VNIVERSIDAD NACIONAL
AVFNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"Análisis e implementación de esquemas de seguridad aplicados a la herramienta integral MoProSoft (HIM)"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA
(COMPUTACIÓN)

P R E S E N T A:

JORGE CRUZ VÁZQUEZ

DIRECTOR DE TESIS: DRA. HANNA OKTABA

México, D.F. 2005.

Agradecimientos

A mi padre Dr. Angel Cruz Espinosa, por preocuparse por mí y apoyarme incondicionalmente, gracias a ti he podido conseguir las metas que me he trazado y te debo lo que soy. Gracias papá.

A mi madre Mtra. Gloria Vázquez Gómez, por creer en mi, apoyarme incondicionalmente, brindándome cariño y amor. Gracias a ti he llegado hasta donde estoy sin perderme del camino correcto. Gracias mamá.

A mi herman: Dr. Angel Cruz Vázquez, por que me haz ayudado pese a lo distinto de mis sueños. Gracias por no abandonarme.

A mi hermana: Dra. Vianney Cruz Vázquez, por que me haz apoyado en cada una de las ideas que tengo en mente, ayudarme cada vez que lo necesité. Gracias hermana por entender lo que soy.

A mi hermana: Dra. Magaly Cruz Vázquez, por entender mi forma de pensar y darme la ayuda que necesitaba en los momentos de mi formación académica.

A mi novia: Dra. Claudia Torres Romero, gracias por escucharme y aconsejarme por el buen camino, gracias por todo el amor que me das.

A mi tutora de tesis: Dra. Hanna Oktaba, por guiarme por el camino adecuado en la realización de la tesis y dedicarme tiempo en su revisión. Además de confiar en mi trabajo y la entrega al mismo.

Índice General

Análisis y diseño para la seguridad HIM

Implementación de los mecanismos de seguridad

Introducción

La realidad actual de las industrias de software tanto en nuestro país como a nivel mundial, se enfrenta con obstáculos y problemáticas que pocas veces son mitigadas o resueltas de la mejor manera, en el momento de realizar la construcción de sistemas de software eficientes y que cubran completamente las necesidades del cliente.

Muchas de las empresas ya sean de pequeña, media o gran escala no tienen bien definido un modelo de procesos que les permita llevar el control y la administración de sus proyectos de software para poder llegar a un producto exitoso y de calidad. La importancia del conocimiento de procesos de desarrollo por parte de las empresas en ocasiones es nula o mal aplicada.

Es de suma importancia que las industrias de software en México sean concientes de la importancia de adoptar un Modelo de Procesos y de la aplicación correcta del mismo.

Por tales motivos en nuestro país se dio por parte de la Secretaria de Economía y la Universidad Nacional Autónoma de México (UNAM), la iniciativa de la creación de un Modelo de Procesos MoProSoft enfocada a las industrias de software mexicanas de pequeña y mediana escala, con el fin de que las mismas pudieran adoptarlo en caso de que no contaran con alguno o simplemente adoptar aquellos procesos que considere necesarios para su mejor funcionamiento.

La suma de esfuerzos dio como origen el Modelo de Procesos MoProSoft[1]. MoProSoft tiene como objetivo ser adoptado por las empresas de software mexicanas y con ello mejorar la calidad en los procesos de construcción de software.

Con el fin de que las industrias de software pudieran realizar un mejor manejo del Modelo de Procesos, se realizó una herramienta de software nombrada como Herramienta Integral MoProSoft, la cual permite realizar la gestión de los proyectos de software con que cuenten además de los propios productos que surgen durante el ciclo de desarrollo.

Para poder realizar la elaboración del sistema HIM se conformó un equipo de trabajo de 6 alumnos del posgrado, todos los integrantes del equipo participamos en las etapas de análisis, diseño e implementación. Cada elemento del equipo de trabajo adoptó distintos roles dentro de la construcción del sistema que nos permitiera llevar un control adecuado.

---

[1] Modelo de Procesos de Software [DR], Derechos reservados, Secretaría de Economía, México. http://software.net.mx

La selección del área en que se enfocaría cada miembro del equipo de trabajo fue definido según las preferencias de cada uno. Las tesis relacionadas que conforman en su totalidad el proyecto de HIM, son las que a continuación se mencionan:

Integrante	Tema de Tesis
Karín Valdivieso Castillo	Utilización de patrones y la arquitectura J2EE para el diseño de la interfaz de usuario de la herramienta Integral MoProSoft (HIM)
Jorge Cruz Vázquez	**Análisis e implementación de esquemas de seguridad aplicados a la herramienta integral MoProSoft (HIM)**
Marcos Oscar Vázquez Morales	Aplicación de patrones basados en J2EE para el diseño e implementación de la capa de control de la Herramienta Integral para MoProSoft (HIM)
Araceli Eugenia Mercado Fernández	La Sincronización de los Elementos de una Base de Conocimiento para MoProSoft y su Aplicación en una Herramienta Integral
Ernesto Hernández Uribe	Uso de la tecnología RDF para representar y manejar los procesos MoProSoft y su aplicación en HIM.
Hafiz Zurita Rendón	Arquitectura de la Herramienta Integral para MoProSoft

La presente tesis que como título lleva: "Análisis e implementación de esquemas de seguridad aplicados a la Herramienta Integral MoProSoft (HIM)", describe aquellos aspectos de seguridad que fueron tomados en consideración para la construcción de la herramienta.

Cabe señalar que como todo sistema de software que se desarrolla debe realizar en la etapa de definición de requerimiento y análisis, la identificación de aquellos rubros relacionadas a las seguridad que deban ser contemplados con el fin de contar con un sistema confiable y seguro. Los aspectos de seguridad pueden definirse con base a los requerimientos de negocio, ambiente del sistema, interacción con otros sistemas, políticas de seguridad, integridad de la información, etc.

Debe darse la ponderación adecuada a los requerimientos de seguridad que demande la construcción de un sistema y no dejar en etapas finales la adopción de mecanismos de seguridad mínimos para que nuestro sistema pueda ser considerado como seguro.

Los objetivos de la tesis son:

- Tener un mecanismo de Autenticación
- Realizar un control de permisos o hablando en terminología de negocio, que el rol pueda realizar únicamente las actividades que le son conferidas.
- Tener un registro de bitácora para posteriores auditorias.
- Revisar, analizar y seleccionar los patrones GoF (Gang of Four) o J2EE que puedan resolver los problemas de seguridad.
- Implementar los patrones seleccionados para atender los requerimientos de seguridad
- Integrar los componentes de seguridad dentro de la Arquitectura definida para HIM.

Descripción de Capítulos

La presente tesis está conformada por 5 capítulos los cuales de describen a continuación:

1. Problemas de seguridad en sistemas informáticos y sus relaciones.- Se describen principalmente las causas que originan que un sistema informático sea inseguro, además de las formas de ataques que pueden presentarse sobre nuestros sistemas de software.

2. Antecedentes para HIM.- Dentro de este capítulo se realiza la revisión de la estructura en que se conformo MoProSoft además de los requerimientos de seguridad que contempla la herramienta HIM.

3. Conceptos de Mecanismos de seguridad.- Se describen los fundamentos teóricos para los mecanismos de seguridad, control de acceso, control de permisos y registro de bitácora.

4. Análisis y Diseño para la seguridad HIM.- Contempla las etapas de análisis y diseño, mostrando aquellos productos generados para su explicación como lo son diagramas de clases y diseño a nivel del análisis y diseño.

5. Implementación de los mecanismos de seguridad.- Principalmente describe los componentes físicos que se tomaron como punto de partida para realizar la implementación e integración del sistema HIM.

Capítulo 1

Problemas de seguridad en sistemas informáticos y sus relaciones.

En este capítulo se describirá los tipos de ataques que puede sufrir todo sistema informático, así como las causas que los originan.

Antecedentes

Conforme evolucionan los medios electrónicos por los cuales nos comunicamos, aumenta en igual proporción y velocidad la inseguridad a través de ellos. Con el advenimiento del Internet surgieron distintos mecanismos de ataques sobre la integridad de la información, denegación de servicios, robo de información, etc. Algunos de estos ataques con fines de lucro o únicamente por demostrar las habilidades del intruso.

Todo esto surge o es alimentado por la falta de medidas de seguridad adecuadas en los sistemas informáticos, desconocimiento por parte de puestos estratégicos dentro de una organización. Para poder mitigar tales adversidades debemos estar concientes sobre las políticas de seguridad que debemos imponer dentro de nuestra organización y los requerimientos adecuados para nuestro sistema de computó. No podemos demostrar que un sistema de cómputo es 100% seguro, pero si adoptamos y aplicamos las medidas requeridas en los sistemas de cómputo podremos disminuir drásticamente las vulnerabilidades que pueda aprovechar el atacante.

De todo lo anterior existe la necesidad latente de conocer los conceptos básicos sobre la seguridad, los mecanismos existentes y hacia donde se enfoca cada uno de ellos. Conociendo las bases teóricas podremos aplicarlos particularmente a nuestras necesidades, teniendo así un sistema confiable y seguro.

1.1 Causas de inseguridad en un sistema informático

Debido al incremento constante de ataques que reciben a diario los usuarios de sistemas informáticos, ha cobrado interés el conocer cómo protegerse de tales ataques a sus sistemas.

Ahora los usuarios no solo deberán de preocuparse o tomar interés en conocer a fondo sus sistemas, para lograr un control máximo del mismo. Además deberán conocer los esquemas de seguridad que les permitan proteger la información contra robo o pérdida.

Otro factor que influyó fuertemente en el incremento de incidencias de seguridad sobre equipos de cómputo fue la aparición del Internet. Haciendo posible que se interconectaran miles de equipos a nivel mundial, lo cual hizo que aparecieran nuevos riesgos y mecanismos de ataque a sistemas de cómputo.

Para poder entender la problemática de la seguridad, que involucra a los sistemas informáticos, se debe entender algunos conceptos mínimos, que se definen a continuación:

- **Vulnerabilidad**.- Es todo aquello que afecta a un sistema informático debido a una mala configuración, instalación o mala codificación, etc.

- **Criptoanálisis**.- Estudio de técnicas matemáticas con el fin de encontrar vulnerabilidades a los sistemas informáticos.

- **Atacante, Intruso**.- Es la persona que hace uso del criptoanálisis, con el fin de hacer uso de las vulnerabilidades que presente el sistema para infiltrase en él. Una clasificación común que se le da a un intruso es la siguiente:

- *Hacker*.- Es aquella persona que explora los detalles de un sistema informático con el fin de poner a prueba sus capacidades.

- *Cracker*.- Es aquella persona que corrompe la seguridad de un sistema informático con un fin malicioso o de lucro.

- *Preacker*.- Es aquella persona que realiza *cracking* sobre la red telefónica, con el fin de realizar llamadas de larga distancia gratuitamente.

- *Lamer o Script Kiddies*.- Es catalogado a nivel intelectual como la de más bajo nivel, ya que no sabe programar y sólo hace uso de herramientas preconfiguradas para atacar a un sistema informático.

1.2 Formas de ataques a un sistema informático.

Existen diversos mecanismos de ataque que puede hacer uso el intruso para poder romper los mecanismos de seguridad con que cuente nuestro sistema, entre los cuales encontramos la siguiente clasificación: [MED98]

i. **Virus**.- El virus es un fragmento de código que se inserta en un programa ejecutable, de modo que cuando el programa es ejecutado, se ejecuta también el virus. La función que un virus pretende realizar es propagarse a sí mismo por todo el sistema, infectando a otros programas en los que inserta el fragmento de código malicioso.

ii. **Gusanos**.- Son programas que se reproducen copiándose de un equipo de cómputo a otro a través de la red. A diferencia de los virus, los gusanos son programas independientes y no requieren de otro programa en donde alojarse. Por lo general, los gusanos no producen ningún tipo de daño en el sistema, excepto malgastar los recursos del sistema, llegando incluso a sobrecargar la red.

iii. **Caballos de Troya**.- El Caballo de Troya es un programa, que imita la ejecución de otros programas, pero que realiza otras funciones completamente diferentes a la esperada. Normalmente causan daños irreversibles como lo puede ser el borrado de grupos de archivos del sistema local.

Existe otro tipo de Caballo de Troya cuya finalidad es comprometer la seguridad del sistema. Normalmente se ocultan bajo el nombre de un archivo del sistema y consiste en reescribir el programa como puede ser el caso de un programa de *login*.

iv. **Bombas de tiempo.**- Las bombas son muy parecidas a un Caballo de Troya en funcionamiento, pero la diferencia recae en que las bombas se activan por causa de un evento, como lo puede ser una fecha determinada.

v. **Bacterias.**- Son aquellos programas que son ejecutados en un sistema y que tiene como única finalidad el reproducirse sin parar, alcanzando su objetivo de hacer uso de todos los recursos del sistema e, incluso, llegar a bloquear al mismo.

vi. **Puertas traseras.**- Las puertas traseras son mecanismos implantados en los programas, que permiten realizar acciones determinadas sin tener que pasar por todas las secciones de código del programa, como lo puede ser el mecanismo de autenticación, el cual podrá ser burlado.

En muchas ocasiones las **puertas traseras** son producto de una mala programación o, en ocasiones, son colocadas intencionalmente por el mismo programador, con el fin de conseguir un acceso posterior y fácil al sistema.

vii. **Negación de Servicio** (*Denial of Service*[2]).- Este tipo de ataque tiene como objetivo el reducir o eliminar la capacidad de un servidor, evitando o denigrando el servicio a los usuarios.

viii. ***SPAM.***- Es la práctica de enviar indiscriminadamente mensajes de correo electrónico no deseados. Estos mensajes tienen como objetivo realizar la publicidad de productos, servicios o de alguna página Web.

1.3 Amenazas y contramedidas a la seguridad

A continuación se definen conceptos importantes de amenazas que puede sufrir la seguridad de un sistema de cómputo, tales como las vulnerabilidades y ataques. En su contraparte existen contramedidas que deben ser tomadas en consideración para mitigar todo tipo de amenaza.

A continuación se explica cada una de ellas con mayor detalle.

1.3.1 Vulnerabilidades

Todo sistema informático es vulnerable a un ataque, por lo que se debe de tomar las medidas necesarias con el fin de disminuir toda vulnerabilidad que pueda presentar nuestro sistema.

[2] DOS, abreviación comúnmente utilizada por las siglas en ingles.

A continuación se presenta una clasificación de los tipos de vulnerabilidades que existen de acuerdo al factor humano, tecnológico o del ambiente natural que pueda influir en un riesgo alto de inseguridad.

Físico.- Este aspecto se refiere a la inseguridad que pueda presentar el espacio de trabajo donde se realizan las actividades cotidianas, donde puede acceder una persona no autorizada y robar información de suma importancia para la organización.

Natural.- Todo equipo de cómputo es susceptible a una amenaza de un factor ambiental, como por ejemplo un incendio en el edificio, un temblor que destruya las instalaciones, una sobrecarga de corriente que dañe el equipo. Todos estos factores se encuentran fuera del alcance del ser humano, pero se debe tomar las medidas adecuadas para disminuir riesgos ante eventos inesperados.

Hardware y Software.- Hasta el momento hemos analizado los tipos de problemas de seguridad que puede presentar un software, pero existe la posibilidad de que un componente de hardware pueda fallar o ser dañado por factores externos. Comprometiendo al equipo y ocasionar la perdida de información.

Emanación.- Todos los equipos eléctricos emiten radiaciones eléctricas y magnéticas. Este tipo de radiaciones pueden ser interceptadas y posteriormente interpretadas por lo que presenta una vulnerabilidad latente.

Comunicación.- Todo equipo de cómputo conectado a la red tiene el riesgo de ser atacado por un intruso, ya que todo el flujo de información que viaje hacia y desde ella puede ser interceptado.

Humano.- Todo ser humano está propenso a cometer errores, pero la persona que menos errores debería cometer es la que se encarga de la administración de un sistema. El administrador debe de ser una persona capacitada ya que en ella recae la responsabilidad de mantener el sistema con mecanismos de seguridad adecuados y libres de posibles vulnerabilidades.

1.3.2 Amenazas

Existen tres clasificaciones de amenazas que puede sufrir un sistema informático entre las que encontramos:

Amenaza Natural y Física.- Son todos aquellos fenómenos naturales que ponen en riesgo el equipo de cómputo, por lo que no podemos prevenir tales vulnerabilidades naturales. Lo único que podemos hacer es disminuir tales factores para que sean los menos severos posibles, evitando todo riesgo de incendio dentro del área de trabajo.

Amenaza no intencionada.- Un administrador de un equipo de cómputo puede no tener nociones de seguridad y de la importancia que implica mantener un sistema estable y libre de vulnerabilidades, que pueda dar paso a que un atacante se infiltre en el sistema.

Amenaza intencionada.- Las amenazas intencionadas pueden provenir de una fuente interna a la organización o externa a ella. Puede ser un empleado que horas antes fue despedido por la empresa y con el afán de desquitarse con la misma, puede tratar de efectuar un ataque al sistema o vender información a la competencia.

Las amenazas externas pueden provenir de un atacante que tiene como fin un lucro o solo demostrar su capacidad intelectual, para librar las barreras de seguridad que presenta el sistema de cómputo de la organización.

1.3.3. Contramedidas

Existen diversos tipos de contramedidas con el fin de proteger los equipos de cómputo y la información que se genera en una organización y es almacenada en un medio de almacenamiento digital.

Seguridad en Cómputo.- En términos generales se enfoca a la protección de la información que se encuentra almacenada en el sistema de cómputo.

Además de llevar un control de quien puede acceder al sistema y la interacción que es permitida al usuario. Otros aspectos importantes son el manejo de contraseñas, la auditoria y los procesos administrativos que se deben de realizar, como lo son el respaldo de la información.

Seguridad en la Comunicación.- Este rubro abarca las medidas necesarias que se requieren cuando la información es transmitida a través de la red. Un método adecuado para proteger a través de un medio inseguro, como lo es la Internet, es la criptografía.

Seguridad Física.- La seguridad física es aquella que se refiere a la protección del equipo de cómputo contra daños naturales, de hardware y contra intrusos. Tecnologías, como el control de acceso a las instalaciones por medio de huellas digitales o dispositivos biométricos, pueden ser tomadas en cuenta como medidas preventivas, aunque su costo es alto.

1.4 Obtener un nivel adecuado de seguridad en nuestros sistemas informáticos.

Debemos darnos cuenta de la importancia y relevancia que juega la seguridad en un sistema de cómputo, por lo que es importante considerar aquellos aspectos que involucren a la seguridad de nuestro sistema. Para conseguir un nivel de seguridad aceptable se deben tomar en cuenta los siguientes puntos:

- **Privacidad.-** El término de privacidad tiene como fin el tratar de evitar que la información alojada en un sistema informático sea accesible únicamente al personal autorizado.

- **Integridad.-** Este concepto tiene como fin el proteger la información, ya sean datos o programas, contra posibles modificaciones o borrado por parte de los usuarios, que no sean los propietarios de dicha información.

- **Disponibilidad.-** Se entenderá por disponibilidad al estado que presenta un sistema informático, el sistema se debe encontrar disponible siempre que el usuario lo demande para su uso. Por lo que se debe evitar cualquier tipo de interrupción del sistema, así como la degradación de los servicios que este proporcione.

- **Consistencia.-** El término de consistencia esta definido en función del comportamiento del sistema informático, es decir, que presente una repuesta de lo que se espera que haga el sistema. Por lo que se debe de evitar todo tipo de fallos de software y hardware.

- **Aislamiento.-** En el caso de que un sistema se encuentre comprometido a causa del acceso no autorizado o comportamiento irregular del sistema, se deberá mantener en aislamiento el sistema con el fin de averiguar la causa de anomalía que presenta el sistema.

- **Auditoria.-** Se debe contar con un registro interno del sistema con el objetivo de mantener un historial de las actividades que realizan los usuarios. Que usuario lo realiza y que se ha afectado, todo esto a causa de conocer si un usuario no autorizado ingresó al sistema y realizo actividades no permitidas.

En este capítulo se ha podido discutir la problemática que involucra a un sistema informático, así como las posibles causas y motivos que originan que un sistema sea atacado.

El fin principal, que encausa este trabajo, es el diseñar e implementar una herramienta de software libre que dé soporte al modelo de procesos para el software en México, mejor conocido como **MoProSoft**[3]**©.** El cual es el modelo de procesos pensado para la Industria de Desarrollo y Mantenimiento de Software, la pequeña y mediana industria.

La herramienta nombrada como **HIM**[4], tiene como objetivo principal el facilitar y encausar el uso del modelo de procesos **MoProSoft,** ayudando así a las industrias de software en México la adopción de este modelo para la mejora en sus procesos internos, y por consiguiente la calidad del software que producen.

Por tal motivo para la construcción de la herramienta integral **MoProSoft**, se debe considerar en las etapas de análisis y diseño, aquellos aspectos de seguridad que demande la herramienta, tomando en consideración el modelo de negocio de la organización.

El siguiente capítulo se describe los aspectos generales que conforma MoProSoft, así como los esquemas mínimos de seguridad propuestos para la herramienta integral, tales como: control de acceso, control de permisos a usuario y un registro de bitácora.

[3] Modelo de Procesos de Software [DR], Derechos reservados, Secretaría de Economía, México. http://software.net.mx

[4] Herramienta Integral MoProSoft

Capítulo 2

Antecedentes para HIM.

En este capítulo se describe aspectos generales que conforman al modelo de procesos MoProSoft. Además de los requerimientos en aspectos de seguridad que contempla la herramienta integral MoProSoft.

2.1 MoProSoft.

MoProSoft es un modelo de procesos de software, dirigido a la pequeña y mediana industria en México. Con el objetivo de estandarizar las mejores prácticas dentro del área de gestión e ingeniería de software.

Dentro de las características que presenta este modelo tenemos que:

- Es fácil de entender.
- Es fácil de aplicar.
- Es flexible, es decir, puede ser adoptado parcial o totalmente.
- Puede ser base a evaluaciones exitosas con otros modelos, como CMM.
- Sirve como punto de referencia para la identificación de la ausencia de algún proceso dentro de una organización.

Como se puede distinguir MoProSoft es un modelo orientado a los procesos, con el objetivo de fomentar las mejores prácticas dentro de una organización. Así también a la evaluación periódica de los procesos y actividades, que culminan en la mejora continua de los procesos.

MoProSoft considera la estructura de una organización para definir los 3 niveles básicos que conforma este modelo. Estos 3 niveles están definidos por: *Alta dirección, Gestión y Operación.*

2.2 Estructura de Modelo de Procesos MoProSoft.

MoProSoft define tres categorías principales tomando como base la estructura de una organización. Donde cada una de ellas se encuentra conformada por procesos y cada proceso tiene asignado un rol responsable para llevar sus actividades.

2.2.1. Categoría de Procesos

En la figura 1 se esquematiza los tres principales categorías que define MoProSoft, así como los procesos que conforma a cada categoría. A continuación se da una breve explicación de cada categoría.

La categoría de **Alta Dirección,** conocida con el acrónimo de *DIR*, tiene como función el proporcionar los lineamientos a los procesos relacionados con la Gestión de Negocio. Además de retroalimentarse con la información que generan sus propios procesos.

La categoría de **Gestión**, conocida con el acrónimo de *GES*, toma como punto de partida los lineamientos establecidos en la categoría de Alta Dirección, para realizar prácticas de gestión de procesos, proyectos y recursos. Además de proporcionar los elementos necesarios para su funcionamiento, a la categoría de Operación.

La categoría de **Operación**, mejor conocida por el acrónimo de *OPE*, toma como punto de partida los lineamientos establecidos por la categoría de Gestión, para realizar prácticas de administración, desarrollo y mantenimiento de software.

La comunicación permanente y transparente existe entre las distintas categorías, por ejemplo, la categoría de Alta Dirección debe estar en comunicación constante con la categoría de Gestión. De igual forma la categoría de Gestión deberá tener una comunicación permanente con la categoría de Operación.

La relación constante entre las categorías promueve la prevención de factores de riesgos que surjan dentro de los procesos. Además de una evaluación y mejora en la realización de los mismos.

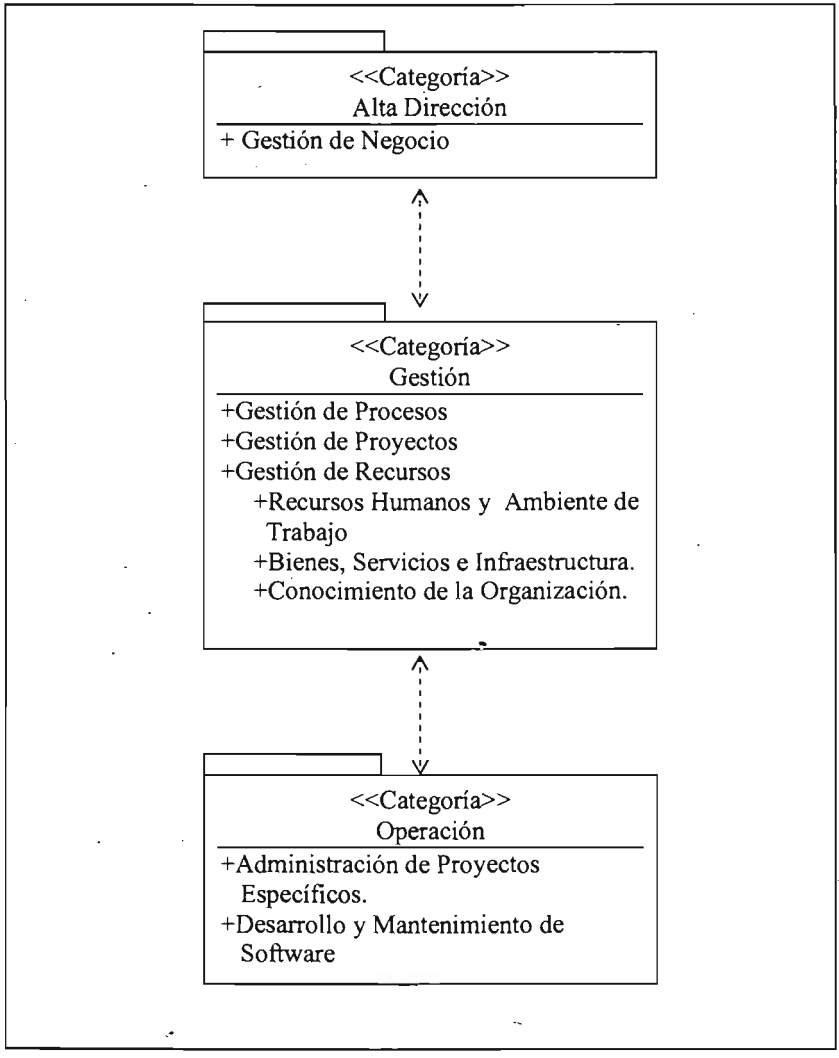


Figura 1. Diagrama de categorías de procesos

2.2.2 Procesos

Previamente definimos las tres principales categorías que define MoProSoft, ahora se presenta brevemente cada uno de los procesos que forman parte de la categoría de Alta Dirección, Gestión y Operación. [MoProSoft]

- **Gestión de Negocio DIR1.-** Establece la razón de ser de la organización, objetivos y condiciones para lograrlos. Además de evaluar los resultados, permitiendo así una mejora continúa.

- **Gestión de Procesos GES1.-** Establece los procesos de la organización, así como define, planea, e implementa las actividades de mejora.

