Arquitectura del Módulo de Gestión de la Información de la Suite de Ingeniería de Software

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor:

Ana Evis Echemendía Díaz

Tutor:

Ing. Yusleydi Fernández del Monte

Ing. Lisandra Cala Hernández

Ing. Mairim Delgado Muñiz

La Habana, Cuba, 2013.

Para ser exitosos no tenemos que hacer cosas extraordinarias.

Hagamos cosas ordinarias, extraordinariamente bien.

Mahatma Gandhi

Declaración de Autoría

Declaro ser autora de la presente tesis y adjudico a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año 2013.

Ana Evis Echemendía Díaz

_____

Firma del Autor

Ing. Mairim Delgado Muñiz

_____

Firma del Tutor

Ing. Yusleydi Fernández del Monte

_____

Firma del Tutor

Dedicatoria

Le dedico el presente trabajo a mi familia, a mi mamá por estar siempre a mi lado, a mi papá porque a pesar de las distancias físicas nunca me ha faltado, a Osvaldito porque ha sido mi segundo papá incondicional, a mis hermanos con el gran deseo que como yo, algún día puedan lograr realizarse en lo que les gusta. A Luis Alberto por ser el impulsor de tantos sueños entre ellos este. Especialmente a mis abuelos, Benera y Aniceto, por haberme permitido ser su nieta preferida, a mi abuelo Rafael que la vida no le permitió estar conmigo en tantos momentos importantes para mí, pero ha sido mi meta nunca decepcionarlo donde quiera que esté y a mi abuela Evis que ha sido mi amiga mi espíritu y mi todo.

Agradecimientos

- A mis abuelos porque siempre han creído en mí,
- A mis padres Ana, Ariel y Osvaldito por estar siempre a mi lado,
- A mis hermanos porque nunca me han fallado,
- A mi familia en especial a mis tíos Minda y Rafelito, por ser tan incondicionales,
- A Yola, Alberto y Luis Alberto por darme tanto cariño por tanto tiempo,
- A Damaris y familia, por acogerme siempre,
- A mi amiga Claudia Espinoza, por ser mi hermana aunque pasen los años,
- A mis incondicionales amigos de tantos años Albert, Néstor y sus familias por quererme y ayudarme tanto,
- A mis padrinos y a Martha, porque siempre me han demostrado que me quieren y que puedo contar con ellos,
- A mis vecinos, por estar tan al pendiente de mi carrera,
- A Rafa y Vitorita, por quererme tanto, incluso sin conocerme,
- A las amistades durante estos años en la UCI: Indira, Mairim, Reidiel, Nana, Frank, Pablo, Sadiel, Hector,
- A Roberto, por ser "el mío" y creerme especial,
- A mi grupo del 1512 por ser el mejor grupo que he tenido en toda mi vida,
- A los muchachos y no tan muchachos del proyecto, por ser un equipo tan unido y acogedor, sobre todo a mis compañeros Yleana, Carlitos, Elizabet.
- A Naida , Yleana, Vivian y Keyla por brindarme su hospitalidad,
- A Miranda, porque además de compartir conmigo su sabiduría me demostró que es un amigo excepcional,
- A Rosalina, por no dejar de cuidarme y atenderme,
- A Lisandra Cala, por apoyarme y confiar en mí en momentos difíciles, por sus consejos y regaños que en estos momento extraño,
- A mis tutoras, por ser partícipe y guías en mi mayor resultado personal que este trabajo.

Resumen

La información es indispensable para la actividad humana, gestionarla correctamente aporta grandes beneficios independientemente del ámbito en que se realice. Durante el desarrollo de un software, se generan los productos de trabajo, los cuales son la evidencia de cómo se realizó este proceso. En la creación de los mismos, se presentan diversas dificultades que afectan el cronograma de desarrollo y la calidad que deben poseer. Para darle solución a dichos problemas se diseñó la arquitectura del Módulo de Gestión de la Información de la Suite de Ingeniería de Software que contribuye a la gestión de la información inherente al ciclo de desarrollo de software. En el diseño se aplicaron los estilos arquitectónicos Cliente/Servidor y 3-Capas y como patrones de diseño: Creador, Experto, Controlador, Alta cohesión y Bajo acoplamiento.

Durante el estudio se hizo uso de los métodos histórico-lógico y el analítico-sintético. La arquitectura fue evaluada utilizando el método ATAM *(Architecture Trade-off Análisis Method)*, validando que la selección realizada pueda manejar correctamente los riesgos que puedan aparecer en el proceso de desarrollo de esta propuesta.

Palabras claves:

Gestión de la Información, Metodología de Desarrollo de Software, Productos de Trabajo, Suite de Ingeniería de Software.

Índice

Índice de figuras

Índice de tablas

Introducción

La actividad humana involucra una gran cantidad de información independientemente de la esfera en que se desenvuelva. Esta información contribuye a la evolución del pensamiento y el quehacer científico, además a la adaptación y perfeccionamiento de las diversas formas de vida. De tal manera, el conjunto de experiencias que se van adquiriendo pueden ser aplicadas para proveer beneficios y ventajas en los diversos entornos donde el hombre se desarrolla y supera (1).

En ocasiones la gestión de dicha información es afectada por factores inherentes al tiempo de procesamiento, a la actualización y validez de los datos, así como a su disponibilidad y organización. Estos factores pueden ocasionar deficiencias en la ejecución de procesos de mayor envergadura asociados a empresas, instituciones o a las diversas estructuras de administración.

A partir del desarrollo de las Tecnologías de la Informática y las Comunicaciones (TIC), es posible concebir soluciones que reduzcan o erradiquen las insuficiencias en el manejo de la información facilitando el surgimiento de los Sistemas de Información (SI). Los mismos sirven de apoyo a la toma de decisiones, la coordinación y el control de los recursos, el análisis de problemas, la visualización de asuntos complejos y la creación de nuevos productos (2).

Los SI sustentan el desarrollo, coordinan las actividades de la cadena de valor empresarial y contribuyen al logro de los objetivos empresariales (3). Las empresas y organizaciones son las más beneficiadas al gestionar su propia información mediante la utilización de estas herramientas, ya que logran alcanzar mayor agilidad y rapidez, sin dejar a un lado la calidad de los procesos (4). Una eficiente gestión de la información contribuye por tanto, a la reducción del costo y el tiempo de desarrollo.

El proyecto Nova QALIT del Centro de Software Libre (CESOL), perteneciente a la Universidad de las Ciencias Informáticas (UCI), no se encuentra exento de una deficiente manipulación de la información durante el desarrollo de un sistema. Esta actividad está presente durante todo el ciclo de creación de un software, para la realización de la misma se utiliza el expediente de proyecto expedido por el Centro de Calidad de Software (Calisoft) que es la única entidad certificadora de este tipo en el país. En los proyectos de desarrollo de software, los equipos de trabajo tienen la responsabilidad de generar su expediente de

proyecto y dentro de este, todo un sinnúmero de documentos e informaciones que tributan al registro de los aspectos significativos del producto y a respaldar la liberación del mismo.

En dicho expediente están contenidas todas las plantillas de los artefactos que se van generando a medida que progresa la creación del nuevo software. Sin embargo, la generación de estos productos de trabajo suele ser un proceso muy tedioso que requiere una alta concentración para describir correctamente cada registro que se realice. Además es posible encontrar errores de omisión, ya que puede suceder que al analista se le olviden campos por llenar o no los complete con la calidad requerida.

Es necesario que el miembro del equipo de desarrollo encargado de realizar esta actividad conozca todos los aspectos relacionados con la misma. En muchas ocasiones esta importante particularidad no se toma en cuenta, y puede ser que la persona que la esté realizando, nunca antes lo haya hecho o no domine del todo la metodología utilizada y no comprenda cómo se realiza el llenado de las plantillas.

La distribución de los expedientes de proyecto se realiza por diferentes vías, la misma suele ser desde un dispositivo de almacenamiento (memorias flash, discos y otras unidades) hasta por correo electrónico. Durante este traslado resulta muy común que se pierda la documentación por algún descuido en el manejo de la tecnología y también pueden encontrarse duplicados innecesarios en la manipulación de estos documentos.

Al terminar cada artefacto, el mismo debe contar con la aprobación de los especialistas del equipo de trabajo, los cuales realizan minuciosamente su revisión. En el caso de que los registros no se encuentren correctamente realizados, se documentan los errores y se procede a realizar las modificaciones pertinentes. Por lo que vuelven a pasar por el mismo proceso de distribución, corriendo los mismos riesgos antes expuestos. El cometer fallas como las antes mencionadas trae como consecuencia falta de calidad en el expediente y atraso en el cronograma del proyecto, lo cual provoca incumplimiento en las tareas que dependen del mismo.

Teniendo en cuenta la **situación problémica** antes descrita**,** se plantea el siguiente **Problema científico:** ¿Cómo automatizar el proceso de gestión de la información en los productos de trabajo generados durante el desarrollo de software?

Definiéndose como **Objeto de estudio** el proceso de gestión de la información, donde el **campo de acción** estará enmarcado en el proceso de gestión de la información de los productos de trabajo generados durante el desarrollo de software.

De acuerdo al problema descrito, se ha planteado como **objetivo general:** Definir la arquitectura de un módulo que se integre a la Suite de Ingeniería de Software y que permita automatizar el proceso de gestión de la información en los productos de trabajo generados durante el desarrollo de software.

Por lo que para sustentar en mayor medida el propósito de esta investigación se plantea la siguiente **idea a defender:** La propuesta arquitectónica de un módulo que permita gestionar la información en el proceso de creación de una solución informática, puede garantizar la organización de los productos de trabajo generados durante el desarrollo de software.

Para poder dar cumplimiento al objetivo se han trazado los siguientes **objetivos específicos:**

- Realizar un estudio del estado del arte de las herramientas estilos y patrones arquitectónicos que contribuyan a la gestión de la información en el desarrollo de software.
- Definir la arquitectura del módulo de gestión de la información de los productos de trabajo generados durante el desarrollo de software.
- Evaluar la arquitectura definida.

Con el fin de desglosar los objetivos anteriormente expuestos, se proponen las siguientes **tareas de la investigación:**

1. Sistematización sobre las herramientas que faciliten la gestión de la información en el proceso de desarrollo de software para determinar elementos que sirvan de base en el módulo a desarrollar.
2. Sistematización sobre las herramientas, lenguajes, metodología y tecnologías a utilizar para el desarrollo del Módulo de Gestión de la Información.
3. Sistematización sobre los estilos arquitectónicos y patrones de diseño para conformar la arquitectura del Módulo de Gestión de la Información de la Suite de Ingeniería de Software.
4. Realización del diseño arquitectónico del Módulo de Gestión de la Información para una mejor comprensión de la arquitectura por parte de los desarrolladores.
5. Realización de la evaluación a la arquitectura propuesta para determinar los riesgos de la misma.

Para el desarrollo de la investigación a partir del cumplimiento de estas tareas, se emplean los siguientes métodos científicos:

Métodos teóricos:

- **Histórico-lógico:**

  Este método se utiliza para estudiar los sistemas relacionados con la gestión de la información que pueden servir como guía para el desarrollo del Módulo de Gestión de la Información. Además de permitir conocer las tendencias del comportamiento, desarrollo y estado actual de estas aplicaciones, aprendiendo de las experiencias y teorías que han surgido en el tiempo.

- **Analítico-Sintético:**

  El empleo de este método permite realizar un análisis de la bibliografía consultada referente a las tecnologías de los gestores de información para sintetizar los elementos relevantes para el desarrollo del módulo de gestión de la información.

El presente documento se encuentra estructurado de la siguiente forma:

Capítulo1: Proceso de gestión de la información durante el desarrollo de software.

Se presentan los resultados obtenidos de investigar sobre los sistemas de gestión de la información. De igual forma se estudian la metodología y el lenguaje a utilizar. Además se indaga sobre algunos aspectos arquitectónicos desde donde se fundamenta la selección de la arquitectura y se explica el modelo de vistas a seguir.

Capítulo2: Obtención de Requisitos y Diseño de la arquitectura del Módulo de Gestión de la Información.

Se exponen los actores que intervienen en el sistema, los requisitos funcionales y no funcionales recopilados luego de analizar las necesidades de la Suite de Ingeniería de Software. También se expone la prioridad de los requisitos funcionales y se describen los más importantes. Se aborda sobre la infraestructura tecnológica que se utiliza. Así mismo se muestra el diseño del Módulo de Gestión de la Información del proceso de desarrollo de software, a través de las vistas definidas y teniéndose en cuenta todos los aspectos analizados en el capítulo anterior.

Capítulo3: Evaluación de la arquitectura del Módulo de Gestión de la Información.

Se presentan los resultados obtenidos al evaluar la arquitectura definida para la realización del módulo de Gestión de la Información, permitiendo determinar si se cumple con los requisitos definidos e identificando los factores de riesgos y las acciones para mitigarlos.

Capítulo 1: Proceso de gestión de la información durante el desarrollo de software

Introducción:

En el presente capítulo se definen conceptos básicos que sirven de base para el entendimiento del presente trabajo de diploma. Además se exponen los resultados de la investigación realizada sobre los sistemas de gestión de contenidos empresariales. Conforme con el estudio se realiza una propuesta de los estilos arquitectónicos y patrones de diseño para la arquitectura del Módulo de Gestión de la Información de la Suite de Ingeniería de Software. También se aborda sobre la metodología de desarrollo de software a seguir, así como la infraestructura tecnológica, protocolos de comunicación y lenguaje de programación a utilizar.

1.1 Definición de gestión de la información del proceso de desarrollo de software

Durante la investigación se consultaron diversos conceptos sobre la gestión de la información que se adaptan al proceso de desarrollo de software. En el evento Habana INFO'99, Solórzano relaciona el concepto de gestión de la información con los sistemas de información, estableciendo que es el "conjunto de acciones que se proyectan y ejecutan – apoyadas en el sistema de información – para formalizar, estructurar e impulsar la aparición del recurso informático para mejorar la productividad y por lo tanto hacer competitivo un negocio o una empresa en un entorno cada vez más abundante en productos y servicios de información"(5).

En el 2012, María J. Vidal y Ana B. Araña exponen su concepción sobre gestión de la información, la cual se adapta de mejor manera a los objetivos de la presente investigación, pues manifiestan diversas acciones que se observan en el marco de la creación de un producto de trabajo, de tal manera definen que: *"La gestión de la información no es más que el proceso de organizar, evaluar, presentar, comparar los datos en un determinado contexto, controlando su calidad, de manera que esta sea veraz, oportuna, significativa, exacta y útil y que esta información esté disponible en el momento que se le necesite. Ella se encamina al manejo de la información, documentos, metodologías, informes, publicaciones, soportes y flujos en función de los objetivos estratégicos de una organización"* (6).

Se debe tener en cuenta durante el desarrollo de un software que la información es transformada en productos de trabajo. Algunas bibliografías lo manejan con el término “registros” y lo describen como la evidencia de que se han completado ciertas actividades durante el ciclo de desarrollo de software. Según la recomendación técnica ISO 90003:2004, apartado 4.2.4, *“los registros deben establecerse y mantenerse para proporcionar evidencia de la conformidad de los requisitos [...], deben permanecer legibles, fácilmente localizables y recuperables”*. También se establece la necesidad de disponer de *“un procedimiento documentado para definir los controles necesarios para la identificación, almacenamiento, protección, recuperación, tiempo de retención y disposición de estos registros”* (7). Además se pueden denominar como artefactos ya que los mismos son descritos como una pieza de información que es producida, modificada o usada por el proceso, definen un área de responsabilidad para un rol y están sujetos a control de versiones; pueden ser un modelo, un elemento de modelo o un documento (8).

La autora considera que los axiomas expuestos abarcan a modo muy general la gestión de la información y no se tiene en cuenta los tratamientos antes mencionados que se deben realizar con los artefactos. Por tal motivo, en el marco de esta investigación se define el proceso de gestión de la información durante el desarrollo de software tomando de apoyo el concepto de María J. Vidal y Ana B. Araña y enfocándolo a los procedimientos típicos que se debe cumplir al tramitar los productos de trabajo, quedando expresado de la siguiente manera:

El proceso de gestión de la información durante el desarrollo de software, es la obtención, modificación, almacenamiento y organización de toda la información generada durante todo el ciclo de vida de un software, la cual puede ser un modelo, un elemento de modelo, un documento o un *ticket* de la herramienta a desarrollar, así como cualquier otro informe que se realice durante su creación.

1.2 Sistemas de gestión de contenidos empresariales

Actualmente existen diversas herramientas encargadas de gestionar documentos y contenidos en diversas áreas de trabajo. Los sistemas de gestión de contenidos empresariales (ECM) son aplicaciones que apoyan a las empresas en el manejo y organización de sus informes. Son capaces de formular estadísticas a través de los datos extraídos de los mismos ayudando a tener una mejor visión en la toma de decisiones. Estas aplicaciones fueron estudiadas en busca de una solución factible que diera cumplimiento a las funcionalidades del Módulo de Gestión de la Información en la Suite de Ingeniería de Software. En caso de que los sistemas o las adaptaciones que se le puedan realizar, no sean suficientes para cumplir con los

requisitos, se utilizan de apoyo para aportar una visión acerca de la herramienta que resuelva los problemas identificados.

Entre los sistemas estudiados se seleccionaron: Nuxeo, Alfresco y Athento para exponer sus características, porque Nuxeo y Alfresco brindan prestaciones tecnológicas y funcionalidades avanzadas, estas herramientas se encuentran entre las más usadas actualmente. Permiten un alto grado de modularidad y rendimiento, además satisfacen la necesidad de las empresas en cuanto a la gestión de la información, documentos y contenidos (9). Mientras Athento es una potente plataforma de ECM, que proporciona una capa inteligente a los ECM que soporten el estándar *Content Management Interoperability Services* (CMIS) (10).

