Reporte Técnico

Ingeniería de Requerimientos para Líneas de Productos de Software

Gilberto Carlo Grajales Arano
Edwin Gutiérrez León

Versión 0.1
Agosto 2006

Histórico de revisiones

Versión	Fecha	Descripción	Autor
0.1	16-08-2006	Primer borrador.	GGA EGL
0.2	29-08-2006	Versión Final	GGA EGL

Tabla de Contenido

Lista de Figuras

Agradecimientos

Este trabajo fue parcialmente financiado por el Consejo de Ciencia y Tecnología del Estado de Guanajuato (CONCYTEG) a través del proyecto titulado Promoviendo calidad en la industria de software: Recursos humanos, investigación, servicios".

Al Doctor Cuauhtémoc Lemus Olalde por sus valuables recomendaciones dadas para la elaboración de este documento.

1. Introducción

La tecnología en las últimas décadas ha avanzado a pasos agigantados, y tal fenómeno lleva implícito el desarrollo de sistemas de software cada vez más complejos debido a las nuevas necesidades que la misma evolución de la tecnología ha propiciado. La “Ingeniería de Software” no es un término nuevo, pues se acuñó desde los años 60’s y aunque años antes no se conocía como tal, ya se lidiaba con la complicada tarea de desarrollar sistemas de software con calidad.

Debido a las demandantes necesidades de desarrollar software cada vez más complejo, también se han desarrollado nuevos métodos y herramientas para mitigar dicha complejidad y desde sus indicios se ha intentado desarrollar software libre de defectos y cuya funcionalidad sea de completa utilidad para los usuarios del mismo. Sin embargo, dado que el software es desarrollado por humanos, no está libre de contener defectos pues los humanos por naturaleza cometen errores y a raíz de eso se ha buscado una solución única para tal problema (llamada “silver bullet” en [1]), pero desafortunadamente hasta la fecha no existe una “silver bullet” que garantice el desarrollo de software con cero de defectos.

Varios esfuerzos han sido generados para encontrar tal “silver bullet”, destacando entre los más relevantes la “reutilización”. La reutilización, de manera general, consiste en desarrollar elementos de software (desde componentes, módulos, funciones, etc) que puedan ser utilizados más de una vez con la mínima cantidad de modificaciones, de esta manera el garantizar que un elemento de software reutilizable está libre de defectos garantiza que el sistema que lo utilice no tendrá problema alguno en lo que respecta a dicho componente. Desafortunadamente, la reutilización no fue la “silver bullet” esperada ya que a pesar de tener cierto éxito los elementos desarrollados para ser reutilizados se han limitado a ser pequeñas funciones o módulos implementadas lo cual es cierto que pueden ser reutilizadas pero no tienen el efecto esperado en un sistema de software de gran tamaño.

Gracias a esta búsqueda por obtener una reutilización más impactante, surgió el concepto de “Líneas de Productos de Software” (LPS de ahora en adelante) también conocido como “Familias de Sistemas de Software”, la cual tiene indicios desde los años 70’s (en [2] se mencionan los beneficios de tener una familia de sistemas compartiendo características entre ellos). Las LPS surgieron como una evolución de la reutilización, sin embargo en una LPS el nivel de reutilización es muchísimo mayor, puesto que no se limita a la reutilización de fragmentos de código sino que se basa en la reutilización de los artefactos más relevantes en un sistema de software tales como: requerimientos y casos de uso, arquitectura, componentes de diseño a más detalle, componentes implementados y casos de prueba entre los más importantes.

Este trabajo de investigación se centra en el área de LPS, demostrando un caso práctico aplicado a la Gerencia de Desarrollo de Software del CIMAT, A.C. en la cual se tiene el propósito de implantar a largo plazo una LPS para el desarrollo de sistemas de optimización para algunos clientes del CIMAT. Se dice a largo plazo puesto que en este trabajo la investigación se limitó solamente a la parte de requerimientos de software para la LPS, la cual se espera extenderse con futuros trabajos de investigación para completar el resto de las fases para la LPS como la arquitectura, la implementación y las pruebas. Este trabajo de investigación se divide principalmente en tres partes las cuales son descritas brevemente a continuación.

La Parte I trata de los conceptos y teoría relacionada a las LPS. Esta parte se divide en los siguientes capítulos:

Capítulo 2. Este capítulo habla sobre los conceptos fundamentales de las LPS, incluyendo los beneficios implícitos en la implantación de una LPS dentro de una organización.

Capítulo 3. En este capítulo se describen los conceptos acerca de los requerimientos de software relacionados con una LPS, aunque los fundamentos de la Ingeniería de Requerimientos para el desarrollo de un sistema de software son similares a los de una familia de productos de software existen algunos conceptos adicionales para ser aplicados a una LPS.

La Parte II de este trabajo consiste en un pequeño estado del arte en lo que respecta a "frameworks" para LPS. Esta parte se divide en 3 capítulos los cuales se describen brevemente a continuación:

Capítulo 4. Este capítulo trata el método FODA el cual es el método en que se baso una parte de este trabajo pues de el se tomó la fase de modelado del dominio, específicamente el modelado de características para obtener el árbol de características de la LPS en cuestión.

Capítulo 5. Este capítulo trata de un enfoque para el modelado de requerimientos utilizando UML. Dicho enfoque está basado en el método Plus desarrollado por Hassan Gomma.

Capítulo 6. Este capítulo contiene una breve descripción de otros "frameworks" de LPS no menos importantes. Entre éstos se encuentran KobrA, PuLSE, FORM, PLUS, FAST, Odyssey, Synthesis entre los principales.

La Parte III de este trabajo consiste en el desarrollo de la investigación del mismo. Se centra principalmente en la descripción del dominio para el cual se llevó a cabo, así como el resultado del análisis aplicado y la descripción de los artefactos de software obtenidos tales como los casos de uso, requerimientos funcionales y el árbol de características de la LPS a implantar. Esta parte se divide en los siguientes capítulos:

Capítulo 7. Este capítulo describe el dominio para el cual se llevó a cabo la investigación; dicho dominio es la Gerencia de Desarrollo de Software del CIMAT. En él se menciona la necesidad de implantar una LPS para una familia de sistemas de optimización que el CIMAT desarrolla para algunos de sus clientes.

Capítulo 8. Este capítulo describe el modelado de características que fue obtenido a partir del análisis realizado al dominio del problema. Como se mencionó anteriormente en este capítulo se utilizó una de las fases del método FODA para la obtención de dicho modelo.

Capítulo 9. Este capítulo describe el análisis efectuado al dominio del problema, así como los resultados obtenidos (el modelo de casos de uso y requerimientos funcionales), los cuales fueron verificados por el Gerente del Departamento mencionado.

Capítulo 10. Describe el modelo de casos de uso y los requerimientos funcionales para uno de los productos de la LPS, tomando como base el documento generado en el capítulo 9.