- **Gestión de Proyectos GES2.-** Se asegura que los proyectos cumplan con los objetivos y estrategias de la organización.

- **Gestión de Recursos GES3.-** Consigue y dota a la organización de los recursos humanos, infraestructura, ambiente de trabajo y proveedores. Además de crear y mantener la base de conocimiento de la organización.

    - **Recursos Humanos y Ambiente de Trabajo GES3.1.-** Proporciona los recursos humanos adecuados para cumplir las responsabilidades asignadas a los roles dentro de la organización.

    - **Bienes, Servicio e Infraestructura GES3.2.-** Proporciona los proveedores de bienes, servicios e infraestructura, para las necesidades de los procesos y proyectos.

    - **Conocimiento de la Organización GES3.**3.- Mantiene disponible y administra la base de conocimiento, la cual contiene la información y productos generados por la organización.

- **Administración de Proyectos Específicos OPE1.-** Establece y lleva acabo las actividades que permitan cumplir los objetivos de un proyecto.

- **Desarrollo y Mantenimiento de Software OPE2.-** Es la realización de actividades como: análisis, diseño, construcción, integración y pruebas.

En la figura 2 podemos ver la interrelación y jerarquía que existe entre los procesos de las diferentes categorías, además de la dependencia que hay entre procesos y subprocesos en la gestión de recursos.

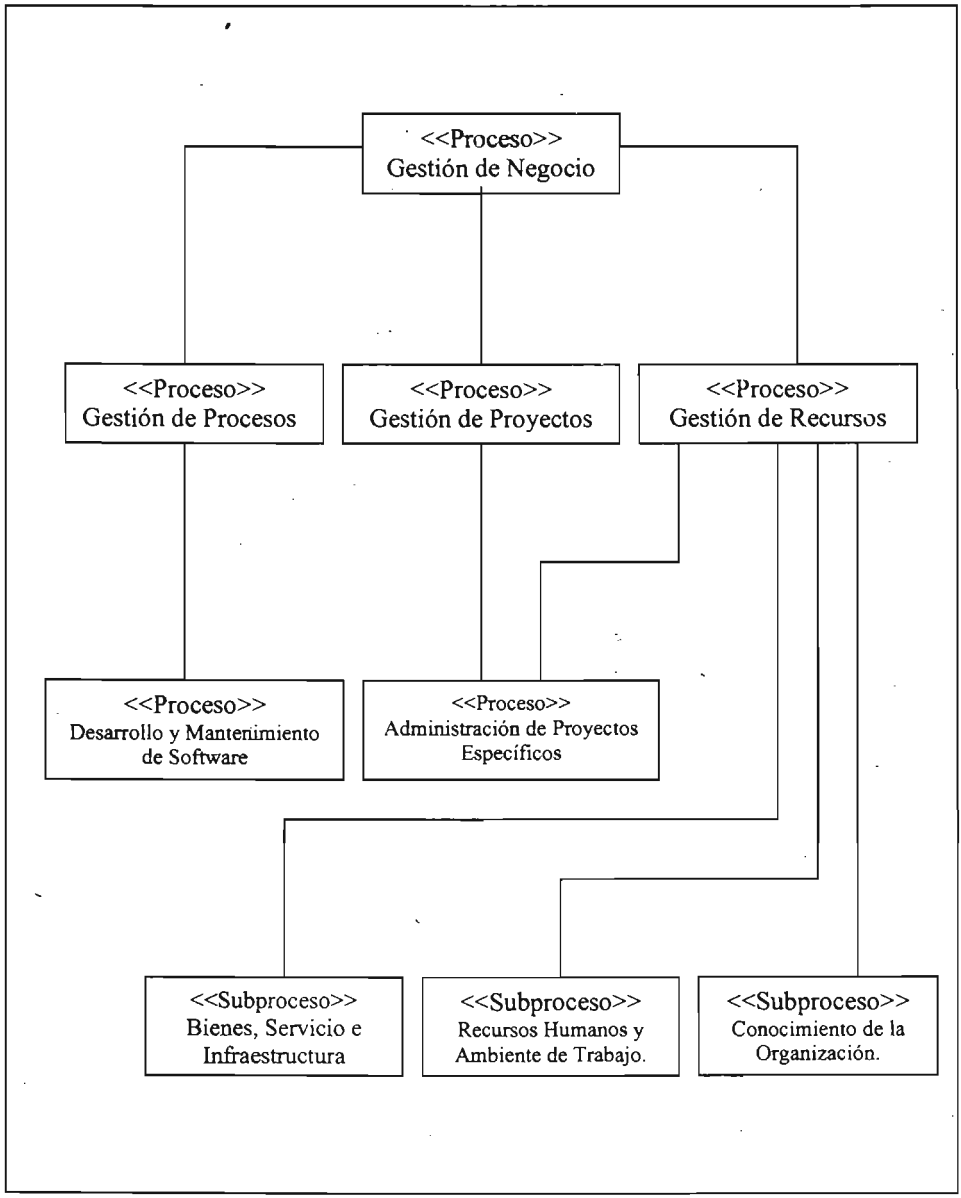


Figura 2.Diagrama de relación entre procesos.

2.2.3 Roles.

Existe una clasificación de los tipos de roles que puede existir dentro del modelo de procesos MoProSoft. Recordemos que un rol es el responsable de un conjunto de actividades de uno o más procesos. Además un rol puede ser asumido por una o más personas en tiempo parcial o completo.

En la figura 3 se representa los diferentes tipos de roles que pueden existir dentro de MoProSoft.

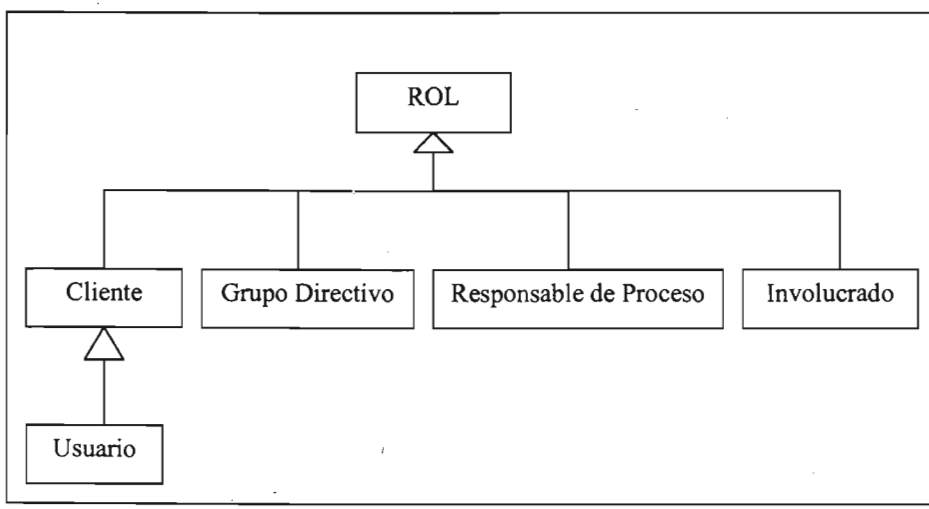


Figura 3. Clasificación general de roles.

El concepto de rol es de suma importancia para la seguridad en nuestro sistema HIM. Con base en el rol asignado a un usuario podrá realizar únicamente las actividades que corresponda a este rol. Por tal motivo debemos implementar un mecanismo de control de permisos, el cual refleje las reglas de negocio definidas en MoProSoft, donde estrictamente muestra que actividades puede realizar cada uno de los roles.

2.2.4 Productos.

Un producto dentro de MoProSoft, es cualquier elemento que se genera en un proceso. Dentro del flujo de trabajo que se realiza dentro de cada proceso se van generando distintos tipos de productos. Un producto puede ser de entrada, salida o corresponder a un producto interno que se genera en el proceso.

En la figura 4 se generaliza la jerarquía de productos que pueden ser generados en cada proceso de MoProSoft.

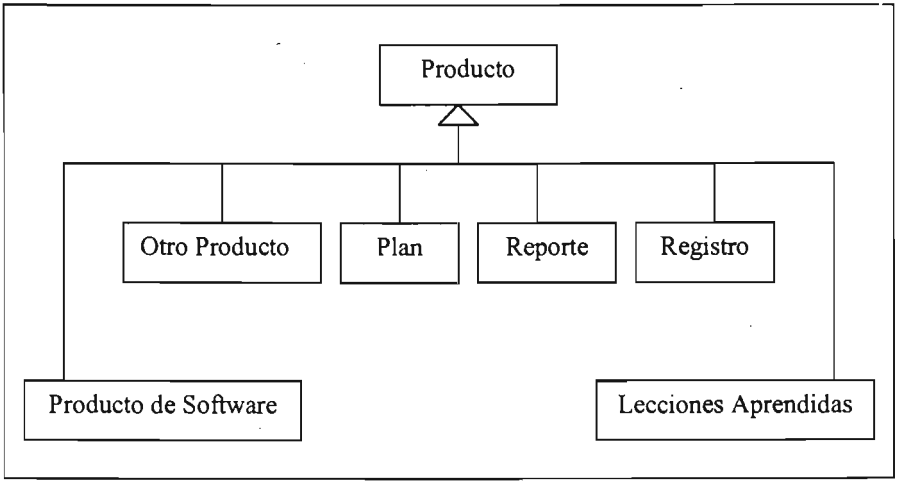


Figura 4. Clasificación general de productos.

2.3 Requerimientos de HIM

2.3.1 Necesidades para la creación de HIM

Para facilitar la adopción de MoProSoft, las empresas requieren de herramientas integradas que den soporte al modelo de procesos. Además, dada la baja disponibilidad de recursos en las empresas es deseable que la herramienta HIM sea de distribución libre.

2.3.2. Requerimientos Generales para HIM

Los requerimientos iniciales que contempla la herramienta integral MoProSoft, están considerados en los siguientes tres puntos:

1. Que forme un ambiente integrado para soportar las actividades de MoProSoft.

2. Que use la tecnología que permita su distribución como software libre.

3. Que tenga las siguientes cualidades:

    a) Portabilidad
    b) Soporte de diversos formatos de documentos
    c) Seguridad en el resguardo y en el acceso
    d) Seguridad en el control de permisos y registro de actividades.
    e) Usabilidad
    f) Soporte de grupos de trabajo
    g) Mantenible por sus propios usuarios

En función a los requerimientos establecidos para la construcción de la herramienta HIM, se estableció la tecnología J2EE[5] como plataforma de construcción, así como el servidor Web Tomcat[6]. Estas tecnologías fueron seleccionadas ya que cubren ampliamente las cualidades que demanda la herramienta.

[5] Java 2 Enterprise Edition.
[6] Asociación Apache Proyecto Jakarta.

Capítulo 3

Conceptos de Mecanismos de seguridad

En este capítulo se describen los tres mecanismos básicos de seguridad, que se proponen para la herramienta integral MoProSoft. Entre los cuales nos encontramos un control de acceso, control de permisos a roles y un registro de bitácora.

3.1 Control de acceso

Antes de definir el concepto de control de acceso, debemos definir y entender conceptos generales de Criptografía, ya que con base en ella podremos definir un control de acceso seguro.

3.1.1 Historia de la Criptografía

La criptografía es tan antigua como la escritura y se dice que las primeras civilizaciones que usaron la criptografía fueron la egipcia, la mesopotámica, la Indu y la china.

Los espartanos 4000 años antes de Cristo, utilizaban papiro en forma de escritura, el cual consistía en un cilindro al cual se colocaba un papiro en forma de espiral. Se escribía entonces el texto en cada una de las vueltas del papiro, pero de arriba hacia abajo, una vez desenrollado, sólo se podía leer una serie de letras aparentemente inconexas. Para descifrar el mensaje era necesario colocar el papiro exactamente en la misma posición en la que había sido escrito.

Antiguos textos judíos fueron cifrados siguiendo el método de sustituir la primera letra del alfabeto por la última y así sucesivamente.

Pero a quien se le atribuye el primer método de cifrado, es al general romano Julio César, quien creó un sistema simple de sustitución de letras, que consistía en escribir el documento cifrado con la tercer letra que le siguiera a la que realmente correspondía, la letra A era sustituida por la letra D, la B por la E y así sucesivamente.

Estos sistemas tan simples evolucionaron posteriormente a elegir una reordenación cualquiera (permutación) del alfabeto, de forma que cada letra se le hace corresponder otra ya sin ningún patrón definido.

Durante la primer guerra mundial se utilizaron extensivamente las técnicas criptográficas, lo que impulso al final de la guerra, el desarrollo de las primeras tecnologías electromecánicas, un ejemplo claro de ello fue el desarrollo de la máquina Enigma utilizada por los alemanes para cifrar y descifrar sus mensajes.

En el año de 1975 **Diffie** y **Hellman** son el parte aguas de la bases teóricas de los algoritmos de llave pública. Hasta entonces no se concebía un sistema de cifrado que no fuese de clave secreta, en la actualidad se usan distintos métodos criptográficos como DES (Data Encription System), RSA, MD5, etc. [Simson].

3.1.2 Conceptos Básicos de la Criptografía

La criptografía es el estudio de las técnicas matemáticas, relacionadas con aspectos de la seguridad de la información, tales como la confidencialización, integridad, autenticación y no repudio.

Objetivos de la criptografía.

Los objetivos principales que pretende alcanzar la criptografía son los siguientes:

- **Autenticación.**- Cuando recibimos algún mensaje de alguna persona, debemos de estar completamente seguros de que dicha persona es quien dice ser, es decir, que no exista un intruso que falseé su identidad.
  A esto daremos el término de autenticación, verificar que la persona es realmente es quien dice ser.

- **Integridad.**- La inseguridad latente que existe sobre nuestra información a través del canal público de la red. Por consecuencia es necesario el asegurarse que la información que enviamos o recibamos no haya sido modificada durante su traslado.

  Entenderemos entonces el término de integridad como, la información que no haya sido manipulada, ni alterada por información falsa.

- **No Repudio.**- Supongamos que recibimos un mensaje de un amigo vía correo electrónico, podríamos asegurar la autenticidad de la persona, pero no cubre la posibilidad de que dicha persona pueda negar de que él nunca nos envío el mensaje.

  Por lo tanto entenderemos entonces el término de no repudio, como la forma de asegurarnos que dicha persona fue realmente quien realiza una acción y no trate de refutar dicha acción.

- **Confidencialización.-** La confidencialización es el servicio utilizado para mostrar el contenido de la información, a las personas que tengan autorización de verla.

Definamos algunos conceptos generales que son utilizados frecuentemente en el medio de la seguridad informática.

Supongamos que deseamos transmitir un mensaje a través del canal público de la Internet a un amigo, este mensaje lo conoceremos como **Texto en Claro** y el proceso que realiza la distinción del mensaje en su ocultamiento lo conoceremos como **Cifrado.**

Donde el resultado obtenido al aplicar un cifrado a un texto en claro será denominado como **Texto Cifrado.**

Otros conceptos importantes son: [Menezes]

- **Criptología.-** La criptología es el estudio de la criptografía y el criptoanálisis.

- **Criptoanálisis.-** El criptoanálisis es el estudio de las técnicas matemáticas con el fin de derrotar las técnicas criptográficas y más generalmente servicios seguros de información.

- **Criptoanalista.-** El criptoanalista es alguien que emplea el criptoanálisis.

- **Criptosistema.-** El criptosistema se refiere a un conjunto de primitivas criptográficas, utilizadas para proveer servicios seguros de información.

3.1.3 Notaciones.

Antes de continuar es conveniente definir algunas notaciones matemáticas que son importantes para la comprensión de los términos utilizados para denotar los conceptos de: cifrado, descifrado, especio de llaves, por mencionar algunos.

- Denotaremos con la letra ***A,*** al conjunto finito del alfabeto de definición, por ejemplo A = {0,1} representa un alfabeto binario.

- Denotaremos con la letra ***M*** al conjunto que representa el espacio de los mensajes, ***M*** consiste de la cadena de símbolos del alfabeto. Debe de quedar claro que un elemento de ***M*** será llamado texto en claro o mensaje en claro. Por ejemplo ***M*** puede estar formado por cadenas binarias.

- Denotaremos con la letra ***C*** al conjunto que representa el espacio de texto cifrado. Sabemos que ***C*** consiste de cadenas de símbolos de un alfabeto, el cual puede diferir del alfabeto por definición de ***M***. Cabe señalar y quedar en claro que un elemento de ***C*** es llamado el texto cifrado.

- Denotaremos con la letra ***K*** al conjunto que representa el espacio de llaves. Debe de quedar en claro que un elemento de ***K*** es llamado llave.

3.1.4 Cifrado y Descifrado.

A continuación se describirán los conceptos generales sobre un cifrado y descifrado.

Definición de cifrado.

Es el proceso de transformación que sufre el texto en claro en un texto cifrado por medio de un algoritmo matemático y con la utilización de una llave.

Formalmente podemos decir que cada elemento de $k_1 \in \boldsymbol{K}$ determina únicamente una bisección de ***M*** a ***C***, denotado por E_k, lo cual representa a la **función de cifrado o un transformación de cifrado.**

El proceso de aplicar la transformación de E_k, a un mensaje $\boldsymbol{m} \in \boldsymbol{M}$, es usualmente nombrado ***cifrado de m*** (cifrado de mensaje en claro).

En conclusión la siguiente notación matemática expresa el proceso que se lleva a cabo al realizar el cifrado de ***m***.

$$\text{Cifrado } E_{k1}(\boldsymbol{M}) = \boldsymbol{C}$$

Definición de descifrado.

El proceso de descifrado consiste en tomar el texto cifrado transformándolo en el texto en claro, utilizando un algoritmo criptográfico y una llave.

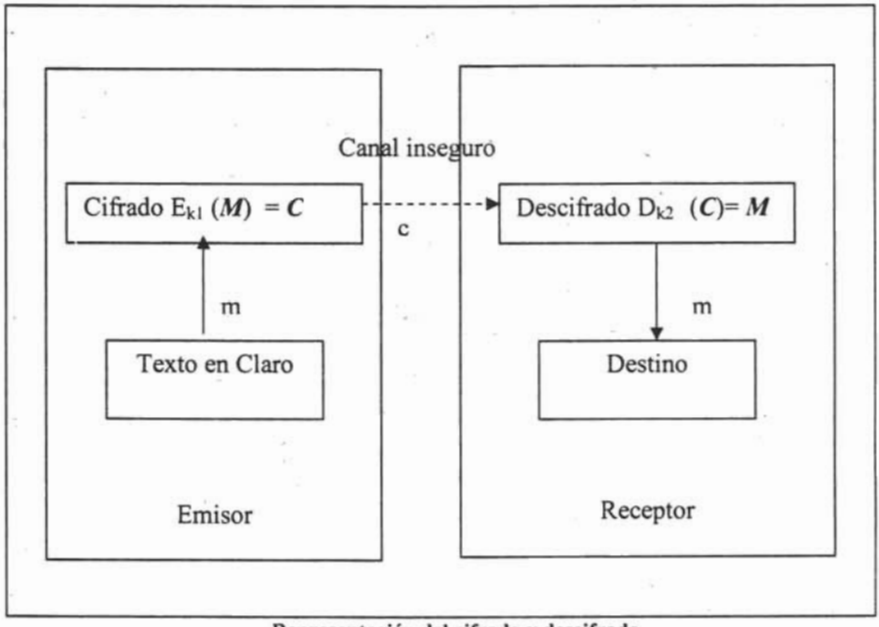
Formalmente podemos decir que para cada elemento de $k_2 \in \boldsymbol{K}$, donde D_{k2} denota una bisección de ***C*** a ***M***, la cual representa a la función de descifrado o transformación de descifrado.

El proceso de aplicar la transformación D_{k2} a un texto cifrado c es nombrando como el ***descifrado de c***.

En conclusión la siguiente notación matemática expresa el proceso que se lleva a cabo al realizar el descifrado a ***c***.

$$\text{Descifrado } D_{k2} \ (C) = M$$

En la siguiente figura queda plasmado gráficamente el proceso de cifrado y descifrado.



Representación del cifrado y descifrado.

3.1.5 Componentes de un Cifrado.

Todo algoritmo de cifrado esta compuesto por los siguientes elementos:

Algoritmo de Cifrado.- Es la función matemática que realiza la tarea de cifrado de datos y descifrado.

Llaves de Cifrado.- Las llaves de cifrado son las que se utilizan para poder cifrar un mensaje en claro y que mantendrán la confidencialidad del mensaje.

Longitud de Llave.- La longitud de la llave la determinará el algo que se utilice para cifrar el mensaje en claro, entre mayor sea la longitud de la llave está será más fuerte y por consecuencia más difícil de romperse.

Texto en Claro.- Se refiere a la información que deseamos proteger.

Texto Cifrado.- Es el texto obtenido después de realizar la transformación al texto en claro.

3.1.6 Participantes en la comunicación

Debemos tener presente y claro las entidades participantes en un sistema Web, estas entidades pueden representar ya sea a una persona o a un sistema. Dentro de las entidades participantes tenemos las siguientes:

- Una *entidad* es alguien o algo que envía, recibe o manipula la información.
- Un *emisor* es una entidad presente en la comunicación que se da entre dos partes, en donde el papel que juega el emisor es la de transmisor de la información.
- Un *receptor* es una entidad presente en la comunicación que se da entre dos partes, en donde el papel adoptado por el receptor es la de recibir la información.
- Un *adversario* es una entidad presente en la comunicación que se realiza entre las dos partes, donde no es el emisor ni el receptor si no que trata de obtener la información intercambiada entre el emisor y el receptor.

3.1.7 Canales de comunicación.

Un canal de comunicación podemos definirlo como el medio de transporte en donde se realiza el intercambio información de una entidad a otra. Podemos dar una pequeña clasificación de los tipos de canales que pueden existir:

- Canal físicamente seguro.- Es aquel tal que no existe un acceso físico al adversario.
- Canal inseguro.- Es aquel tal que la información que se intercambia entre la entidad emisora y receptora, podrá ser borrado, alterado o conocida la propia información por la entidad adversaria.
- Canal seguro.- Es aquel tal que la información que se intercambia entre la entidad emisora y receptora, no podrá ser borrado, alterado o consultado por la entidad adversaria.

3.1.8 Técnicas Criptográficas.

Una técnica criptográfica o también conocida como cifrado, es la función matemática utilizada para el cifrado y descifrado. Existen dos tipos de técnicas criptográficas que son utilizadas en la actualidad y una tercera técnica que combina a las dos anteriores. Estas tres técnicas criptográficas son:

- Cifrado simétrico.
- Cifrado asimétrico.
- Cifrado híbrido público/privado.

3.1.8.1 Cifrado simétrico.

Las características de este tipo de algoritmo están plasmadas en la utilización de la llave, donde la llave es utilizada para realizar el cifrado. Donde la misma llave es utilizada para descifrar.

Este algoritmo también es conocido como algoritmo de llave privada, algoritmo de llave única; requiriendo que la entidad emisora y la receptora realicen un acuerdo de llaves previo al inicio de la comunicación.

La seguridad del algoritmo simétrico recae sobre la llave, divulgando la llave significa que cualquiera podrá cifra y descifrar el mensaje. Por lo tanto la llave debe de permanecer en secreto.

Cifrado $\mathbf{E_k(M) = C}$ Donde k es llave privada

Descifrado $\mathbf{E_k(C) = M}$ Donde k es llave privada.

Conociendo las bases teóricas sobre la Criptografía podemos realizar un análisis sobre que aspectos debemos cubrir cómo mínimo en el desarrollo de los sistemas de software y llevado de la mano de los requerimientos del negocio, podemos tomar las decisiones correctas sobre qué y cómo solucionar aquellos aspectos de autenticación, no repudio, integridad y confidencialización.

La respuesta a la pregunta de qué deseamos proteger, podemos encontrarnos distintas respuestas a distintos niveles, como por ejemplo la autenticación de los usuarios finales del sistema que deseen ingresar, la integridad de la información, la confidencialización de la información, el no repudio por parte de los usuarios sobre las actividades que les fueron asignadas, etc.

El requerimiento sobre la construcción de la herramienta integral MoProSoft, fue el contemplar aquellos que como mínimo garantizara un nivel y control de seguridad adecuado. El primero de ellos fue considerar un control de acceso seguro y apoyado de las técnicas criptográficas adecuadas.

La respuesta a la pregunta de el cómo solucionar los requerimientos que se nos presenta será apoyado fuertemente de las técnicas criptográficas que se han descrito.

Cada una de estas técnicas se encuentra orientada a mitigar distintos aspectos de seguridad, para nuestro caso un cifrado Simétrica es el más adecuado y el que se ajusta a resolver el requerimiento de autenticación, a través del manejo de su llave privada y un desempeño adecuado en el tiempo que demanda realizar el cifrado y descifrado de los password's.

Por lo tanto se propone el uso de un cifrado Simétrico con la definición de llaves privadas únicas para cada uno de los usuarios del sistema, permitiendo un manejo individual de cifrado/descifrado de su password, con ello se tendrá un manejo descentralizado de las llaves privadas, con el fin de que si alguna llave privada se compromete su confidencialidad sea localizable de una manera más sencilla. En el siguiente capitulo se describirá a más detalle el uso y características principales del cifrado Simétrico que se utilizo en la implementación del sistema MoProsoft.

3.1.8.2 Cifrado Asimétrico

También conocido como algoritmo de llave pública el cual es diseñado para que la llave utilizada para el cifrado se diferente de la llave utilizada para descifrar. Además, la llave de descifrado no podría ser calculada por la llave de cifrado.

El algoritmo es llamado de llave pública, porque la llave de cifrado puede ser pública al mundo exterior, donde un extraño pudiera utilizar la llave pública para cifrar algún mensaje, pero únicamente una persona específica que conozca la llave privada podrá descifrar el mensaje.

Podemos distinguir dos aspectos importantes en el cifrado Asimétrico, el primero es que existen dos llaves en este tipo de esquema donde se conocerán como *llave pública* a la llave utilizada para cifrar y la segunda llave se conocerá como *llave privada* a la llave utilizada para realizar el descifrado.

Formalmente podemos definir que:

Cifrado $\mathbf{E_k(M) = C}$ Donde k es la llave pública.

Descifrado $\mathbf{D_k(C) = M}$ Donde k es la llave privada.

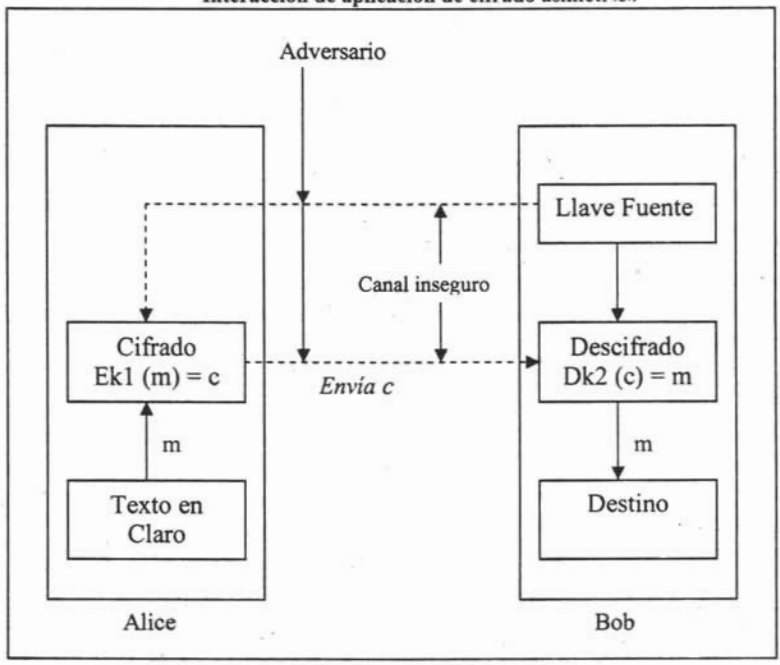
Sea $\{E_{k1}$: donde $_{k1} \in K\}$ un conjunto de transformaciones de cifrado y sea $\{D_{k2}$: donde $_{k2} \in K\}$ un conjunto de correspondientes transformaciones de descifrado y donde K es el conjunto del espacio de la llave.

