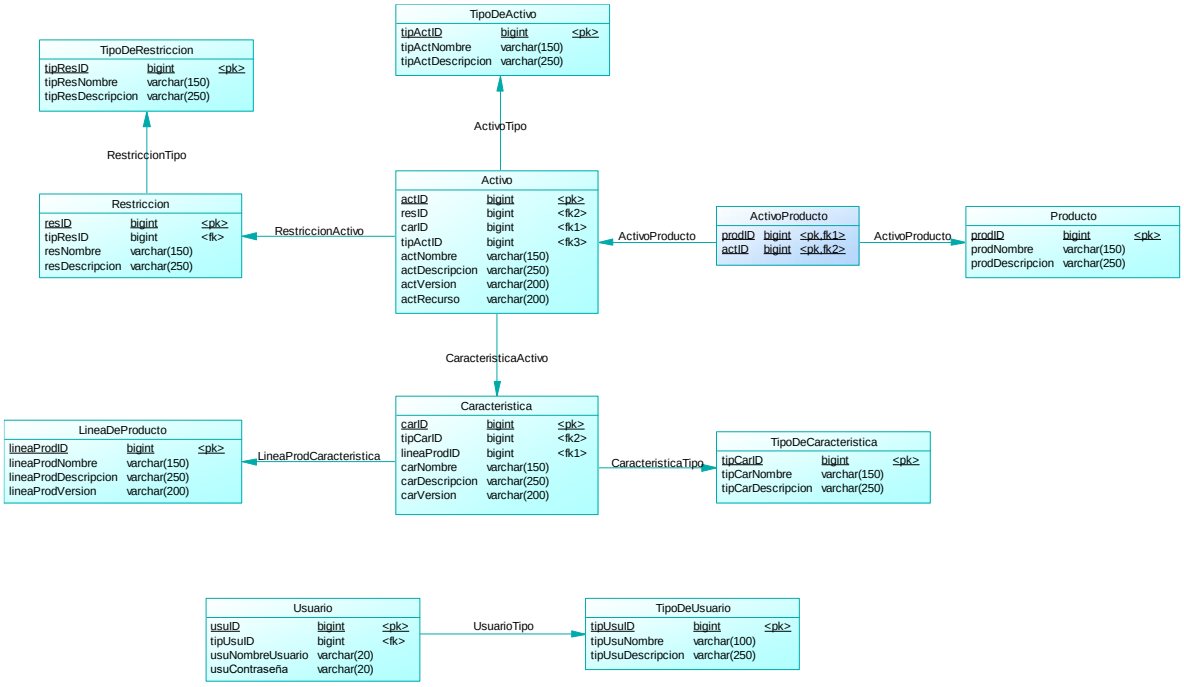


Figura 16. Modelo Relacional de R-GAC.

Las principales entidades son las siguientes:

- LineaDeProducto: Esta entidad guarda los datos de las Líneas de Productos. Contiene los siguientes datos:
  - lineaProdID: Este campo es la llave primaria de la entidad Línea de Producto.
  - lineaProdNombre: En este campo se almacena el nombre de la Línea de Producto.
  - lineaProdDescripcion: En este campo se representa la descripción de la Línea de Producto.
  - lineaProdVersion: Este campo es la versión de la Línea de Producto.

- Caracteristica: Esta entidad guarda los datos de las Características de las Líneas de Productos.

  Contiene los siguientes datos:

  - carID: Este campo es la llave primaria de la entidad Característica.
  - tipCarID: Este campo es la llave primaria del Tipo de Característica.
  - lineaProdID: En este campo se almacena el identificador de la Línea de Produto al cual pertenece la Característica.
  - carNombre: En este campo se almacena el nombre de la Característica.

- carDescripcion: En este campo se representa la descripción de la Característica.
- carVersion: Este campo es la versión de la Característica.

- Activo: Esta entidad guarda los datos de los Activos de las Características.
  Contiene los siguientes datos:
  - actID: Este campo es la llave primaria de la entidad Activo.
  - restID: Este campo es la llave primaria de la Restricción del Activo.
  - carID: Este campo es la llave primaria de la entidad Característica a la cual pertenece el Activo.
  - tipActID: Este campo es la llave primaria del Tipo de Activo.
  - actNombre: En este campo se almacena el nombre del Activo.
  - actDescripcion: En este campo se representa la descripción del Activo.
  - actVersion: Este campo es la versión del Activo.
  - actRecurso: En este campo se almacena la ubicación del Activo.

- Producto: Esta entidad guarda los datos de los Productos creados en el repositorio.
  Contiene los siguientes datos:
  - prodID: Este campo es la llave primaria de la entidad Producto.
  - prodNombre: En este campo se almacena el nombre del Producto.
  - prodDescripcion: En este campo se representa la descripción del Producto.

- Usuario: Esta entidad guarda los datos de los Usuarios del repositorio.
  Contiene los siguientes datos:
  - usuID: Este campo es la llave primaria de la entidad Usuario.
  - tipUsuID: Este campo es la llave primaria del Tipo de Usuario.
  - usuNombreUsuario: En este campo se almacena el nombre del Usuario.
  - usuContraseña: En este campo se representa la contraseña del Usuario.

4.5.2.2.1 Modelo de Implementación de las Capas

El sistema de R-GAC se compone en su arquitectura de tres capas: Capa de Datos, Capa de Negocios y Capa de Presentación.

La Capa de Datos posee elementos que en conjunto nos permiten salvaguardar y hacer persistente la información completa de los activos de software y los activos de base conceptual, La Capa de Negocios nos integra los elementos que permiten la gestión del repositorio y La Capa de Presentación contiene componentes que nos administran la comunicación con los usuarios del sistema, permiten presentar los activos de software, los activos de base conceptual y toda la información y herramientas relacionadas.

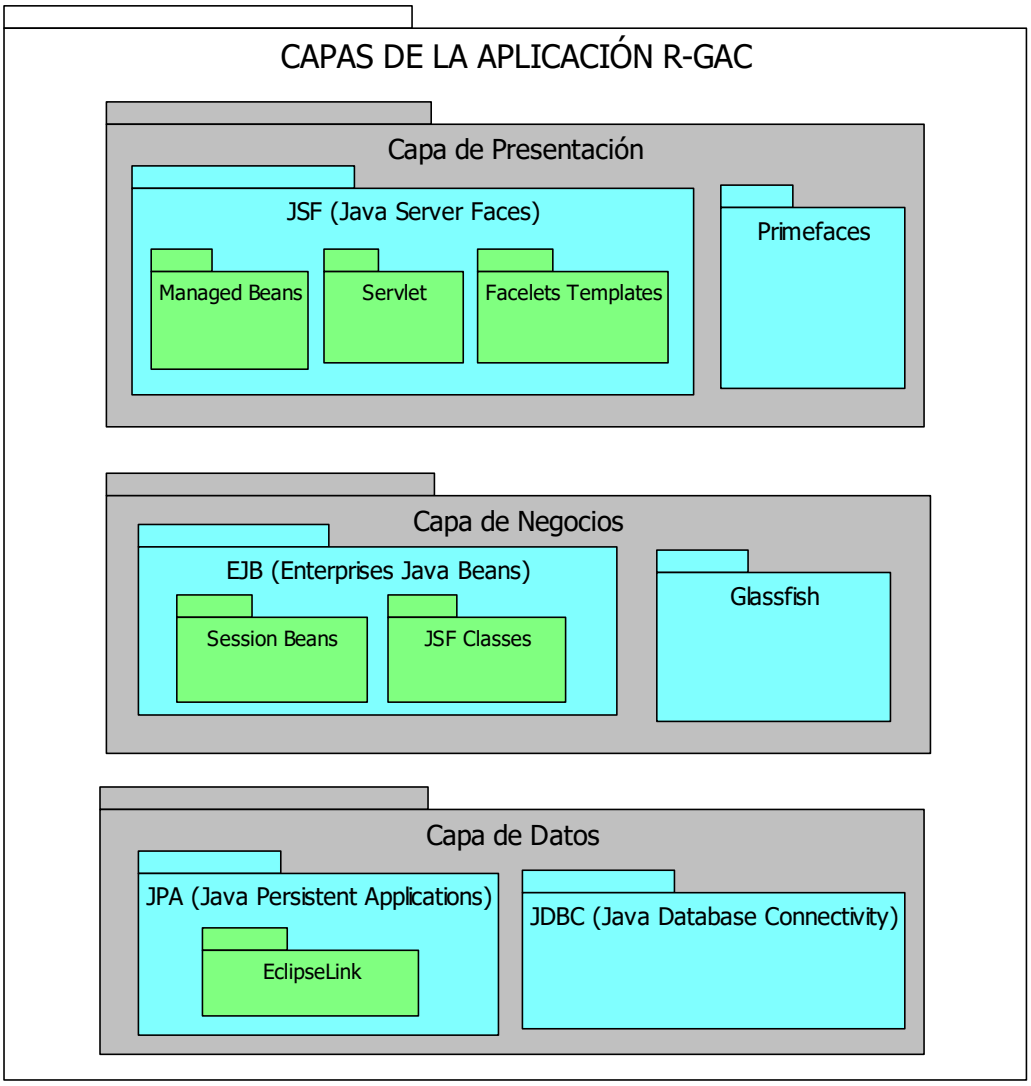


Figura 17. Modelo de Implementación de las Capas de RGAC.

Capa de Presentación

A continuación veremos los elementos que posee la capa de presentación de la aplicación:

- JSF: (Java Server Faces) Es una tecnología y marco de trabajo (framework) para crear la aplicación Java basada en web que nos ayuda a simplificar el desarrollo de interfaces de usuario en Java EE, habilidad para separar los componentes en diferentes archivos y un buen sistema para el reporte de errores. JSF usa Managed Beans en la creación de las clases de las entidades de la base de datos, Servlet para la gestión de peticiones realizadas desde la interfaz de usuario y Facelets Templates como plantillas de la página web que nos ayuda para obtener la

aplicación en un menor tiempo de desarrollo. También utiliza JavaServer Pages (JSP) como la tecnología que nos permite hacer el despliegue de las páginas[85].

- Primefaces: Es una librería de componentes o extensión para JavaServer Faces (JSF) de código abierto que nos facilita la creación de la aplicación web ya que cuenta con un conjunto de componentes enriquecidos que nos ayuda a organizar la interfaz de usuario[86]. Es muy sencillo de utilizar, ligero y fácil de configurar para integrarlo en la aplicación. Primefaces es una librería con simplicidad, rendimiento y fácil uso para la ayuda del desarrollo de la aplicación. Primefaces nos ayudó con gran cantidad de componentes disponibles en la creación de la aplicación para un desarrollo más rápido.

Capa de Negocios

A continuación veremos los elementos que posee la capa de negocios de nuestra aplicación:

- EJB: (Enterprise JavaBeans) Son una de las API (Intefaz de Programación de Aplicaciones) que forman parte del estándar de construcción de la aplicación J2EE (ahora JEE)[87]. EJB usa a SessionBeans para la creación de los modelos del proyecto y JSF Classes para la creación de las clases de las entidades de la base de datos, para la realización del control de estas mismas. Se utilizó la versión 3.1 para esta aplicación.

- Glassfish: Es un servidor de aplicaciones de software libre que implementa las tecnologías en la plataforma Java Enterprise Edition (JEE) y que permite ejecutar aplicaciones que siguen con esta especificación[88]. Se utilizó la versión 4 para esta aplicación.

Capa de Datos

La capa de datos de nuestra aplicación posee los siguientes elementos:

- JPA: (Java Persistent Applications) Es un marco de trabajo del lenguaje de programación Java que nos maneja los datos relacionales en nuestra aplicación, usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE)[89]. JPA utiliza el marco de trabajo EclipseLink para interactuar con la base de datos, los servicios web del sistema y otras funciones.

- JDBC: (Java Database Connectivity) Es una API (Interfaz de Programa de Aplicaciones) que nos permite realizar la ejecución de operaciones en la base de

datos desde el lenguaje Java y con el dialecto SQL(Structure Query Language) del modelo de la base de datos utilizada[90].

4.5.2.2.2 Modelo de Comportamiento

A continuación veremos la descripción del comportamiento del sistema.