1.2.1 Nuxeo

Nuxeo es un ECM de fuente abierta, basado en Java. Permite administrar los documentos de manera colaborativa estableciendo versiones y ciclos. Dispone de una estructura orientada a servicios y su configuración suele ser fácil. Su interfaz es sencilla y accesible, las opciones están distribuidas en forma de pestañas.

La navegación y uso rutinario se facilita gracias a las tareas del plugin *Drag and Drop*, la opción de *Live Edit* y la posibilidad de previsualizar ficheros directamente con un servicio de OpenOffice. Además posee una navegación por las nubes de *tags* y un buscador simple y avanzado que facilita encontrar documentos. El resto de las opciones avanzadas que incluye la aplicación son más complicadas de utilizar y de entender.

Nuxeo permite trabajar los documentos directamente sobre la herramienta, mediante una serie de plugins. El anteriormente mencionado plugin *Live Edit* para OpenOffice/ Office y Mozilla Firefox/ Internet Explorer, permite crear documentos y salvarlos directamente en hojas de texto, presentaciones y hojas de cálculo.

Sin embargo, Nuxeo posee diversas insuficiencias en algunas de sus funcionalidades. En cuanto a la funcionalidad *Drag and Drop*, presenta problemas con la subida de un determinado número de archivos o con su tamaño. La utilización de un servidor de OpenOffice para configurarse con Nuxeo, es una tarea que requiere de conocimientos más profundos para lograrlo, esta configuración se ejecuta manualmente pues no tiene un proceso automático para realizarla. Otras carencias son la no incorporación de un sistema de copias de seguridad integrado (9).

1.2.2 Alfresco

Alfresco está basado en estándares abiertos y de escala empresarial. Incluye un repositorio de contenidos, un *framework* de portal *Web* para administrar y usar contenido estándar en portales. Posee una interfaz CIFS que provee compatibilidad de sistemas de archivos en Windows y sistemas operativos tipo Unix. Además tiene un sistema de administración de contenido web, virtualiza aplicaciones Web y sitios estáticos vía Apache Tomcat, sus búsquedas son realizadas a través del motor Lucene. Está desarrollado en Java y tiene un flujo de trabajo en jBPM.

Alfresco hace uso de secuencias de comandos (*scripts*) para simplificar la adición de nuevas funcionalidades y el desarrollo de nuevas interfaces de programación. Esta parte de la arquitectura se conoce como secuencia de comandos web y se utilizan los servicios de datos y presentación. Su arquitectura ligera, es fácil de descargar, instalar e implementar, es la típica arquitectura de una aplicación web basada en Java.

Alfresco se utiliza para implementar soluciones de ECM, como la gestión de documentos (MS), *Web Content Management* (WCM), la administración de registros (RM), y gestión de activos digitales (DAM). En el núcleo del sistema hay un repositorio que se apoya de un servidor, el mismo persiste los metadatos de contenidos, las asociaciones y los índices de texto completo; en este núcleo se encuentra la plataforma fuente que ofrece la posibilidad de modular las funcionalidades como el control de versiones, seguridad y reglas.

Además los usuarios acceden mediante un navegador a la interfaz de usuario Alfresco. Todos los documentos subidos a su repositorio, así como toda la información adicional (metadatos) son persistentes en la capa de almacenamiento, separando a nivel lógico el repositorio de las aplicaciones que acceden a él. Hay muchas maneras de modular e implantar Alfresco, pero la mayoría de las implantaciones siguen un patrón general (9).

1.2.3 Athento

Athento es un software que gestiona los documentos y contenidos empresariales. Está construido en una plataforma de código abierto. Es una herramienta web considerada eficiente, segura e inteligente. Es capaz de clasificar los documentos y almacenarlos en las carpetas correspondientes sin intervención humana al poder deducir el tipo del documento en dependencia de los tipos ya establecidos. Posee una arquitectura en capas desacoplando capas y servicios. Este sistema es extensible para ser adaptado fácilmente a distintos casos de uso.

La novedad se encuentra en que Athento toma de base cualquier gestor de soporte CMIS que le proveen los servicios básicos, sobre los cuales se implanta para hacer uso de las tecnologías semánticas y generar relaciones entre datos que permitan automatizar ciertas tareas dentro del contexto de la gestión documental. Además permite el almacenamiento masivo de contenidos, es compatible con Nuxeo 5.3.2 o superior y con Alfresco a partir del versión 3.2. Es capaz de crear y estructurar expedientes, posibilita adjuntar documentos a expedientes existentes y garantiza una gestión centralizada de documentos contenidos en diferentes repositorios. Se integra con sistemas como CMIS, OSGi, RSS, OWL, RDF, REST, JCR, EJB3, y de arquitectura SOA. Brinda soporte para la integración con bases de datos externas. Posee una interfaz adaptable a la imagen corporativa y permite el acceso del gestor mediante dispositivos móviles.

Esta herramienta se instala en plataformas independientes o cualquier sistema operativo que soporte Java, pero si se quiere lograr un mayor rendimiento, es recomendado el uso de GNU/Linux. Al trabajar con las bases de datos, se puede utilizar PostgreSQL Server 8.4 o superior, Oracle 10 y 11, Microsoft SQL Server 2005 o superior.

A continuación se muestra una tabla comparativa con los principales aspectos de las herramientas expuestas anteriormente.

Aspectos/Sistemas	Nuxeo	Alfresco	Athento
Tipo de aplicación	Web	Web	Web
Plataformas	Multiplataforma	Multiplataforma	Multiplataforma
Prestaciones del hardware	RAM: 2GB es la mínima cantidad de memoria para ejecutarse CPU: Intel core 2, equivalente o superiores	1 GB de RAM reservados para el proceso de la máquina virtual Java (JVM) 2 CPU por servidor (o 1 Dual-core)	Depende de los sistemas que integre
Trabajo sin conexión	No	No	No
Licencias	LGPL de código abierto	LGPL de código abierto	Licencia comercial

Tratamiento de la información	Una vez creado un documento, Nuxeo muestra el área de datos del documento, donde se pueden consultar, modificar y gestionar todas sus características.	Alfresco ofrece la habilidad de editar un documento directamente desde MS Office a través de un plugin y apoyan el protocolo SharePoint.	Depende de los sistemas que integre.

Tabla 1. Comparación entre ECM

Luego de estudiar las características de cada una de las herramientas y comparar con los aspectos que exige la Suite de Ingeniería de Software recogidos en la Planilla 0113_Especificación de requisitos de software, se evidencia que existen varios aspectos que no contribuyen a la solución del problema. Entre ellos se encuentra que estos sistemas requieren altas prestaciones de hardware, las cuales no se pueden cubrir con la tecnología de los usuarios finales. Asumiendo que lo antes mencionado no se tome en cuenta porque pueda ser solucionado de alguna manera, está la desventaja relacionada con que la herramienta que se necesita tiene que ser de escritorio, debido a que se requiere un software rápido, que sea capaz de poderse ejecutar aun cuando no sea posible acceder a Internet. Pero el aspecto fundamental por el cual no se pueden utilizar es por la forma en que se le da tratamiento a la información. Se puede generalizar en este aspecto que los ECM extraen la información de los documentos y archivos que se les incorpora, en cambio lo que se precisa en el Módulo de Gestión de la Información, es guardar datos o información para posteriormente transformar estos datos en informes y de esta manera conformar el expediente de proyecto (11).

Por los motivos antes expuestos, se decide proponer el desarrollo del Módulo de Gestión de la Información para la Suite de Ingeniería de Software. Este módulo debe permitir su integración a la Suite y poder realizar todas las operaciones necesarias para manejar la información de los productos de trabajo, del expediente de proyecto.

1.3 Lenguaje de programación

El equipo de trabajo determina como parte de las restricciones de diseño que se plantean en la planilla 0113_Especificación de requisitos de software, que la Suite de Ingeniería de Software fuera implementada en el lenguaje de programación Python. A continuación se abordan aspectos de este lenguaje que permiten argumentar el por qué de la selección realizada (11).

El lenguaje de programación Python fue creado por Guido Van Rossum a principios de los años 90. Su nombre está inspirado en el grupo de cómicos ingleses "*Monty Python*". Es un

lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Además este es un lenguaje interpretado o de script, con tipado dinámico, multiplataforma y orientado a objetos. En Python, como en Java y muchos otros lenguajes, la primera vez que se ejecuta, el código fuente se traduce a un pseudo código máquina intermedio llamado "bytecode", generando archivos .pyc o .pyo (bytecode optimizado), que son los que se ejecutan en sucesivas ocasiones (12).

Descripciones de las principales características:

- **Tipado dinámico:** está dado por la facilidad de declarar las variables sin forzar el tipo de dato que la misma va a contener, pues su tipo se define en tiempo de ejecución de acuerdo al valor que se le asigne.
- **Fuertemente tipado**: necesidad de tratar a una variable obligatoriamente por el tipo de dato que almacena, ya que es preciso convertir de forma explícita dicha variable al nuevo tipo que se le asigne.
- **Multiplataforma:** el intérprete de Python está disponible en multitud de plataformas (Solaris, GNU/Linux, DOS, Window, OS/2, Mac OS) por lo que si no se utilizan bibliotecas específicas de cada plataforma el programa puede correr en todos estos sistemas sin grandes cambios.
- **Orientado a Objeto:** la programación orientada a objetos constituye un paradigma en el que los conceptos relevantes del mundo real se trasladan a clases y objetos del programa. La ejecución del programa consiste en una serie de interacciones entre los objetos.

1.4 Aspectos arquitectónicos para el diseño del Módulo de Gestión de la Información

Se conoce como Arquitectura de Software (AS) a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para guiar al equipo durante el desarrollo de software dentro de un sistema informático. Según la IEEE Std 1471-2000 PP, se define como AS a la organización fundamental de un sistema formado por sus componentes, las relaciones entre ellos, el contexto en el que se implanta y los principios que orientan su diseño y evolución (13). David Garlan y Mary Shaw definen que la *"Arquitectura en un buen nivel de diseño hace foco en aspectos más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un tipo de problema"* (14).

1.4.1 Estilos arquitectónicos

Los estilos arquitectónicos o patrones se definen después de haberse realizado la ingeniería de requisitos. Luego de definir las características y restricciones del sistema que ha de ser construido, se define qué estilo o qué combinación de los mismos brindan una mejor solución (15). Después de esta breve acotación se exponen algunos patrones arquitectónicos estudiados para realizar el análisis de los mismos y poder elegir los que forman parte de la arquitectura del Módulo de Gestión de la Información.

- **Cliente/Servidor:** define una relación entre dos aplicaciones, como su nombre lo indica una es el "cliente", el cual envía peticiones a la otra capa, el "servidor", que no es más que la fuente de datos, que a su vez envía las respuestas al "cliente". Este patrón garantiza mayor seguridad en el manejo de los datos, ya que los mismos se almacenan en el servidor que generalmente ofrece más control sobre la seguridad. Al contener los datos centralizados en el servidor facilita el manejo y actualización de los mismos (16).
- **Basados en componentes:** describe un acercamiento al diseño de sistemas como un conjunto de componentes que exponen interfaces bien definidas y que colaboran entre sí para resolver el problema. Los componentes son diseñados de manera que puedan ser reutilizados en distintos escenarios, aunque algunos pueden realizar tareas específicas, se deben manejar de forma independiente para evitar afectar a otros componentes, además de que se deben operar en diferentes entornos y contextos. Toda la información debe ser pasada al componente o que este acceda a ella en lugar de incluirla a él. Las aplicaciones que utilizan este tipo de estilo, generalmente ejecutan procedimientos con pocos o ningún dato de entrada (16).
- **En Capas:** el patrón arquitectónico en Capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades que implementan. Entre sus características principales se puede mencionar que se descomponen los servicios de forma que la mayoría de las interacciones ocurren entre capas vecinas, estas capas pueden residir en la misma máquina o se pueden encontrar distribuidos entre varios equipos. La comunicación entre capas está basada en una abstracción que proporciona un bajo acoplamiento entre capas. Este estilo es usado cuando la aplicación es compleja y el alto nivel de diseño requiere la separación para que los distintos equipos puedan conectarse en distintas áreas de funcionalidades, además cuando se debe soportar distintos tipos de clientes y dispositivos (16).

- **Desacoplada:** el estilo de presentación desacoplado, indica cómo debe realizarse el manejo de las acciones del usuario, la manipulación de la interfaz y los datos de la aplicación, separando los componentes de la interfaz de flujo de datos y de la manipulación. Usa eventos para notificar a la vista cuando hay datos del modelo que han sido modificados, el controlador maneja los eventos disparados desde los controles de usuario en la vista. Este patrón es utilizado cuando se quiere separar la tarea de crear la interfaz de la lógica que la maneja, además cuando no se quiere que la interfaz contenga código de procesamiento de eventos (16).
- **Bus de servicios:** el presente estilo define un sistema de software que puede enviar y recibir mensajes usando uno o más canales de forma que las aplicaciones puedan interactuar sin conocer detalles específicos la una de la otra. Las comunicaciones entre aplicaciones suelen ser asíncronas, muchas de las implementaciones consisten en aplicaciones individuales que se comunican usando esquemas comunes y una infraestructura compartida para el envío y recepción de mensajes. Este patrón es utilizado cuando se tienen aplicaciones que interactúan unas con otras para realizar tareas o aplicaciones que realizan tareas separadas y se desea combinar esas tareas en una sola operación (16).
- **En la nube:** Esta arquitectura está orientada a trasladar una aplicación de nivel empresarial de un entorno físico a un entorno de nube virtualizado. La arquitectura de software es una vista del sistema que incluye los componentes principales del mismo, la conductora de esos componentes según se le percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. Es la vista conceptual de la estructura de un sistema informático (17).

1.4.2 Patrones de diseño

Los patrones de diseño se encargan de definir la estructura de un sistema, estos a su vez se componen de subsistemas con sus responsabilidades. También tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño (18).

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son además soluciones basadas en la experiencia y que se ha demostrado que funcionan (19).

No es obligatorio utilizar los patrones, solo es aconsejable en el caso de tener el mismo problema o similar al que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error.

GRASP

Al realizar el diseño de un sistema informático, la asignación de responsabilidades presenta gran variación. El tomar una decisión incorrecta da origen a que se cree un producto frágil, difícil de mantener, entender, reutilizar o extender. Para poder mitigar estos problemas, se pueden utilizar los patrones de asignación de responsabilidades GRASP. En la siguiente tabla se sintetizan los patrones asociados a este grupo.

Patrón	Descripción
Experto	¿Quién asume la responsabilidad en el caso general? Asignar una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con la responsabilidad.
Creador	¿Quién crea? Asignar a la clase B la responsabilidad de crear una instancia de la clase A, si se cumple una de las siguientes condiciones: B contiene A B agrega A B tiene los datos de inicialización de A B registra A B utiliza A muy de cerca.
Controlador	¿Quién administra un evento del sistema? Asignar la responsabilidad de administrar un mensaje de eventos del sistema a una clase que represente una de las siguientes opciones: El negocio o la organización global (un controlador de fachada) El sistema global (un controlador de fachada) Un ser animado del dominio que realice el trabajo (un controlador de papeles) Una clase artificial (fábrica pura) que represente el caso de uso (un controlador de caso de uso).
Bajo acoplamiento	¿Cómo dar soporte a poca dependencia y a una mayor reutilización? Asignar las responsabilidades de modo que se mantenga bajo acoplamiento.
Alta cohesión	¿Cómo mantener controlable la complejidad? Asignar las responsabilidades de modo que se mantenga una alta cohesión.
Polimorfismo	¿Quién, cuando el comportamiento varía según el tipo? Cuando varía el tipo (clase) de alternativas o comportamientos relacionados, asignar la responsabilidad del comportamiento, mediante operaciones polimórficas, a los tipos en que varía el comportamiento.
Fabricación pura	¿Quién, cuando está desesperado y no quiere violar los patrones Alta

	cohesión ni Bajo acoplamiento? Asignar un conjunto muy alto de cohesión de responsabilidades a una clase artificial que no represente nada en el dominio del problema, a fin de brindar soporte a una alta cohesión, a bajo acoplamiento y a la reutilización.
Indirección	¿Quién, para evitar el acoplamiento directo? Asignar la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios, de modo que no se acople directamente.
No hables con extraños (ley de Demeter)	¿Quién, para no conocer la estructura de los objetos indirectos? Asignar la responsabilidad al objeto directo del cliente para colaborar con el objeto indirecto, de modo que el cliente no necesita conocer el objeto indirecto.

Tabla 2. Patrones para generar responsabilidades GRASP (20)

1.4.3 Vistas arquitectónicas a utilizar

La arquitectura de un software se construye utilizando varias vistas arquitectónicas. Estas vistas dan el panorama de la arquitectura desde varios enfoques como puede ser del negocio, de la aplicación, de la información o de la tecnología.

En el presente trabajo se utiliza el modelo **"4+1 vista"** propuesto desde 1995 por Philippe Kruchten. Este modelo está vinculado a la metodología *Rational Unified Process* (RUP) en la siguiente tabla se muestra un resumen de cada vista.

