MARCO DE TRABAJO PARA LA DEFINICIÓN DE PROCESOS DE DESARROLLO DE SOFTWARE
FRAMEWORK PDS

FABIAN RODRIGO GUERRERO GUZMAN
JOHN FREDY MARTINEZ GOMEZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
POPAYÁN
2006

MARCO DE TRABAJO PARA LA DEFINICIÓN DE PROCESOS DE DESARROLLO DE SOFTWARE
FRAMEWORK PDS

FABIAN RODRIGO GUERRERO GUZMAN
JOHN FREDY MARTINEZ GOMEZ

Director
PhD. JUAN CARLOS VIDAL ROJAS
Ingeniero en Electrónica y Telecomunicaciones

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
POPAYÁN
2006

CONTENIDO

INDICE DE FIGURAS

1 INTRODUCCIÓN

La ingeniería del software surge a mediados del siglo XX para permitir un enfoque multidisciplinar en el desarrollo de productos software de alta complejidad. Aunque su aplicación en los comienzos fue en grandes proyectos del sector defensa y aeroespacial, actualmente la diversidad de productos del mercado, ha supuesto un mayor campo de aplicación.

En sus inicios, la ingeniería del software estaba orientada a la documentación utilizando descripciones textuales en un conjunto de documentos de especificaciones y diseño, con planos que describían la arquitectura física del producto, con la dificultad de mantener la documentación al día, la ambigüedad del lenguaje natural para describir los problemas de ingeniería y la dificultad de la evaluación rigorosa de las soluciones escogidas.

Actualmente, la ingeniería del software esta cambiando su orientación hacia los procesos, colocando más atención a la forma de realizar los productos, sin dejar a un lado su documentación, ya que se afirma que la calidad del producto, depende de la calidad del proceso que se sigue para obtenerlo. Se habla entonces de la "Tecnología del Proceso Software" (TPS), un campo dentro de la ingeniería del software que como disciplina autónoma, inició en los años 80's a través de una serie de eventos (*International Software Process Workshop*, *European Workshop on Software Process Technology*...) y publicaciones (*Journal of Software Process: Improvement and Practice*...), y la primera contribución importante de la TPS fue la confirmación de que el desarrollo y mantenimiento del software son procesos complejos, que requieren un esfuerzo colectivo y creativo. Por tanto, la calidad de un producto software depende fuertemente de las personas, la organización y los procedimientos utilizados para crearlo, entregarlo y mantenerlo [30]. En la actualidad, los investigadores alrededor del mundo han enfocado sus esfuerzos a la optimización de tecnologías que buscan mejorar los procesos internos de una organización, creando las herramientas necesarias para que la industria cuente con un nuevo objeto que facilite su desarrollo y crecimiento, entre las cuales se encuentran estándares y guías, modelos de mejoramiento de procesos y métodos internos de evaluación, modelos del ciclo de vida del software, modelos de ingeniería de sistemas, entre otros.

Pero en Colombia al igual que el resto de países latinoamericanos la industria de software es incipiente. Muchas de las pequeñas y medianas empresas del sector se enfrentan a una indudable desventaja dada por la falta de competitividad que dificulta su crecimiento y aumenta la dependencia existente con tecnologías de países desarrollados. Uno de los factores que contribuye a esta problemática es que en muchas empresas, no se tienen en cuenta los conceptos que encierra la definición de Proceso Software[1], causando la falta de preparación para ser competitiva internacionalmente, antes debe priorizar los problemas a los que se enfrenta, tales como: el desconocimiento de la importancia que tiene el proceso de desarrollo sobre la calidad producto, la dependencia tecnológica del país y la construcción de software de forma artesanal, empírica y caótica, debido a errores de desarrollo comunes como la inexistencia de métricas de calidad, la limitada posibilidad de valorar el tiempo necesario para el desarrollo de proyectos y la no cuantificación del impacto de los proyectos cuando éstos involucran varias áreas de la empresa [1]. Todos estos problemas hacen que la calidad del software que se desarrolla sea baja, el tiempo de desarrollo inapropiado, los costos no sean competitivos, las actividades de operación y mantenimiento del software difíciles y desde luego, el incremento de la insatisfacción de los clientes y usuarios finales.

A pesar de las desventajas competitivas que tiene el país, el crecimiento de la industria del software aumenta progresivamente y es imprescindible preguntarse: ¿Cómo motivar a las empresas de desarrollo de software Colombianas para que mejoren sus procesos de desarrollo, de tal manera que les permita incrementar su competitividad nacional y mundial? [2]. La respuesta a esta pregunta no es fácil, ni tampoco se encuentra en una sola actividad a seguir o en un solo ente de la sociedad; es necesario un compromiso real e impostergable por parte del Estado, la Academia y el sector empresarial. Concientes de esta problemática y del papel protagónico que debe presentar la Universidad del Cauca, el Grupo de Investigación y Desarrollo en Ingeniería de Software – IDIS, propone el Sistema Integral para el Mejoramiento de los Procesos de Desarrollo de Software en Colombia (SIMEP-SW) que es una solución donde el aseguramiento de la calidad es el área de trabajo que requiere de un esfuerzo crítico. Para consolidar una industria de tipo nacional e internacional, se necesita de un proceso maduro que brinde muchos

[1] El proceso software se define como un conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software [4].

beneficios, entre ellos: que la gente desarrolle su potencial de forma más efectiva, permita a las empresas enfocarse en el sistema y que puedan definir, gestionar, evaluar y mejorar sus propios procesos [3]. El proyecto SIMEP-SW, busca definir un modelo de referencia y un modelo de mejoramiento con las especificaciones que debe cumplir una empresa colombiana, así como una base de conocimientos basada en la experiencia que permita definir las recomendaciones para implementar dichas especificaciones donde se presentan las actividades que una determinada empresa puede realizar con sus procesos. Para ello, es necesario y primordial definir, visualizar, instanciar, valorar y mejorar sus procesos de desarrollo de software.

Sin lugar a duda la mayor dificultad se presenta en el momento en que la empresa decide seguir un proceso, adelantar planes de mejoramiento y adquirir una cultura hacia el mejoramiento continuo, con repercusión real en la organización, pues el costo y esfuerzo de integrar áreas en pro de los objetivos del negocio puede ser mucho mayor al esfuerzo realizado para la implantación y mejora de aplicaciones. Nos encontramos entonces con una industria de software inmadura debido a muchos factores entre los que encontramos: procesos improvisados, procedimientos importantes que no se realizan, la responsabilidad de los proyectos recae sobre unos pocos individuos, se sacrifica funcionalidad y calidad de los productos con en fin de cumplir con fechas límite, empresas no preparadas para cambios constantes de tecnología y la inexistencia de parámetros que puedan predecir la calidad de los productos [1]. En este contexto, es pertinente preguntarse: ¿cómo facilitar a las empresas de desarrollo de software definir y visualizar sus procesos? La respuesta a esta inquietud, no es puntual, porque trae inmersa una serie de actividades que no solo dependen de las "tecnologías de procesos", sino que también depende del compromiso y convicción de las empresas desarrolladoras, una vez dado este primer paso se inicia la implantación del proceso y su maduración dentro de su organización.

Un factor clave a tener en cuenta como parte del diagnóstico de la industria del software es considerar las herramientas de "tecnologías de procesos" que utilizan las empresas para definir y madurar sus procesos, lo cual permite concluir que no existe un marco de trabajo adaptado a las necesidades de la industria de software colombiana, debido a que generalmente están sobredimensionadas en su composición, dificultando su implantación. Esto trae consecuencias negativas, porque las empresas que proyectan su utilización, se encuentran con problemas como la resistencia al cambio por parte de sus empleados, la

considerable inversión económica para lograr la transición a los procesos controlados y el inconveniente que representa la barrera del idioma para la interpretación de dichas herramientas. Además, en muchas ocasiones, son poco flexibles y no proporcionan a las empresas la libertad de construir su propio proceso de desarrollo de software.

Por todo lo anterior, en el presente trabajo se propone un Marco de Trabajo para la definición de Procesos de Desarrollo de Software denominado Framework PDS, que es una plataforma libre de licenciamiento, flexible y de código abierto, basada en el modelo de referencia SPEM (Software Process Engineering Metamodel) [27] que facilita a las empresas de software la definición, visualización y mejoramiento de sus procesos de desarrollo, suministrando ejemplos de modelos de procesos y su explicación para que una organización en un determinado momento, los tome como ejemplos para el entrenamiento del personal en cuanto al modelado de procesos se refiere. De esta manera, el seguimiento y evaluación de proyectos en estas empresas contará con un instrumento más para el control de sus procesos y como resultado una incidencia directa en la calidad de los mismos.

La definición y visualización de los procesos de una compañía, trae consigo una serie de beneficios que retribuyen la inversión de los recursos destinados para ello. Su importancia radica en la evolución de la empresa al ritmo de la industria, manteniendo sus procesos actualizados, organizados y controlados para facilitar la mejora de los mismos. Las ventajas asociadas a esta actividad están enmarcadas en el seguimiento que se puede realizar a los procesos, la definición y asignación de las tareas específicas a los participantes y la facilidad que éstos tienen al momento de requerir documentación relacionada al proceso. Todas estas características convergen en el aumento de la productividad de los procesos y de su personal encargado, un mejor ambiente laboral, el incremento de la calidad de sus productos y por consiguiente un mejor posicionamiento de la organización en el mercado.

La construcción del Marco de Trabajo para Procesos de Desarrollo de Software en Colombia, propone además un aporte académico tanto para docentes como estudiantes, dado que en las instituciones de educación superior como la Universidad del Cauca, se cultiva el saber de la ingeniería del software en la que se adaptan y aplican métodos, técnicas, prácticas y demás componentes asociados al proceso. Toda esta exploración va acompañada de la continua motivación de mantener a la Universidad del Cauca a la

vanguardia en la experimentación de nuevas tecnologías, las cuales podrían en un futuro comenzar a dominar el mercado del desarrollo de software en el contexto mundial.

Se adoptaron diferentes estrategias de divulgación para el proyecto, que consistieron en la presentación de una ponencia en el Seminario de Calidad de Software, realizado en junio de 2005 por el Departamento de Sistemas de la Universidad del Cauca, un trabajo de campo con la empresa SIDEM LTDA que consistió básicamente en la identificación, estructuración y modelado del proceso de desarrollo de software, llevándolo a un estado de definición y visualización. Igualmente, se presentó a consideración del Comité Académico del "IV Simposio Internacional de Sistemas de Información e Ingeniería de Software en la Sociedad Del Conocimiento"[2] el cual fue aceptado para ser presentado durante los días de la realización del evento en la ciudad de Cartagena de Indias – Colombia.

[2] http://www.sisoftw.com

2 OBJETIVOS

2.1 OBJETIVO GENERAL

Construir un Marco de Trabajo que soporte la definición, visualización y aplicación de procesos de desarrollo de software, estructurado bajo la especificación SPEM garantizando su portabilidad y adaptabilidad a estándares de WfmC.

2.2 OBJETIVOS ESPECIFICOS

- Crear una base de conocimiento centrada en las tecnologías de procesos de software que pueda ser aprovechada por la industria del software colombiana en sus programas SPI (Software Process Improvement)

- Desarrollar una herramienta software, de libre licenciamiento, flexible y de código abierto, basada en la especificación SPEM y en cumplimiento con la *interfaz 1* (*Workflow Definition Interchange)* del modelo de referencia Workflow Management Coalition

- Entregar a la comunidad empresarial y académica un conjunto de herramientas, plantillas, informes técnicos y demás documentos con el fin de aportar a la mejora de la calidad y la productividad de la ingeniería del Software en el país.

3 MARCO TEÓRICO

3.1 DEL PROCESO AL PROCESO DE DESARROLLO DE SOFTWARE

Los procesos de software, involucran un sin número de conceptos que llevan a obtener diferentes definiciones dependiendo de la situación y puntos de vista en los cuales los investigadores y desarrolladores se desenvuelven. El contexto base de SIMEP-SW y particularmente de Framework-PDS esta basado en el trabajo de Alfonso Fuggetta [4], donde se afirma que "los procesos de software también son procesos". Igualmente consideramos el proceso de software una forma "especial" y "única" de proceso debido a la constante reutilización de metodologías y prácticas de otras comunidades que hacen de esto una sucesión de experiencias enfocadas a satisfacer una serie de objetivos. La definición más básica de proceso nos lleva a tomar este concepto como un conjunto de fases sucesivas de un fenómeno natural o de una operación artificial [5]. En informática se manejan varias definiciones que aluden a diversos elementos: proceso puede ser simplemente una operación o conjunto combinado de operaciones con datos, o bien una secuencia de acontecimientos definida única y delimitada, que obedece a una intención operacional en condiciones predeterminadas. Así mismo se señala proceso a una función que se está ejecutando [6].

3.2 EL PROCESO SOFTWARE

Un proceso define *quién* está haciendo *qué, cuándo* y *cómo* alcanzar un determinado objetivo. En la ingeniería del software el objetivo es construir un producto software o mejorar uno existente.

Un proceso efectivo proporciona lineamientos para el desarrollo eficiente de software de calidad; captura y presenta las mejores prácticas que el estado actual de la tecnología lo permite [7]. De esta forma, un proceso de desarrollo de software debería ser capaz de evolucionar con los años. Durante esta evolución y en un momento de tiempo dado,

debería limitar su alcance a las realidades que permitan las tecnologías, herramientas y stakeholders[3] de una organización. Así, el proceso debe construirse sobre las tecnologías; los procesos y las herramientas deben desarrollarse en paralelo. Un creador del proceso debe determinar el conjunto de habilidades necesarias para desenvolverse en él y finalmente, el creador del proceso debe adaptar el proceso a las realidades del momento. Cada una de estas cuatro circunstancias *(Tecnologías, Herramientas, Personas y Patrones de organización)* debe mantener el equilibrio del proceso de desarrollo de software, de modo que el proceso pueda evolucionar. La Figura 1 representa el concepto de proceso de software.

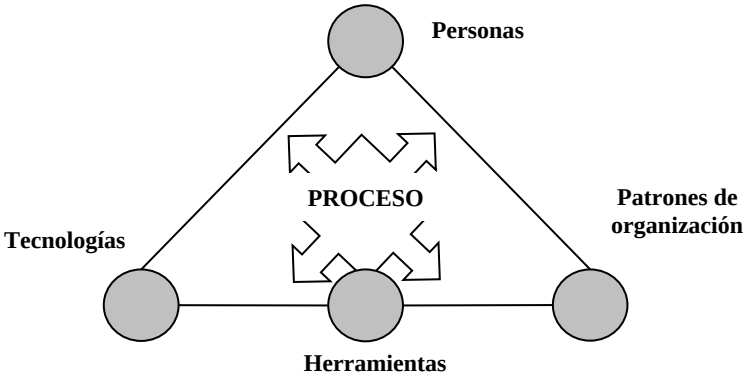


Figura 1 - El Proceso de Software

Esta evolución debe ir encaminada a la consecución de los requisitos de calidad por parte de la empresa. A la hora de satisfacer los requisitos de calidad de los procesos software es necesario que produzcan los resultados esperados, que estén correctamente definidos y que sean mejorados en función de los objetivos de negocio, muy cambiantes ante la gran competitividad de las empresas hoy en día. Estos son los objetivos de la Gestión del Proceso Software que, para ser aplicada de forma efectiva, supone asumir cuatro responsabilidades claves: Definir, Medir, Controlar y Mejorar el Proceso. Estas responsabilidades y sus relaciones se representan en la Figura 2.

Para el Framework PDS la evolución del proceso considera cuatro etapas que inician con la definición, el modelado, la transformación y la visualización.

[3] *stakeholder* es cualquier actor (persona, grupo, entidad) que tenga una relación o intereses (directos o indirectos) con o sobre la organización (Thompson et al., 1991; Donaldson & Preston, 1995).

La definición del proceso está determinada por una serie de pasos que es necesario abordar para cumplir con tal objetivo. Estos pasos comprenden:

- Establecimiento de los objetivos del proceso
- Límites de inicio y fin del proceso
- El flujo del proceso, es decir la secuencia de actividades que se llevan acabo durante su ejecución.
- Asignación de recursos como personal, tiempo, equipo y demás materiales.
- Los clientes y los proveedores del proceso (internos y externos).

El modelado, corresponde a la representación del proceso mediante un lenguaje de modelado de procesos (LMP), garantizando el cumplimiento de todas las reglas semánticas y sintácticas que el LMP tenga establecidas.

La transformación del proceso es un caso alterno de la definición del proceso, pero se expone aquí por requerimiento del Framework PDS. Esta transformación se refiere a la adaptación de un modelo de procesos a otro lenguaje de definición de procesos, mediante unas reglas gramaticales claramente definidas.

Finalmente la visualización del proceso hace referencia a la exposición del mismo al stakeholder, para efectos de seguimiento, el cual puede hacerse de forma manual o mediante un motor de automatización del proceso.

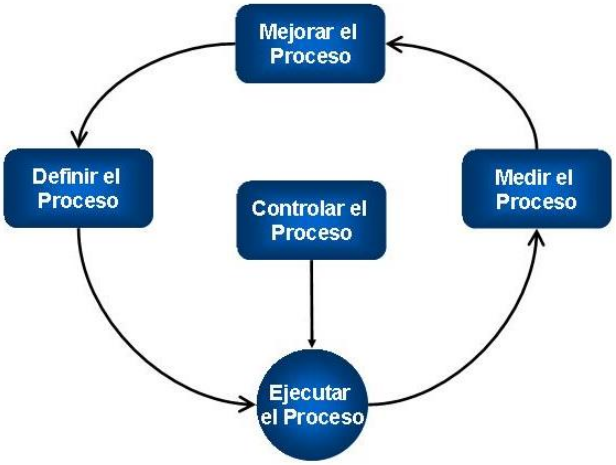


Figura 2 - Responsabilidades de la Gestión de Procesos

3.2.1 Modelos de Procesos de Desarrollo de Software

Existen numerosos modelos de proceso de desarrollo de software creados a través de la historia, que buscan de una u otra manera, ser más eficientes a la hora de conseguir los objetivos propuestos por las empresas, logrando que la representación de procesos pueda ser usada para conducir los actuales procesos de desarrollo y mantenimiento de software [8]. El concepto de representación del proceso ha evolucionado al punto de involucrar la noción de Modelo del Proceso (MP) que obedece a una representación abstracta de un conjunto de procesos que se especifican formalmente a partir del modelado de procesos. La existencia de los modelos de proceso permite en gran medida aprovechar capacidades adicionales para completar procesos, automatizarlos, dirigirlos y aumentar su eficiencia, gracias a que sirve de base para lograr una comunicación fluida entre los procesos. De una manera no exhaustiva, los modelos de procesos más conocidos y utilizados en la industria del software son: Cascada, Construcción de Prototipos, Espiral, Desarrollo Rápido de Aplicaciones – DRA, Ensamblaje de Componentes [9] y Modelo en V [11].

3.2.2 Metodologías de Desarrollo de Software

Las metodologías de desarrollo de software están enmarcadas en dos grandes grupos, por un lado tenemos aquellas propuestas más tradicionales (metodologías pesadas) que se centran especialmente en el control del proceso, estableciendo rigurosamente roles, las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán, incluyendo un modelado y documentación detallada. Este esquema ha demostrado ser efectivo en proyectos de gran tamaño (Tiempo y recursos), algunas de estas metodologías son el RUP [14], Catálisis [15], Métrica [17], Kobra [18], EUP [20], MSF [13], entre otras. Pero este enfoque tradicional no es el más apropiado para muchos de los proyectos actuales, donde el entorno del sistema es muy cambiante, los tiempos de desarrollo bastante cortos pero sin dejar a un lado la calidad en los productos [12]. Por esta razón, muchos equipos de desarrollo buscan otra aproximación, centrándose en otras dimensiones, como por ejemplo, el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles (livianas), apoyadas en el

manifiesto ágil[4], las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Algunas de las metodologías más sobresalientes de este tipo son: eXtreme Programming (XP) [22], Scrum [23], Evolutionary Project Management (Evo) [23], Cristal Methods [25], Agile Modeling [26], entre otras.

Hasta el momento, nos hemos referido a los modelos y las metodologías existentes para el seguimiento y control del proceso de desarrollo de software y aunque éstos modelos y metodologías podrían ser adaptados a cualquier proceso dentro de una organización, no es lo mas optimo, dada la naturaleza por la cual han sido creados (el proceso de desarrollo de software). Entonces, surge una inquietud: ¿Cómo una empresa puede documentar y en algún momento mejorar sus procesos internos y externos?

Una empresa de desarrollo de software no solo la constituye el proceso de desarrollo de software, sino que necesariamente tendrá procesos complementarios que hacen que funcione como empresa y que cobran mucha importancia al momento de realizar un proceso de mejora en ella. Particularmente, para ellos, no hay modelos o patrones que guíen su comportamiento y se hace necesario definirlos, modelarlos y documentarlos al igual que el proceso de desarrollo de software.

3.3 MODELADO DE PROCESOS

El Modelado de Procesos es un paso fundamental para la comprensión y mejora continua de los procesos de una organización. El modelado del proceso software trata de capturar las características principales del proceso que se lleva a cabo, identificando las actividades necesarias, los responsables de llevarlas a cabo, los productos obtenidos, etc. [8].

Las metodologías de modelado de procesos principalmente se dividen en tres categorías: basadas en comunicaciones, basadas en artefactos y basadas en actividades [28].

[4] http://www.agilemanifesto.org

3.3.1 Modelado basado en comunicaciones

Este tipo de metodologías representa una acción en una comunicación que se ha establecido entre un cliente y un proveedor. Esta comunicación consiste en 4 fases:

- **Petición**, cuando un cliente requiere que se lleve a cabo una acción
- **Negociación**, el cliente y proveedor se ponen de acuerdo en la acción que ha de ser ejecutada
- **Realización**, Fase en la que la acción a llevar a cabo es ejecutada
- **Aceptación**, donde el cliente informa de la conformidad con la acción realizada por el proveedor [28].

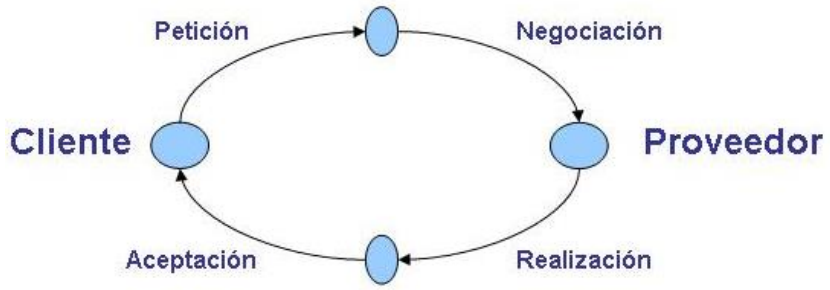


Figura 3 - Modelo basado en comunicación

3.3.2 Modelado basado en Actividades

Las metodologías basadas en las actividades se centran en el modelo de trabajo, en contraste con los casos previos donde los acuerdos establecidos entre los participantes son las bases del modelado de procesos. Esta aproximación está basada en la descomposición de procesos en tareas que son ordenadas de acuerdo con las dependencias existentes entre ellas. La principal ventaja reside en que este modelo puede ser fácilmente convertido en la especificación de un workflow (flujo de trabajo) del cual se dará una breve explicación más adelante. Además, este tipo de modelado es susceptible de ser representado mediante un lenguaje orientado a objetos [28].

Figura 4 - Modelo basado en actividades

3.3.3 Modelado Basado en Artefactos

La aproximación basada en artefactos se centra en los objetos (artefactos) que son creados, modificados y utilizados en un proceso (Figura 5). Esto supone que el modelado de un proceso está basado en la creación de los documentos generados durante él mismo y su camino a través de una serie de actividades del flujo de trabajo [28].

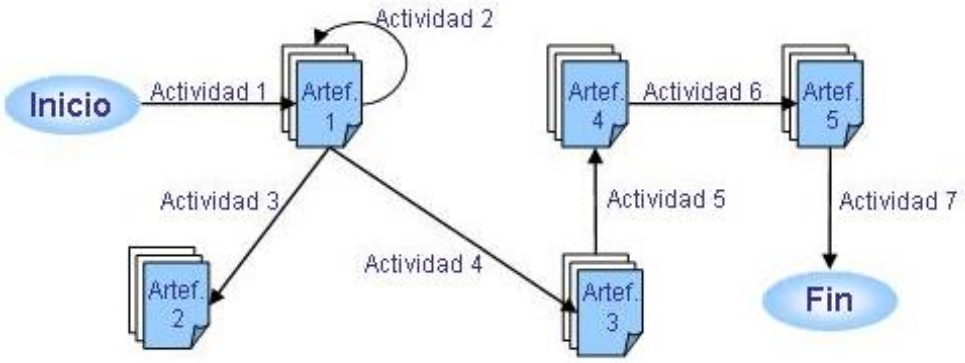


Figura 5 - Modelo basado en artefactos

3.3.4 Modelado de WorkFlows

El modelado de workflows es una metodología que deriva del modelado de actividades. Un flujo de trabajo (workflow) viene definido por la representación gráfica de un proceso de trabajo o de negocio. La Figura 6 muestra un ejemplo de modelado de workflows.

Workflow se relaciona con la automatización de los procedimientos donde los documentos, la información o tareas son pasadas entre los participantes del sistema de acuerdo a un conjunto de reglas previamente establecidas. El fin de lo anterior es llegar a culminar una meta común impuesta por la empresa [29].

Figura 6 - Modelo basado en Workflows

3.3.5 Lenguajes de Modelado de Procesos – LMPs[5]

Los procesos de software al igual que cualquier entidad abstracta requieren de un medio o plataforma para poderlos expresar –El Modelo de Procesos (MP)-, estos a su vez, para su entendimiento y facilidad de interpretación requieren de la utilización de una serie de reglas sintácticas y semánticas para describir sus elementos, como por ejemplo, roles, actividades, recursos, etc. Un LMP permite hacer la transición de un proceso del mundo real a modelos de proceso. Según [28], los elementos primarios de un proceso que debe ser capaz de modelar un LMP son: Actividades, productos, roles, personas, herramientas y soporte para la evolución. Un LMP debe ofrecer soporte para la evolución del modelo de procesos tanto a nivel técnico como a nivel conceptual, este último mediante un modelo asociado. Además de los elementos primarios los LMPs incluyen la posibilidad de representar elementos propios a nivel de meta proceso, entre los cuales podemos mencionar: Proyectos/organizaciones, contextos de trabajo, vistas de usuario, etc [30]. Los lenguajes de modelado de procesos se pueden dividir en tres categorías:

- **Lenguajes Formales** que tienen sintaxis y semántica formales y, por tanto, proveen soporte para verificación y análisis formales, simulación y ejecución. Este tipo de lenguajes utiliza diferentes tipos de connotaciones lingüísticas y

[5] En algunos textos este término se encuentra como PML (Process Modeling Language) Por sus siglas en Inglés

paradigmas entre los cuales encontramos: lenguajes *basados en reglas*, *basados en estados, funcionales, procedurales, orientados a objetos, entre otros.* Algunos ejemplos de LMPs formales son: basados en reglas: Marvel, Oz, Atlantis; Orientados a Objetos: E3, EPOS/Spell, Lenguajes de Programación: APPL/A, JIL, Grafos y gramáticas: Hakoniwa, Redes de Petri: SLANG/SPADE [31].

- **Lenguajes Semiformales** están provistos de una notación formal (normalmente gráfica) pero no tienen semántica formal. Esto último hace que no sean completamente ejecutables.
- **Lenguajes Informales** no tienen sintaxis ni semánticas formales (por ejemplo, el lenguaje natural).

3.3.6 Evaluación y Mejora de Procesos

La evaluación de los procesos software tiene como objetivo detectar los aspectos de un proceso software que se pueden mejorar. Para ello es necesario proporcionar un marco efectivo para la medición de los procesos y productos software en una organización.

La integración de esta área con el modelado de procesos es un factor fundamental para que una organización alcance un alto grado de madurez en sus procesos tal como identifican diversos estándares entre los que destacan especialmente: CMM, ISO y sobre todo CMMI. Por ello, es imprescindible comprender bien los procesos, para lo cual es necesaria una definición de los mismos [8].

Para que una organización pueda realizar una gestión integrada de sus procesos software es muy importante que establezca una base rigurosa para:

- La definición de sus modelos de procesos con una terminología única y con una semántica precisa y bien definida [8].
- La gestión integrada de los procesos en la organización, tomando como base un marco de trabajo que facilite la definición, medición, evaluación y ejecución de los procesos.

El modelado de procesos cobra mucha importancia al momento de llevar a cabo un proceso de mejora dentro de una organización, teniendo en cuenta que las actividades internas que se llevan a cabo, son de gran valor y aporte tanto económico y logístico a

todos los empleados de la misma. Los ya mencionados enfoques metodológicos, brindan el camino a seguir para conseguir el objetivo de plasmar o modelar el proceso, volverlo visible y útil a todos sus participantes. Los lenguajes de modelados de procesos permiten su modelado formal, permitiendo adoptar estándares que faciliten su mejor entendimiento y hacen la transición del proceso del mundo real, al proceso modelado. La evaluación y mejora de procesos, proporcionan a la compañía, una "actitud" proactiva con relación a la adaptación que la misma industria lo exige.

3.4 EL CONCEPTO DE MARCOS DE TRABAJO[6] (FRAMEWORKS)

Un framework es una Infraestructura software que crea un entorno común para integrar aplicaciones e información compartida dentro de un dominio dado [41], además según [42] los frameworks corresponden a estructuras escritas de una idea y/o conjunto de metas para facilitar a una organización la aplicación de las mismas. Existen muchas más definiciones encaminadas a diferentes objetivos siendo los más comunes, los frameworks de desarrollo de componentes software de los cuales se puede encontrar mucha bibliografía.

Mediante los frameworks se permite que todo el personal de una organización se dirija en la misma dirección. En la Ingeniería del Software se pueden catalogar los frameworks por el propósito que cumplen (Figura 7), sin embargo un framework puede estar compuesto por varias categorías. Así, Framework PDS se puede estructurar como un framework que utiliza Estándares, modelos de mejoramiento y modelos del ciclo de vida de los procesos de una organización.

[6] Algunos autores traducen workflow como Marco de Trabajo, en adelante estos términos se utilizarán indistintamente.