- **View Package Interacción Unidades Gestión de Usuarios:**

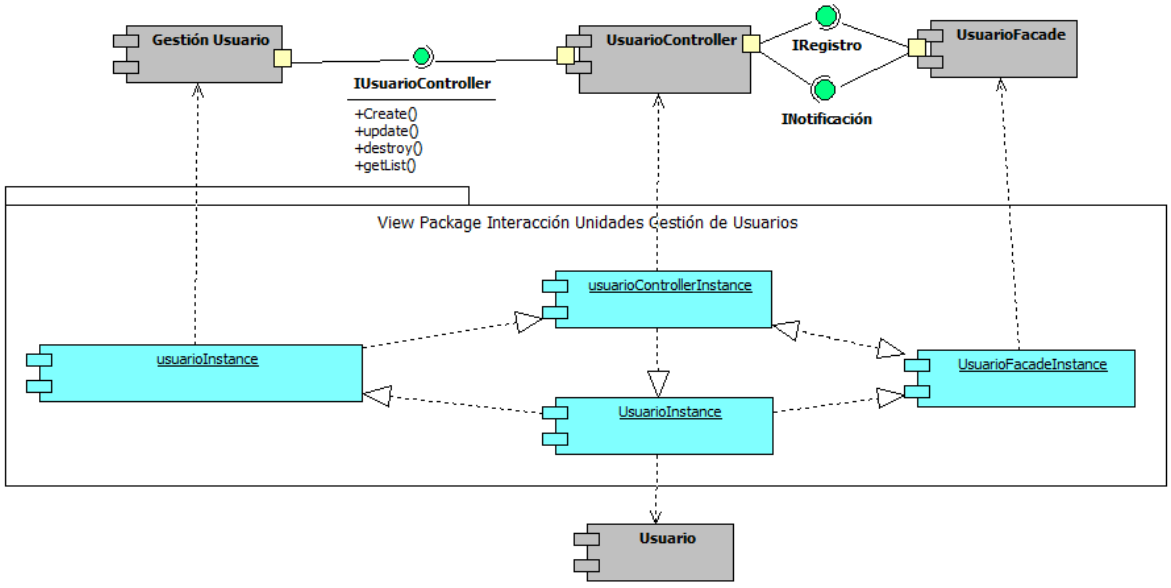
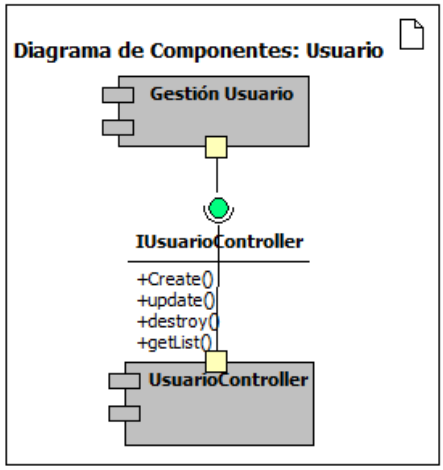


Figura 18. C&C View para View Package Interacción Unidades Gestión de Usuarios.

Catálogo de Componentes

Nombre del View Package	Interacción Unidades Gestión de Usuarios
Rationale	El patrón arquitectónico de Cliente-Servidor se considera adecuado ya que brinda gran facilidad de adicionar nuevos usuarios sin un mayor impacto en la arquitectura. Si la aplicación cambia en algún momento, dicho cambio no afecta la lógica que se viene manejando en el proceso de la Gestión de Usuarios.
Componente	usuarioController
Descripción	Este componente tiene como objetivo realizar la gestión de los datos que se requieren para el manejo de los usuarios del sistema.

Composición interna	

- **View Package Interacción Unidades Gestión de Línea de Productos:**

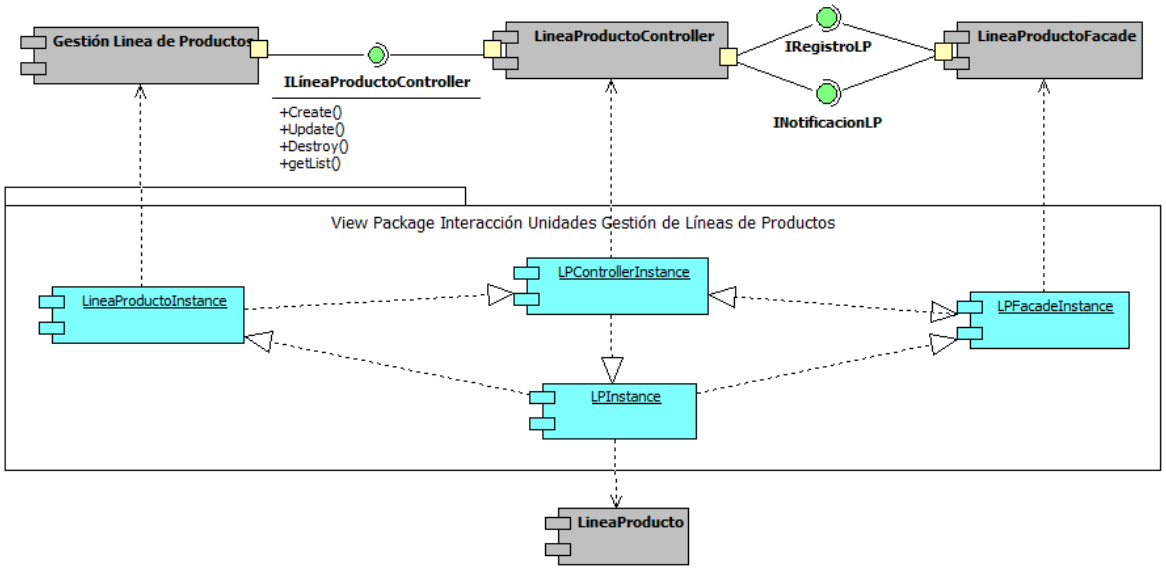
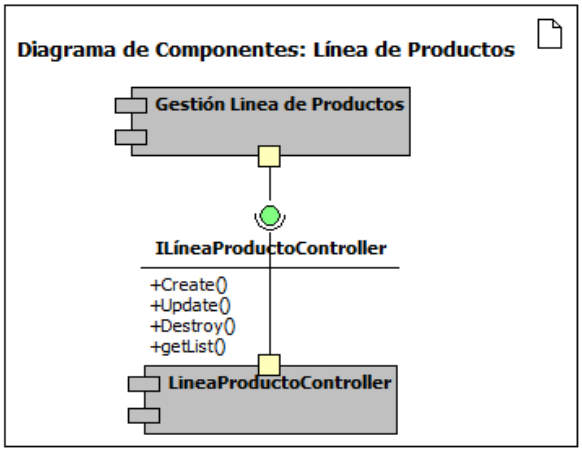


Figura 19. C&C View para View Package Interacción Unidades Gestión Línea de Productos.

Catálogo de Componentes

Nombre del View Package	Interacción Unidades Gestión de Líneas de Productos
Rationale	El patrón arquitectónico de Cliente/Servidor nos ayuda en la integración entre diferentes sistemas, además nos facilita también la integración de nuevas tecnologías. Si la aplicación cambia en algún momento, dicho cambio no debería afectar la

	lógica que se viene manejando en el proceso de la Gestión de Línea de Productos.
Componente	LíneaProductoController
Descripción	Este componente es el encargado de realizar la gestión de las líneas de productos que se requieren para el manejo de los activos del sistema.
Composición interna	

- **View Package Interacción Unidades Gestión de Características:**

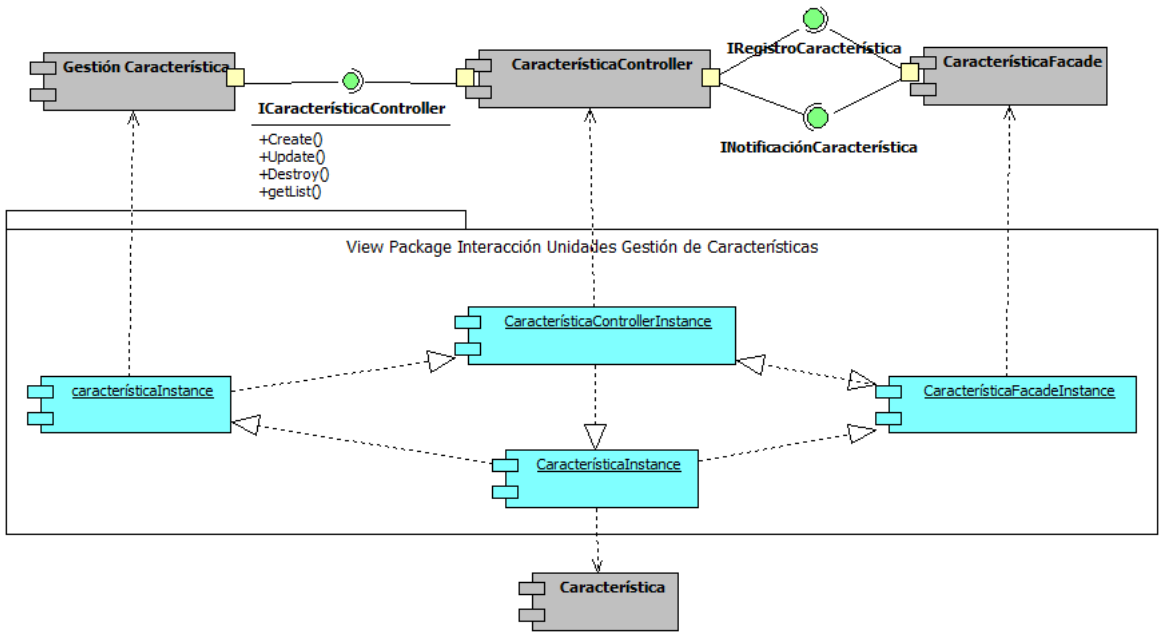
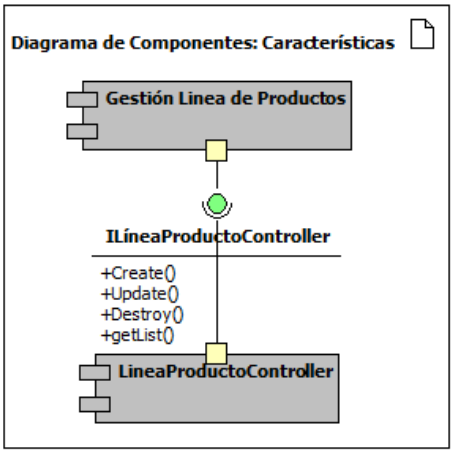


Figura 20. C&C View para View Package Interacción Unidades Gestión de Características.

Catálogo de Componentes

Nombre del View Package	Interacción Unidades Gestión de Características
Rationale	El patrón arquitectónico de Cliente/Servidor facilita la integración entre diferentes sistemas, además modular también facilita la integración de nuevas tecnologías. Si la aplicación realiza un cambio en algún momento, no debería afectar en la lógica que se viene manejando para el proceso de la Gestión de Características.
Componente	CaracterísticaController
Descripción	Este componente tiene como objetivo ejecutar la gestión de las características que se necesitan para el manejo de los activos del sistema.
Composición interna	

- **View Package Interacción Unidades Gestión de Activos:**

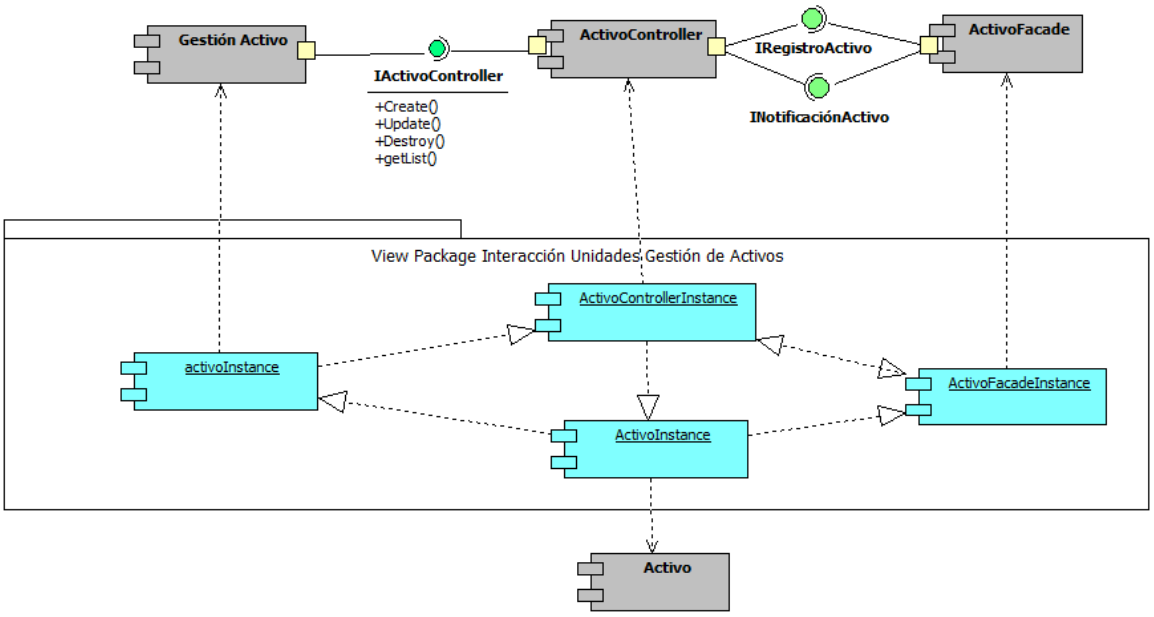
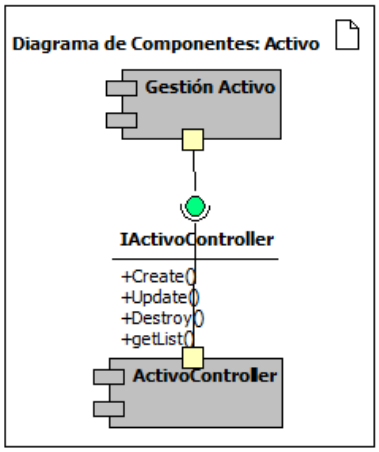


Figura 21. C&C View para View Package Interacción Unidades Gestión de Activos.