Vista	Descripción
Vista Caso de Uso	Representa una síntesis del modelo de casos de usos del sistema. La vista de casos de uso es igual al modelo de casos de uso, la única diferencia es que la vista de la arquitectura solo tiene aquellos casos de usos y actores que son arquitectónicamente significativos mientras que el modelo de casos de uso tienen todos los casos de usos del sistema. Esta vista puede ser modelada estáticamente mediante diagramas de casos de usos (22).
Vista Lógica	Comprende las abstracciones fundamentales del sistema a partir del dominio de problemas. Representa los elementos de diseño más significativos para la arquitectura del sistema, describe las clases más importantes, su organización en paquetes y subsistemas, y estos a su vez en capas. También describe las realizaciones de casos de uso más importantes como por ejemplo los que describen aspectos dinámicos del sistema. Esta vista puede ser modelada estáticamente mediante diagramas de paquetes (22).
Vista de Implementación	Organización estática de módulos en el entorno de desarrollo. Representa la captura de las decisiones arquitectónicas tomadas para la implementación. Normalmente, esta vista contiene una enumeración de todos los subsistemas del modelo de implementación, diagramas de componentes que ilustran cómo se organizan los subsistemas en capas y

	jerarquías, ilustraciones de dependencias de importación entre subsistemas, puede ser modelado estáticamente a través del diagrama de componente (22).
Vista de Despliegue	Un mapeado del software sobre el hardware. Representa el diagrama de despliegue del cual el arquitecto es el responsable de considerar los aspectos referentes a la futura distribución física o despliegue del sistema. En esta vista deben ser detalladas las condiciones que debe cumplir el hardware, así como la configuración de red existente especificando los protocolos utilizados. Además se explica la necesidad de que esta configuración se realice correctamente (22).
Vista de Proceso	Conjunto de procesos de ejecución independiente, a partir de las abstracciones anteriores. Representa una base para la comprensión de la organización de los procesos de un sistema, mapeo de las clases y subsistemas en procesos e hilos. Solo suele usarse cuando el sistema presenta procesos concurrentes o hilos. Puede ser modelada estáticamente mediante diagramas de interacción (secuencia y colaboración) (22).

Tabla 3. Resumen del modelo 4+1 Vista (21)

El estudio realizado referente a la arquitectura de los sistemas informáticos, brinda información suficiente para, en conjunto con las características del módulo que se desea crear, elaborar la siguiente propuesta:

Los datos con los que trabaja el módulo están ubicados en un servidor, esto centraliza la información y permite el acceso desde varias estaciones de trabajo. Esta característica se cubre aplicando el estilo arquitectónico Cliente-Servidor. Para facilitar la reutilización de código, así como hacer un correcto uso de los principios de alta cohesión y bajo acoplamiento, se utiliza el estilo arquitectónico N-Capas, específicamente el patrón 3-Capas. Esto permite hacer una separación bien delimitada del código respecto a su funcionalidad u objetivo, dentro del módulo, lo que garantiza una mejor integración del mismo con la arquitectura general de la aplicación, que es basada en componentes. A continuación se muestra una imagen que ayuda a entender cómo queda estructurada la arquitectura del módulo.

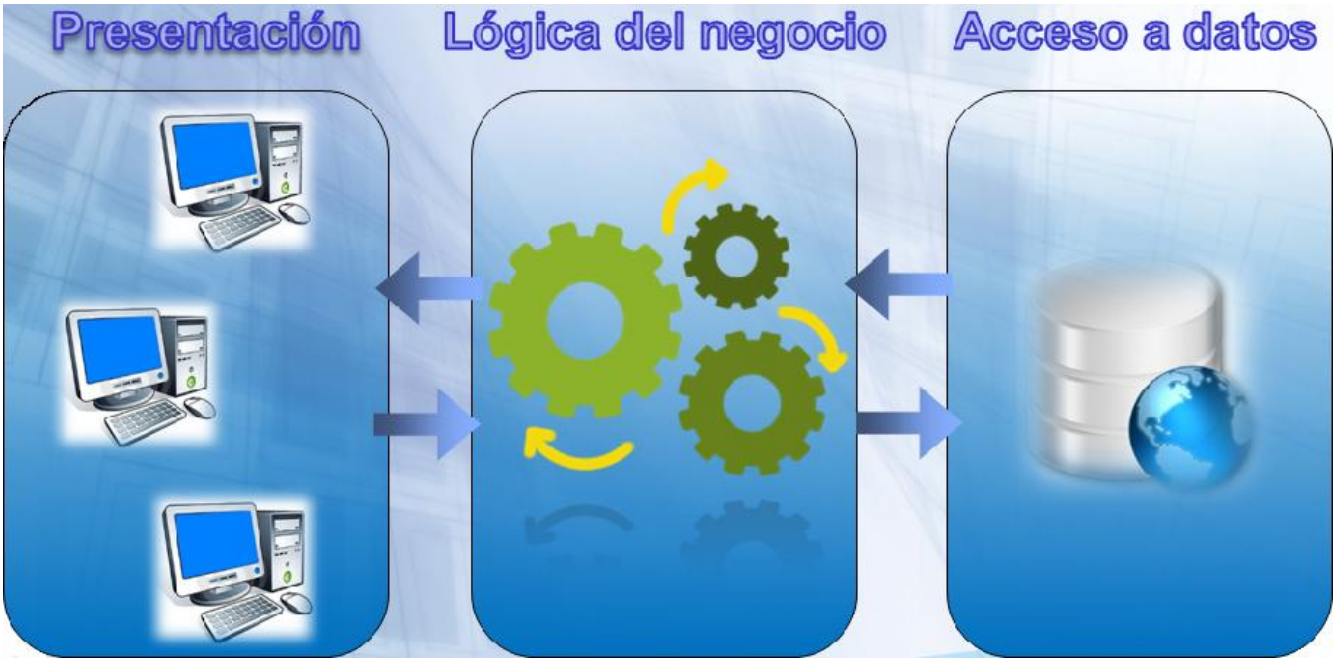

Imagen 1. Arquitectura del Módulo de Gestión de la Información

Como patrones de diseño se emplean:

- **Experto:** Es el principio básico de asignación de responsabilidades. Indica que, la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Con la utilización de este patrón, se define dónde colocar en cada clase las funcionalidades que necesitan de esa información, esa clase sería el experto en información.
- **Creador:** Ayuda a identificar quién debe ser el responsable de la instanciación de nuevos objetos o clases. Este patrón se utiliza para identificar qué clase A debe crear elementos de una clase B, apoyándose en que la clase A debería: contener, agregar, registrar, utilizar y tener los datos de inicialización de la clase B.
- **Alta Cohesión:** Indica que, la información almacenada en una clase debe de ser coherente y está, en la mayor medida de lo posible, relacionada con la clase. En este software es necesario controlar la complejidad de cada clase utilizada para mantener un buen comportamiento de las mismas; por esto, las que se identificaron con una gran cantidad de funcionalidades, se dividieron en otras de manera que se repartiera equitativamente el peso de la complejidad, manteniendo además, la coherencia entre ellas.
- **Bajo Acoplamiento:** Se utiliza con la idea de tener las clases lo menos ligadas entre sí posible, de forma que, en caso de que se produzca una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

- **Controlador**: Un defecto muy común cuando se diseña una clase controladora consiste en asignarle demasiadas responsabilidades. Este patrón delega a otros objetos el trabajo que ha de realizarse mientras coordina la actividad.

1.5 Protocolos de comunicación

Los protocolos de comunicación son reglas que permiten el flujo de información entre computadoras. Dentro de las distintas redes existen numerosos tipos de protocolos, entre ellos podemos mencionar:

- **TCP/IP:** Es definido como el conjunto de protocolos básicos para la comunicación de redes y es por medio de él que se logra la transmisión de información entre computadoras pertenecientes a una red. Gracias al protocolo TCP/IP, los distintos ordenadores de una red se logran comunicar con otros diferentes y así enlazar las redes físicamente independientes en la red virtual conocida bajo el nombre de Internet. Este protocolo es el que provee la base para los servicios más utilizados, como por ejemplo transferencia de ficheros, correo electrónico y acceso remoto.
- **TCP (*Transmision Control Protocol*):** Es un protocolo orientado a las comunicaciones y ofrece una transmisión de datos confiable. El TCP es el encargado del ensamblaje de datos provenientes de las capas superiores hacia paquetes estándares, asegurándose de que la transferencia de datos se realice correctamente.
- **HTTP (*Hypertext Transfer Protocol*):** Permite la recuperación de información y realizar búsquedas indexadas que permiten saltos intertextuales de manera eficiente. Por otro lado, permiten la transferencia de textos de los más variados formatos, no sólo HTML. El protocolo HTTP fue desarrollado para resolver los problemas surgidos en sistemas distribuidos en diversos puntos de la red.
- **FTP (*File Transfer Protocol*):** Es utilizado a la hora de realizar transferencias remotas de archivos. Permite enviar archivos digitales de un lugar local a otro que sea remoto o al revés. Generalmente, el lugar local es la PC mientras que el remoto el servidor.
- **SSH (*Secure Shell*):** Fue desarrollado con el fin de mejorar la seguridad en las comunicaciones de Internet. Para lograr esto el SSH elimina el envío de aquellas contraseñas que no son cifradas y codificando toda la información transferida.
- **UDP (*User Datagram Protocol*):** El protocolo de datagrama de usuario está destinado a aquellas comunicaciones que se realizan sin conexión y que no cuentan con mecanismos para transmitir datagramas. Esto se contrapone con el TCP que está destinado a comunicaciones con conexión. Este protocolo puede resultar poco

confiable excepto si las aplicaciones utilizadas cuentan con verificación de confiabilidad.

- **SNMP (*Simple Network Management Protocol*):** Este protocolo usa el protocolo (PDU) como mecanismo para el transporte. Por otro lado, utiliza distintos términos de TCP/IP como agentes y administradores en lugar de servidores y clientes. El administrador se comunica por medio de la red, mientras que el agente aporta la información sobre un determinado dispositivo.
- **TFTP (*Trivial File Transfer Protocol*):** Se caracteriza por sencillez y falta de complicaciones. No cuenta con seguridad alguna y también utiliza el Protocolo de Datagrama del Usuario como mecanismo de transporte.
- **SMTP (*Simple Mail Transfer Protocol*):** Está compuesto por una serie de reglas que rige la transferencia y el formato de datos en los envíos de correos electrónicos. SMTP suele ser muy utilizado por clientes locales de correo que necesiten recibir mensajes de correos electrónicos almacenados en un servidor cuya ubicación sea remota.
- **ARP (*Address Resolution Protocol*):** por medio de este protocolo se logran aquellas tareas que buscan asociar a un dispositivo IP, el cual está identificado con una dirección IP, con un dispositivo de red, que cuenta con una dirección de red física. ARP es muy usado para los dispositivos de redes locales *Ethernet*. Por otro lado, existe el protocolo RARP y este cumple la función opuesta a la recién mencionada (23).

Se decide utilizar el protocolo de comunicación TCP/IP porque el mismo consume pocos recursos. Tiene un grado muy elevado de fiabilidad, es adecuado para redes grandes y medianas, así como en redes empresariales. Este protocolo es compatible con las herramientas estándares para analizar el funcionamiento de la red.

1.6 Metodología de desarrollo de software

La metodología de desarrollo seleccionada para guiar el presente trabajo es OpenUp. La misma fue seleccionada por el equipo de trabajo del proyecto Nova QALIT. Esta metodología es extensible, está dirigida a la gestión y al desarrollo de proyectos de software basados en el desarrollo iterativo e incremental, por lo que es apropiado para proyectos pequeños y de bajos recursos. Es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo.

OpenUp es un marco del proceso del desarrollo del software *Open Source*. Al ser apropiado para proyectos pequeños y de bajo recursos, permite disminuir las probabilidades de fracasos e incrementa las de éxito. Permite detectar errores tempranos a través de un ciclo iterativo,

además evita la elaboración de documentos, diagramas e interacciones innecesarias de otras metodologías. Al ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas.

Fases del OpenUP

- **Concepción:** Primera de las 4 fases en el proyecto del ciclo de vida, trata acerca del entendimiento del propósito y los objetivos y obteniendo suficiente información para confirmar que el proyecto debe hacer. El objetivo de ésta fase es capturar las necesidades de los interesados en los objetivos del ciclo de vida para el proyecto.
- **Elaboración:** En esta fase es donde se tratan los riesgos significativos para la arquitectura. El propósito de esta fase es establecer la base la elaboración de la arquitectura del sistema.
- **Construcción:** Esta fase está enfocada al diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo. El propósito de esta fase es completar el desarrollo del sistema basado en la Arquitectura definida.
- **Transición:** Es la última fase, cuyo propósito es asegurar que el sistema es entregado a los usuarios, y evalúa la funcionalidad y rendimiento del último entregable de la fase de construcción (24).

1.7 Infraestructura Tecnológica

La infraestructura tecnológica es la base que define la vida de un sistema. La selección de la misma tiene una importancia estratégica ya que limita o potencia el crecimiento y el desarrollo de una organización. En ella se agrupan y organizan el conjunto de elementos tecnológicos que integran un proyecto, soportan las operaciones de una organización o sustentan una operación (25).

1.7.1 Herramientas para el desarrollo

A continuación se caracterizan algunas herramientas existentes para trabajar con el lenguaje de programación seleccionado (Python).

Librerías

Una librería es un conjunto de recursos (algoritmos) prefabricados, que pueden ser utilizados por el programador para realizar determinadas operaciones.

- **Python-relatorio**

Python-relatorio es una biblioteca de plantillas, la misma crea informes a partir de objetos de Python en diferentes formatos de documentos. Esta biblioteca es la que se propone utilizar porque además de generar reportes en PDF, los genera en ODT (26).

- **PyQt4**

QT es un paquete de desarrollo de interfaces gráficas, es de código abierto y multiplataforma, PyQt 4 es la implementación de esta librería en Python para QT 4, está doblemente licenciado bajo la licencia GPL (versión 2 y 3, con excepciones de las licencias adicionales) y una licencia comercial (27).

1.7.2 Herramientas CASE

Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Las mismas ayudan en todo los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. Las herramientas que se plantean a continuación son las que se definen para la creación de la Suite de Ingeniería de Software, por tal motivo son las que se utilizan en el Módulo de Gestión de la Información.

Eclipse

Es un entorno de desarrollo integrado, de código abierto y multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como aplicaciones de Clientes Enriquecidos. Está considerado como una potente y completa plataforma de programación, desarrollo y compilación. A la plataforma base se le pueden añadir extensiones (plugin) para aumentar sus funcionalidades (28).

Visual Paradigm

Visual Paradigm proporciona un conjunto de ayudas para el desarrollo de programas informáticos. Ha sido concebida para soportar el ciclo de vida completo del desarrollo de software a través de la representación de todo tipo de diagramas. Fue diseñada para usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. Es una herramienta privada disponible en varias ediciones, existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 Community Edition (29).

QtDesigner

QtDesigner es una plataforma cruzada GUI (interfaz gráfica de usuario) para el diseño y construcción de formas. Permite diseñar y construir rápidamente widgets y dialogar usando formularios en pantalla utilizando los mismos widgets que se utilizan en la aplicación. Los formularios creados con esta herramienta son totalmente funcionales y pueden ser vistos de antemano (30).

RapidSVN

RapidSVN es un cliente de interfaz gráfica para la comunicación con servidores Subversion. Está escrito en C++ y distribuido bajo licencia GPL. Este facilita el mantenimiento de las diferentes versiones de ficheros, desde una interfaz sencilla e intuitiva, y se encuentra disponible para plataformas Windows, Linux, MAC OS X y Solaris. Es una herramienta simple, ya que proporciona una interfaz fácil de usar para las funciones de Subversion (30).

1.8 Conclusiones parciales

A partir del estudio realizado en el capítulo, se puede concluir que los sistemas analizados no constituyen posibles soluciones al problema planteado, ya que es preciso que la solución deba permitir trabajar sin conexión y que en vez de manejar documentos, pueda gestionar datos que cuando el usuario lo desee pueda expórtalos en planillas con formatos pdf u odt. Por lo que se evidencia la necesidad de crear un Módulo de Gestión de la Información para la Suite de Ingeniería de Software inherente a esta investigación. En adición, se asumió como lenguaje de programación a Python, ya que es el que se emplea para la construcción de los restantes módulos de la Suite, así como la metodología de desarrollo de software OpenUp definida por el proyecto. De acuerdo a las especificidades del módulo en cuestión, se desarrolla la solución con la arquitectura Cliente-Servidor en tres capas, empleando patrones de software y el protocolo TCP/IP.

Capítulo 2: Obtención de Requisitos y Diseño de la arquitectura del módulo de Gestión de la Información.

Introducción:

En el presente capítulo se exponen los aspectos relacionados con el análisis y el diseño del Módulo de Gestión de la Información basado en la metodología seleccionada y atendiendo a los elementos que se hacen referencia en el capítulo anterior. Para lograrlo se define la arquitectura de dicha herramienta, así como el proceso de funcionamiento que permite dar cumplimiento a los requisitos funcionales y no funcionales del sistema. Además se representa la arquitectura del módulo utilizando el modelo 4+1 vista.

2.1 Actores del sistema

El actor del sistema es toda entidad externa a él con la que mantiene una relación y que le demanda una funcionalidad. El actor representa a un tipo de usuario que interactúa con el sistema mediante casos de uso. Otro tipo de actor son todos los sistemas externos que en algún momento interactúan con el sistema (31).

Nombre del actor	Descripción
Analista del sistema	Es la persona encargada de realizar toda la gestión de la información del proyecto.
Ingeniero de proceso	Además de poder gestionar la información del proyecto, será el encargado de crear la línea base del mismo.