Consideremos un par asociado de transformaciones de cifrado y descifrado (E_{k1}, D_{k2}) y supongamos que cada par tiene la propiedad de que conociendo E_{k1} es computacionalmente indecidible, dado un texto cifrado aleatorio $c \in C$ poder encontrar el mensaje $m \in M$ tal que $E_{k1}(m) = c$.

Esta propiedad implica que dada $_{k1}$ es poco probable determinar la correspondiente llave de descifrado $_{k2}$.

Ejemplificando el cifrado asimétrico supongamos que se da la comunicación entre Alice (emisor) y Bob (receptor), como se observa en la siguiente figura.

Interacción de aplicación de cifrado asimétrico



Bob selecciona un par de llaves denotadas por (k_1, k_2), donde envía la llave $_{k1}$ conocida como llave pública a Alice a través del canal de comunicación, pero guardando en secreto la llave de descifrado $_{k2}$ conocida como llave privada.

Alice posteriormente envía a Bob un mensaje cifrado aplicando la llave pública que le envío Bob, definido por $c = E_{k1}(m)$.

Una vez que Bob recibe el mensaje c, para descifrar el mensaje deberá aplicar la transformación inversa D_{k2} aplicando la llave privada $_{k2}$ que él únicamente conoce. Cabe señalar que en la figura mostrada anteriormente la llave k1 es transmitida a Alice sobre un canal inseguro, este medio puede ser el mismo por el cual el texto cifrado esta siendo transmitido.

3.1.8.3 Cifrado Híbrido

Hasta el momento se han analizado los dos algoritmos de cifrados más comunes, donde el algoritmo de llave privada es ampliamente utilizado con el fin de proteger la información almacena en un dispositivo de almacenamiento o para cifrar aquella información que viaja a través del canal inseguro como lo es la Internet.

Mientras que un algoritmo de llave pública es ampliamente utilizado con el fina de generar firmas digitales, en correos electrónicos, certificados, etc.

Con esta referencia podemos definir que un cifrado de tipo híbrido mezcla ambas ideologías, donde el cifrado asimétrico es utilizando principalmente para el intercambio de la llave de sesión y el cifrado simétrico para cifrar la información que viajará a través del canal de comunicación, además que este último algoritmo se beneficia por el menor tiempo que se presenta en realizar el proceso de cifrado.

3.1.8.4 Cifrado Simétrico vs. Asimétrico.

Los esquemas de llave privada y pública presentan algunas ventajas y desventajas realizando una comparación entre ambos, donde algunas de estas características son comunes para ambos.

Los sistemas criptográficos actuales explotan los beneficios de cada uno de los cifrados, por ejemplo el cifrar grandes cantidades de datos requiere un alto tiempo de procesamiento, en el cual un cifrado simétrico es altamente más eficiente que uno asimétrico. En su contraparte el establecer un acuerdo de llaves un cifrado asimétrico es más eficiente y contiene el mecanismo adecuado para realizarlo que un cifrado simétrico.

Para ver los pros y contras que presenta cada esquema de seguridad a continuación se presentan las ventajas y desventajas que presenta cada tipo de cifrado.

Ventajas – Desventajas del Cifrado Simétrico

Ventajas

- Puede ser diseñado para cifrar una alta proporción de información, algunas implementaciones de hardware consiguen cifrar proporciones de cientos de megabytes por segundo, mientras que una implementación de software puede obtener una proporción del rango de los megabytes por segundo únicamente.
- Las llaves son relativamente cortas.
- Puede ser utilizado para construir varios mecanismos criptográficos incluyendo el generador de números aleatorios, funciones hash y esquemas de firmas digitales computacionalmente eficientes.

Desventajas

- El acuerdo de la llave debe ser por un medio distinto y debiendo permanecer en absoluto secreto entre ambas partes.
- Debe existir un cambio frecuente de la llave utilizada, de ser posible en cada nueva sesión de comunicación.

Ventajas – Desventajas del Cifrado Asimétrico

Ventajas.

- Únicamente la llave privada debe permanecer en secreto.
- Ambas llaves pueden permanecer gran periodo de tiempo sin ser actualizadas.
- En una red extensa, el número de llaves necesarias puede ser considerablemente pequeño en relación a las llaves que se necesiten en un cifrado simétrico.

Desventajas

- El tamaño de la llave sin típicamente mucho más grandes que los requeridos para el cifrado de llave simétrica.
- La autenticación de la llave pública se debe garantizar para su uso.

3.2 Control de Permisos a Roles

El control de permisos a roles nos permitirá llevar un adecuado control sobre los recursos humanos que pertenecen a un grupo de trabajo dentro de la organización, permitiendo realizar las actividades que le fueron asignadas al rol.

Para ello es indispensable apoyarse en las nuevas tecnologías que surgen día con día, este es el caso de RDF[7] con el cual se analizó y mitigó el requerimiento que se tenía de llevar el control de permisos en el sistema.

3.2.1 Características de RDF

RDF es un *framework* con el fin de realizar la representación de la información a través de la Web, estableciendo una sintaxis abstracta que refleja un modelo de datos gráfico.

También establece una semántica formal definiendo la idea de suposición, proporcionando las bases para una adecuada deducción en los datos de RDF.

Dentro de las características que presenta RDF se destacan las que a continuación se describen:

- **Es un meta-modelo para la Web.**- Proporcionar la información referente a los recursos que existen en la Web y sobre los sistemas que utilizan.

- **Trabajar a través de aplicaciones.**- Combinar información entre aplicaciones con el fin de formar una nueva información.

- **Modelo de información abierto.**- Para aquellas aplicaciones que requieren un modelo abierto en lugar de un modelo de información restringido, facilitará el conocer por ejemplo, la descripción de los procesos organizacionales.

- **Proceso automatizado de la información.**- Actualmente la Web se inclina a proporcionar un banco de datos de información para su lectura, lo cual orilla a que exista un proceso de cooperación entre las distintas aplicaciones que viven en la Web. Donde RDF se consolida con la lengua universal que permitirá un proceso de cooperación en el manejo de la información.

3.2.2 Objetivos de RDF

Dentro de los objetivos que dio origen al nacimiento de RDF se destacan los que a continuación se mencionan:

[7] Resource Description Framework

- Tiene un modelo de datos simple
- Tiene una semántica formal y una probable inferencia.
- Utiliza un vocabulario basado en URI's.
- Sintaxis basada en XML.
- Soporta el uso de los tipos de datos definidos en los esquemas de XML.
- Permite establecer declaraciones sobre cualquier recurso.

Modelo de Datos Simple

RDF posee un modelo de datos simple que facilita a las aplicaciones su procesamiento y manipulación. El modelo de datos es independiente de alguna específica sintaxis de serialización.

Semántica Formal e Inferencia

RDF posee una semántica formal la cual proporciona las bases necesarias para el razonamiento acerca del significado de las expresiones de RDF. Permitiendo así la noción de entendimiento, la cual proporciona las bases para la definición de reglas confiables de inferencia en los datos de RDF.

Vocabulario Basado en URI's

El vocabulario es completamente extensible, basado en las URI's con un opcional uso de identificadores de fragmentos conocidas como referencias URI[8]. Las referencias de URI's son utilizadas para nombrar todo tipos de objetos en RDF.

Sintaxis definida en XML

RDF tiene una adecuada forma de serializar los documentos XML, lo cual pude ser utilizado para codificar el modelo de datos para el intercambio de la información a través de las aplicaciones.

Tipos de datos del esquema de XML

RDF puede hacer uso de los valores representados en los tipos de datos definidos en el esquema de XML, permitiendo el intercambio de información entre RDF y otras aplicaciones XML.

[8] También conocidas por la abreviatura de URIrefs

Declaraciones sobre recursos.

RDF permite abiertamente la definición de declaraciones sobre cualquier recurso, esto no quiere decir que la declaración contemple que la información se encuentre completa o que la afirmación sobre el recurso no sea incongruente o inconsistente con alguna otra declaración. Se debe tener presente este hecho en el momento de realizar algún diseño con RDF, estando concientes de que pueden existir incompletos o inconsistentes recursos de información.

3.2.3 Conceptos Básicos de RDF

Para poder comprender el funcionamiento de RDF, se debe comprender en primera instancia los siguientes conceptos que a continuación se mencionan:

- Modelo de Datos Gráfico
- Tipos de datos
- Literales
- Suposición

Modelo de Datos Gráfico

La estructura más importante de una expresión dentro de RDF corresponde a la colección de tripletas, donde una tripleta consiste de un **Sujeto**, un **Predicado** y un **Objeto**.

Un conjunto de tripletas es llamado una "gráfica RDF", puede ser ilustrado por un diagrama de un nodo y un arco dirigido en el cual cada tripleta es representada por un enlace entre nodo – arco – nodo, de aquí el término de gráfica.

Cada tripleta representa una declaración de relación entre objetos, denotado por los nodos que se encuentran ligados. Como ya lo mencionamos anteriormente cada tripleta esta compuesta por:

- Un **Sujeto**
- Un **Objeto** y
- Un **Predicado**, también conocido como propiedad, el cual denota la relación.

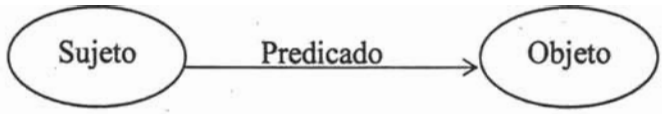


Figura1. Representación gráfica de los elementos que componen una tripleta

La dirección del arco tiene un valor significativo, siempre apunta hacia el nodo **Objeto**. Aclarando que los nodos dentro de un modelo gráfico en RDF únicamente pueden estar conformados por el nodo **Sujeto** o el nodo **Objeto**.

La afirmación de una triple de RDF indica que una relación es indicada por el **Predicado** mantiene la relación entre los otros dos objetos que enlaza en su caso al nodo **Sujeto** y el nodo **Objeto** de la tripleta.

Así también podemos afirmar que la representación completa de una gráfica RDF, para afirmar todas las tripletas que contiene dicha gráfica, corresponde a la conjunción de las declaraciones correspondientes a todas las tripletas que contiene en su totalidad el modelo grafico.

Para poder afirmar que dos gráficas RDF son equivalentes debe existir una biyección denotada con ***M*** entre el conjunto de nodos de las dos graficas, tal que:

Definamos la gráfica 1 denotada con **G** y la gráfica 2 con **G'**, entonces

1. ***M*** mapea los nodos blancos que $\in$ **G** con los nodos blancos que $\in$ **G'**
2. **M**(lit) = lit para todas las literales RDF que son nodos de la gráfica G
3. **M**(uri) = uri para todas las referencias URI de RDF que son nodos de la gráfica **G**.
4. Si la tripleta definida como (s, p, o) se encuentra en la gráfica **G** si y solo si la tripleta (***M***(s), p, ***M***(o)) se encuentra en **G'**

Con esta definición, ***M*** muestra como cada nodo blanco que se encuentra en la gráfica ***G*** puede ser remplazado con un nuevo nodo en **G'**.

Identificación de Nodos

Dentro del modelo gráfico tenemos la representación del nodo, cuyo valor asignado podrá ser de un URI con una referencia opcional a URI conocidas también como **URIref,** también podrá tener el valor de una literal o un nodo blanco.

Cuando una referencia URI es utilizada como un predicado se dice que identifica una relación entre los objetos representados por los nodos que este conecta.

Un nodo blanco es un nodo que no es una referencia URI o una literal, dentro de la sintaxis abstracta de RDF el nodo blanco es solamente un único nodo que puede ser utilizado en una o más declaraciones , pero que por la naturaleza no tiene un nombre propio.

Una convención utilizada en la representación gráfica de RDF, para poder permitir a varias declaraciones referirse a un mismo recurso no definido es el uso del identificador de nodo blanco, el cual es un identificador local que puede ser distinguido de todas las URI's y literales dentro del modelo gráfico. Cabe aclarar que un identificador de nodo blanco no forma parte de la sintaxis abstracta de RDF y el hecho de que la representación de tripletas contengan nodos blancos es completamente dependiente sobre una particular sintaxis utilizada.

Tipos de Datos

Los tipos de datos son utilizados por RDF para la representación de valores tales como los enteros, números de punto flotante y fechas.

Un tipo de dato esta conformado por los siguientes elementos:

- Un espacio léxico.
- Un espacio de valores y
- Un mapeo del espacio léxico – valor.

Por regla se define que cada miembro del especio léxico le corresponde exactamente un miembro del espacio de valores y cada miembro del espacio de valores puede corresponderle cualquier número natural que corresponda a la representación del espacio léxico.

Por ejemplo, el mapeo del espacio léxico – valor para un tipo de dato xsd:boolean del esquema de XML, donde cada miembro del espacio de valores presenta dos representaciones léxicas, como a continuación se presenta:

Espacio de valores	{ T, F }
Espacio léxico	{ "0", "1", "true", "false" }
Mapeo espacio léxico - valor	{ <"true",T> , <"1", T> , <"0", F> , <"false", F>

Cabe señalar que RDF solamente define un tipo de dato el cual es rdf :XMLLiteral, el cual es utilizado para poder incluir declaraciones XML dentro de RDF.

RDF no proporciona algún mecanismo para la definición de un nuevo tipo de dato, por lo que se espera o lo deseado es el uso de los tipos de datos predefinidos en los esquemas de XML así de igual manera hacer uso de la posibilidad de extender estos tipos de datos manejados dentro de los esquemas de XML.

Literales

Las literales son utilizadas para identificar los valores de los tipos de datos numéricos y fechas para el entendimiento de la representación léxica. Cualquier cosa representada por una literal puede también ser representada por una URI, pero es más conveniente o intuitivo el uso de las literales.
Una literal puede representar a un **Objeto** en una declaración RDF, pero no puede representar a un **Sujeto** o **Predicado** dentro de la declaración.

Una literal es clasificada en dos tipos distintos, la literal simple y la literal tipificada:

- Una literal simple.- Es una cadena combinada con un lenguaje opcional de etiquetas; también puede ser utilizada para alojar una cadena de texto plano.

- Una literal tipificada.- Es una cadena combinada con un tipo de dato URI, así también denota el valor del tipo de dato aplicando el mapeo entre el espacio léxico – valor a la cadena literal.

Ejemplificando el caso de una literal tipificada esta puede ser definida utilizando el tipo de dato xsd:boolean del esquema de XML, como se muestra a continuación:

Literal tipificada	Mapeo espacio léxico - valor	Valor
<xsd:boolean, "true">	<"true" , T>	T
<xsd:boolean, "1">	<"1", T>	T
<xsd:boolean , "false">	<"false", F>	F
< xsd:boolean , "0">	<"0", F>	F

Para definir formalmente que dos literales son idénticamente las mismas, debemos cumplir las siguientes condiciones:

- Dos literales son iguales si el correspondiente valor léxico de sus cadenas, son carácter por carácter idénticos.
- Si ninguna o ambas literales hacen uso del lenguaje de etiquetas.
- Si es utilizado un lenguaje de etiquetas, comparando etiqueta por etiquetas son las mismas.
- Si ninguna o ambas literales presentan tipos de datos URI's.
- Si los tipos de datos URI's comparados carácter por carácter son los mismos.

Suposición

Las ideas sobre el significado y la inferencia en RDF se encuentran sustentadas por el concepto de **Suposición.**

Supongamos que tenemos una expresión ***A*** la cual es una **suposición** de otra expresión ***B*** si existe una concordancia entre los objetos que hacen ***A*** verdadera también hacen verdadera a ***B***. Sobre estas bases, si el razonamiento sobre la expresión ***A*** es demostrado como verdadero entonces se puede inferir que la expresión ***B*** también es verdadera.

Una vez analizado y comprendido los conceptos de RDF, podemos definir y proponer las bases para un mecanismo de control de permisos enfocados a los roles definidos en **MoProSoft.**

La idea fundamental es apoyarse en las suposiciones lógicas que podemos definir a través de las sentencias de RDF para poder llevar a cabo la inferencia de los permisos a los cuales cada rol tiene permiso de realizar.

Con ello queremos extraer las reglas de negocio que se definen en el modelo de procesos **MoProSoft**[9] sobre la relación entre los roles y sus correspondientes actividades, las cuales serán mapeadas a sentencias RDF´s con el fin de inferir de estas suposiciones las actividades que podrá realizar cada rol. En el siguiente capitulo se describirá más detalle este mecanismo propuesto.

3.3 Registro de Bitácora.

Para el mecanismo de registro de bitácora se revisó el concepto de patrones que permitiera concentrar en un único objeto la responsabilidad de llevar la tarea del registro en un archivo con formato XML, con el fin de realizar una auditoria eficiente de estos log´s. Para lo cual se estudiara en forma breve los conceptos de Patrones y XML.

3.3.1 Patrones

La idea principal de los patrones es el concepto de estandarización de la información sobre un problema común y su solución. Uno de los resultados más útiles del trabajo de Alexander[10] fue el desarrollo de una plantilla para representar los patrones, que se suele denominar formulario o formato.

El formato de Alexander utiliza cinco secciones para formalizar la explicación de un patrón y su solución, es importante que un patrón proporcione un nombre descriptivo y un objetivo claro de lo que hará.

---

[9] [Hernández]

[10] Christopher Alexander catedrático de arquitectura de la U.C. Berkeley , al cual se le atribuye la creación de los patrones de diseño en el desarrollo de software.

Además se debe incluir una exposición del problema, una explicación de cómo el patrón soluciona el problema y una lista de ventajas, inconvenientes y compromisos asociados con el uso de los patrones.

Actualmente existen varios formatos, la mayoría de los formatos que se utilizan actualmente derivan del cánonico o el definido por *Gang of Four* (GoF)[11] ó la pandilla de los cuatro.

El formato de GoF esta conformado por los siguientes campos principales que a continuación se detallan:

Nombre	Un nombre descriptivo del patrón.
Propiedades	Clasificación del patrón, donde la clasificación del patrón se encuentra definida con base a dos aspectos principales que son el Tipo y Nivel.
Tipo	El tipo del patrón puede ser clasificado en: • Patrones de **creación de objetos** • Patrones de **comportamiento**, que coordina la interacción funcional entre objetos. • Patrones **estructurales**, que gestionan relaciones estáticas y estructurales entre objetos. • Patrones de **sistemas**, que gestionan la interacción a nivel de sistema.
Nivel	El nivel del patrón se clasifica en: • Clase única, se aplica a una sola clase. • Componente, implica un grupo de clases. • Arquitectónico, utilizado para coordinar las acciones de los sistemas y subsistemas.
Propósito	Explicación breve de las implicaciones del patrón.
Presentación	Descripción de un problema que puede ser afrontado con este patrón.
Aplicabilidad	Cuándo y por qué sería deseable usar este patrón de diseño.
Descripción	Explicación más detallada del patrón, indicando qué hace y cómo se comporta.
Implementación	Explicación de qué debe hacerse para implementar el patrón.
Ventajas e inconvenientes	Consecuencias de usar el patrón y los compromisos asociados al uso del mismo.
Variantes	Posibles implementaciones alternativas y variaciones del patrón.

[11] La pandilla de los cuatro esta conformada por: *Erich Gamma, Richard Helm, Ralph Johnson* y *John Vlissides*

Patrones relacionados	Otros patrones con los que están asociados o con los que tiene una relación estrecha.
Ejemplo	Ejemplificación del código fuente.

3.3.1.1 Patrón de diseño GoF y Core-J2EE aplicado a los mecanismos de seguridad.

Tomando como referencia los requerimientos en el desarrollo de la seguridad de HIM, el patrón utilizado para generar la bitácora fue el conocido como **Singleton** (Único), el cual se describe a continuación brevemente

Singleton (Único)

Propiedades: patrón de tipo creación-objeto

Propósito: Permitir tener una única instancia de esta clase en el sistema, a la vez que se permite que todas las clases tengan acceso a esa instancia.

Presentación: En algún momento se deseará tener un objeto global; uno que sea accesible desde cualquier parte pero que sólo deba ser creado una vez. Se desea entonces que todas las partes de la aplicación sean capaces de utilizar el objeto, pero que todas deban utilizar la misma instancia.

Aplicabilidad: El patrón de **Singleton** debe utilizarse cuando se quiere una sola instancia de una clase, pero que quiera que este disponible globalmente.

Descripción: El patrón de **Singleton** asegura que se crea un máximo de una instancia en el JVM (Java Virtual Machine), para asegurar que tiene el control sobre la instanciación, haga que el constructor sea privado. Esto implicará no poder crear una instancia, por lo que se debe proporcionar un método estático denominado **getInstance()** para poder acceder a una instancia de esa clase. Ese método crea una instancia única, en caso de que aún no exista, y devuelve la referencia al objeto. La referencia al **Singleton** también se almacena como un atributo privado estático de la clase **Singleton** para llamadas futuras.

Implementación: por simplicidad, la implementación de este patrón se realizará en la sección de análisis y diseño de seguridad, por lo que se mostrará en el Capítulo 4.

Ventajas e inconvenientes: Entre las ventajas e inconvenientes encontramos que la clase **Singleton** es la única clase que puede crear una instancia de sí misma, por lo que no se puede obtener ninguna instancia sin utilizar el método estático proporcionado. Sin embargo el patrón puede presentar problemas de acceso múltiple, dependiendo de cómo se haya realizado la implementación.

Variaciones.- Entre las variaciones que puede presentar el patrón son las de tener más de una instancia dentro de la clase y la otra es que dentro del método se tome como punto de acceso al conjunto de instancias.

Patrones relacionados: Fábrica Abstracta (Abstract Factory), Construtor (Builder), Prototipo (Prototype)

Intercepting Filter

Propósito: Cuando se requiere un pre-procesamiento y un post-procesamiento de las peticiones o respuestas de un cliente Web.

Presentación: El mecanismo de manejo de peticiones de la capa de presentación recibe muchos tipos de peticiones, cada uno de los cuales quizás se requiere un tipo distinto de procesamiento. Algunas peticiones simplemente requieren su reenvió al componente manejador apropiado, mientras que otras peticiones deben ser modificadas, auditadas, o descomprimidas antes de su procesamiento posterior.

Aplicabilidad: Cuando una petición es recibida dentro de una aplicación Web, normalmente debe pasar varias pruebas de entrada antes del estado de procesamiento principal. Podemos cuestionarnos lo siguiente: ¿Se ha autenticado al cliente? ¿Tiene el cliente una sesión válida? ¿La dirección IP del cliente proviene de una red conocida? Algunos de este tipo de verificaciones, que deben proporcionar una respuesta de un si o un no, la cual determinara si continúa el procesamiento, caracterizan a los filtros. La clave para solventar este problema de forma flexible es tener un mecanismo para añadir y eliminar componentes de procesamiento, en el que cada componente realiza una acción de filtrado específica.

Descripción: Crear filtros conectables para procesar servicios comunes de una forma estándar sin requerir cambios en el código principal de procesamiento de la petición. Los filtros interceptan las peticiones las peticiones que se reciben y las peticiones de salida, permitiendo así un pre y post-procesamiento. Existirá la ventaja de añadir y eliminar estos filtros a discreción, sin necesitar cambios en nuestro código ya existente.

Implementación: por simplicidad, la implementación de este patrón se realizará en la sección de análisis y diseño de seguridad, por lo que se mostrará en el Capítulo 4.

Ventajas e inconvenientes: 1) Puede ser centralizado el control de múltiples peticiones, los filtros se encuentran preparados para manejar las peticiones y respuestas. 2) Mejora la reutilización a través de un claro desacoplamiento entre los filtros, pudiendo combinar una cadena de filtros para lograr un cierto comportamiento.3) El compartir información entre los diversos filtros puede resultar ineficiente, debido a la naturaleza de los filtros que presentan un alto desacoplamiento entre los demás, por lo que resulta complicado el intercambio de información.

Patrones Relacionados: Controlador Central (Front Controller), Plantilla de Método (Template Method),

3.3.2 XML

XML (Extensible Markup Language) se caracteriza por ser simple, con un formato de texto muy flexible y que derivo del SGML (Standard Generalizad Markup Language). Originalmente XML fue diseñado para soportar la gran escala de publicaciones electrónicas, además de estar incrementando con gran importancia el rol en el intercambio de una variedad de datos a lo largo de la Web.

La definición nos introduce formalmente en el concepto de XML. ¿Pero qué es y de qué se conforma XML para ser tan utilizado?

- XML es considerado un lenguaje de marcado, igual que HTML.
- XML fue diseñado para describir datos.
- Las etiquetas con que cuenta no son predefinidas, uno debe predefinir sus propias etiquetas.
- XML utiliza la definición de tipo de documentos también conocidas como DTD (Document Type Definition) o utiliza los conocidos Esquemas XML para poder describir datos.
- XML con DTD's o XML con Esquemas fue diseñado para ser por si mismo descriptivo.

Debe quedarnos claro que XML fue creado con el fin de definir estructuras de información bien formadas, para el almacenamiento y envío de la información. Así también XML presenta la cualidad de ser libre y extensible, es decir XML permite al autor definir sus propias etiquetas así como sus propias estructuras en el documento.

El registro para el control de todas las actividades que realizará el usuario del sistema, se propone que se realice a través de una clase **Singleton** que será global para todo el sistema y registrará en un archivo de texto con formato XML, donde se definió una estructura particular para este archivo manteniendo la integridad que caracteriza a un archivo XML de bien formado.

Con todo esto se pretende llevar un registro controlado de las actividades que cada uno de los roles realiza dentro del sistema HIM y mantenerlo en un formato que día a día se estandariza para la manipulación e intercambio de información.

En el siguiente capítulo se describen las etapas de análisis y diseño que se llevaron acabo para su posterior implementación, sobre los tres mecanismos de seguridad control de acceso, control de permisos y registro de bitácora.

Capítulo 4

Análisis y Diseño para la seguridad de HIM

En este capítulo se plasma las etapas del análisis y diseño que se realizaron para la construcción del sistema HIM, enfocándonos principalmente en los componentes de seguridad. Antes de iniciar con la descripción de las etapas de análisis y diseño, se describe brevemente los requerimientos y causas que dieron origen a la construcción de nuestra herramienta.

4.1 Requerimientos de seguridad para HIM.

Los requerimientos mínimos de seguridad que demanda la herramienta HIM, se encuentran en función al modelo estructural de procesos que conforma MoProSoft.

El modelo estructural de MoProSoft define diversos roles, los cuales se encargan de realizar las actividades apropiadas de los proceso. Por tal motivo la herramienta HIM, debe soportar un esquema multiusuarios, es decir, que el sistema deberá atender simultáneamente a distintos usuarios que adoptarán un rol al ingresar al sistema.

Todo sistema debe presentar un **mecanismo de ingreso al sistema**, por lo que se debe proporcionar un mecanismo seguro de autenticación a HIM. El mecanismo de autenticación debe apoyarse de técnicas criptográficas, con el fin de proteger la confidencialidad de la contraseña de cada usuario que desee ingresar al sistema.