Catálogo de Componentes

Nombre del View Package	Interacción Unidades Gestión de Activos
Rationale	Se selecciona el patrón arquitectónico de Cliente/Servidor, ya que facilita la integración entre diferentes sistemas y también en la integración de nuevas tecnologías. Si la aplicación luego cambia, esto no debería afectar la lógica que se viene manejando en el proceso de la Gestión de Activos.
Componente	ActivoController
Descripción	Este componente está encargado de realizar la gestión de los activos que se solicitan para el manejo de la creación de los productos en el sistema.
Composición interna	

- **View Package Interacción Unidades Gestión de Productos:**

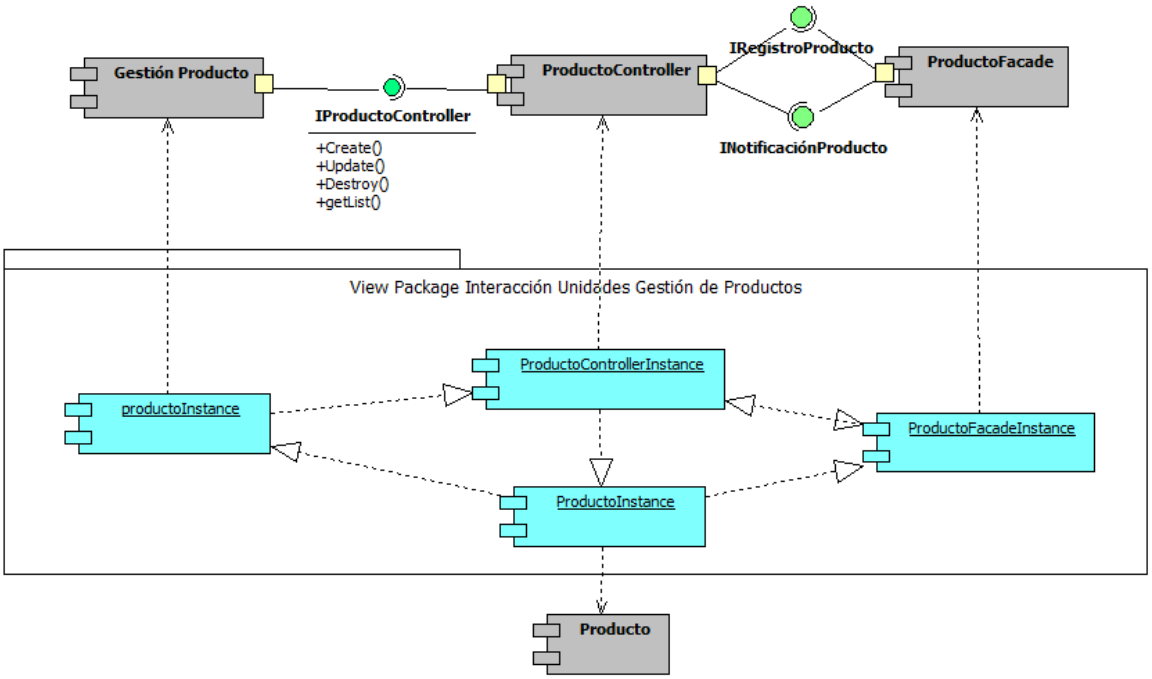
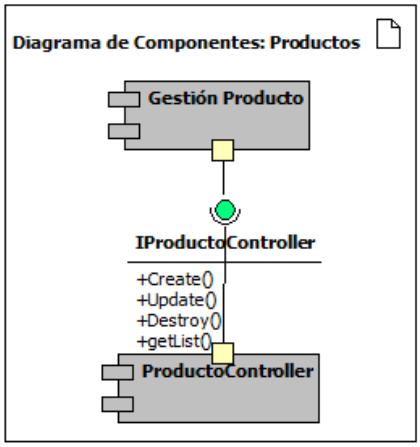


Figura 22. C&C View para View Package Interacción Unidades Gestión de Productos.

Catálogo de Componentes

Nombre del View Package	Interacción Unidades Gestión de Productos
Rationale	El patrón arquitectónico de Cliente/Servidor nos facilita la integración entre diferentes sistemas, y también la modificación se facilita al realizar la integración de nuevas procesos. Si la aplicación cambia en algún momento, dicho cambio no afecta la lógica que se viene manejando en el proceso de la Gestión de Productos.
Componente	ActivoController
Descripción	Este componente tiene como objetivo realizar la gestión de los productos que son creados en el sistema.
Composición interna	

4.5.2.2.3 Modelo Físico

El modelo físico nos muestra la infraestructura necesaria para dar soporte al repositorio RGAC. A continuación veremos el modelo físico del sistema en forma general:

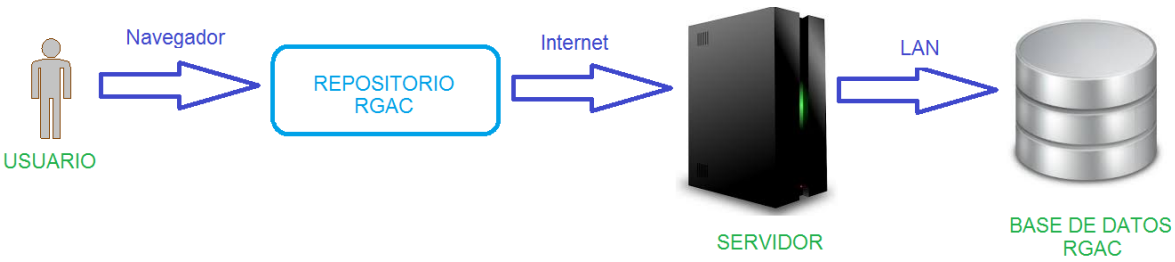


Figura 23. Modelo Físico de RGAC.

Ahora en la siguiente figura mostraremos la vista de despliegue con los componentes principales del sistema:

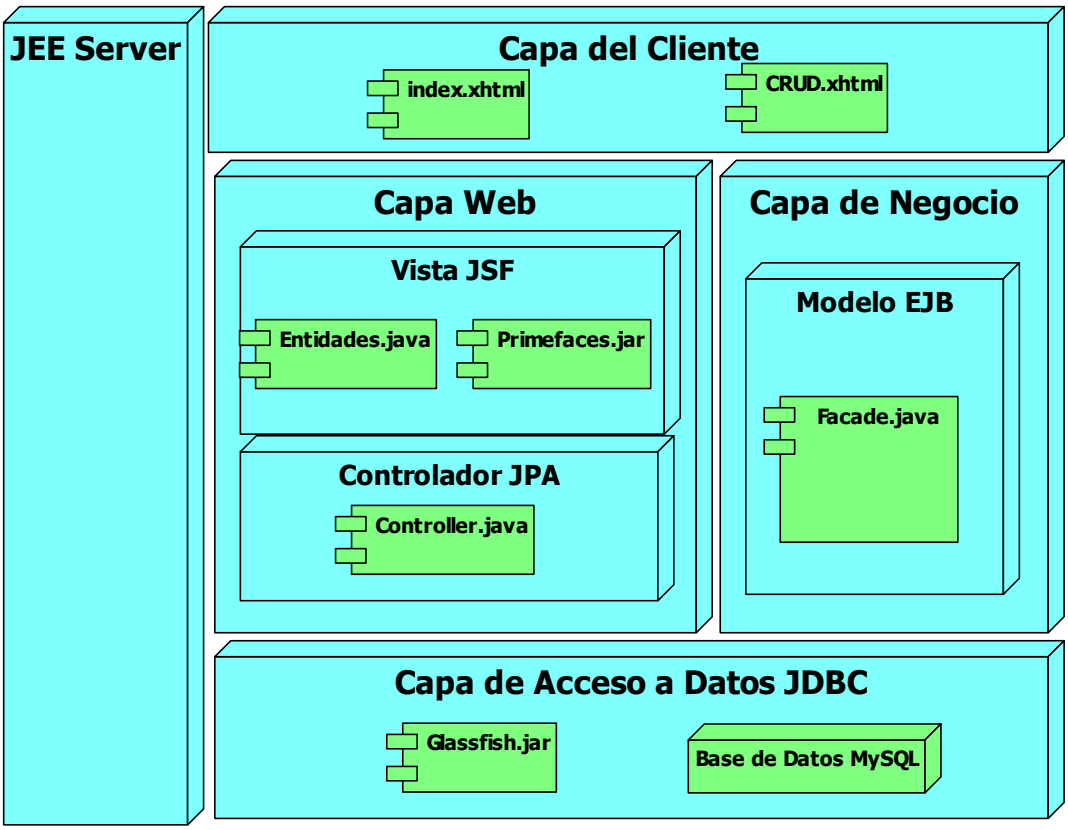


Figura 24. Modelo Físico General con los componentes de despliegue de RGAC.

4.6 Implementación del Repositorio R-GAC

A continuación veremos las Plataformas de Implementación y el Ambiente de desarrollo de RGAC:

4.6.1 Plataformas de implementación

En este apartado se describe tanto los lenguajes de programación usados como los entornos en los que se ha desarrollado el proyecto.

Los lenguajes de programación web utilizados son:

- XHTML: (eXtensible HyperText Markup Languaje) Es un lenguaje de marcado expresado como XML con la que se creó la página web del sistema. Este lenguaje define una estructura básica y un código para la definición del contenido de las páginas web, como son el texto, las imágenes, entre otros.

- JAVA: Es un lenguaje de programación de propósito general, concurrente, basado en clases, y orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible[91].

- XML: Es un lenguaje de marcas que permite definir la gramática de lenguajes específicos para estructurar documentos grandes. Se utiliza en la aplicación para guardar configuraciones como por ejemplo los ficheros “web.xml” o “persistent.xml”.

Los marcos de trabajo o frameworks utilizados para la elaboración de esta aplicación son:

- Java EE: (Java Platform Enterprise Edition) Es la plataforma de programación en la que se desarrolla y se ejecuta la aplicación. Nos permite utilizar una arquitectura de capas distribuidas y se apoya en componentes de software modulares ejecutándose en el servidor Glassfish. Java EE configura unas especificaciones que utilizamos como son el JavaServer Pages, JDBC JavaBeans, Servlets, entre otras.

- JSF: (Java Server Faces) El desarrollo de la aplicación web se realizó con el marco de trabajo Java Server Faces para la creación de las funciones de la gestión de los datos, utilizando sus componentes JavaServer Pages para el despliegue en las páginas web y con Managed Beans para el manejo de las funciones. En este marco de trabajo también se utilizaron las plantillas facelets y la librería Primefaces para la creación de las interfaces de usuario, la cual presentaremos más adelante.