Tabla 2. Actores del sistema

2.2 Requisitos del sistema

El análisis de requisitos permite al ingeniero de sistemas especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz con otros elementos del sistema y establece las restricciones que debe cumplir el software. Los requisitos del software son la base de las medidas de la calidad (15).

2.2.1 Requisitos funcionales

A continuación se presenta los requisitos funcionales agrupados en los casos de uso a los que pertenecen. Además su prioridad teniendo en cuenta la plantilla 0132_Evaluación de Requisitos, la cual se encuentra en el expediente de proyecto del programa de mejora del proyecto Nova QALIT.

Casos de uso	Requisitos	Prioridad
CU1 Gestionar metodología	RF_1 Adicionar Metodología de desarrollo de software	Alta
	RF_2 Modificar Metodología de desarrollo de software	Alta
	RF_3 Eliminar Metodología	Media
CU_2 Mostrar detalles de la Metodología	RF_4 Mostrar detalles de la Metodología	Media
CU_3 Seleccionar Metodología	RF_5 Seleccionar Metodología	Media
CU_4 Gestionar fase de la metodología	RF_ 6 Adicionar fase	Alta
	RF_7 Modificar fase	Media
	RF_8 Eliminar fase	Media
CU_5 Mostrar detalles de la fase	RF_9 Mostrar detalles de la fase	Media
CU_6 Seleccionar fase	RF_10 Seleccionar fase	Media
CU_7 Gestionar disciplina de la metodología	RF_11 Adicionar disciplinas de una metodología	Alta
	RF_12 Modificar disciplinas	Alta
	RF_13 Eliminar disciplinas	Media
CU_8 Mostrar detalles de las disciplinas	RF_14 Mostrar detalles de las disciplinas	Media
CU_9 Seleccionar disciplina	RF_15 Seleccionar disciplinas	Media
CU_10 Gestionar actividad	RF_16 Adicionar actividad	Media
	RF_17 Modificar actividad	Alta
	RF_18 Eliminar actividad	Media
CU_11 Mostrar detalles de la actividad	RF_19 Mostrar detalles de las actividades	Media
CU_12 Seleccionar actividad	RF_20 Seleccionar actividad	Media
CU_13 Gestionar artefactos	RF_21 Adicionar artefacto	Alta
	RF_22 Eliminar artefacto	Media
CU_14 Mostrar detalles de los artefactos	RF_23 Mostrar detalles de los artefacto	Media

CU_15 Seleccionar artefacto	RF_24 Seleccionar artefacto	Media
CU_16 Introducir información del artefacto	RF_25 Introducir información del artefacto.	Alta
CU_17 Generar reporte	RF_26 Generar reporte de artefacto en formato PDF	Baja
	RF_27 Generar reporte de artefacto en formato ODT	Baja
CU_18 Obtener línea base	RF_28 Obtener línea base	Baja

Tabla 3. Casos de Uso, requisitos y sus prioridades

2.2.2 Requisitos no funcionales

Seguidamente se exponen los requisitos no funcionales que se identificaron en la primera fase de la metodología.

Usabilidad

RNF_1. La aplicación está orientada a desarrolladores de software.

La aplicación va a ser utilizada por desarrolladores de software, los cuales dominan el proceso de ingeniería de software.

RNF_2. Debe presentar una interfaz ergonómica.

La aplicación debe ser fácil de usar brindándoles a los usuarios la posibilidad de ejercer sus actividades de una manera rápida y cómoda.

RNF_3. El módulo es una aplicación de escritorio.

Debido a que la Suite principal requiere ser un software que sea rápido, que permita realizar modelos con diferentes niveles de complejidad, el ingreso de grandes cantidades de datos, seguridad, y la continuidad del trabajo aun cuando Internet esté ausente.

RNF_4. La aplicación debe ser multiplataforma.

La aplicación es multiplataforma, por lo que debe funcionar en sistemas operativos Windows, Linux y Mac.

RNF_5. El sistema debe funcionar en hardware con bajas prestaciones.

El hardware con requerimientos mínimos sobre el que debe funcionar la Suite de IS se caracteriza por: En las PC clientes Micro Celeron 266, RAM 256MB.

Confiabilidad

RNF_6. El módulo debe ser capaz de reanudar las operaciones si el funcionamiento de las PC clientes se ve afectado por la ausencia del fluido eléctrico.

Si el funcionamiento del sistema se ve interrumpido por la ausencia del fluido eléctrico en el área donde están alojadas las PC clientes, una vez que vuelva este, se debe actualizar nuevamente el sistema con la última información guardada en el servidor de datos.

RNF_7. El módulo debe ser capaz de reanudar las operaciones si el funcionamiento de las PC clientes se ve afectado por la ausencia de conexión a través de la red.

Si el funcionamiento del sistema se ve interrumpido por la ausencia de conectividad al servidor desde las PC clientes, una vez que se establezca la conexión, se debe actualizar en el servidor la información que el usuario había guardado hasta el momento.

RNF_8. El módulo debe ser capaz de reanudar las operaciones si el funcionamiento del servidor de datos se ve afectado por la ausencia de conexión a través de la red.

Si el funcionamiento del sistema se ve interrumpido por la ausencia de conectividad en el servidor, una vez que se establezca la conexión, se debe enviar una notificación al usuario y permitir actualizar en el servidor la información que el usuario había guardado hasta el momento.

RNF_9. El módulo debe permitir al usuario trabajar desde su estación local

El módulo debe dar la opción al usuario de trabajar en una copia de trabajo local y actualizar los cambios cuando él lo requiera en el servidor de datos, evitando así la sobrecarga de operaciones en el servidor de datos.

RNF_10. El módulo debe permitir la impresión de los reportes generados.

Si se está trabajando con una impresora y esta se desconecta cuando se están haciendo operaciones con ella, el sistema debe mostrar un mensaje al usuario para indicándole el incidente.

RNF_11. Las excepciones del módulo deben ser notificadas al usuario final.

Cuando ocurre una excepción, el módulo debe mostrar un mensaje notificándola y mantenerse en el estado que estaba antes de esta ocurrir si la excepción es leve; si por el contrario es una excepción crítica, después de ser notificada el sistema debe cerrar.

RNF_12. La información debe estar disponible en todo momento para los usuarios finales.

La información guardada debe estar disponible siempre para los usuarios finales, permitiendo que estos puedan acceder a ella siempre que deseen, potenciando así la toma de decisiones.

Eficiencia

RNF_13. El módulo debe permitir que trabajen concurrentemente hasta 100 usuarios.

El módulo debe garantizar una concurrencia de 100 usuarios, es decir, el trabajo de equipos de desarrollo grandes y que pueden necesitar utilizar las mismas funcionalidades a la vez.

Soporte

RNF_14. El equipo de desarrollo debe brindar soporte por un período mínimo de un año.

El equipo de desarrollo debe brindar soporte al menos por un año y en los casos que lo requieran el tiempo de soporte debe ser mayor llegando este a un período máximo de dos años.

RNF_15. Si se encuentran no conformidades en el módulo, estas deben ser corregidas en un período máximo de un mes.

Restricciones de diseño

RNF_16. La aplicación debe hacerse utilizando el lenguaje de programación Python.

RNF_17. La aplicación debe hacerse utilizando sistema el gestor de base de datos PostgreSQL.

RNF_18. Para cumplir con los requisitos de diseño se va a utilizar la herramienta QTCreator.

RNF_19. Para la codificación de la herramienta se va a utilizar el IDE (Entorno Integrado de Desarrollo) Eclipse en su versión Galileo.

RNF_20. Para realizar el modelado se debe utilizar la herramienta Visual Paradigm.

Interfaces Software

RNF_21. La aplicación debe funcionar sobre los sistemas operativos Linux y Windows, recomendándose utilizar Ubuntu 10.4 o superior o Nova 2011 o superior o Windows NT o superior.

Interfaces de Comunicación

RNF_22. Se debe tener implementado en las estaciones de trabajo o servidores redes LAN (Cableadas o Inalámbricas).

Requisitos Legales, de Derecho de Autor y otros

RNF_23. Las herramientas seleccionadas para el desarrollo del producto están respaldadas por licencias libres, bajo las condiciones de software libre. En el caso de la herramienta Visual Paradigm como no es libre se utiliza la licencia que posee la universidad.

Estándares Aplicables

RNF_24. La documentación técnica generada del desarrollo de software debe quedar guardada en los artefactos que provee el expediente de proyecto propuesto por el programa de mejora emitido por Calisoft.

2.3 Descripción de los requisitos

A continuación se realiza la descripción de los requisitos arquitectónicamente significativos. Para ver las demás descripciones y los prototipos principales de interfaz gráfica, dirigirse al expediente de proyecto, plantilla 0129_Descripción de requisitos (32).

Gestionar metodología

Precondiciones	Poseer instalada la suite
Flujo de eventos	
Flujo básico <Gestionar metodología>	
1.	Acceder a la suite desde el menú Inicio de su pc.
2.	Acceder a la suite por la sesión de administrador o de analista.
3.	Acceder a la sección <Gestión de la información> del Menú desplegable <Módulos>.
4.	Acceder a la sección <Gestionar metodología> del módulo de <Gestión de la Información>.
5.	Seleccionar la sección que desee: <Adicionar metodología>, <Modificar metodología> o <Eliminar metodología>.
Pos-condiciones	
1.	Muestra la interfaz de la sección seleccionada.
Sección Adicionar metodología	
1.	Se muestra una ventana emergente en la que el usuario debe trabajar para completar la información necesaria antes de llegar al área de trabajo. En la ventana se muestran los datos que se necesitan para inicializar la acción de crear una metodología, los mismos son: El nombre de la metodología y los detalles de la misma, opcionalmente se puede adicionar un logo que caracterice la metodología.
2.	Completar los campos correctamente.
3.	Dar clic en el botón <Siguiente> que se encuentra en la parte inferior derecha de la ventana emergente.
Pos-condiciones	
1.	El usuario añadirá una metodología al sistema.
2.	Se muestra en la ventana emergente la vista para adicionar las fases y las disciplinas (Ver <Sección Adicionar fase> y <Sección Adicionar disciplina>).
3.	En la barra de herramienta situada a la izquierda aparece el proyecto inicializado guiado por la metodología creada representado por un árbol descendente donde se inicializa con el nombre del proyecto, seguido por el nombre de la metodología creada y de la misma se deriva <Fase>, <Disciplina>, <Actividad> y <Artefactos>, en un inicio con estos nombres hasta que no se adicionen los que exige la metodología que se está creando.
4.	En el área de trabajo situada a la derecha aparece la descripción de la metodología.
Flujos alternativos	

Flujo alternativo 1.A <Introducir datos incorrectamente>		
1.	Si se comete el error de introducir datos incorrectos o dejar vacíos los campos, el sistema lo informa y no permite dicha acción.	
Pos-condiciones		
1.	Se muestra un mensaje explicando el error.	
Validaciones		
1.	Nombre de la metodología.	
2.	Características de la metodología.	
Conceptos	**Nombre de la metodología**	Nombre de la metodología: Este campo es obligatorio y debe garantizar que solo se pueda escribir letras en el mismo. El componente a visualizar es un textEdit.
	Características de la metodología	Características de la metodología: Este campo es opcional y debe garantizar que solo se pueda escribir letras y números en el mismo. El componente a visualizar es un textEdit.
Sección Modificar metodología		
1.	Se muestra en el área de trabajo situada a la derecha de la aplicación los campos nombre de la metodología y detalles editables para realizar los cambios necesarios.	
2.	Realizar los cambios necesarios.	
3.	Dar clic en el botón <Guardar> situado a la derecha inferior del área de trabajo.	
Pos-condiciones		
1.	Se muestra en el área de trabajo situada a la derecha de la aplicación la vista con los detalles de la metodología actualizados.	
Flujos alternativos		
Flujo alternativo 1.A <Introducir datos incorrectamente>		
1.	Si se comete el error de introducir datos incorrectos o dejar vacío los campos, el sistema lo informa y no permite dicha acción.	
Pos-condiciones		
1.	Muestra un mensaje explicando el error.	
Validaciones		
1.	Nombre de la metodología.	
2.	Características de la metodología.	
Conceptos	**Nombre de la metodología**	Nombre de la metodología: Este campo es obligatorio y debe garantizar que solo se pueda escribir letras en el mismo. El componente a visualizar es un textEdit.
	Características de la metodología	Características de la metodología: Este campo es opcional y debe garantizar que solo se pueda escribir letras y números en el mismo. El componente a visualizar es un textEdit.
Sección Eliminar metodología		
1.	Se muestra una ventana emergente con el listado de las metodologías existentes en el sistema.	
2.	Marcar la/las metodologías que se desea eliminar.	
3.	Dar clic en el botón <Eliminar>, situado a la derecha inferior de la ventana emergente.	
Pos-condiciones		
1.	Se elimina del sistema las metodologías que se seleccionaron.	
Validaciones		
1.	No Procede	

Tabla 4. Gestionar metodología

Gestionar disciplina

Precondiciones	Poseer instalada la suite	
Flujo de eventos		
Flujobásico<Gestionardisciplina>		
1.	Acceder a la suite desde el menú Inicio de su pc.	
2.	Acceder a la suite por la sesión de administrador o de analista.	
3.	Acceder a la sección <Gestión de la información> del Menú desplegable <Módulos>.	
4.	Acceder a la sección <Gestionar disciplina> del módulo de <Gestión de la Información>.	
5.	Seleccionar la sección que desee: <Adicionar disciplina>, <Modificar disciplina> o <Eliminar disciplina>.	
6.	Aparece una ventana emergente para seleccionar la metodología a la cual se le desea gestionar la disciplina.	
7.	Seleccionar la metodología y dar clic en el botón <Seleccionar>.	
Pos-condiciones		
1.	Muestra la interfaz de la sección seleccionada.	
Sección Adicionar disciplina		
1.	Se muestra en la ventana emergente la vista para adicionar la disciplina en el área derecha.	
2.	Seleccionar la sección <Adicionar> perteneciente al área de las disciplinas.	
3.	Aparece en la ventana emergente los datos que se necesitan para inicializar la acción de adicionar una disciplina, los mismos son: nombre de la disciplina y detalles de la misma.	
4.	Completar los campos correctamente.	
5.	Seleccionar la sección <Aceptar> en el extremo inferior derecho de la ventana emergente.	
6.	Se muestra en la ventana emergente la vista para adicionar las fases y las disciplinas.	
7.	Seleccionar la sección <Siguiente> en el extremo inferior derecho de la ventana emergente.	
Pos-condiciones		
1.	El usuario adiciona una disciplina al sistema.	
2.	Se muestra en la ventana emergente la vista para adicionar las actividades (Ver <Sección Adicionar actividad>).	
Flujos alternativos		
Flujo alternativo 1.A <Introducir datos incorrectamente>		
1.	Si se comete el error de introducir datos incorrectos o dejar vacío los campos, el sistema lo informa y no permite dicha acción.	
Pos-condiciones		
1.	Muestra un mensaje explicando el error.	
Validaciones		
1.	Nombre de la disciplina.	
2.	Características de la disciplina.	
Conceptos	**Nombre de la disciplina**	Nombre de la disciplina: Este campo es obligatorio y debe garantizar que solo se pueda escribir letras en el mismo. El componente a visualizares un textEdit.

	Características de la disciplina	Características de la disciplina: Este campo es opcional y debe garantizar que solo se pueda escribir letras y números en el mismo. El componente a visualizares un textEdit.
Sección Modificar disciplina		
1.	Se muestra en la ventana emergente la vista para modificar las disciplinas en el área derecha.	
2.	Seleccionar la sección <Modificar> perteneciente al área de las disciplinas.	
3.	Aparece en la ventana emergente los datos que se necesitan para inicializar la acción de modificar una disciplina, los mismos son: El nombre de la disciplina y los detalles de la misma.	
4.	Modificar los campos correctamente.	
5.	Seleccionar la sección <Guardar> en el extremo inferior derecho de la ventana emergente.	
Pos-condiciones		
1.	Se muestra en la ventana emergente la vista para modificar las fases y las disciplinas.	
Flujos alternativos		
Flujo alternativo 1.A <Introducir datos incorrectamente>		
1.	Si se comete el error de introducir datos incorrectos o dejar vacío los campos, el sistema lo informa y no permite dicha acción.	
Pos-condiciones		
1.	Muestra un mensaje explicando el error.	
Validaciones		
1.	Nombre de la disciplina.	
2.	Características de la disciplina.	
Conceptos	**Nombre de la disciplina**	Nombre de la disciplina: Este campo es obligatorio y debe garantizar que solo se pueda escribir letras en el mismo. El componente a visualizares un textEdit.
	Características de la disciplina	Características de la disciplina: Este campo es opcional y debe garantizar que solo se pueda escribir letras y números en el mismo. El componente a visualizares un textEdit.
Sección Eliminar disciplina		
1.	Se muestra en el área de trabajo las disciplinas pertenecientes a la metodología seleccionada.	
2.	Seleccionar las disciplinas que se desean eliminar y dar clic en la sección <Eliminar> en el extremo inferior derecho del área de trabajo.	
Pos-condiciones		
1.	Se actualiza la vista sin las disciplinas eliminadas.	
Validaciones		
1.	No Procede	

Tabla 5. Gestionar disciplina

Gestionar actividad

Precondiciones	Poseer instalada la suite
Flujo de eventos	
Flujo básico <Gestionar actividad>	

1.	Acceder a la suite desde el menú Inicio de su pc.	
2.	Acceder a la suite por la sesión de administrador o de analista.	
3.	Acceder a la sección <Gestión de la información> del Menú desplegable <Módulos>.	
4.	Acceder a la sección <Gestionar actividad> del módulo de <Gestión de la Información>.	
5.	Seleccionar la sección que desee: <Adicionar actividad>, <Modificar actividad> o <Eliminar actividad>.	
6.	Aparece una ventana emergente para seleccionar la metodología a la cual se le desea gestionar la actividad.	
7.	Seleccionar la metodología y dar clic en el botón <Seleccionar>.	
Pos-condiciones		
1.	Muestra la interfaz de la sección seleccionada.	
Sección Adicionar actividad		
1.	Se muestra en la ventana emergente la vista para adicionar las actividades.	
2.	Dar clic en el botón <Adicionar> aparece una ventana emergente donde se debe completar el nombre de la actividad, sus características principales y las fase y disciplinas de la metodología para escoger a cuales pertenece dicha actividad.	
3.	Completar correctamente los datos.	
4.	Seleccionar la sección <Guardar> en el extremo inferior derecho de la ventana emergente.	
5.	Se regresa a la vista de adicionar las actividades.	
6.	Seleccionar la sección <Siguiente> en el extremo inferior derecho de la ventana emergente.	
Pos-condiciones		
1.	El usuario adiciona una actividad al sistema.	
2.	Se muestra en la ventana emergente la vista para adicionar los artefactos (Ver <Sección Adicionar artefacto>).	
Flujos alternativos		
Flujo alternativo 1.A <Introducir datos incorrectamente>		
1.	Si se comete el error de introducir datos incorrectos o dejar vacío los campos, el sistema lo informa y no permite dicha acción.	
Pos-condiciones		
1.	Muestra un mensaje explicando el error.	
Validaciones		
1.	Nombre de la actividad.	
2.	Características de la actividad.	
3.	Fase	
4.	Disciplina	
Conceptos	**Nombre de la actividad**	Nombre de la actividad: Este campo es obligatorio y debe garantizar que solo se pueda escribir letras en el mismo. El componente a visualizar es un textEdit.
	Características de la actividad	Características de la actividad: Este campo es opcional y debe garantizar que solo se pueda escribir letras y números en el mismo. El componente a visualizar es un textEdit.
	Fase	Fase: Este campo es obligatorio y debe garantizar al menos que se marque una fase. El componente a visualizar es un RadioButton.