Cada uno de los roles en MoProSoft tiene definidas claramente las actividades que puede realizar. Sería no adecuado si el sistema le permitiera a un rol realizar actividades no permitidas. Por lo tanto se debe de implementar un mecanismo, que nos permita tener un **control de permisos**, es decir, control hacia las actividades que puede realizar cada uno de los roles.

Por último MoProSoft define la actividad de registro de rastreo sobre las tareas que llevan acabo cada rol. Por lo que se propone un **registro de bitácora**, donde se aloje qué rol realizó cuál actividad y desde qué equipo de cómputo lo hizo.

Por conclusión los requerimientos mínimos en aspectos de seguridad que serán considerados en la implementación de la herramienta HIM son:

- Control de Acceso.
- Control de permisos a roles.
- Registro de bitácora.

En el siguiente capítulo se describen a más detalle los tres mecanismos de seguridad propuestos para la herramienta HIM. Además de conceptos generales de técnicas criptográficas que nos ayudaran en el confidencialidad de las contraseñas. Así también hablaremos de **RDF**[12], el cual nos ayudará a proporcional un nivel adecuado de control de permisos a roles.

4.2 Requerimientos.

Toda construcción de un sistema de software tiene como origen las necesidades que existen dentro de una empresa o negocio, las cuales requieren la automatización o control sobre procesos o actividades cotidianas que realizan para su funcionamiento.

La necesidad inicial que existió y dieron origen a la construcción de nuestra herramienta HIM para soportar el modelo de procesos MoProSoft, tenemos que:

"MoProSoft es un modelo de procesos para el desarrollo y mantenimiento de software dirigido a la pequeña y mediana industria y a las áreas internas de informática. Para facilitar su adopción las empresas requieren de herramientas que lo soporten. Dada la baja disponibilidad de recursos en las empresas es deseable que las herramientas sean de distribución libre "

La idea conceptual que dio origen al proyecto para la construcción de la herramienta HIM, surgió del curso impartido por la Dra. Hanna Oktaba y la M. en C. Guadalupe Ibargüengoitia, dentro del posgrado de ciencias e ingeniería de la computación de la UNAM.

Sabemos bien que en la actualidad las industrias de software en México, les es difícil adoptar y acoplarse a un Modelo de Procesos de Software, el cual permita madurar durante el transcurso del tiempo y alimentarlo con las experiencias de sus desarrollos.

De aquí la necesidad de crear una herramienta sencilla y eficaz que les permita adoptar un Modelo de Procesos para llevar la administración, gestión y control de sus proyectos.

La herramienta HIM deberá cubrir todo el ciclo que define un modelo de procesos y no únicamente de forma parcial, como lo hacen otras herramientas existentes en el mercado, es decir, existen herramientas para la administración de nuestros recursos humanos o para realizar la planeación del proyecto, pero de forma aislada. Lo que se pretende es tener una herramienta que permita control y administración de todas las actividades dentro del proceso interno del desarrollo de software.

[12] Resource Description Framework

Formalizando los requerimientos de la herramienta HIM, se encuentran los siguientes puntos:

- Que forme un ambiente integrado para soportar todas las actividades de MoProSoft.
- Que use una tecnología que permita su distribución como software libre.
- Que tenga las siguientes cualidades:
    - Control de acceso
    - Soporte de diversos formatos de documentos
    - Seguridad en el control de acceso
    - Seguridad en los permisos sobre las actividades de cada rol.
    - Soporte de grupos de trabajo
    - Mantenible por sus propios usuarios
    - Registro de actividades realizadas por los roles.
    - Usabilidad
    - Portabilidad

Una vez definido los requerimientos y alcance de HIM, podemos iniciar a realizar el análisis de los componentes de seguridad que abarcan los requerimientos de control de acceso, manejo de permisos y registro de actividades.

4.3 Análisis de los requerimientos funcionales de Seguridad[13]

Los diagramas de casos de uso plasman de forma gráfica los requerimientos de un sistema de software, es el primer artefacto que se debe generar para dar la pauta al inicio del análisis. El diagrama de casos de uso se representa a través de un Actor mostrado en la parte izquierda del diagrama y con una elipse la funcionalidad que estamos representando. La representación visual muestra el alcance que tendrá el sistema, reflejando que actor o actores realizan una determinada funcionalidad.

4.3.1 Diagrama General de casos de uso de HIM

Se comenzará describiendo el diagrama general de casos de uso de la herramienta HIM, mostrando de manera clara el alcance y los requerimientos que se cubrirán como mínimo en el desarrollo, se presenta en la **Figura 1**.

[13] Realización del análisis sobre el proceso unificado [Jacobson].

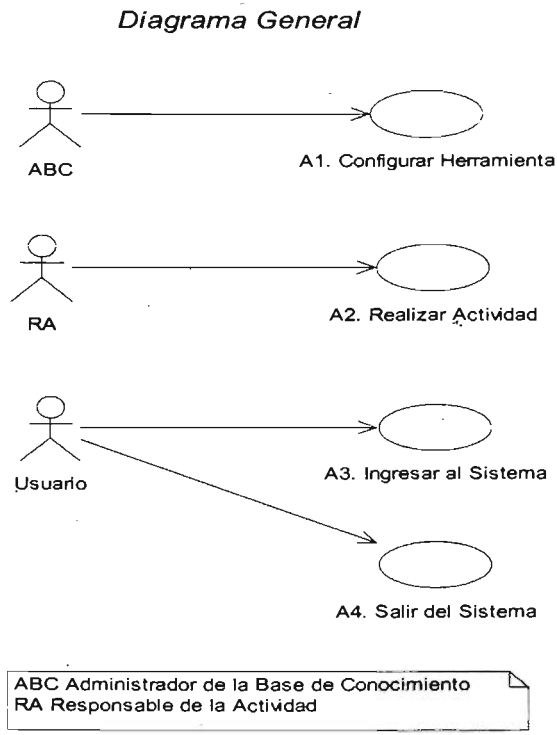


Figura 1. Diagrama General de casos de uso HIM

En la Figura 1 presenta de manera general, la funcionalidad que cubrirá el sistema, donde podemos observar la existencia de 3 Actores que tienen asociadas sus respectivas funcionalidades.

El actor *Administrador de la Base de Conocimiento* (ABC) es el encargado de realizar la actividad para generar la base de conocimiento la cual involucra definir una ruta donde se alojará la herramienta, construir la estructura jerárquica y construir el marco de trabajo para los proyectos o procesos.

El actor *Responsable de la Actividad* (RA) es un estereotipo utilizado para generalizar a los actores encargados de realizar las actividades de construcción y modificación de los productos.

El actor *Usuario* representa al usuario final, el cual tendrá la interacción directa con el sistema, este actor podrá ingresar y salir del sistema. El ingreso al sistema describe el detalle de la autenticación y manejo de permisos que tiene asignado cada rol en MoProSoft.

4.3.2 Diagramas detallados de casos de uso relacionados a la seguridad

Dentro de esta sección se presentan los diagramas detallados provenientes del diagrama general, de los cuales únicamente nos centraremos en detallar aquellos que involucren algún aspecto de seguridad, entre los que se encuentran la Realización de Actividades, el Ingreso y Salida del sistema.

Se comenzará detallando el diagrama de casos de uso *Ingresar al Sistema*, mostrado en la Figura 2, el cual contempla la *Autenticación del usuario* como caso de uso principal. La autenticación del usuario consiste en validar la existencia del usuario que la contraseña que proporcionó sea correcto.

Dentro de este caso de uso se puede observar la inclusión de otros casos de uso de los que requiere para completar su funcionalidad.

Entre los que se encuentra la *Recuperación de los Roles*, que tiene como finalidad obtener de la base del conocimiento la información relacionada con los usuarios registrados en el sistema y sus correspondientes contraseñas. Así también se encuentra la invocación al caso de uso de *Registro de Actividad*, el cual simplemente dejará registrado en bitácora de quién ingresó o si alguien intentó ingresar sin éxito.

El otro caso de uso *Construcción de Permisos al rol*, es una extensión del caso de uso principal el cual podrá o no realizar su funcionalidad, dependiendo de las condiciones que se den dentro de *la Autenticación del usuario*.

Por tal motivo el caso de uso de *Construcción de Permisos al rol* tiene como objetivo definir el Marco de Trabajo para el usuario basándose en las actividades que tendrá permiso a realizar el rol con que ingreso al sistema.

Para completar dicha funcionalidad se hace la inclusión de otro caso de uso para la recuperación de los proyectos, a los cuales están asignados al rol, para saber de qué proyecto se deberá construir el Marco de Trabajo del rol (ver tesis [Hernández]).

A3. Ingresar al Sistema

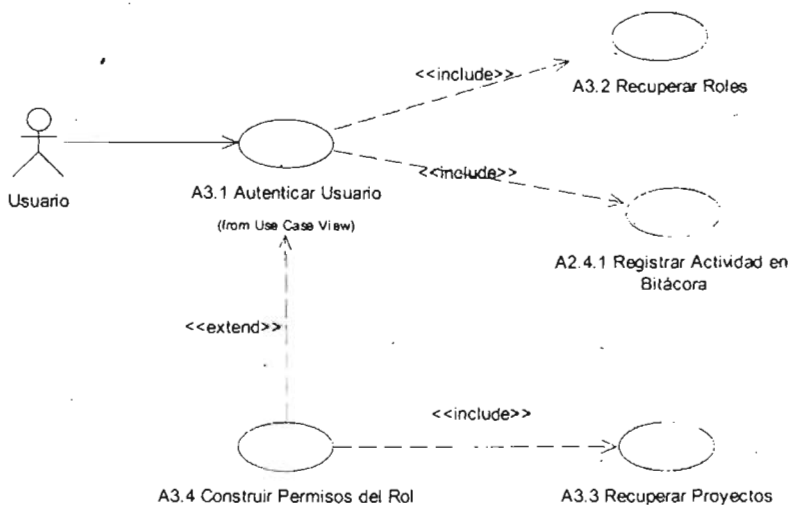


Figura 2. Diagrama Detallado para Ingresar al sistema

De manera sencilla se presenta en la Figura 3 la *Salida del Sistema*, donde únicamente se destaca el realizar en registro de bitácora de la hora y quién realiza la salida del sistema.

A4. Salir del Sistema

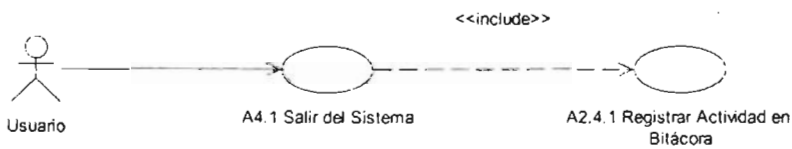


Figura 3. Diagrama Detallado para Salir del Sistema

Por último se presenta en la Figura 4, el diagrama detallado que representa del diagrama general presentado en la sección 4.1.2 la realización de actividades, en la cual se muestra que el actor *Responsable de Proyecto* podrá editar, consultar y guardar los artefactos que se generen. Estos dos casos de uso incluyen el registro de estas actividades en la bitácora.

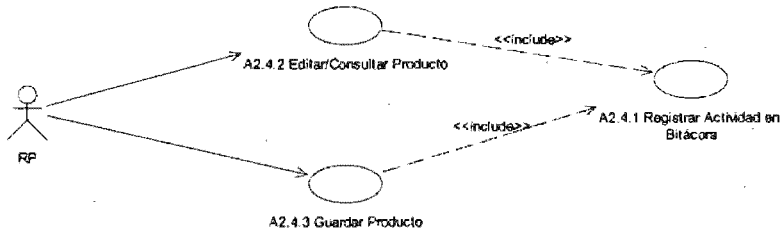


Figura 4.Diagrama Detallado para Realizar Actividades

4.3.3 Actores involucrados en MoProSoft

Como nos hemos podido dar cuenta, en los diagramas de casos de uso se muestran algunos actores que se detectaron en la creación de los diagramas. Pero dentro del análisis realizado al Modelo de Procesos MoProSoft, se detectaron una serie de actores involucrados, de los cuales se iniciará presentando la conceptualización que define MoProSoft de la manera jerárquica de actores, los cuales posteriormente se detallarán.

En la Figura 5, se muestra como MoProSoft clasifica a los actores involucrados en el proceso interno del modelo.

Dentro de la clasificación se puede observar la generalización de actores, donde el actor principal representa al actor *Rol* el cual juega el papel más importante dentro de la seguridad, para poder llevar un control de permisos con base al Rol del actor.

Del actor *Rol* se realiza una especialización a los actores de *Cliente, Responsable de Procesos, Involucrado y Grupo Directivo*, los cuales podremos conceptualizarlos como roles específicos que tienen bien definido su perfil de actividades. Así a su vez el Cliente se especializa como Usuario Final.

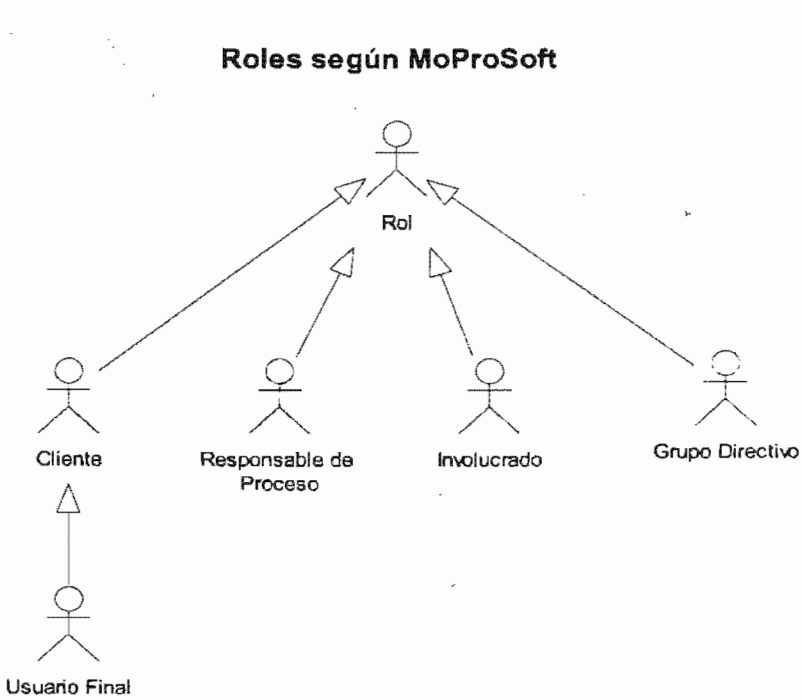


Figura 5. Clasificación Jerárquica de Actores según MoProSoft

La clasificación jerárquica presentada anteriormente, muestra únicamente de forma general aquellos actores involucrados en MoProSoft. Recordando un poco cómo se mencionó en el Capítulo 2, MoProSoft se encuentra clasificado en 3 categorías principales tomando como base la estructura organizacional de una empresa, donde en cada categoría intervienen roles bien definidos para realizar aquellas actividades que le fueron asignadas.

Esta clasificación por categoría – roles se muestra a continuación, dando una descripción breve de las características que presenta cada actor.

Dentro de la Figura 6, podemos destacar la presencia de 3 actores involucrados en la *Alta dirección*, cada uno de ellos juega un rol importante. La descripción de las características que presenta cada uno de estos roles se describe a continuación:

Dentro de la categoría de *Alta dirección*, presenta como actividades principales la planeación estratégica, la preparación para la realización de las estratégicas y la valoración y mejora continúa.

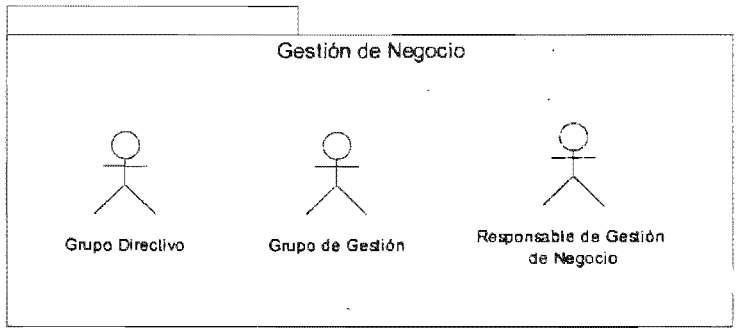


Figura 6. Actores involucrados en la Categoría de Alta Dirección

Rol	Nomenclatura	Características
Grupo Directivo	GD	Conocimiento del esfuerzo requerido para llevar a cabo la planeación estratégica, y sobre todo estar comprometido con éste.
Responsable de Gestión de Negocio	RGN	Conocimiento de las actividades necesarias para definir e implantar exitosamente el proceso de Gestión de Negocio.
Grupo de Gestión	GG	Conocimiento para administrar los proyectos e implantar los procesos definidos.

Para la segunda categoría que conforma MoProSoft, nos encontramos a la *Gestión* conformada por tres gestiones principales; *Gestión de Procesos*, *Gestión de Proyectos* y *Gestión de Recursos*, donde este último se subdivide a su vez en: *Recursos Humanos y Ambiente de Trabajo*, *Bienes, Servicios e Infraestructura* y *Conocimiento de la Organización*.

En la Figura 7, se presentan los 4 actores involucrados dentro de las actividades de *Gestión de Procesos*. La descripción de las características que presenta cada uno de estos roles se describe a continuación:

Entre las actividades principales en que se encontrarán involucrados los actores para la categoría de *Gestión de Procesos*, se encuentran realizar la planeación, la preparación para la implantación y por último la evaluación y control de los mismos.

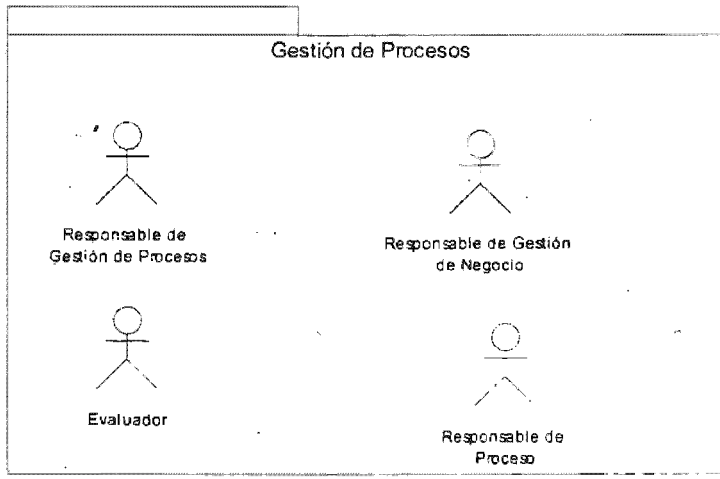


Figura 7. Actores involucrados en la Categoría de Gestión de Procesos

Rol	Nomenclatura	Características
Responsable de Gestión de Negocio	RGN	Conocimiento del esfuerzo requerido para lleva a cabo la Gestión de Procesos y sobre todo estar comprometido con éste.
Responsable de Gestión de Procesos	RGP	Conocimiento de las actividades necesarias para definir e implantar exitosamente el proceso de Gestión de Procesos.
Responsable de Proceso	RP	Conocimiento del proceso del cual es responsable.
Evaluador	EV	Conocimiento en metodología y aplicación de evaluación.

En la Figura 8, concerniente a la categoría de *Gestión* se encuentran involucrados dos actores para la *Gestión de Proyectos*. La descripción de las características que presenta cada uno de estos roles se describe a continuación:

Rol	Nomenclatura	Características
Responsable de Gestión de Negocio	RGN	Conocimiento del esfuerzo requerido para lleva a cabo la planeación de Gestión de Proyectos.
Responsable de Gestión de Proyectos	RGPY	Conocimiento de las actividades necesarias para llevar a cabo la Gestión de Proyectos.

Las actividades que involucra el realizar la *Gestión de un Proyecto*, consiste en realizar la planeación de las actividades y recursos que se requieren para cada uno de los proyectos, la realización, evaluación y control de los mismos.

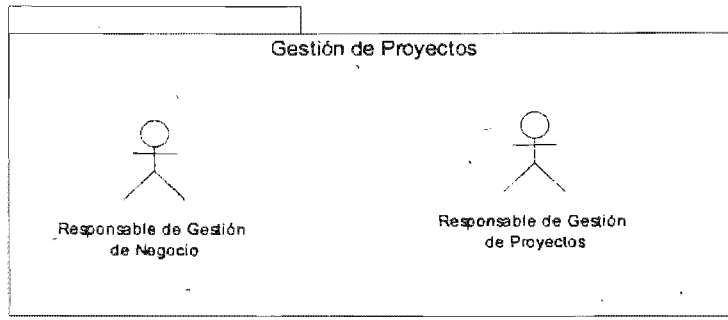


Figura 8. Actores involucrados en la Categoría de Gestión de Proyectos

Por último en la *Gestión de Recursos*, interactúan 5 actores con distintos roles, representado esquemáticamente en la Figura 9. La descripción de las características que presenta cada uno de estos roles se describe a continuación:

Rol	Nomenclatura	Características
Responsable de Gestión de Negocio	RGN	Conocimiento del esfuerzo requerido para lleva a cabo la planeación de Gestión de Recursos.
Responsable de Gestión de Recursos	RGR	Conocimiento de las actividades necesarias para definir e implantar exitosamente el proceso de Gestión de Recursos.
Responsable de Recursos Humanos y Ambiente de Trabajo	RRHAT	Conocimiento de las actividades necesarias para implantar exitosamente el subproceso de Gestión de Recursos Humanos.
Responsable de Bienes, Servicios e Infraestructura.	RBSI	Conocimiento de las actividades necesarias para implantar exitosamente el subproceso de Bienes, Servicios e Infraestructura.
Responsable de Conocimiento de la Organización	RCO	Conocimiento necesario para garantizar la integridad, seguridad y eficiencia de la Base de Conocimiento.

Dentro de las actividades gruesas que se realizan internamente en la *Gestión de Recursos* y en que los actores mencionados anteriormente manejan un rol definido, se encuentra la planeación de los recursos humanos y de bienes, el seguimiento y control de la planeación y por último la investigación de tendencias tecnológicas.

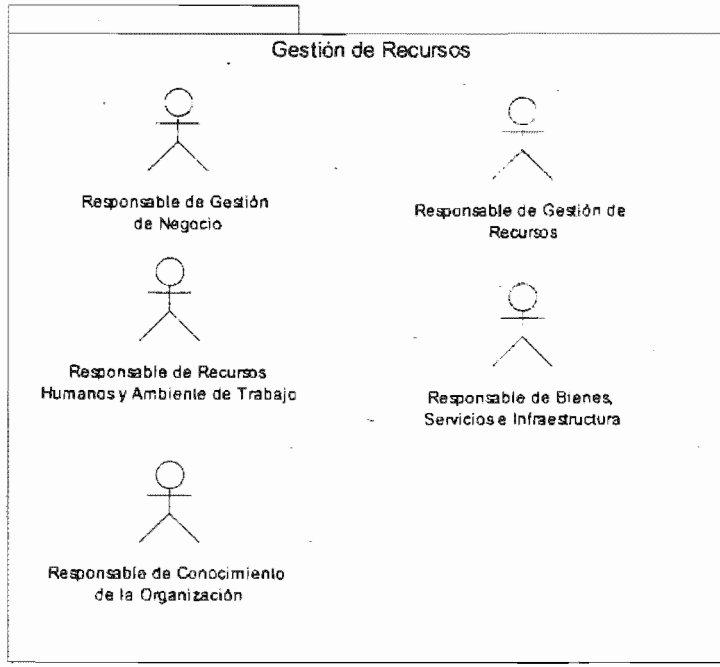


Figura 9. Actores involucrados en la Categoría de Gestión de Recursos.

En la Figura 10, se visualiza a los actores involucrados en realizar las actividades y cumplir con los objetivos marcados en la subcategoría de *Recursos Humanos y Ambientes de Trabajo*. La descripción de las características que presenta cada uno de estos roles se describe a continuación:

Rol	Nomenclatura	Características
Responsable de Gestión de Recursos	RGR	Conocimiento de las actividades necesarias para planear exitosamente el subproceso de Recursos Humanos y Ambiente de Trabajo
Responsable de Recursos Humanos y Ambiente de Trabajo	RRHAT	Conocimiento de las actividades necesarias para implantar exitosamente el subproceso de Recursos Humanos y Ambiente de Trabajo.
Responsable de Capacitación	RC	Conocimiento de las actividades necesarias para implantar exitosamente la capacitación solicitada.

Cabe mencionar las actividades generales que deben realizar cada uno de los actores involucrados en esta subcategoría entre las que se encuentran la preparación para la planeación y evaluaciones de desempeño dentro de la organización, la Instrumentación que conlleva la selección, asignación y aceptación de los recursos humanos. Por último la generación de reportes para la medición y sugerencias de mejora.

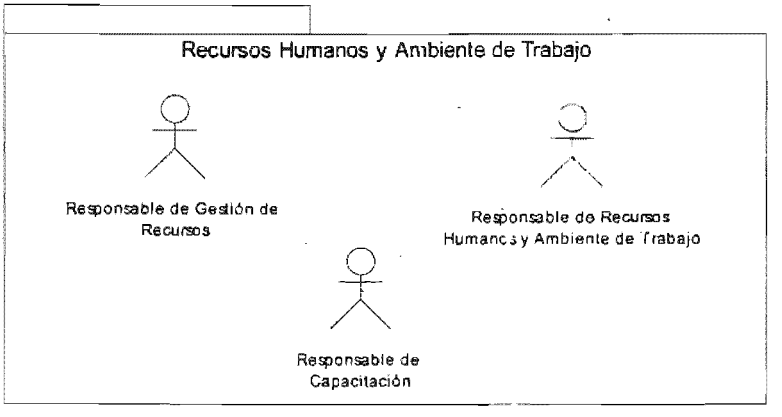


Figura 10. Actores involucrados en la Subcategoría de Recursos Humaos y Ambiente de Trabajo

La siguiente subcategoría corresponde a la de *Bienes, Servicios e Infraestructura*, representada en la Figura 11, cuyo fin es la de proporcionar a los proveedores de los propios bienes, servicios e infraestructura, con el fin de satisfacer los requisitos que demanden los proyectos. Dentro de estas actividades se ven involucrados únicamente dos actores los cuales se describen a continuación:

Rol	Nomenclatura	Características
Responsable de Gestión de Recursos	RGR	Conocimiento de las actividades necesarias para planear el subproceso de Bienes, Servicios e Infraestructura.
Responsable de Bienes, Servicios e Infraestructura	RBSI	Conocimiento de las actividades necesarias para implantar el subproceso de Bienes, Servicios e Infraestructura.

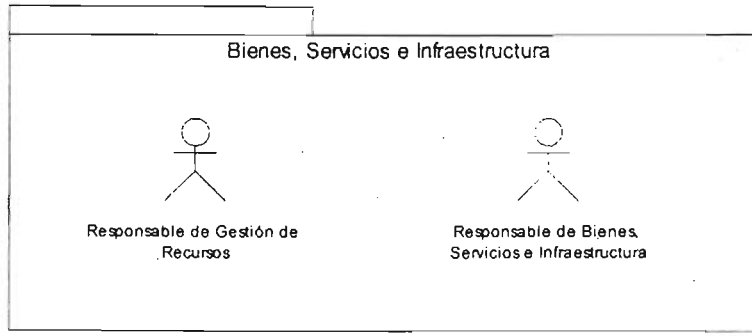


Figura 11. Actores involucrados en la Subcategoría de Bienes, Servicios e Infraestructura.