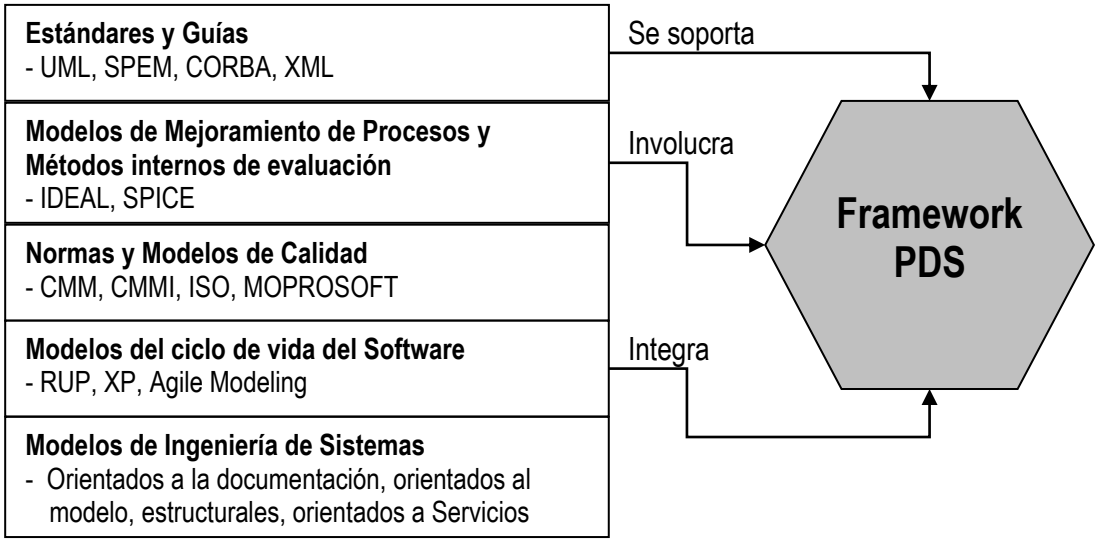


Figura 7 - Las categorías de frameworks y el Framework PDS

La finalidad de un framework es la de mejorar los procesos de software, brindar pautas para efectuar evaluaciones de la unidad informática, determinar la potencialidad y el desempeño de sus procesos y la madurez de la organización [42].

Actualmente se encuentran en el mercado gran diversidad de Marcos de trabajo, que generalmente se enfocan en el ciclo de vida del software, sin estar concebidos específicamente para dar soporte al proceso software, el cual como ya se ha tratado anteriormente, contiene una serie de características particulares que no son tomadas en su totalidad. Por ésta razón, se hace necesario realizar un framework de soporte al proceso en una organización, que facilite la documentación y visualización de los diferentes componentes y además, permita el entrenamiento de todos sus integrantes.

Sería muy difícil y complejo tratar todos los framework existentes, sin embargo, listamos algunos ejemplos que facilitan la comprensión del concepto.

3.4.1 El Framework de Procesos RUP

El RUP es un proceso de ingeniería de software que proporciona una aproximación disciplinada a la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad, resolviendo

las necesidades de los usuarios finales, dentro de un tiempo y presupuesto razonable [14]. RUP define tres elementos esenciales:

1. Un conjunto de filosofías y prácticas para un desarrollo de software exitoso: Estas filosofías, *Prácticas principales y Elementos esenciales,* son el fundamento sobre el cual, RUP ha sido desarrollado. Las Prácticas principales consisten en mostrar y aplicar varias de las mejores prácticas de ingeniería de software como lo son Desarrollo iterativo, Gestión de requerimientos, Uso de una arquitectura de componentes, etc. Los elementos esenciales, hacen parte del proceso de desarrollo de software y consiste en conseguir un balance entre un software de calidad y una entrega rápida, ajustando el proceso de la mejor manera con las necesidades del proyecto. Algunos de los elementos esenciales del RUP son la visión, el plan de actividades, los riesgos, entre otros.
2. Un modelo de proceso y una librería asociada: define la base del proceso de ingeniería de software del RUP mediante el cual se pueden crear su propia configuración del proceso.
3. El lenguaje de definición del proceso: Contiene un metamodelo de proceso, este modelo provee un lenguaje de elementos de definición de procesos para describir un proceso de ingeniería de software. Este lenguaje esta basado en la extensión SPEM del UML para ingeniería de procesos de software y la metodología del Proceso Unificado (UP) [14].

3.4.2 Eclipse Process Framework (EPF)

Eclipse Process Framework es una iniciativa de IBM para crear un framework de procesos de código abierto que ayude a las compañías a establecer consistencia en la planeación y ejecución de proyectos de desarrollo de software.

EPF define tres objetivos: el primero es desarrollar estándares de desarrollo de herramientas y lenguajes de procesos que permitan a los usuarios finales, organizaciones, clientes y comunidades de procesos colaborar entre si con el fin de intercambiar sus mejores prácticas para de esta manera acelerar la innovación, generando prácticas de software más efectivas para ser desarrolladas. El Segundo objetivo es proveer a la comunidad de desarrollo de software procesos de código abierto que promuevan las prácticas de desarrollos ágiles e iterativos como integración continúa

y pruebas a través del ciclo de vida del proyecto mientras se minimiza el alto grado de formalización. El tercer objetivo es hacer el hacer el Framework de Procesos de Eclipse fácil de usar y adoptar. Siendo necesario aplicar a un conjunto amplio de plataformas y aplicaciones.

Los anteriores han sido sólo algunos ejemplos de frameworks que están encaminados a cumplir objetivos en diferentes áreas de las tecnologías de la información. Note que cada uno de los casos mencionados involucra diferentes componentes conceptuales y/o tecnológicos que están fuertemente relacionados. Lo anterior nos lleva a cuestionarnos sobre cual sería el framework que mejor se adapta a una organización o qué características se adaptan mejor al modelo organizacional de una determinada empresa.

3.5 META OBJECT FACILITY - MOF

La consecución de un marco de trabajo que permita hacer una gestión integral del proceso, cumpliendo una semántica clara y bien definida, establece una jerarquía de especificaciones sobre las cuales se soporta su construcción; la OMG[7] define varias de estas especificaciones las cuales se constituyen en estándares para la industria del software y entre las cuales se destacan MOF, UML, CORBA, SPEM y XMI.

El Meta Object Facility (MOF) es un estándar adoptado por la OMG para proveer un framework de gestión de metadatos y fijar un conjunto de servicios para habilitar el desarrollo e interoperabilidad del modelo y los sistemas de manejo de metadatos. Entre los ejemplos de sistemas que usan MOF se incluyen herramientas de desarrollo y modelado, sistemas de data warehouse, repositorios de metadatos etc. Actualmente existen varios perfiles UML que utilizan MOF y tecnologías derivadas de MOF (Especialmente XMI y más recientemente JMI que permiten hacer mapeo de MOF a XML y JAVA respectivamente) [32].

MOF ha contribuido significativamente a los principales núcleos de las nacientes arquitecturas de gestión de modelos de la OMG. En la construcción del modelamiento base establecido para UML, MOF introduce el concepto de metamodelos formales y

[7] OMG: Object Management Group

plataformas independientes de modelos de metadato (algunos ejemplos incluyen varios metamodelos estándar OMG incluyendo UML, el mismo MOF, CWM, SPEM, Java EJB, EDOC, EAI etc.).

3.5.1 Arquitectura MOF

El objetivo de MOF es la especificación y gestión de metadatos en diferentes niveles de abstracción. Una aproximación de modelamiento orientada a objetos de una familia de procesos de software relacionados la haremos usando UML como notación. La Figura 8 muestra las cuatro capas arquitectónicas de modelamiento definidas por la OMG.

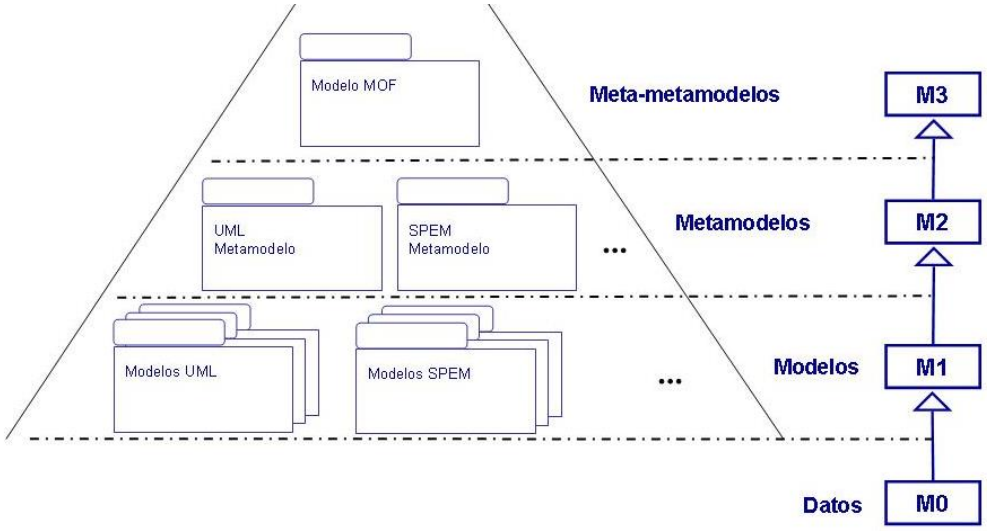


Figura 8 - Niveles conceptuales en MOF

Los elementos incluidos en cada nivel de la arquitectura conceptual son:

3.5.1.1 Nivel de Meta-MetaModelo (M3)

En el nivel superior de la arquitectura conceptual se encuentra el modelo MOF, que es un lenguaje abstracto para la definición de metamodelos. Sus principales constructores son las clases-MOF y las asociaciones-MOF. MOF es utilizado por el metamodelo SPEM para la representación común de todos los conceptos relacionados con la definición del proceso software, conceptos que se incluyen en los metamodelos del nivel M2 [8].

3.5.1.2 Nivel de Metamodelo (M2)

En el nivel M2 de la arquitectura se incluyen los metamodelos necesarios para la integración del modelado y medición del proceso software. Framework-PDS utiliza éste nivel el "*Software Process Engineering Metamodel" (SPEM)*, por medio de la integración y utilización de una herramienta con dicho perfil. SPEM facilita la gestión integrada del proceso software al poder definir los procesos usando una terminología común y estándar.

3.5.1.3 Nivel de Modelo (M1)

La definición del proceso está representada por el nivel M1, en este modelo y de acuerdo a la arquitectura propuesta, se incluyen modelos específicos para la definición y medición del proceso software. Desde el punto de vista de la definición, de pueden incorporar modelos de procesos concretos, como modelos para el desarrollo, mantenimiento, evaluación y mejora, etc. Por ejemplo Rational Unified Process 2001, DMR Macroscope, IBM Global Services Method y Fujitsu SDEM estarían definidos en este nivel. También procesos genéricos como el RUP u otras especificaciones adaptadas a un proyecto dado están en el nivel M1 [27].

3.5.1.4 Nivel de Datos (M0)

En el nivel inferior de la arquitectura se encuentra el **proceso en ejecución** que es un proceso de producción del mundo real, los datos, que son el resultado de la ejecución de los modelos definidos en M1. De esta forma en este nivel se encontrarían los datos relacionados con la ejecución de los modelos de procesos como Framework-PDS.

Aunque los métodos de modelado de procesos difieren ligeramente, todos tienen un aspecto en común: un metamodelo de procesos cuyo propósito es el establecimiento de la terminología a utilizar en el modelado, además de sugerir un nivel de abstracción en el que el modelo de procesos pueda ser elaborado. Como parte de Framework-PDS, es la construcción de una plataforma tecnológica base para la definición de procesos de desarrollo de software, portable y adaptable a cualquier empresa dedicada a este fin. Se ha definido la especificación SPEM como metamodelo base para la consecución de tal objetivo. La tecnología para soportar la automatización de los procesos, de modo que la

plataforma desarrollada tenga cierto nivel de interoperabilidad, es decir, que pueda comunicarse con herramientas y/o plataformas similares para poder realizar las distintas tareas involucradas en un sistema de Workflow corresponde a la WFMC[8] [29].

3.6 SOFTWARE PROCESS ENGINEERING METAMODELING – SPEM

SPEM es una especificación para la definición del ciclo de vida de los procesos y sus componentes, extiende del Unified Modeling Process (UML) con estereotipos de procesos específicos. SPEM es usado para describir un proceso de desarrollo de software o una familia de procesos de desarrollo de software relacionados, y a diferencia de UML, posee las bases conceptuales y específicas que hacen que los diferentes procesos sean interpretados y modelados con mayor exactitud y entendimiento. La especificación SPEM está estructurada como perfil UML y además como metamodelo basado en MOF (Meta Object Facility) de esta manera facilita las funciones de intercambio entre herramientas UML y herramientas y repositorios basados en MOF [27].

3.6.1 SPEM Como un Metamodelo

El metamodelo autónomo SPEM es construido extendiendo un subconjunto del metamodelo físico de UML 1.4. Este subconjunto UML es conocido como **SPEM_Foundation** que proporciona la base necesaria para modelar procesos. Adicionalmente se presenta el paquete **SPEM_Extensions,** el cual adiciona los constructores y semánticas requeridas para la ingeniería de procesos de Software.

3.6.1.1 Modelo Conceptual

En la base del Metamodelo de Ingeniería de Procesos de Software (SPEM) está la idea de que un proceso de desarrollo de software es una colaboración entre las entidades activas abstractas llamadas Roles del proceso que realizan operaciones llamadas actividades en entidades concretas, tangibles llamadas productos de trabajo [27].

[8] The Workflow Management Coalition Specification

La Figura 9 describe un ejemplo de modelo conceptual fundamental usando la notación UML para una clase.

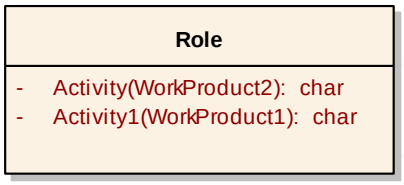


Figura 9 - Modelo Conceptual

Múltiples roles interactúan o colaboran para intercambiar productos de trabajo y provocar la ejecución, o representación de ciertas actividades. El objetivo principal de un proceso es traer unos productos de trabajo a un estado bien definido. Para este modelo, un primer paso consistirá en reestructurar el rol, la actividad y el producto de trabajo. Esto nos conduce al modelo simple mostrado en la Figura 10.

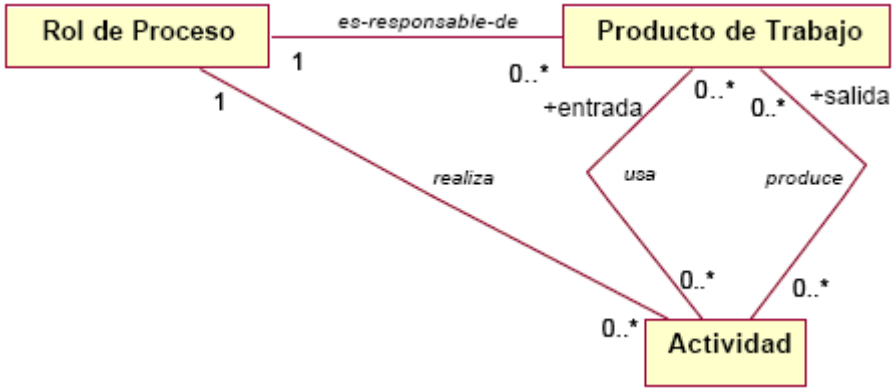


Figura 10 - Modelo Conceptual Reificado

3.6.1.2 Estructura de SPEM

El metamodelo SPEM se estructura básicamente en 5 paquetes: Elementos Básicos (*Basic Elements*), Dependencias (*Dependences*), Estructura del Proceso (*Process Structure*), Componentes del Proceso (*Process Components*), y Ciclo de Vida del Proceso (*ProcessLyfecycle*). La Figura 11 muestra la estructura interna del paquete SPEM_Extensions en términos de sub-paquetes y muestra las dependencias con el paquete SPEM_Foundation.

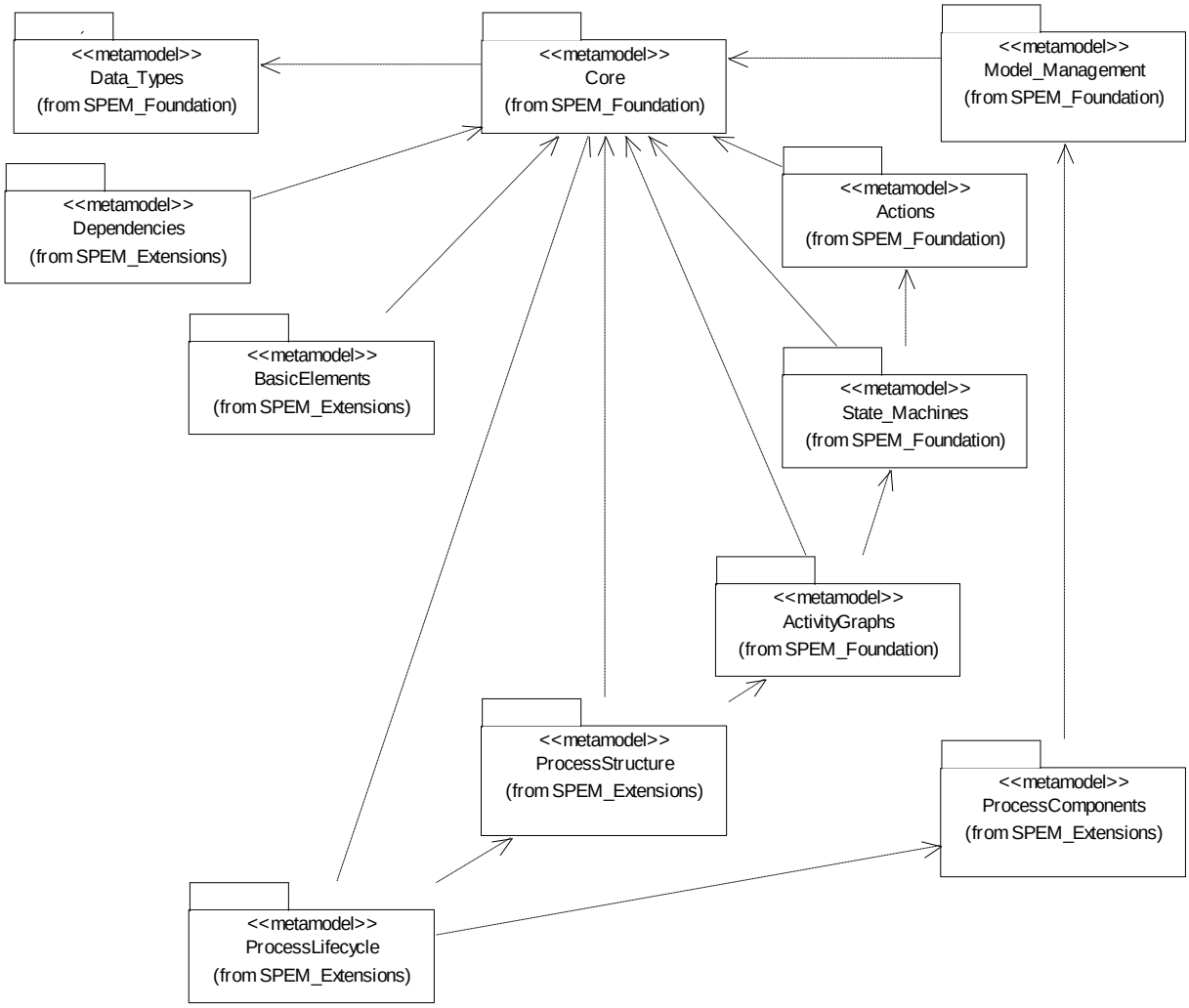


Figura 11 - Estructura general del Metamodelo SPEM [27].

3.6.1.2.1 Paquete Elementos Básicos

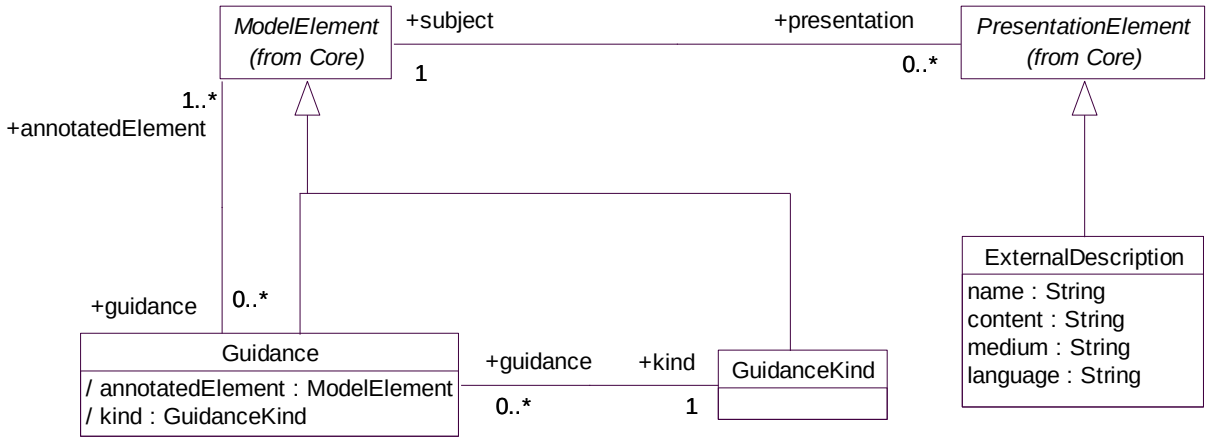


Figura 12 - Paquete Elementos básicos [27].

Los Elementos Básicos (Figura 12) definen los elementos usados para la descripción de procesos y están constituidos básicamente por los elementos *ExternalDescription y Guidance.*

- ***ExternalDescription.*** Cada Elemento del Modelo (*ModelElement*) tiene asociado una o más Descripciones externas (*ExternalDescription*) las que contienen la descripción correspondiente del elemento de modelo para el lector de la descripción del proceso.
- ***Guidance.*** Los elementos guía pueden ser asociados a un elemento del modelo para proveer una información más detallada a los participantes acerca de un elemento de modelo relacionado [27].

3.6.1.2.2 Paquete Dependencias

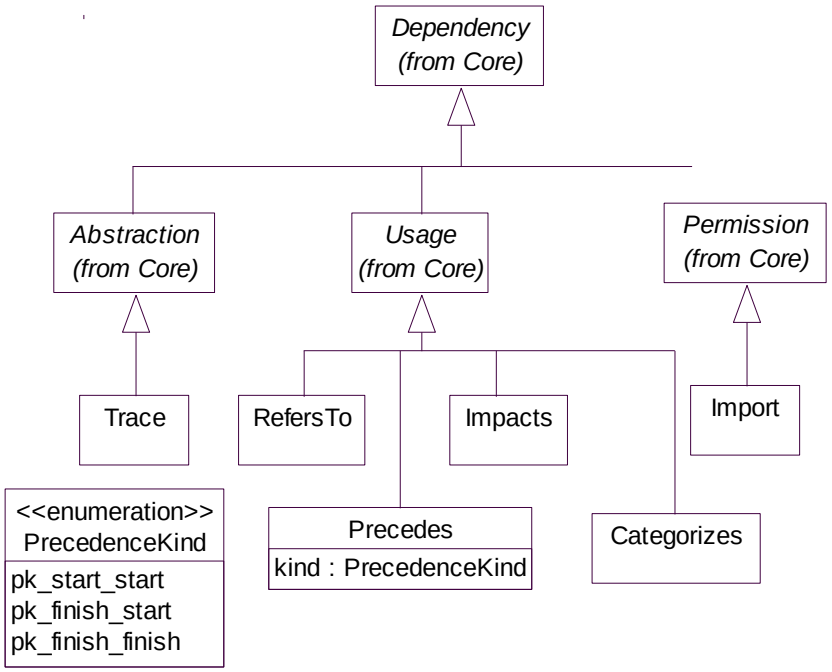


Figura 13 - Paquete Dependencias [27].

La Figura 13 muestra las dependencias contenidas en SPEM, definidas como subclases de SPEM_Foundation dependiente de las clases Abstraction, Usage y Permission. Las cuales tienen las semánticas definidas por UML 1.4[9]

Las siguientes dependencias son soportadas por SPEM para ingeniería de procesos.

- **Categorizes.** Una dependencia *Categorizes* actúa desde un Paquete a un elemento individual de procesos en otro paquete, y proporciona una manera de asociar elementos de procesos con múltiples categorías. Esta característica también es generalmente útil en actos particulares conjuntamente con *Disciplina* para proveer una categorización de alto nivel a todos los elementos.
- **Impacts.** Una dependencia *impacts* actúa de un WorkProduct a otro, e indica que la modificación de un WorkProduct puede invalidar a otra.

[9] En UML, los tipos específicos de Dependencia se definen usando estereotipos, Solo en SPEM, los estereotipos no están permitidos, ellos son definidos usando subclases.

- **Import.** Una Dependencia *Import* denota que los contenidos de un Paquete específico están adicionados al namespace de la fuente del paquete. Este tiene la misma semántica del UML *Import* excepto que en SPEM todos los elementos tienen visibilidad pública.
- **Precedes.** Una dependencia Precedes actúa de una actividad a otra, o de un WorkDefinition a otro, para indicar dependencias inicio-inicio, fin-inicio o fin-fin entre el trabajo descrito dependiendo del valor del atributo de la clase.
  - Si la actividad B tiene una dependencia fin-inicio en la actividad A, entonces B solo puede estar después de que A ha finalizado (secuenciación estricta, no paralelismo).
  - Si la actividad B tiene dependencia fin-fin en la actividad B, entonces B puede finalizar solamente después de que A ha finalizado (el paralelismo es posible y la sincronización va al final).
  - Si la actividad B tiene una dependencia inicio-inicio hacia la actividad A, entonces B puede iniciar solamente después de que A ha iniciado (paralelismo es posible y la sincronización va al inicio).
- **RefersTo.** Una dependencia *RefersTo* actúa de un elemento de procesos a otro, para asegurarse que están incluidos en el mismo ProcessComponent. La situación normal donde esto aplica en cuando el texto de un elemento de procesos, hace referencia, por nombre o contenido, a otro elemento. Para asegurar la consistencia del significado de un texto, Una dependencia *RefersTo* debe ser establecida para dar una representación estructural explícita de tal dependencia, de modo que cuando el elemento de referencia se incluye en un ProcessComponent, el elemento referido debe también ser incluido.
- **Trace.** Una dependencia Trace, actúa entre WorkDefinition's y es principalmente utilizado para hacer un seguimiento de los requerimientos y cambios a través de los modelos. Esta dependencia tiene la misma semántica de su similar en UML [27].

3.6.1.2.3 Paquete Estructura del Proceso (Figura 14)

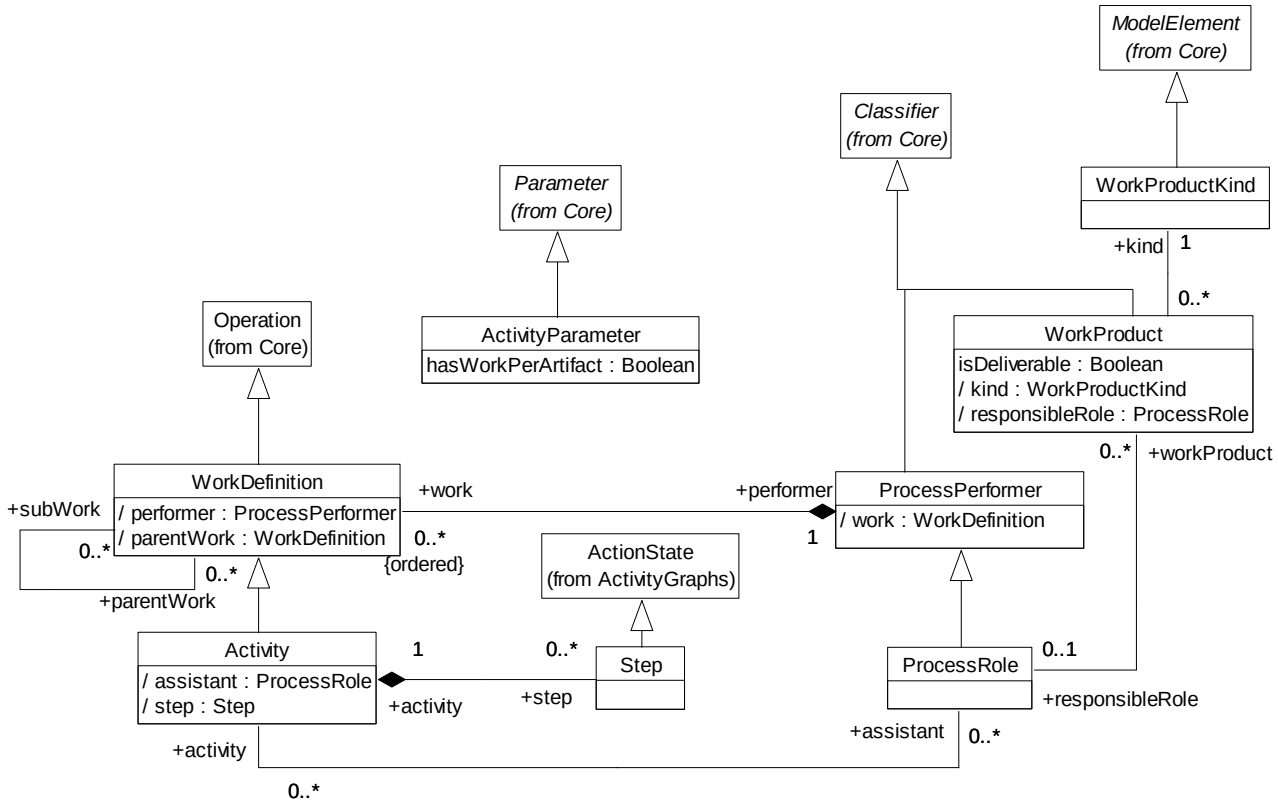


Figura 14 - Paquete Estructura del Proceso [27].

Este paquete muestra la estructura principal de los elementos con los cuales la descripción de un proceso es construida.

WorkProduct. Un producto de trabajo (WorkProduct) es cualquier elemento producido, consumido, o modificado por un proceso. Esto puede ser una pieza de información, un documento, un modelo, código fuente y demás. Un *WorkProduct* describe una clase de producto de trabajo producido en un proceso [27].