	Disciplina	Disciplina: Este campo es obligatorio y debe garantizar al menos que se marque una disciplina. El componente a visualizar es un RadioButton.
Sección Modificar actividad		
1.	Se muestra en la ventana emergente la vista para modificar las actividades.	
2.	Dar clic en el botón <Modificar> aparece una ventana emergente donde se puede corregir el nombre de la actividad, sus características principales y las fase y disciplinas de la metodología si es necesario.	
3.	Completar correctamente los datos.	
4.	Seleccionar la sección <Guardar> en el extremo inferior derecho de la ventana emergente.	
Pos-condiciones		
1.	Se muestra en la ventana emergente la vista para modificar las actividades.	
Flujos alternativos		
Flujo alternativo 1.A <Introducir datos incorrectamente>		
1.	Si se comete el error de introducir datos incorrectos o dejar vacío los campos, el sistema lo informa y no permite dicha acción.	
Pos-condiciones		
1.	Muestra un mensaje explicando el error.	
Validaciones		
1.	Nombre de la actividad.	
2.	Características de la actividad.	
3.	Fase	
4.	Disciplina	
Conceptos	**Nombre de la actividad**	Nombre de la actividad: Este campo es obligatorio y debe garantizar que solo se pueda escribir letras en el mismo. El componente a visualizar es un textEdit.
	Características de la actividad	Características de la actividad: Este campo es opcional y debe garantizar que solo se pueda escribir letras y números en el mismo. El componente a visualizar es un textEdit.
	Fase	Fase: Este campo es obligatorio y debe garantizar al menos que se marque una fase. El componente a visualizar es un RadioButton.
	Disciplina	Disciplina: Este campo es obligatorio y debe garantizar al menos que se marque una disciplina. El componente a visualizar es un RadioButton.
Sección Eliminar actividad		
1.	Se muestra en la ventana emergente la vista para eliminar las actividades las mismas están listadas y organizadas por las fases y disciplinas en las que se realiza que se encuentran a la derecha de cada actividad.	
2.	Seleccionar las actividades que se desean eliminar y dar clic en la sección <Eliminar> en el extremo inferior derecho de la ventana.	
Pos-condiciones		
1.	Se actualiza la vista sin las actividades eliminada.	
Validaciones		
1.	No Procede	

Tabla 6. Gestionar actividad

Gestionar artefacto

Precondiciones	Poseer instalada la suite
Flujo de eventos	
Flujo básico <Gestionar artefacto>	
1.	Acceder a la suite desde el menú Inicio de su pc.
2.	Acceder a la suite por la sesión de administrador o de analista.
3.	Acceder a la sección <Gestión de la información> del Menú desplegable <Módulos>.
4.	Acceder a la sección <Gestionar artefacto> del módulo de <Gestión de la Información>.
5.	Seleccionar la sección que desee: <Adicionar artefacto> o <Eliminar artefacto>.
6.	Aparece una ventana emergente para seleccionar metodología a la cual se le desea gestionar el artefacto.
7.	Seleccionar la metodología y dar clic en el botón <Seleccionar>.
Pos-condiciones	
1.	Muestra la interfaz de la sección seleccionada.
Sección Adicionar artefacto	
1.	Se muestra en la ventana emergente la vista para adicionar los artefactos.
2.	Dar clic en el botón <Adicionar>.
3.	aparece una ventana emergente donde se debe cargar el artefacto que se necesita de la lista de artefactos que existe en la base de datos (Las planillas orientadas por Calisoft), se selecciona en que actividad se debe generara y en cual se utiliza como artefacto de entrada.
4.	Seleccionar la sección <Guardar> en el extremo inferior derecho de la ventana emergente.
Pos-condiciones	
1.	Se regresa a la vista anterior para si se desea continuar adicionando artefactos.
Validaciones	
1.	No Procede
Sección Eliminar artefacto	
1.	Se muestra en la ventana emergente la vista para eliminar los artefactos los mismos están listados y organizados en dependencia de las fases, disciplinas y la actividad en las que son generados. Los datos de se encuentran a la derecha de cada artefacto.
2.	Seleccionar los artefactos que se desean eliminar y dar clic en la sección <Eliminar> en el extremo inferior derecho de la ventana.
Pos-condiciones	
1.	Se elimina el artefacto de la metodología.
2.	Se actualiza la vista sin los artefactos eliminados.
Validaciones	
1.	No Procede

Tabla 7. Gestionar artefacto

Introducir información al artefacto

Precondiciones	Poseer instalada la suite

Flujo de eventos	
Flujo básico <Introducir información>	
1.	Seleccionar el artefacto que corresponde realizar.
2.	Se muestra, en el área de trabajo situada a la derecha de la aplicación el componente TabWidget, el mismo tendrá seleccionada la primera etiqueta que no se haya completado para continuar la realización del artefacto.
3.	En el panel de trabajo de esta pestaña están indicados los campos que se deben completar.
4.	Realizar el completamiento de los campos correspondientes.
5.	Al concluir la realización de esta área pulsar el botón <Guardar y continuar> situado en la parte inferior derecha del componente TabWidget.
Pos-condiciones	
1.	Se inicializa la próxima pestaña.
Flujos alternativos	
Flujo alternativo 1.A <<Introducir información del Artefacto incorrectamente>>	
1.	Si se comete el error de introducir datos incorrectos o dejar vacío los campos, el sistema lo informa y no permite dicha acción.
Pos-condiciones	
1.	Muestra un mensaje explicando el error.
Flujo alternativo 2.B <<Completar última pestaña>>	
1.	Concluir el llenado del artefacto.
Pos-condiciones	
1.	Aparece el próximo artefacto que debe ser llenado.
Validaciones	
1.	No Procede

Tabla 8. Introducir información al artefacto

Generar reportes

Precondiciones	Poseer instalada la suite
Flujo de eventos	
Flujo básico <Generar reportes>	
1.	Acceder a la suite desde el menú Inicio de su pc.
2.	Acceder a la suite por la sesión de administrador o de analista.
3.	Acceder a la sección <Gestión de la información> del Menú desplegable <Módulos>.
4.	Acceder a la sección <Gestionar reporte> del módulo de <Gestión de la Información>.
5.	Seleccionar la sección que desee: <Generar reporte en PDF> o <Generar reporte en ODT>.
6.	Aparece una ventana emergente para seleccionar metodología a la cual corresponda el artefacto que se le desea realizar el reporte.
7.	Seleccionar la metodología y dar clic en el botón <Seleccionar>.
Pos-condiciones	
1.	Muestra la interfaz de la sección seleccionada.
Sección Reporte en PDF	
1.	Aparece una ventana emergente donde se podrá seleccionar entre los artefactos que ya se han completado a cual se le realiza el reporte.
2.	Seleccionar el lugar donde se guarda el documento.

3.	Seleccionar destino.
4.	Dar clic en la sección <Guardar> en el extremo inferior derecho de la ventana emergente.
Pos-condiciones	
1.	Se genera el reporte del artefacto en la dirección seleccionada.
Validaciones	
1.	No Procede
Sección Reporte en ODT	
1.	Aparece una ventana emergente donde se podrá seleccionar entre los artefactos que ya se han completado a cual se le realiza el reporte.
2.	Seleccionar el lugar donde se guarda el documento.
3.	Seleccionar destino.
4.	Dar clic en la sección <Guardar> en el extremo inferior derecho de la ventana emergente.
Pos-condiciones	
1.	Se genera el reporte del artefacto en la dirección seleccionada.
Validaciones	
1.	No Procede

Tabla 9. Generar reportes

2.4 Modelado de las Vistas del Sistema

A continuación se representa la arquitectura a través de las vistas anteriormente tratadas.

2.4.1 Vista de Caso de Uso

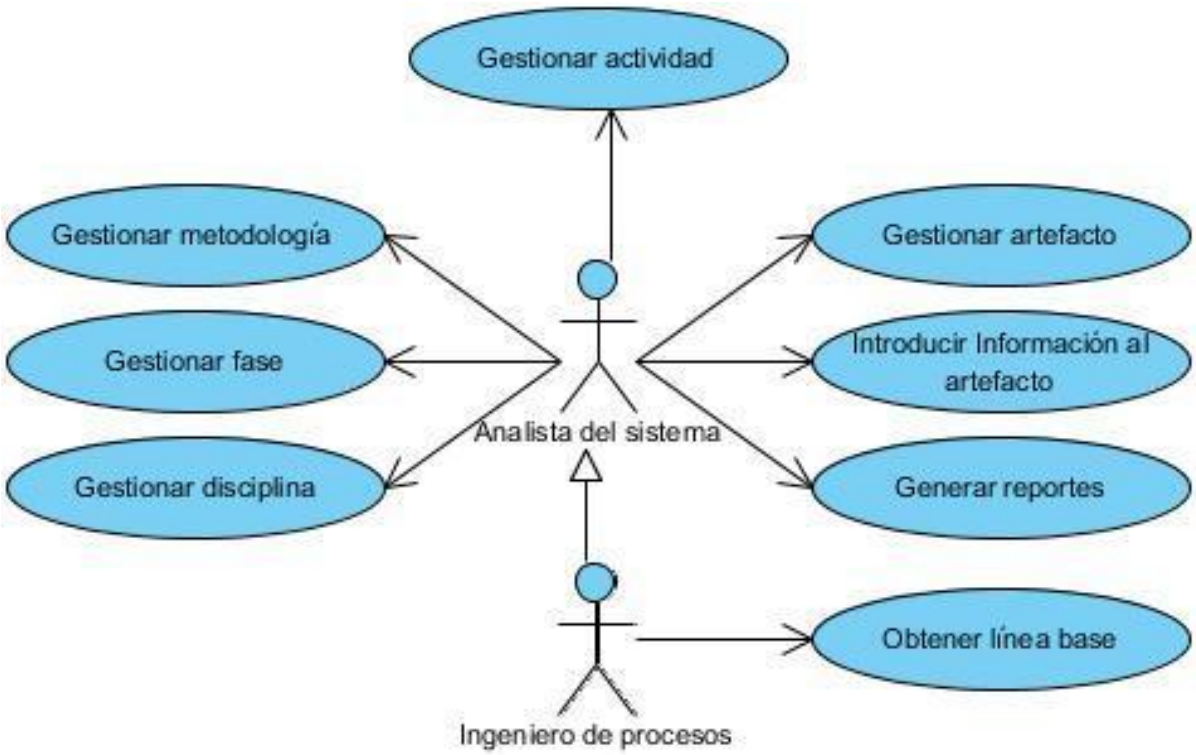

Imagen 2. Diagrama de Caso de uso

Gestionar metodología: permite adicionar, modificar y eliminar metodologías. Puede ser realizado por el analista del sistema o por el ingeniero de proceso.

Gestionar fase: permite adicionar, modificar y eliminar fases. Puede ser realizado por el analista del sistema o por el ingeniero de proceso.

Gestionar disciplina: permite adicionar, modificar y eliminar disciplinas. Puede ser realizado por el analista del sistema o por el ingeniero de proceso.

Gestionar actividad: permite adicionar, modificar y eliminar actividades. Puede ser realizado por el analista del sistema o por el ingeniero de proceso.

Gestionar artefactos: permite adicionar y eliminar metodología. Puede ser realizado por el analista del sistema o por el ingeniero de proceso.

Introducir información del artefacto: permite realizar el completamiento de los campos necesarios del artefacto que corresponda con la metodología, la fase y disciplina en que se esté trabajando para ir completando el expediente de proyecto. Puede ser realizado por el analista del sistema o por el ingeniero de proceso.

Crear reportes: permite realizar los reportes tanto en formato PDF u ODT de los artefactos seleccionados. Puede ser realizado por el analista del sistema o por el ingeniero de proceso.

Obtener línea base: se realiza una vez terminado todo el desarrollo del software y con él, los productos de trabajo correspondientes. Se exporta en formato PDF todos los artefactos correspondientes al expediente de proyecto. Sólo puede ser realizado por el ingeniero de proceso.

Se utiliza en el diagrama los patrones de múltiples actores y CRUD parcial. El primero porque el caso de uso **Obtener línea base** no puede ser inicializado por varios actores, sólo por el Ingeniero de procesos el cual puede realizar también las demás actividades del Analista del sistema. El patrón CRUD se evidencia a la hora de reunir las acciones básicas en un solo caso de uso, un ejemplo sería el caso de uso **Gestionar metodología,** en el mismo se realiza la acción de adicionar, modificar y eliminar metodología.

2.4.2 Vista Lógica

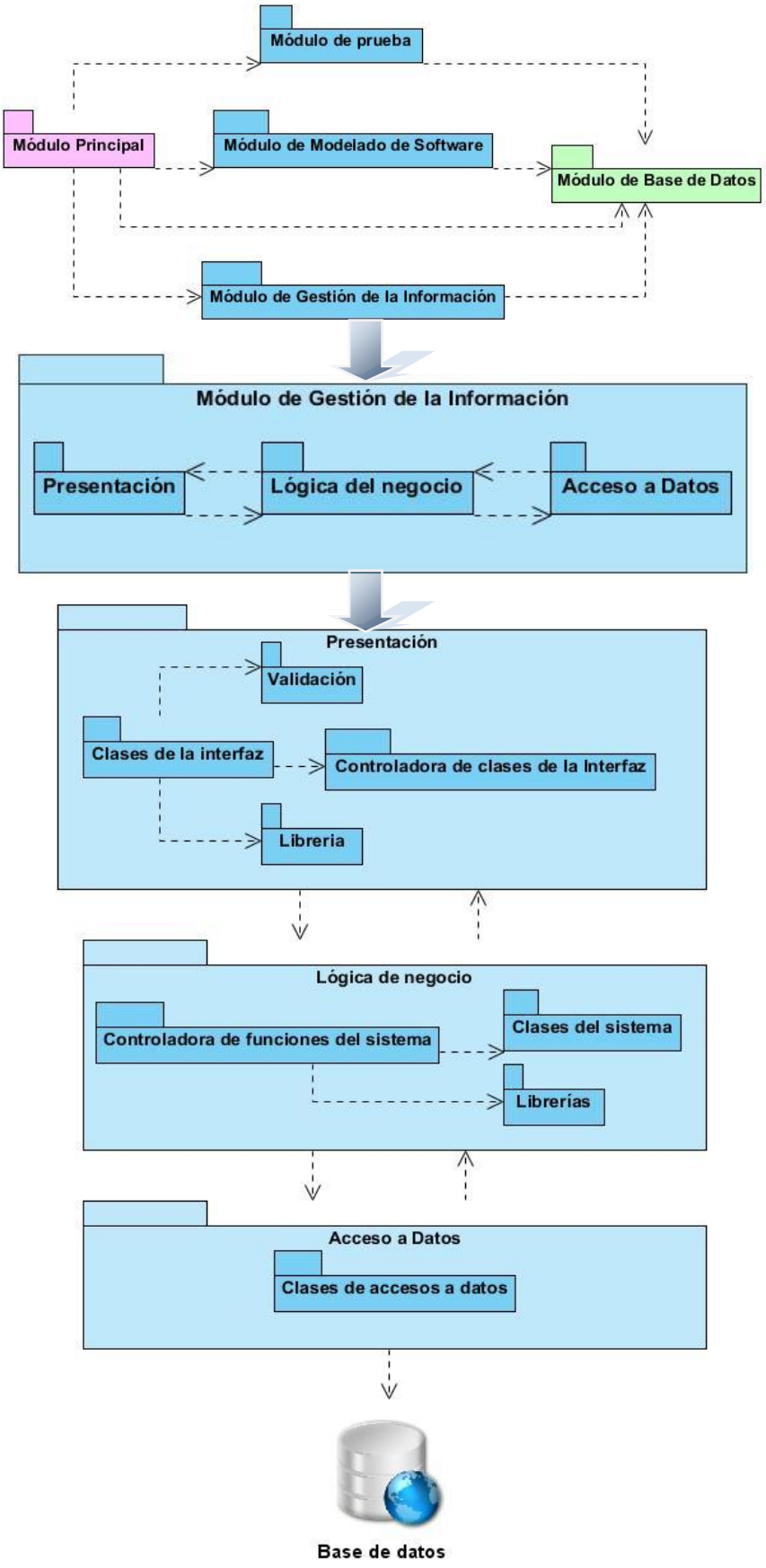

Imagen 3. Diagrama de Paquetes

En la realización de los diagramas de paquetes se utilizó el patrón recursivo, el mismo es guiado por los niveles de abstracción de cada agrupación (21).