La última subcategoría corresponde al *Conocimiento de la Organización*, representada en la Figura 12, cuyo propósito es la de mantener disponible y administrar la Base del Conocimiento, en la que se alojan los productos generados por la organización. Entre estas actividades se ven involucrados los tres actores que conforman a esta, la descripción de cada actor se muestra a continuación:

Rol	Nomenclatura	Características
Responsable de Gestión de Recursos	RGR	Conocimiento en el esfuerzo necesario para la administración de la Base de Conocimiento.
Responsable del Conocimiento de la Organización	RCO	Conocimiento de definición y administración de repositorios documentales o automatizados.
Grupo de Responsables de Procesos	GRP	Conocimiento de necesidades del proceso con respecto a la Base de Conocimiento.

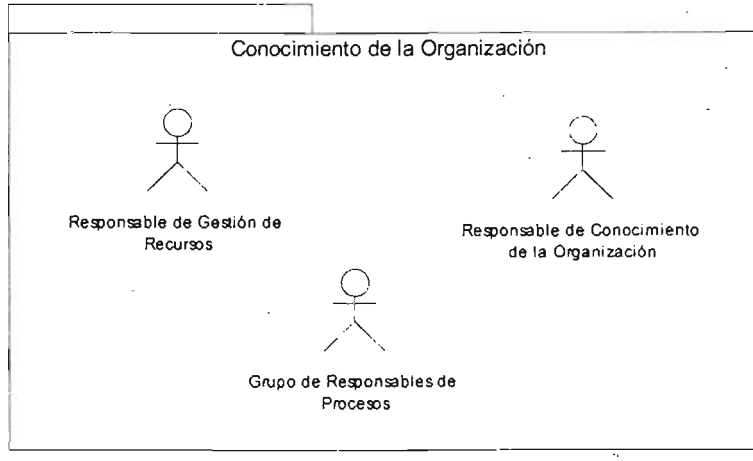


Figura 12. Actores involucrados en la Su categoría Conocimiento de la Organización

Hasta el momento se han descrito los actores involucrados en la categoría de *Alta dirección y Gestión*, a continuación se describirá la última categoría denominada *Operación* conformada por la *Administración de Proyectos Específicos* y el *Desarrollo y Mantenimiento de Software*.

La categoría de *Administración de Proyectos Específicos* tiene como objetivo establecer y llevar a cabo sistemáticamente actividades que permitan cumplir con los objetivos de un proyecto en tiempo y costo esperados, entre estas múltiples actividades se ven involucrados 5 actores mostrados en la Figura 13, la descripción de cada uno de los actores se describe a continuación:

Rol	Nomenclatura	Características
Responsable de Gestión de Proyectos	RGPY	Conocimiento sobre las actividades necesarias para llevar a cabo la gestión de proyectos.
Responsable de la Administración del Proyecto Específico	RAPE	Capacidad de liderazgo con experiencia e la toma de decisiones, planeación estratégica, manejo de personal, delegación y supervisión, finanzas y desarrollo de software.
Cliente	CL	Conocimiento en la experiencia de Solicitudes de Cambios.
Responsable del Subcontrato	RSC	Conocimiento en la administración de proyectos.
Responsable de Desarrollo y Mantenimiento de Software	RDM	Conocimiento y experiencia en el desarrollo y mantenimiento de software.

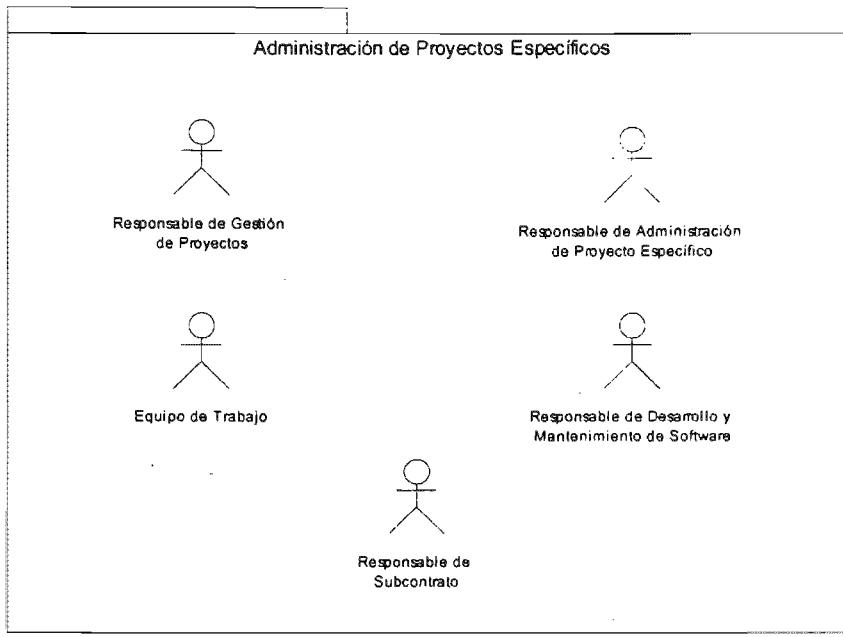


Figura 13. Actores involucrados en la Categoría Administración de Proyectos Específicos

La categoría de *Desarrollo y Mantenimiento de Software* tiene como objetivo llevar acabo las actividades de análisis, diseño, construcción, integración y pruebas de productos de software, entre estas múltiples actividades se ven involucrados 10 actores mostrados en la Figura 14, la descripción de cada uno de los actores se describe a continuación:

Rol	Nomenclatura	Características
Responsable de la Administración del Proyecto Específico	RAPE	Capacidad de liderazgo con experiencia e la toma de decisiones, planeación estratégica, manejo de personal y desarrollo de software.
Responsable de Desarrollo y Mantenimiento de Software	RDM	Conocimiento y experiencia en el desarrollo y mantenimiento de software.
Analista	AN	Conocimiento y experiencia en la obtención, especificación y análisis de los requerimientos.
Diseñador de Interfaz de Usuario	DIU	Conocimiento en diseño de interfaces de usuario y criterios ergonómicos.
Diseñador	DI	Conocimiento y experiencia en el diseño de la estructura de los componentes de

		software.
Programador	PR	Conocimiento y/o experiencia en la programación, integración y pruebas unitarias.
Responsable de Pruebas	RP	Conocimiento y experiencia en la planeación y realización de pruebas de integración y de sistema.
Revisor	RE	Conocimiento en las técnicas de revisión y experiencia en el desarrollo y mantenimiento de software.
Responsable de Manuales	RM	Conocimiento en las técnicas de redacción y experiencia en el desarrollo y mantenimiento de software.
Equipo de Trabajo	ET	Conocimiento y experiencia de acuerdo a su rol
Cliente	CL	Interpretación del estándar de la especificación de requerimientos.
Usuario	US	Ninguna

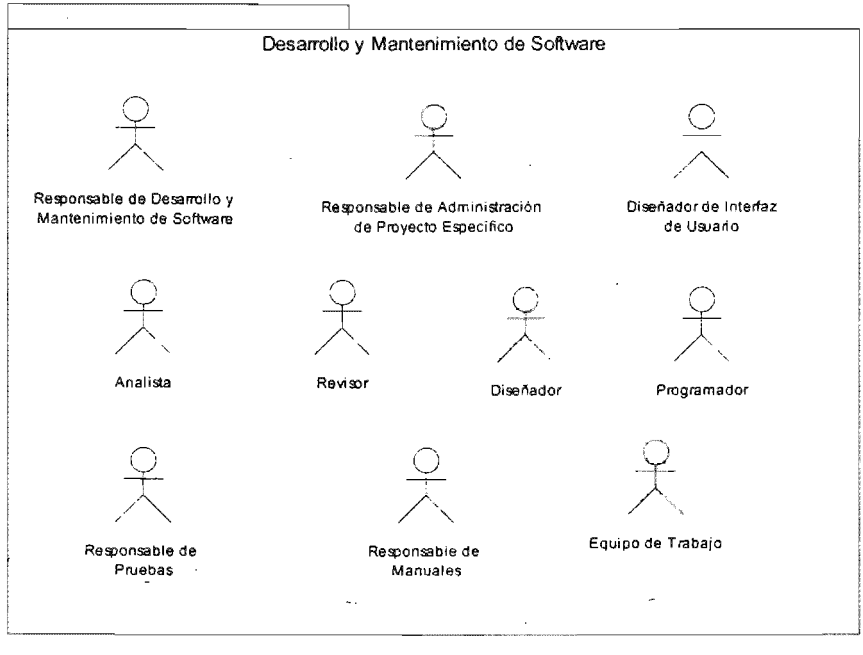


Figura 14. Actores involucrados en la Categoría de Desarrollo y Mantenimiento de Software

Cabe señalar la existencia de dos roles importantes que juegan un papel especial dentro de cada una de las a categorías y subcategorias que se han mencionado, del conjunto de actores detectados del modelo de procesos, un subconjunto de actores juega un rol adicional al que se fue conferido, se refiere al rol de *Verificador* y *Validador* de los productos que se van generando en cada etapa. En los siguientes diagramas Figura 15 y 16 se visualizan aquellos actores que adoptan un rol de verificador y validador.

El rol de *Verificador* tiene como función garantizar que los productos se apeguen a los estándares establecidos, no presenten ambigüedades y que cumpla con las normas de calidad.

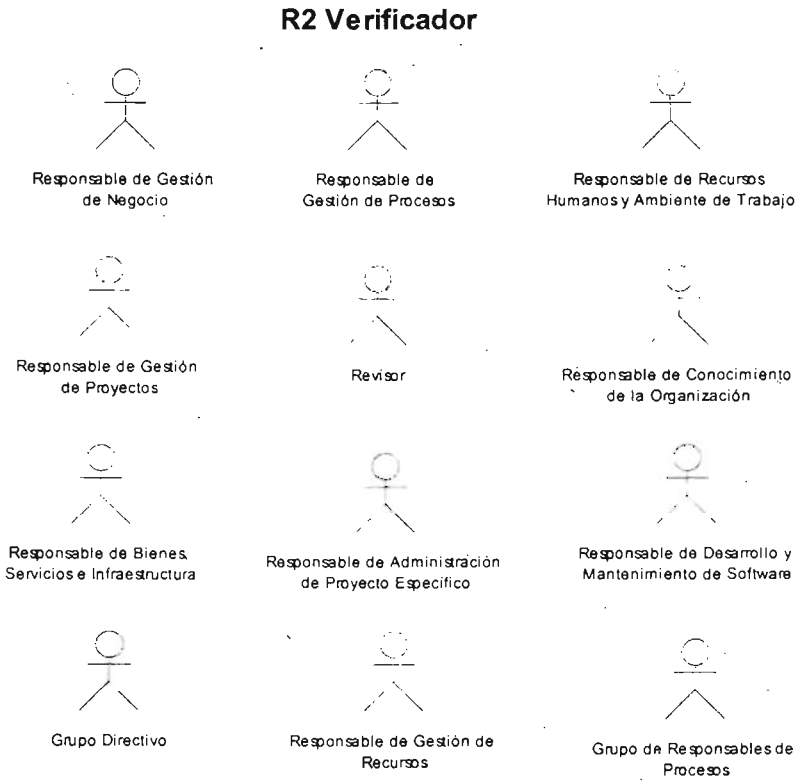


Figura 15. Actores que adoptan el rol de Verificador

El rol de *Validador* tiene como función confirmar que los producto cubran los requerimientos por los cual se genero y que se cumpla al 100% todos aquellos productos generados que conformarán los entregables internos o externos a la organización.

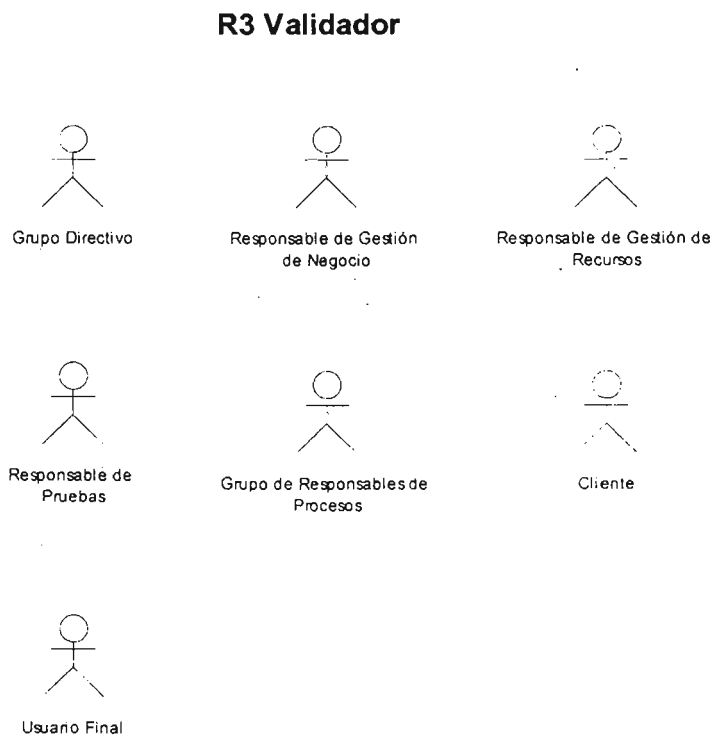


Figura 16. Actores de adoptan el rol de validador.

Se detectaron todos los posibles actores involucrados en el modelo de proceso, identificando en que categorías y procesos cada rol interviene, como las actividades que puede realizar. De este análisis se partirá para generar las Sentencias RDF las cuales permitirán controlar las actividades de cada rol, por medio de suposiciones lógicas.

4.3.4 Diagramas de Clases a nivel de Análisis

Antes de comenzar a describir los diagramas de clase y secuencia, cabe mencionar que el paso siguiente a la identificación de la funcionalidad que contendrá el sistema, el cual se realiza a través del modelado de los diagramas de casos de uso, consiste en realizar el detalle de cada uno de los casos de uso identificados como requerimiento funcional.

Una narrativa de casos de uso es un documento electrónico, con una plantilla bien definida, en la cual se describe las precondiciones, poscondiciones, escenario primario, escenario secundario, reglas de negocio, validaciones sobre datos y excepciones. En una narrativa se describe la interacción que existe entre el o los actores y el sistema, con el fin de abarcar la funcionalidad de la cual es responsable dicha narrativa (las narrativas de casos de uso de HIM se puede consultar en [CD]).

Una vez generada una narrativa, deberá pasar por lo controles de calidad, la cual tiene como función revisar los estándares de los documentos, evitar ambigüedades en la narrativa, realizar un revisión funcional a nivel de negocio.

Ya terminado el proceso de calidad, se comienza a realizar el análisis con modelado de clases y secuencia, que nos permitirá tener un entendimiento más claro del comportamiento del sistema.

Un diagrama de clases nos permite establecer la relación entre las posibles clases que existirán el sistema. La identificación de clases se obtiene de las narrativas de casos de uso.

Un diagrama de clases permite visualizar de manera estática la relación que existe entre las clases, permitiendo observar la asociación, dependencia, agregación, etc, por mencionar alguna de las relaciones que se pueden dar en nuestro diagrama.

Es importante aclarar que un diagrama de clases a nivel de análisis, no debe ser dependiente de una plataforma en particular.

Los estereotipos utilizados para la identificación de las clases, son los definidos en PU[14], conocidos como Vista, Control y Modelo. El manejo de estos estereotipos propuestos por PU es con el fin de realizar la identificación por capas conforme a un modelo MVC[15].

---

[14] Unified Process

[15] Model View Controller, para mayor referencia http://java.sun.com/blueprints/patterns/MVC-detailed.html

La representación gráfica para estos estereotipos es la siguiente:

A continuación se muestran los diagramas de clases correspondientes a los diagramas de casos de uso.

Diagrama de Clases para el caso de uso Ingresar al Sistema

En la Figura 17, se presentan las clases y sus relaciones, que permitirán abarcar toda la funcionalidad que requiere el ingreso al sistema. El diagrama muestra una clase *Vista* de Ingreso al sistema que depende de una clase *Control* Ingresar al Sistema, donde esta clase *Control* se asocia con otros controladores para la construcción de permisos, la autenticación del usuario y el registro en la bitácora.

Cada controlador se asocia a su vez con las clases *Modelo* para la consultar de información que se requiera.

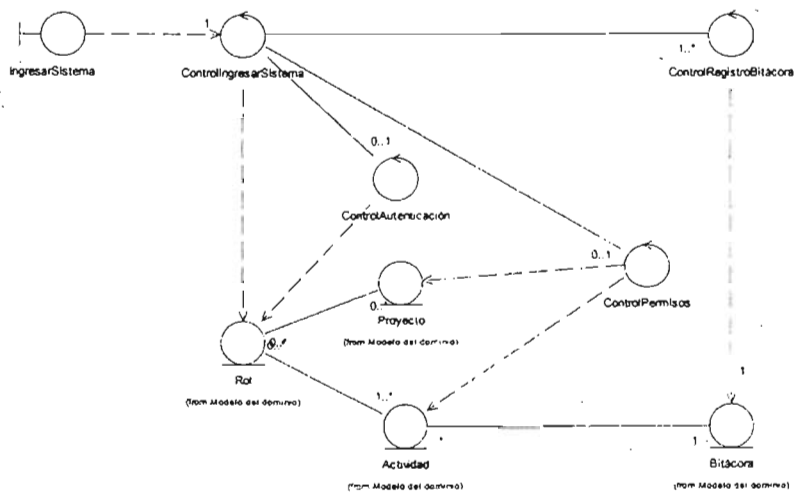


Figura 17. Diagrama de Clases – Ingresar al Sistema

Diagrama de Clases para el caso de uso Salir del Sistema

En la Figura 18, se presenta las clases que interacción para realizar la salida del sistema del usuario. Esta conformada por una clase *Vista* Salir del Sistema la cual presenta una relación de dependencia con la clase *Control* Salir del Sistema, donde esta última se asocia con otra clase controlador para realizar el registro de salida de los usuarios.

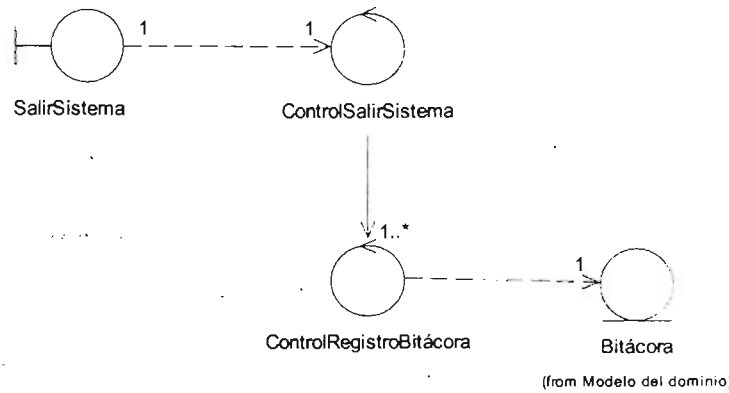


Figura 18. Diagrama de Clases – Salir del Sistema

Diagrama de Clases para el caso de uso Editar/Consultar Producto

En la Figura 19, se observan las clases que se requieren para poder llevar a cabo la consulta y edición de los productos que se van generando dentro de la herramienta HIM. Existe la clase *Vista* nombrada como *Consultar Producto* que presenta una asociación de dependencia con la clase *Control Consultar Producto*, donde esta última se asocia a la clase *Modelo Producto* para la obtención de la lista de productos existentes.

Una vez que el usuario seleccione el Producto que editará, el sistema deberá presentar en pantalla el producto en cuestión para su modificación, debido a ello existe la asociación entre las clases de *Control Consultar Producto* y la clase *Vista Editar Producto* que de igual forma esta última depende de la clase *Control Editar Producto* para encargase de la responsabilidad del flujo de control del negocio de esta funcionalidad. Para concluir existe el control encargado de realizar el registro de la actividad en la clase *Modelo Bitácora*.

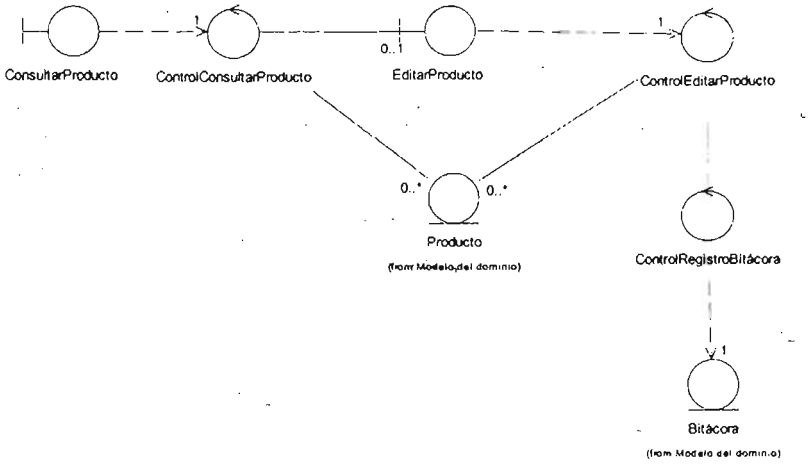


Figura 19. Diagrama de Clases – Editar/Consultar Producto

Diagrama de Clases del caso de uso Guardar Producto

Para finalizar con los diagramas de clases, presentamos en la Figura 20, presentamos las clases involucradas en el almacenamiento de un producto. Donde lo importante a destacar del diagrama es la asociación que existe entre la clase *Control Guardar Producto* y la clase *Modelo Base Conocimiento* la cual muestra el almacenamiento en el repositorio común del sistema.

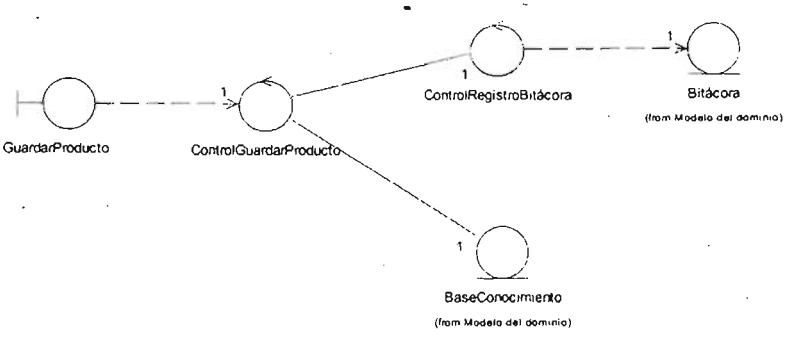


Figura 20. Diagrama de Clases – Guardar Producto

4.3.5 Diagramas de secuencia a nivel de análisis

Dentro de esta sección se mostrará la parte dinámica del sistema, ya que es la función que se pretende visualizar a través de un diagrama de secuencias. En un diagrama de secuencia se muestra en la parte superior izquierda al actor o actores involucrados, así como las clases participantes que intervienen en las distintas capas.

La interacción entre las clases y el actores se realiza a través del envío de mensajes, que expresa la parte dinámica entre la interacción a través de las capas de las clases.

A continuación se presentan los diagramas de secuencia correspondientes.

Diagrama de Secuencia para el caso de uso Ingresar al Sistema

En la Figura 21, se muestra la interacción a través del envío de mensajes entre las distintas capas, con el fin de autenticar al usuario y poder generar un marco de trabajo adecuado al perfil del rol.

Cuando el actor Usuario intenta realizar ingresar al sistema, se realizar como primer paso la verificación de la existencia del usuario en el sistema, en caso de ser satisfactoria esta validación se continua con el flujo de secuencia la consiste en realizar la autenticación de la contraseña, esta lógica de descifrar y validar la contraseña se encuentra considerada en la clase *Control Autenticación*.

Siguiendo el flujo y una vez autenticado al usuario, se realiza la construcción de las actividades que podrá realizar con base al rol que tiene asignado el usuario, para ello se requiere obtener de las entidades de nivel del negocio los proyectos y actividades relacionadas.

Con base a lo anterior se podrá realizar la construcción del marco de trabajo personalizado para el usuario, permitiendo así realizar solo las actividades que tenga permisos el rol.

Como último paso en la secuencia, consiste en reflejar en la bitácora el estado del intento de ingreso al sistema, en caso de ser exitoso o intentos fallidos para ingresar.

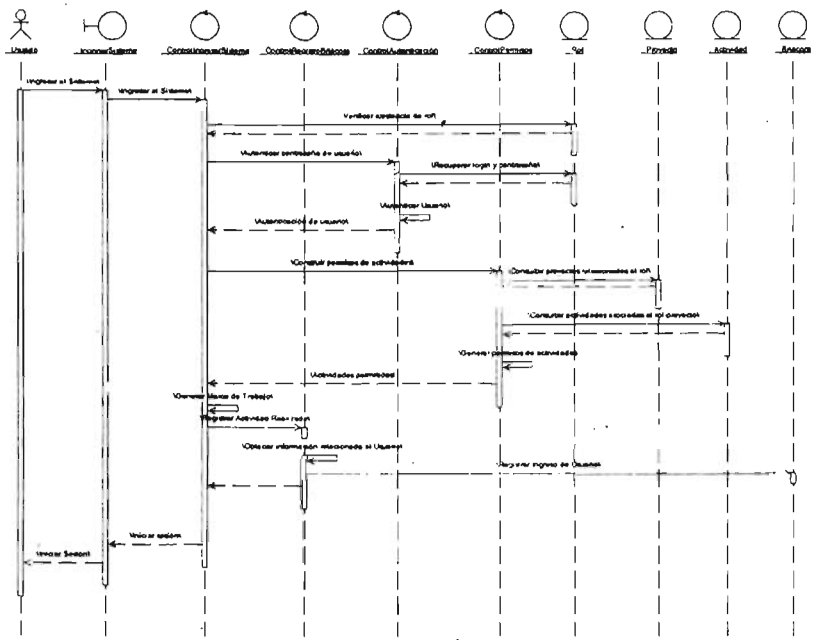


Figura 21. Diagrama de Secuencia relacionado al Ingreso al sistema

Diagrama de Secuencia para el caso de uso Salir del Sistema

El diagrama para salir del sistema, mostrado en la Figura 22, es bastante sencillo únicamente presenta la solicitud por parte del actor usuario por abandonar el sistema, que deberá ser registrada en la bitácora para tener conocimiento de que usuario, con qué rol y a que hora abandono el sistema, está información servirá como histórico.

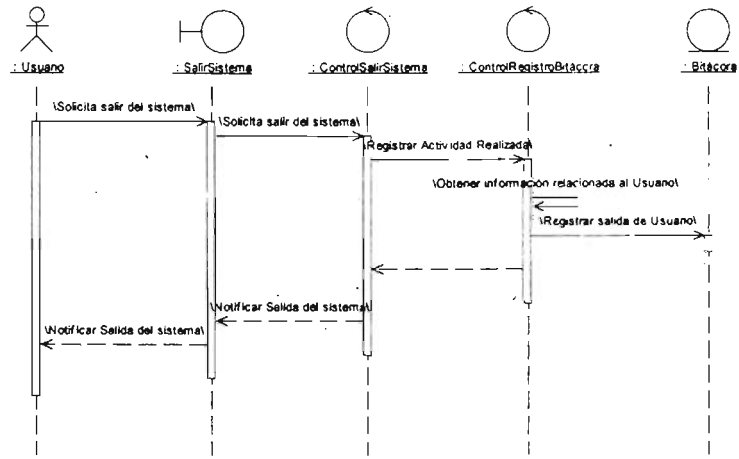


Figura 22. Diagrama de Secuencia relacionado a la salida del sistema.