- JPA: **(**Java Persistent Applications) Este marco de trabajo lo utilizamos en la aplicación para el manejo de los datos relacionales del sistema. JPA utiliza la extensión llamada EclipseLink para la realización de la gestión de manera persistente, la cual presentaremos a continuación.

- EclipseLink: Este software provee un extensible marco de trabajo que nos permite interactuar con la base de datos y el servidor web.

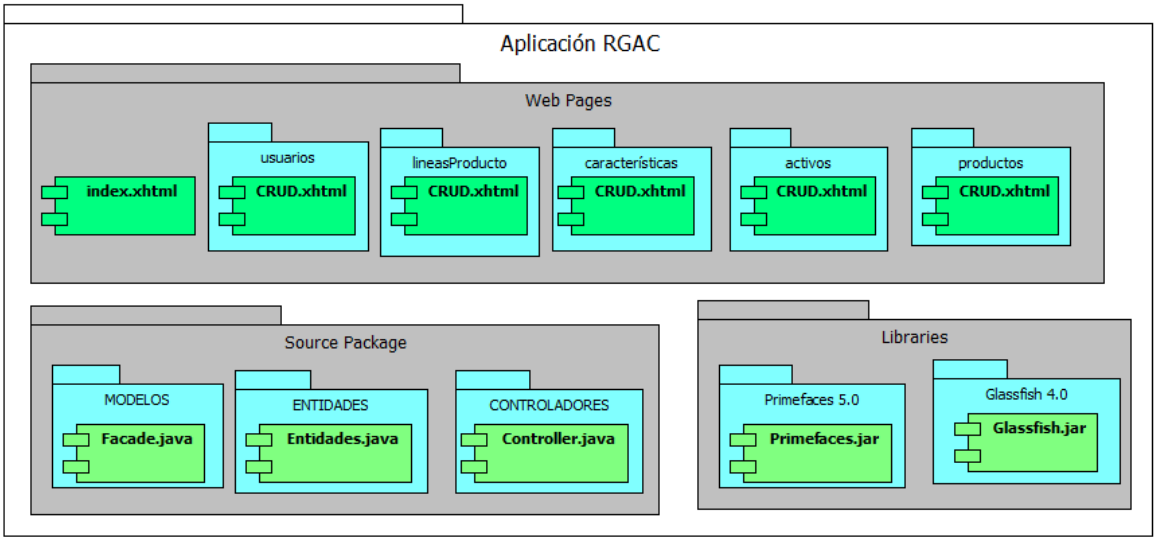


Figura 25. Plataformas de Implementación de RGAC.

Las herramientas utilizadas para la elaboración de esta aplicación son las siguientes:

- Netbeans: Esta herramienta nos permite editar, debugar, compilar y construir las interfaces web de la aplicación. La versión 8.0 fue la que se utilizó para la creación de la aplicación.

- MySQL: Para la gestión de la base de datos de la aplicación se implementó con esta herramienta.

Al final del capítulo se encuentran los principales casos de pruebas probados para verificar el correcto funcionamiento de la aplicación.

4.6.2 Ambientes de Desarrollo

RGAC utiliza como patrón de arquitectura de software el Modelo-Vista-Controlador. El cliente en la interfaz realiza las peticiones, cuales las recibe el Controlador. Luego el Controlador accede al Modelo para manejar la información de RGAC y delega a la Vista para que reciba los resultados del Modelo y los muestre en la interfaz. A continuación mostraremos la gráfica del Ambiente de Desarrollo de la aplicación:

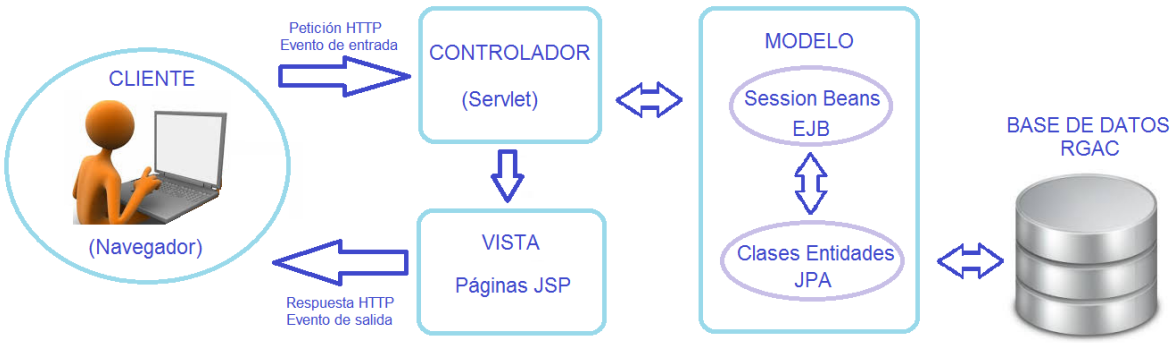


Figura 26. Ambiente de Desarrollo de RGAC.

4.6.3 Implementación de la Base de Datos

Para la realización de la Implementación de la Base de Datos se utilizó JPA (Java Persistent Applications) de la versión 2.0 que se utilizó para el manejo de los datos relacionales en las bases de datos del sistema. A continuación mostraremos la gráfica de la implementación de la Base de Datos de la aplicación:

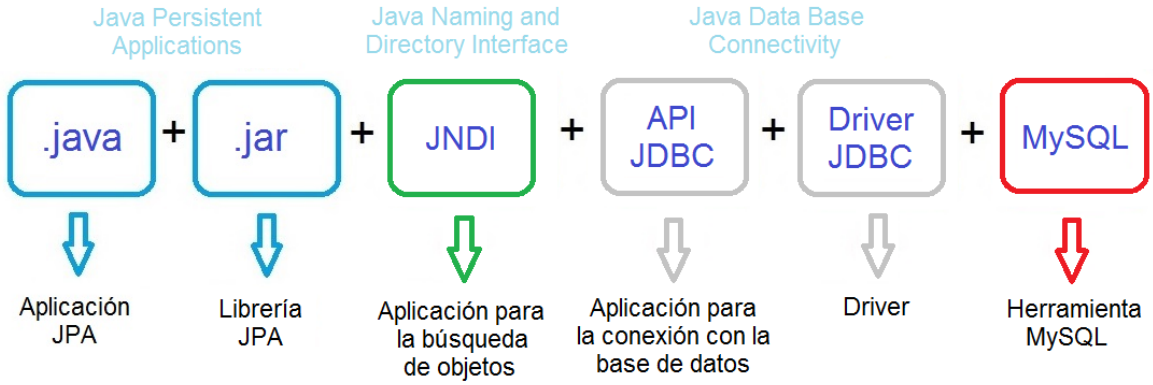


Figura 27. Implementación de la Base de Datos de RGAC.

4.6.4 Implementación de la Capa de Presentación

La capa de presentación de nuestra aplicación está formada por los componentes JavaServer Faces (JSF) utilizando las plantillas de Facelets y la librería de Primefaces, que nos permitió utilizar HTML dinámicamente, cuales mostraremos a continuación:

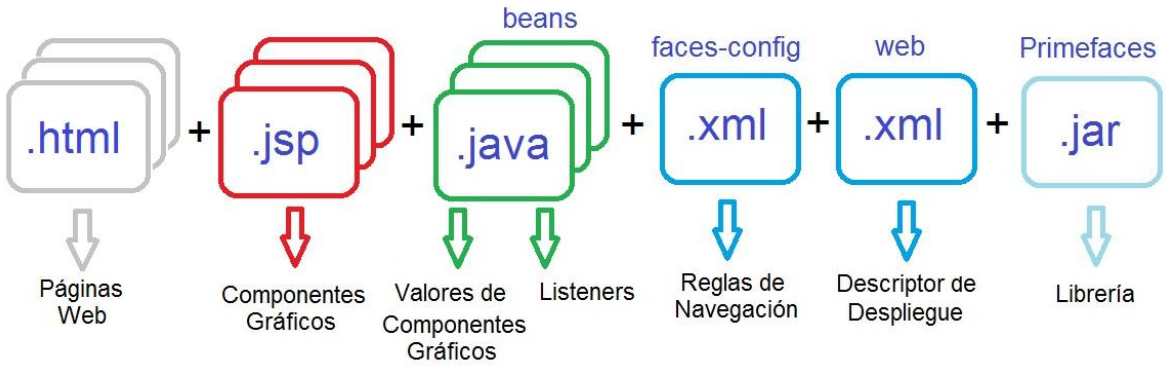


Figura 28. Capa de Presentación de RGAC.

4.6.5 Implementación de la Capa de Negocios

En la capa de negocios están los componentes Enterprise JavaBeans y el servidor Glassfish, cuales mostraremos a continuación:

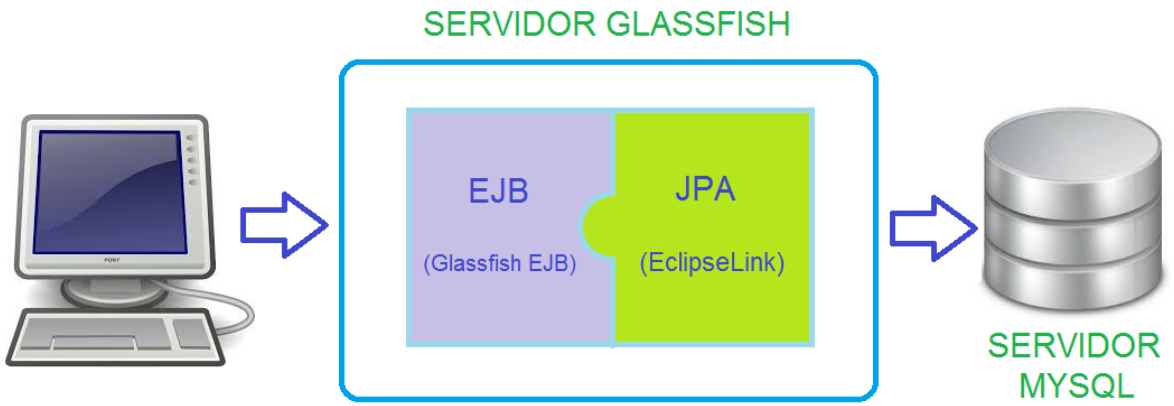


Figura 29. Servidor Glassfish del Sistema de RGAC.

4.6.6 Implementación de la Capa de Datos

La capa de acceso a datos contiene la lógica principal del almacenamiento de los datos y la persistencia de los datos de nuestra aplicación web. También realiza la gestión de la recuperación de los datos de forma compleja y la concurrencia de los diferentes usuarios para la disponibilidad de esta información.

En nuestra aplicación se realizó la gestión de datos utilizando los componentes JPA(Java Persistent Applications) y el driver JDBC cuales mostraremos a continuación:

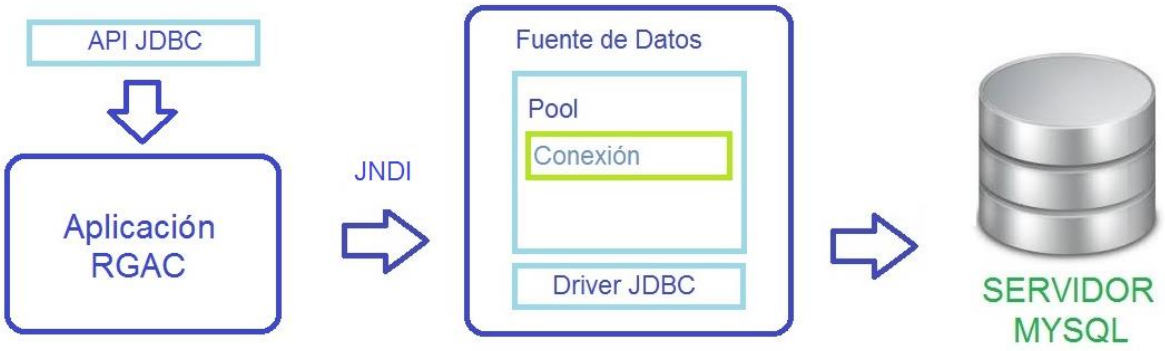


Figura 30. Capa de Datos del Sistema de RGAC.

5. Estudio de Caso: Aplicabilidad de RGAC en pequeñas entidades desarrolladoras de software

En este capítulo se reporta una aplicación empírica para evaluar RGAC. Para ello se diseñó y aplicó un estudio de caso en el contexto de una experiencia planeada en la que se hizo uso del repositorio para la construcción de un producto, a partir de una línea de productos. La idea es conducir un conjunto de proyectos pequeños desarrollando líneas de productos con Small SPL y RGAC.