Imagen 4. Nivel de empaquetamiento utilizado

Nivel de sistema: Es la vista global de la Suite de Ingeniería de Software.

Nivel de subsistema: Es la vista general del Módulo de gestión de la información.

Nivel de componente: Es la vista del nivel más bajo de integración en la dimensión lógica de la integración.

2.4.3 Vista de Componentes

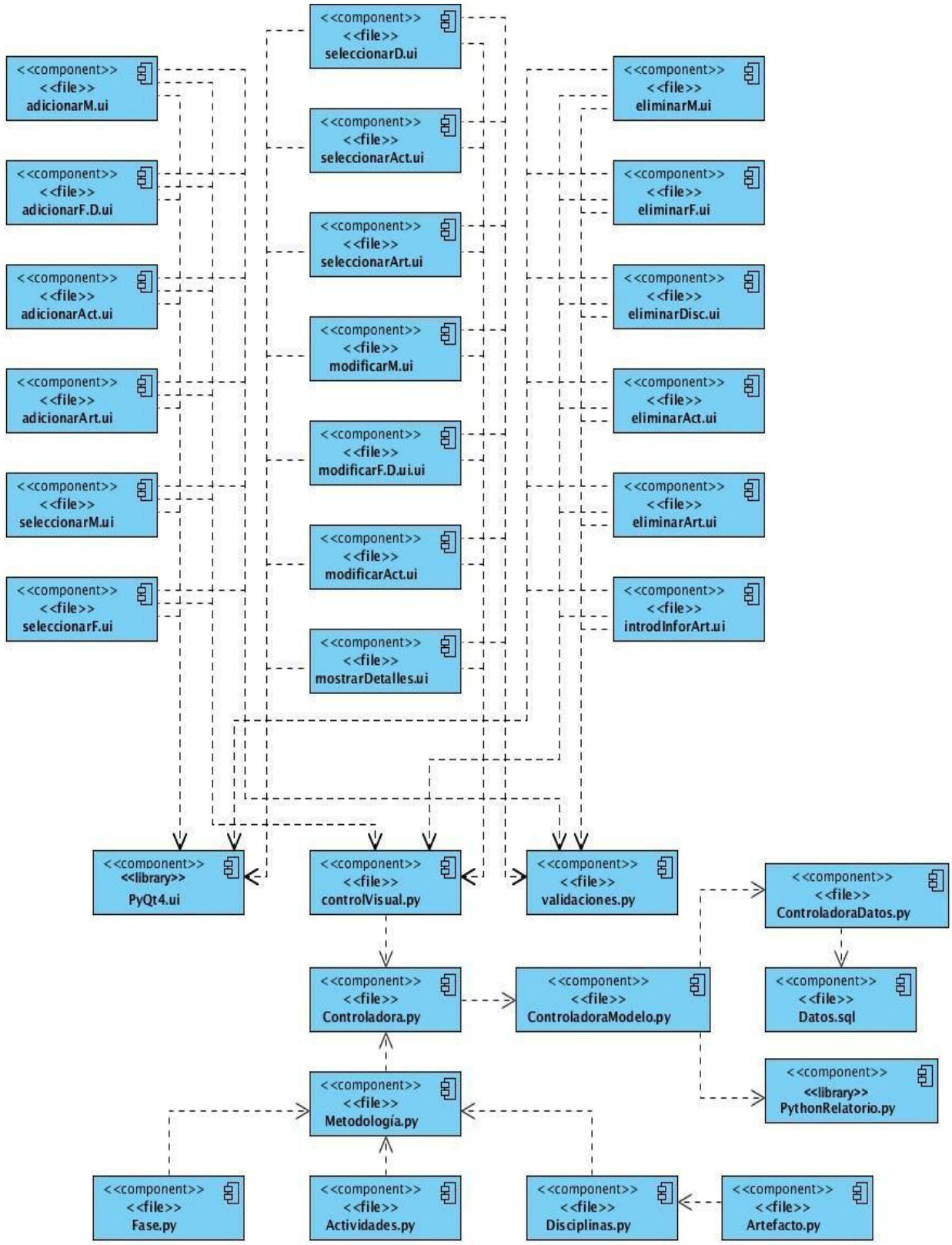

Imagen 5. Diagrama de componentes

adicionarM.ui: corresponde a la interfaz para adicionar una metodología.

adicionarF.D.ui: corresponde a la interfaz para adicionar una fase o una disciplina.

adicionarAct.ui: corresponde a la interfaz para adicionar una actividad.

adicionarArt.ui: corresponde a la interfaz para adicionar un artefacto.

seleccionarM.ui: corresponde a la interfaz para seleccionar una metodología.

seleccionarF.ui: corresponde a la interfaz para seleccionar una fase.

seleccionarD.ui: corresponde a la interfaz para seleccionar una disciplina.

seleccionarAct.ui: corresponde a la interfaz para seleccionar una actividad.

seleccionarArt.ui: corresponde a la interfaz para seleccionar un artefacto.

modificarM.ui: corresponde a la interfaz para modificar una metodología.

modificarF.D.ui: corresponde a la interfaz para modificar una fase o una disciplina.

modificarAct.ui: corresponde a la interfaz para modificar una actividad.

mostrarDetalles.ui: corresponde a la interfaz para mostrar los detalles de la metodología, las fases, las disciplinas, las actividades y los artefactos.

eliminarM.ui: corresponde a la interfaz para eliminar una metodología.

eliminarF.ui: corresponde a la interfaz para eliminar una fase.

eliminarDisc.ui: corresponde a la interfaz para eliminar una disciplina.

eliminarAct.ui: corresponde a la interfaz para eliminar una actividad.

eliminarArt.ui: corresponde a la interfaz para eliminar un artefacto.

introdInforArt.ui: corresponde a la interfaz para introducir la información al artefacto.

PyQt4.ui: se utiliza para crear las interfaces y generarlas a código Python.

validaciones.py: se utiliza para validar los datos que introduzcan los usuarios.

controlVisual.py: controla el flujo de datos dentro de la capa de presentación y la de la lógica del negocio.

Controladora.py: controla el flujo de datos correspondiente a la capa de la lógica del negocio y su relación con la capa de presentación y la de acceso a datos. Además de ser la clase creadora de las metodologías porque en ella se encuentran los datos necesarios para esta tarea.

Metodología.py: corresponde a la clase Metodología, que es la encargada de crear las fases, las actividades y los artefactos.

Fase.py: corresponde a la clase Fase, y en ella se encuentran sus datos.

Actividades.py: corresponde a la clase Actividades, y en ella se encuentran sus datos.

Disciplinas.py: este componente corresponde a la clase Disciplina, y en ella se encuentran sus datos. Además es la encargada de adicionar los artefactos.

Artefacto.py: corresponde a la clase Artefacto, y en ella se encuentran sus datos.

ControladoraModelo.py: controla el flujo de datos entre el componente Controladora y el PythonRelatorio.

PythonRelatorio.py: corresponde a la librería de Python-Relatorio.

ControladoraDatos.py: controla el flujo de datos correspondiente a la capa de la lógica del negocio y la de acceso a datos.

Datos.sql: corresponde a los datos guardados en la base de datos correspondiente al módulo.

2.4.4 Vista de Despliegue

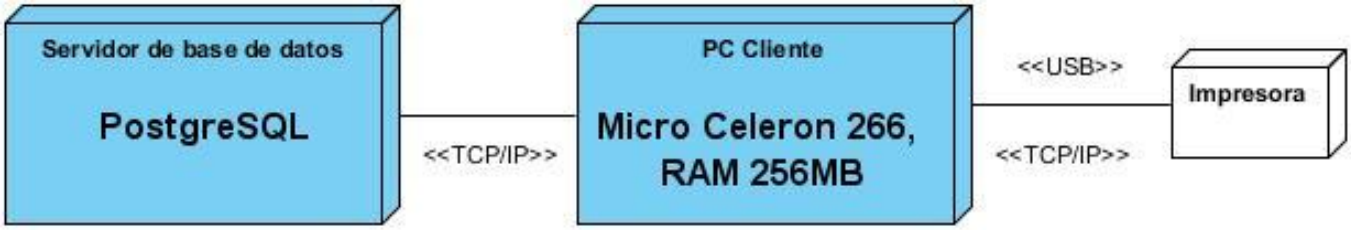

Imagen6. Diagrama de despliegue

El Módulo de Gestión de la Información utiliza un servidor de base de datos PostgreSQL para guardar los datos que ingresen los usuarios. Se comunica con la PC cliente a través del protocolo de comunicación TCP/IP. El sistema debe poder ejecutarse en las PC cliente con prestaciones de hardware mínimas de: Micro Celeron 266, RAM 256MB. Las mismas utilizan los protocolos TCP/IP o USB para comunicarse con la impresora.

2.4.5 Vista de Proceso

Por decisión del grupo de trabajo del proyecto esta vista la realiza el programador del módulo.

2.5 Conclusiones parciales

Durante el desarrollo del capítulo, fue posible especificar las disposiciones del proyecto para la construcción de la Suite, de manera que en la elaboración de los entregables se empleara la herramienta CASE Visual Paradigm y el framework Qt Designer. Por lo tanto la propuesta de solución quedó sintetizada a través del diseño de los artefactos: diagrama de caso de uso, diagrama de clases, diagrama de paquetes, diagrama de componentes y diagrama de despliegue. Los diagramas anteriormente mencionados muestran la organización y distribución de los aspectos importantes para la construcción de la propuesta.

Capítulo 3: Evaluación de la arquitectura del módulo de Gestión de la Información

Introducción:

La arquitectura de software es importante como disciplina debido a que los sistemas de software crecen de forma tal que resulta muy complicado que sean diseñados, especificados y entendidos por un solo individuo (33). Cuanto más temprano se encuentre un problema en un proyecto de software, mejor. El costo de arreglar un error durante las fases de elaboración cuando se tenga realizado el diseño de la arquitectura, es mucho menor al costo de arreglar ese mismo error en la fase de construcción. Dado que la arquitectura es un producto temprano y tiene un profundo efecto en el sistema y en el proyecto.

Luego de proponer la arquitectura del Módulo de Gestión de la Información, es preciso evaluar si la misma está correctamente diseñada. Por lo antes expuesto, en el presente capítulo se exponen aspectos relacionados con la evaluación de la arquitectura. Se investiga sobre los atributos de calidad y los métodos de evaluación de la arquitectura para seleccionar los indicados. Luego de la selección se realizan las pruebas y se exponen los resultados de las mismas.

3.1 Atributos de calidad a evaluar en la arquitectura

El tener en cuenta atributos de calidad para la arquitectura de un sistema provee una base para la toma de decisiones objetivas sobre acuerdos de diseño y permite a los ingenieros realizar predicciones razonablemente exactas sobre los atributos del sistema que son libres de prejuicios y asunciones no triviales. El objetivo de fondo es lograr la habilidad de evaluar cuantitativamente y llegar a acuerdos entre múltiples atributos de calidad para alcanzar un mejor sistema de forma global (34).

En 1998, Bass establece una clasificación de los atributos de calidad en dos categorías: Observables y los No Observales. La descripción de algunos de estos atributos se presenta en la siguiente tabla (35).

Observables: aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución

Atributos de calidad	Descripción
Disponibilidad	Es la medida de disponibilidad del sistema para el uso.
Confidencialidad	Es la ausencia de acceso no autorizado a la información.
Funcionalidad	Habilidad del sistema para realizar el trabajo para el cual fue concebido.
Desempeño	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. Es la capacidad de respuesta, ya sea el tiempo requerido para responder a aspectos específicos o el número de eventos procesados en un intervalo de tiempo. Además a la cantidad de comunicación e interacción existente entre los componentes del sistema.
Confiabilidad	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo.
Seguridad externa	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información.
Seguridad interna	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos.
No observables: aquellos atributos que se establecen durante el desarrollo del sistema.	
Atributos de calidad	**Descripción**
Configurabilidad	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema.
Integrabilidad	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados.
Integridad	Es la ausencia de alteraciones inapropiadas de la información.
Interoperabilidad	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de integrabilidad.
Modificabilidad	Es la habilidad de realizar cambios futuros al sistema.
Mantenibilidad	Es la capacidad de someter a un sistema a reparaciones y evolución, modificándolo de manera rápida y a bajo costo.
Portabilidad	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos.
Reusabilidad	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones.
Escalabilidad	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental.
Capacidad de Prueba	Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas. Es la probabilidad de que, asumiendo que tiene al menos una falla, el software fallará en su

	próxima ejecución de prueba.

Tabla 10. Atributos de calidad (Bass)

3.2 Modelos de calidad

Para lograr predecir la confiabilidad y la calidad durante el proceso de desarrollo, se hace uso de los modelos de calidad. Pressman en el 2002 expone que los factores que afectan a la calidad del software no cambian (15), por esta razón se realiza un estudio de los modelos de: McCall, Boehm e ISO/IEC 9126.

3.2.1 McCall (1977)

McCall, elabora su concepto de calidad mediante las relaciones jerárquicas entre factores de calidad que se basan en criterios y métricas de calidad. Este enfoque permite cuantificar la calidad mediante los siguientes factores (36):

Factores	Criterios
Correctitud	• Rastreabilidad • Completitud • Consistencia
Confiabilidad	• Consistencia • Exactitud • Tolerancia a fallas
Eficiencia	• Eficiencia de ejecución • Eficiencia de almacenamiento
Integridad	• Control de acceso • Auditoría de acceso
Usabilidad	• Operabilidad • Entrenamiento • Comunicación
Mantenibilidad	• Simplicidad • Concreción
Capacidad de Prueba	• Simplicidad

	• Instrumentación • Auto-descriptividad • Modularidad
Flexibilidad	• Auto-descriptividad • Capacidad de expansión • Generalidad • Modularidad
Portabilidad	• Auto-descriptividad • Independencia del sistema • Independencia de máquina
Reusabilidad	• Auto-descriptividad • Generalidad • Modularidad • Independencia del sistema • Independencia de máquina
Interoperabilidad	• Modularidad • Similitud de comunicación • Similitud de datos.

Tabla 11. Factores de calidad (McCall)

3.2.2 Boehm (1978)

El modelo que presenta Boehm se basa en algunas de las características que expone McCall como la operatividad, la capacidad de soportar el cambio y su adaptabilidad a nuevos entornos y agrega 19 criterios que incluyen la evaluación del desempeño del hardware. Este modelo representa una estructura jerárquica de características donde cada una de ellas contribuye a la calidad total. Las características se descomponen en tres niveles (usos principales, componentes intermedios y componentes primitivos) previos a la aplicación de métricas (37).

El modelo tiene como finalidad que a través de la calidad del software, el mismo permita:

- Realizar lo que el usuario desea.
- Utilizar recursos informáticos de manera correcta y eficiente.

- Ser fácil de utilizar y aprender.
- Estar bien diseñado, codificado, probado y mantenido.

Las métricas directas e indirectas son usadas para determinar el nivel de acuerdo a un criterio en particular que afecta a los principales factores de calidad. Factores tales como portabilidad, confiabilidad, facilidad de mantenimiento y facilidad de modificación son propiedades estáticas. Cada factor es descompuesto en varios criterios. La facilidad de prueba y la eficiencia dependen del comportamiento de las interpretaciones específicas y constituyen propiedades dinámicas (38).

3.2.3 ISO/IEC 9126 (1991)

El estándar ISO/IEC 9126 es una simplificación del modelo McCall. Identifica seis características básicas de calidad que pueden estar presentes en cualquier producto de software. Este modelo provee una descomposición de las características en subcaracterísticas como se muestra a continuación.

Característica	Subcaracterística
Funcionalidad	• Adecuación • Exactitud • Interoperabilidad • Seguridad
Confiabilidad	• Madurez • Tolerancia a fallas • Recuperabilidad
Usabilidad	• Entendibilidad • Capacidad de aprendizaje • Operabilidad
Eficiencia	• Comportamiento en tiempo • Comportamiento de recursos
Mantenibilidad	• Analizabilidad • Modificabilidad • Estabilidad

	• Capacidad de pruebas
Portabilidad	• Adaptabilidad • Instalabilidad • Reemplazabilidad

Tabla 12. Características y subcaracterísticas (ISO/IEC 9126)

En el 2003 se propone una adaptación de este modelo para efectos de la evaluación de arquitecturas de software. El modelo se basa en los atributos de calidad que se relacionan directamente con la arquitectura: funcionalidad, confiabilidad, eficiencia, mantenibilidad y portabilidad. Posteriormente se presentan los atributos de calidad planteados por Losavio que poseen subcaracterísticas asociadas con elementos de tipo arquitectónicos.