Por último, la Parte IV de este trabajo, consta simplemente de las conclusiones asimiladas por el desarrollo del mismo, así como las oportunidades de trabajos futuros que pueden ser realizados como extensión de esta breve investigación.

Capítulo 11. Conclusiones de este trabajo de investigación.

Capítulo 12. Este capítulo menciona las oportunidades que existen para continuar este trabajo de investigación.

Sin más preámbulo, se espera que este modesto trabajo de investigación sea clave para adentrarse en esta fascinante área que son las LPS y al mismo tiempo propiciar el interés en

contribuir al mismo, el cual es una muy pequeña parte de todo lo que implica la implantación de una LPS.

PARTE I

Esta parte introduce al lector al área de las LPS, describiendo los conceptos fundamentales, sus beneficios y motivaciones. Así mismo se espera que al finalizar esta parte, el lector tenga los fundamentos necesarios para entender el trabajo de investigación realizado.

Esta primera parte se divide en dos capítulos: el capítulo 2 explica los conceptos teóricos de las LPS mientras que el capítulo 3 se basa específicamente al área de Ingeniería de Requerimientos para Líneas de Productos.

2. Fundamentos de la Ingeniería de Líneas de Productos de Software

Si bien la revolución industrial inició el movimiento de producción en masa, no fue sino con Henry Ford cuando creó las líneas de producción. Esta nueva revolución "tecnológica" propicio la producción de productos en forma masiva, pues anteriormente todo se producía de manera artesanal imposibilitando satisfacer el mercado en grandes cantidades. De esto parten los fundamentos de una LPS, en la producción de software en forma masiva (o serial) lo cual da la capacidad a una organización de satisfacer las demandas del mercado, entre otros beneficios significativos.

2.1. Adaptación de productos de forma masiva

Sin embargo, las líneas de producción de Ford, tenían la desventaja de que no podía darse la diversificación entre los productos generados debido a que todos eran ensamblados de la misma manera. Esto evolucionó con la necesidad de que las personas empezaban a exigir productos personalizados para satisfacer sus necesidades individuales. Para el caso de la industria automotriz a pesar de que la línea de producción fuera para un modelo particular, cada auto producido de ese modelo podría tener variantes uno a otro como lo son: el color, el tamaño del motor, número de puertas, etc. Esto propició a tener una familia de productos que comparten características similares pero conservando algunas características particulares también llamado variabilidad.

De lo anterior surgieron las familias de productos que hasta hace algunos años la Ingeniería de Software tomó como filosofía para la producción de software de manera "masiva". No obstante, a pesar de que las líneas de productos en el área de software son un área "reciente" pues inició formalmente no hace más de dos décadas, existen indicios de los años 70's cuando Parnas en [2] mencionó algunos de los beneficios de tener productos compartiendo similitudes entre ellos.

2.2. Líneas de Productos de Software

Aunque en sus inicios, en la Unión Europea se les conocía como Familias de Productos de Software mientras que en América tenían ya el término actual, el cual es por el que son conocidas. Según el Instituto de Ingeniería de Software de la Universidad Carnegie Mellon (SEI por sus siglas en inglés) una LPS es definida como se presenta en [3]:

"Una Línea de Productos de Software es un conjunto de sistemas de software compartiendo un conjunto de características comunes y administradas que satisface las necesidades específicas de un segmento de mercado particular o misión y que son desarrolladas de forma prescrita a partir de un conjunto común de activos clave (core assets de ahora en adelante)."

Como lo dice la definición, una LPS consiste en un conjunto de core assets para producir sistemas de software que comparten un conjunto de características (similitudes) pero al mismo tiempo contienen características propias de cada uno (variabilidad). Para proporcionar las características individuales a cada miembro de la familia de productos la adaptación masiva debe ser lo suficientemente flexible para satisfacer tal necesidad, pues además el número distinto de miembros está limitado a una serie de restricciones las cuales deben ser satisfechas para tal motivo.

En una LPS lo que se propicia es una reutilización de forma planeada a un alto nivel desarrollando los core assets que servirán como base para la generación de cada miembro de la familia (como se explica en [4]). Para esto, las actividades esenciales a llevar a cabo en una LPS son principalmente: a) una etapa de Ingeniería de Dominios en la cual se desarrollan los core assets; b)

la etapa de Ingeniería de Aplicaciones donde se generan los sistemas que son miembros de la familia es análoga a la de desarrollo individual de sistemas de software pero con la diferencia que se desarrollan a partir de los core assets y se toma en cuenta el modelo de variabilidad para cada miembro; y c) las actividades necesarias para orquestar las dos actividades anteriores ya que es necesario un proceso con el cual llevarlas a cabo. Las dos primeras actividades (Ingeniería de Dominios e Ingeniería de Aplicaciones) son compuestas por las fases tradicionales del modelo de cascada pero con objetivos distintos como ya se mencionó.

En este trabajo de investigación solo se abarca una de las fases de la Ingeniería de Dominios dentro de la actividad de generación de los core assets, dicha fase es la Ingeniería de Requerimientos de la cual se obtuvieron los casos de uso comunes para el conjunto de sistemas analizados pertenecientes a la familia. En [3] se describen más exhaustivamente estas tres actividades.

2.3. Enfoques de adopción de una LPS

Se han planteado varios enfoques como estrategia para realizar la adopción de una línea de productos, entre ellos algunos tienen nombre distinto de acuerdo al “framework” o metodología en la que están contenidos (se hablará de tales “frameworks” en el capítulo 4) pero consisten en la misma estrategia. Para este caso, se explicarán los modelos de adopción identificados en [5] donde se exponen unos puntos de vista encontrados entre Paul Clements del SEI y Charles Krueger CEO de BigLever Software. Estos enfoques son:

Proactivo. Este enfoque funciona cuando la implantación de la LPS se hace desde cero, es decir, que aún no se cuenta con sistemas de software que pertenecerán a la familia. Este enfoque es similar al modelo de cascada en el aspecto de que se deben realizar las etapas de requerimientos o análisis, diseño y arquitectura, implementación y pruebas a partir de la visión que se tenga en ese momento acerca de las características similares y variables que tendrán los productos que serán miembros de la familia. Debido a la complejidad y al enorme esfuerzo que demanda este enfoque, es apropiado para las empresas que tienen una visión demasiado clara de los productos que conformarán la familia, además de que tienen niveles de madurez para predecir con gran certeza tal proceso y cuentan con los recursos económicos y humanos suficientes para realizar dicha inversión.

Extractivo. Este enfoque inicia con la selección de uno o más sistemas ya existentes que se dice serán parte de la familia de productos y efectuando un tipo de ingeniería inversa para extraer los artefactos de software comunes entre ellos para establecerlos como core assets y modelar la variabilidad que entre ellos existen. Este enfoque es ad-hoc para organizaciones que quieren efectuar la transición de una manera rápida de desarrollar sistemas individuales a generar sistemas a partir de una LPS.