5.1 Metodología

Según lo descrito por Runeson et al [92], el estudio de caso es una metodología de investigación adecuada para la ingeniería del software debido a que estudia un fenómeno contemporáneo en su contexto real, buscando mantener la integridad y las características significativas de los eventos, y es ejecutado cuando el investigador tiene poco control sobre los eventos y cuando los sujeto de estudio son más fáciles de observar en grupo que de manera aislada. Para la conducción de este caso de estudio ha sido necesario seguir los siguientes pasos:

- **Diseño**: El estudio de caso es establecido para comprobar si RGAC ayuda en el manejo de líneas de Producto software para modelar nuevos productos. En este objetivo se determinó el tipo de caso de estudio, su unidad de análisis, se seleccionó el caso, se diseñaron indicadores, métricas e instrumentos. Así mismo se determinaron los aspectos de recolección, análisis y reporte del caso de estudio.

- **Preparación**: Se han diseñado los instrumentos para desarrollar el caso: capacitación, documentación del proceso, protocolo de observación y los

instrumentos de recolección (Encuesta, Planilla de Tiempos, Planilla de Observaciones, preguntas abiertas).

- **Ejecución y Recolección de información**: En la Ejecución se realiza una sesión de n horas con n participantes. Primero se hace una introducción de las funciones realizadas por el RGAC y se realiza una pequeña demostración del prototipo. Luego se les entrega a los participantes una guía, la cual incluye un proceso de desarrollo. Después cada participante de forma independiente utiliza el sistema y se va tomando la información necesaria en las planillas para realizar el estudio. Al final se realizan las preguntas de investigación para que los participantes evalúen el sistema.

- **Análisis**: Se realizó un análisis de datos de tipo cuantitativo y cualitativo teniendo en cuenta la información obtenida de la observación de los sujetos investigados. Además, se obtuvo datos de la encuesta y las recomendaciones hechas por los sujetos investigados.

- **Reporte:** los resultados del estudio de caso fueron reportados y hacen parte de este documento.

5.2 Pregunta de Investigación

RGAC es un prototipo que se ha implementado como repositorio para el manejo activos de software en Small SPL basada en características de manera sistematizada con el fin de facilitar a la industria de software regional la aplicación de este proceso de desarrollo. Esto condujo a formular la siguiente pregunta: ¿RGAC facilita conceptualmente y tecnológicamente la ejecución de las actividades relacionadas con el almacenamiento, búsqueda y recuperación de activos dentro del proceso Small SPL a partir del modelo de características?

5.3 Objetivo del Estudio de Caso

El objetivo de este estudio de caso es evaluar la propuesta RGAC en un caso práctico analizando los resultados en términos de facilitación de las actividades de almacenamiento, búsqueda y recuperación de activos.

5.4 Selección del Estudio de Caso

Debido a que el modelado de la línea de productos es desarrollado por un equipo de desarrollo de software, son los equipos los indicados para poner en marcha la propuesta del RGAC y así, a través de un proyecto de línea de producto asignado, evaluar las facilidades del sistema en el almacenamiento, búsqueda y recuperación de activos. Así, la unidad de análisis es el proyecto de modelado de línea de producto utilizando el RGAC y

las fuentes de información primarias del mismo proyecto. De acuerdo a Yin[93], el caso de estudio es de tipo holístico, ya que se considerarán una unidad de análisis.

5.5 Contexto del Estudio de Caso

5.5.1 La Organización

En el marco del Semillero de Investigación en Ingeniería de Software se desarrolló la Actividad: “Diseño de Software para la Reutilización Siguiendo el Enfoque SPL”. La idea es desarrollar una línea de productos que permita manejar ejemplos de modelos de características. En el taller se les entrega tres documentos: el caso de negocio, los requisitos preliminares del cliente y el documento preliminar del alcance de la línea de productos. Para la obtención de estos documentos iniciales se desarrolla previamente una actividad de diseño y ejecución de un pequeño estudio de los ejemplos. El equipo está conformado por ingenieros de software con experiencia en el desarrollo de líneas de productos: ingeniero de dominio, administrador de activos e ingeniero de productos.

5.5.2 El Caso y sus Sujetos de Investigación

El estudio de caso es de tipo holístico, donde el objeto de investigación es el sistema RGAC en acción a través de su aplicación en una unidad de análisis que corresponde a un proyecto de desarrollo. Desde la perspectiva técnica los sujetos de investigación será el repositorio de la línea de productos y desde una perspectiva humana, los desarrolladores de software involucrados en el proyecto, quienes usarán el repositorio cuentan con experiencia en la construcción de líneas de productos.

5.6 Indicadores y Métricas

Para evaluar de manera objetiva este caso, particularmente, y para dar respuesta a la pregunta de investigación fue necesario definir un conjunto de métricas e indicadores. La tabla muestra un resumen de los indicadores y métricas identificados.

Preguntas de Investigación	Indicadores	Mediciones	Instrumento	Aplicado a
¿RGAC facilita conceptualmente y tecnológicamente la ejecución de las actividades relacionadas con el almacenamiento,	Esfuerzo de la recuperación de activos = Tiempo[en horas] de Recuperación * Número de Personas Involucradas en la	Tiempo de Recuperación de una activo.	Planilla	Tiempo
		Número de Personas Involucradas en la Recuperación de un activo.	Planilla	Tiempo

búsqueda y recuperación de activos dentro del proceso Small SPL a partir del modelo de características?	Recuperación[en Personas]			
	Esfuerzo del almacenamiento de activos = Tiempo[en horas] de Recuperación * Número de Personas Involucradas en la Recuperación[en Personas]	Tiempo Usado para la Organización y almacenamiento de un activo.	Planilla	Tiempo
		Número de Personas Involucradas en el Almacenamiento.	Planilla	Tiempo
	Esfuerzo del búsqueda de activos = Tiempo[en horas] de Recuperación * Número de Personas Involucradas en la Recuperación[en Personas]	Tiempo de Búsqueda	Encuesta	Tiempo
		Número de Personas Involucradas en la Búsqueda.	Encuesta	Tiempo
	Comprensión = Porcentaje de funcionalidades entendidas con éxitos respecto a todos los disponibles por la interface	Funcionalidades que los usuarios entienden con éxito.	Encuesta	Participantes del Proyecto
		Funcionalidades cuales los usuarios tienen disponibles en la interfaz.	Encuesta	Participantes del Proyecto
	Tiempo de Aprendizaje = Tiempo_Aprendizaje	Tiempo promedio que toman para aprender a utilizar una función correctamente.	Encuesta Planilla	Participantes del Proyecto Tiempo

A continuación, se describen en detalle los indicadores y la forma en que estos son calculados a través de las métricas identificadas:

- **Esfuerzo de la recuperación de activos:** El esfuerzo se define como la diversidad de elementos que componen una situación, los cuales se encuentran entrelazados y/o interconectados que contiene información adicional y oculta al observador. La fórmula que se ha definido para hacer el cálculo del esfuerzo de la recuperación de activos está dada por:

  Esfuerzo de la recuperación de activos = Tiempo [en horas] de Recuperación * Número de Personas Involucradas en la Recuperación [en Personas]

- **Esfuerzo del almacenamiento de activos:** El esfuerzo se define como el grado en que se producen los resultados esperados. La fórmula que se ha definido para hacer el cálculo del esfuerzo de almacenamiento de activos es:

  Esfuerzo del almacenamiento de activos = Tiempo [en horas] de Almacenamiento * Número de Personas Involucradas en el Almacenamiento [en Personas]

- **Esfuerzo de búsqueda de activos:** El esfuerzo se define como el grado en que se producen los resultados esperados. La fórmula que se ha definido para hacer el cálculo del esfuerzo de búsqueda de activos es:

  Esfuerzo de búsqueda de activos = Tiempo [en horas] de Búsqueda * Número de Personas Involucradas en la Búsqueda [en Personas]

- **Comprensión:** Se define como la comprensión de los usuarios de los datos de entrada y salida para RGAC. La fórmula que se definió es la división entre los datos de entrada y salida cuales los usuarios entienden con éxito y los datos de entrada y salida cuales los usuarios tienen disponibles en la interfaz. La fórmula que se ha definido para hacer el cálculo de la división es:

  Comprensión = DES_C / DES_D

  Donde:

  DES_C = DES_C - Datos de entrada y salida comprendidos

  DES_D = DES_D - Datos de entrada y salida disponibles

  El valor de la respuesta se interpreta como sigue:

  0 <= Comprensión <= 1 donde la más cercana a 1 es la mejor respuesta.

- **Tiempo Aprendizaje:** Se define cuanto tiempo el usuario aprende a utilizar una función de RGAC.

  El valor de la respuesta se interpreta como sigue:

  0 < Tiempo_Aprendizaje donde, entre más bajo sea el tiempo, es mejor.

5.7 Ejecución del Estudio de Caso, Resultados y Síntesis

Se realizó el estudio de caso como fue planeado, en el cual se realizó el uso de RGAC para la construcción de un producto a partir de una línea de productos.

Para esto los Magister Cecilia Camacho, Pablo Ruiz y el estudiante de tesis Edwar Giraldo organizaron una pequeña línea de productos, junto con la derivación de un producto utilizando RGAC y respondiendo a encuestas de esta aplicación. Es importante decir que para evaluar la herramienta se necesitó personas con experiencia en el desarrollo de líneas

de productos, no como expertos sino como gente con experiencia previa que pudiera evaluar objetivamente el propósito y utilidad de la herramienta. La ingeniera Cecilia Camacho tiene la experiencia en el desarrollo de líneas de productos para pequeñas organizaciones, Pablo Ruiz cuenta con la conceptualización y experiencia necesaria a través del modelado y generación del proceso unificado como una línea de producto y el estudiante Edwar Giraldo está haciendo su tesis en la recuperación de arquitecturas para lo cual debió recuperar y adaptar una arquitectura de un producto para una línea de productos en la empresa Nexura. Cada uno ejerce un rol de Ingeniero de Dominio (Cecilia Camacho), Administrador de Activos (Pablo Ruiz) e Ingeniero de Productos (Edwar Giraldo).

En el inicio de la sesión se realizó una presentación de la introducción de RGAC y los conceptos, porque los Ingenieros ya poseen buen conocimiento de las Líneas de Productos Software. Además se explicó cómo se realiza el manejo de RGAC, mostrando como están los elementos distribuidos en el prototipo nombrando las funciones que ejerce para la gestión de las líneas de productos, la gestión de las características y la gestión de los activos, así como también para la gestión de los productos que se crean. Luego se les presentó el ejemplo de Línea de Productos Software llamado SCL - Software Clock Lines para que luego realizaran su manejo en RGAC y crearan un producto software. Esto se realizó en un tiempo de 30 minutos, debido a que los integrantes ya contaban con el conocimiento base de construcción de líneas de productos.

El ejemplo SCL – Software Clock Lines es una línea de productos creada para soportar tareas pedagógicas y experimentales con líneas de productos. La línea de productos incluye 4 relojes. Este ejemplo ha sido desarrollado incrementalmente en el marco del curso de ingeniería de software II. Los artefactos iniciales se enfocaron a la construcción de un reloj que aplicaba un conjunto de patrones de diseño y que poco a poco fue dando lugar a una línea de productos. SCL es una línea de producto de juguete, pero tiene el objetivo de ser completa y comprensible.