WorkProductKind. Un WorkProductKind describe una categoría del producto de trabajo, puede ser un documento de texto, un modelo UML, un Ejecutable, Librería de Código y demás. El rango de clases de productos de trabajo es dependiente del proceso que está siendo modelado.

WorkDefinition. Es una clase de Operación que describe el trabajo desarrollado en los procesos. Su subclase principal es Actividad, pero fase, iteración y ciclo de vida (en el

paquete del proceso LifeCycle) son también subclases del WorkDefinition. WorkDefinition, no es una clase abstracta, y las instancias de WorkDefinition se pueden crear a si mismo para representar piezas compuestas de trabajo que serán separadas en el futuro. WorkDefinition tiene entradas explícitas y salidas dirigidas hacia ActivityParameter.

***Activity**. E*s la principal subclase de WorkDefinition. Esta describe una parte del trabajo desarrollado por un ProcessRole: las tareas, operaciones y acciones que son desempeñadas por un rol o las que el rol puede asistir. Un Activity se puede componer de elementos atómicos llamados pasos (Steps).

ProcessPerformer. Un *ProcessPerformer* o ejecutor de proceso define la ejecución para un conjunto de definiciones de trabajo (WorkDefinition) en un proceso. ProcessPerformer es una representación abstracta del "proceso completo" o uno de sus componentes.

ProcessRole. Un Rol de Proceso define responsabilidades sobre productos de trabajo específicos y define los roles que ejecutan y asisten en actividades específicas. En RUP, ejemplos de un ProcessRole son *Arquitecto, Analista, Revisor Técnico o Administrador de Proyectos* para nombrar algunos [27].

3.6.1.2.4 Paquete Componentes del proceso

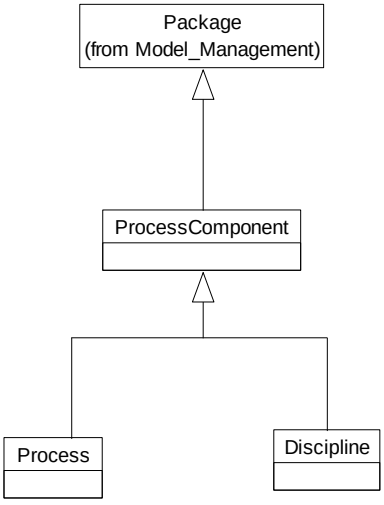


Figura 15 - Paquete Componentes del Proceso [27]

La Figura 15 detalla el paquete de Componentes del Proceso. Las clases en este paquete se refieren a la división de una o más descripciones del proceso dentro de las partes que el mismo contiene y que pueden ubicarse bajo la administración de la configuración y control de la versión.

ProcessPackage. Al igual que en UML, un paquete puede contener procesos propios e importar elementos de definición de procesos. Los elementos Activity y WorkDefinition son contenidos, respectivamente, por ProcessRole y ProcessPerformer; Las maquinas de estado son contenidas por WorkProducts y sus propios estados y transiciones. Los diagramas de actividad pueden ser contenidos por Paquetes, Clasificadores o Características de conducta; Otros Elementos de Modelo SPEM pueden ser contenidos por Paquetes de procesos.

ProcessComponent. Un ProcessComponent es una parte del proceso que es consistente internamente y puede ser reutilizado junto con otros ProcessComponents para ensamblar un proceso completo.

Un ProcessComponent importa un conjunto no arbitrario de elementos para definición de procesos, modelados en SPEM por ModelElements. Dicho sistema debe ser contenido a sí mismo; esto significa que no existen dependencias 'RefersTo' de los elementos que están dentro del componente a los que no están dentro de él. Deben ser internamente consistentes en el sentido de que las multiplicidades y restricciones definidas por el metamodelo sean satisfechas dentro del alcance del componente.

Process. Se entiende un proceso como un *ProcessComponent* que es capaz de mantenerse aislado de principio a fin. El proceso se distingue de un componente de proceso normal por que no está pensado para ser accedido por otros componentes.

Discipline. Una Disciplina es una especialización particular de Paquete que divide las actividades dentro de un proceso de acuerdo a un tema común. El particionamiento de actividades de esta forma implica que la guía asociada (Guidance) y la salida de WorkProducts se categorizan similarmente en un tema. La inclusión de una actividad en una Disciplina es representada por la dependencia de Categorías, con una restricción adicional de que toda Actividad es categorizada por exactamente una Disciplina. Por ejemplo, RUP define nueve disciplinas: *Modelado de negocios, Administración de*

requerimientos, Análisis y Diseño, Implementación, Pruebas, Despliegue, Gerencia del proyecto, Administración de configuración y de cambios y ambiente.

3.6.1.2.5 Paquete ciclo de vida del proceso

En este paquete (Figura 16), se introducen los elementos de definición de procesos que ayudan a describir como puede ser ejecutado el proceso. Estos elementos especifican o restringen todo el comportamiento la ejecución del proceso, y se utilizan como asistentes en el planeamiento, ejecución y monitoreo del proceso. Para dirigir su representación, podemos restringir el orden en el cual las actividades deben ser, o pueden ser ejecutadas. También es necesario definir la "forma" del proceso en un determinado tiempo, y la estructura del ciclo de vida en términos de fases y de iteraciones.

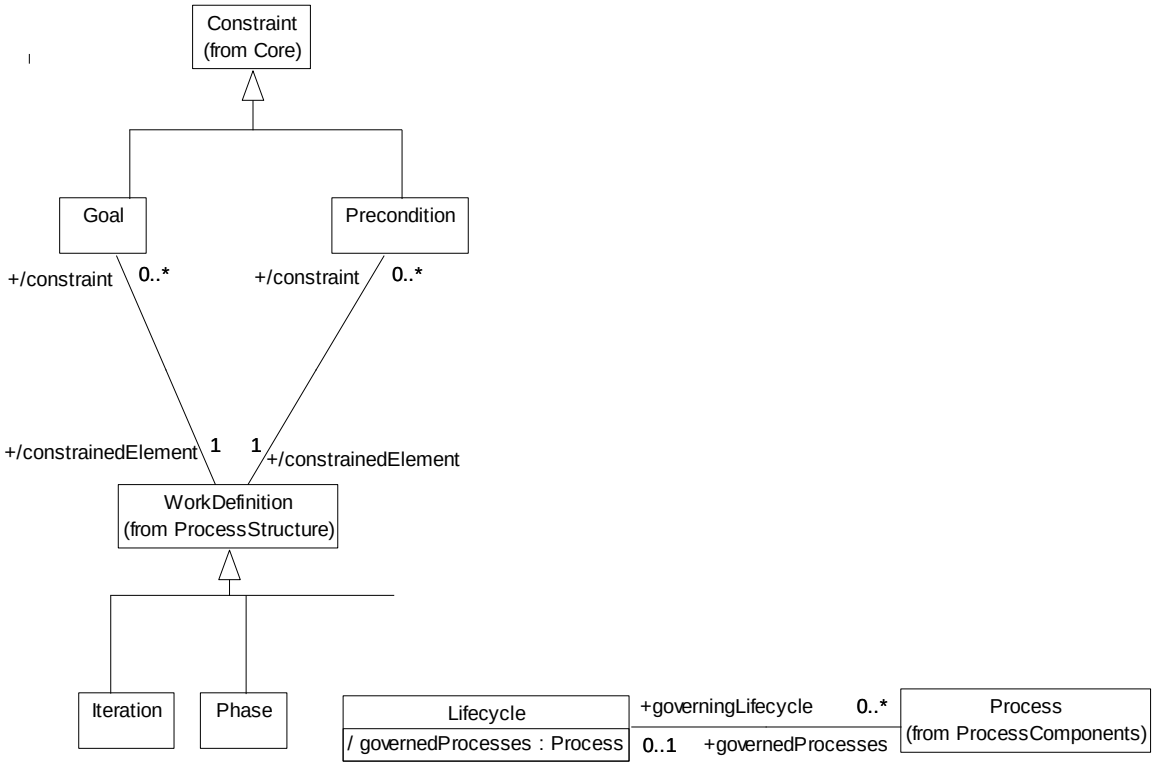


Figura 16 - Paquete Ciclo de Vida [27].

Phase. Una fase es una especialización de WorkDefinition tal que su condición previa define los criterios de entrada de la fase y su meta (a menudo llamada "milestone") define los criterios de la salida de la fase. Las fases son definidas con una restricción adicional de secuencialidad, que es, sus representaciones son ejecutadas con una serie

de fechas de tiempos de programación (milestone) que normalmente toman un mínimo (o ningún) traslapo de sus actividades en el tiempo. Por ejemplo, RUP define 4 fases secuenciales: I*nicio, Elaboración, Construcción y Transición.* RUP define estas fases como consecuencia de cierto número de iteraciones, las cuales son los flujos de trabajo que tienen menos tiempo de programación.

LifeCycle. El ciclo de vida de un proceso es definido como una secuencia de fases que alcanzan una meta específica. Este define el comportamiento de un proceso completo que será representado en un proyecto o un programa dado.

Iteration. Una Iteración es un componente WorkDefinition que se caracteriza por tener el menor "milestone" [27].

3.6.2 SPEM como Perfil UML

Un perfil UML es una variación UML que usa los mecanismos de extensión de UML de una forma estandarizada, para un propósito particular. Un estereotipo de perfil de un paquete contiene una o más extensiones relacionadas de la semántica UML estándar. Esas extensiones normalmente tienen la intención de personalizar UML para un dominio específico. Los perfiles pueden contener estereotipos, definición de etiquetas y restricciones, estos también pueden contener tipos de datos que son usados por las etiquetas de definición para dar claridad informalmente de los tipos de valores que están asociados a las etiquetas de definición.

Un paquete del perfil puede especificar una librería del modelo relacionado e identificar un subconjunto del metamodelo UML que es aplicable al perfil. En principio, los perfiles refinan simplemente la semántica estándar del UML agregando otras restricciones e interpretaciones que capturan la semántica de un dominio específico y los patrones de modelamiento. Los perfiles no adicionan nuevos conceptos fundamentales.

Como perfil UML, SPEM define capacidades de modelado destinadas al dominio concreto de los procesos software aprovechando las facilidades de expresividad que proporciona UML. Por ejemplo, el modelado de casos de uso, que en ocasiones se usa para modelar procesos, no está definido como una facilidad específica de SPEM, pero puede ser heredada de UML. En general, el uso de UML como base para el modelado de procesos

aprovecha las ventajas en cuanto al posible alineamiento con otros lenguajes de modelado. Además, UML cuenta con tecnología de soporte (herramientas CASE variadas) que también puede ser aprovechada en el dominio de los procesos software.

Para definir un perfil de UML para SPEM, se deben seguir los siguientes pasos:

1. Identificar que subconjunto de clases del metamodelo UML se debe incluir en el perfil.
2. Para la mayoría de clases en el metamodelo SPEM, identifique una "clase base" en el subconjunto del metamodelo UML que, cuando esté estereotipada apropiadamente, actúe en lugar de la clase de SPEM. El hecho es que SPEM es definido a sí mismo como una extensión del subconjunto UML que hace esto de una forma más directa. Para la primera clase (GuidanceKind) en el metamodelo SPEM donde la técnica de clase base no aplica. La semántica de instancias de esa clase se emula usando estereotipos UML.
3. Para cada atributo y asociación en el metamodelo SPEM defina una manera de emular ese atributo o asociación. En el perfil SPEM, los atributos son emulados por medio de valores etiquetados. La mayoría de asociaciones tienen analogías parecidas en el metamodelo UML. Aquellos que no tienen tratamiento especial se detallan posteriormente.
4. Para esas partes del subconjunto UML que tienen un mapeo aceptable en los conceptos de SPEM, pero no son usados directamente para emular el metamodelo SPEM, muestre como son mapeados en los conceptos relacionados dentro de SPEM, Esto aplica particularmente a la utilización de diagrama de casos de uso.
5. Aplique las restricciones adicionales sobre el metamodelo UML que deben ser aplicadas por el uso del perfil.
6. Defina los íconos notacionales para los conceptos SPEM que son representados por estereotipos UML [30].

3.6.2.1 Estereotipos del Perfil SPEM

La Tabla 1 proporciona un completo resumen de todos los estereotipos del perfil SPEM. Los siguientes estereotipos son utilizados por conveniencia notacional:

Estereotipo	Clase Base	Estereotipo Padre	Notación
WorkProduct	Core::Class ActivityGraphs::ObjectFlow State		
ActivityParameter	Core::Parameter		
Goal	Core::Constraint	postcondition	
Precondition	Core::Constraint	Precondition	
WorkDefinition	Core::Operation ActivityGraphs::ActionState UseCases::UseCase		
Step	ActivityGraphs::ActionState		
Guidance	Core::Comment		
ExternalDescription	Core::PresentationElement		
Activity	Core::Operation ActivityGraphs::ActionState UseCases::UseCase	WorkDefinition	
ProcessPerformer	UseCases::Actor		
ProcessRole	UseCases::Actor	ProcessPerformer	
ProcessPackage	ModelManagement::Package		
ActivityParameter	Core::Parameter		
Phase	Core::Operation ActivityGraphs::ActionState UseCases::UseCase	WorkDefinition	
Iteration	Core::Operation ActivityGraphs::ActionState UseCases::UseCase	WorkDefinition	
LifeCycle	Core::Operation	WorkDefinition	

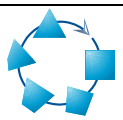
	ActivityGraphs::ActionState UseCases::UseCase		
Discipline	ModelManagement::Package	ProcessPackage	
ProcessComponent	ModelManagement::Package	ProcessPackage	
Process	ModelManagement::Package	ProcessPackage	
Document	Core::Class ActivityGraphs::ObjectFlow State	WorkProduct	
UMLModel	Core::Class ActivityGraphs::ObjectFlow State	WorkProduct	
Guideline	Core::Comment	Guidance	
Technique	Core::Comment	Guidance	
UMLPRofile	Core::Comment	Guidance	
ToolMentor	Core::Comment	Guidance	
CheckList	Core::Comment	Guidance	
Template	Core::Comment	Guidance	
Trace	Core::Abstraction		
refersTo	Core::Usage		
categorizes	Core::Usage		
precedes	Core::Usage		
impacts	Core::Usage		
import	Core::Permission		
governs	Core::Abstraction		
assist	Core::Association		
perform	Core::Association		

Tabla 1 - Estereotipos del perfil SPEM [27].

Los estereotipos, proporcionan una facilidad visual y cognitiva de la definición real, estricta y teórica que se tiene de las meta clases o clases en el metamodelo SPEM, permitiendo por medio de éstos, conseguir modelos de procesos menos abstractos y mas comprensibles.

3.7 XMI: XML METADATA INTERCHANGE.

XMI es la tecnología adoptada por la OMG para el intercambio de modelos en forma serializable, se focaliza en el intercambio de metadatos MOF, que son los metadatos que se adaptan al modelo MOF [34]. Su principal propósito, es facilitar el intercambio de metadatos, en entornos distribuidos heterogéneos, entre diferentes tipos de herramientas software y, en especial, entre herramientas de modelado basadas en UML y repositorios de metadatos basados en la propuesta MOF. Para ello, XMI integra tres normas que son claves en la industria actual de sistemas de información. Estos estándares de facto son:

a) XML: Lenguaje de Marcas Extensible (eXtensible Markup Language), especifica un formato universal para documentos y datos estructurados en la web.
b) UML: Lenguaje Unificado de Modelado (Unified Modeling Language), estándar propuesto por el OMG.
c) MOF: Facilidad para Meta -Objetos (Meta-Object Facility), ya comentado [33].

La integración de estos estándares en XMI une lo mejor de las tecnologías sobre metadatos y modelado de OMG y W3C, permitiendo que los desarrolladores de sistemas de información puedan compartir modelos y otros metadatos vía Internet (Figura 17). El uso combinado de las tres normas anteriores permite reducir de forma drástica la cantidad de formatos de intercambio de metadatos [30].

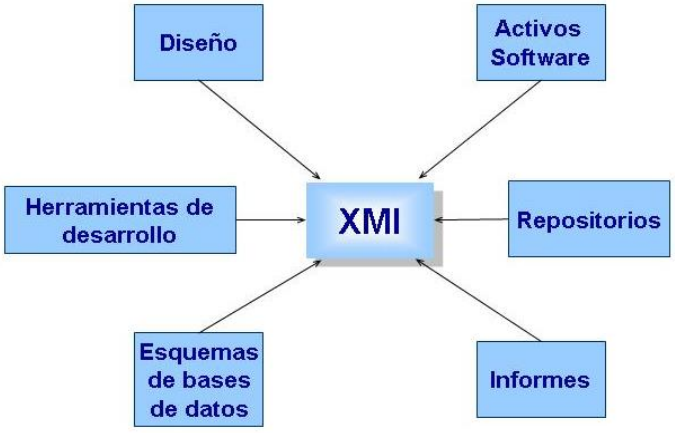


Figura 17 - Intercambio abierto entre distintas herramientas mediante XMI.

XMI permite que los metadatos puedan ser intercambiados como flujos o archivos con un formato estándar basado en XML. La arquitectura completa ofrece un amplio rango de

posibilidades de implementación a los desarrolladores de herramientas, repositorios y marcos de trabajo orientados a objetos. Para ello, la especificación XMI define los siguientes aspectos:

- Un conjunto de reglas de producción de DTD's para transformar metamodelos basados en MOF en DTD's (*Document Type Definition*) en formato XML. Estos DTD's se utilizan como sintaxis para la construcción de documentos XMI. Por tanto, dado un metamodelo MOF (nivel M2) expresado en XMI, el DTD asociado permite validar modelos MOF (nivel M1).
- Un conjunto de reglas de producción de documentos XMI, que sirven para la codificación y decodificación de metadatos basados en MOF, es decir, para representar los metamodelos y modelos MOF (niveles M2 y M1) en formato XMI.
- Una serie de principios de diseño para los DTD's y flujos XML basados en XMI.
- Archivos con los DTD's específicos del metamodelo de UML y del Modelo MOF.

Dadas todas estas características proporcionadas por XMI, los procesos modelados en SPEM tendrán su representación en esta tecnología para facilitar su intercambio entre diferentes herramientas que interpretan metadatos.

3.8 FLUJOS DE TRABAJO - WORKFLOWS

Un workflow o producto de trabajo es la automatización total o parcial de un proceso de negocio, durante la cual, la información, los documentos, o las tareas son pasadas de un participante a otro por una acción conforme a un conjunto de reglas procedimentales, mientras que las personas se ocupan solamente de las excepciones. Un workflow puede ser organizado manualmente, en la práctica, la mayor parte de los workflows son normalmente organizados en contexto de los sistemas de información para proveer el apoyo computarizado a la automatización procesal [35]. Un workflow comprende un número de pasos lógicos, conocidos como actividades. Una actividad puede involucrar interacción manual con el usuario o ser ejecutada por una máquina [34].

Cabe mencionar que los workflows son solo un camino para la información, para reducir tiempo, dinero y esfuerzo en la ejecución de un proceso de negocio. Las funciones más comunes que proporcionan los workflows son:

- Asignación de tareas al personal.
- Aviso al personal de tareas pendientes.
- Permitir la colaboración en las tareas comunes.
- Optimización de recursos humanos y técnicos, adoptándolos al funcionamiento de la empresa.
- Automatización y optimización de las secuencias de los procesos de negocio.
- Agilización de los procesos de negocio, logrando un mejor servicio al cliente.
- Control y seguimiento de dichos procesos [37].

Un motor workflow es un software que controla la ejecución de las actividades definidas en el workflow. Típicamente el motor provee facilidades para:

- Interpretación de la definición de procesos.
- Control de las instancias de los procesos: creación, activación, terminación, etc.
- Navegación entre actividades.
- Soporte de interacción con el usuario.
- Control de datos al usuario o hacia aplicaciones.
- Invocación de aplicaciones externas [37].

La automatización de los procesos de negocio de una empresa trae beneficios como la reducción del tiempo de búsqueda de documentos o menos gasto en papelería, estos problemas son los primeros que se atacaron con la tecnología de workflows. Algunas razones por las cuales las organizaciones podrían considerar adoptar una solución de workflow son:

- Eficiencia en los procesos y estandarización de los mismos conduce a una reducción de costos dentro de una empresa; la estandarización de los procesos lleva a tener un mayor conocimiento y mejor calidad de los mismos. Con la utilización de la tecnología de Workflow es posible monitorear el estado actual de las tareas así como también observar como evolucionan los planes de trabajo realizados. Permite ver cuales son los embotellamientos dentro del sistema, es decir aquellas tareas o decisiones que están requiriendo de tiempo no planificado y se tornan en tareas o decisiones críticas.
- Asignación de tareas a los stakeholders mediante la definición de roles dentro de la empresa.

- Se asegura que los recursos de información (aplicaciones y datos) van a estar disponibles para los stakeholders cuando ellos los requieran.
- El nuevo diseño de procesos fomenta a pensar los procesos de una manera distinta a la forma jerárquica tradicional.

La WfMC define en el documento Modelo de Referencia Workflow (*Workflow Reference Model*) una API provista de 5 interfases para la interoperabilidad de diferentes productos con un motor workflow. El modelo de referencia propuesto por la WfMC intenta reunir las características comunes de cualquier producto para la gestión de workflows de manera que sea posible la interoperabilidad entre ellos, a través de estándares comunes para cada una de las funciones que se puedan realizar. En primer lugar, se han identificado las distintas áreas funcionales y a continuación se han desarrollado especificaciones para la implementación de las mismas, garantizando la interoperabilidad entre los distintos componentes y su integración con otras aplicaciones informáticas [35].

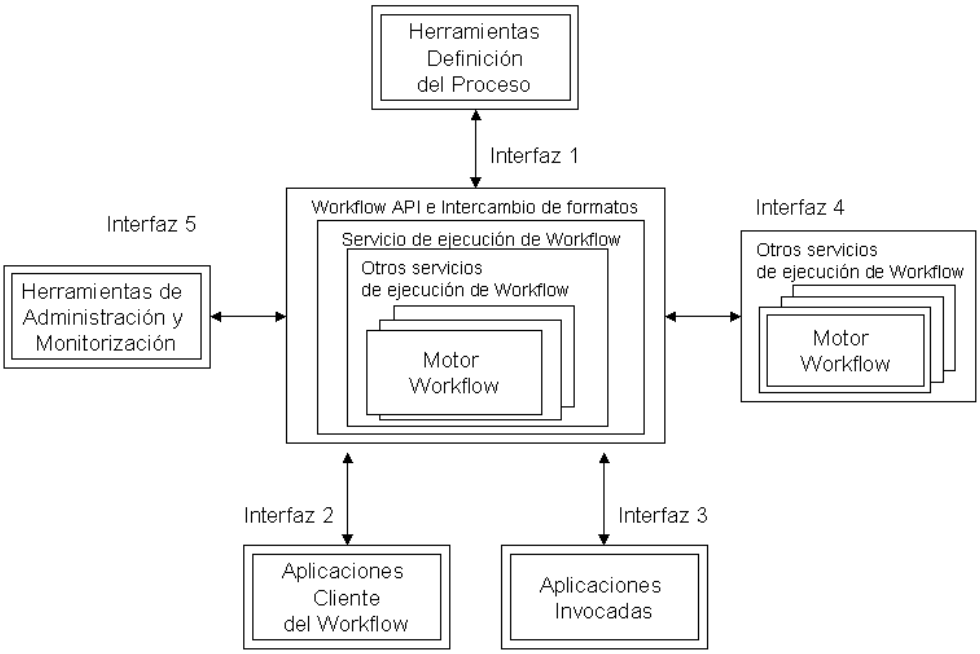


Figura 18 - Modelo de referencia WfMC

Dentro de todo el contexto del Modelo de Referencia Workflow, el Framework PDS se centrará específicamente en integrar una herramienta que cumpla con la interfaz 1, que especifica el formato de intercambio común para soportar la transferencia de definiciones

de procesos entre productos diferentes, utilizando un Lenguaje de definición de Procesos (*XML Process Definition Language* - XPDL)

3.8.1 Lenguajes de Definición de Procesos – XPDL

El Lenguaje de Definición de Procesos (*XML Process Definition Language* - XPDL) es el lenguaje definido para trabajar con la interfaz 1 de un motor workflow.

XPDL permite la transferencia de definiciones de procesos entre productos diferentes. Esta especificación usa XML como el mecanismo para el intercambio de definición de proceso. El XPDL forma un estándar de intercambio común que ofrece soporte a los productos con representaciones internas arbitrarias de definiciones de proceso una función de importación/exportación para mapearlos de/hacia el estándar desde los límites del producto.

Una variedad de diferentes mecanismos pueden ser usados para transferir datos de definición de proceso entre sistemas según las características de varios escenarios de negocio. En todos los casos, la definición de proceso debe ser expresada en una forma consistente, que es derivada del conjunto común de objetos, relaciones y atributos que expresan sus conceptos inferiores.

Una de las claves del XPDL es que puede administrar y usar información de diferentes herramientas, y uno de los elementos más importante es un constructor genérico que soporta atributos específicos del fabricante para usarlos dentro de la representación de procesos [36].

En la Figura 19 se presenta el meta-modelo de Intercambio de Definición de Procesos con las principales entidades y relaciones que participan en XPDL, éste describe las entidades de alto nivel contenida en una Definición de Proceso, sus relaciones y atributos (incluso algunos que están definidos para simulaciones o monitoreo). También define varias convenciones para agrupar Definiciones de Procesos dentro del modelo de procesos relacionado y el uso de datos de definición comunes [36].

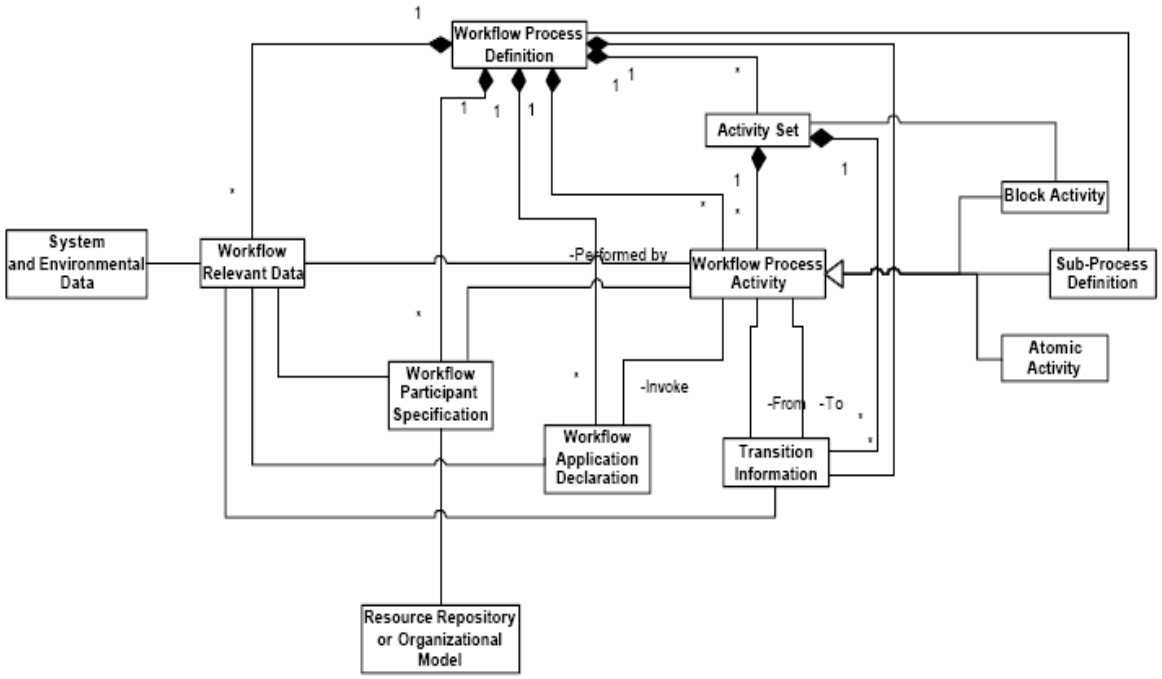


Figura 19 – Metamodelo de las entidades XPDL

Para una definición de procesos las siguientes entidades se deben definir en forma directa o indirectamente:

- *Workflow Process Activity*: Usado para definir cada actividad elemental que se hacen en un proceso workflow.
- *Transition Information*: Describe las posibles transiciones entre actividades y las condiciones que la habilitan o deshabitan, durante la ejecución del workflow.
- *Workflow Participant Specification*: Puede ser un "rol", un "humano", u "otro sistema". Esto es un nivel de abstracción entre el ejecutor real y la actividad que será ejecutada.
- *Workflow Application Declaration*: Es una lista de todas las aplicaciones o herramientas requeridas e invocadas por el proceso workflow definido dentro de la definición de procesos.
- *Workflow Relevant Data*: Representan las variables de un proceso workflow.

La Interfaz 1 también define una separación formal entre el desarrollo y el entorno de ejecución, permitiendo a una definición de proceso, generada por una herramienta, ser usada como entrada para un número de diferentes productos workflow.

3.9 MODELO INTEGRAL PARA LA MEJORA DEL PROCESO DE DESARROLLO DE SOFTWARE - AGILE SPI

Agile SPI (Agile Software Process Improvement), es un modelo integral para la mejora del proceso de desarrollo de software adaptado a la industria colombiana, propuesto por el grupo de desarrollo SIMEP-SW y presenta las siguientes consideraciones controladoras de su arquitectura [2]:

De la industria:

- La industria Colombiana está naciendo, las empresas apenas se están organizando como tal.
- Hay un gran interés en innovar en cuanto a productos o servicios, pero no en cuanto a la organización de sus procesos.
- Los procesos seguidos son ad-hoc, livianos o caóticos, no hay gran motivación en mejorar este aspecto.
- Dada la realidad de la industria Colombiana, una industria en crecimiento, cuyas organizaciones son pequeñas en número de personas, el modelo debe ser liviano, fácil de implementar
- El aseguramiento de la calidad está más enfocado a la prueba que a tener un programa completo de aseguramiento de calidad.

De la academia:

- La academia está acogiendo nuevas formas de desarrollo, sin embargo hay más empatía por las tecnologías, que por los procesos o metodologías.
- La mejora del proceso se considera un trabajo remoto, al alcance de pocos y para empresas gigantescas, se desconoce que es posible adelantarlo desde los equipos, y sus integrantes y nivel empresarial, así las empresas seamos dos.
- Las universidades que imparten este trabajo muestran los modelos de calidad, su interpretación y toda la infraestructura requerida para hacerlo, esto puede motivar trabajar por mejorar el estado del arte, pero desmotiva a la hora de trabajar por cambiar el estado de la práctica [2].