Reactivo. Este enfoque toma la esencia del proceso de espiral o de los métodos ágiles para efectuar la transición poco a poco. Para esto se realizan los pasos como en el enfoque proactivo (requerimientos, arquitectura, implementación, etc) pero para cada ciclo o espiral, de esta manera se van a eliminando riesgos y se va aclarando la visión de las similitudes y variabilidades de los productos que serán miembros de la familia. Este enfoque es adecuado para organizaciones que mantienen planes de trabajo agresivos y no pueden dedicar los recursos humanos suficientes para la adopción y solo pueden disponer de un subconjunto de ellos.

2.4. Beneficios de una LPS

La implantación de una LPS no es una tarea sencilla, sino que requiere un gran esfuerzo y sobre todo una gran inversión para su éxito, sin embargo los beneficios obtenidos motivan a realizar dicho esfuerzo. Los beneficios de un proyecto de mejora de procesos de software (SPI por sus siglas en inglés) han sido presentados y documentados ([6], [7] y [8]) y se dice que el retorno de inversión (ROI por sus siglas en inglés) de dichos proyectos van en un rango de 5:1 hasta 8:1. En un caso de estudio de la empresa Cummins Engine Inc. (presentado en [3]) se adoptó una LPS después de haber realizado un proyecto de mejora de procesos y reportan que la mejora de procesos por sí sola tuvo un radio de beneficio-costo entre 2:1 y 3:1; y la implantación de la LPS en adición a la mejora de procesos arrojo un beneficio-costo de 10:1 lo cual muestra evidencia de que si un SPI vale la pena, una LPS vale aún más.

Existen muchos beneficios que una LPS puede proporcionar, destacando los siguientes entre los más relevantes:

- Obtención de las ganancias de una producción a gran escala. Como se mencionó en la adaptación masiva, una LPS habilita a producir sistemas de software a partir de una plataforma de artefactos comunes de manera que los productos solo necesiten ser ensamblados en vez de ser desarrollados individualmente.

- Mejora la capacidad para el "Time-to-market". Muchas veces las demandas del mercado exigen tener estrategias de mercadotecnia que implican tener productos o implementar ideas innovadoras antes de cierta fecha. Bajo un esquema de desarrollo de sistemas individualmente estas estrategias son demasiado difíciles de ejecutar haciendo perder a una organización la oportunidad de crecer en un mercado cambiante pues como se dice por ahí "el que pega primero pega dos veces". Con una LPS, una organización puede tomar el reto de establecer estrategias de mercadotecnia agresivas ya que cuentan con la capacidad de generar productos en el tiempo establecido. En la figura 2.1 se muestra una gráfica que representa el tiempo de desarrollo de sistemas de software con y sin una LPS donde el eje horizontal representa el número de sistemas distintos desarrollados y el eje vertical el tiempo de liberación en el mercado; como se puede ver, al inicio el tiempo desarrollo de los core assets de la LPS es muchísimo mayor que el desarrollo de los primeros sistemas individuales, sin embargo conforme se empiezan a desarrollar más sistemas distintos el tiempo de liberación de sistemas en una LPS se reduce radicalmente en comparación al desarrollo individual, lo cual demuestra que a un mediano plazo los beneficios serán mucho mayores pues como se mencionó, una organización se puede comprometer a liberar cierto producto de su LPS en menos de la mitad del tiempo que le llevaría desarrollarlo individualmente.

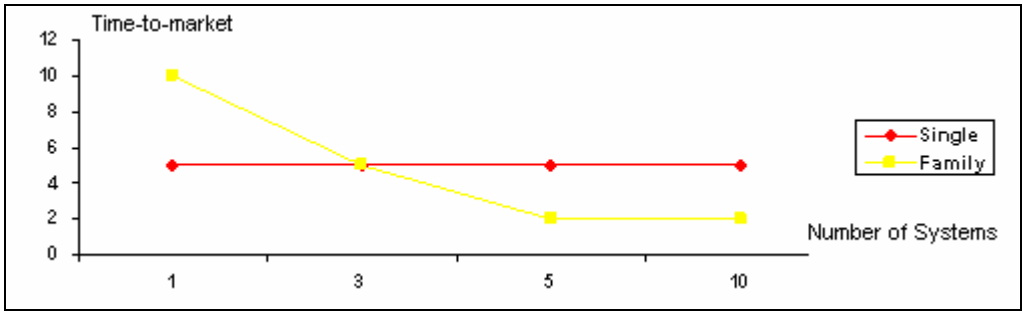

Figura 2.1. "Time-to-market" con y sin una Línea de Productos de Software.

- Reducción de costos de desarrollo. Con una LPS los costos de desarrollo de los sistemas miembros de la familia están muy por debajo de los costos de desarrollo de sistemas individualmente. Al igual que en el "time-to-market" al inicio, el desarrollo de los core assets

tiene mayor costo que desarrollar un sistema individualmente, sin embargo una vez obtenidos estos core assets el costo disminuye. Como se muestra en la figura 2.2, donde el eje horizontal representa el número de sistemas distintos a producir y el eje vertical representa los costos de desarrollo acumulados para los sistemas producidos, se puede decir que la taza de incremento en el costo acumulado es mucho menor en una LPS que un sistemas individuales (es decir que la derivada de la función de costo acumulado de una LPS es mucho menor que la derivada de la función de costo acumulado para sistemas individuales).

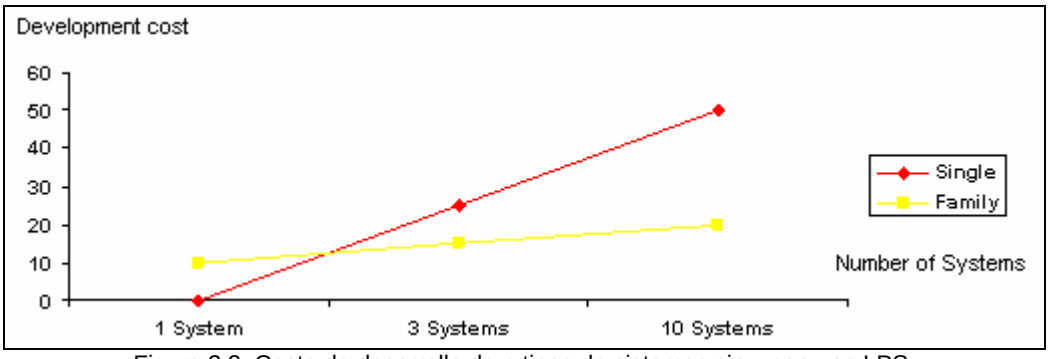

Figura 2.2. Costo de desarrollo de n tipos de sistemas sin y con una LPS.

- Mejora de la calidad de los productos. Desde que el desarrollo de los core assets es similar al desarrollo de los artefactos para un sistema individual, dichos assets son probados constantemente por cada producto generado, y por formar parte de todos los productos de la familia se puede decir que la calidad es más fácil y fielmente asegurada en todos ellos.