Diagrama de Secuencia para el caso de uso Editar/Consultar Producto

Para el flujo de secuencia de la consulta y edición de un producto, mostrado en la Figura 23, inicia cuando el actor Rol solicita consultar los productos existentes para ellos se realiza el envío de mensajes a través de la capa de la Vista hacia el control para consultar del modelo de dominio los productos existentes y se le presentan al actor para su selección. El actor selecciona un producto que editará, cuando selecciona el producto comienza nuevamente la interacción entre las capas para permitir la edición del producto al actor.

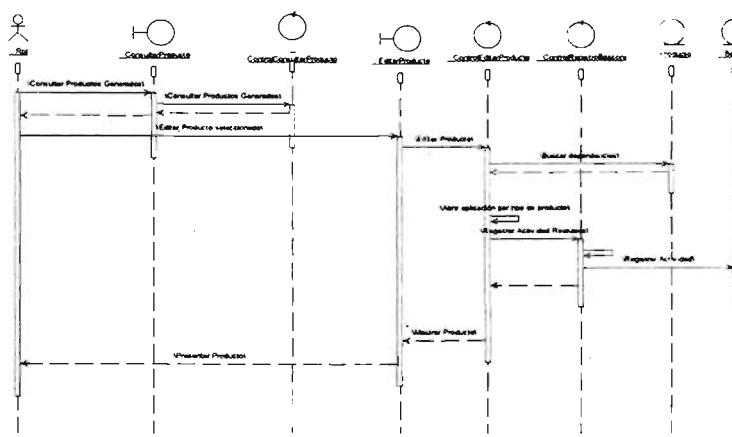


Figura 23. Diagrama de secuencia relacionado a la Edición y consulta de productos.

Diagrama de Secuencia para el caso de uso Guardar Producto

Dentro de los últimos diagramas de secuencia que presentamos, es el diagrama relacionado al almacenamiento de los productos, presentado en la siguiente Figura 24. En el diagrama se observa la interacción que realiza el actor con la clase *Vista Guardar Producto*, con la cual comienza la interacción entre clases a través de las capas. La siguientes secuencia consiste en que el sistema solicita al actor que proporciona la ubicación de donde se alojara el producto (en este caso podemos referirnos a un documento electrónico). Paso siguiente consiste en almacenar el producto en el repositorio común específicamente en la entidad conocida como base de conocimiento del modelo del dominio. Por último se registra la información relacionada al actor Rol y la actividad que acaba de realizar.

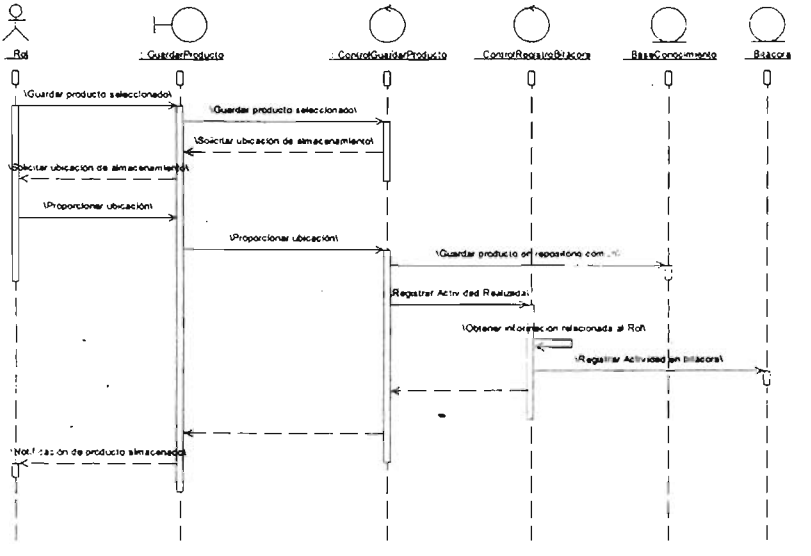


Figura 24. Diagrama de secuencia relacionado al almacenamiento de productos

4.4 Diseño sobre requerimientos funcionales de seguridad

Una vez realizado y culminado la etapa del análisis, podemos comenzar a realizar el diseño de nuestro sistema, donde los artefactos producidos en la etapa del análisis refiriéndonos a los diagramas de casos de uso, de clases y secuencia, serán el suministro y punto de inicio para poder iniciar nuestro diseño.

Además de los artefactos producidos en la etapa de análisis, existe otro artefacto generado en la etapa de análisis y madurado en la etapa de diseño, me refiero a la Arquitectura del sistema. El concepto de Arquitectura de un sistema de software, es ampliamente conocido y las ventajas que conllevan el uso y definición de una arquitectura adecuada a las necesidades del sistema y apegada a la tecnología utilizada.

Las ventajas que presenta la definición de una buena Arquitectura[16] son las siguientes:

- Modular a nivel de componentes
- Escalabilidad
- Definición de nivel de capas.
- Uso de patrones.

Una Arquitectura hace el manejo y presenta las características antes mencionadas, estructurando la arquitectura a nivel de capas separando la responsabilidad del despliegue de datos, de la lógica y reglas de negocio, así como la separación de la persistencia e integridad de los datos.

Dentro de cada uno de los niveles de capas que se hayan definido para la Arquitectura, se debe descomponer en componentes que tienen bien definida su funcionalidad y en su caso su servicio interno o hacia otra capa de la Arquitectura.

Con la intensión de generar una arquitectura robusta y escalable debemos hacer uso de los patrones en sus distintos géneros ya sean patrones de análisis, diseño y de arquitectura. Recordando un patrón es una propuesta de solución a un problema específico, donde ya ha sido probada la propuesta de solución. Conjuntando estas cualidades debemos tener una Arquitectura confiable y estable que dará soporte a nuestro sistema.

¿Por qué es importante la Arquitectura para la etapa de diseño?, desde la etapa de análisis se inicia el correspondiente análisis y propuestas de prototipos de la Arquitectura. Todos los artefactos que se generaron en la etapa del análisis son completamente independientes de cualquier plataforma y tecnología.

---

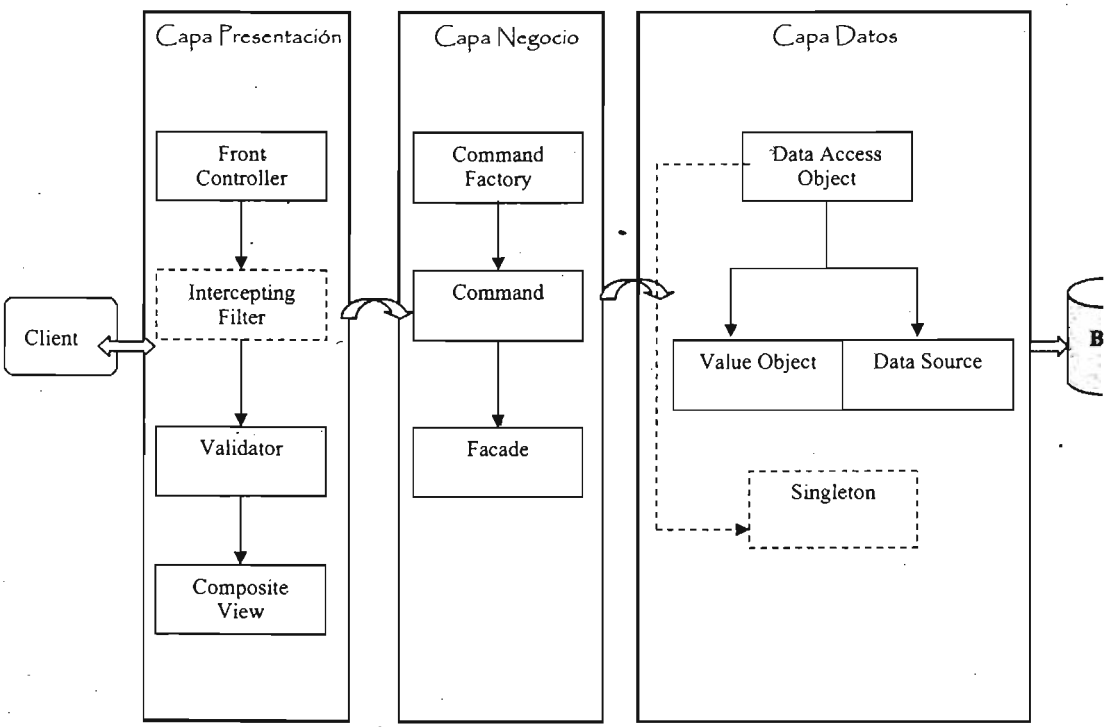
[16] [Zurita]

Una Arquitectura debe estar apegada a la tecnología que se utilizará para su construcción, por lo general muchos componentes que conforman a una arquitectura se apegan a tecnologías con la finalidad de solucionar requerimientos específicos.

En la etapa de diseño tiene como objetivo acoplar todo el análisis que hemos realizado específicamente los diagramas de clases y secuencia, a la arquitectura propuesta. Esto quiere decir que realizaremos el diseño a clases específicas de la arquitectura y tecnología.

A continuación en la Figura 25 se muestra la Arquitectura propuesta y utilizada para el sistema HIM. La Arquitectura HIM, esta formada por las tres capas básicas que son la capa de presentación, de negocio y de datos. Dentro de cada una de estas capas se encuentra conformada por distintos patrones GoF y algunos conocidos como Core de J2EE.

Dentro de la siguiente figura se muestran en contorno punteado los patrones relacionados a la seguridad que fueron implementados, dentro de la capa de la vista se encuentra el patrón *Intercepting Filter,* el cual tiene como fin realizar la autenticación de los usuarios y el control de actividades. El otro patrón es una mezcla de patrones entre un *Data Access Object* y un *Singleton*, el cual tiene la responsabilidad de realizar el registro de todas las actividades en la bitácora XML.



4.4.1 Diagramas de clases de diseño

Como ya se comento anteriormente, tomaremos como base los diagramas de clases de la etapa del análisis para diseñar los propios de esta etapa, aterrizando el diseño a la arquitectura definida para el sistema HIM.

4.4.1.1 Diagrama de clases a nivel diseño para el caso de uso Ingresar al sistema

Recordando un diagrama de clases tiene como fin mostrar las clases participantes y sus relaciones. Para un mejor entendimiento, se clasificaran las clases que se relacionan en el ingreso del sistema y control de permisos; en las tres capas principales a las cuales pertenecen:

Capa Vista		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
VMIngresarSistema	JSP	Composite View
ActionServlet	*Clase interna de Struts*	
IngresarSistemaAction	Class / extends ActionServlet	Front Controller
ActionForm	*Clase interna de Struts*	
IngresarSistemaActionForm	Class / extends ActionForm	
AdministradorFiltros	Class	Intercepting Filter
CadenaFiltros	Class	Intercepting Filter
CMDFiltrado	Interface	Intercepting Filter
IFFiltroLogin	Class / implements CMDFiltrado	Intercepting Filter
IFFiltroPassword	Class / implements CMDFiltrado	Intercepting Filter
CipherClass	Class	
ManejoPermisos	Class	

Capa de Negocio		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
FabricaComandosHIM	Interface	Command Factory
DescripciónProyectoFabricaComandosHIM	Class/implements Fabrica ComandosHIM	Command Factory
CMDIngresar/Salir	Interface	Command
IngresarSistema		Command
BaseConocimiento		Facade

Capa de Datos		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
DAORegistroBitacora	Class	Data Access Object
MarcoGeneral	Class	
DAORDFProceso	Class	Data Access Object
DAORDFProyecto	Class	Data Access Object

Las clases de ActionServlet y ActionForm, son parte del framework Struts que se utilizó como apoyo en la construcción del Framework HIM, en la diagrama de clases se muestra con un color distinto a las demás clases que fueron implementadas y que forman parte del Framework HIM. Se agregaron las clases mencionadas al diagrama para su mejor comprensión en su lectura. El detalle de interacción entre clases se describirá a nivel de diagrama de secuencia, descrito en la sección 4.3.1.5.

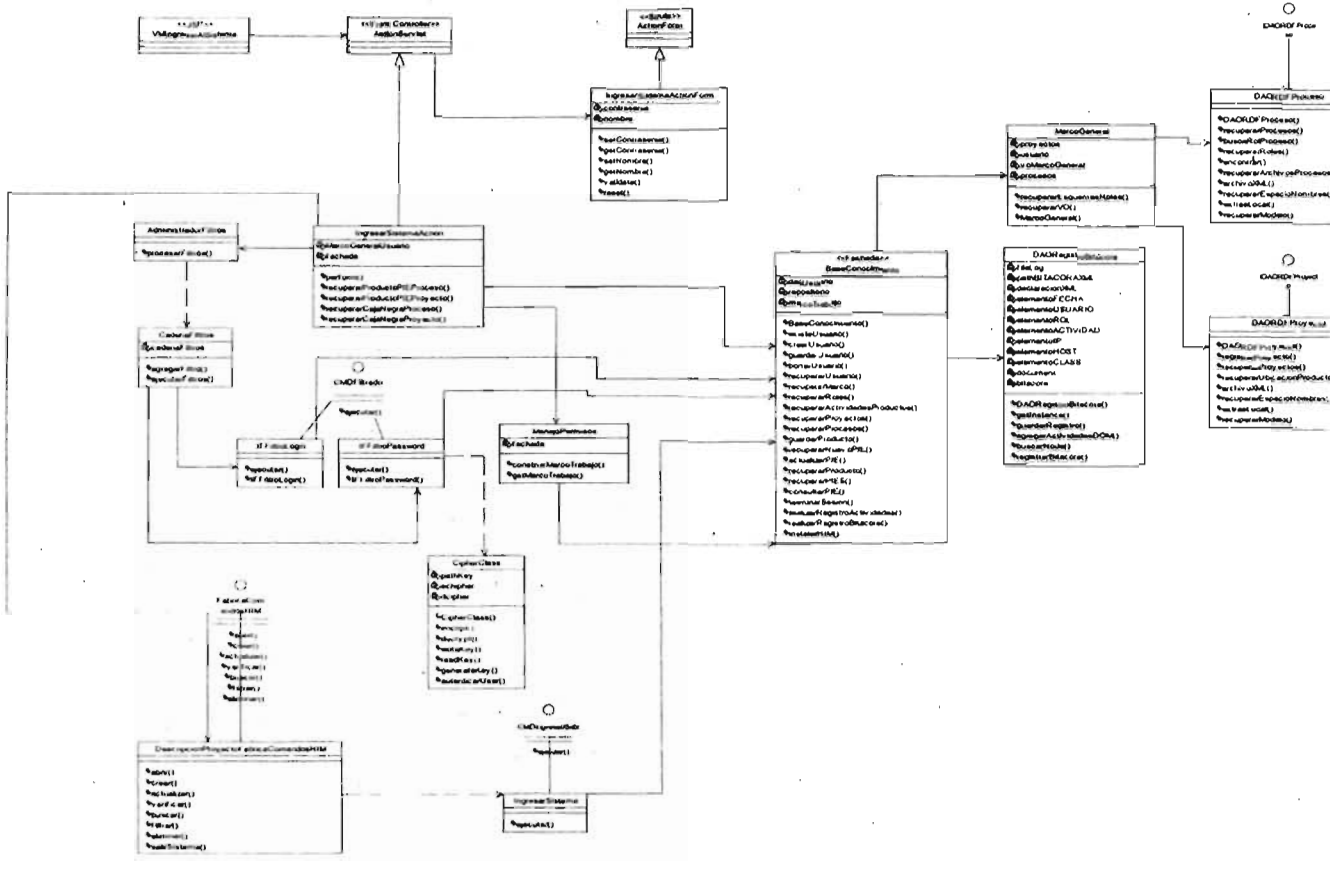


Figura 26. Diagrama de clases de diseño para Iugresar al Sistema

4.4.1.2 . Diagrama de clases a nivel diseño para el caso de uso Salir del sistema

En la Figura 27 se despliega el diagrama de clases relacionado al abandono del sistema, donde lo importante a destacar es el registro de la salida de quien y desde donde sale del sistema en su caso el rol.

Para un mejor entendimiento, se clasificaran las clases que intervienen en la salida del sistema en las tres capas principales a las cuales pertenecen:

Capa Vista		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
VMSalirSistema	JSP	Composite View
ActionServlet	*Clase interna de Struts*	
SalidaAction	Class / extends ActionServlet	Front Controller

Capa de Negocio		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
FabricaComandosHIM	Interface	Command Factory
DescripciónProyectoFabricaComandosHIM	Class/implements Fabrica ComandosHIM	Command Factory
CMDIngresar/Salir	Interface	Command
SalirSistema		Command
BaseConocimiento		Facade

Capa de Datos		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
DAORegistroBitacora	Class	Data Access Object

Dentro del diagrama de clases podemos observar la asociación de generalización entre la clase *ActionServlet* y la clase *SalidaAction*, donde la clase *ActionServlet* forma parte del framework de Struts y no se implemento dentro de nuestra construcción. La clase *SalidaAction* tiene como responsabilidad atender todas las peticiones par abandonar el sistema y de llevar la lógica de comunicación con la capa de negocio a través de la asociación con la clase *DescripciónProyectoFabricaComandosHIM*.

La clase *DescripciónProyectoFabricaComandosHIM* es la encargada de instanciar el comando adecuado en su caso la clase *SalirSistema* que a su vez se comunicará con la clase fachada *BaseConocimiento* que es la responsable de comunicarse con la capa de negocio con los DAO's, en su caso la clase *DAORegistroBitacora* la cual es responsable por dejar registro en el archivo XML.

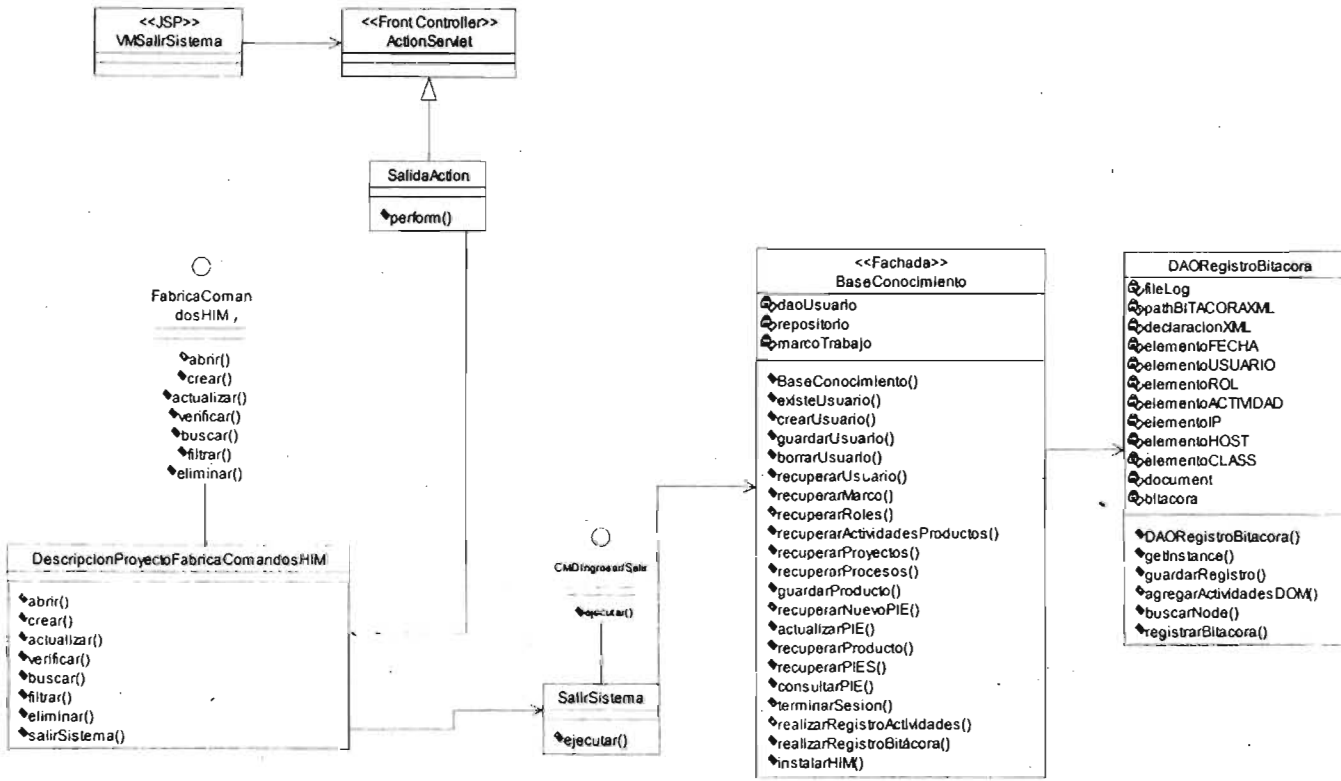


Figura 27. Diagrama de clases de diseño para salir del sistema

4.4.1.3 Diagrama de clases a nivel diseño para el caso de uso Consultar Descripción de Proyecto

Dentro de las tres estructuras básicas en que se conforma MoProSoft presenta una serie de actividades que conforman al flujo de procesos. Queremos notar que en la etapa del análisis se generalizo la tarea de realizar actividades, no importando a que escala se realizaba o mejor dicho en que categoría del modelo ni que actividad implícitamente realizamos.

Para la etapa del diseño se tomo una actividad representativa dentro de la serie de actividades que pueden existir dentro de un flujo de proceso. En su caso nos referimos a la actividad de creación, actualización y/o consulta de la descripción o detalle de un proyecto específico. Donde se pretende mostrar aquellas clases involucradas para llevar acabo esta actividad y el control de seguridad para mantener un registro de bitácora consistente.

En la siguiente Figura 28, se despliega el diagrama relacionada a la actividad de consulta de descripción de proyecto, presentando clases participantes y su relación entre ellas.
Para un mejor entendimiento, se clasificaran las clases que intervienen en la consulta de la descripción de proyecto, en las tres capas principales a las cuales pertenecen:

Capa Vista		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
ConsultarDescripciónProyecto	JSP	Composite View
ActionServlet	Clase interna de Struts	
ConsultarDescripciónProyectoAction	Class/extends de ActionServlet	Front Controller
ActionForm	Clase interna de Struts	
ContultarDescripciónProyectoActionForm	Class/extends de ActionForm	

Capa de Negocio		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
FabricaComandosHIM	Interface	Command Factory
DescripciónProyectoFabricaComandosHIM	Class/implements Fabrica ComandosHIM	Command Factory
CMDBusqueda	Interface	Command
BuscarDescripciónProyecto	Class/implements CMDBusqueda	Command
BaseConocimiento	Class	Facade

Capa de Datos		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
DAORegistroBitacora	Class	Data Access Object
IDAORDFPIE	Interface	Data Access Object
DAORDFPIE	Class/implements IDAORDFPIE	Data Access Object
VOProducto	Abstract	Value Object
VOProductoPIE	Class/extends *VOProducto*	Value Object

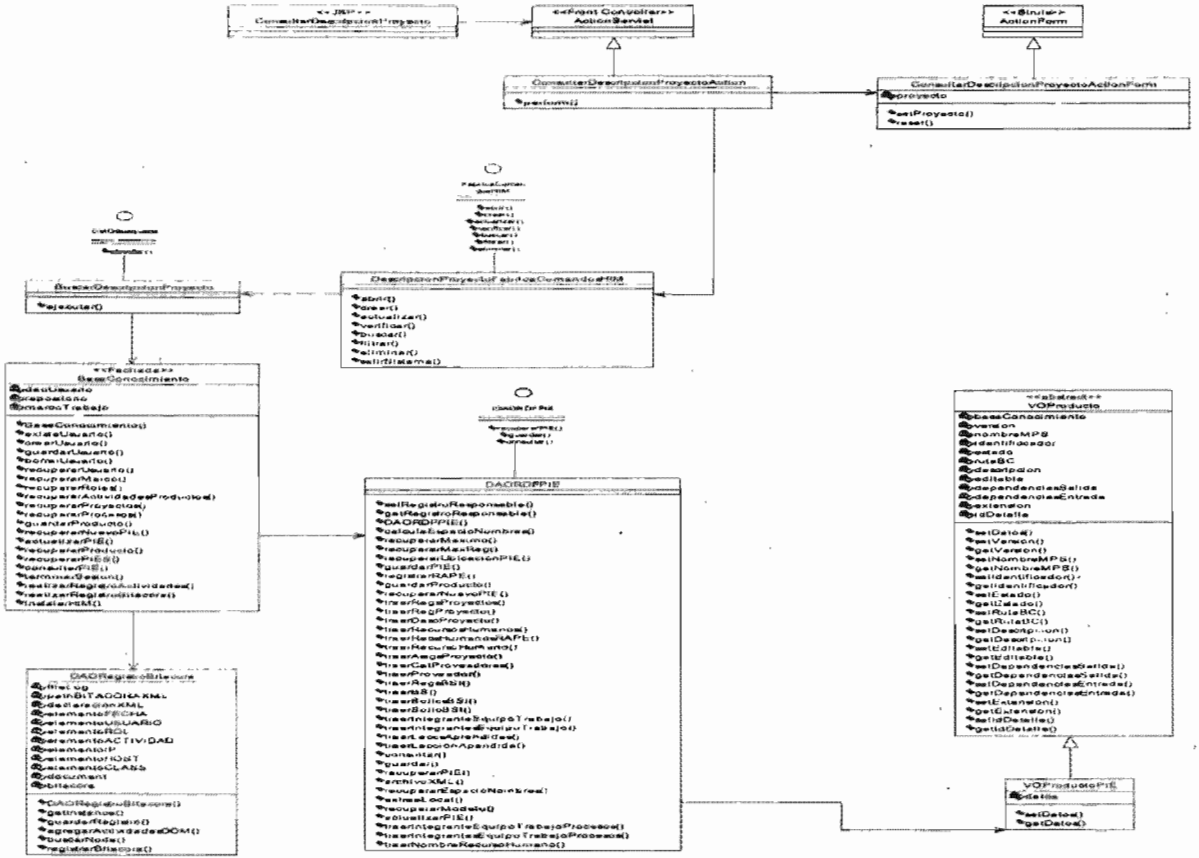


Figura 28. Diagrama de clases de diseño Consultar Descripción de Proyecto

4.4.1.4 Diagrama de clases a nivel diseño para el caso de uso Generar/Modificar Descripción de Proyecto

Como se menciono en la sección anterior se tomo el caso particular de la actividad de creación y/o actualización de la descripción de proyecto. La cual se describirá a continuación en la siguiente Figura 29. Para un mejor entendimiento, se clasificaran las clases que intervienen en la creación y/o modificación del detalle de proyecto, en las tres capas principales de la Arquitectura HIM:

Capa Vista		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
CrearDescipcionProyecto	JSP	Composite View
ActionServlet	*Clase interna de Struts*	
CrearDescripcionDelProyectoAction	Class/extendí de ActionServlet	Front Controller
ActionForm	*Clase interna de Struts*	
CrearDescripcionProyectoActionForm	Class/extendí de ActionForm	

Capa de Negocio		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
FabricaComandosHIM	Interface	Command Factory
DescripciónProyectoFabricaComandosHIM	Class/implements Fabrica ComandosHIM	Command Factory
CMDCrear	Interface	Command
CrearDescripciónProyecto	Class/implements CMDCrear	Command
BaseConocimiento	Class	Facade

Capa de Datos		
Nombre Clase	**Tipo de Clase**	**Patrón Relacionado**
DAORegistroBitacora	Class	Data Access Object/Singleton
IDAORDFProducto	Interface	Data Access Object
DAORDFProducto	Class/implements *IDAORDFProducto*	Data Access Object
IDAORDFPIE	Interface	Data Access Object
DAORDFPIE	Class/implements *IDAORDFPIE*	Data Access Object
VOProducto	Abstract	Value Object
VOProductoCajaNegra	Class/extends VOProducto	Value Object
VOProductoPIE	Class/extends VOProducto	Value Object

Cabe hacer la señalización de que se homogenizo en un solo diagrama el realizar la tarea de creación y/o modificación, ya que ambas actividades presentan el mismo flujo de procesos y de cierta manera su comportamiento es el mismo.
La interacción a más detalle se describirá en su correspondiente diagrama de secuencia que se describe en la sección 4.3.1.8. del presente capítulo.