Estas son las influencias iniciales para el AGILE SPI, las cuales, a medida que se adelante el trabajo con el sector empresarial, con la retroalimentación adquirida a lo largo del proyecto, permitirá completar el ciclo de influencias y madurar el modelo integral de mejora del proceso de software para que sea asimilable por la industria del software Colombiana.

3.9.1 Arquitectura AGILE SPI

La arquitectura preliminar de AGILE SPI, presenta los siguientes componentes [2]:

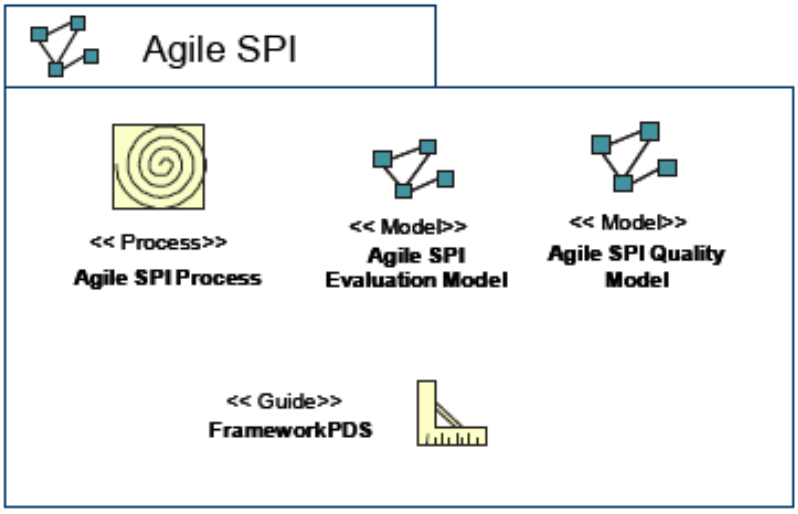


Figura 20 - Arquitectura conceptual de AGILE SPI

- Agile SPI Process es un proceso ágil de guía a un programa de mejora de procesos en el marco de un proyecto de mejora. Es un proceso que cuenta con los elementos básicos para hacer posible que una empresa que está surgiendo, una empresa pequeña o mediana, pueda adelantar esfuerzos hacia la adecuación de un proceso de desarrollo acorde a sus necesidades.
- Un modelo de calidad liviano, que integre personas, equipos, proceso y producto, y que guíe la organización de las personas y los equipos, las disciplinas y las áreas de trabajo asociadas a la definición, aplicación y mejora del proceso hacia un nivel de madurez definido.
- Un modelo de evaluación que permita identificar y diagnosticar problemas de la industria en cuanto al proceso y que permita trazar unos planes de mejora de acuerdo a un modelo/estándar de calidad definido.

- Un marco conceptual y tecnológico para la definición y visualización de Procesos de Desarrollo de Software (Framework PDS)
- Una comunidad regional con proyección nacional en Internet, que permite ingeniería de software de soporte [2].

Cada uno de estos elementos del AGILE SPI está desarrollado de manera integrada por grupos independientes, manteniendo la autonomía de las disciplinas y de su correspondiente módulo, ya que la idea es que cada uno de ellos sea capaz de integrase a otros modelos; por ejemplo, podría quererse usar el AGILE SPI Process con ISO:9000, por tanto el proceso debe ser independiente del modelo de calidad a seguir. De igual forma, podría decidirse usar el modelo IDEAL para armar el proyecto de mejora y usarse AGILE SPI Quality Model. Por ello, el establecimiento de la Arquitectura de AGILE SPI parte de los controladores más significativos.

3.9.2 El Framework-PDS en el AGILE SPI

En la Figura 20, se ilustra la arquitectura conceptual del modelo AGILE SPI, en la cual, se observa el módulo del Framework PDS. Este módulo es de gran importancia en los objetivos del modelo AGILE SPI, dado que es una plataforma conceptual y tecnológica, flexible y basada en un modelo de referencia que facilita a las empresas de software la definición, visualización y mejoramiento de sus propios procesos de desarrollo. Tiene como base de su estructura el metamodelo SPEM para el modelado del proceso, facilitando su definición en formato XMI. Todas estas características permiten que los procesos de una compañía, estén visibles, documentados y pueden ser utilizados en diferentes plataformas según lo requiera cada compañía.

Framework PDS integra un mecanismo de transformación del lenguaje XMI a XPDL, favoreciendo la portabilidad del proceso definido a la interfaz 1 de la WfMC, proporcionando escalabilidad y ofreciendo la posibilidad de interoperar con otras herramientas y componentes amparados bajo esta especificación. Lo anterior facilita el cumplimiento del objetivo inicial del proyecto propuesto como componente del Modelo Integral para la Mejora de Procesos de Software, AGILE SPI.

Dado que el alcance del Framework PDS no es solo tecnológico sino que además es metodológico y conceptual, proporciona una base de conocimiento centrada en las

tecnologías de procesos de software que puede ser aprovechada por la industria del software colombiana en sus programas de mejora de procesos, donde cada una de las conclusiones y resultados obtenidos a partir de este trabajo quedarán a disposición de la comunidad empresarial y académica lo que incluye un conjunto de herramientas, plantillas, informes técnicos y demás documentos con el fin de aportar a la mejora de la calidad y la productividad de la ingeniería del Software en el país.

3.10 TRABAJOS RELACIONADOS CON EL FRAMEWORK-PDS

Aunque los trabajos relacionados con el Framework-PDS son pocos, en este aparte se relaciona de manera resumida el desarrollo que acerca de plataformas para la definición y visualización de procesos de software han adelantado otros autores, que parcialmente tienen alguna similitud con el trabajo desarrollado en esta investigación. Se pretende de esta manera, identificar claramente las diferencias y similitudes que el Framework-PDS tiene con estas plataformas.

3.10.1 MANTIS: Definición de un Entorno para la Gestión del Mantenimiento de Software

El objetivo de este trabajo es Definir un entorno extendido de ingeniería del software que permita abordar de forma integrada (desde una perspectiva amplia de proceso de negocio), la gestión de proyectos de mantenimiento del software.

Se incide en la mejora de la gestión de proyectos de mantenimiento de software, es decir, proyectos cuyo objetivo es prestar un servicio de mantenimiento de software, se trata de aportar la rigurosidad propia de los métodos ingenieriles no sólo a los procesos puramente de ingeniería del software (desarrollo o mantenimiento) sino también a los otros procesos de soporte y organizacionales, que son completamente necesarios para mejorar la calidad y eficiencia en la realización de los proyectos software. Es un entorno cuyas principales características son: integración por medio de herramientas, orientación a procesos, especialización en mantenimiento, escalabilidad y adaptabilidad. Las tres clases de elementos del Entorno MANTIS son: Un marco de trabajo conceptual para la gestión de proyectos de mantenimiento, que engloba una arquitectura conceptual multinivel (basada en el estándar OMG MOF); Una colección de procedimientos para los

procesos de gestión y organizacionales (según son definidos en el modelo de ciclo de vida de ISO 12207); y Una colección de prototipos definidos como componentes software del Entorno [30].

Las tres clases de elementos del Entorno MANTIS son:

1) Un marco de trabajo conceptual para la gestión de proyectos de mantenimiento, que engloba una arquitectura conceptual multinivel (basada en el estándar OMG MOF), un sistema de procesos basado en los estándares ISO, una colección de ontologías (del mantenimiento, de los flujos de trabajo, y de la medida) que pueden ser compartidas por las herramientas software y por los agentes humanos, y una colección de metamodelos representados en forma de documentos XML, DTD o XMI.
2) Una colección de procedimientos para los procesos de gestión y organizacionales (según son definidos en el modelo de ciclo de vida de ISO 12207), que complementan y mejoran la metodología MANTEMA, un completo y preciso modelo de cómo llevar a cabo el proceso de mantenimiento.
3) Una colección de prototipos definidos como componentes software del Entorno. La arquitectura software del sistema MANTIS establece tres tipos de componentes:

- Verticales: Herramientas para automatizar un determinado tipo de actividad; incluyen, entre otros, un gestor de peticiones de modificación, una herramienta para la gestión de recursos humanos entre una cartera de proyectos de mantenimiento, o un gestor de cuestionarios de evaluación de la madurez de un servicio de mantenimiento.
- Horizontales: Dan soporte a todo el Entorno. Los prototipos desarrollados son un gestor del repositorio integrado de datos y metadatos (en formatos XML y XMI), una herramienta para metamodelización, un gestor de la base de conocimientos y una interfaz de integración.
- Externas: No incluidas directamente en el Entorno MANTIS, pero que pueden ser invocadas desde sus herramientas internas. Las principales herramientas externas consideradas en el Entorno MANTIS son los "Sistemas de Gestión de Flujos de Trabajo", que se proponen como "motor de proceso" para la reificación de los procesos, es decir, para automatizar el seguimiento y control de los proyectos.

Si realizamos un análisis comparativo entre los elementos del Entorno MANTIS con los módulos del Framework PDS, encontramos similitudes interesantes. El primer elemento de MANTIS coincide con el Framework PDS en ser un marco de trabajo basado en MOF, con un sistema de procesos asociado y que utiliza el metamodelo XMI. La diferencia radica en que MANTIS se enfoca al mantenimiento de los procesos, Framework PDS está enmarcado en un programa de mejora de procesos de desarrollo de software, pero si se desea, su utilidad puede ser expandida a los procesos restantes de una organización.

El tercer elemento de MANTIS contiene una colección de prototipos que ofrecen soporte a las tareas desarrolladas por este entorno, como lo son el repositorio de datos (XML y XMI), la herramienta de metamodelado basada en SPEM, un gestor base de conocimiento y una interfaz de integración. Todos estos elementos tienen su equivalente en el Framework PDS, sin embargo, como se mencionó anteriormente, el objetivo en conjunto es diferente.

3.10.2 GenMETRIC

Es una herramienta genérica y extensible para la definición, cálculo y visualización de métricas software. Esta herramienta, suministra soporte a la definición y gestión de métricas tanto de artefactos, como de procesos software. Igualmente, ofrece soporte al metamodelo de la medida basado en ISO 15939 que ha sido propuesto para un mejor apoyo y gestión del proceso integrado de medición.

Para la gestión del proceso de medición, la herramienta importa información organizada en distintos niveles de abstracción de acuerdo a la arquitectura conceptual propuesta y almacenada en un repositorio en forma de documento XMI. Estos elementos de información son:

- **Metamodelos del Dominio,** que representan las entidades software sobre las que se definen los modelos de medición. Por ejemplo, si se requiere la medición de bases de datos relacionales es necesario tener en el repositorio el metamodelo relacional, compuesto por elementos tales como tablas, atributos, claves ajenas, etc., sobre los cuales se definirán modelos de medición compuestos por diferentes métricas, como el número de tablas, número de atributos de una tabla, etc.

- **Modelos del Dominio**. Estos modelos son instancias de los metamodelos del dominio y son sobre los que se aplican los modelos de medición. En el caso de una base de datos relacional, un modelo del dominio sería un esquema concreto de base de datos, como por ejemplo el esquema de una base de datos de una sucursal bancaria. Sobre este esquema se podrían aplicar y calcular las métricas de modelos de medición definidos sobre el metamodelo relacional.
- **Modelos de Medición.** A partir del metamodelo de la medida definido en base al estándar ISO 15939 es posible crear modelos de medida sobre entidades software concretas. Con ello se proporciona un soporte sistemático a la aplicación de un proceso de medición, ya que no sólo se recogen los datos del proceso, sino también los metadatos en forma de modelos.

GenMETRIC se ha tomado como trabajo relacionado porque su filosofía permite a una organización adelantar tareas que contribuyen a la documentación y mejora del proceso en el sentido de que se establecen métricas que evalúan el rendimiento de las actividades inmersas en un proceso. La persistencia la realiza en documentos XMI.

Las diferencias radican en que GenMETRIC está enfocada a las métricas software, dejando por fuera los conceptos restantes de las tecnologías de procesos, como lo son su definición, soporte a la mejora y en general, la documentación del proceso.

3.10.3 MT – ECMA

Este modelo es una base conceptual y funcional para describir y comparar Entornos de Ingeniería de Software (EIS) o componentes de éstos (incluidos componentes de marcos de trabajo). Sus principales objetivos son:

- Servir para describir, comparar y contrastar marcos de trabajo EIS.
- Orientar para la interoperabilidad e integración de herramientas.
- Ser útil para describir marcos de trabajo EIS diversos, buscando un adecuado equilibrio entre generalidad y especificidad.
- Ser independiente de las técnicas de implementación y de los métodos de construcción del software.
- Servir de base para la formación sobre EIS de los ingenieros software.

El modelo de referencia MT-ECMA consiste en un conjunto de servicios que corresponden a las capacidades funcionales de un marco de trabajo EIS describiéndolo en base a servicios. Estos servicios están clasificados en varios grupos para facilitar la comprensión de las actividades a realizar. Por ejemplo, existen varios grupos que tienen que ver con las tres facetas de integración de herramientas: integración de la presentación, integración del control e integración de datos [30].

Su relación con el Framework PDS radica en que su modelo de referencia permite la comprensión de las actividades que los stakeholders deben realizar en un entorno de ingeniería de software, facilitando su trabajo y permitiendo que se cuente con documentación asociada al proceso.

Sin embargo, MT-ECMA esta sujeto a las características propias del entorno de ingeniería del cual depende, sin contar con una metodología propia o una línea base que garantice una buena documentación del proceso. Tampoco proporciona una herramienta para realizar el modelo del proceso y su transformación en archivos que provean portabilidad con diferentes plataformas.

La Tabla 2 presenta un resumen de las características más relevantes de los trabajos relacionados y el FrameWork PDS.

Característica / **Solución**	**LMP**	**Interoperabilidad Interface 1 y XMI**	**Licencia**	**Plataforma**	**Modelo de calidad asociado**
MANTIS	MOF	✓	_	_	ISO
GENMetrics	UML	✓	_	_	ISO 15939
MT-ECMA	-	×	_	_	ISO 15940
FrameWork PDS	SPEM	✓	Open Source	Multi-plataforma	Agile SPI

Tabla 2 - Paralelo de características en herramientas enfocadas al proceso de software

4 FRAMEWORK PDS

Framework PDS es un entorno conceptual y tecnológico basado en la especificación SPEM que facilita a las empresas de software la definición, modelado, visualización, documentación y mejoramiento de sus propios procesos de desarrollo.

Su base de modelado de procesos se soporta sobre la especificación SPEM como metamodelo, dado que esta especificación contiene los elementos necesarios para la definición del ciclo de vida de los procesos y sus componentes, características necesarias para que una organización pueda mantener visibles sus procesos y mas importante aún, pueda actualizarlos y mejorarlos a través del tiempo.

De la misma forma y para facilitar el soporte a la automatización de los procesos adicionalmente, Framework PDS facilita un mecanismo de transformación del lenguaje XMI a XPDL que define la interfaz 1 de la WfMC, proporcionando escalabilidad y ofreciendo la posibilidad de interoperar con otras herramientas y componentes amparados bajo esta especificación. Lo anterior facilita el cumplimiento del objetivo inicial del proyecto propuesto como componente del Modelo Integral para la Mejora de Procesos de Software, AGILE SPI.

Dado que el alcance del Framework PDS no es solo tecnológico sino que proporciona una base de conocimiento centrada en las tecnologías de proceso, permite a las empresas y a su personal, encontrar documentación relacionada y ejemplos que facilitan la interpretación del proceso de desarrollo de software. Esta documentación es gestionada como activos de procesos y las organizaciones podrán crear y gestionar sus propios documentos en cualquier formato, por ejemplo: videos, ejecutables, normas, imágenes, etc, contribuyendo con una documentación exacta y adaptada a las necesidades particulares de cada proceso en una organización.

La definición del proceso viene relacionada muy fuertemente con el proceso de mejora en las empresas, por ello ha sido necesario constituir un modelo de negocios que involucra las áreas funcionales más relevantes de una organización con los equipos de mejora definidos por el modelo IDEAL.

4.1 CONSIDERACIONES DE DISEÑO

Las consideraciones de diseño del Framework PDS se han abordado teniendo en cuenta dos definiciones, a saber: un marco de trabajo es una Infraestructura software que crea un entorno común para integrar aplicaciones e información compartida dentro de un dominio dado [41]; y, Los marcos de trabajo corresponden a estructuras escritas de una idea y/o conjunto de metas para facilitar a una organización la aplicación de las mismas [42]. De esta manera podemos interpretar al Framework PDS como un conjunto de guías, prácticas, estándares y herramientas que se conjugan para definir, contextualizar y visualizar un proceso en un entorno organizacional.

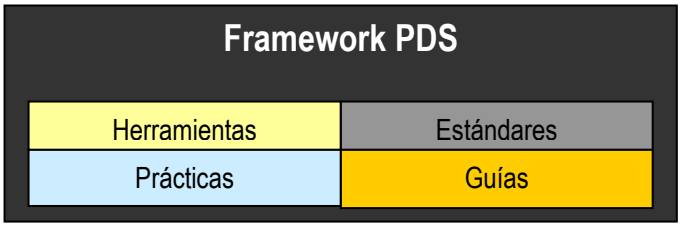


Figura 21 - Framework PDS como marco de trabajo

La finalidad del framework PDS en lo que al proceso respecta es gestionar su evolución, desde la definición hasta la visualización pasando por el modelado y la transformación del proceso.

La definición del proceso tomando como base las especificaciones del Framework PDS se hace mediante una plantilla guía que toma la información relevante a partir de una metodología de entrevistas, realizadas a los stakeholders que hacen parte de la organización, permitiendo identificar claramente el proceso y los detalles que éste puede llevar dentro del contexto real. La información más relevante para iniciar la definición del proceso, corresponde a los límites de inicio y fin del proceso; El flujo del proceso, es decir la secuencia de actividades que se llevan acabo durante su ejecución; la asignación de recursos como personal, tiempo, equipo y demás materiales; y los clientes y proveedores del proceso tanto internos como externos.

El proceso capturado entra a una etapa de estructuración de modo que cumpla con la semántica básica para pasar a la fase de modelado, donde se representa gráficamente el proceso con una herramienta CASE integrada al Framework PDS.

Con el proceso modelado se entra en una fase de transformación a un lenguaje estándar basado en XML que facilite la portabilidad e interoperabilidad del modelo entre diferentes plataformas. Esta consideración se tiene en cuenta para organizaciones que centran su funcionamiento en modelos distribuidos, de modo que no tengan restricciones de migración de modelos a plataformas heterogéneas.

Debido a que uno de los objetivos principales del Framework PDS es garantizar la portabilidad de los modelos a un motor de workflow, se hace necesaria una segunda transformación que garantice este objetivo. Dicha transformación se hace entre documentos XML, a través de un parser y utilizando una serie de reglas de transformación que cumplen con la estructura establecida por el estándar workflow.

La transformación del proceso es un caso alterno de la definición del proceso, pero se expone aquí por requerimiento del Framework PDS. Esta transformación se refiere a la adaptación de un modelo de procesos a otro lenguaje de definición de procesos, mediante unas reglas gramaticales claramente definidas.

Para cumplir con esta funcionalidad, se adaptó una aplicación que permite a partir de la interpretación del modelo de proceso estructurado bajo la especificación SPEM y escrito en XMI, definir un modelo de objetos con el proceso de desarrollo particular y como salida obtener un documento XPDL con la definición de procesos para un workflow. Toma la base de que la incorporación de workflow permite automatizar los flujos de información que circulan a través de toda la industria, posibilitando el secuenciamiento de las actividades desarrolladas y la optimización del uso de los recursos.

Este trabajo propone la optimización del proceso de producción de software, mediante la automatización de las metodologías de desarrollo. Para lograr el objetivo, se siguieron los siguientes pasos:

- Se seleccionó una metodología de desarrollo de software para ser utilizada como caso de estudio de este trabajo. La metodología en cuestión es una instancia de la metodología RUP *(Rational Unified Process)* denominada SmallRUP, definida por Gary Pillice en su artículo "*Using the RUP for small projects*".

- Se utilizó el Metamodelo para la Ingeniería de Procesos de Software SPEM (*Software Process Engineering Metamodel*) definido por la OMG *(Object Management Group)* para el modelado de procesos de desarrollo.
- Una vez especificado el proceso de desarrollo SmallRUP en SPEM, el paso siguiente fue la definición de un conjunto de reglas de transformación para convertir la especificación SPEM en una especificación XPDL (*XML Process Definition Language*). Este último, ya es una definición estándar de un proceso Workflow, por lo que podrá ser ejecutada como cualquier proceso automatizado por un sistema workflow. Las reglas de transformación fueron definidas utilizando el lenguaje XSLT (*XSL Transformations*).

La importancia de ésta iniciativa para el Framework PDS y por la cual fue seleccionada para ser integrada como parte de éste, es porque permite instanciar manualmente el modelo SPEM en un modelo de objetos que posteriormente es transformado en un documento XMI. En seguida, por medio de un parser integrado en ésta herramienta, se obtiene la representación del documento XMI en XPDL.

Cada uno de los productos obtenidos, es decir, la guía de definición, el modelo y los documentos XML del proceso, deben ser accesibles y visualizables en el tiempo por los participantes de la organización, razón por la cual se hizo necesario involucrar un mecanismo de soporte a un repositorio de procesos, sin embargo esto no es suficiente para quedar completamente disponible a los usuarios finales. Para completar la solución del Framework PDS, se hizo necesario implementar una utilidad que describiera literalmente los procesos facilitando la entrada de información relevante. Así mismo y para una mejor interpretación se estableció vincular a los procesos, los activos que los complementan, como guías, diagramas, plantillas, scripts, porciones de código y demás.

4.2 ARQUITECTURA FRAMEWORK PDS

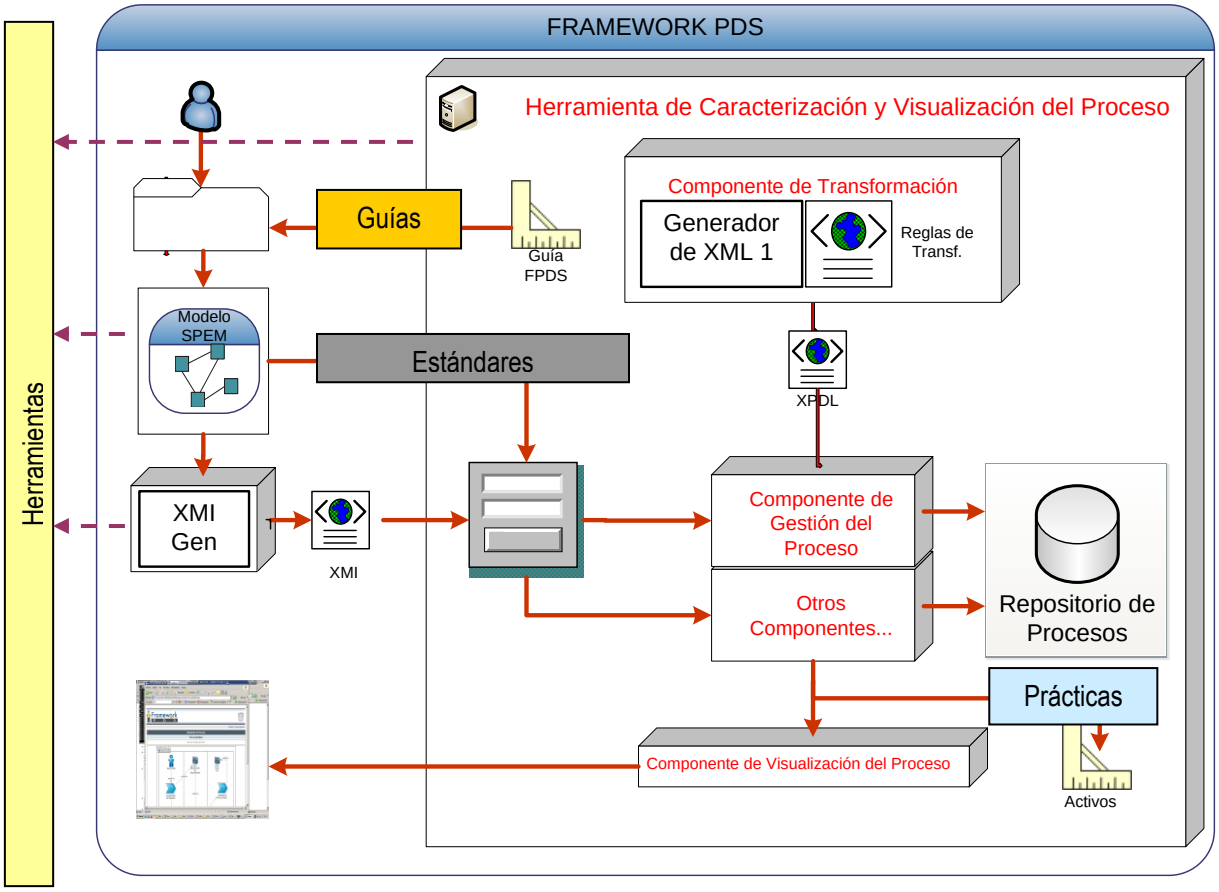


Figura 22 - Arquitectura conceptual del Framework PDS

4.3 DETALLES DE IMPLEMENTACIÓN

Los detalles de implementación del Framework PDS son variados en cada una de sus etapas. Para lograr un mayor grado de comprensión de las tecnologías empleadas en la construcción de esta solución iniciaremos tomando cada una de las etapas de la evolución del proceso establecida para el Framework PDS.

Retomando un poco las consideraciones de diseño, podemos afirmar que la definición del proceso involucra crearlo en el repositorio de la aplicación. Este procedimiento es llevado a cabo por el Rol "Ingeniero de Procesos", (que es una persona que tiene la preparación

idónea para llevar a cabo esta labor), con la ayuda de los stakeholders involucrados en el proceso a definir.

El ingeniero de procesos, debe estructurar el proceso de modo que pueda ser modelado en la herramienta para tal fin. La herramienta definida debe estar soportada sobre el metamodelo SPEM especificado por la OMG. Este metamodelo fue escogido por el Framework PDS porque proporciona las bases necesarias para que el modelado de procesos se interprete de manera autónoma con un mayor grado de detalle. Al extender de MOF facilita el intercambio entre herramientas basadas en UML y repositorios MOF. Sin embargo y muy a pesar de las potencialidades que ofrece SPEM no ha sido posible lograr una exploración a fondo de las herramientas basadas en esta especificación, primero porque la cantidad de soluciones en el mercado es reducida; para rescatar, Enterprise Architect de Sparx System[10], con la aclaración, que esta herramienta incorpora SPEM como perfil UML y no como metamodelo. Soluciones como IRIS Suite[11] de la empresa Osellus aun cuando tiene una herramienta basada en el metamodelo, no proporciona ni siquiera versiones de evaluación para su exploración. Rational Process Workbench[12] es un acercamiento al metamodelo SPEM solo que el modelado de sus procesos obedece a instancias de RUP. Existen proyectos que a futuro se ven muy alentadores como el que adelanta IBM con Eclipse Process Framework Project (EPF)[13] que es una iniciativa para crear un framework de procesos de código abierto que ayude a las compañías a establecer consistencia en la planeación y ejecución de proyectos de desarrollo de software. Vale la pena aclarar que según el concepto que maneja es algo muy similar a lo que propone Framework PDS, solo que EPF está soportado por la comunidad de software libre.

La evolución del proceso, cuando se tiene el modelo SPEM, debe ser exportado a un tipo de documento XMI, para facilitar que las organizaciones puedan migrar fácilmente sus modelos entre diferentes plataformas.

[10] http://www.sparxsystems.com.au/ea.htm
[11] http://www.osellus.com/products/irispas.html
[12] http://www.ibm.com/software/awdtools/rup/workbench
[13] http://www.eclipse.org/epf/

El componente XMI Generator es una solución standalone basada en Java que consta de dos paquetes básicos. El modelo de objetos, hereda de la clase Model la cual es una representación de las clases SPEM y un mecanismo de producción de documentos XMI – XPDL que toma el modelo de objetos dispuesto en la clase java y lo transforma a XMI para luego pasar por un parser XML que transforma los documentos XMI a la especificación XPDL. Figura 23 muestra gráficamente esta solución.

La construcción del modelo de objetos se podría omitir si se tiene a mano una herramienta case que permita modelar bajo la especificación del metamodelo SPEM, esta bien podría ser IRIS Suite, para tomar un ejemplo, pero comercialmente no tiene una versión trial de prueba.

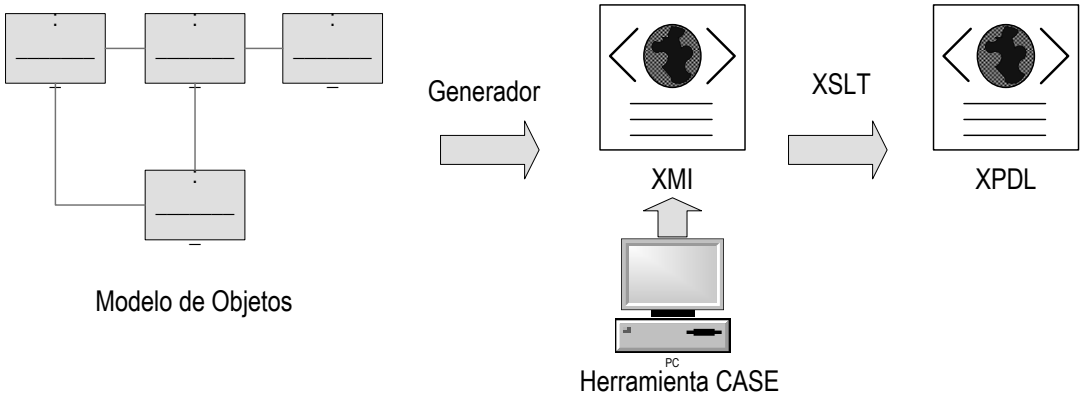


Figura 23 - Producción de XMI – XPDL a partir del modelo de objetos

El modelo de objetos es entonces una instanciación de las clases SPEM básicas representadas en la solución XMI Generator, a continuación se muestra una porción del modelo que representa la instanciación de fases y roles del proceso SIDEM LTDA.

```
// Las tres fases del proceso SIDEM
  private Instance Faseinicio =
      new Instance (SPEM_PHASE, new String[][]{{"name","Inicio"}});
  private Instance FaseElaboracion =
      new Instance (SPEM_PHASE, new String[][]{{"name","Elaboracion"}});
  private Instance FaseConstruccion =
      new Instance (SPEM_PHASE, new String[][]{{"name","Implantación"}});

// roles
  private Instance Desarrollador =
      new Instance(SPEM_PROCESS_ROLE, new String[][]{{"name","Desarrollador"}});
```

Figura 24 – Modelo de Objetos de las Fases del proceso SIDEM LTDA

Con el modelo de objetos construido, el paso a seguir es la generación del documento XMI y luego la transformación a documento XPDL, este proceso lo hace la solución XMI Generator de forma automática.