- Logro de las metas de reutilización de la organización. Dado que las LPS se basan en la reutilización de artefactos para un conjunto de sistemas, la reutilización planeada es obtenida al desarrollar la plataforma de core assets.

- Reducción de la necesidad de contratar nuevo personal. Desde que el proceso de generación de los productos de software se automatiza cada vez más, la necesidad de contratar gente disminuye debido a que dichos productos son generados cada vez con menos ingenieros de software, lo cual de alguna manera ayuda a disminuir los costos a la organización.

- Incremento de la satisfacción del cliente. Por varios de los beneficios mencionados anteriormente, los clientes y usuarios de los sistemas generados con la LPS obtienen productos de software de mucha mayor calidad y a precios muchísimo más bajos que los que obtendrían con un proveedor que no cuenta con una LPS.

Existen otros beneficios no menos importantes, sin embargo los que aquí se presentaron son los más significativos. Si se desea obtener más información acerca de estos otros beneficios se recomienda consultar el capítulo 2 en [3].

3. Procesos y Técnicas de Ingeniería de Requerimientos

En un proceso típico de Ingeniería de Requerimientos utilizado en el desarrollo de un producto individual hay cinco etapas [19]. Estas etapas son: captura, análisis, especificación, verificación y administración de requerimientos. En el desarrollo de una línea de productos, estas actividades son llevadas a cabo en etapas tempranas del Análisis de Dominio e Ingeniería. Los objetivos principales del Análisis de Dominio e Ingeniería son identificar los requerimientos y productos potenciales de la familia, de tal forma que se puedan analizar para diseñar e implementar un 'framework' de dominio reutilizable. Las cinco etapas mencionadas anteriormente pertenecientes a la Ingeniería de Requerimientos pueden ser utilizadas para identificar, analizar, especificar, verificar y administrar los requerimientos de una línea de productos, así como otros aspectos relacionados. Varias herramientas y técnicas han sido desarrolladas para cada una de las etapas de la Ingeniería de Requerimientos. La aplicabilidad de tales técnicas para el desarrollo de una línea de productos es evaluada en las siguientes secciones.

3.1. Técnicas para la captura de Requerimientos

En esta etapa los requerimientos para un sistema son recolectados a partir de varias fuentes [26]. Entrevistas, cuestionarios, análisis de mercado, 'Cooperative Requirements Capture' (CRC), y el Diseñador como Aprendiz, son algunas de las técnicas para captura de requerimientos. En el desarrollo de una línea de productos, las técnicas para captura de requerimientos utilizadas deben recolectar información acerca del objetivo del dominio, dominios relacionados, clientes potenciales y sus requerimientos. Técnicas tales como diagramas de contexto o mapeo de conceptos pueden ser utilizadas para analizar los conceptos del dominio. Las fuentes de información pueden incluir libros, expertos del dominio, expertos de mercado y productos existentes. Los clientes potenciales y 'stakeholders' involucrados deben ser identificados, y al mismo tiempo, sus requerimientos deben ser recolectados. Las características de productos existentes, así como las características de productos de la competencia también deben ser recolectadas. Se deben crear todos los artefactos relacionados con los métodos de desarrollo de Líneas de Productos, así como también se debe definir la especificación de las plantillas que serán utilizadas. Dependiendo de la situación, cualquiera de las técnicas siguientes pueden ser utilizadas.

3.1.1. Entrevistas

La entrevista es una técnica que involucra una discusión estructurada o no estructurada entre el ingeniero de requerimientos y el cliente [27]. Las entrevistas por si solas no son suficientes para la captura de los requerimientos en el desarrollo de Líneas de Productos. Además de que existen diversos 'stakeholders' involucrados que serán beneficiados con el desarrollo de la Línea de Productos, también es necesario satisfacer los objetivos de negocio de la organización. Sin embargo, las entrevistas pueden proporcionar soporte para otras técnicas de captura de requerimientos. Después de analizar los conceptos del dominio tanto con expertos del dominio como con expertos de mercado, los clientes potenciales pueden ser entrevistados para capturar información relacionada acerca de tópicos particulares. Las entrevistas estructuradas pueden ser conducidas si el ingeniero de requerimientos posee buen conocimiento acerca de un concepto y quiere obtener repuestas claras de los clientes sobre preguntas concretas relacionadas con el dominio. Las entrevistas no estructuradas pueden ser utilizadas para comunicar aquellos requerimientos (comunes o variables) que hayan sido recolectados por los usuarios, de tal forma que se de un intercambio de información entre ingeniero de requerimientos y usuario.

3.1.2. Encuestas, Cuestionarios y Minería de Datos

En las encuestas y cuestionarios, se distribuyen listas de preguntas a diversas muestras de una población con el objetivo de recolectar sus respuestas [27]. Estas técnicas pueden ser auxiliares a otras técnicas para determinar características comunes y variables de una familia de productos. Diferentes 'stakeholders' pueden expresar puntos de vista distintos sobre un concepto. La diversidad en las respuestas de los usuarios es una forma excelente para identificar variaciones. Aplicando minería de datos, es posible encontrar patrones en los datos recolectados, los cuales pueden ser utilizados para bosquejar conclusiones importantes acerca de los requerimientos del cliente.

3.1.3. Cooperative Requirements Capture (CRC)

CRC es una sesión de grupo similar a JAD [27]. CRC puede ser una práctica útil en el desarrollo de Líneas de Productos dado que soporta el desarrollo de un producto genérico. Una sesión CRC no solamente incluye a los clientes y a los desarrolladores, también incluye a otros 'stakeholders' que de alguna forma se vean afectados con el desarrollo del proyecto. El proceso de CRC incluye actividades tales como: Identificación del Problema, Selección del Equipo y Exploración de Usuarios y sus Ambientes de Trabajo. En un enfoque de Líneas de Productos, estas actividades podrían corresponder al análisis de concepto del dominio e identificación de clientes potenciales y sus requerimientos. Actividades tales como identificación del alcance del sistema pueden ser extendidas para analizar características comunes y variables e identificar a potenciales miembros de la familia. CRC podría convertirse n una buena práctica de las Líneas de Productos sin muchas modificaciones a la técnica original, sin embargo, la selección de los 'stakeholders' representativos y la facilidad con la que transmitan el conocimiento acerca del dominio es importante para su éxito.

3.1.4. Análisis de Mercado

El proceso de analizar el mercado del producto para evaluar tendencias actuales y futuras es llamado análisis de mercado [27]. El análisis de mercado es un buen punto de comienzo tanto para el desarrollo de un producto individual como para el de una Línea de Productos. El análisis de productos rivales, sus características y tendencias de mercado actuales y futuras proporcionan pistas acerca de productos potenciales y sus características. El análisis de mercado puede revelar como podría cambiar la demanda del producto en los años venideros. Esto podría ayudar a la organización a hacer una mejor selección de los productos miembro, así como de sus características. Por ejemplo, si el mercado de análisis revela que la demanda para una característica en particular se incrementará en unos años, entonces la organización puede decidir seleccionar que dicha característica sea incluida en todos sus productos.