Características	Subcaracterística	Elementos de tipo arquitectónico
Funcionalidad	Adecuación	Refinamiento de los diagramas de secuencia.
	Exactitud	Identificación de los componentes con las funciones responsables de los cálculos.
	Interoperabilidad	Identificación de conectores de comunicación con sistemas externos.
	Seguridad	Mecanismos o dispositivos que realizan explícitamente la tarea.
Confiabilidad	Tolerancia a fallas	Existencia de mecanismos o dispositivos de software para manejar excepciones.
	Recuperabilidad	Existencia de mecanismos o dispositivos de software para restablecer el nivel de desempeño y recuperar datos.
Eficiencia	Desempeño	Componentes involucrados en un flujo de ejecución para una funcionalidad.
	Utilización de recursos	Relación de los componentes en términos de espacio y tiempo.
Mantenibilidad	Acoplamiento	Interacciones entre componentes.
	Modularidad	Número de componentes que dependen de un componente.
	Adaptabilidad	Presencia de mecanismos de adaptación.

Portabilidad	Instalabilidad	Presencia de mecanismos de instalación.
	Coexistencia	Presencia de mecanismos que faciliten la coexistencia.
	Reemplazabilidad	Lista de componentes reemplazables para cada componente.

Tabla 13. Atributos de calidad (ISO/IEC 9126)

La Norma ISO/IEC 9126 es un estándar internacional para la evaluación del software y está enfocada a la calidad del producto. Se encuentra dividida en cuatro partes las cuales dirigen, respectivamente lo siguiente (39):

Parte 1: Calidad de proceso

Parte 2: Métricas externas (permiten evaluar la calidad del producto software durante el ensayo o la operación)

Parte3: Métricas internas (permiten atender los aspectos de la calidad desde las etapas más tempranas antes de que el producto software devenga ejecutable)

Parte4: Métricas de calidad en el uso (miden hasta qué punto un producto satisface las necesidades de usuarios específicos para lograr las metas especificadas con la eficacia, productividad, seguridad y satisfacción en un contexto determinado de uso).

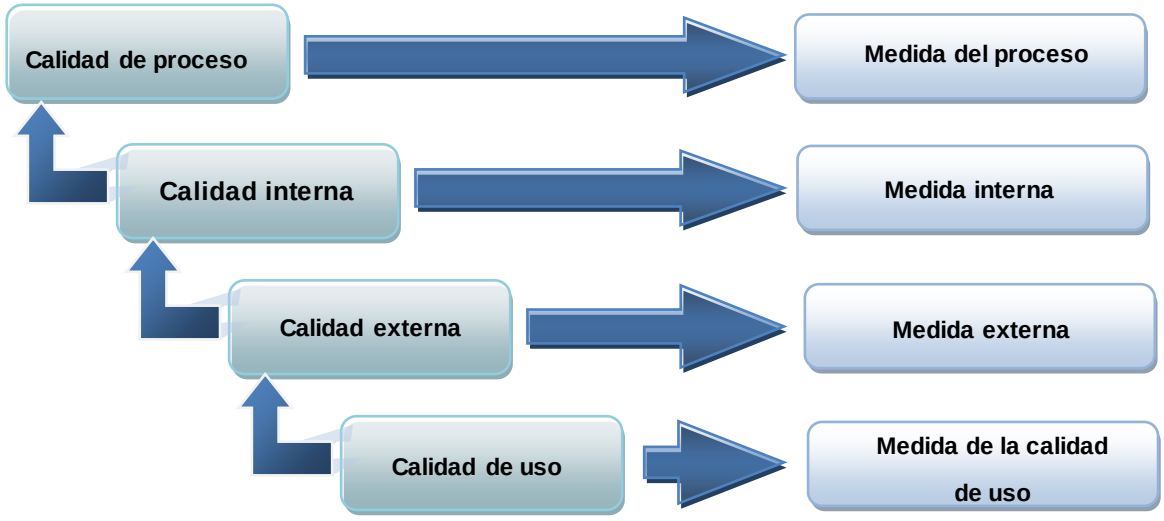

Imagen 7. Modelo de calidad según ISO/IEC 9126

En la tabla que se presenta a continuación se realiza una comparación entre los modelos estudiados.

Modelo	Componentes	Niveles	Elemento de cada Nivel	Métricas definidas
McCall	* Factores * Criterios * Métricas	* Subdivisión principal * Factores * Criterios * Métricas	* 14 factores(3 en subdivisión principal más 11 en el segundo nivel) * 23 criterios * Métricas no indicadas(definibles por el usuario)	No define métricas.
Boehm	* Características de alto nivel * Características primitivas * Métricas	* 2 capas de características de alto nivel * 1 nivel de primitivas * 1 nivel de métricas	* 9 características de alto nivel * 15 primitivas * Métricas no indicadas(definibles por el usuario)	No define métricas.
ISO/IEC 9126 (1991)	* Características * Subcaracterísticas * Atributos de calidad * Métricas	* 2 niveles superiores * Niveles de atributos: no indicados (definidos por el usuario) * Niveles de métricas: asociados con cada nivel de atributos.	* 6 Características * 27 Subcaracterísticas * Atributos de calidad: no indicados (definidos por el usuario) Métricas: no indicados (definidos por el usuario) pero se exponen las métricas estándar.	Si define métricas.

Tabla 14. Comparación entre los modelos

El modelo seleccionado por la autora del presente documento fue el ISO/IEC 9126 (1991) porque un procedimiento robusto y flexible para la medición de la calidad de software a partir de la evaluación de la conformidad de iteraciones de pruebas tempranas utilizando una norma internacional para evaluar la conformidad (ISO/IEC 14598). Además es el utilizado en la UCI para medir la calidad de los productos de software (40).

3.3 Métodos de evaluación de la arquitectura

Kazman expone que resulta de poco interés la caracterización o medición de atributos de calidad en las fases tempranas del proceso de diseño, porque los parámetros por lo general

son dependientes de la implementación. Este autor se enfoca hacia la mitigación de riesgos, donde el atributo de calidad se ve afectado por decisiones arquitectónicas y presenta tres métodos de evaluación de arquitecturas de software, ellos son (39):

3.3.1 *Architecture Trade-off Análisis Method* (ATAM)

Este método está inspirado en tres áreas distintas, los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM, que será abordado posteriormente. ATAM se centra en la identificación de los estilos o enfoques arquitectónicos utilizados. Permite describir la forma en la que el sistema puede crecer, responder a los cambios e integrarse con otros sistemas. El método comprende nueve pasos, estos pasos están agrupados en cuatro fases, a continuación se presenta la descripción de los mismos (34).

Fase 1: Presentación	
1. Presentación del ATAM	El líder de evaluación describe el método a los participantes, trata de establecer las expectativas y responde las preguntas propuestas.
2. Presentación de las metas del negocio	Se realiza la descripción de las metas del negocio que motivan el esfuerzo, y aclara que se persiguen objetivos de tipo arquitectónico.
3. Presentación de la arquitectura	El arquitecto describe la arquitectura, enfocándose en cómo ésta cumple con los objetivos del negocio.
Fase 2: Investigación y análisis	
4. Identificación de los enfoques arquitectónicos	Estos elementos son detectados, pero no analizados.
5. Generación del Utility Tree	Se utilizan los atributos de calidad que engloban la “utilidad” del sistema (desempeño disponibilidad, seguridad, modificabilidad, usabilidad, etc.), especificados en forma de escenarios. Se anotan los estímulos y respuestas, así como se establece la prioridad entre ellos.
6. Análisis de los enfoques arquitectónicos	Con base en los resultados del establecimiento de prioridades del paso anterior, se analizan los elementos del paso 4. En este paso se identifican riesgos arquitectónicos, puntos de sensibilidad y puntos de balance.
Fase 3: Pruebas	
7. Lluvia de ideas y establecimiento de	Con la colaboración de todos los involucrados, se complementa el conjunto de escenarios.

prioridad de escenarios.	
8. Análisis de los enfoques arquitectónicos	Este paso repite las actividades del paso 6, haciendo uso de los resultados del paso 7. Los escenarios son considerados como casos de prueba para confirmar el análisis realizado hasta el momento.
Fase 4: Reporte	
9. Presentación de los resultados	Basado en la información recolectada a lo largo de la evaluación del ATAM, se presentan los hallazgos a los participantes.

Tabla 15. Fases del método ATAM

3.3.2 *Software Architecture Analysis Method* (SAAM)

Este método fue creado para el análisis de la modificabilidad de una arquitectura, pero ha demostrado ser muy útil para evaluar de forma rápida distintos atributos de calidad como son: modificabilidad, portabilidad, escalabilidad e integridad. SAAM se enfoca en la enumeración de un conjunto de escenarios que representan los cambios probables a los que está sometido el sistema en el futuro. A continuación se presenta los pasos que contemplan este método (34).

Pasos	Descripción
1. Desarrollo de escenarios	Un escenario es una breve descripción de usos anticipados o deseados del sistema. De igual forma, estos pueden incluir cambios a los que puede estar expuesto el sistema en el futuro.
2. Descripción de la arquitectura	La arquitectura (o las candidatas) debe ser descrita haciendo uso de alguna notación arquitectónica que sea común a todas las partes involucradas en el análisis. Deben incluirse los componentes de y conexiones relevantes, así como la descripción del comportamiento general del sistema. El desarrollo de escenarios y la descripción de la arquitectura son usualmente llevados a cabo de forma intercalada, o a través de varias iteraciones.
3. Clasificación y asignación de prioridad de los escenarios	La clasificación de los escenarios puede hacerse en dos clases: directos e indirectos. Un escenario directo es el que puede satisfacerse sin la necesidad de modificaciones en la arquitectura. Un escenario indirecto es aquel que requiere modificaciones en la arquitectura para poder satisfacerse.

	Los escenarios indirectos son de especial interés para SAAM, pues son los que permiten medir el grado en el que una arquitectura puede ajustarse a los cambios de evolución que son importantes para los involucrados en el desarrollo.
4. Evaluación individual de los escenarios indirectos	Para cada escenario indirecto, se listan los cambios necesarios sobre la arquitectura, y se calcula su costo. Una modificación sobre la arquitectura significa que debe introducirse un nuevo componente o conector, o que alguno de los existentes requiere cambios en su especificación.
5. Evaluación de la interacción entre escenarios	Cuando dos o más escenarios indirectos proponen cambios sobre un mismo componente, se dice que interactúan sobre ese componente. Es necesario evaluar este hecho, puesto que la interacción de componentes semánticamente no relacionados revela que los componentes de la arquitectura efectúan funciones semánticamente distintas. De forma similar, puede verificarse si la arquitectura se encuentra documentada a un nivel correcto de descomposición estructural.
6. Creación de la evaluación global	Debe asignársele un peso a cada escenario, en términos de su importancia relativa al éxito del sistema. Esta asignación de peso suele hacerse con base en las metas del negocio que cada escenario soporta. En el caso de la evaluación de múltiples arquitecturas, la asignación de pesos puede ser utilizada para la determinación de una escala general.

Tabla 16. Pasos del método SAAM

3.3.3 *Active Intermediate Designs Review* (ARID)

ARID es conveniente para evaluar los diseños en las etapas tempranas de los diseños de una arquitectura de software, según sus creadores es un híbrido entre *Active Design Review* (ADR) y ATAM. Es utilizado para la evaluación de diseño detallado de unidades del software como los componentes o módulos. La tabla que se presenta a continuación describirá los de este método (34).

Fase 1: Actividades Previas	
1. Identificación de los encargados de la revisión	Los encargados de la revisión son los ingenieros de software que se espera que usen el diseño, y todos los involucrados en el diseño. En este punto, converge el concepto de encargado de revisión de ADR e involucrado del ATAM.
2. Preparar el informe de diseño	El diseñador prepara un informe que explica el diseño. Se incluyen ejemplos del uso del mismo para la resolución de problemas reales. Esto permite al facilitador anticipar el tipo de preguntas posibles, así

	como identificar áreas en las que la presentación puede ser mejorada.
3. Preparar los escenarios bases	El diseñador y el facilitador preparan un conjunto de escenarios base. De forma similar a los escenarios del ATAM y el SAAM, se diseñan para ilustrar el concepto de escenario, que pueden o no ser utilizados para efectos de la evaluación.
4. Preparar los materiales	Se reproducen los materiales preparados para ser presentados en la segunda fase. Se establece la reunión, y los involucrados son invitados.
Fase 2: Revisión	
5. Presentación del ARID	Se explica los pasos del ARID a los participantes.
6. Presentación del diseño	El líder del equipo de diseño realiza una presentación, con ejemplos incluidos. Se propone evitar preguntas que conciernen a la implementación o argumentación, así como alternativas de diseño. El objetivo es verificar que el diseño es conveniente.
7. Lluvia de ideas y establecimiento de prioridad de escenarios	Se establece una sesión para la lluvia de ideas sobre los escenarios y el establecimiento de prioridad de escenarios. Los involucrados proponen escenarios a ser usados en el diseño para resolver problemas que esperan encontrar. Luego, los escenarios son sometidos a votación, y se utilizan los que resultan ganadores para hacer pruebas sobre el diseño.
8. Aplicación de los escenarios	Comenzando con el escenario que contó con más votos, el facilitador solicita pseudo-código que utiliza el diseño para proveer el servicio, y el diseñador no debe ayudar en esta tarea. Este paso continúa hasta que ocurra alguno de los siguientes eventos: * Se agota el tiempo destinado a la revisión * Se han estudiado los escenarios de más alta prioridad * El grupo se siente satisfecho con la conclusión alcanzada Puede suceder que el diseño presentado sea conveniente, con la exitosa aplicación de los escenarios, o por el contrario, no conveniente, cuando el grupo encuentra problemas o deficiencias.
9. Resumen	Al final, el facilitador recuenta la lista de puntos tratados, pide opiniones de los participantes sobre la eficiencia del ejercicio de revisión, y agradece por su participación.

Tabla 17. Fases del método ARID

El método seleccionado para la realización de las pruebas de la arquitectura del Módulo de Gestión de la Información es ATAM. Este método se enfoca en la detección de riesgos apoyándose de los atributos de calidad que son de interés, permitiendo evaluar si la

arquitectura es capaz de mitigar estos riesgos. Además evalúa con más profundidad, en relación con otros métodos, cuestiones referentes a los atributos de calidad.

3.4 Priorización y ordenamiento de escenarios.

Los escenarios describen la interacción de los involucrados en el desarrollo del sistema con este. Están compuesto por:

- Estímulo, parte del escenario que explica o describe lo que el involucrado en el desarrollo hace para iniciar la interacción con el sistema.
- Contexto, describe qué sucede en el sistema al momento del estímulo.
- Respuesta, describe, a través de la arquitectura, cómo debería responder el sistema ante el estímulo.

Este último elemento es el que permite establecer cuál es el atributo de calidad asociado. Los escenarios proveen un vehículo que permite concretar y entender atributos de calidad (40).

Una vez realizados los escenarios los mismos se organizan utilizando los siguientes criterios:

- Evaluar el riesgo de no considerar el escenario. Se deben responder las preguntas: ¿qué pasa si este escenario no se cumple?, ¿cuántas personas se ven afectadas?, ¿es posible compensar el no responder a este escenario?
- Evaluar el esfuerzo necesario para lograr cumplir con el escenario. En este caso se determina que cambios o integraciones de nuevos componentes se deben realizar para satisfacer el escenario. Los escenarios más difíciles son aquellos en los que se debe consumir más tiempo de análisis y el resto debe ser guardado como parte del registro (41).

A continuación se exponen los escenarios analizados, los mismos ya se encuentran ordenados por los criterios antes presentados:

Escenario 1: Los tiempos de respuesta y procesamientos son breves.

Atributos: Eficiencia – Desempeño.

Estímulo: Crear metodología.

Respuesta: Se Muestra una ventana para adicionar las fases y las disciplinas.

Escenario 2: La aplicación asimila con facilidad los cambios que se le puedan realizar.

Atributos: Mantenibilidad - Acoplamiento.

Estímulo: Realizar cambios en una clase por parte del programador.

Respuesta: Se realiza el cambio sin que este tenga grandes afectaciones en las demás clases.

Escenario 3: La aplicación debe poder realizar los requisitos funcionales.

Atributos: Funcionalidad - Exactitud.

Estímulo: Inicializar la aplicación y probar las funcionalidades.

Respuesta: Se puede realizar todas las funcionalidades que se exige.

Escenario 4: La aplicación protege los datos que se le introduce.

Atributos: Funcionalidad - Seguridad.

Estímulo: Guardar cambios realizados.

Respuesta: Los cambios son guardados en la base de datos.

Escenario 5: La arquitectura del sistema debe permitir el mantenimiento correcto y rápido ante cualquier fallo de software.

Atributos: Confiabilidad - Tolerancia a fallas.

Estímulo: Adicionar un artefacto a una actividad.

Respuesta: El sistema no contiene la planilla.

Escenario 6: La aplicación debe poder recuperar sus capacidades y datos luego de un fallo.

Atributos: Confiabilidad - Recuperabilidad.

Estímulo: Introducir información a un artefacto.

Respuesta: Muestra la interfaz para realizar esta operación.

Escenario 7: La aplicación puede realizar un grupo de tareas asignadas por un usuario en un corto tiempo.

Atributos: Funcionalidad - Adecuación.

Estímulo: Varias tareas son solicitadas por el usuario que interactúa en la aplicación.