El modelo de casos de uso de la línea de productos usado se presenta en la siguiente figura:

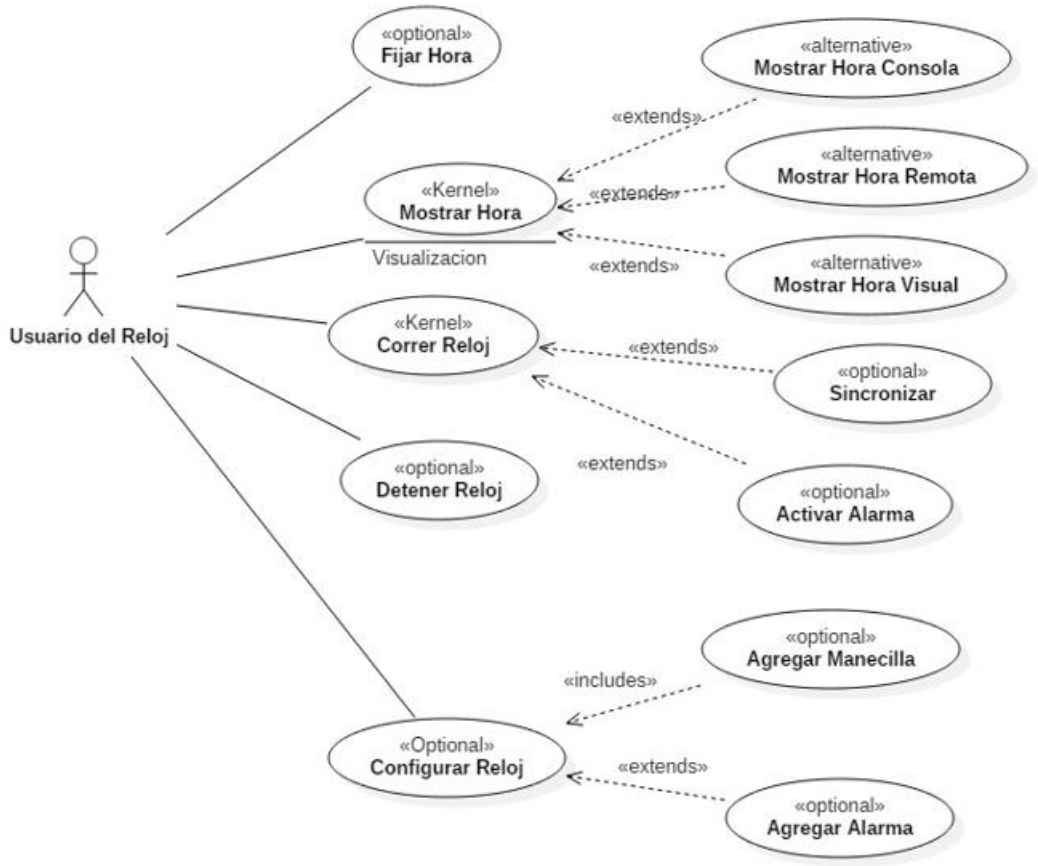


Figura 31. Casos de Uso SCL-Software Clock Lines.

Todo el proceso de desarrollo duró 3 horas, durante el cual los participantes recibieron un acompañamiento y se hicieron las anotaciones de las observaciones de acuerdo al protocolo definido. También en el acompañamiento se les resolvieron dudas que surgieron al momento de manejar el sistema.

Durante el desarrollo de la experiencia se fueron agregando los artefactos reutilizables de la línea de productos y luego derivando los artefactos haciendo uso de la herramienta y de acuerdo a las dinámicas de cada uno de los roles como lo determina el proceso SmallSPL. Para este caso se pidió elaborar uno de los productos no implementados y que correspondía a un reloj para ajedrecistas que contabilice el tiempo invertido por cada jugador, para que cuando un reloj se detenga, ponga en marcha el otro. Por ejemplo, para este caso el modelo de casos de uso derivado por el ingeniero de producto a partir del modelo de caso de uso de la línea de producto se presenta en la Figura 32. El resto de modelos y código del estudio de caso se encuentra en el Anexo 5.

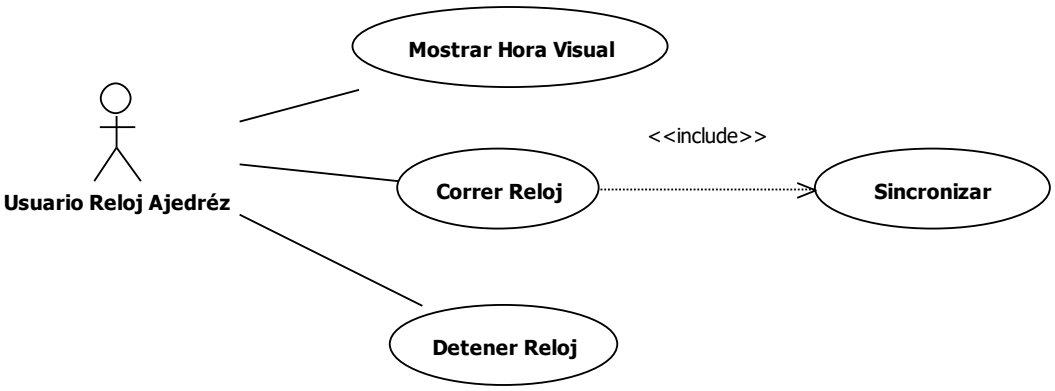


Figura 32. Casos de uso Reloj para juego de ajedrez.

La derivación y generación del producto siguió una estrategia manual, pero reutilizando los modelos y código, siguiendo la base de información que brindó RGAC, con lo que se hizo la experiencia desde los requerimientos del producto hasta su codificación ejecutable, por lo que tiene el suficiente alcance para que los ingenieros participantes pudieran hacer una evaluación evaluativa de la herramienta de gestión de activos evaluada en el marco de este estudio de caso.

En la siguiente figura se presenta como los participantes están trabajando en sus respectivos procesos en RGAC para crear la línea de productos y también al ir creando el nuevo producto software.

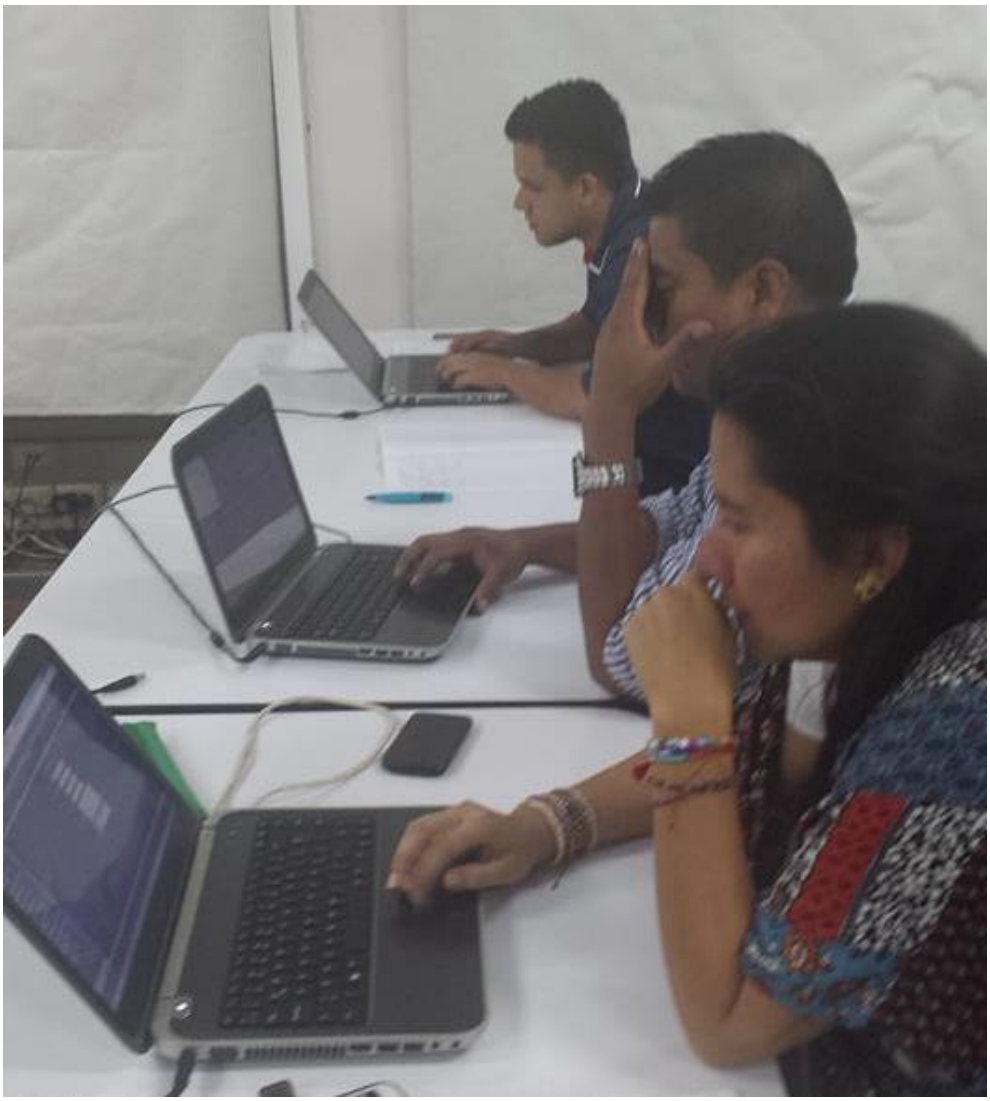


Figura 33. Análisis de la Aplicación RGAC.

Por último, en la siguiente figura se muestra cuando los participantes están solucionando las encuestas que se les entregó, con el fin de obtener la información cualitativa, cuantitativa y las observaciones del sistema.

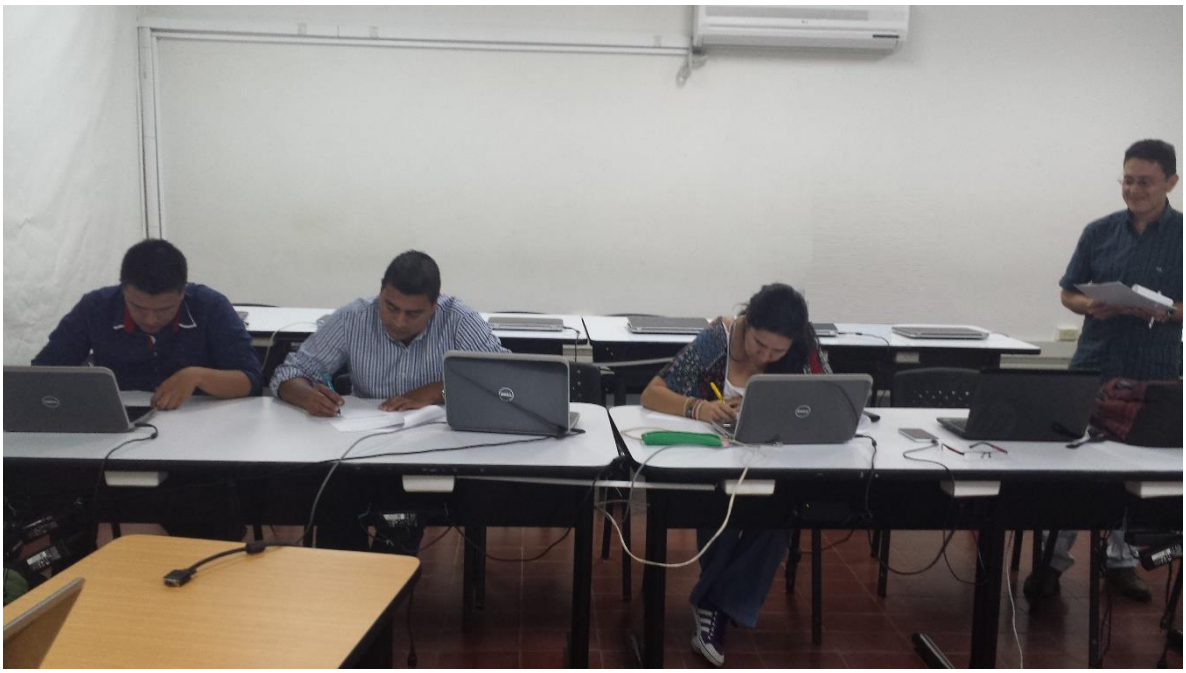


Figura 34. Realización de las encuestas de la Aplicación RGAC.

5.8 Resultados

5.8.1 Resultados Cuantitativos

A continuación mostramos los datos obtenidos en la realización del manejo de RGAC, donde los ingenieros recibieron la capacitación y realizaron el análisis del modelo de la línea de productos software para manejar en el sistema:

En la siguiente tabla se muestran los resultados de los indicadores que se utilizaron para en la evaluación del sistema el tiempo es contabilizado en minutos y el esfuerzo es contabilizado en minutos-persona:

SISTEMA RGAC	Ingeniero de Dominio	Administrador de Activos	Ingeniero de Productos	Promedio
Esfuerzo de Recuperación de activos	26	23	22	23.6
Esfuerzo de Almacenamiento de activos	25	28	21	24.6
Esfuerzo de Búsqueda de activos	34	35	32	33.6
Comprensión	0.9	0.9	0.9	0.9
Tiempo Aprendizaje	30	25	28	27.6

El promedio del indicador del esfuerzo de recuperación de activos nos muestra que se realizó de manera eficiente, demostrándonos que se encuentran bien entrelazados los activos. También el indicador del esfuerzo de almacenamiento de los activos tuvo rendimiento con un promedio de tiempo bajo, ya que obtuvieron los resultados esperados de manera relativamente rápida. En el tiempo obtenido del indicador de esfuerzo de búsqueda de activos fue mayor, esto es debido a que el grado en que se producen los resultados esperados en algunas tablas de la interfaz se demoró más tiempo, además el sistema no cuenta con un buscador. Además, el promedio de la comprensión al ser cercana a 1 nos muestra que en el manejo del sistema es sencillo entender los datos de entrada y salida. Asimismo el tiempo de aprendizaje como es razonable, nos muestra que el sistema es fácil de manejar.