Ahora bien, el proceso ha quedado definido a través de las diferentes etapas de su evolución. Solo queda por establecer la visualización y caracterización del proceso que se lleva a cabo mediante la herramienta Web del Framework PDS.

Para la construcción del Framework PDS, se utilizó la tecnología J2EE (Java 2 Enterprice Edition) dado que es una aplicación WEB. J2EE es la edición empresarial del paquete Java creada y distribuida desde 1997 por Sum Microsystems. Comprende un conjunto de especificaciones y funcionalidades orientadas al desarrollo de aplicaciones empresariales. Esta plataforma ofrece mejores perspectivas de desarrollo para quienes quieran basar su arquitectura en productos fundados en software libre.

La plataforma J2EE define un modelo de programación encaminado a la creación de aplicaciones basadas en n-capas. Típicamente, una aplicación puede tener 5 capas diferentes:

- Capa de cliente: Representa el interfaz de usuario que maneja el cliente.
- Capa de presentación: Representa el conjunto de componentes que generan el la información que se representará en el interfaz de usuario del cliente. Típicamente se creará a través de componentes basados en Servlets y JSP.
- Capa de lógica de negocio: Contiene nuestros componentes de negocio reutilizables. Normalmente se forma a partir de componentes EJB.
- Capa de integración: Aquí se encuentran componentes que permiten hacer más transparente el acceso a la capa de sistemas de información. Por ejemplo este es el lugar idóneo para implementar una lógica de objetos de acceso a datos.
- Capa de sistemas de información: Esta capa engloba los sistemas de información: bases de datos relacionales, bases de datos orientadas a objetos, etc.

Las ventajas en un modelo como este son muy importantes. Al tener capas separadas, existe poco acoplamiento entre las mismas, de modo que es muy más fácil realizar modificaciones en ellas sin que interfieran en las demás, asegurando la inclusión de patrones de diseño y todas las ventajas que ello conlleva. Todo esto se ve reflejado en

las mejoras en cuanto a la mantenibilidad, extensibilidad y reutilización de componentes. Se promueve la heterogeneidad de los clientes, ya que la inclusión de nuevos clientes (móviles, PC's, etc) se reduce a añadir nuevas capas de interfaz de usuario y presentación, sin tener que modificar el resto de capas.

Los servidores de aplicaciones son el corazón de la plataforma J2EE. En ellos residen todos los componentes, ya sean objetos distribuidos accesibles remotamente, páginas web, o incluso aplicaciones completas accesibles utilizando Java Web Start. Así, en un servidor de aplicaciones se cuenta con un contenedor de servlets, un contenedor de EJBs, sistemas de mensajería, y un gran número de herramientas que ayudan a incrementar la productividad en un equipo de desarrollo.

El servidor de aplicaciones utilizado en el desarrollo de este proyecto fue el JBoss 4.0.1, dado que es el servidor de aplicaciones libre por excelencia, esta implementado 100% en JAVA y desde su versión 3.0 soporta el estándar J2EE. Entre sus características se destacan el soporte de EJB, la posibilidad de crear clusters y su descarga se puede realizar en versiones que integran Apache Tomcat o Jetty como servidores web. Para el propósito de este trabajo, el servidor Web utilizado fue Apache Tomcat.

Sin duda, la pieza más importante dentro de un servidor de aplicaciones es el contenedor de EJBs. Los EJBs son objetos reutilizables que contienen la lógica de negocio de nuestro sistema. Los contenedores de EJBs son servidores que se encargan de controlar todos estos componentes, esto es muy importante ya que se automatizan tareas como la gestión del ciclo de vida, la gestión de las transacciones, la gestión de persistencia, etc, permitiendo la utilización de patrones de diseño, con esto, los desarrolladores, no tienen que preocuparse en crear servicios de bajo nivel y pueden centrarse en la creación de lógica de negocio empresarial.

J2EE soporta el framework JSF (Java Server Faces) que es el API Java estándar para construir interfaces de usuario en aplicaciones Web. Se enfoca en la capa de la Vista de la arquitectura Modelo-Vista-Controlador, aunque proporciona suficiente capacidad de control para escribir aplicaciones moderadamente complejas utilizando un único API, La Figura 25 muestra una vista general del Framework JSF.

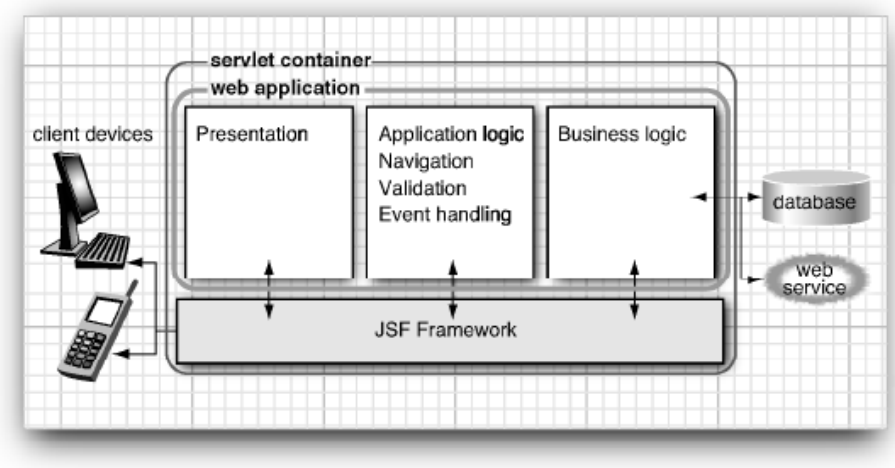


Figura 25 - Una vista del Alto nivel del Framework JSF

Para la persistencia de los datos, se utilizó PostgreSQL, que es un sistema de bases de datos Objeto-Relacionales (ORDBMS). Posee muchas características técnicas que tradicionalmente sólo se podían disfrutar en la utilización de productos comerciales de alto reconocimiento; características de las cuales se puede resaltar: Cumple con ANSI SQL, Integridad referencial, Replicación (soluciones comerciales y no comerciales) que permiten la duplicación de bases de datos maestras en múltiples sitios de replica, Interfaces nativas para ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python y Ruby, Reglas, Vistas, Triggers, Unicode, Secuencias, Herencia, Outer Joins, Sub-selects, Una API abierta, Procedimientos almacenados, Soporte nativo SSL, Lenguajes procedurales, Respaldo en caliente, entre otras.

A excepción de la solución Web del Framework PDS, las herramientas que se utilizan han sido integradas como componentes externos, por esta razón, a continuación se hace una breve descripción del proceso de desarrollo de la solución Web, encargada de la visualización y caracterización del proceso, sin embargo y para efectos de la presente monografía solo se explica el caso de uso más relevante que es la gestión del proceso.

4.3.1 Proceso de desarrollo de la solución Web del Framework PDS

4.3.1.1 Modelo de Casos de Uso Simplificado

Este caso de uso contiene las funcionalidades generales que ofrece el Framework PDS y es el primer paso para identificar los actores que intervienen en la definición de los

procesos en una organización. La Figura 26 contiene los casos de uso generales del Framework PDS.

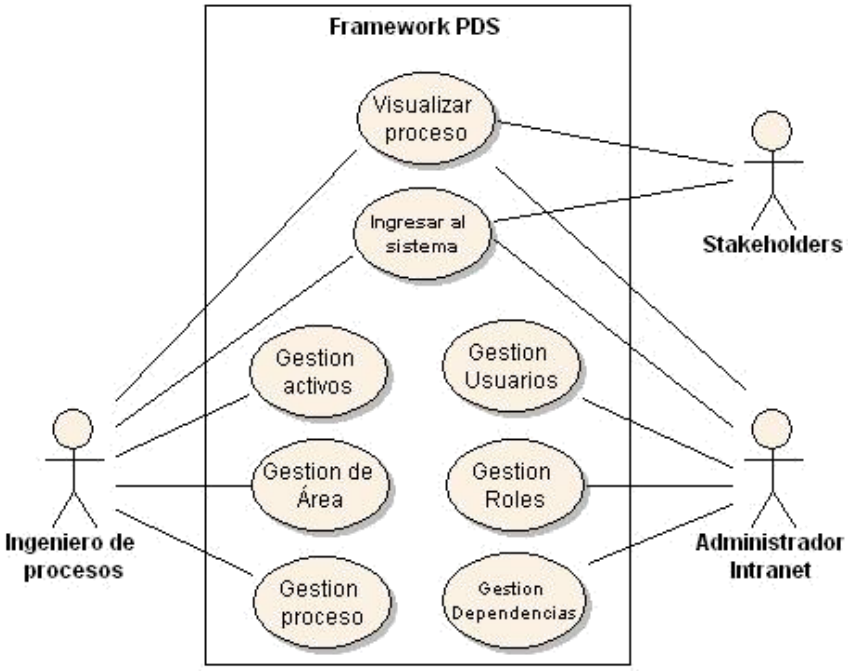


Figura 26 – Diagrama General de Casos de Uso

El actor Ingeniero de procesos es la persona especializada en las tecnologías de procesos. Su misión fundamental es la de diseñar, seleccionar los activos y la documentación relacionada con los procesos de la organización. Pone en marcha y ejecuta todo lo necesario para obtener un óptimo funcionamiento de los sistemas o procesos y en general, es el encargado de mantener el programa de mejora en una compañía.

El Administrador Intranet es el encargado de la administración general del Framework PDS, su función es la de gestionar la información de los usuarios, sus roles y las dependencias en la que esta organizada una empresa. Al igual que los demás actores que intervienen o interactúan con el sistema, este actor puede visualizar los procesos almacenados en el Framework PDS.

El Stakeholders cuenta con la funcionalidad de visualizar el proceso, condescendiéndole el acceso a los recursos asociados a éste, con la diferencia que no los puede modificar.

Es importante realizar una descripción de los casos de uso para comprenderlos mejor:

Nombre: **Ingresar al sistema**

Actores: Administrador Intranet, Ingeniero de Proceso, Stakeholders

Propósito: Permitir el ingreso a las funcionalidades del sistema.

Descripción: El Framework PDS contiene divisiones funcionales: una visualización general para cualquier usuario del sistema, que es representada por el caso de uso Visualizar proceso, la administración de la información relacionada al proceso y la administración de la información relacionada a los usuarios. Para contar con una buena seguridad a nivel de usuario, es necesario restringir el ingreso solo al personal autorizado a cada módulo, funcionalidad que se consigue con éste caso de uso.

Tipo: Primario

Nombre: **Visualizar Proceso**

Actores: Administrador Intranet, Ingeniero de procesos, Stakeholders

Propósito: Permitir la visualización de los procesos existentes en el sistema

Descripción: Todos los usuarios registrados están facultados para visualizar los procesos del Framework PDS, con el fin que sean de fácil acceso para su rápida utilización en las actividades cotidianas de la compañía.

Tipo: Primario

Nombre: **Gestión activos**

Actores: Ingeniero de Procesos

Propósito: Administrar la información de los activos de procesos de la organización en el repositorio de datos

Descripción: Los activos de procesos cumplen un papel muy importante en la documentación del proceso; por esta razón, es necesario gestionar toda la información relacionada con estos, que consiste en la creación, eliminación y actualización.

Tipo: Primario

Nombre: **Gestión Área**

Actores: Ingeniero de procesos

Propósito: Administrar la información relacionada con las áreas que componen la organización.

Descripción: Las compañías, independientemente sea su razón social, tienen en su estructura organizacional áreas que colaboran con la realización de las

actividades de la misma. Estas áreas no son las mismas en todas las organizaciones e incluso, son diferentes dependiendo del modelo de calidad adoptado por la organización. Por esta razón, se ha realizado éste caso de uso. El Ingeniero de procesos es el encargado de administrar la información relacionada con las áreas de la empresa, la administración o gestión consta de creación, modificación y eliminación de la información.

Tipo: Primario

Nombre: **Gestión Proceso**

Actores: Ingeniero de procesos

Propósito: Administrar toda la información perteneciente a un proceso

Descripción: Este caso de uso representa la principal funcionalidad del Framework PDS. La administración de los procesos consta de crear, modificar y eliminar procesos, así como la asociación de activos que hacen parte de la documentación del proceso, facilitando su entendimiento además de la importación del modelo, que permite importar el documento XMI, el documento XPDL y la gráfica del modelo del proceso.

Tipo: Primario

Nombre: **Gestión de usuarios**

Actores: Administrador Intranet

Propósito: Administrar la información de los usuarios del sistema

Descripción: El Administrador de la Intranet es el encargado de administrar la información relacionada con los usuarios del Framework PDS. Tal gestión consiste en la creación, modificación y eliminación de la información de usuarios.

Tipo: Primario

Nombre: **Gestión de roles**

Actores: Administrador Intranet

Propósito: Administrar la información de los roles de usuario existentes en una compañía

Descripción: En cualquier compañía, los empleados manejan diferentes roles en un determinado proceso, máxime si se trata en particular de un proceso de desarrollo de software. Se hace necesario entonces administrar la

información de dichos roles para que la organización pueda variar dependiendo las tendencias o exigencias de la tecnología y el mercado. Esta administración consiste en la creación, modificación y eliminación de roles.

Tipo: Primario

Nombre: **Gestión de dependencias**

Actores: Administrador Intranet

Propósito: Administrar la información de las dependencias que conforman una empresa.

Descripción: El Administrador de la Intranet es el encargado de administrar la información relacionada con las dependencias de una organización. Tal gestión consiste en la creación, modificación y eliminación de la información de las dependencias.

Tipo: Primario

4.3.1.2 Arquitectura preliminar

Se hace necesario que todos los conceptos de la solución conceptual y metodológica que provee el Framework PDS estén disponibles en una solución tecnológica que permita involucrar estos conceptos para finalizar con la visualización y seguimiento de los procesos definidos.

Esta solución está conformada por dos componentes: el primero corresponde a una herramienta SPEM que nos facilite el modelado de los procesos y su representación en formato XMI y formato XPDL. El segundo componente corresponde a una aplicación Web que toma la definición del proceso hecha en la herramienta SPEM, y la envía a un repositorio de modelos; estos modelos son alimentados con información adicional como la descripción y el nombre del proceso, usuarios, roles y dependencias dentro de la empresa. Con esta información se elabora un árbol de seguimiento de procesos a los cuales también se les relacionan los activos necesarios como plantillas, documentos, modelos, guías y demás que ayudan a mejorar la comprensión y seguimiento de un proceso dado.

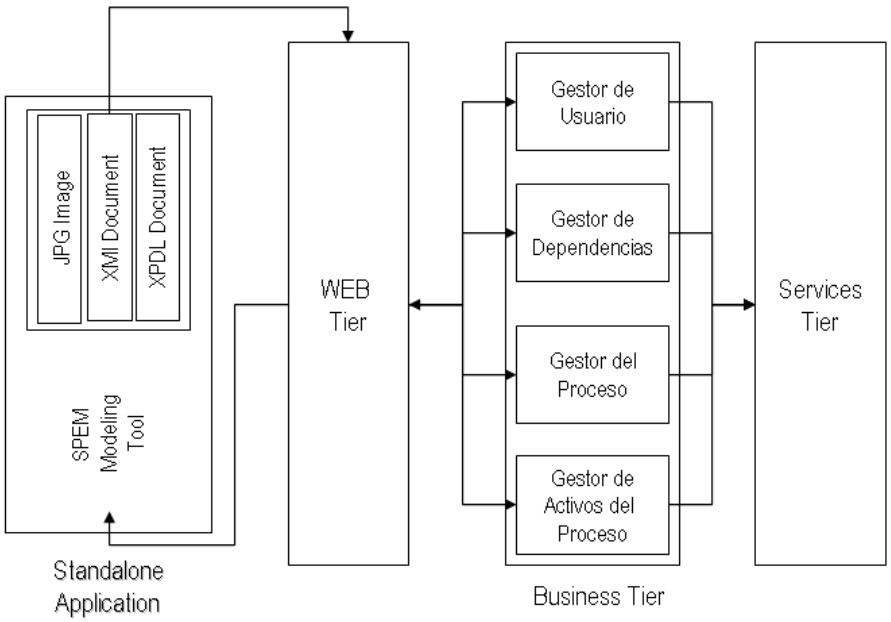


Figura 27 - Arquitectura general Framework PDS

La Figura 27, muestra la arquitectura general del Framework PDS, donde se puede evidenciar la existencia de la herramienta SPEM integrada, así como cada uno de los gestores propios de la solución. Estos son:

- Gestor de Usuarios: Permite administrar el acceso de los usuarios al Framework-PDS dependiendo del Rol que desempeñen dentro de la empresa.
- Gestor de Dependencias: Facilita la administración de las dependencias dentro de la empresa de modo que se puedan segmentar por áreas de operación (Administrativa, de Producción, Otras).
- Modelado de Procesos (Componente externo): Módulo standalone para integrar al Framework PDS que permite el modelado de un proceso basado en el metamodelo SPEM de modo que se pueda obtener de él, el proceso en formato XMI, el proceso en formato XPDL y la representación gráfica del mismo.
- Gestor del Proceso: Corresponde al módulo mas relevante del Framework PDS, pues es el que permite tomar los procesos definidos y organizarlos de modo que puedan ser visibles para los stakeholders de la empresa. Maneja dos fases, la primera corresponde a la importación del proceso, que facilita la obtención desde el sistema de archivos los documentos XMI, XPDL y la representación gráfica del proceso para vincularlos al Framework PDS. La segunda fase tiene que ver con la alimentación del proceso con información relevante para los usuarios finales como

el nombre del proceso, la descripción y como complemento, la asociación de activos del proceso.

- Gestor de Activos del Proceso: Módulo que permite administrar los activos asociados a la empresa, como plantillas, modelos, guías y demás artefactos de apoyo que faciliten la comprensión y seguimiento de un proceso. Los activos del proceso se hacen teniendo en cuenta el área o dependencia de la empresa.

4.3.1.2.1 Integración de componentes

En la Figura 27 se puede observar el módulo Standalone Application, el cual contiene los sub-módulos que proveen la funcionalidad para el modelado y la definición del proceso, estos son:

- SPEM Modeling Tool (XMI Generator): es una herramienta que permite realizar el modelado de un proceso a partir del metamodelo SPEM.
- JPG Image: Este módulo permite asociar al proceso, su respectivo modelo en un archivo de imagen, para una mejor comprensión por parte de los stakeholders participantes del proceso. Esta imagen es generada por el módulo XMI Generador.
- XMI Document y XPDL Document: el documento XMI (XMI Document) es generado por el sub-módulo XMI Generator a partir de un caso de negocio y su función es la de permitir el intercambio del modelo de procesos entre diferentes plataformas de modelado basadas en UML, facilitando la utilización de modelos y otros metadatos como flujos o archivos vía Internet. El documento XPDL (XPDL Document) es creado a partir del Documento XMI siguiendo las reglas de transformación específicas, con el fin de cumplir con la Interfaz 1 de la WfMC, admitiendo la portabilidad a workflows para la automatización del proceso.

A partir del módulo de Gestor de procesos, presente en la solución Web para la visualización de los mismos, se integra cada uno de los activos resultado de la instanciación del XMI Generator.

4.3.1.3 Diagrama de Casos de Uso Extendidos

El objetivo de estos casos de uso es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que contribuya a

estructurar el sistema entero. Para este caso particular, la Figura 28 contiene el diagrama extendido del caso de uso Gestión proceso y de los demás casos de uso pertenecientes a las funcionalidades del ingeniero de procesos, como ya se mencionó anteriormente, solo se describirán los casos de uso pertenecientes a la Gestión de procesos.

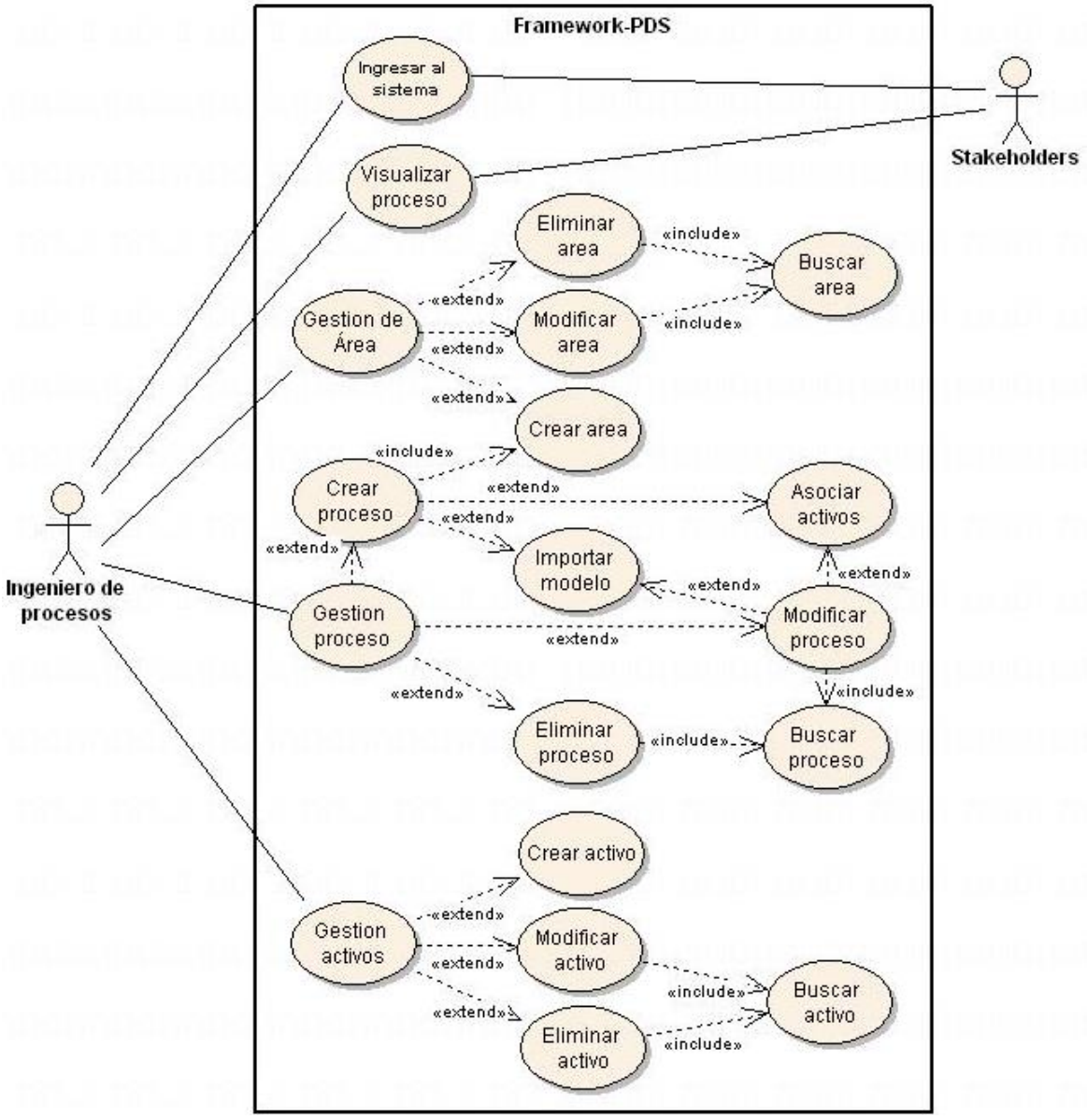


Figura 28 – Casos de Uso extendidos

Los casos de uso que conforman la Gestión proceso de la figura anterior, se describen a continuación:

Los casos de uso Ingresar al Sistema y Gestión proceso, ya se describieron anteriormente; a continuación, se detalla los casos de uso reales y el curso normal de los eventos respectivamente:

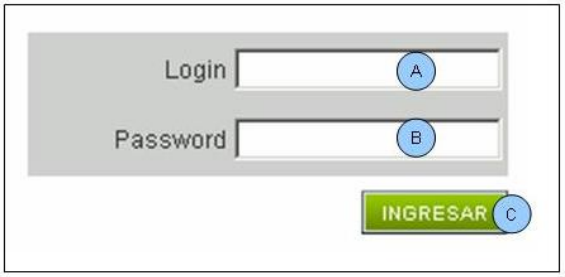


Figura 29 – Caso de uso real: Ingresar al sistema

Acción de los actores		*Respuesta del sistema*
1. Este caso de uso inicia cuando el usuario (Administrador de la Intranet, Ingeniero de procesos o stakeholders) desea ingresar al sistema.		*2.* Muestra el menú inicial
3. Selecciona la opción de ingresar al sistema		*4.* Solicita Login y password (Nombre de usuario y clave)
5. Ingresa su login en el campo A, el password en el campo B y pulsa el botón C para enviar la solicitud.		*6.* Comprueba existencia de usuario
		7. Despliega las opciones adecuadas para cada tipo de usuario en una nueva página.

Figura 30 - Caso de uso real: Gestión proceso

Acción de los actores		*Respuesta del sistema*
1. Este caso de uso inicia cuando el ingeniero de procesos desea administra la información de los procesos documentados en el Framework PDS.		
2. Ingresa al sistema (caso de uso anterior)		*3.* Despliega las opciones del ingeniero de procesos.
4. Selecciona la opción de Gestión de Procesos que la llama el link A.		*5.* Muestra las opciones comprendidas en la Gestión de procesos, listando todos los procesos por área, panel E. El botón D llama la página de nuevo proceso. Los literales F y G muestran información relativa al proceso (Nombre y área) y los botones H e I modifican o eliminan el respectivo proceso.

Los casos de uso restantes se describen a continuación:

Nombre: **Crear Proceso**

Actores: Ingeniero de procesos

Propósito: Crear un nuevo proceso, capturando la información relevante y admitiendo que el ingeniero de procesos asocie archivos que posteriormente facilitarán la compresión del proceso por parte de los stakeholders que intervienen en su ciclo de vida.

Descripción: El ingeniero de procesos es el encargado dentro de la compañía de la identificación de los procesos, su modelado y mantenimiento a través del tiempo. Cuando identifica un proceso, debe crearlo en el sistema, ingresando la información solicitada y asociándole los documentos que considere pertinentes para su mejor interpretación por parte de los demás participantes o stakeholders.

Tipo: Primario

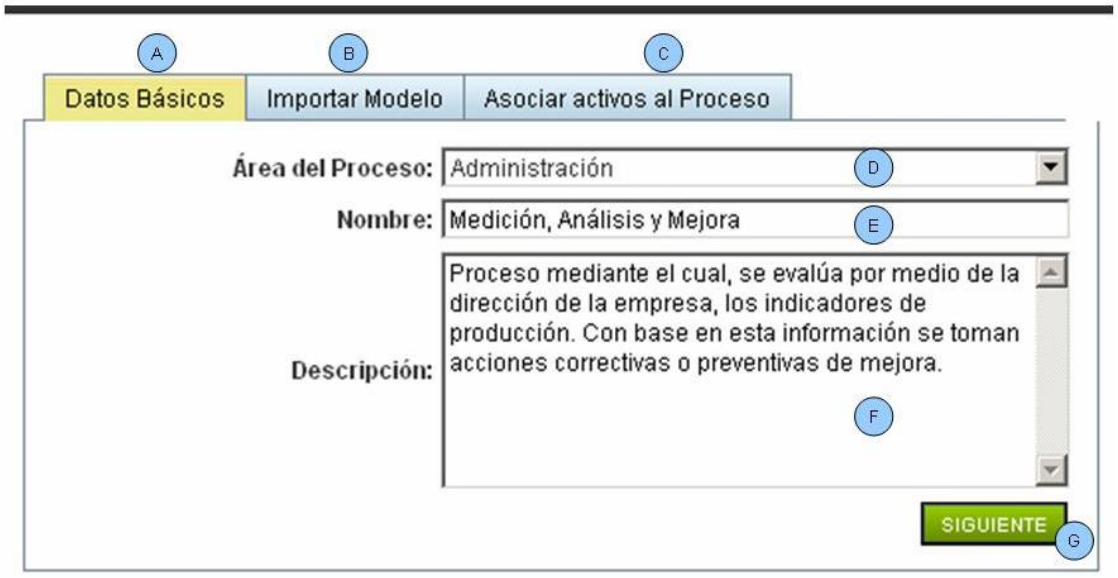


Figura 31 - Caso de uso real: Nuevo proceso

Curso normal de los eventos:

Acción de los actores		*Respuesta del sistema*
1. Inicia cuando el ingeniero de procesos desea crear un nuevo proceso.		
2. Ingresa al menú de opciones de Gestión de procesos y selecciona la opción de crear un nuevo proceso (caso de uso anterior).		*3.* Por defecto, muestra la pagina de nuevo proceso en la pestaña A, aquí solicita la información básica del nuevo proceso, como lo es nombre en

		la caja de texto E, descripción en la caja de texto F y el área donde pertenece, que se selecciona en la lista de selección D. Si en la lista de áreas, no se encuentra la deseada, la lista de selección D tiene una opción de crear un área, por lo que pasaría al caso de uso Crear área
4. Ingresa la información solicitada por el sistema y pulsa el botón G.		*5.* Realiza validación de datos a nivel de usuario
		6. Almacena el nuevo proceso en el sistema y pasa a la pestaña B para que el usuario asocie los modelos al proceso.

Al momento en el que un ingeniero de procesos crea un nuevo proceso, el sistema debe facilitar la asociación de los archivos que permitirán la documentación del mismo; esta funcionalidad la describen los casos de uso Asociar activos e Importar modelo que se describen a continuación:

Nombre: **Importar modelo**

Actores: Ingeniero de procesos

Propósito: Importar el modelo del proceso en diferentes formatos y asociarlos a los registros.