3.1.5. El diseñador como aprendiz

La técnica del diseñador como aprendiz puede ser utilizada en combinación con otras técnicas para la captura de requerimientos. Cuando el concepto del problema es nuevo o cuando existe una falta de expertise del dominio en la organización, los ingenieros pueden trabajar como aprendices en el sitio del cliente. Esto debería proporcionar al diseñador conocimiento práctico acerca de los requerimientos del usuario. Utilizando este conocimiento como punto de partida, los diseñadores pueden analizar estos requerimientos para identificar las características comunes y variables. Esto debería también ayudar a los diseñadores a identificar clientes y productos potenciales de la familia.

3.2. Análisis de Requerimientos

En la etapa de Análisis de Requerimientos, los requerimientos recolectados en la etapa de captura son analizados y posteriormente refinados. Análisis Estructurado, Análisis Orientado a Objetos,

'Joint Application Design' y QFD son algunas de las técnicas utilizadas para el análisis de requerimientos. En el desarrollo de una Línea de Productos, la información recolectada durante la captura, puede ser analizada para identificar características comunes y variables, para determinar el alcance y para modelar la familia. Las características comunes y variables pueden ser identificadas ya sea utilizando técnicas específicas de Líneas de Productos o algunas otras técnicas tales como el modelado de características en FODA o el 'Commonality Analysis' de FAST. Una vez que las características comunes y variables han sido determinadas se identifican a los miembros potenciales de la familia. Posteriormente, haciendo uso de los detalles obtenidos, se analiza la factibilidad del desarrollo de la familia de productos. Técnicas similares a 'Product Map' o QFD pueden ser utilizadas para este análisis. Para la realización de dicho análisis es importante considerar los siguientes factores: ventaja en el mercado, prioridad del cliente, ventaja competitiva y contribución al dominio. Basándose en el resultado de este análisis, se seleccionan los productos y las características que serán incluidas. Posteriormente, los productos y características seleccionados deben ser modelados. El modelado proporciona mayor entendimiento y simplifica el diseño. Dependiendo del enfoque de la Línea de Productos utilizado, se deben seleccionar las técnicas de modelado adecuadas. El campo de aplicación de varias técnicas de análisis de requerimientos es descrito en las siguientes secciones.

3.2.1. Análisis Estructurado

Análisis Estructurado es un enfoque de análisis de requerimientos orientado a funciones [27]. Las técnicas de análisis estructurado pueden ser utilizadas por una Línea de Productos de forma similar a como son utilizadas para el desarrollo de productos individuales. Algunas notaciones especiales pueden ser utilizadas para representar a los requerimientos comunes y variables de una Línea de Productos de Software. Por ejemplo, en un diagrama de flujo de datos, las tareas correspondientes a requerimientos comunes podrían ser representadas usando líneas sólidas, mientras que las tareas alternativas o variables podrían ser representadas utilizando líneas punteadas. Sin embargo, los enfoques orientados a funciones no proporcionan un soporte robusto para la reutilización (por ejemplo, no proporciona soporte para la encapsulación). A pesar de que algunas técnicas como las utilizadas con los diagramas de entidad-relación, son de ayuda para identificar a las entidades del dominio, no existe una encapsulación precisa de los conceptos en objetos. El Análisis Estructurado tampoco soporta el diseño y desarrollo basado en componentes, además de que los sistemas desarrollados utilizando programación estructurada tienden a ser más complejos de administrar.

3.2.2. Análisis Orientado a Objetos (OOA)

Las técnicas de Análisis Orientado a Objetos son técnicas basadas en el concepto de Programación Orientada a Objetos [27]. OOA también puede ser utilizada para el análisis de requerimientos de una Línea de Productos. Algunos métodos de desarrollo de Líneas de Productos, como Sherlock y KobrA utilizan OOA para realizar el análisis. En ambos se utilizan diagramas de casos de uso y diagramas de clases los cuales son usados para el modelado de requerimientos. Las entidades correspondientes a requerimientos comunes son representadas con líneas sólidas y aquellas correspondientes a requerimientos variables son representadas con líneas punteadas. En los enfoques orientados a objetos, los requerimientos variables pueden ser fácilmente representados haciendo uso del concepto de herencia. Los puntos de variación y sus posibles valores pueden ser representados usando las relaciones de super-clase y sub-clase.
La mayor ventaja de la programación orientada a objetos es que proporciona soporte para la reutilización de componentes. Adicionalmente, la encapsulación de los atributos y comportamiento de los objetos soporta un diseño y desarrollo modularizado. A los componentes pequeños es más fácil proporcionarles mantenimiento que a un sistema orientado a funciones, el cual tiene toda su funcionalidad en un solo bloque [28].

Con el objetivo de reducir el costo y tiempo de mercado, una familia de productos debería tener la mayoría de sus artefactos reutilizables a lo largo de la familia. Es por esto, que los enfoques de líneas de productos deberían suportar la reutilización. Desde este punto de vista, el análisis orientado a objetos es un enfoque útil en el análisis de requerimientos para una línea de productos.

3.2.3. Joint Application Design (JAD)

JAD es un enfoque de sesiones en grupo el cual involucra a los usuarios en el diseño del sistema [27]. Tomando en cuenta que el desarrollo de una Línea de Productos involucra varios 'stakeholders', un enfoque como JAD es bueno para analizar los requerimientos de una Línea de Productos.
Una sesión JAD puede ser conducida de una forma similar a como se realiza para un producto individual. El líder de sesión puede guiar la sesión, sin embargo es importante, identificar a la mayor cantidad de usuarios representativos. Actividades como la definición de requerimientos de alto nivel y definición del alcance del sistema, pueden ser extendidas para identificar los requerimientos comunes y variables, así como también el alcance de la familia. Todos los 'stakeholders' pueden participar en el proceso de toma de decisiones. El enfoque JAD original utiliza técnicas de Análisis Estructurado para el diseño del sistema, sin embargo, como se ha mencionado anteriormente, estas técnicas no son lo más adecuadas para el desarrollo de Líneas de Productos [27]. En la medida de lo posible, se deberían utilizar técnicas OOA en lugar de técnicas de Análisis Estructurado.