En la siguiente tabla se presentan los resultados que se obtuvieron realizando encuestas cuantitativas con puntuación de 0 a 5 evaluando la funcionalidad (preguntas del 1 al 4), confiabilidad (preguntas del 5 al 8), usabilidad (preguntas del 9 al 14) y eficiencia (preguntas del 15 al 17) del sistema. Las preguntas realizadas en las encuestas se encuentran en el Anexo 4 Plantillas de Estudio de Caso. (Resultado = ∑puntos / #preguntas)

SISTEMA RGAC	Ingeniero de Dominio	Administrador de Activos	Ingeniero de Productos	Promedio
Funcionalidad	3.5	3.8	3.7	3.7
Confiabilidad	2.5	4.3	4.3	3.7
Usabilidad	3.8	3.2	5.0	4.0
Eficiencia	4.4	4.0	4.8	4.4
Totales	3.6	3.8	4.5	4.0

La funcionalidad del sistema posee un promedio intermedio, lo que nos indica que se necesitan adicionar opciones para mejorar su manejo y en la interfaz permitir visualizar el árbol de la línea de productos software desde todo el sistema, para una mejor comprensión. La confiabilidad en la gestión de activos y gestión de productos se obtuvo un promedio que nos declara ser satisfactoria. Pero para la gestión de líneas de productos, la gestión de características y la gestión de usuarios poseen una baja confiabilidad, ya que manifiestan necesario evitar la duplicidad y obtener una mayor seguridad. La usabilidad del sistema obtuvo en la gestión de productos un alto promedio, lo que nos indica ser bueno, pero en las demás gestiones es intermedio, lo que significa que la interfaz debe ser mejor organizada y también que se debe obtener la actualización de las tablas de una manera más rápida. El promedio de la eficiencia del sistema como es alto, nos manifiesta que la aplicación soportaría adecuadamente a una empresa de desarrollo gracias a las funciones de manejo de los activos.

5.8.2 Resultados Cualitativos

Apreciaciones de los Ingenieros de Procesos

En el transcurso de la experimentación los participantes manifestaron que RGAC posee un buen soporte al manejo de líneas de productos software así como también en la creación de un nuevo producto software, aunque si se modelara una línea de productos más

compleja, se necesitaría mejorar la búsqueda de información para una mejor gestión. También los participantes manifestaron que la capacidad en que se pudo manejar la base de datos es amplia y que aunque ya poseen conocimiento de estos temas, el aprendizaje del manejo de RGAC es sencillo. Además expresaron que la herramienta ayudaría a poseer una mejor eficiencia en la creación de los productos software. Sin embargo, los participantes manifestaron que la aplicación permite el acceso a todas las funcionalidades del sistema, lo que debería ser reservado según el tipo de usuario del sistema, así como también que se debería permitir la observación del árbol de la línea de productos desde otras interfaces, para una mejor comprensión. Además comentaron que la actualización de algunas tablas del sistema demoraba en realizarse y que el mecanismo de la búsqueda de los activos debería ser más avanzado para un mejor manejo. Asimismo comentaron que en algunas tablas de la interfaz no debería aparecer el identificador, sino el nombre.

Apreciaciones del Investigador durante el desarrollo del caso

Durante el desarrollo del estudio de caso se observó que los participantes tienen muy pocos problemas en la utilización de RGAC, ya que en la presentación se les explica cómo manejar las funciones. Además los participantes poseen conocimientos de las líneas de productos. Sin embargo durante la evaluación de RGAC se observó que los participantes de los procesos manifestaron demora en la actualización de algunas tablas del sistema, aunque luego observo los resultados esperados. También expresaron que para un mejor entendimiento deberían aparecer en las tablas el nombre de las líneas de producto y características en lugar de sus identificadores, ya que necesitaron ver en otra tabla su nombre. También opinaron que se debería permitir la observación del árbol de líneas de productos en las demás interfaces de la gestión para un más fácil manejo. Además manifestaron que se debe mejorar la presentación en la aplicación como por ejemplo en dividir el menú principal para mayor comprensión, debido a que se adicionaron funciones para la gestión de los tipos de características, los tipos de activos, los tipos de usuarios y los tipos de restricciones. También comentaron mejorar la presentación de los avisos de las notificaciones y un combo box más grande en la descripción de los elementos.

5.8 Análisis de Resultados

5.8.1 Análisis de Resultados Cuantitativos

Con los resultados obtenidos del tiempo se puede confirmar que con las funciones gestionadas por el Ingeniero de Producto son las más útiles y eficaces, ya que le facilitaron el manejo de la reutilización de los activos. El Administrador de Activos también con los resultados obtenidos manifiesta que las funciones de su gestión poseen ventajas con su rendimiento. Asimismo al Ingeniero de Dominio aunque le pareció un poco más complejo el manejo de las líneas de productos y características, debido a que no encontraba funciones de búsqueda más avanzadas, obtuvo los resultados en un tiempo razonable.

Esto nos quiere decir que RGAC posee una manera sencilla en su manejo, ya que el tiempo de aprendizaje fue bajo, así como también el tiempo para ejecutar las actividades en RGAC es poco.

5.9 Síntesis y Discusión

Los participantes que colaboraron en el estudio de caso manifestaron que RGAC ayuda en la reutilización de los activos al crear un nuevo producto software, así como en el manejo de una línea de productos software. Igualmente comentaron que esta aplicación es fácil conceptualmente y tecnológicamente en la ejecución de las actividades relacionadas con el almacenamiento, búsqueda y recuperación de los activos dentro del proceso Small SPL a partir del modelo de características.

Sin embargo, los participantes manifestaron que la aplicación debe ser mejorada sustancialmente antes de ser trasladada a la industria de software en aspectos funcionales y de usabilidad, aspectos que pueden ser considerados en la evolución de la herramienta en próximos proyectos de grado.

6 Conclusiones, Limitaciones y Trabajos Futuros

La reutilización de activos software puede ayudar en abundancia a las empresas desarrolladoras de software para la creación de sus productos, ya que agiliza y organiza su trabajo desde varios puntos de vista. La creación de la aplicación software RGAC para la reutilización de activos es una tarea necesaria para su gestión y rendimiento. La herramienta RGAC se basó en el Marco de proceso SmallSPL en el cual cada línea de producto posee características, las cuales tienen los activos reutilizables que serán adicionadas para la creación de nuevos productos.

6.1 Conclusiones

- En este trabajo de grado se ha creado un repositorio para la gestión de activos de software basada en características que nos ayuda en el uso de activos de software reutilizables. Este repositorio realiza principalmente la gestión de las líneas de productos de software, la gestión de las características, la gestión de los activos y la gestión de los productos en los cuales se realiza el manejo de los activos que serán reutilizados en varios productos de software. En cada gestión se realiza su creación, edición, muestra de los datos y eliminación de sus componentes.
- En esta aplicación también se tuvieron en cuenta la gestión de los tipos de características, de los tipos de activos y las restricciones de los activos que se

utilizarían para la creación de un nuevo producto. Para la elaboración de esta aplicación se realizó la investigación y el estudio de líneas de productos software, así como también para la creación del repositorio se realizó su estudio y desarrollo con herramientas y lenguajes de programación avanzadas.

- También la aplicación se creó con la ventaja de poder seguir siendo avanzada, ya que se realizó orientada a objetos con el patrón de arquitectura de software Modelo Vista Controlador para así poder adicionar funciones y continuar con su desarrollo.
- La investigación y la aplicación RGAC entonces nos ayuda a soportar conceptual y tecnológicamente las actividades de almacenamiento, búsqueda y recuperación de activos de software a partir del modelo de características, con el fin de hacer aplicables las prácticas de la gestión de activos en proyectos de software que siguen el proceso Small SPL.

6.2 Limitaciones

- Como la aplicación RGAC está basada en el proceso Small SPL el ingeniero que la utilice debe tener conocimiento acerca de éste.
- La herramienta RGAC posee un solo nivel en la clasificación de las características, lo cual hará que se tomen en forma más general, por lo que es recomendable trasladar el concepto de árbol de características para organizar la línea.
- También la aplicación cuando se realiza la función de adición de los activos a un producto, tener en cuenta que no se ejecuta de manera múltiple a la vez. Además el repositorio posee condiciones en la gestión de las restricciones que se les adicionan a los activos.
- La solución presenta algunas limitaciones funcionales en términos de actualización inmediata y de búsquedas, y varias limitaciones en términos de facilidad de uso que deben ser sometidas a una evaluación completa en términos de usabilidad.

6.3 Trabajos Futuros

En este trabajo de grado se han analizado algunos puntos que pueden ser tenidos en cuenta para trabajos futuros, cuales son los siguientes:

- Se espera como trabajo futuro la creación de avance en la aplicación RGAC, para que sea posible el manejo del sistema con más niveles de características.

- También se espera en el avance de la aplicación RGAC la selección de los activos a un producto software se pueda realizar de manera múltiple y que mejore el manejo de las restricciones entre los activos adicionados.
- Además se espera como trabajo futuro realizar un mejor estudio en las empresas de productos software para evidenciar las funciones que hacen falta para una mejor ayuda.
- Un trabajo concreto e interesante es realizar una evaluación formal de la usabilidad de la herramienta, con el fin de determinar aspectos ergonómicos y estéticos de la aplicación. La solución actual es un prototipo de ingeniería por lo que estos aspectos no fueron parte relevante de la propuesta.
- Se espera en un futuro desarrollar una siguiente versión del modelo que mejore aspectos funcionales y de usabilidad que la práctica detectó y que son relevantes para desarrollar cualquier experiencia con la industria.

7 Bibliografía

[1] Conradi, H. and A. Fuggetta, *Improving software process improvement.* Software, IEEE, 2002. **19**(4): p. 92-99.

[2] Hafedh Mili, Ali Mili, Sherif Yacoub, Edward Addy. Reuse Based Software Engineering: Techniques, Organizations, and Measurement.

[3] Clements, P., and Northrop, L. *Software Product Lines: Practices and Patterns.* Addison-Wesley Publishing Company, 2002.

[4] Krueger, C. W. “Easing the Transition to Software Mass Customization,” 282-293. Proceedings of the 4th International Workshop on Software Product Family Engineering. Bilbao, Spain, October 3-5, 2001. Berlin, Germany: Springer-Verlag, 2002.

[5] Camacho, Marta Cecilia; Hurtado Julio Ariel, "Analyzing the viability for adopting the software process line approach in small entities," *Computing Congress (CCC), 2012 7th Colombian* , vol., no., pp.1,6, 1-5 Oct. 2012

[6] Camacho, Marta Cecilia, SPL en las Pymes Desarrolladoras de Software del Cauca: Una experiencia desde Colmayor. Universidad del Cauca, Facultad de Ingeniería Electrónica y Telecomunicaciones, Grupo de Investigación en Ingeniería de Software IDIS, 2013.

[7] J.Sodhi, y P.Sodhi. Software reuse: Domain analysis and design process. McGraw-Hill.1999.

[8] Johannes Sametinger, Software Engineering with Reusable Components, March 3, 1997, Springer Verlag.

[9] Frakes, William; Terry, Carol. Software Reuse: Metrics and Models, ACM Computing Surveys, 1996.

[10] Anaya de Páez, Raquel. Un acercamiento a la reutilización en ingeniería de software. Revista Universidad EAFIT, [S.l.], v. 35, n. 114, p. 51-63, jul. 2012. ISSN 0120-341X.

[11] Arango, Guillermo. (1991). Domain Analysis - From Art Form to Engineering Discipline. En Domain Analysis and Software Systems Modeling. IEEE Computer Society Press.