Descripción: La importación del modelo debe tener las funcionalidades de asociar tres clases de archivos diferentes:

- Modelo en XMI
- Modelo en XPDL (Para cumplir con la interfaz 1 de la WfMC)
- Grafica del Modelo en el perfil SPEM

Tipo: Primario

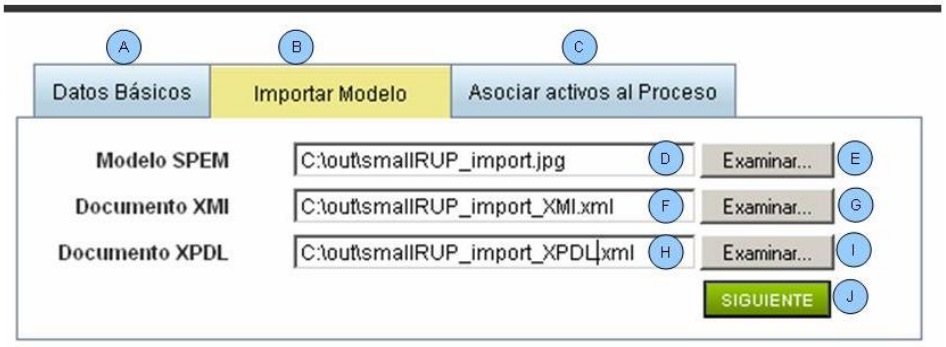


Figura 32 - Caso de uso real: Importar modelo

Curso normal de los eventos:

Acción de los actores		*Respuesta del sistema*
1. El ingeniero de procesos selecciona la opción de importar modelo en la pestaña B de la página de nuevo proceso.		*2.* Despliega el formulario de asociación de modelos
3. Asocia los modelos al proceso desde los botones E, G e I, los cuales despliegan una ventana de búsqueda de archivos y colocan la dirección de los mismos en las cajas de texto D, F y H. Por último, envía el formulario desde el botón J.		*4.* Verifica la información y realiza el almacenamiento de los archivos en el repositorio de datos
		5. Informa el éxito de la operación

Nombre: **Asociar activos**

Actores: Ingeniero de procesos

Propósito: Documentar el proceso con archivos creados para tal fin.

Descripción: La finalidad de la asociación de los activos es documentar lo mejor posible los procesos que se van definiendo en una empresa. Estos activos pueden ser plantillas, documentos, normas, etc; que con su utilización, facilitan la comprensión del proceso. El ingeniero de procesos tiene la facultad de asociarlos a los procesos si éstos lo requieren.

Tipo: Secundario

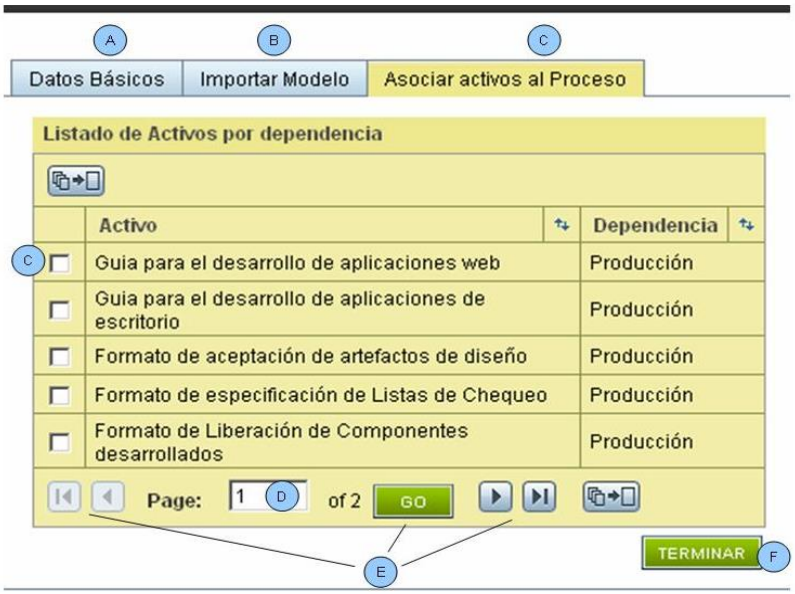


Figura 33 - Caso de uso real: Asociar activos

Curso normal de los eventos:

Acción de los actores		*Respuesta del sistema*
1. Inicia cuando se selecciona la pestaña C de asociar activos al proceso.		*2.* Despliega el formulario de asociación de activos
3. Asocia los activos seleccionando los campos de chequeo C. El campo D señala la pagina de activos en la que esta el usuario, si desea cambiar la pagina de los activos, puede hacerlo con los botones de navegación E. Cuando el usuario a terminado de seleccionar los activos que va a asociar, envía el formulario desde el botón F.		*4.* Verifica la información y realiza el almacenamiento de los archivos en el repositorio de datos
		5. Informa el éxito de la operación y devuelve al ingeniero de procesos al menú principal.

El curso normal de los eventos de los casos de uso siguientes: Modificar y Eliminar proceso, se basa en la Figura 30.

Nombre: **Modificar proceso**

Actores: Ingeniero de procesos

Propósito: Modificar los procesos registrados en el sistema

Descripción: Los mercados y las tecnologías cambian constantemente y es necesario que las organizaciones evolucionen al mismo ritmo. Para lograrlo, deben implementar políticas que aseguren que sus procesos cambien o mejoren. La funcionalidad que representa este caso de uso permite modificar los procesos según sea necesario.

La modificación en ocasiones puede necesitar la asociación de activos o importar nuevamente los modelos, por esta razón, este caso de uso extiende los dos anteriores (Importar modelo y Asociar activos).

Tipo: Secundario

Curso normal de los eventos:

Acción de los actores		*Respuesta del sistema*
1. Inicia cuando al ingeniero de procesos necesita actualizar la información de un proceso.		
2. Selecciona la pestaña A de gestionar proceso (Figura 30).		*3.* Carga los procesos existentes en el Framework PDS
4. Da clic en el link H del proceso que se desea modificar.		*5.* Busca el proceso en los registros del framework PDS y lo presenta en forma editable en una ventana igual a la Figura 31 para que el usuario ingrese los cambios que desee.
6. Modifica la información, de manera similar al caso de uso Nuevo Proceso		*7.* Guarda la nueva información e informa el éxito de la operación.

Nombre: **Eliminar Proceso**

Actores: Ingeniero de procesos

Propósito: Eliminar un proceso registrado en el sistema

Descripción: El Ingeniero de procesos decide eliminar un proceso del sistema, para lo cual, debe inicialmente buscarlo en los registros. Si el proceso es encontrado, será eliminado.

Tipo: Secundario

Curso normal de los eventos

Acción de los actores		*Respuesta del sistema*
1. Inicia cuando al ingeniero de procesos desea eliminar un proceso.		
2. En la Gestión de procesos (Figura 30), elige el link I del proceso a eliminar.		*3.* Muestra una ventana de confirmación de eliminación.
4. Confirma la eliminación.		*5.* Realiza la eliminación e informa de la eliminación.

Nombre: **Buscar proceso**

Actores: Ingeniero de procesos

Propósito: Buscar un área en el sistema

Descripción: Este caso de uso es necesario para cumplir con los casos de uso de Modificar y Eliminar proceso, ya que lo primero que se debe hacer es la búsqueda del registro para tener la información precisa que se va a modificar o eliminar. Este caso de uso no presenta interfaz de usuario.

Tipo: Secundario

4.3.1.4 Diagrama de Clases

La definición de la arquitectura del Framework PDS muestra una estructura multicapa con capacidad de distribuir cada uno de sus componentes (capa de presentación, componentes EJB y persistencia). La utilización de algunos patrones contribuye de manera significativa a que esta arquitectura sea consistente.

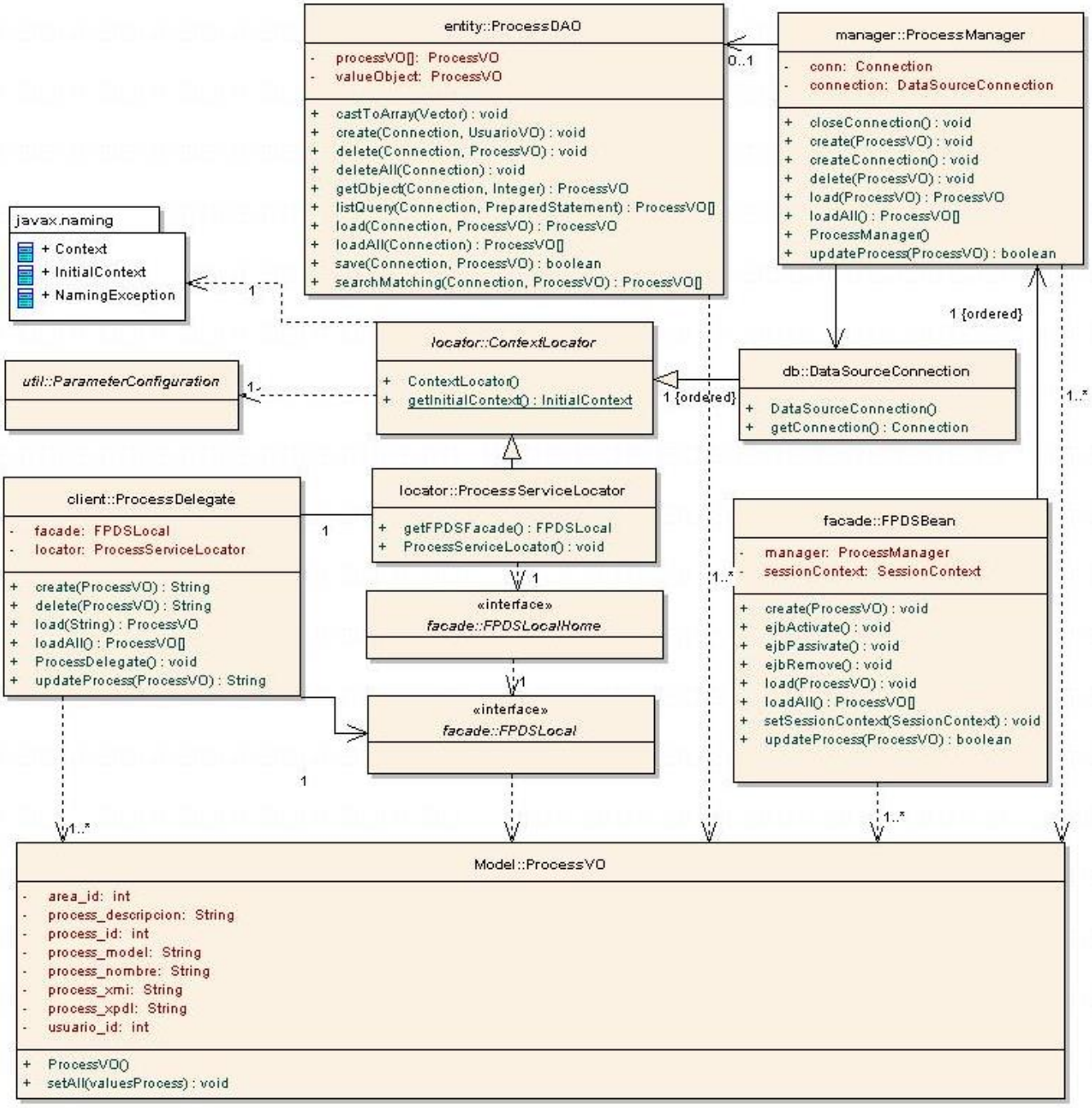


Figura 34 - Diagrama de Clases – Gestión del Proceso

El diagrama de clases de la Figura 34 expone tangiblemente los patrones que a continuación se describen:

Según [40], las mejores prácticas que se pueden utilizar en el desarrollo de aplicaciones J2EE, son las siguientes:

4.3.1.4.1 Patrón Value Object

Su contexto esta enmarcado cuando la aplicación cliente necesita intercambiar datos con algún Enterprise Java Bean (EJB).

El Problema: Las aplicaciones J2EE implementan componentes de negocios del lado de servidor como beans de sesión (session beans) y beans de entidad (entity beans). Los beans de sesión representan los servicios de la lógica de negocios y permiten mantener una relación uno a uno con el cliente. Los beans de entidad, por su parte, son multiusuarios y representan la persistencia de los datos.

Un bean de sesión provee métodos de servicio de grano grueso cuando es implementado por medio del patrón Session Facade. Algunos de estos métodos pueden retornar datos al cliente que realiza alguna invocación a los mismos.

Un bean de entidad implementa los componentes de negocio persistente. Permite exponer los valores de sus atributos mediante los métodos de acceso (métodos get) para cada atributo. Cuando un cliente necesita obtener múltiples valores de un entity bean, debe invocar a los métodos get respectivos hasta obtener la data correspondiente a cada atributo que requiera.

La Solución: Utilizar un Value Object para encapsular los datos del negocio. El llamada a un método sencillo puede ser utilizado para enviar y recibir el Value Object. Cuando el cliente le solicita al Enterprise Bean los datos de negocio, el Bean puede construir el Value Object, rellenarlo con los valores de los atributos, y retornarlo al cliente.

Los clientes por lo general requieren más de un valor de un enterprise bean. Para reducir el número de llamadas remotas y evitar por consiguiente la sobrecarga asociada, resulta más conveniente utilizar el Value Object para transportar la data desde el enterprise bean hacia el cliente.

Cuando el enterprise bean utiliza un value object, el cliente realiza una sola invocación al bean obteniendo como resultado el value object en lugar de realizar múltiples llamadas para obtener cada uno de los atributos. Luego el bean instancia un value object y copia los valores de sus atributos en la instancia del objeto que acaba de crear. Finalmente retorna el value object al cliente.

La Figura 35 muestra el diagrama de clases que representa al patrón Value Object:

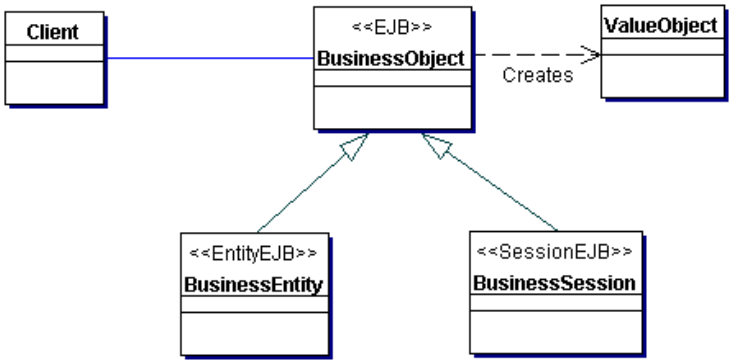


Figura 35 - Diagrama de clases para el patrón Value Object

Consecuencias: Simplifica al Entity Bean y a su interfaz remota, reduce el tráfico de la red, propagación de la actualización, sincronización y control de versiones y la reducción de la duplicación de código

4.3.1.4.2 Patrón Business Delegate

Su contexto es el sistema cuando expone sus servicios de negocio por medio de un API a sus clientes, a través de una red.

El problema: Los componentes de la capa de presentación interactúan directamente con los servicios de la lógica de negocios. Esta interacción expone los detalles de implementación del API de la capa de lógica de negocios hacia la capa de presentación. Como resultado, los componentes de la capa de presentación son vulnerables a los cambios de implementación que se realicen en la capa de lógica de negocios, adicionalmente podría existir un detrimento serio en el rendimiento de la red, debido a que los componentes de la capa de presentación que utilizan el API de la capa de lógica de negocios, podrían realizar múltiples invocaciones a través de la red. Esto ocurre cuando los componentes de la capa de presentación no poseen mecanismos de caché o servicios agregados.

Solución: Se puede utilizar el patrón Business Delegate para reducir el acoplamiento entre la capa de presentación y los servicios de la lógica de negocios. El Business Delegate esconde los detalles de implementación de la lógica de negocios y los detalles

de acceso a la arquitectura de los EJB. Este patrón se podría pensar como una abstracción de la lógica de negocios de lado del cliente, también puede manejar las excepciones que se generen desde la lógica de negocios como por ejemplo las excepciones de los EJB y generar excepciones a nivel de aplicación que busquen explicar con mayor exactitud la falla o el error suscitado en un momento determinado.

Finalmente, hay que destacar que este patrón puede reducir la complejidad entre otras capas y no sólo entre la capa de presentación y la de lógica de negocios.

La Figura 36 muestra el diagrama de clases representa al patrón Business Delegate:

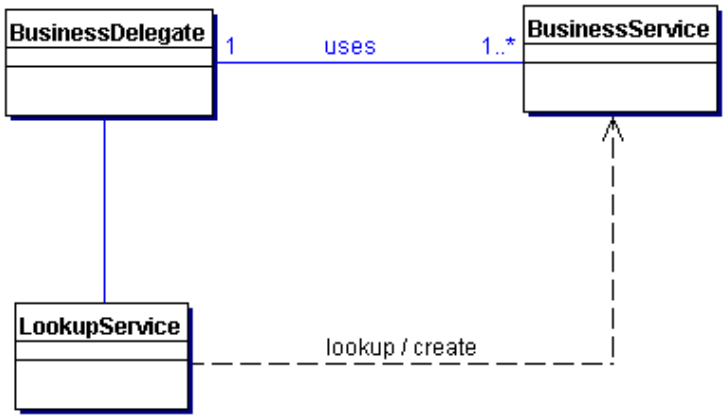


Figura 36 - Diagrama de clases para el patrón Business Delegate

Consecuencias: Reduce el acoplamiento, mejora la gestión, traducción de excepciones específicas e impacto en el rendimiento.

4.3.1.4.3 Patrón Service Locator

Su contexto es la búsqueda y creación de algún servicio que involucre interfaces complejas y operaciones de red.

El Problema: Los clientes J2EE interactúan con los componentes del servicio como los EJB y los componentes de JMS, los cuales proveen servicios de negocios y persistencia. Para realizar la interacción con dichos componentes, los clientes deben localizar el componente del servicio (esto se conoce también como operación de búsqueda) o crear un nuevo componente. Por ejemplo, un cliente EJB debe localizar el objeto home del enteprise

bean, que posteriormente utilizará para localizar un objeto, o bien crear o remover alguno de los ya existentes.

La Solución: Se debe utilizar el objeto Service Locator para realizar una abstracción del uso de JNDI y así ocultar su complejidad para realizar los procesos de la creación del contexto inicial, la búsqueda del objeto EJB Home y la re-creación del objeto EJB. Múltiples clientes pueden reutilizar el objeto Service Locator para reducir la complejidad del código y así proveer un punto particular de control y aumentando el rendimientos, mediante la inclusión de mecanismos de caché.

Este patrón reduce la complejidad del cliente que resulta de la dependencia y la necesidad de realizar procesos de búsqueda y creación. Para eliminar estos problemas, este patrón provee un mecanismo que realiza una abstracción de todas las dependencias y de los detalles de red dentro del Service Locator.

La Figura 37 ilustra el diagrama de clases del patrón Value List Handler.

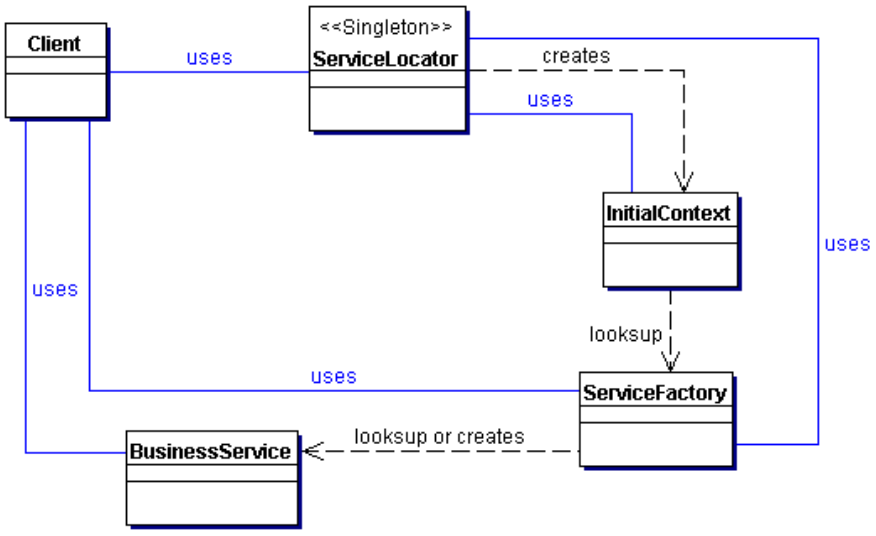


Figura 37 - Diagrama de clases para el patrón Service Locator

Consecuencias: Abstracción de la complejidad, provee a los clientes acceso uniforme al servicio, adición de nuevos componentes de negocios, rendimiento de la red y rendimiento de la caché.

4.3.1.4.4 Patrón Data Access Object

Su contexto es que el acceso a los datos puede variar dependiendo del origen de los mismos (datasource). El acceso a datos persistentes, tales como una base de datos, varía en gran parte del tipo de almacenamiento (bases de datos relaciones, orientadas a objetos, etc) y la implementación del fabricante.

El problema: Muchas aplicaciones J2EE pueden requerir persistencia de datos en algún momento. Para muchas aplicaciones, los repositorios persistentes utilizan mecanismos diferentes y existen grandes diferencias entre las APIs usadas por los distintos repositorios. Otras aplicaciones pueden necesitar acceder a los datos que residen en un mainframe, un servicio B2B o un repositorio LDAP.

Generalmente, las aplicaciones usan componentes distribuidos compartidos para representar la persistencia de datos. Una aplicación es analizada para determinar si debe utilizar persistencia manejada por el bean (BMP) o por el contenedor de beans (CMP). Las aplicaciones pueden usar el API JDBC para acceder a los datos residentes en una base de datos relacional. El API JDBC permite un acceso estándar y manipulación de datos en un almacenamiento persistente. JDBC permite a las aplicaciones J2EE usar sentencias SQL, las cuales son estándar para el manejo de tablas en bases de datos relacionales. Sin embargo, incluso dentro de un ambiente relacional, la sintaxis y el formato de las sentencias SQL puede variar dependiendo del manejador de base de datos que se utilice.

La solución: Usar un Objeto de Acceso a Datos (DAO) para abstraer y encapsular todos los accesos realizados al origen de datos y son el elemento principal de este patrón. Los DAO manejan e implementan los mecanismos de acceso requeridos para trabajar la conexión con el origen de datos. Los orígenes de datos pueden ser un repositorio persistente tal como un Sistema Manejador de Bases de Datos Remoto, un servicio externo (por ejemplo una transacción entre dos empresas), una base de datos LDAP o una función de negocio que puede ser accedida vía CORBA ó IIOP. Los componentes de negocios que dependen de los DAO usan interfaces simples para la interacción con los clientes. Debido a que la interfaz mostrada por los DAO a los clientes no cambia cuando la implementación subyacente del origen de datos cambia, este patrón permite a los DAO adaptarse a diferentes esquemas de almacenamiento sin afectar a sus clientes o a los objetos de lógica de negocio. Los DAO actúan como un adaptador entre los componentes y los datasource.

La Figura 38 muestra el diagrama de clases y las relaciones usadas por el patrón Data Access Object.

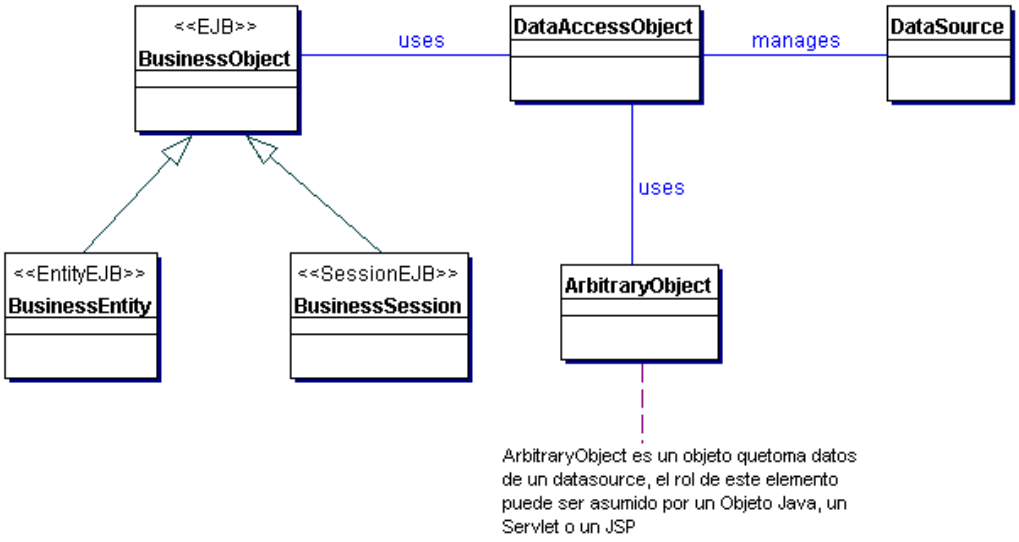


Figura 38 - Diagrama de clases para el patrón Data Access Object

La clase BusinessObject representa a los clientes y es un objeto que requiere acceso a los datasource para obtener y almacenar datos. Un objeto de tipo *Bussiness Object* puede ser implementado como un bean de sesión, un entity bean, o algún otro objeto Java, incluyendo un servlet o un helper bean para acceder a los datos.

La clase DataAccessObject (DAO) es el objeto principal de este patrón, estos proveen las características de encapsulamiento y delegación al *Business Object.* Los DAO se abstraen de la implementación subyacente para acceso a los datos permitiendo que los objetos de negocio tengan un acceso transparente al origen de datos. Los objetos de Negocio delegan a los DAO las operaciones de almacenamiento y obtención de los datos.

La clase DataSource representa la implementación usada por un datasource. Estos pueden ser una base de datos relacional, una base de datos orientada a objetos, un repositorio XML o un sistema de archivos planos entre otros.

Consecuencias: Facilita la Transparencia de los objetos, facilita la migración a diferentes implementaciones de almacenes de datos, reduce la complejidad de código de los Objetos de Negocio y centraliza todos los accesos a datos en una capa independiente.

4.3.2 Diagrama de Secuencia

Un diagrama de secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia de tiempo.

El eje vertical representa el tiempo y en el eje horizontal se colocan los objetos yactores participantes en la interacción sin un orden prefijado. Cada objeto o actor tiene una línea vertical y los mensajes se representan mediante flechas entre los distintos objetos. La Figura 39, contiene el diagrama de secuencia definido para el caso de uso crear proceso, el cual, presenta los diferentes mensajes que fluyen a través de las instancias de las clases definidas en el diagrama de clases.

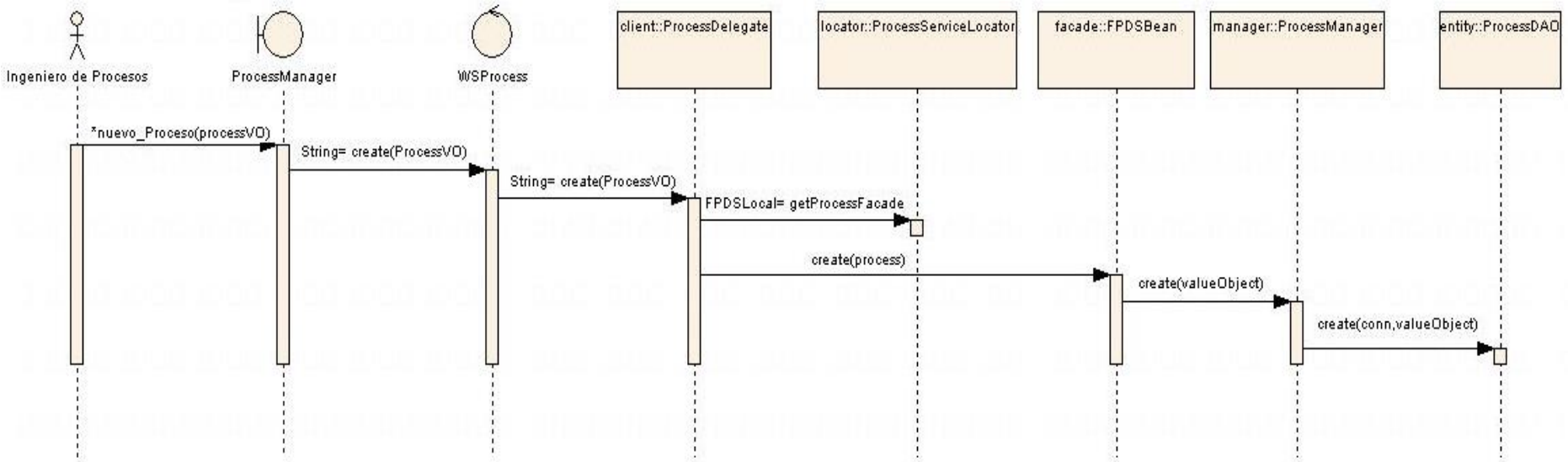


Figura 39 – Diagrama de Secuencia: Creación del proceso

4.3.2.1 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones, representan todos los tipos de elementos software que participan en la fabricación de aplicaciones informáticas, pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente.

El diagrama de componentes del Framework PDS (Figura 40), se compone de cuatro grandes módulos que contienen toda su arquitectura. El componente XMI Generator contiene la herramienta de modelado en SPEM, tiene una relación con el componente contenedor de los documentos o artefactos mas significativos del proceso (Modelo SPEM, el documento XMI y el documento XPDL) dado que éste componente es el encargado de generar el modelo SPEM y el documento XMI.

El componente Jboss App Server contiene la lógica de negocio de la aplicación, utiliza los documentos generados y representados en el componente contenedor de los documentos para la documentación del proceso. Para el almacenamiento de la información gestionada en este modulo, se comunica con el componente BDMS, donde se encuentran la persistencia del Framework PDS.