3.2.4. Quality Function Deployment (QFD)

QFD es un enfoque para traducir los requerimientos del cliente hacia requerimientos técnicos apropiados [27]. Además de las asociar características técnicas con los requerimientos del cliente, QFD considera ciertas características de los productos de la competencia, prioridades de los clientes y la correlación entre características. En una familia de productos existe más de un producto que será analizado. En este caso QFD debe ser modificado para incluir a todos los productos de la familia. Si es necesario QFD puede ser aplicado a cada miembro de la familia por separado. En tales casos, el manejo de requerimientos comunes resultará complicado, sin embargo, analizando apropiadamente los resultados para cada producto, es posible determinar el alcance y las características de la familia. PuLSE tiene una técnica similar llamada 'Product Map' para determinar el alcance de una Línea de Productos. En 'Producto Map', todos los productos y sus características son relacionados mediante el uso de una matriz. Los productos de la competencia también son considerados en esta matriz. Basándose en una evaluación subjetiva de relevancia y prioridades se seleccionan a los potenciales miembros de la familia y sus requerimientos.

3.2.5. Diseño Participativo

El enfoque de Diseño Participativo permite a los diseñadores y usuarios trabajar de forma conjunta. Ambas partes aprenden una de otra [27]. Como se ha mencionado, en una Línea de Productos existen más de un producto y un cliente, y adicionalmente las prácticas de trabajo de cada cliente son diferentes, por lo tanto se requiere de una gran coordinación entre diversos diseñadores y clientes. Si los diseñadores y clientes trabajan de forma conjunta, es posible realizar un diseño que satisfaga las necesidades de todos los clientes. Además, el trabajo con diferentes clientes guía a la resolución de conflictos de requerimientos. Tomando en cuenta estos puntos, el Diseño Participativo es un buen enfoque para el diseño de Líneas de Productos en las cuales una gran diversidad de clientes está involucrada.

3.3. Especificación de Requerimientos

En la etapa de Especificación de Requerimientos, los requerimientos son documentados para futura referencia haciendo uso de determinados formatos o lenguajes de especificación [27]. La Especificación de Requerimientos de Software y Métodos Formales son dos técnicas utilizadas para la especificación de requerimientos. Para el desarrollo de Líneas de Productos, si el enfoque utilizado ya tiene un lenguaje para especificación del dominio, puede ser utilizado para especificar los requerimientos de la Línea de Productos. Las organizaciones también pueden utilizar sus propias plantillas de especificación o algunos documentos estandarizados SRS. Si la organización ya cuenta con un lenguaje de especificación de dominio, entonces la labor de especificación se facilitará. En cambio, si se utilizan documentos SRS, se debe desarrollar un documento SRS para la familia y un documento SRS por cada miembro de la familia. Al momento de escribir el documento SRS de la familia, todos los requerimientos comunes y variables deben ser documentados. Todos los posibles valores para cada punto de variación también deben documentarse. Para un producto individual, el SRS debe contener requerimientos comunes, así como requerimientos específicos que únicamente le pertenezcan.

3.4. Verificación de Requerimientos

En esta etapa, los requerimientos son verificados para corroborar su completes, relevancia, precisión y aplicabilidad [19]. Revisiones Formales, Prototipos y Testing de Requerimientos son algunas de las técnicas utilizadas en esta etapa. Una familia de productos tiene más de un producto y la mayoría de los requerimientos son comunes a lo largo de la familia. Cualquier defecto o mala interpretación en dichos requerimientos debería afectar a la familia entera. Hay requerimientos que son específicos a cada producto. Es por esto, que la etapa de verificación es muy importante para una Línea de Productos. La verificación de requerimientos puede ser conducida de forma similar a como se realiza en el desarrollo de productos individuales. Todas las técnicas de verificación utilizadas para productos individuales también pueden ser aplicadas a las Líneas de Productos.

3.4.1. Revisión de Requerimientos

Las revisiones de requerimientos son técnicas que comprueban la completes, relevancia y precisión de los requerimientos [26]. Pueden ser formales o informales. Las revisiones formales incluyen una sesión grupal para verificar los requerimientos. Las revisiones informales involucran una discusión entre el ingeniero de requerimientos y el cliente. El equipo de revisión debe incluir expertos del dominio, ingenieros de requerimientos, clientes y 'stakeholders' involucrados. Estas revisiones pueden ser conducidas como reuniones grupales agilizadas por el ingeniero de requerimientos. Primero, la familia de productos (incluyendo los requerimientos comunes y variables) se considera para revisión. Todos los comentarios y cambios requeridos deben ser documentados. Posteriormente, se revisan los requerimientos para cada producto de la familia. Además, el proceso de revisión debería asegurar que los requerimientos de la familia y sus miembros están adecuadamente especificados.

Las revisiones informales, pueden ser conducidas entre el ingeniero de requerimientos y clientes para validar requerimientos específicos de un producto en particular.

3.4.2. Prototipos

El desarrollo de prototipos es una técnica mediante la cual los productos son implementados de forma parcial con el objetivo de aprender más acerca de ciertos problemas o para demostrar que ciertas características están trabajando como es debido [26]. Los requerimientos para una familia

de productos pueden ser verificados desarrollando prototipos. En este caso, el prototipo que representa las características comunes puede ser reutilizado para todos los miembros que constituyen la familia.

3.4.3. Pruebas de Requerimientos

El proceso en el cual se pone a prueba el producto contra cada uno de los requerimientos que debiera satisfacer, es llamado Testing de Requerimientos [27]. Los requerimientos de una familia de productos pueden ser verificados definiendo casos de prueba para cada uno de los requerimientos. Al momento de definir los casos de prueba, es posible descubrir algunos defectos tempranamente. En el caso del desarrollo de Líneas de Productos, los casos de prueba deben ser definidos tanto para los requerimientos comunes como para los variables.

3.5. Administración de Requerimientos

La administración de requerimientos asegura que todas las actividades de la Ingeniería de Requerimientos están bien organizadas y que todos los productos finales están documentados adecuadamente de tal forma que se les puede dar seguimiento con facilidad. En las Líneas de Productos, la administración de requerimientos es mucho más complicada debido a que existen varios productos y actividades relacionadas. En la etapa de captura de requerimientos, la administración de requerimientos tiene que documentar toda la información recolectada y debe asegurarse que esté disponible sin ningún problema. También, debe asegurar que todos los procesos y productos a ser utilizados en otras etapas hayan sido definidos. La administración debe asegurar que las peticiones de cambios son manejadas apropiadamente; así como también, debe verificar que existen mecanismos adecuados para mapear requerimientos y otros artefactos del dominio con la familia y sus productos. Las actividades de administración pueden ser llevadas a cabo con herramientas de soporte.

PARTE II

En esta parte se presentan los métodos para LPS utilizados en este trabajo. Primero se da una descripción del método FODA, seguido de una descripción del método Plus y por último se explican muy brevemente otros frameworks que han surgido en esta área.

El capítulo 4 trata del método FODA. Se dedicó un capítulo a dicho método debido a que fue la base para el modelado de características en que está basada una parte de este trabajo.

El capítulo 5 trata al método Plus, el cual es un método práctico para diseñar LPS basado en UML. Dicho método fue utilizado para la fase de Ingeniería de Requerimientos para la LPS presentado en este trabajo.

El capítulo 6 contiene breve descripciones de otros frameworks surgidos a la fecha.