Respuesta: Procesamiento de las tareas en el orden en que son solicitadas por el usuario.

Escenario 8: Las funcionabilidades solo las puede realizar el usuario que posea el permiso.

Atributos: Funcionalidad - Seguridad.

Estímulo: Crear línea base.

Respuesta: Solo la puede crear el ingeniero de procesos.

Escenario 9: La aplicación debe poder ejecutarse en diferentes sistemas operativos.

Atributos: Portabilidad – Adaptabilidad

Estímulo: Inicializar la aplicación en el sistema operativo NOVA y Windows.

Respuesta: Se podrá utilizar la aplicación.

3.5 Árbol de utilidad

La prioridad del escenario es definida por un par de variables (X,Y) en donde X define el esfuerzo de satisfacer el escenario, y Y indica los riesgos que se corren al excluirlos del árbol de utilidad. Los posibles valores para X e Y son A (Alto), B (Bajo) y M (Medio) (40).

Características	Subcaracterística	Prioridad	Escenarios
Eficiencia	Desempeño	(A,A)	Los tiempos de respuesta y procesamientos serán breves.
Mantenibilidad	Acoplamiento	(A,A)	La aplicación asimila con facilidad los cambios que se le puedan realizar.
Funcionalidad	Exactitud	(A,A)	La aplicación debe poder realizar los requisitos funcionales.
	Seguridad	(M,A)	La aplicación protege los datos que se le introduce.
	Adecuación	(A,M)	La aplicación puede realizar un grupo de tareas asignadas por un usuario en un corto tiempo.
	Seguridad	(B,B)	Las funcionalidades solo las puede realizar el usuario que posea el permiso.
Confiabilidad	Tolerancia a fallas	(A,A)	La arquitectura del sistema debe permitir el mantenimiento correcto y rápido ante cualquier fallo de software.
	Recuperabilidad	(M,M)	La aplicación debe poder recuperar sus capacidades luego de un fallo, así como de

			datos.
Portabilidad	Adaptabilidad	(M,M)	La aplicación debe poder ejecutarse en diferentes sistemas operativos.

Tabla 18. Árbol de utilidad

3.6 Análisis de riesgos

A continuación se exponen los riesgos que se pueden presentar, y luego de estudiar y analizar nuevamente la arquitectura propuesta se exhibe las características que la misma posee que ayudan a mitigar estos riesgos.

Riesgos identificados	Decisión
Problemas para realizar algunas de las funcionalidades del sistema.	El uso de las herramientas CASE: Eclipse, Visual Paradigm, Qt Designer y RapidSVN, el apoyo de losframework Python-augeas, Python-paramiko, Python-relatorio, PyQt4 y utilizando los estilos Cliente/Servidor organizado en 3-Capas, permiten desarrollar el Módulo de Gestión de la Información garantizando que el mismo cumpla con sus funcionalidades de forma eficiente.
Tardanza en los tiempos de respuesta y procesamiento de la información.	La eficiencia se garantiza a través la organización en 3 Capas de la arquitectura y de los patrones utilizados, que permite que el módulo posea un bajo acoplamiento y una alta cohesión.
Problemas para realizar un rápido mantenimiento del sistema	El diseño arquitectónico fue concebido para que el sistema sea de fácil entendimiento. Al existir un bajo acoplamiento entre sus clases se podrá realizar cambios que no afectaran en gran medida a las otras clases.
Problema para garantizar la seguridad de la información y de los datos.	El módulo trabaja con 2 roles, el analista del sistema y el ingeniero de proceso, a los cuales se les crea accesos separados a las operaciones dando cumplimiento a las normas de seguridad.

Tabla 19. Riesgos y decisiones

3.7 Conclusiones parciales

En el capítulo recién concluido se evidencia como los atributos de calidad son elementos importantes para lograr desarrollar una arquitectura sólida. Las características de la norma ISO/IEC 9126 garantizan un elevado grado de calidad en las pruebas realizadas. A partir del uso del método ATAM, se pudo evaluar la arquitectura mediante la identificación de sus riesgos. La detección de pocos riesgos y la posibilidad de que los mismos sean mitigados por las decisiones arquitectónicas tomadas, se concluye que la arquitectura propuesta satisface las necesidades del sistema a desarrollar, por lo que la misma contribuye con la consecución del objetivo de esta investigación.

Conclusiones

Al finalizar la presente propuesta se puede afirmar que:

- Los sistemas ECM no satisfacen las necesidades de la Suite de Ingeniería de Software, por lo que se hace necesaria la creación del Módulo de Gestión de la Información.
- Con el objetivo de realizar el análisis y diseño de dicho módulo, se realizó un estudio sobre varios de los estilos arquitectónicos que existen en la actualidad, para hacer uso de las mejores técnicas de diseño.
- La arquitectura propuesta permitió desagregar al software en un conjunto de componentes de bajo acoplamiento y alta cohesión, permitiendo la reutilización de estos componentes y a su vez garantizando una respuesta más rápida del módulo.
- Se cumplieron todos los objetivos trazados en esta investigación y a la vez se dio respuesta a la pregunta planteada en el problema científico a partir de la definición del marco arquitectónico para el Módulo de Gestión de la Información de la Suite de Ingeniería de Software.

Recomendaciones

Se recomienda:

- Realizar por parte de los desarrolladores del equipo de trabajo, la vista de Proceso.
- Realizar la implementación del Módulo de Gestión de la Información de la Suite de Ingeniería de Software a partir de la arquitectura definida.

Bibliografía

1. **Benitez, Danny Duménico.** Sistemas de información, aplicaciones en empresas. [En línea] 2010. http://www.eumed.net/ce/2012/ddb.html.

2. **Laudon, K, J.P. Laudon.***Sistemas de información gerencial.* 2004.

3. **López-Hermoso, J, A, Montero, S, Martín-Romo, C, De Pablos, V, Izquierdo, J, Nájera.***Informática Aplicada a la Gestión de Empresas.*

4. **Adony González Cárdenas, Zuzel Díaz Pérez.***Sistema Gestor de Trabajos de Diploma.* La Habana : s.n., 2009. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informática.

5. **Solórzano, B.***Instituciones de información, sus perspectivas y oportunidades.* La Habana : s.n., 1999. En INFO'99.

6. **María J Vidal Ledo, Ana B Araña Pérez.** Gestión de la información y el conocimiento. [En línea] 2012. La Habana, Cuba. http://www.ems.sld.cu/index.php/ems/article/view/56/46.

7. **Eíto-Brun, Ricardo.***Gestión de documentos en entornos de desarrollo de software: normativa y aplicabilidad de soluciones open source.* 2008.

8. Proceso de desarrollo de software. [En línea] Departamento de Sistemas Informáticos y Computación.. http://www.google.com.cu/url?sa=t&rct=j&q=def+artefactos+dise%C3%B1o+software&source=web&cd=6&ved=0CEcQFjAF&url=http%3A%2F%2Fwww.dsic.upv.es%2Fasignaturas%2Ffacultad%2F.

9. **Linares, Yoleida Carolina Cámara.***Gestores de Contenido Empresarial de Código Abierto: Comparativa Entre Alfresco Y Nuxeo.* Salamanca : s.n., 2011. Trabajo de fin de máster de la Universidad de Salamanca Facultad de Traducción y Documentación Máster en Sistemas de Información Digital .

10. **Garcia, Andres Eduardo.** [En línea] 16 de 1 de 2012. http://andreseduardogarcia.blogspot.com/2012/01/revisando-gestion-documental-16012012.html.

11. *0113_ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE.* **Monte, Yusleydi Fernández del.** La Habana : s.n., 2012.

12. **Duque, Raúl González.***Python para todos.*

13. **Neptalí, Egdo Alcides.***Estudio de la Arquitectura de Software.* 2010.

14. **David Garlan, Mary Shaw.***An introduction to software architecture.* 1994. CMU Software Engineering Institute Technical Report.

15. **Pressman, Roger S.***Ingeniería de software, un enfoque práctico.* Quinta edición.

16. **César de la Torre, Unai Zorrilla, Miguel A. Ramos, Javier Calvarro.***Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0.* 2010.

17. **Varia, Jinesh.***Arquitectura para la nube: Prácticas recomendadas.* 2010.

18. **Gracia, Joaquín.***Patrones de diseño orientado a objetos.* 2005.

19. **Richard Helm, Ralph Johnson y John Vlissides, Erich Gamma.***Design Patterns: Elements of Reusable Object-Oriented Software.* 1995. ISBN: 0-201-63361-2.

20. **Larman, Craig.***UML y patrones, introducción al análisis y diseño orientado a objetos.* 1999.

21. **Osvaldo Díaz Verdecia, Virgen Damaris Quevedo Campins.***Una guía práctica de Arquitectura de Software.* La Habana : s.n., 2009. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas..

22. **Annie Leyva Simón, Devora L. Covas Ramos.***Manual del Arquitecto de Software.* La Habana : s.n., 2009. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

23. [En línea] http://www.forest.ula.ve/~mana/cursos/redes/protocolos.html.

24. **Gimson, Lic. Loraine.***Metodologías ágiles y desarrollo basado en conocimiento.* 2012. Trabajo final integrador presentado para obtener el grado de Especialista en Ingeniería de Software, Universidad Nacional de la Plata Facultad de Informática.

25. Infraestructura Tecnológica de Software. [En línea] http://www.funiber.org/areas-de-conocimiento/tecnologias-de-la-informacion/infraestructura-tecnologica-de-software/.

26. *Manual relatorio para reportes.* 2008.

27. Riverbank. [En línea] http://www.riverbankcomputing.co.uk./news.

28. Ecured. [En línea] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

29. Ecured. [En línea] http://www.ecured.cu/index.php/Visual_Paradigm.

30. **Reidiel Castillo Arbelo, Pablo Soria Acosta.***Herramienta para la Migración y Administración de Servidores (HMAS).* La Habana : s.n., 2012. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

31. **Rosa, Eugenio Rosales.***Módulo para la administración de NAS en Nova para servidores.* La Habana : s.n., 2012. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

32. **Díaz, Ana Evis Echemendía.***Expediente de proyecto del Módulo de Gestión de la Información de la Suite de Ingeniería de Software.* La Habana : s.n., 2013. Proyecto Nova QALIT, Centro CESOL, Universidad de las Ciencias Informáticas.

Bibliografía consultada

1. **Benitez, Danny Duménico.** Sistemas de información, aplicaciones en empresas. [En línea] 2010. http://www.eumed.net/ce/2012/ddb.html.

2. **Laudon, K, J.P. Laudon.***Sistemas de información gerencial.* 2004.

3. **López-Hermoso, J, A, Montero, S, Martín-Romo, C, De Pablos, V, Izquierdo, J, Nájera.***Informática Aplicada a la Gestión de Empresas.*

4. **Adony González Cárdenas, Zuzel Díaz Pérez.***Sistema Gestor de Trabajos de Diploma.* La Habana : s.n., 2009. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informática.

5. **Solórzano, B.***Instituciones de información, sus perspectivas y oportunidades.* La Habana : s.n., 1999. En INFO'99.

6. **María J Vidal Ledo, Ana B Araña Pérez.** Gestión de la información y el conocimiento. [En línea] 2012. La Habana, Cuba. http://www.ems.sld.cu/index.php/ems/article/view/56/46.

7. **Eíto-Brun, Ricardo.***Gestión de documentos en entornos de desarrollo de software: normativa y aplicabilidad de soluciones open source.* 2008.

8. Proceso de desarrollo de software. [En línea] Departamento de Sistemas Informáticos y Computación.. http://www.google.com.cu/url?sa=t&rct=j&q=def+artefactos+dise%C3%B1o+software&source=web&cd=6&ved=0CEcQFjAF&url=http%3A%2F%2Fwww.dsic.upv.es%2Fasignaturas%2Ffacultad%2F.

9. **Linares, Yoleida Carolina Cámara.***Gestores de Contenido Empresarial de Código Abierto: Comparativa Entre Alfresco Y Nuxeo.* Salamanca : s.n., 2011. Trabajo de fin de máster de la Universidad de Salamanca Facultad de Traducción y Documentación Máster en Sistemas de Información Digital .

10. **Garcia, Andres Eduardo.** [En línea] 16 de 1 de 2012. http://andreseduardogarcia.blogspot.com/2012/01/revisando-gestion-documental-16012012.html.

11. *0113_ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE.* **Monte, Yusleydi Fernández del.** La Habana : s.n., 2012.

12. **Duque, Raúl González.***Python para todos.*

13. **Neptalí, Egdo Alcides.***Estudio de la Arquitectura de Software.* 2010.

14. **David Garlan, Mary Shaw.***An introduction to software architecture.* 1994. CMU Software Engineering Institute Technical Report.

15. **Pressman, Roger S.***Ingeniería de software, un enfoque práctico.* Quinta edición.

16. **César de la Torre, Unai Zorrilla, Miguel A. Ramos, Javier Calvarro.***Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0.* 2010.

17. **Varia, Jinesh.***Arquitectura para la nube: Prácticas recomendadas.* 2010.

18. **Gracia, Joaquín.***Patrones de diseño orientado a objetos.* 2005.

19. **Richard Helm, Ralph Johnson y John Vlissides, Erich Gamma.***Design Patterns: Elements of Reusable Object-Oriented Software.* 1995. ISBN: 0-201-63361-2.

20. **Larman, Craig.***UML y patrones, introducción al análisis y diseño orientado a objetos.* 1999.

21. **Osvaldo Díaz Verdecia, Virgen Damaris Quevedo Campins.***Una guía práctica de Arquitectura de Software.* La Habana : s.n., 2009. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas..

22. **Annie Leyva Simón, Devora L. Covas Ramos.***Manual del Arquitecto de Software.* La Habana : s.n., 2009. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

23. [En línea] http://www.forest.ula.ve/~mana/cursos/redes/protocolos.html.

24. **Gimson, Lic. Loraine.***Metodologías ágiles y desarrollo basado en conocimiento.* 2012. Trabajo final integrador presentado para obtener el grado de Especialista en Ingeniería de Software, Universidad Nacional de la Plata Facultad de Informática.

25. Infraestructura Tecnológica de Software. [En línea] http://www.funiber.org/areas-de-conocimiento/tecnologias-de-la-informacion/infraestructura-tecnologica-de-software/.

26. *Manual relatorio para reportes.* 2008.

27. Riverbank. [En línea] http://www.riverbankcomputing.co.uk./news.

28. Ecured. [En línea] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

29. Ecured. [En línea] http://www.ecured.cu/index.php/Visual_Paradigm.

30. **Reidiel Castillo Arbelo, Pablo Soria Acosta.***Herramienta para la Migración y Administración de Servidores (HMAS).* La Habana : s.n., 2012. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

31. **Rosa, Eugenio Rosales.***Módulo para la administración de NAS en Nova para servidores.* La Habana : s.n., 2012. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

32. **Díaz, Ana Evis Echemendía.***Expediente de proyecto del Módulo de Gestión de la Información de la Suite de Ingeniería de Software.* La Habana : s.n., 2013. Proyecto Nova QALIT, Centro CESOL, Universidad de las Ciencias Informáticas.

33. Augeas — Main. [En línea] http://augeas.net/.

34. **B. Capote Marrero, D. González Machín.***La gestión de información como herramienta fundamental en el desarrollo de los centros toxicológicos.* La Habana : s.n., 2003. Centro Nacional de Toxicología (CENATOX).

35. **Kruchten, Philippe.***Planos Arquitectónicos: El Modelo de "4+1" Vistas de laArquitectura del Software.* 2006.

36. **Urdaneta, Páez.***Gestión de la inteligencia, aprendizaje tecnológico y modernización del trabajo internacional. Retos y oportunidades.* Caracas : s.n., 1992. Instituto de Estudios del Conocimiento de la Universidad Simón Bolívar.

37. **Y. Pérez Rodríguez, A. Coutín Domínguez.** La gestión del conocimiento: un nuevo enfoque en la gestión empresarial. [En línea] 2005. http://bvs.sld.cu/revistas/aci/vol13_6_05/aci040605.htm.

Glosario de términos

CASE: Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costos de las mismas en términos de tiempo y de dinero.

CMMI: Integración de Modelos de Madurez de Capacidades o Capability Maturity Model Integration (CMMI) es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

Calidad: La palabra calidad tiene múltiples significados. La calidad de un producto o servicio es la percepción que el cliente tiene del mismo. Es una fijación mental del consumidor que asume conformidad con un producto o servicio determinado, que solo permanece hasta el punto de necesitar nuevas especificaciones. La calidad es un conjunto de propiedades inherentes a un objeto que le confieren capacidad para satisfacer necesidades implícitas o explícitas.

Estándar: Los estándares requieren ser establecidos con el fin de contar con una referencia que permita identificar oportunamente las variaciones presentadas en el desarrollo de los procesos y aplicar las medidas correctivas necesarias. Es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad.

Framework: en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Conjunto integrado de componentes que colaboran para proporcionar una arquitectura reutilizable para una familia de aplicaciones. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Herramienta: Son los ambientes de apoyo necesario para automatizar las prácticas de Ingeniería de Requisitos.

IDE: Entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE'). Es un programa compuesto por un conjunto de herramientas para un programador desde el que se pueden editar programas, compilarlos y depurarlos.

IEEE: Corresponde a las siglas de (Institute of Electrical and Electronics Engineers) en español Instituto de Ingenieros Electricistas y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización.

Instalar: incorporar a la computadora un programa o dispositivo para ser utilizado.

Módulo: Parte de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivo, un módulo realiza una o varias de dichas tareas.

Proceso: Secuencia de actividades que tienen un marcado inicio y fin.