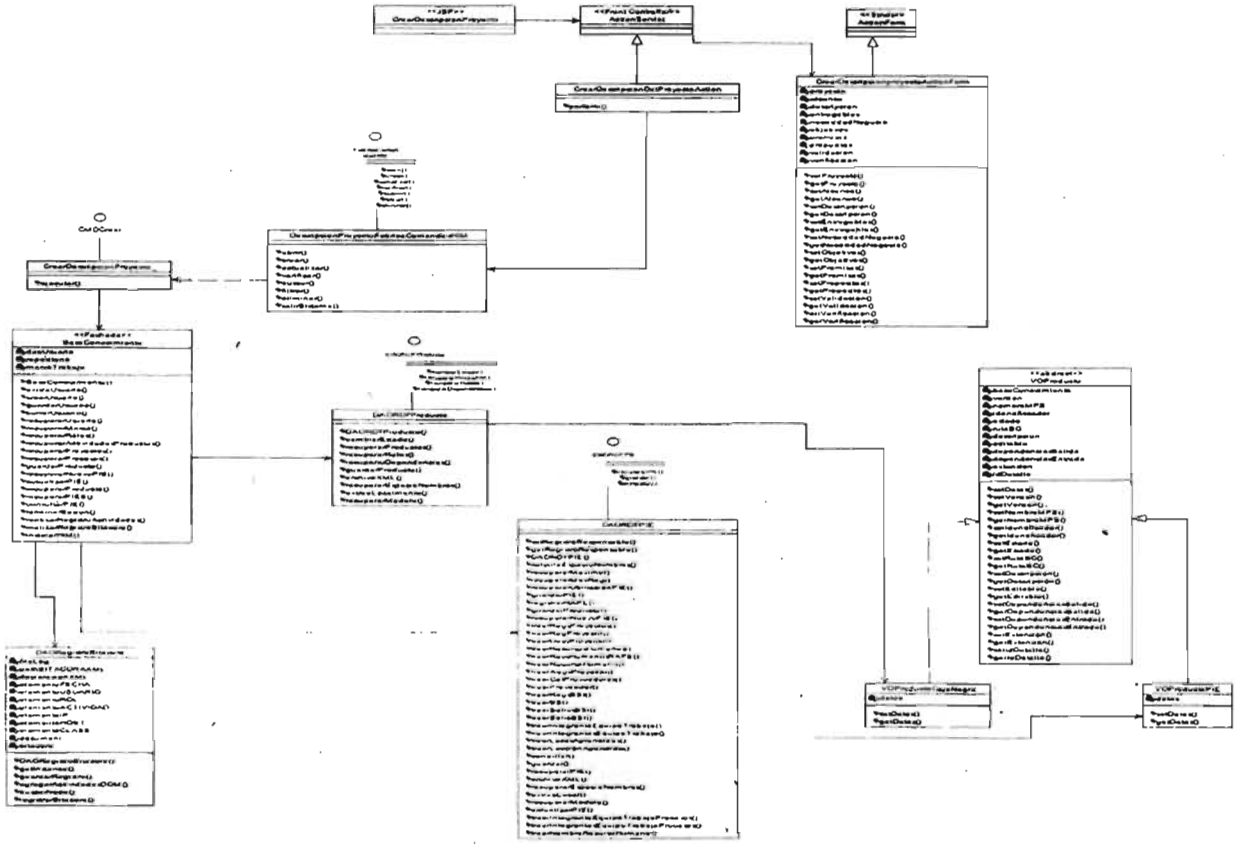


Figura 29. Diagrama de clases de diseño para Crear/Actualizar la descripción de proyecto.

4.4.1.5 Diagrama de secuencia a nivel de diseño para el caso de uso Ingresar al sistema

La interacción que se realiza entre las clases participantes se expresa a través de un diagrama de secuencia, el cual se describe a continuación. Todo inicia cuando el Actor Rol solicita ingresar al sistema HIM para lo cual se le presenta en el navegador la pantalla para ingresar al sistema la cual corresponde a la clase *JSP VMIngresarAlSistema*, solicitándolo su respectivo login y contraseña para poder ingresar.

Una vez que el Actor proporciona la información que le fue solicitada, contesta la solicitud para ingresar, la cual es atendida por el controlador central que en su caso es la clase *ActionServlet.* Esta clase es responsable de crear una instancia de la clase *IngresarSistemaActionForm* mapeando los datos definidos en la pantalla de ingreso al sistema al objeto de la clase. Posteriormente la clase ActionServlet deberá delegar a la clase *IngresarSistemaAction* para que sea ella la que atienda la solicitud realizada por el Actor.

La clase *IngresarSistemAction* es realmente la clase encargada de tener la lógica de negocio y el control sobre el flujo lógico que deberá llevarse, esta clase obtiene la referencia a la instancia de la clase *IngresarSistemaActionForm* con el fin de recuperar los datos que fueron proporcionados por el Actor. Además de contener los datos mapeados a un objeto de la clase *IngresarSistemaActionForm,* esta clase puede tener la responsabilidad de realizar la validación de la información, es decir, que no se encuentre vacío algún campo, que cumpla con el formato adecuado, etc.

El paso siguiente consiste en validar que el usuario (Actor), que intenta ingresar al sistema HIM realmente exista en el modelo de dominio, para lo cual la clase *IngresarSistemAction,* obtiene una instancia de la clase *AdministradorFiltros,* cuya responsabilidad es de obtener una cadena de filtros que contenga la secuencia de filtros los cuales deberán realizar la validación que exista usuario en el sistema. Para ello la clase *AdministradorFiltros,* crea una instancia de la clase *CadenaFiltros* que a su vez podrá tener un conjunto de filtros con un orden lógico si es que se requiere de filtros particulares, en su caso únicamente crea y obtiene la instancia de la clase *IFFiltroLogin.*

Al final la clase IngresarSistemaAction obtendrá la respuesta de si existe o no el usuario para continuar con el flujo, supongamos que la respuesta fue afirmativa. Para ello dentro la secuencia el siguiente paso consiste en Autenticar al usuario, para ello nuevamente se crea una instancia de la clase *AdministradorFiltros,* pero en su caso con la cadena de filtros que realicen la tarea de Autenticación. Nuevamente la clase *AdministradorFiltros,* crea la cadena de filtros correspondiente en su caso nos referimos a la clase *CadenaFiltros* que obtiene una instancia de la clase *IFFiltroPassword*; esta última tiene la responsabilidad de recuperar la información relaciona al usuario principalmente la contraseña que se relaciona.

Una vez recuperada esta información procede a realizar el desciframiento de la contraseña para si final autenticación del usuario. Para tal función se apoya de la clase *CipherClass*.

Una vez ya autenticado el usuario, el siguiente paso consiste en construir el Marco de Trabajo para el usuario, controlando las actividades que podrá realizar el rol, el Marco de Trabajo deberá estar personalizado mostrando únicamente aquellas actividades permitidas.

Para realizar la construcción del Marco de Trabajo la clase *IngresarSistemaAction*, solicita el servicio a la clase *ManejoPermisos* para realizar la construcción del Marco de Trabajo para el usuario con las actividades que le correspondan de acuerdo al rol que presente. La clase *ManejoPermisos* realiza la comunicación con la clase fachada *BaseConocimiento*, para invocar la recuperación del esquema de roles, esto quiere decir que apoyará de la clase *MarcoGeneral* la cual con base al Proceso y Proyecto que este asociado el Rol involucrado llevará el control de actividades a las que esta asociado.

Para ello la clase *MarcoGeneral* hace uso de las clases *DAORDFProceso* y *DAORDFProyecto*, las cuales tiene como objetivo extraer de las suposiciones lógicas de RDF las actividades a las cuales tiene permiso el Rol de realizar y por lo tanto con las que se deberá realizar la construcción de su Marco de Trabajo.

Ya para finalizar la lógica de la que es responsable la clase *IngresarSistemaAction*, se realiza el registro en bitácora sobre ingreso al sistema por parte del usuario. Para lo cual se realiza la creación de la instancia de la clase comando *IngresarSistema* a través de su fábrica de creación de comandos la cual corresponde a la clase *DescripcionProyectoFabricaComandosHIM*.

A través de la ejecución del comando *IngresarSistema* se inicia la comunicación con la clase fachada *BaseConocimiento*, la que a su vez se apoya de la clase *DAORegistroBitacora* para realizar el acceso al modelo de dominio y dejar reflejado el registro de la actividad.

Como paso final en la secuencia es mostrarle al Actor su Marco de Trabajo, es decir el área de trabajo en la que podrá realizar sus actividades. El diagrama de secuencia correspondiente al ingreso al sistema se presenta en la siguiente Figura 30.

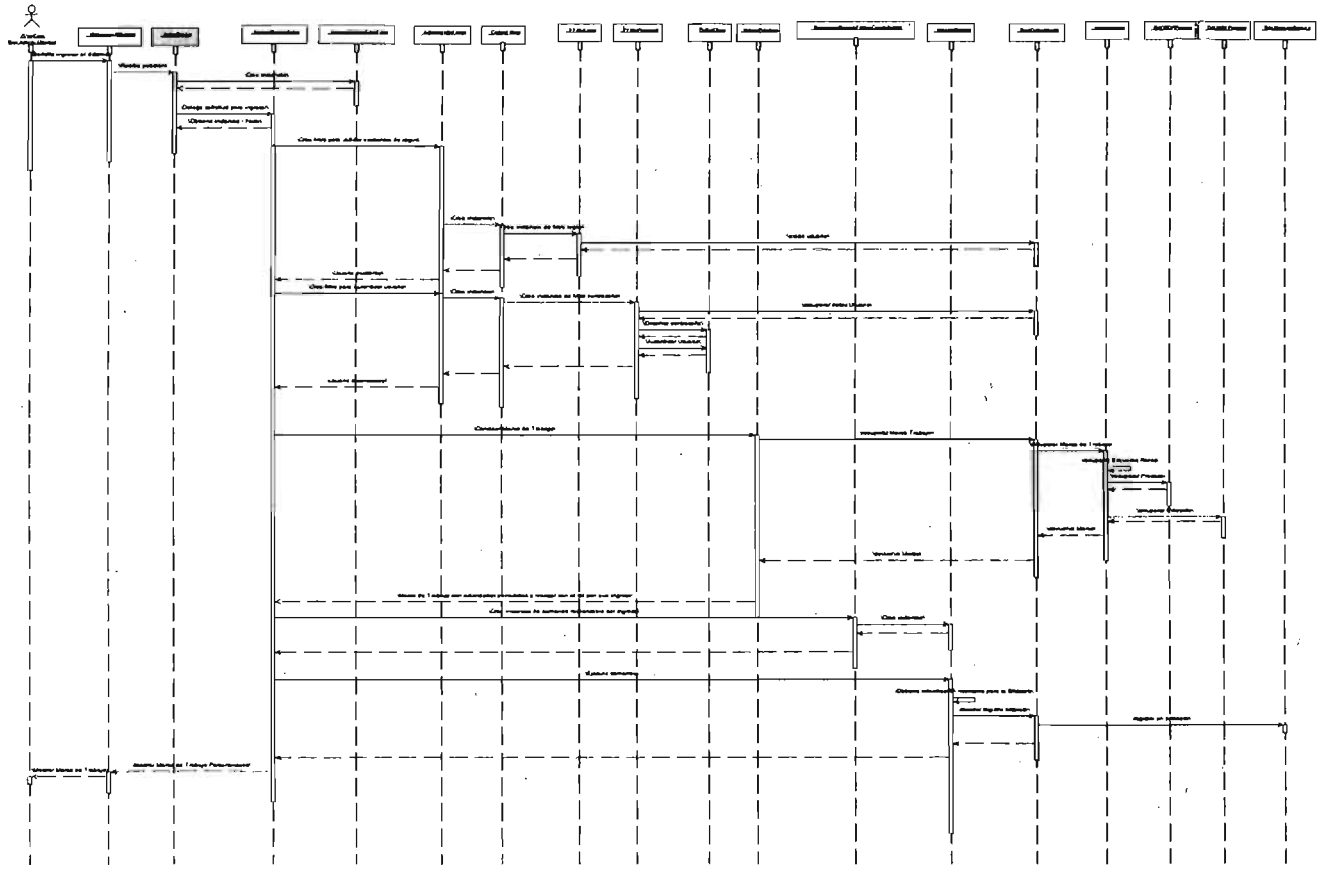


Figura 30. Diagrama de secuencias nivel diseño – Ingresar al sistema

4.4.1.6 Diagrama de secuencia a nivel de diseño para el caso de uso Salir del sistema

El diagrama de secuencia que se muestra en la Figura 31, comienza cuando el Actor solicita abandonar el sistema HIM, esta petición la realiza a través de la pantalla asociada a la clase JSP *VMSalirSistema* es atendida por el controlador central cuya clase es el *ActionServlet* del framework de Struts, el cual sirvió como base para la implantación del framework HIM.

La clase *ActionServlet* delega la responsabilidad de atender formalmente la petición a la clase *SalidaAction,* donde esta clase tiene como responsabilidad dejar en bitácora la salida de todos los usuarios. Para tal caso instancia el comando adecuado en realizar el registro de salida de sistema a través de la fábrica de comando cuyo nombre de clase es *DescripciónProyectoFabricaComandosHIM.*

Cuando desde la clase controladora *SalidaAction* se ejecuta el comando específico, clase comando *SalirSistema,* cuya clase comando tiene la responsabilidad de obtener la información requerida para el registro de bitácora, como lo son dirección IP remota, nombre usuario, rol, actividad realizada, por mencionar alguno de los datos que se almacenan en la bitácora.

Posteriormente se lleva acabo la comunicación con la clase fachada BaseConocimiento que a su vez hace uso de la clase *DAORegistroBitacora,* donde esta última fisicamente realiza el registro en el archivo XML creado para llevar el control de actividades.

Finalmente se le notifica al usuario su salida del sistema HIM.

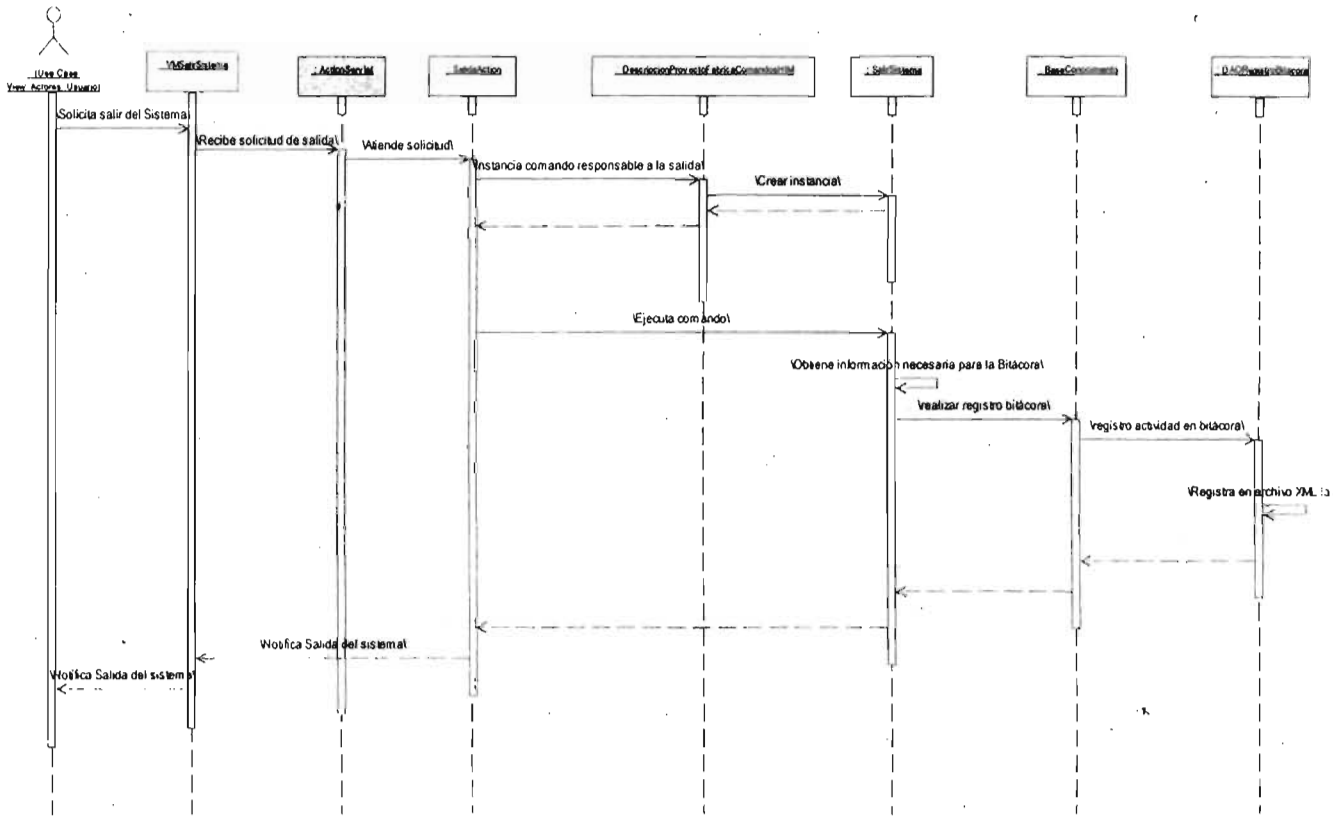


Figura 31. Diagrama de secuencia nivel diseño – Salir del Sistema

4.4.1.7 Diagrama de secuencia a nivel de diseño para el caso de uso Consultar detalle de Proyecto

El caso particular de realizar la actividad de consulta del detalle de un proyecto, se despliega en la Figura 32, en la cual inicia cuando el Actor solicita consultar el detalle del proyecto. A tal solicitud es atendida por el controlado centralizado cuya clase es *ActionServlet,* la cual canaliza y delega el atender dicha petición a la clase *ConsultarDescripciónProyectoAction,* cuya responsabilidad es tener la lógica necesaria para obtener la información de consulta y mostrarla al usuario.

El siguiente paso en la secuencia del diagrama consiste en la obtención de la instancia de la clase comando *BuscarDescripcionProyecto* por medio de la fábrica de comando cuya clase es *DescripciónProyectoFabricaComandosHIM.* Una vez obtenida la instancia del objeto se ejecuta el comando *BuscarDescripcionProyecto,* cuya clase se comunica con la fachada del framework para recuperar el Proyecto que se desea consultar.

La clase *BaseConocimiento* consulta la información relacionada al proyecto con ayuda de la clase *DAORDFPIE,* la cual consulta de las sentencias definidas en RDF la información del proyecto cargando la información actual en un objeto cuya clase es *VOProducto.* Se devuelve el valor del objeto para su posterior despliegue al usuario.

Nuevamente se repite el proceso de registro de bitácora, el comando *BuscarDescripciónProyecto*, se comunica con la clase fachada *BaseConocimiento* la cual a su vez realiza el acceso a datos por medio de la clase *DAORegistroBitácora.*

Como último paso la clase controladora redirige el valor del objeto que contiene el detalle del proyecto hacia pantalla para ser presentado al Actor.

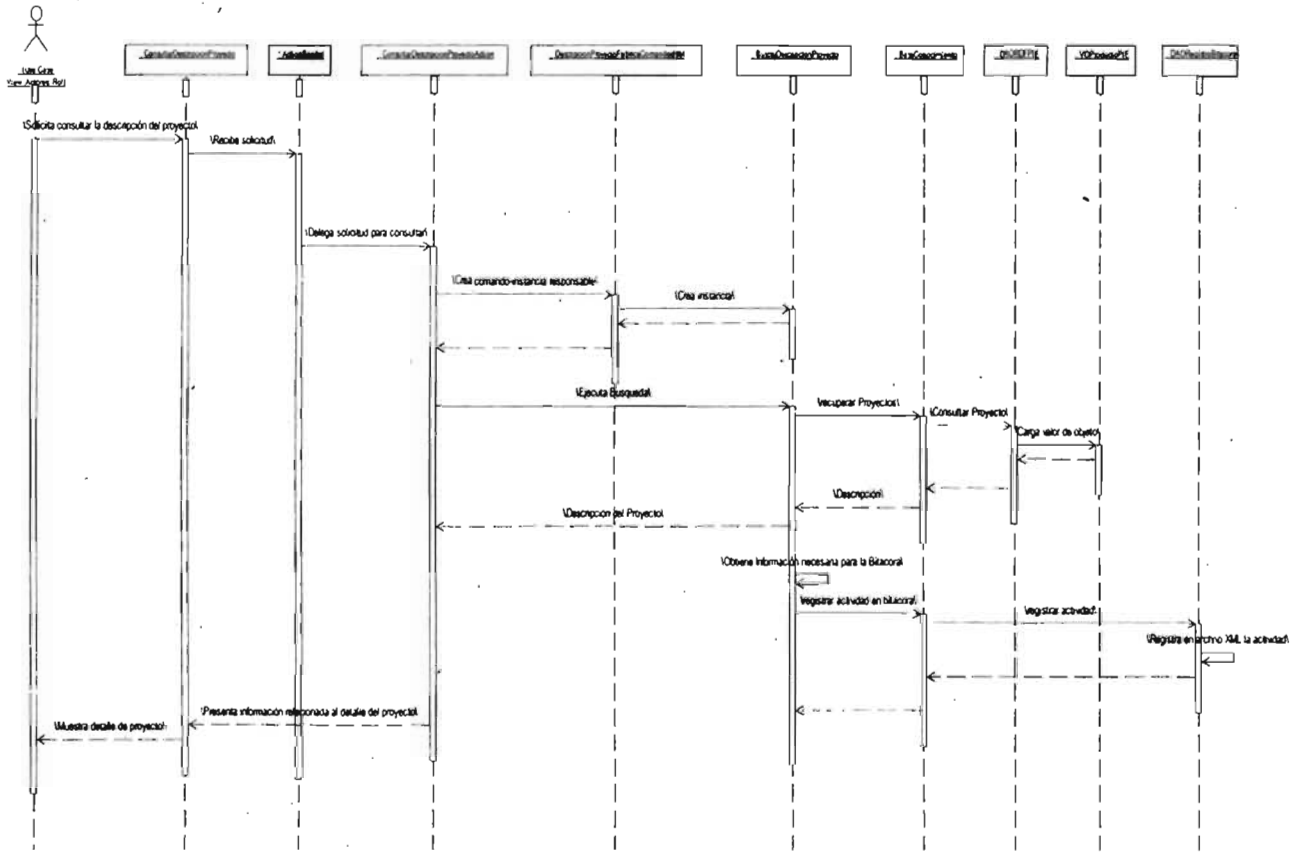


Figura 32. Diagrama de clases nivel diseño – Consultar detalle de proyecto

4.4.1.8 Diagrama de secuencia a nivel de diseño para el caso de uso Crear/Modificar descripción de Proyecto

Para completar los diagramas referentes al la actividad de detalle del proyecto, se muestra en la Figura 33, el diagrama de secuencia de creación y/o actualización del detalle del proyecto, el cual inicia cuando el actor solicita realizar la creación ó modificación de la información del detalle de proyecto.

La solicitud es recibida por la clase *ActionServlet* que crea una instancia de *CrearDescripciónDelProyectoAction* para mapear los datos recibidos del formulario proveniente del *JSPCrearDescripciónProyecto*.

Posteriormente el *ActionServlet* canaliza la responsabilidad de atender la petición a la clase *CrearDescripciónDelProyectoAction*, en la cual se contiene la lógica de control, para ello crea la clase comando *CrearDescripciónProyecto* a través de la fábrica de comandos nombrada *DescripciónProyectoFabricaComandoHIM*

Una vez que se obtiene la instancia del objeto comando se ejecuta con el fin de que la misma clase se comunique con la fachada *BaseConocimiento*, esta crea ó actualiza la información del proyecto por medio de la clase *DAORDFPI*.

Por último dentro de la responsabilidad del comando *CrearDescipciónProyecto* es la de obtener la información requerida para realizar el registro en bitácora y enviar esta información a la fachada *BaseConocimiento*. La clase *BaseConocimiento* se apoya del *DAORegistroBitácora* que actualiza el archivo XML cargando el árbol jerárquico del documento por medio del parser DOM, para agregar un nuevo registro de bitácora.

Como paso final se le notifica al actor la finalización de la creación o actualización del detalle del proyecto.

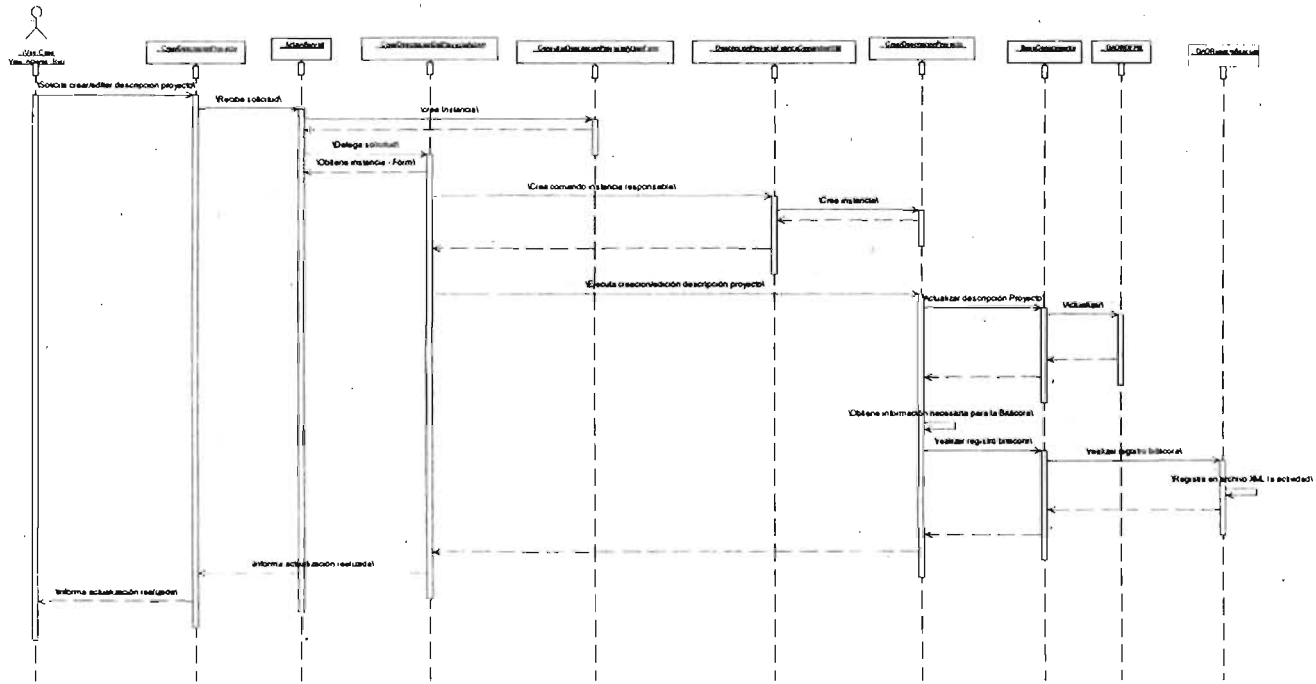


Figura 33. Diagrama de secuencia nivel diseño - Crear/Actualizar detalle del proyecto

Capítulo 5

Implementación de los mecanismos de Seguridad.