[12] Prieto-Díaz, Rubén. (1993). Status Report: Software Reusability. IEEE Software. Mayo de 1993.

[13] Katz et al. 1994, Dabrowski, C., Miles, K., and Law, M. (1994). Glossary of software reuse terms. NIST Special Publication 500-222, National Institute of Standards and Technology, Gaithersburg, MD.

[14] Withey. 1996. Investment analysis of software assets for product lines. Technical Report CMU/SEI-96-TR-010 (ESC-TR-96-010), Software Engineering Institute. Carnegie Mellon University, Pittsburgh, PA 15213 (USA).

[15] Cybulski. 1996. Introduction to software reuse. Technical Report TR 96/4, Department of Information Systems. University of Melbourne.

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

[17] Pankaj Vohra, Ashima Singh. A Tool incorporating different Techniques for Effective Component Storage and Retrieval.

[18] Luqi and Jiang Guo, "Toward Automated Retrieval for a Software Component Repository", Proceedings of IEEE International Conference and Workshop on the Engineering of Computer Based Systems (IEEE ECBS), Nashville, USA, March 7-12, 1999. Pp. 99-105.

[19] Jiang Guo and Luqi, A Survey of Software Reuse Repositories.

[20] +1 Software Engineeering Corporate Mission, URL http://www.plus-one.com/company.html

[21] Elisabetta Morandin, "SALMS v5.1: A System for Classifying, Describing, and Querying about Reusable Software Assets", The Proceedings of 5th International Conference on Software Reuse (ICSR '98).

[22] Project Management Tool Suite System (Automated Software Reuse Repository), URL http://wv.ewa.com/srr_overview.html

[23] S Universal Repository, URL http://www.marketplace.unisys.com/urep/

[24] Reuse Library Toolset of EVB Software Engineering, http://gopher.metronet.com:70/0/newprod/byvendor/E/evb_software_e/941208.01

[25] J.-J. Jeng and B. H. C. Cheng, ``A formal approach to using more general components,'' in Proceedings of the 9th Knowledge-Based Software Engineering Conference, pp. 90-97, September 1994.

[26] Rubin Prieto-Diaz, "Implementing Faceted Classification for Software Reuse", Communication of the ACM, pp. 89-97, May 1991.

[27] DSRS - Defense Technology for Adaptable, Reliable Systems URL:http://ssed1.ims.disa.mil/srp/dsrspage.html

[28] STARS - Software Technology for Adaptable, Reliable Systems URL: http://www.stars.ballston.paramax.com/index.html

[29] D. Garlan, "Research Directions in Software Architecture", ACM Computing Surveys, 27(2), June 1995.

[30] R. Girardi, "Towards Effective Software Abstractions for Application Engineering", in Procs. NASA Focus on Reuse workshop, Sept. 1996.

[31] Asset Library Open Architecture Framework, Version 1.2; STARS-TC-04041/001/02; 14 August 1992.

[32] ELSA - Electronic Library Services & Applications URL: http://rbse.mountain.net/ELSA/

[33] R. Girardi, "Classification and Retrieval of Software through their Descriptions in Natural Language", Technical report, University of Geneva - CUI, December 1995.

[34] R. T. Price and R. Girardi. A class retrieval tool for an object-oriented environment. In Procs. 3rd Conf. Technology on object-Oriented Languages and Systems, pages 26-36, November 1990.

[35] ASSET - Asset Source for Software Engineering Technology URL: http://source.asset.com/asset.html

[36] PAL - Public Ada Library URL: http://web.cnam.fr/Languages/Ada/PAL/

[37] Luqi, and M. Ketabchi, “A Computer-Aided Prototyping System,” IEEE Transactions on Software Engineering, October 1988.

[38] Scott Dolgoff, “Automated Interface for Retrieving Reusable Software Components”, Master’s Thesis, Naval Postgraduate School, September 1992.

[39] Jiang Guo and Luqi, A Survey of Software Reuse Repositories.

[40] J. Penix, P. Baraona, and P. Alexander, ``Classification and retrieval of reusable components using semantic features,' in Proceedings of the 10th Knowledge-Based Software Engineering Conference, pp. 131-138, Nov. 1995.

[41] Software Product Lines: State of the art Patrick HEYMANS and Jean-Christophe TRIGAUX, FUNDP - Equipe LIEL, Institut d’Informatique Rue Grandgagnage, 15 September 2003.

[42] Northrop, L., Clements, P., Bachmann, F., Bergey, J., Chastek, G., Cohen, S., y otros. (2007). A Framework for Software Product Line Practice, Version 5.0. Recuperado el Febrero de 2010, de Software Engineering Institute (SEI): http://www.sei.cmu.edu/productlines/frame_report

[43] Krueger, Ch. Introduction to Software Product Lines. [on-line] www.softwareproductlines.com 2006

[44] Montilva2006 Jonás A. Montilva C., Ph.D. IEEE Member “Desarrollo de Software Basado en Líneas de Productos” Conferencia DVP IEEE Computer Society Región 9 -Capítulo Argentina URL http://www.ieee.org.ar/downloads/2006-montilva-productos-prt.pdf. 3 de Diciembre 2010.

[45] Generative Programming - Methods, Tools, and Applications by Krzysztof Czarnecki and Ulrich W. Eisenecker Addison-Wesley, June 2000.

[46] Kwanwoo Lee, Kyo Chul Kang, Jaejoon Lee: Concepts and Guidelines of Feature Modeling for Product Line Software Engineering. ICSR 2002: 62-77. 2002.

[47] Pamela Zave, FAQ Sheet on Feature Interactions, www.research.att.com/~pamela/faq.html Copyright AT&T, 1999, 2004.

[48] Kun Chen, Wei Zhang, Haiyan Zhao, Hong Mei. An Approach to Constructing Feature Models Based on Requirements Clustering 13th IEEE International Conference on Requirements Engineering (RE'05) pp. 31-40. 2005.

[49] Pohl, K.; Bockle, G,; Linden, F. Software Product Line Engineering: Foundations, Principles and Techniques. Springer, 2005.

[50] BigLever Software, Inc. Gears. Link actualizado a Junio 2010. http://www.biglever.com/index.html

[51] Gomaa, H. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison-Wesley 2004

[52] Metzger, A.; Pohl, K.; Heymans, P.; Schobbens, P.; Saval, G. Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis. Requirements Engineering Conference, 2007. RE'07. 15th IEEE International, pp. 243-253. 2007.

[53] Czarnecki, K.; Antkiewicz, M. Mapping Features to Models: A Template Approach Based on Superimposed Variants. GPCE 2005, pp. 422-437. 2005.

[54] Pohl, K.; Bockle, G; Linden, F. Software Product Line Engineering: Foundations, Principles and Techniques. Springer, 2005.

[55] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report, CMU/SEI-90-TR-21, Software Engineering Institute (Carnegie Mellon), Pittsburgh, PA 15213. 1990

[56] Kang, K. C., Kim, S., Lee, J. y Kim, K. FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Annals of Software Engineering, 5:143-168. 1998.

[57] Griss, M., Favaro, J., and d' Alessandro, M. Integrating feature modeling with the RSEB. In Proceedings of the Fifth International Conference on Software Reuse, pages 76--85. IEEE Computer Society Press, 1998.

[58] Generative Programming - Methods, Tools, and Applications by Krzysztof Czarnecki and Ulrich W. Eisenecker Addison-Wesley, June 2000

[59] Pohl, K.; Bockle, G,; Linden, F. Software Product Line Engineering: Foundations, Principles and Techniques. Springer, 2005.

[60] Eriksson, M.; Borstler, J.; Borg, K. The PLUSS Approach - Domain Modeling with Features, Use Cases and Use Case Realizations. Software Product Lines, 9th International Conference. pp. 33-44. SPLC 2005.

[61] BigLever Software, Inc. Gears. Link actualizado a Junio 2010. http://www.biglever.com/index.html

[62] Eriksson, M.; Borstler, J.; Borg, K. The PLUSS Approach - Domain Modeling with Features, Use Cases and Use Case Realizations. Software Product Lines, 9th International Conference. pp. 33-44. SPLC 2005.

[63] Pohl, K.; Bockle, G,; Linden, F. Software Product Line Engineering: Foundations, Principles and Techniques. Springer, 2005.

[64] Kotonya G. y Sommerville, Requirements Engineering Processes and Techniques. 1998, New York, NY: Wiley.

[65] IEE, Std, 610-1990, Glossary of Software Engineering Terminology. 1990, The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Stds. Collection.

[66] Greenfield, J. y Short, K., Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. 2004, Indianapolis: Wiley.

[67] Pohl, K., Böckle, G., y van der Linden, F., Software Product Line Engineering. Foundations, Principles and Techniques. 2005, Berlin Heidelberg: Springer.

[68] Kang, K., Cohen, S., Hess, J., Novak, W., y Peterson, A., Feature-Oriented Domain Analysis (FODA) Feasibility Study. 1990, SEI tech. report, CMU/SEI-90-TR-021. SEI (Software Engineering Inst.), Carnegie Mellon University: Pittsburgh, PA.

[69] John, I. y Muthig, D. Product Line Modeling with Generic Use Cases. Intl. Workshop on Requirements Engineering for Product Lines (REPL'02). 2002. Essen, Germany.

[70] Bayer, J., Gerard, S., Haugen, O., Mansell, J., Moller-Pedersen, B., Oldevik, J., Tessier, P., Thibault, J.-P., y Widen, T., *Consolidated Product Line Variability Modeling*. En *Software Product Lines. Research Issues in Engineering and Management*, Käkölä, T. y Dueñas, J.C., Editores. 2006, Springer: Berlin Heidelberg. p. 195-241.

[71] Bühne, S., Halmans, G., Lauenroth, K., y Pohl, K., Scenario-Based Application Requirements Engineering. En Software Product Lines: Research Issues in Engineering and Management, Käkölä, T. y Dueñas, J.C., Editores. 2006, Springer: Berlin Heidelberg. p. 161-194.

[72] Sitio web Apache UIMA (Unstructured Information Management Arquitecture) http://uima.apache.org/

[73] Sitio web SourceCodeOnline http://www.sourcecodeonline.com

[74] Sitio web Software Artefact Infraestructure Repository (SIR) http://sir.unl.edu/php/index.php

[75] Sitio web VCLComponents.com http://www.vclcomponents.com/

[76] Sitio web NetLib http://www.netlib.org/

[77] Sitio web Splot http://www.splot-research.org/

[78] Sitio web Fama FW http://www.isa.us.es/fama/

[79] Sitio web Big Lever http://www.biglever.com/

[80] Mærsk-Møller, H. M., & Jørgensen, B. N. (2010). Experiences initiating software product line engineering in small teams with PULSE. *In Proceedings of IASTED International Conference on Software Engineering*, *SE'07. ACTA Press 2010*, págs. 125--134. Innsbruck, Austria.

[81] Verlage, M., & Kiesgen, T. (2005). Five Years of Product Line Engineering in a Small Company. *ICSE '05 Proceedings of the 27th international conference on Software engineering*, (págs. 534-543).

[82] Sitio Web Wikipedia https://en.wikipedia.org/wiki/Feature_model

[83] Bosch, J. (2002). Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization. En L. N. Science (Ed.), *Proceedings of the Second Conference Software Product* (págs. 257-271). San Diego, USA: Springer.

[84] Jacobson, I., P. Jonsson, M. Christerson and G. Overgaard, Ingeniería de Software Orientada a Objetos - Un acercamiento a través de los casos de uso. Addison Wesley Longman, Upper Saddle River, N.J., 1992.

[85] Sitio Web Wikipedia https://es.wikipedia.org/wiki/JavaServer_Faces

[86] Sitio Web Wikipedia https://es.wikipedia.org/wiki/PrimeFaces

[87] Sitio Web Wikipedia https://es.wikipedia.org/wiki/Enterprise_JavaBeans

[88] Sitio Web Wikipedia https://es.wikipedia.org/wiki/GlassFish

[89] Sitio Web Wikipedia https://es.wikipedia.org/wiki/Java_Persistence_API

[90] Sitio Web Wikipedia https://es.wikipedia.org/wiki/Java_Database_Connectivity

[91] Sitio Web Wikipedia https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[92] Runeson & Martin Höst: Guidelines for conducting and reporting case study research in software engineering.

[93] R. K. Yin. *Case Study Research: Design and Methods*, 3rd edition. SAGE Publications, 2003.