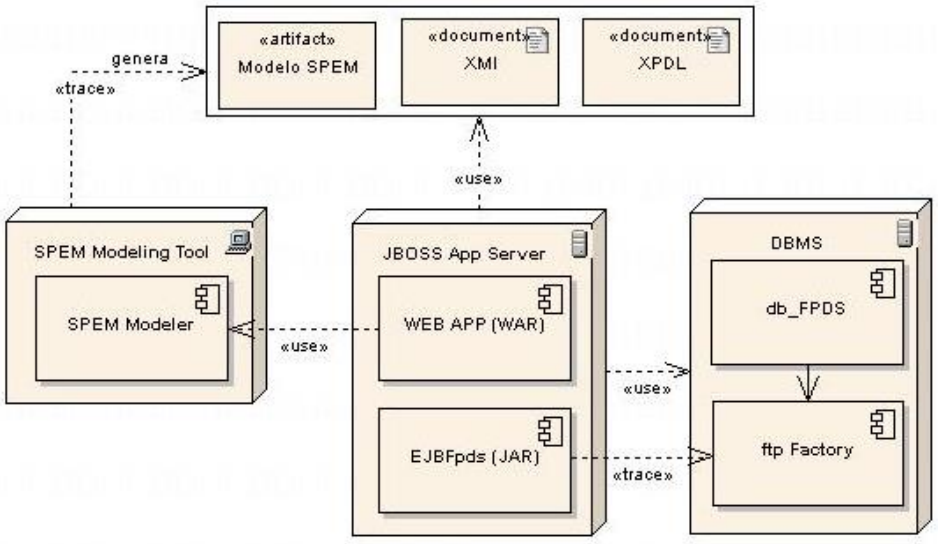


Figura 40 - Diagrama de componentes Framework PDS

4.3.2.1.1 Modelo de Persistencia

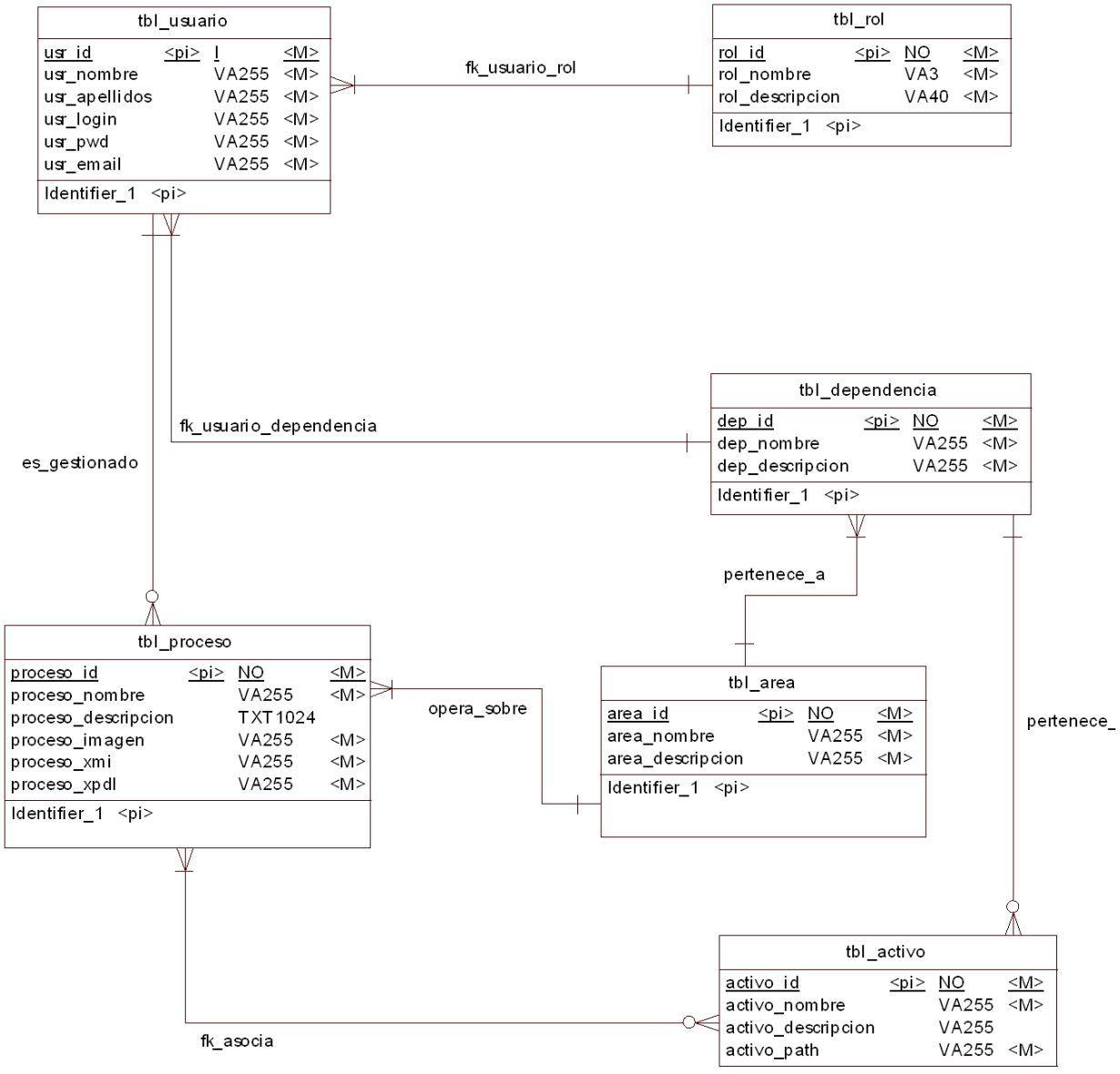


Figura 41 - Modelo Entidad relación de la base de datos del Framework PDS

El modelo entidad relación representa las tablas que son necesarias para el Framework PDS y en las cuales se almacenará la información gestionada por los usuarios del sistema, tal información consta de los usuarios y roles asociados, las dependencias organizacionales y las áreas que las componen, los procesos con todas las características que permitirán su documentación y visualización, incluyendo los activos que le serán asociados.

5 CASO DE ESTUDIO: APLICACIÓN DEL FRAMEWORK PDS AL PROCESO DE DESARROLLO DE SOFTWARE DE SIDEM LTDA

5.1 FRAMEWORK PDS ENMARCADO EN UN PROCESO DE MEJORA

Para contextualizar el Framework PDS, es necesario definir un modelo de negocio, que es una técnica para comprender los procesos de negocio de la organización. El objetivo es identificar los casos de uso del software y las entidades de negocio relevantes que el software debe soportar, de forma que se puede modelar sólo lo necesario para comprender el contexto [7]. El resultado de esta actividad es un modelo del dominio derivado de la comprensión del funcionamiento del sistema de negocio que lo rodea, presenta un sistema (en este caso, el negocio) desde la perspectiva de su uso y esquematiza cómo proporciona valor a sus usuarios.

Dado que el Framework PDS esta enmarcado dentro del proyecto de mejora AGILE SPI, la Figura 42 muestra de manera muy general, el modelo de negocio asociado a la unidad encargada de liderar la definición y mejora del proceso software dentro de una organización.

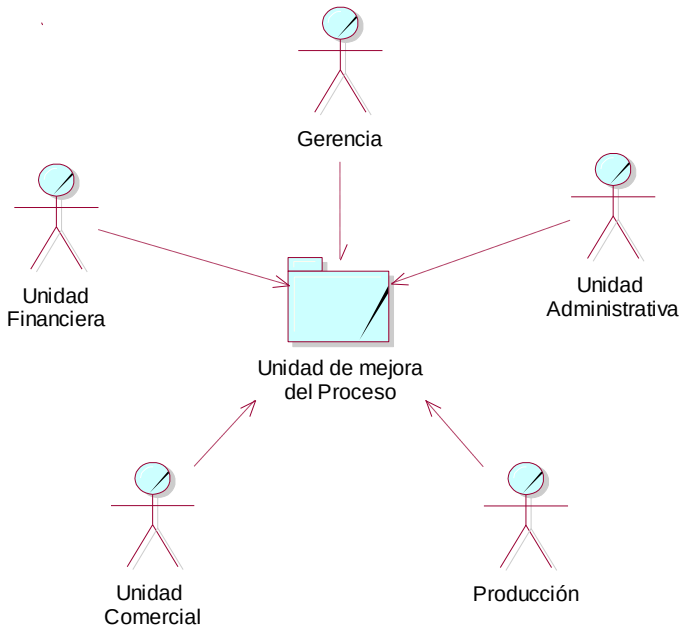


Figura 42 - Modelo de negocio general Framework PDS

La *Unidad de Mejora del Proceso* (que para nuestro caso es el modelo AGILE SPI) tiene como objetivo principal la mejora continua del proceso productivo. Para lograrlo, es necesario involucrar activamente las diferentes áreas de una organización que de una u otra manera inciden en la mejora de los procesos internos, las cuales han sido definidas como actores del negocio.

La *Gerencia* toma y apoya la decisión de crear una unidad de mejora de procesos, independientemente que se busque una certificación o no. El gerente de la organización debe tener claro que la mejora de los procesos no es una actividad sencilla y que requiere mucho tiempo, recursos y esfuerzo para lograrlo, que su participación es fundamental en la toma de decisiones y asignación de roles y responsabilidades y que además, debe acompañar a la unidad durante todo el proceso. Por esta razón, ha sido involucrada como un actor del negocio.

La mejora de procesos software requiere la asignación constante de recursos tanto humanos, logísticos y económicos. Por esta razón, se ha involucrado la *Unidad Financiera* como otro actor de negocio fundamental. El actor *Unidad comercial* en una organización desarrolladora de software, generalmente es el encargado de determinar los requisitos relacionados con el producto, la revisión de los mismos, la comunicación con los clientes y en si, la comercialización del producto. Es muy importante involucrar esta unidad en la mejora del proceso dado que con él, están involucrados procesos críticos, que no solo están en el interior de la organización sino que son visibles al exterior, en donde es importante mostrar una empresa organizada. Igualmente, la *Unidad Comercial* impone restricciones de calidad y de tiempos a los proyectos, restricciones que tienen implicaciones en la definición de los procesos.

El actor de negocio *Unidad Administrativa* juega un papel importante en la organización de las actividades y los recursos, para que la mejora del proceso tenga éxito dentro de la organización.

En la *Unidad de Producción* es en donde se sintetiza la mayoría del trabajo de mejora de procesos, dado que es en ésta unidad en donde se instancian los procesos y se ve reflejado la mejora de los mismos. Igualmente, los requisitos de mejoramiento son obtenidos del diagnóstico que se realice a sus procesos.

En la Figura 43, se muestra el modelo de casos de uso de negocio identificados para el Framework PDS dentro de AGILE SPI.

El caso de uso de negocio *Instalar Proyecto de Mejora* se inicia cuando la *Gerencia* decide dedicar un esfuerzo a la mejora de sus procesos, para ello se desarrolla un plan de mejora, en el cual se fijan objetivos, se asignan responsabilidades y recursos necesarios para llevarlos a cabo. Los actores de negocio que participan en este caso de uso son: La *Gerencia* quien apoya la iniciativa, la *Unidad de Producción,* sobre la cual se realizará el trabajo de mejora, la *Unidad Financiera* analiza y aprueba los recursos financieros del proyecto y la *Unidad Administrativa* aprueba la infraestructura de soporte del proyecto.

El caso de uso *Evaluar/Certificar Proceso* sintetiza las actividades que estarán encaminadas a evaluar los procesos actuales de la organización por medio de una herramienta de valoración, que para el proyecto SIMEP-SW, es SPQA.WEB, la cual, además de obtener las áreas del proceso que están débiles en la organización, realiza recomendaciones basadas en modelos de calidad. Las actividades de certificación están encaminadas a conseguir el reconocimiento de los procesos de la organización ante entidades externas autorizadas por las organizaciones certificadoras de modelos de calidad como ISO y CMMI.

Diseñar Procesos es un caso de uso que contiene las actividades necesarias para que una organización diseñe todos sus procesos. La participación del actor de negocio *Producción* es fundamental, dada la importancia que los procesos tienen en ésta área organizacional, la *Unidad Administrativa* desarrolla un papel importante en lo referido a la gestión de los recursos de infraestructura, los cuales son elementos determinantes de los procesos nuevos o actualizados y la *Unidad Financiera* destina los recursos necesarios para cumplir los objetivos e impone restricciones de tipo financiero a éstos procesos.

Después de tener todos los procesos diseñados y mejorados, es necesario implementarlos, por esta razón, tenemos el caso de uso *Instanciar Procesos*, en el cual, el proceso es "montado" en producción. Es por esta razón que solo se ha considerado la participación del área organizacional de *Producción*.

Cuando se han instanciado los procesos se debe seguir con el plan de mejora continua, y es por esto que es necesario evaluar la mejora realizada a éstos procesos, el caso de uso *Evaluar Mejora*, ofrece la posibilidad de medir el rendimiento de la mejora y realizar una retroalimentación a los procesos. Por esta razón, es necesario que el área de *Gerencia* defina unos objetivos para continuar con la mejora de los procesos en la organización, apoyada por la *Unidad Administrativa* y la *Unidad Financiera*, para así tener a la *Unidad Productiva* involucrada con ésta meta.

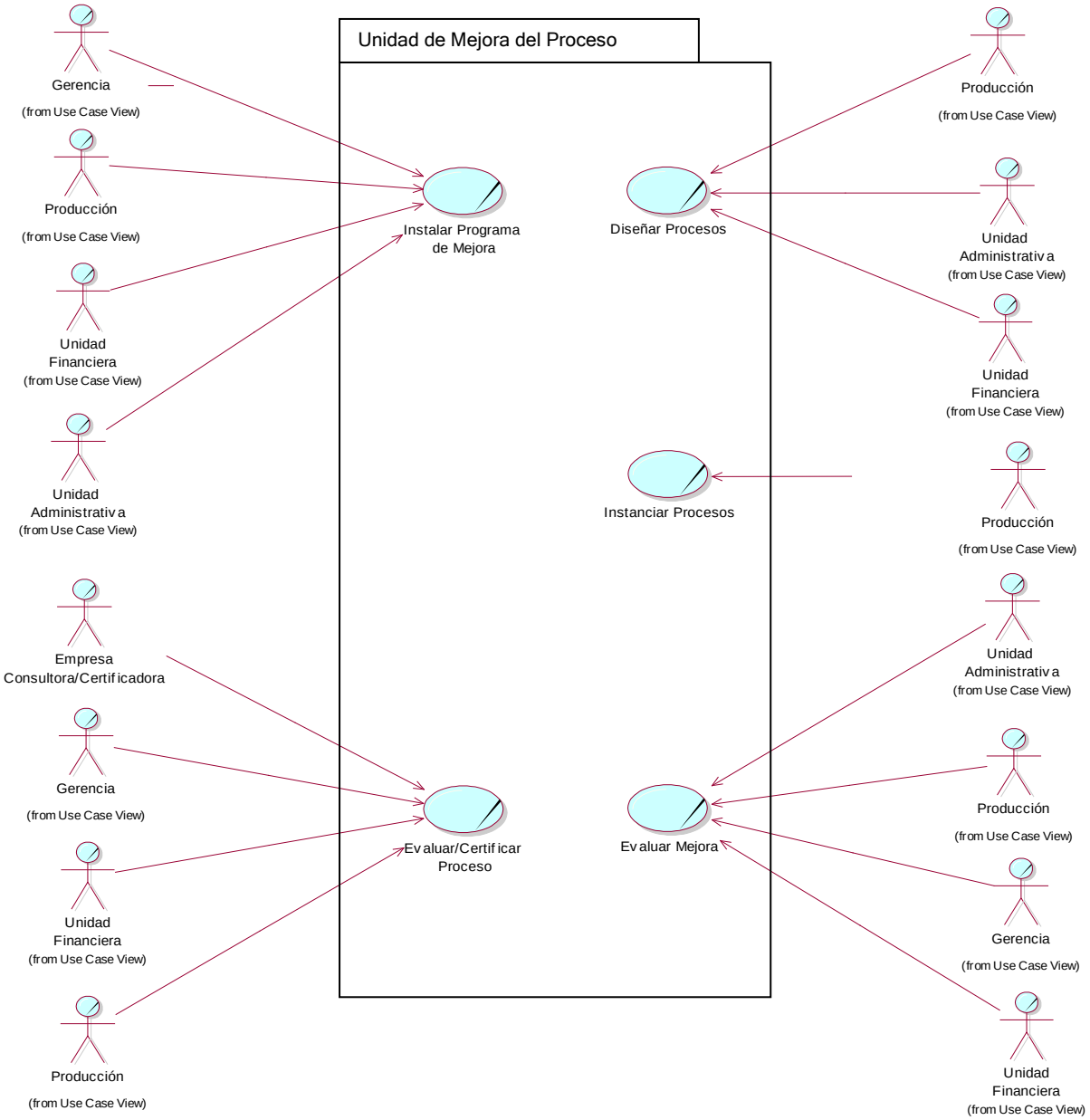


Figura 43 - Modelo de casos de uso de negocio Unidad de Mejora del Proceso

Para los casos de uso de negocio Diseñar Procesos e Instanciar Procesos, las herramientas software juegan un papel importante para facilitar el trabajo en estas áreas, dado que ofrecen la posibilidad de generar marcos de trabajo en donde el personal de la organización puede diseñar y documentar los procesos existentes y es en el diseño de procesos en donde el Framework PDS se encuentra enmarcado.

El modelo de negocio, facilita realizar una aproximación del Framework PDS en una organización y sobre todo, ofrece más claridad al momento de ubicarlo dentro de un proceso de mejora, además permite identificar las áreas que serán importantes dentro de la definición del proceso. Luego de tener identificados estos aspectos, el siguiente paso el siguiente paso es definirle modelo de casos de uso generales, el cual tiene las principales funcionalidades del Framework PDS.

5.2 LA DEFINICIÓN DEL PROCESO

El objetivo del trabajo desarrollado en la empresa piloto SIDEM LTDA, es tener un caso de estudio con una organización que refleja en su mayoría, la realidad de la industria de software en el sur occidente colombiano[14], y en la cual son aplicables los conceptos abordados por este trabajo en lo relacionado a la definición y visualización de procesos, documentación de las tecnologías de procesos y los conceptos propuestos por el proyecto SIMEP-SW.

Con este trabajo se busca dar el primer paso para la colaboración entre academia-industria, formando una alianza estratégica que permita el crecimiento de ambos sectores con el enriquecimiento y la retroalimentación en todas sus áreas, contribuyendo de esta manera con el desarrollo de la industria de software colombiana y en general, con el desarrollo del País.

[14] Esta afirmación esta soportada por los resultados de encuestas realizadas a pequeñas y medianas empresas (en los departamentos de Valle, Cauca y Nariño) por el personal de SIMEP-SW.

5.2.1 Metodología Implementada en la Identificación de los Procesos de SIDEM LTDA

SIDEM LTDA un grupo empresarial con mas de cinco años de experiencia, dedicado a la producción, integración, mantenimiento, respaldo y asesoría de sistemas de información, con diseño multiplataforma estructurado para soportar los constantes retos de renovación de los procesos productivos de las organizaciones modernas. En la actualidad, cuentan con más de trescientos clientes en el territorio nacional que utilizan sus soluciones administrativas, financieras y de automatización.

Para la identificación de los procesos de la empresa SIDEM LTDA, se utilizó la metodología de entrevistas, realizadas a los funcionarios que hacen parte de los mismos, permitiendo identificar claramente el proceso y los detalles que éste puede llevar dentro del contexto real.

Las entrevistas se realizaron de una manera informal, realizando un diálogo con el personal clave de los diferentes procesos de la empresa. Se hicieron preguntas dirigidas a capturar la información necesaria para conocer y posteriormente definir los procesos y a su vez, permitiendo que fueran los mismos funcionarios los que identificaran las áreas críticas en el funcionamiento de la organización. A diferencia de los funcionarios encargados del departamento de Calidad, también se realizaron entrevistas a:

- Subgerente General
- Jefe de Producción
- Jefe del Departamento de Atención al Cliente
- Departamento de Calidad
- Coordinador de Desarrollo
- Desarrolladores

En dichas entrevistas, la mayoría del personal coincidió en que los procesos que actualmente poseen mayores inconvenientes o presentan un caos mayor en su realización son el proceso de Desarrollo y el proceso de Atención al Cliente. El trabajo inicial se centrará en éstos dos procesos, dado que éstos son en gran parte, la razón social de SIDEM LTDA.

Las ventajas de utilizar esta metodología es el conocimiento propio que el equipo externo toma al estar personalmente en la empresa, se capturan fácilmente detalles que serian muy difíciles de observar si no se tiene este tipo de estrategia. Las entrevistas en forma de dialogo, permiten que los diferentes equipos se conozcan y se genere un mejor ambiente para la realización del proyecto de mejora, también, éstos diálogos no cierran las respuestas de los entrevistados, obtienen relatos completos que facilitan el objetivo inicial que es conocer el proceso que se tiene y los detalles puntuales del mismo.

5.2.2 Identificación y Descripción de los Procesos de SIDEM LTDA

A continuación, se hace una descripción de los procesos que actualmente se llevan en SIDEM LTDA, tal y como lo reflejó la información de las entrevistas realizadas. Al final de cada descripción, se encuentra la figura del modelo de proceso basado en el perfil SPEM.

5.2.2.1 El Proceso de Desarrollo

La descripción del Proceso de Desarrollo de SIDEM LTDA, es una consolidación de las diferentes entrevistas realizadas a los Ingenieros de la organización, con lo que se logró tomar diferentes perspectivas, haciendo el trabajo más objetivo y logrando en el mismo, capturar detalles de gran importancia para la identificación y documentación de los procesos internos y con esto, dar inicio al programa de mejora. La Figura 44, ilustra el modelo del Proceso de Desarrollo definido en SIDEM LTDA.

El proceso de desarrollo inicia cuando el Gerente o Subgerente toma la decisión del negocio y lo comenta al Jefe de Producción quien a su criterio, compone el equipo de trabajo que será el encargado de la realización del proyecto desde su inicio hasta que el producto es liberado al cliente. Este equipo de trabajo, generalmente lo compone el Jefe de Producción y dos desarrolladores.

El equipo de trabajo se dirige al cliente para levantar los requerimientos, para lo cual, en ocasiones se utiliza el Formato de Reunión de Especificación de Requerimientos. Este formato posee dos versiones, que se diferencian en el espacio disponible para definir los requerimientos y su razón, es porque algunos stakeholders, prefieren detallar más los requerimientos textualmente en este campo.

El siguiente paso es el diseño de la base de datos, para lo cual, se reúne el equipo de trabajo para generar y definir el modelo entidad relación y se realiza un análisis de la aplicación. Para éste análisis, la empresa no tiene parámetros o documentos que describan una determinada metodología de desarrollo. Cada empleado realiza esta labor según su formación académica. Algunos Ingenieros utilizan UML con diagramas de casos de uso y un diagrama general de clases. Actualmente, se está implementando parte de la Metodología para la Implementación de Servicios, que utiliza un modelo en espiral y esta basada en UP y UML, facilitando y permitiendo documentación referente al proceso.

Luego de tener los modelos, se genera el código ayudados por la herramienta de desarrollo Genexus. No se realiza documentación del código, es una actividad que se hace a libre albedrío y en el momento en que se tiene un prototipo funcional, se realizan pruebas por parte del desarrollador, no se cuenta con personal dedicado específicamente a esta labor y las pruebas no son rigurosas, se prueba funcionalmente el software y no se realiza ningún tipo de documentación o almacenamiento de los resultados de dichas pruebas. Las aplicaciones no cuentan con el modulo de Ayuda, aunque se realiza un manual de usuario que guía al cliente en la utilización del producto.

El software es entregado al cliente, un funcionario de SIDEM LTDA es el encargado de instalarlo (generalmente es el mismo desarrollador del proyecto). Se capacita a un funcionario o empleado del cliente en el manejo del producto. El cliente se encarga de probar el prototipo y sugiere cambios, los cuales son tomados por escrito por el funcionario de SIDEM LTDA. Al final, estas sugerencias son discutidas con la Gerencia para estudiar su viabilidad. Si existe la viabilidad de los cambios, estos son realizados siguiendo los mismos pasos de desarrollo mencionados anteriormente.

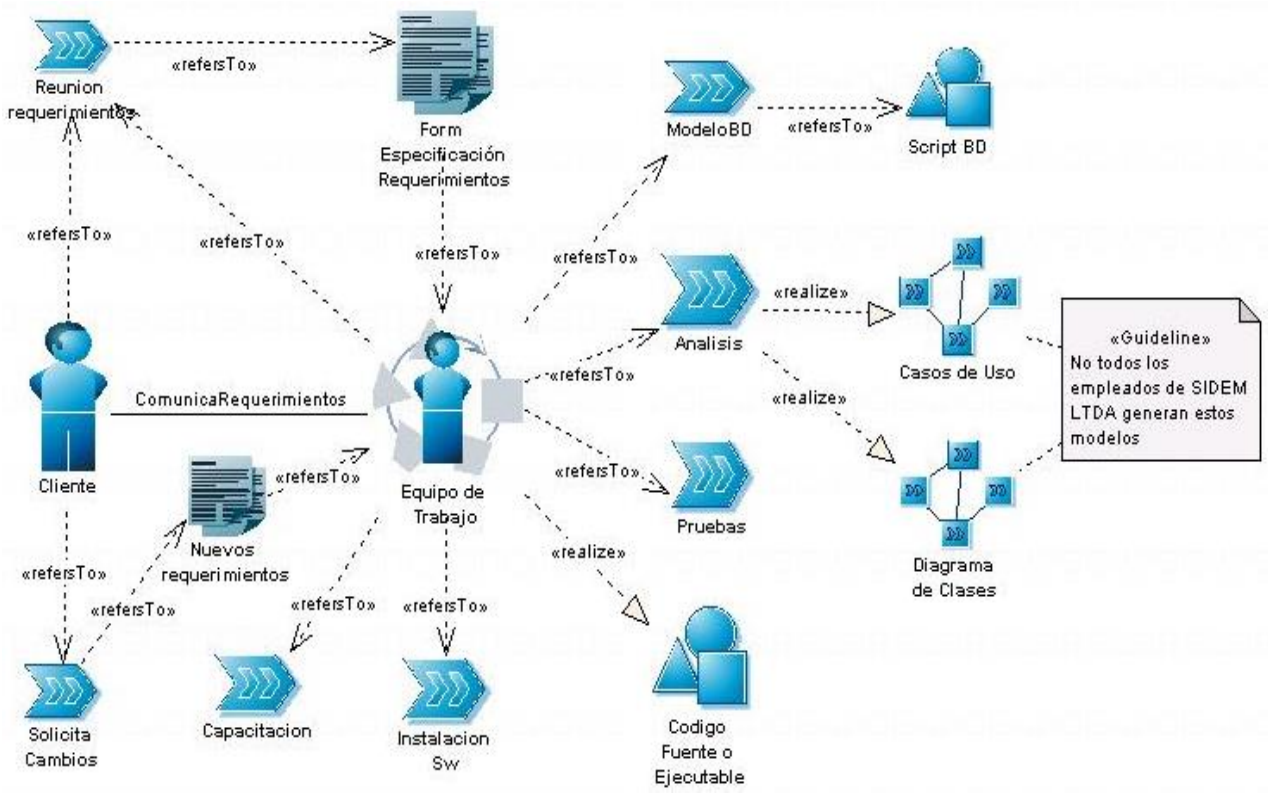


Figura 44 - Modelo del Proceso de Desarrollo de SIDEM LTDA basado en SPEM

5.2.3 Proceso de Atención al Cliente

La descripción de éste proceso parte de la entrevista realizada al Jefe del Departamento de Atención al Cliente (Figura 45) e inicia cuando un cliente realiza una llamada de consulta de alguna de las aplicaciones software que SIDEM LTDA tiene en el mercado. La secretaria es quien recibe las llamadas de la empresa y cuando es una de soporte o atención al cliente, ésta es transferida al jefe del Departamento de Atención al Cliente, el cual, soluciona en un 40% las dudas. Si la duda no la soluciona satisfactoriamente, se le comunica a un desarrollador. Si la duda del cliente persiste, se pasa por escrito y es enviado a las instalaciones del cliente, un funcionario que, generalmente es un desarrollador para que solucione las dudas. Esta visita es programada en un cronograma de visitas, donde se especifica el cliente, la fecha y el funcionario. Existen clientes que realizan las llamadas telefónicas directamente a los desarrolladores que en ocasiones anteriores se han dirigido a sus instalaciones, dudas que son solucionadas y su procedimiento no es conocido por el jefe de atención al cliente.

Cuando el funcionario encuentra un error grave, se lleva los datos a SIDEM LTDA para que se le de solución en la empresa y antes de que un empleado realice algún cambio, se hacen copias de seguridad de la información de los clientes para evitar contratiempos.

Otra forma de solucionar las dudas es por medio del Messenger, Skype, correo electrónico o en ocasiones se utiliza el acceso remoto, pero estos medios no son muy preferidos por los clientes.

Cuando se termina con la visita, el funcionario de SIDEM LTDA se encarga de llenar el formato físico denominado Reporte de Visita, donde se describe la fecha, el tipo de servicio prestado, los módulos atendidos, la descripción de la actividad, la hora de inicio y la hora de terminación. Se deja copia al cliente y el original es guardado en los archivos de SIDEM LTDA.

La experiencia del Jefe del Departamento de Atención al Cliente le permite afirmar que los clientes que mas inconvenientes tienen con el software son los del sistema de presupuesto estatal de hospitales y alcaldías. Los clientes de activos fijos también utilizan mucho el servicio, pero más por dudas de funcionamiento del software.

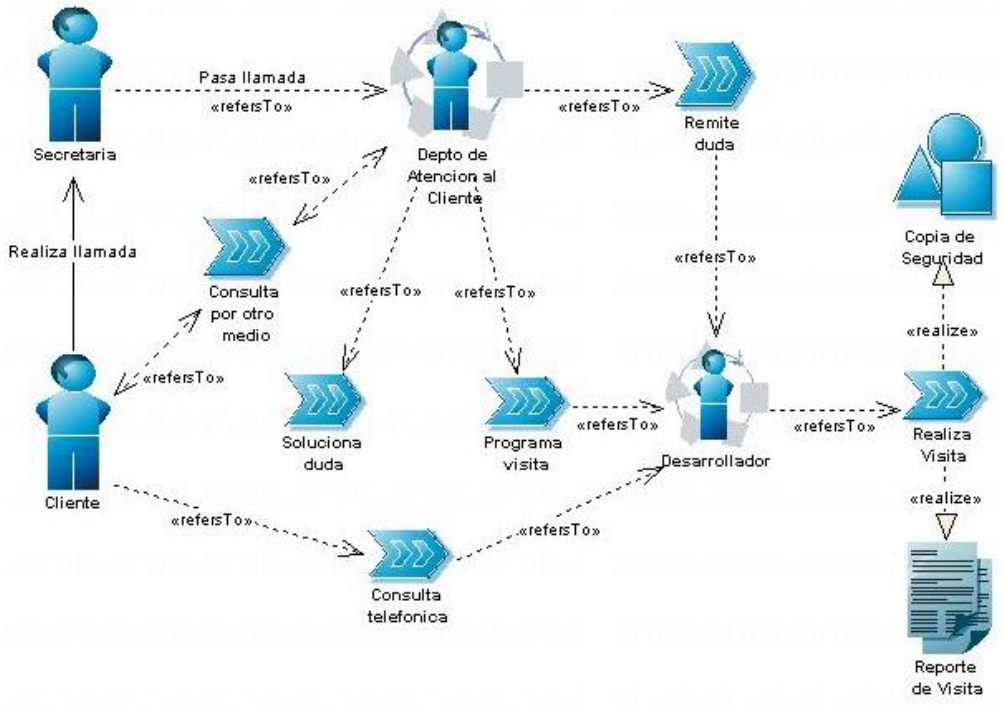


Figura 45 - Modelo del Proceso Atención al Cliente SIDEM LTDA basado en SPEM

En las Figuras 44 y 45 se presentaron los modelos de proceso de desarrollo y de atención al cliente. En el presente trabajo y por cuestiones de presentación, se seguirá la descripción detallada del proceso de desarrollo, dado que es más complejo y facilita que el trabajo sea más amplio y detallado.