4. FODA

Desde antes de existir el concepto de Líneas de Productos de Software, existía ya el de Análisis de Dominio (AD). El AD se enfocaba a analizar los aspectos de un dominio particular el cual podía consistir de un conjunto de sistemas relacionados. A partir de esto, surgieron muchos métodos de AD siendo el método FODA (Feature-Oriented Domain Análisis) el que marcó una pauta en el desarrollo del AD, pues a partir de él muchos investigadores empezaron a utilizar este método para extraer las similitudes y variabilidades de un dominio.

El principal objetivo de FODA es la identificación de características relevantes de sistemas de software que pertenecen a un dominio. Se dice que las características son aspectos del dominio que son visibles al usuario final. FODA define aspectos comunes y aspectos distintos entre los sistemas relacionados. FODA además, soporta la reutilización a nivel funcional y arquitectónico y como resultado del análisis de dominio se generan productos que representan la funcionalidad y arquitectura común de dicho dominio.

El método FODA se divide principalmente en tres fases: Análisis de Contexto, Modelado del Dominio y Modelado de la Arquitectura, en la figura 4.1 se muestra las fases de FODA con los productos que pueden ser producidos de dichas fases. A continuación se describe brevemente cada uno de ellos.

4.1. Análisis de Contexto

El objetivo principal de este paso es definir el alcance de un dominio del cual se producirán ciertos productos. En esta fase se identifica el alcance del dominio, a través de los resultados de la fase así como con otros factores tales como la disponibilidad de expertos del dominio, datos y restricciones del dominio. Como resultado final se obtiene un modelo de contexto el cual es compuesto de un conjunto de diagramas estructurales y de flujo de datos.

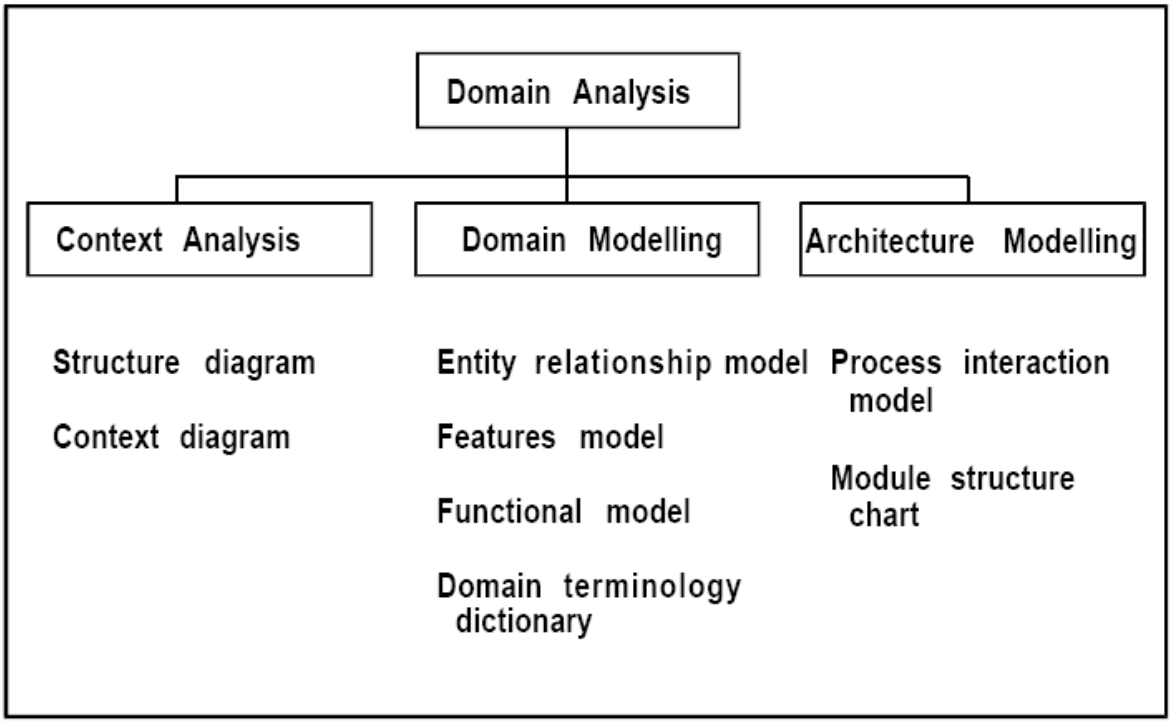

Figura 4.1. Fases en que se divide FODA y los artefactos producidos en ellas ([9]).

4.2. Modelado del Dominio

En esta fase se analizan las similitudes y diferencias de las aplicaciones que formarán el dominio para el cual se definió su alcance dentro de la fase de análisis de contexto y como resultado se producen un conjunto de modelos para varios aspectos de los sistemas del dominio. Esta fase se divide principalmente en tres actividades: análisis de características, modelado entidad-relación y análisis funcional; de dichas actividades la más relevante es el análisis de características.

4.2.1. Análisis de Características

Esta actividad tiene como propósito proveer al usuario las capacidades que tendrán los sistemas del dominio. Las características son atributos que afectan directamente a los usuarios finales pues son ellos los que decidirán las capacidades que dicho sistema contendrá.

Un modelo de características es representado por medio de un árbol como el mostrado en la figura 4.2 el cual representa el modelo características de una línea de automóviles, en el cual se muestran las características comunes que todo el conjunto de sistemas relacionados contendrá (también llamadas características mandatorias) tales como la transmisión y el caballaje. De igual manera que las características comunes, las características que son variables de sistema a sistema dentro de la familia son modeladas en dicho árbol. Dentro de los tipos de variabilidad en dicho modelo se encuentran las características opcionales las cuales pueden estar contenidas o no dentro de un sistema particular del dominio, para el ejemplo de la figura el aire acondicionado es una característica opcional pues un automóvil de la línea puede o no llevar dicha característica; las características alternativas significan que cierta característica puede ser especializada en sólo una de sus características alternativas, en el ejemplo la transmisión a pesar que es mandatoria debe

ser elegida de dos características alternativas transmisión manual ó transmisión automática. Por último, dentro del modelo de características también pueden ser modeladas ciertas reglas de composición las cuales definen algunas semánticas entre características opcionales y/o alternativas; existen dos principales siendo una de ellas de que la existencia de cierta característica variable implica que también exista otra característica variable, esta regla es llamada “requires” y como ejemplo se muestra en el modelo de la figura la regla de que si se elige la característica opcional de aire acondicionado se requiere que el caballaje sea mayor a 100; la regla que define lo contrario al “requires” se llama en FODA “mutually exclusive with” la cual significa que si se selecciona cierta característica variable implica que alguna otra característica variable no pueda ser seleccionada.

4.2.2. Modelado Entidad-Relación

Esta actividad no es contenida en este trabajo puesto que se refiere al clásico modelo entidad-relación muy utilizado en el área de bases de datos. Si se quiere profundizar en este modelo por favor vea [10].