En este capítulo se describe la estructura de implantación de los componentes de seguridad que intervienen en la Herramienta Integral MoProSoft, así como la integración con el resto de los componentes del sistema. También se detalla la estructura física de los archivos que componen la BD-RDF y la bitácora.

5.1 Introducción

El objetivo que se pretenden alcanzar con los diagramas de componentes es plasmar la organización física del sistema. Para lo cual se requiere decidir como las clases serán organizadas, enfocándonos en los archivos físicos que darán la estructura del sistema de software.

Entre otras actividades que se deben considerar en la generación de los diagramas de componentes se encuentra la exploración de los tipos de componente y la generación de los mismos, realizando el mapeo de las clases que conformarán cada uno de los componentes.

Los tipos de componentes que pueden existir, dependerá del comportamiento y las características de las clases que conformen al componente. Por ejemplo, podemos encontrar componentes que alojan únicamente subprogramas que tienen definida una específica tarea y que realmente no conforman a una clase. Así también tendremos componentes que representan principalmente a módulos de software que tiene bien definida su *interface*.

Cabe recalcar que dentro del diagrama de componentes se debe reflejar la existencia de clases de tipo *interface*, cuyo símbolo es representado por un círculo amarillo.

El proceso que conllevan los diagramas de componentes, consistirá en mapear las clases representadas en los diagramas de clases de nivel de diseño, que fueron presentadas en el Capítulo 4, a un componente único por clase o quizás muchas clases conformen un solo componente.

En las siguientes secciones se presentan los diagramas de componentes correspondientes a los diagramas de clase de nivel de diseño.

5.2 Diagrama de componentes para el Ingreso al Sistema

En la siguiente Figura 1, que corresponde al diagrama de clases nivel de diseño mostrado en el Capítulo 4 sección 4.3.1.1, se muestra la relación entre componentes que conforman el realizar el ingreso al sistema.

Como se puede observar en la Figura 1, se muestra un transparente mapeo entre las clases definidas a nivel de diseño, exceptuando y recalcando el componente denominado Filtro de Autenticación, cuyo componente está conformado por la clase de *IFFiltroPassword* y la clase *ChipreClass*, cuyas clases se encuentran asociadas para cumplir con la Autenticación del usuario.

Por tal motivo estas dos clases se agrupan en un mismo componente. La representación de la relación entre los componentes es la denominada dependencia la cual consiste en una flecha con un cuerpo de línea punteada.

Como se aprecia en el diagrama se sigue conservando la representación entre un componente y su posible *interface* que implementa su clase o clases que conforman al componente. En tales casos encontramos la *Interface CMDFiltrado*, la cual define la estructura que deberá contener los comandos concretos que se encuentran agrupados en los componentes de *Filtro de Login* y *Filtro de Autenticación*.
A continuación se realiza una pequeña tabla donde se realiza una agrupación de que componente aloja que clases.

Nombre del Componente	***Nombre de la(s) clase(es)***
Ingresar al Sistema	i. <<JSP>>VMIngresarAlSistema
Ingresar al Sistema Action	i. <<FrontController>> ActionServlet ii. IngresarSistemaAction
Ingresar al Sistema Form	i. <<Struts>> ActionForm ii. IngresarSistemaActionForm
Administrador de Filtros	**i. AdministradorFiltros**
Cadena de Filtros	**i. CadenaFiltros**
Filtro de Login	**i. IFFiltroLogin**
Filtro de Autenticación	**i. IFFiltroPassword** **ii. CipherClass**
Control de Permisos	**i. ManejoPermisos**
Base de Conocimiento	i. <<Facade>> BaseConocimiento
Fabrica Concreta de Comandos	i. DescripcionProyectoFabricaComandosHIM
Comando Concreto Ingresar al Sistema	i. IngresarSistema
Marco General de Trabajo	**i. MarcoGeneral**
RDF Proceso	**i. DAORDFProceso**
RDF Proyecto	**i. DAORDFProyecto**
Bitácora	**i. DAORegistroBitacora**

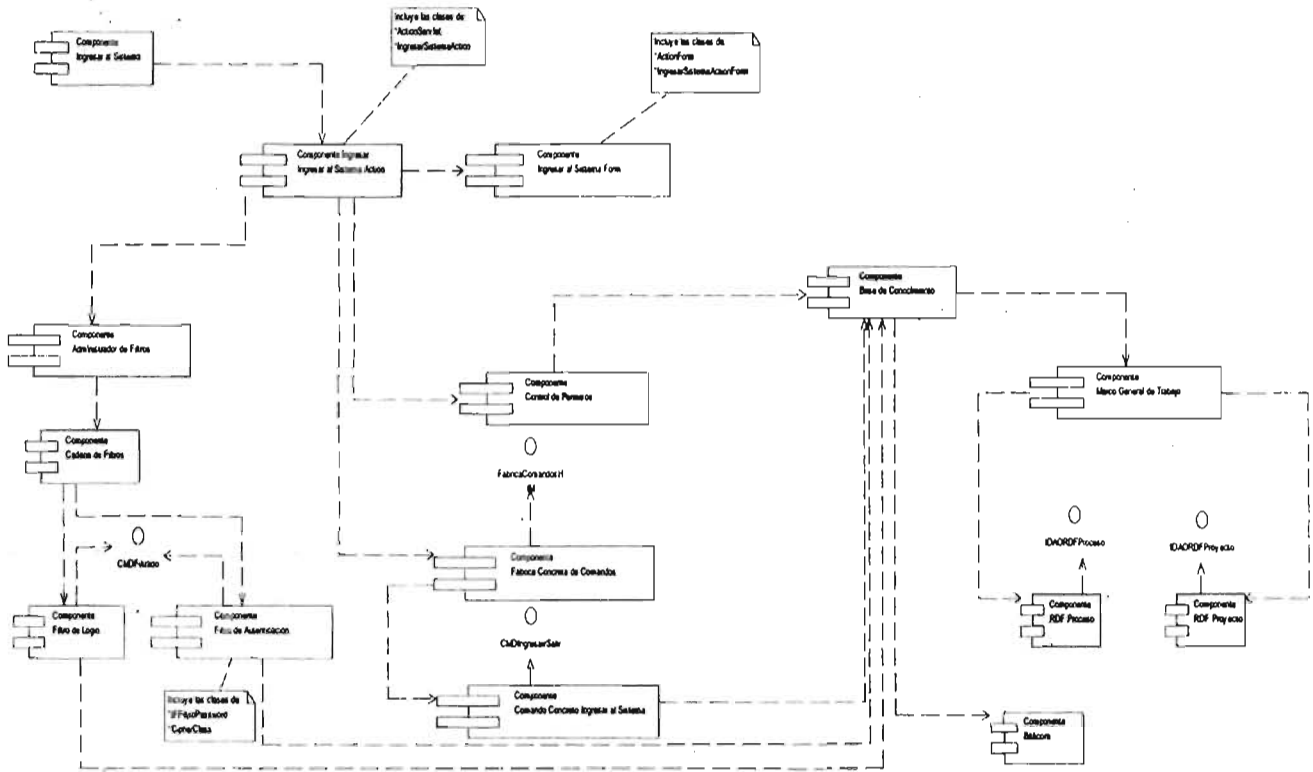


Figura 1. Diagrama de componentes para ingreso al sistema.

5.3 Diagrama de componentes para la Salida del Sistema

El diagrama de componentes que refleja al diagrama de clases de nivel de diseño del Capítulo 4 sección 4.3.1.2, se muestra en la siguiente Figura 2.Como se puede observar en el diagrama de componentes, existió un mapeo por cada una de las clases a un componente físico, respetando la dependencia existente con las *interfaces* y sus correspondientes componentes.

Entre las *interfaces* definidas se encuentran las siguientes:

- La *interface* denominada como *FabricaComandosHIM* que es implementada por la clase *DescripcionProyectoFabricaComandosHIM* la cual se encuentra físicamente representada por el componente nombrado como *Fábrica Concreta de Comandos.*
- De igual forma se encuentra definida dentro del diagrama la interface *CMDIngresar/Salir* la cual presenta una relación de dependencia con el componente que contiene la clase que la implementa, nos referimos al componente nombrado como *Comando Concreto Salir del Sistema.*

A continuación se realiza una pequeña tabla donde se realiza una agrupación de que componente aloja que clases.

Nombre del Componente	***Nombre de la(s) clase(es)***
Salir del Sistema	i. <<JSP>>VMSalirSistema
Salir del Sistema Action	i. <<FrontController>> ActionServlet ii. SalidaAction
Fabrica Concreta de Comandos	i. DescripcionProyectoFabricaComandosHIM
Comando Concreto para Salir del Sistema	i. SalirSistema
Base de Conocimiento	i. <<Facade>> BaseConocimiento
Bitácora	**i. DAORegistroBitacora**

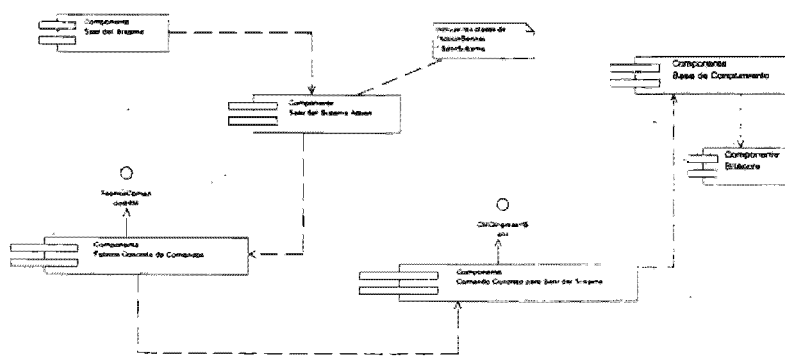


Figura 2. Diagrama de Componentes para la Salida del Sistema

5.4 Diagrama de componentes para realizar la actividad de consulta del detalle del proyecto

El siguiente diagrama de componentes presentado en la Figura 3, se representa la estructura física que se requiere para realizar la consulta del detalle de un proyecto. El diagrama de clases correspondiente fue presentado en el Capítulo 4 sección 4.3.1.3.

Como podemos observar en el diagrama de componentes, se sigue manteniendo la representación gráfica con las interfaces, como lo son el caso de la Interface *FabricaComandosHIM* que presenta una dependencia con el componente *Fabrica Concreta de Comandos.*

Así de igual forma la Interface *CMDBusqueda* que tiene una relación de dependencia con el componente *Comando Concreto para Consultar el Detalle del Proyecto.*

La relación de las clases que agrupa cada componente se detalla a continuación en la siguiente tabla.

Nombre del Componente	***Nombre de la(s) clase(es)***
Consultar la Descripción del Proyecto	i. <<JSP>> ConsultarDescripcionProyecto
Consultar la Descripción Proyecto Action	i. <<FrontController>> ActionServlet ii. ConsultarDescripcionProyectoAction
Consultar la Descripción Proyecto Action	i. <<Struts>> ActionForm ii. ConsultarDescripcionProyectoActionForm
Fabrica Concreta de Comandos	i. DescripcionProyectoFabricaComandosHIM
Comando Concreto para Consultar Detalle Proyecto	i. BuscarDescripcionProyecto
Base de Conocimiento	i. <<Facade>> BaseConocimiento
Bitácora	**i. DAORegistroBitacora**
RDF Pie	i. DAORDFPIE ii. <<abstract>> VOProducto iii. VOProductoPIE

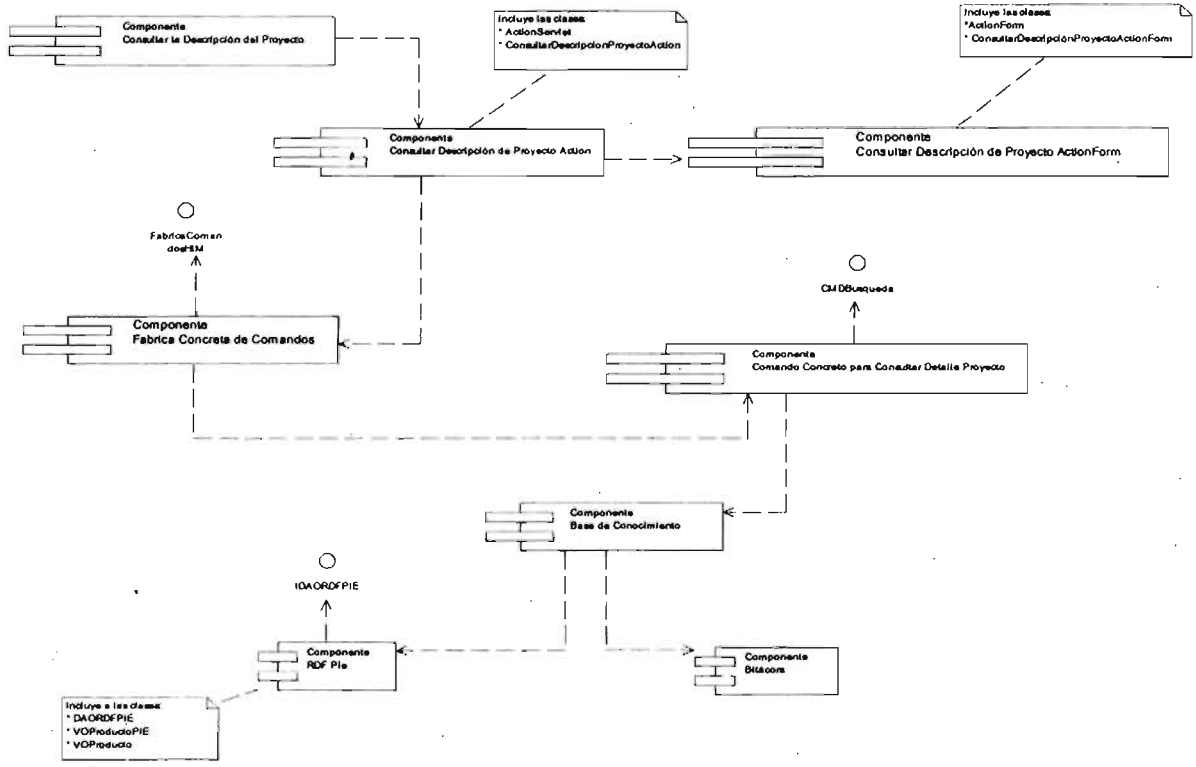


Figura 3. Diagrama de Componentes para realizar la consulta.

5.5 Diagrama de componentes para realizar la creación y/o actualización del detalle del proyecto.

El último de los diagramas de componentes constituye la creación y/o actualización de la descripción del proyecto, el diagrama de clases a nivel de diseño que se tomo como base fue presentado en el Capítulo 4 sección 4.3.1.4.

La estructura del diagrama es la misma que la del diagrama de componentes para realizar la consulta, claro que con las respectivos componentes que conforman la composición física de la creación y/o modificación.

La relación de las clases que agrupa cada componente se detalla a continuación en la siguiente tabla.

Nombre del Componente	***Nombre de la(s) clase(es)***
Crear/Actualizar Descripción del Proyecto	i. <<JSP>> CrearDescripcionProyecto
Crear/Actualizar Descripción Proyecto Action	i. <<FrontController>> ActionServlet ii. CrearDescripcionProyectoAction
Crear/Actualizar Descripción Proyecto Action	i. <<Struts>> ActionForm ii. CrearDescripcionProyectoActionForm
Fabrica Concreta de Comandos	i. DescripcionProyectoFabricaComandosHIM
Comando Concreto para Consultar Detalle Proyecto	i. BuscarDescripcionProyecto
Base de Conocimiento	i. <<Facade>> BaseConocimiento
Bitácora	**i. DAORegistroBitacora**
RDF Pie	i. DAORDFPIE ii. <<abstract>> VOProducto iii. VOProductoPIE

Para concluir se hace notar que dentro de cada diagrama de componentes se encuentran resaltados aquellos componentes que tienen como finalidad cubrir aquellos aspectos de seguridad que hemos venido comentado a lo largo de los capítulos anteriores.
Dentro de los cuales se encuentran los componentes de:

- *Administrador de Filtros, Cadenas de Filtros, Filtro de Login y Filtro de Autenticación.* Esta lista de componentes agrupan la funcionalidad de realizar la verificación y validación de la existencia del usuario antes de poder realizar la autenticación del usuario que intenta ingresar.

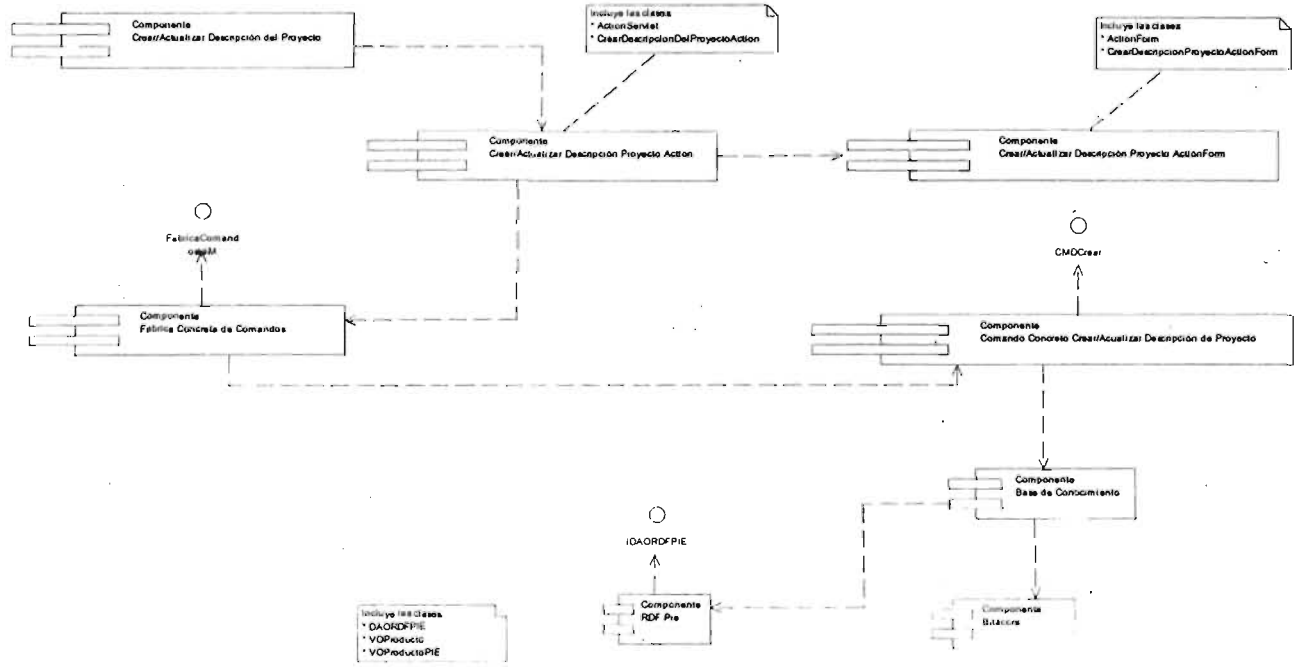


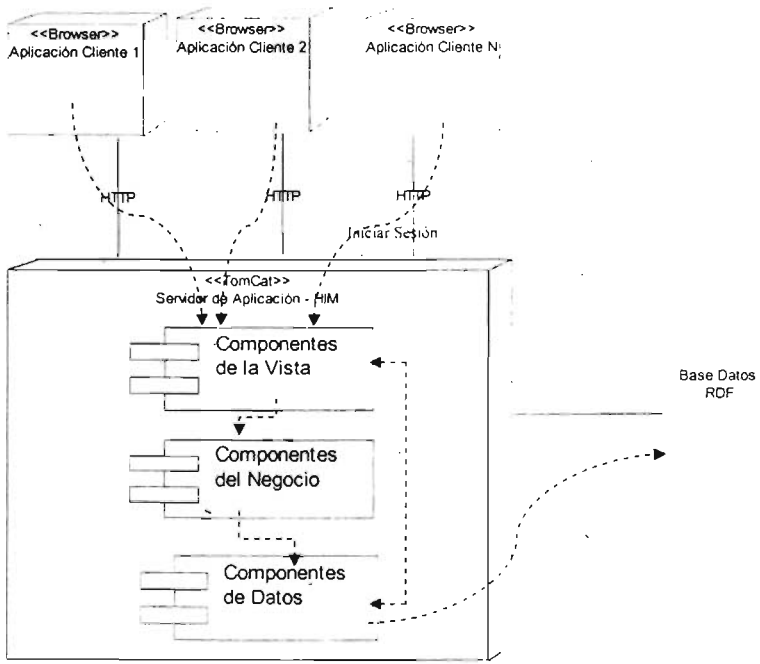
Figura 4. Diagrama de componentes para realizar la creación y/o actualización.

- *Control de Permisos, Marco General de Trabajo, RDF Proceso y RDF Proyecto.* Estos componentes constituyen el control de las actividades que podrá realizar el rol con que ingresa el usuario del sistema, obteniendo de las sentencias RDF del proyecto y de los procesos que conformaran el Marco de Trabajo del usuario.

- *Bitácora.* Este componente tiene como finalidad llevar el registro en archivo XML todas las actividades que se realicen en el sistema.

5.6 Diagrama de distribución

El diagrama de distribución se muestran los elementos físicos por medio de nodos que se encuentran interconectados, es decir, son elementos hardware sobre los cuales pueden ejecutarse los elementos software.



Dentro del diagrama de distribución se encuentra la participación de los siguientes nodos que tienen como función:

- Nodo <Browser>.- Estos nodos representan el medio por el cual el cliente podrá interactuar con el sistema de software, en su caso será vía el navegador donde se desplegará la vista de la aplicación.

- Nodo <Base de Datos RDF>.- Principalmente representa el modelo de datos donde se alojará toda la persistencia de la información del negocio.

- Nodo Servidor de Aplicación.- Pone en servicio la funcionalidad completa del sistema HIM, atendiendo las peticiones realizadas por la interacción con el sistema que realiza el cliente.

Conclusiones

Durante el transcurso de la tesis se tocaron temas importantes acerca de la inseguridad que puede presentar cualquier sistema de software y principalmente en la construcción de uno si no son considerados los mecanismos de seguridad necesarios para mitigar vulnerabilidades que puedan darse. Por tal motivo se hace hincapié en la importancia de aplicar de manera adecuada técnicas criptográficas, y el modelado de componentes para nuestro *framework* de arquitectura.

Haciendo referencia a los patrones utilizados se concluyó que los patrones de *Intercepting Filter*, *Singleton* y *Data Access Object* fueron adecuados para realizar los componentes de seguridad.

En su caso, el patrón *Intercepting Filter* nos permitió realizar la autenticación del usuario, controlando desde este componente el acceso al sistema. Los otros dos patrones *Data Access Object* y *Singleton* se utilizaron en conjunto para definir el registro de actividades en la Bitácora XML, centralizando en único objeto la responsabilidad de escribir en el archivo de bitácora.

El uso de la nueva tecnología conocida como RDF (*Resource Description Framework*), acoplada a las necesidades que se presentaron, dejó en claro el valor que puede tener la explotación de la información a través de sentencias lógicas que se declaran en RDF, teniendo así un mejor entendimiento del modelo de dominio de negocio. Se pudo definir un esquema de control de permisos adecuados a través de las sentencias lógicas, con el cual se pudo realizar una manipulación y construcción del marco de trabajo de la herramienta, que estuviera conformado por actividades que el rol tuviera permiso a realizar.

Con base en los requerimientos que se establecieron en el alcance inicial para la construcción de la herramienta HIM y objetivos que se pretendían cubrir con la presente tesis, se concluye que fueron cubiertos totalmente los requerimientos.

Quizás la visión y el alcance fueron muy acotados para esta primer versión el diseño del *framework* HIM, fue pensado con el fin de que pudiera ser extendido con nuevas funcionalidades, realizando un desacoplamiento adecuado entre capas y componentes que permitiera su futura evolución, y que permitirá agregar de forma adecuada aquellos requerimientos que quedaron fuera del alcance.

Dentro de los posibles requerimientos que pueden ser retomados para mejorar los aspectos de seguridad para la herramienta, se proponen los siguientes:

- Usar técnicas criptográficas con el fin de cifrar la información a través del protocolo http, evitando así que la información viaje en claro. Actualmente muchos servidores Web permiten el manejo del protocolo https donde se incorpora una nueva capa de seguridad.

- Hacer uso de firmas digitales para realizar la actividad dentro del modelo de procesos de validar y verificar productos generados dentro del proceso.

- Construir un servidor de firmas digitales, los servidores que existen son de dominio comercial.

- Realizar el control de integridad de los archivos RDF que funge como base de datos.

Bibliografía

1. [FW00] Fisch Eric A., White Gregory B. "Secure Computers and Networks. Analysis, Design and Implementation".
2. [IBM] Hagemo Lloyd & Kalidindi Ravi. "Creating a Framework - J2EE pattern frameworks provide template for flexible and modular architecture"
3. [Hernández] Hernández, U.E. "Uo de la tecnología RDF para representar y manejar los procesos MoProSoft y su aplicación en HIM." Tesis de maestría en ingeniería del posgrado de la UNAM 2005.
4. [MED98] Mediavilla Matriz Manuel. "Seguridad en Unix. Ra-Ha"
5. [Menezes] Menezes Alfred J., Van Oorschot Paul C., Vanstone Scott A. "Handbook of Applied Cryptography".
6. [Mercado] Mercado, F.A.E. "La Sincronización de los elementos de una Base de Conocimiento para MoProSoft y su Aplicación en una Herramienta Integral" Tesis de maestría en ingeniería del posgrado de la UNAM 2005.
7. [MoProSoft] Oktaba Hanna, Claudia Alquicira Esquivel, Angélica Su Ramos, Alfonso Martínez Martínez, Gloria Quintanilla Osorio, Mara Ruvalcaba López, Francisco López Liraa Hinojo, María Elena Rivera López, María Julia Orozco Mendoza, Yolanda Fernández Ordónez, Miguel Ángel Flores Lemus. "Modelo de Procesos para la Industria de Software. Versión 1.1 Mayo 2003".
8. [Valdivieso] Valdivieso, C.K. "Utilización de patrones y la arquitectura J2EE para el diseño de la interfaz de usuario para la herramienta Integral MoProSoft (HIM)" Tesis de maestría en ingeniería del posgrado de la UNAM 2005.
9. [Vázquez] Vázquez, M.M.O. "Aplicación de patrones basados en J2EE para el diseño e implementación de la capa de control de la herramienta integral para MoProSoft (HIM)" Tesis de maestría en ingeniería del posgrado de la UNAM 2005.
10. [Zurita] Zurita, R.H. "Arquitectua de la Herramienta Integral para MoProSoft" Tesis de maestría en ingeniería del posgrado de la UNAM 2005.
11. [DE2] http://struts.apache.org
12. [RDF1] http://www.w3.org/rdf/
13. [CD] Documentación digital HIM