La Figura 44, se presenta el proceso de desarrollo de manera muy general, no se hace énfasis en las fases que éste contiene, pero al realizar un estudio mas detallado del proceso, se han identificado tres fases (Figura 46) que serán descritas mas adelante, cada una con su representación SPEM, las cuales, aunque SIDEM LTDA no las tiene explícitamente identificadas, nos permiten tener una mejor organización del proceso llevado a cabo por ellos.

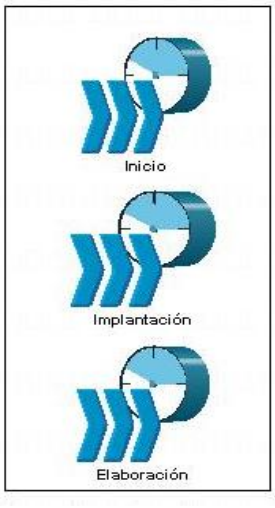


Figura 46 – Fases del Proceso de Desarrollo

Las fases contienen las mismas actividades definidas en el modelo de la Figura 44, como se puede apreciar, estas vistas proporcionan observar el proceso mas ordenado.

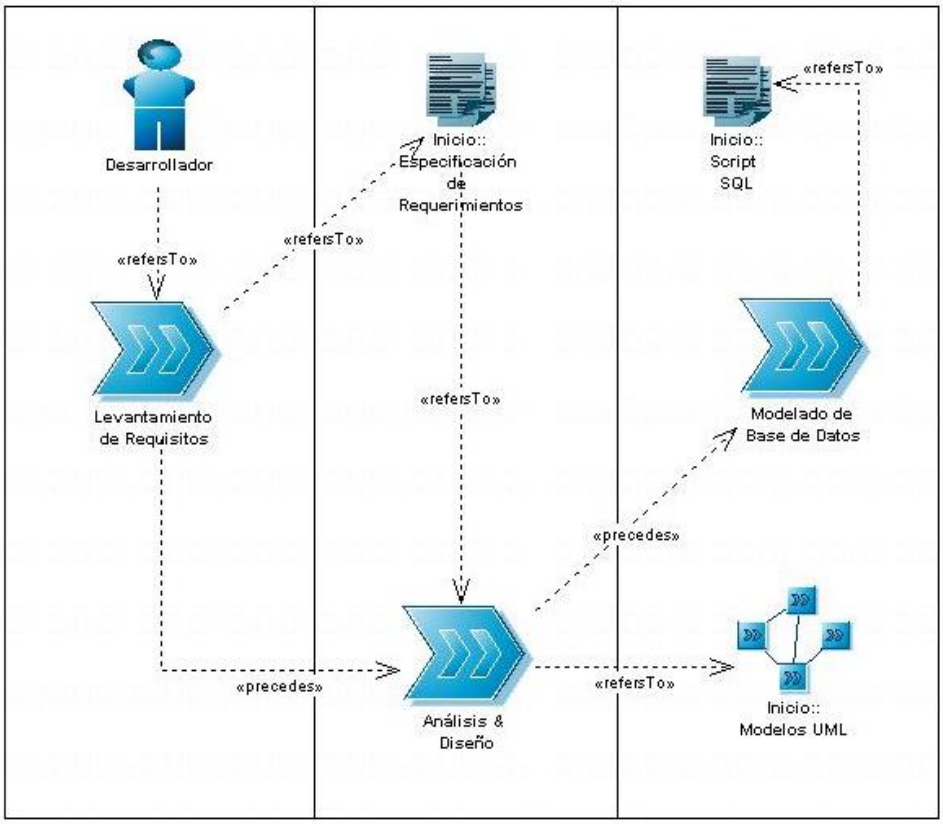


Figura 47 – Fase de Inicio

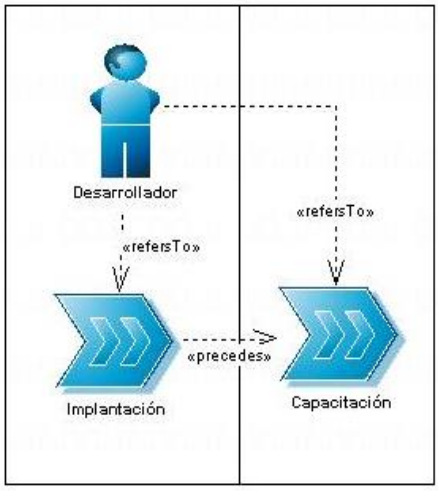


Figura 48 – Fase de Implantación

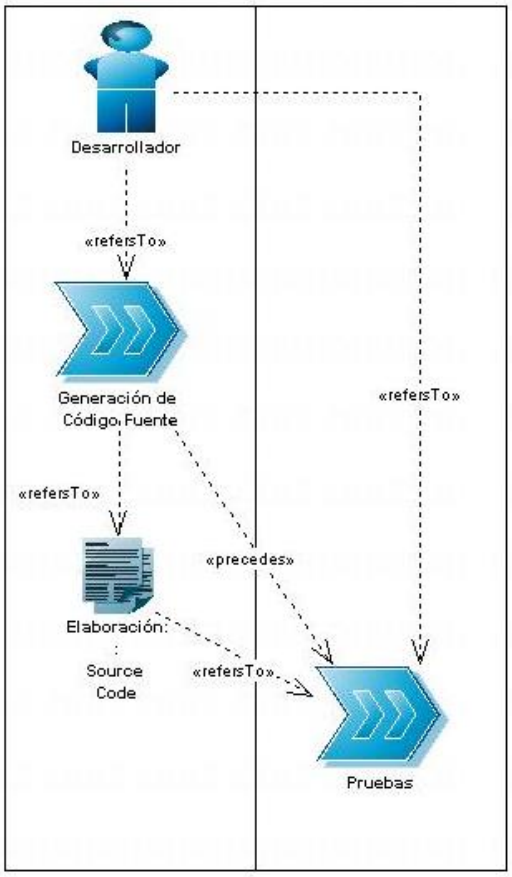


Figura 49 – Fase de elaboración

5.2.4 Resultados del Framework PDS en el caso de estudio

Tanto la definición como la visualización de los procesos software requieren un análisis del flujo de actividades para la consecución de tales objetivos. Para el primer objetivo que corresponde a la definición del proceso, hemos optado por integrar el módulo de modelado y transformación de SPEM propuesto por [34] donde a partir de una gestión previa de especificación del modelo SPEM se pasa a hacer un mapeo a modelo de objetos, como se muestra en la Figura 50. Con los artefactos obtenidos producto de la transformación del modelo, se hace una caracterización del proceso a partir de la solución Framework PDS, esto es: los artefactos obtenidos son importados, descritos y alimentados por activos del proceso que facilitan el entendimiento y seguimiento del proceso. Cuando la caracterización ha sido completada se almacena en un servidor de bases de datos dispuesto para tal fin. A continuación vamos a hacer una descripción más

detallada de la gestión del proceso, desde el modelado hasta lograr la persistencia y visualización de cada uno de los componentes que conforman la solución.

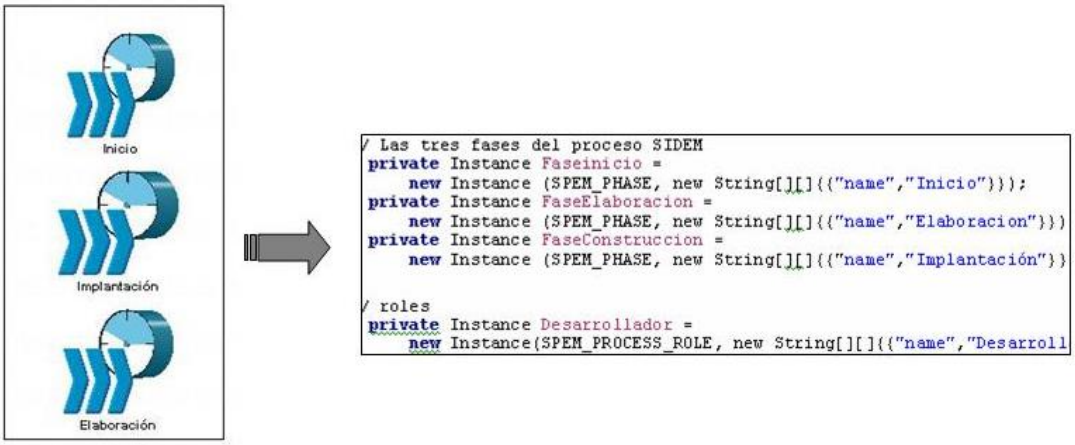


Figura 50 – Abstracción del modelo de objetos a partir de SPEM

Como ya se ha mencionado en apartados anteriores, la metodología tomada para el presente trabajo ha sido el modelado SPEM para luego ser llevado a la especificación Workflow, sin embargo debemos tener en cuenta que no todos los componentes de la especificación SPEM son adaptables a la especificación WfMC. Por lo anterior se hace una abstracción de los elementos comunes para las dos especificaciones.

La especificación de un modelo de procesos en SPEM implica hacer una abstracción de las clases del metamodelo para definir el modelo de objetos correspondiente a dichos procesos. Para esto, [34] muestra dos vistas de las clases SPEM en las que se da una visión más clara para el modelo de objetos que se presenta luego. El modelo de objeto se encuentra representado en un documento XMI que respeta el XMI-DTD del SPEM y fue generado según las reglas de producción de documentos XMI.

La Figura 51 muestra las clases involucradas para la definición de actividades de un proceso de desarrollo de software.

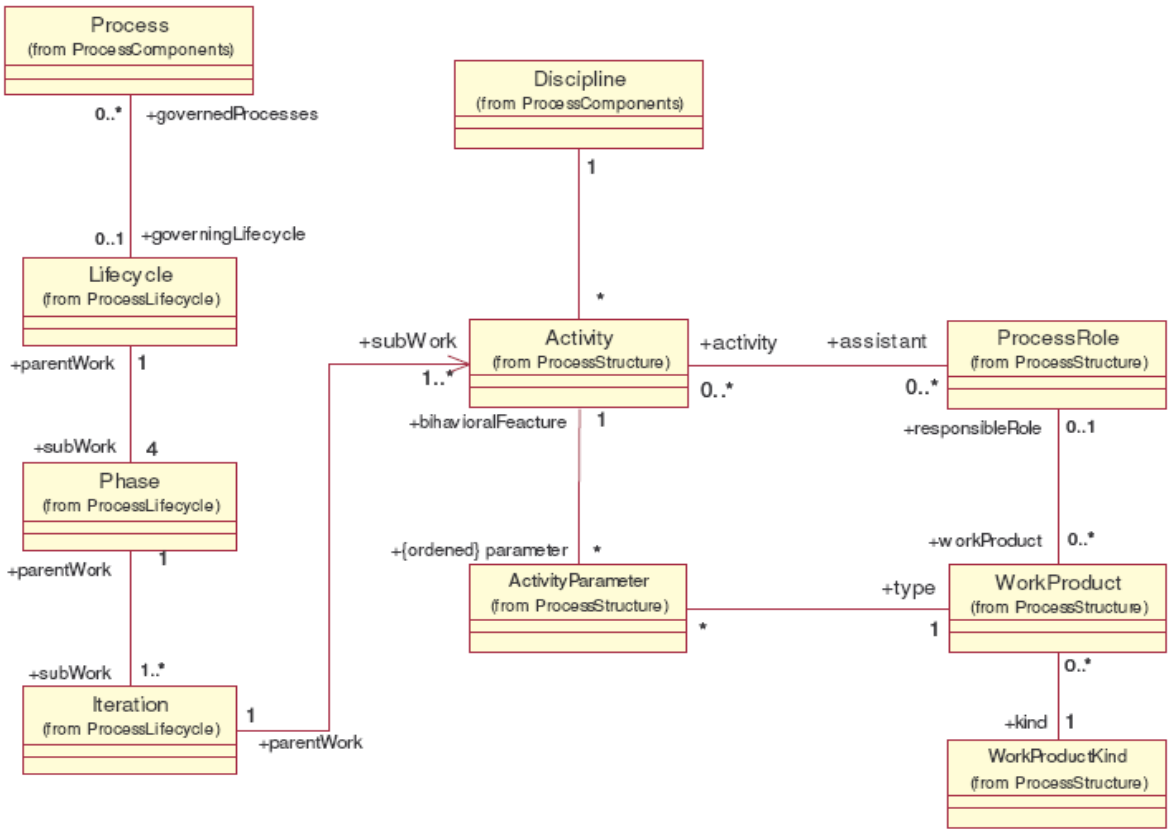


Figura 51 - Vista de definición de las actividades de un proceso de desarrollo [34]

Para formar esta vista se utilizaron los paquetes ProcessComponents, ProcessLifecycle y ProcessStructure, que son parte del paquete SPEM_Extensions, que es el que agrega los constructores y la semántica requerida para la ingeniería de procesos de software [34].

La Figura 52 muestra la definición de dependencias entre actividades en el proceso de desarrollo de software. En su mayoría las clases involucradas pertenecen al paquete State_Machines de SPEM. Debido a que para el modelado del Caso de estudio, no se tiene en cuenta la elaboración de diagramas de estado, la aplicación de esta vista no se tiene en cuenta.

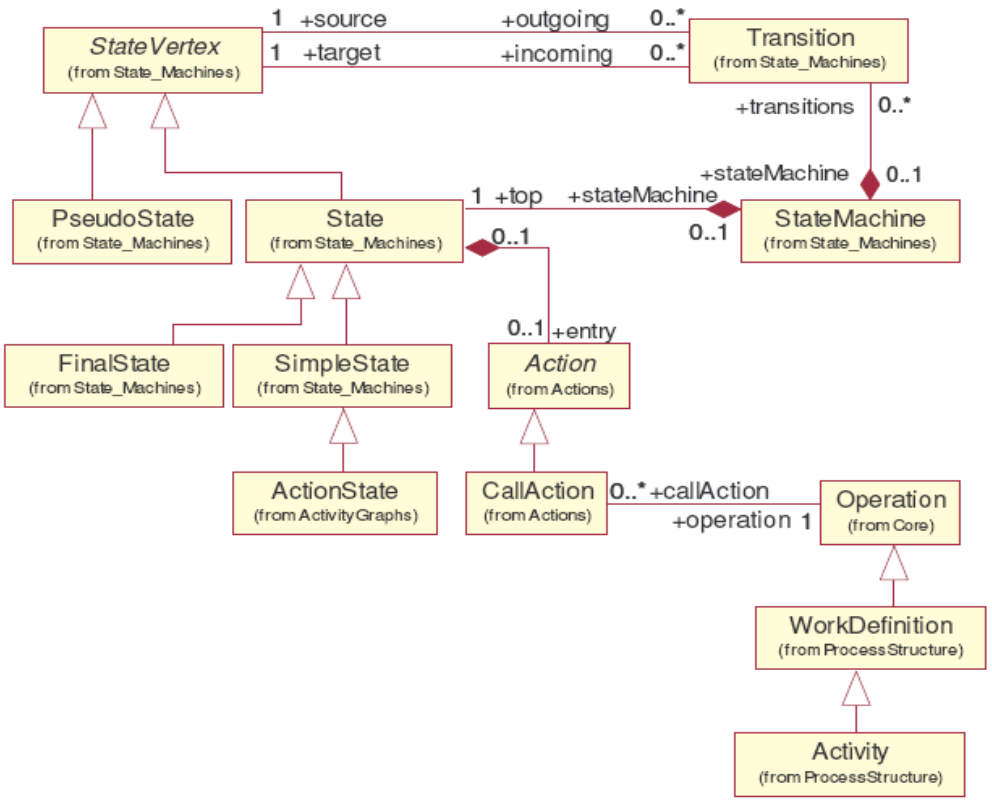


Figura 52 - Vista de definición de dependencias entre actividades [34].

El primer acercamiento a la definición del proceso, como está planteado en el Framework PDS es especificar el modelo del proceso utilizando SPEM. Para llevar a cabo esta tarea, es necesario tener claro el proceso de desarrollo que adelanta la empresa. Para ver el proceso de desarrollo de SIDEM LTDA remítase al numeral 0 del presente documento donde se especifican las fases, actividades y roles así como los artefactos tanto de entrada como de salida que intervienen en el proceso. Adicionalmente es necesario un mecanismo de producción de documentos XMI que representa el modelo de objetos obtenido. Framework PDS integra el componente XMI Generator, desarrollado por [34] para cumplir con este objetivo.

5.2.5 Ventajas de Utilizar el Framework PDS en la Empresa Piloto SIDEM LTDA

La organización en los últimos años ha crecido considerablemente tanto en los procesos internos así como en la vinculación de nuevo talento humano. Dicho crecimiento ha generado cambios en sus procesos que no se han realizado siguiendo un estándar o

patrón que asegure la calidad de los mismos. El interés mostrado por SIDEM LTDA a la investigación realizada en el proyecto SIMEP-SW está enfocado a mejorar la calidad de sus procesos y a futuro, buscar certificaciones de calidad que la acrediten como una empresa confiable para sus clientes y para el marcado nacional e internacional en general.

Las ventajas ofrecidas por el módulo de definición y visualización de los procesos Framework PDS a SIDEM LTDA son:

- Los procesos no contaban con ninguna documentación ni un orden específico, haciendo que las tareas o actividades incluidas en éstos, se realizaran según lo consideraba el stakeholders, sin seguir un orden definido para tal fin, obteniendo productos de trabajo sin un formato similar. Con la documentación de los procesos, se consigue ordenarlos y mantenerlos visibles para toda la organización, unificando formatos de los productos de trabajo y facilitando las labores cotidianas.
- Dado el tamaño de la organización y el corto presupuesto, una buena elección es una herramienta de gestión de procesos libre; una de las características principales que posee el Framework PDS.
- Dado el desconocimiento que los stakeholders de los procesos de SIDEM LTDA tienen en cuanto a las tecnologías de procesos, el marco de referencia es muy útil en su entrenamiento, puesto que ofrece los conceptos básicos para transmitir o infundir la importancia de éstos en la calidad de los productos, economizando tiempo en la consecución de ésta documentación.
- La facilidad de organización de los procesos en el Framework PDS permite que los participantes o stakeholders asuman una mejor organización en la realización de las actividades, contribuyendo a que los procesos sean más organizados y productivos.

Todos los beneficios mencionados anteriormente, permiten a SIDEM LTDA, conseguir una aproximación en la consecución de certificaciones que le permitan ser más competitiva con mejores productos y una mejor percepción organizacional ante sus clientes.

6 RECOMENDACIONES, CONCLUSIONES Y PERSPECTIVAS

- A pesar de los nacientes esfuerzos encaminados a fortalecer la industria de software en Colombia, las organizaciones aun se resisten a enfocar sus esfuerzos en la mejora de sus procesos, sin embargo Framework PDS es una buena iniciativa que gracias a sus estrategias de divulgación ha fortalecido la sociedad Academia – Empresa.

- Framework PDS proporciona un soporte documental muy importante al facilitar una estructura de repositorio de procesos bien definida y con facilidad de documentación, que le permite al usuario final, hacer un seguimiento no traumático del proceso en el cual está involucrado. Esto se ve reflejado en ahorro de recursos, y aumento en la productividad de los empleados, en la medida en que se adapten al cambio organizacional.

- Framework PDS más que ser una solución tecnológica es una solución conceptual soportada sobre estándares ampliamente reconocidos, fácilmente integrable a cualquier proceso de mejora adelantado por una empresa. Integra diferentes componentes que dan como resultado una solución completa para favorecer a las organizaciones la definición, contextualización y visualización de sus procesos y deja abierta la posibilidad de automatizar los procesos de desarrollo al cumplir con la interfaz 1 de WfMC.

- La utilización de XMI facilita la exportación del proceso a diferentes herramientas dado que esta basado en XML, y aunque la portabilidad no sea completa, es un paso importante para que en futuros trabajos se trabaje en este campo para que los stakeholders participantes, cuenten con más herramientas que faciliten su labor dentro de la organización.

- El diseño arquitectónico del Framework PDS está basado en componentes y publicación de servicios Web, además de ser construido sobre un ambiente multiplataforma garantiza que sea una solución altamente escalable. La integración de patrones flexibiliza su mantenimiento y las reestructuraciones.

- Framework PDS abre la posibilidad de trabajar con Ingeniería de Procesos. Los futuros investigadores cuentan desde ya con una base teórica y tecnológica para seguir madurando y aportando a la evolución de la industria del software Colombiana.

- La utilización de Workflow ha favorecido que los procesos modelados con SPEM puedan ser automatizados, sin embargo, la representación de procesos workflow, restringe en parte los modelos originales hechos en SPEM, debido a que la especificación WfMC no soporta una gran cantidad de elementos SPEM.

El trabajo futuro se puede analizar desde muchas perspectivas, a continuación se listan algunas:

- La solución presentada permite la definición y visualización del proceso; se ha hecho una aproximación para cumplir con la entrada a la automatización del proceso. Esta entrada deja la posibilidad de concluir este objetivo, es decir, teniendo en cuenta que la automatización del proceso está fuera del alcance de SPEM y que el presente documento proporciona el cumplimiento de la Interfaz 1 de WfMC, se puede continuar con el objetivo de cumplir con las interfaces 2, 3, 4 y 5 para que el proceso no quede solo definido sino que también se pueda instanciar en un motor de workflow.

- El primer paso en esta investigación estaba encaminado a consolidar una base conceptual de los procesos de desarrollo. Con esta base y la aproximación tecnológica que se proporciona, se hace necesario que futuros investigadores generen una solución tecnológica que facilite el modelado SPEM basado en el metamodelo, con la facilidad de integrar el Framework PDS.

- La exploración de las mejores prácticas para la definición de procesos sería un gran aporte a las organizaciones en su objetivo de establecer un proceso de mejora. Esta exploración permitiría que con una base sólida que permita hacer un balance de recursos, las organizaciones establezcan sus procesos de desarrollo con las mejores condiciones y adaptados a sus necesidades.

7 BIBLIOGRAFIA

[1] HERNÁNDEZ, José L. "Rentabilidad del desarrollo de proyectos de cómputo". México 2002.
http://www.monografias.com/trabajos13/renta/renta.shtml

[2] HURTADO, Julio A. "El modelo integral de mejoramiento Agile SPI".
Departamento de Sistemas, Universidad del Cauca. Popayán, Agosto de 2004.

[3] SIMEP-SW "Sistema Integral para el Mejoramiento de los Procesos de Desarrollo de Software en Colombia (SIMEP-SW)". Departamento de Sistemas, Universidad del Cauca. Popayán, Julio de 2003.

[4] FUGGETA, A. "SOFTWARE PROCESS: A Roadmap". Dipartimento di Elettronica e Informazione. Politecnico di Milano. 1999.
http://www.cs.ucl.ac.uk/staff/A.Finkelstein/fose/finalfuggetta.pdf

[5] REAL ACADEMIA DE LA LENGUA ESPAÑOLA. Diccionario de la Lengua Española. Vigésima segunda edición. 2004. http://www.rae.es/

[6] UNIVERSIDAD NACIONAL DE SAN JUAN. "Glosario Informático". Argentina 2004. http://adig.com.ar/cg3/glosario/

[7] JACOBSON, I. BOOCH, G. RUMBAUGH, J. "El Proceso Unificado de Desarrollo de Software". Edición en Español. Ed. Addison Wesley. 2000.

[8] GARCÍA RUBIO, Félix Ó. "Gestión Integrada de modelado y de la medición del proceso software". Grupo Alarcos. Escuela Superior de Informática. Universidad de Castilla – La Mancha. 2002.
http://lsi.ugr.es/~gedes/actividades/Dolmen4/a5.pdf

[9] PRESSMAN, Roger. "Ingeniería del Software, Un Enfoque Práctico". Cuarta Edición. Ed. McGraw-Hill. Septiembre de 1998.

[10] INSTITUTO TECNOLÓGICO DE MORELIA. "III Enfoques Metodológicos del Desarrollo de Sistemas de Información."
http://antares.itmorelia.edu.mx/cursos/file.php?file=/3/enfoquesmetodologicos.pdf

[11] SPILLNER, Andreas. "The W-MODEL – Strengthening the bond between development and test". University of Applied Sciences Bremen. Germany.
http://www.iti.upv.es/~squac/JTS/JTS2004/docs/Wmodel.pdf

[12] LETELIER, P. PENADÉS, C. "Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)". Facultad de Informática. Universidad Politécnica de Valencia. 2004
http://www.willydev.net/descargas/masyxp.pdf

[13] MANDLIK, Saifulgani. "Comparing Microsoft Solution Framework & Rational Unified Process". Marzo de 2004.
http://www.devSynergy.net

[14] RATIONAL SOFTWARE CORPORATION. Rational Unified Process Version 2003.06.00.65. Documento digital.

[15] PEREZ, G. GIANDINI, R. PONS, C. "Un Metamodelo para Catalysis Basado en el Metamodelo de UML".
http://ideas2004.spc.org.pe/html/pdfs/176.pdf

[16] CERNUDA, Agustín. "Sistema de verificación de componentes software". Febrero de 2002
http://www.di.uniovi.es/~cernuda/pubs/tesis.pdf

[17] RIVERO B, Carlos. "Certificación de procesos de desarrollo de software".
http://www.itba.edu.ar/capis/webcapis/tesisdemagister/rivero-tesisdemagister.pdf

[18] ATRINSON, C. BAYER, J. MUTHIG, D. "Component-Based Product Line Development: The KobrA Approach".

[19] HAPPEL, Hans. "A Case Study in Component-based Software Engineering using the KobrA-Method". Febrero de 2004.

[20] ENTERPRICE UNIFIED PROCESS. Abril de 2005.
http://www.enterpriseunifiedprocess.com
Página principal

[21] www.agilemanifesto.org

[22] DON, Wells. "Extreme Programming: A gentle introduction".
http://www.extremeprogramming.org/

[23] REYNOSO, Carlos. "Métodos heterodoxos en desarrollo de software".
http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.asp

[24] GILB, Tom. "10 Evolutionary Project Management (Evo) Principles". Abril de 2002.
http://www.iti.upv.es/~squac/JTS/JTS2004/docs/Wmodel.pdf

[25] COCKBURN, Alistair. "Balancing Lightness with Sufficiency". Septiembre de 2000.
http://alistair.cockburn.us/crystal/articles/blws/balancinglightnesswithsufficiency.html.

[26] AMBLER, Scott. "Agile Modeling and the Unified Process". Enero de 2005.
http://www.agilemodeling.com

[27] OBJECT MANAGEMENT GROUP. "Software Process Engineering Metamodel". An Adopted Specification of the Object Management Group, Inc; Versión 1.0 formal/02-11-14. Noviembre 2002.

[28] UNIVERSIDAD CARLOS III DE MADRID. "Modelado de Procesos" Documento digital.

[29] HOLLINGSWORTH, David. "Workflow Management Coalition the Workflow Reference Model". 55 p. Enero de 1995

[30] RUIZ G, Francisco. "MANTIS: Definición de un Entorno para la Gestión del Mantenimiento de Software". Tesis Doctoral. Departamento de Informática. Universidad de Castilla – La Mancha. Junio de 2003.

[31] RUIZ G, Francisco. "Mantenimiento Avanzado de Sistemas de Información". Universidad de Castilla – La Mancha. Ciudad Real. España 2004.

[32] OBJECT MANAGEMENT GROUP. "Meta Object Facility (MOF) 2.0 Core Specification". Octubre de 2004.

[33] OBJECT MANAGEMENT GROUP. "XML Metadata Interchange (XMI) Specification" An Adopted Specification of the Object Management Group, Inc; Versión 2.0 Mayo 2003.

[34] ROMERO, Daniel. "Un Workflow que Automatice los Procesos de Desarrollo basados en SPEM". Departamento de Computación; Facultad de Ciencias Exactas Físico Químicas y Naturales. Universidad Nacional de Río Cuarto. Abril de 2005.

[35] WORKFLOW MANAGEMENT COALITION. "The Workflow Reference Model". The Workflow Management Coalition Specification; WFMC-TC-1003 Version 1.1 Issue. Enero de 1995.

[36] WORKFLOW MANAGEMENT COALITION. "Workflow Process Definition Interface -- XML Process Definition Language". Workflow Standard; WFMC-TC-1025 Version 1.0 Final Draft. Octubre de 2002.

[37] GERONIMO G. CANSECO. V. "Breve introducción a los sistemas colaborativos: Groupware & Workflow". Universidad Tecnológica de Mixteca. Mexico.

[38] SQS S.A. "Herramienta que implementa eXtreme Programming para la gestión de requisitos" SQS RequerimentsWORKFLOW. Vizcaya, España www.sqs.es/documentos/eXtreme.pdf

[39] MOLPECERES T, Alberto. PEREZ M, Martín. "Arquitectura empresarial y software libre, J2EE". http://www.javahispano.org/articles.article.action?id=70#recurso-1

[40] ALUR, Deepak. CRUPI, John. MALKS Dan. "Core J2EE Patterns: Best Practices and Desing Strategies" Second Edition. June 2003.

[41] SEMATECH. "*Computer Integrated Manufacturing (CIM) Framework Specification. Version 2.0*". Technology Transfer #93061697J-ENG. International SEMATECH, Inc. http://www.sematech.org. January 31, 1998.

[42] BEDINI G, Alejandro. "Calidad Tradicional y de Software". Departamento de Industrias. Universidad Técnica Federico Santa María, Chile.