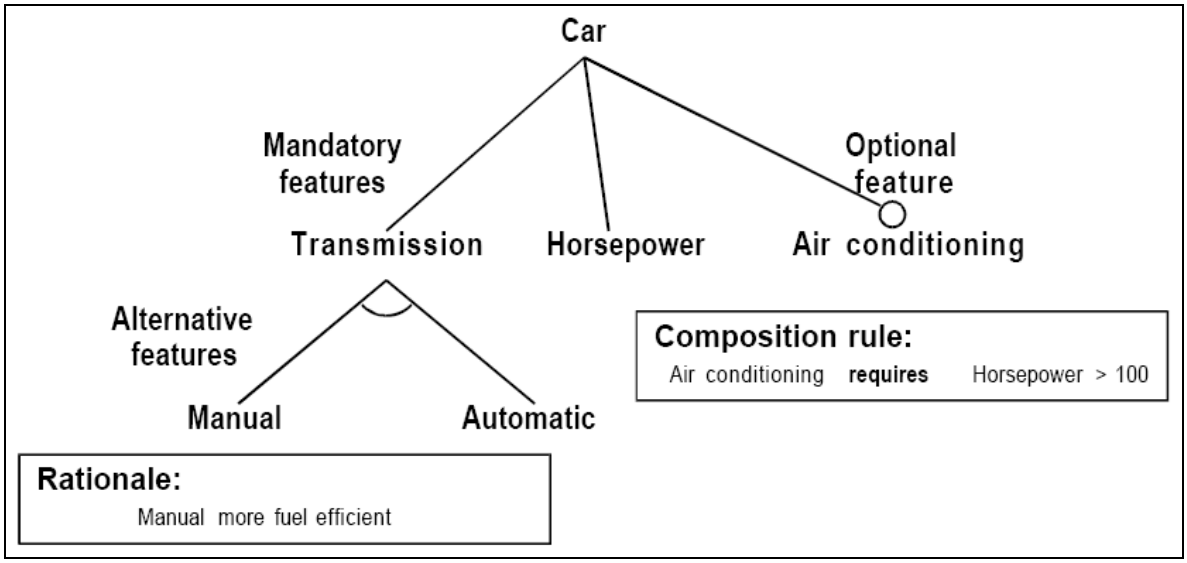

Figura 4.2. Modelo de características para una línea de automóviles ([9]).

4.2.3. Análisis Funcional

Esta actividad tiene como propósito la identificación de las funcionalidades comunes y diferentes en los sistemas del dominio. Como resultado se tiene un modelo funcional a partir del modelo de características y el modelo entidad-relación. Un de las notables ventajas de FODA sobre otros métodos de análisis de dominio es el concepto de parametrización, el cual es embebido dentro de un modelo funcional para especificar las características variables (opcionales y alternativas) que llevarán los sistemas del dominio. En la figura 4.3 se muestran los tipos de parametrización que FODA incorpora. El primer caso es por medio de desarrollo de los componentes por separados, refinando cada alternativa distinta para cierto producto de la familia; el segundo caso es por medio del desarrollo de un componente con parámetros en el cual la adaptación a cada alternativa sea dinámica; y como tercer caso es por medio de la herencia donde se define un componente general y se desarrolla cada alternativa como instancias de dicho componente general.

4.3. Modelado Arquitectónico

El propósito de esta fase es la de proveer la solución para el dominio que se está analizando con un enfoque a la reutilización. Dado que el modelado arquitectónico está fuera del alcance de este trabajo de investigación no se profundizará en esta fase.

Cabe señalar que el método FODA es el más utilizado en cuanto a modelado de características se refiere pues el árbol que de él se genera es utilizado por varios de los frameworks que hay para LPS. Se recomienda leer el reporte generado por el SEI [9] acerca de FODA.

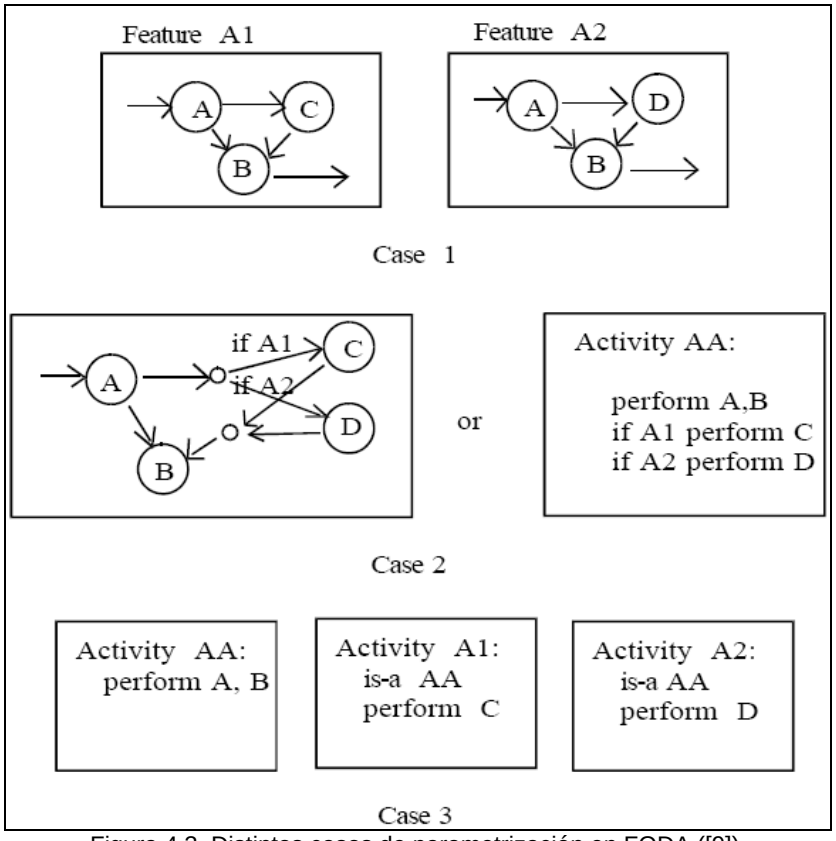

Figura 4.3. Distintos casos de parametrización en FODA ([9]).

5. PLUS (Product Line UML-Based Software Engineering)

PLUS es un método de diseño de Líneas de Productos de Software, el cual utiliza la notación del Lenguaje Unificado de Modelado (UML) [17].

PLUS extiende los métodos de modelado basados en UML utilizados en el desarrollo de sistemas individuales para adecuarlos al desarrollo de Líneas de Productos. Con PLUS, el objetivo es explícitamente modelar las características comunes y variables en una Línea de Productos de Software.

PLUS proporciona elementos para el modelado de requerimientos, modelado del análisis, modelado del diseño, y procesos relacionados con la Ingeniería de Aplicación. Sin embargo, en esta sección dedicada a PLUS, se le dará mayor importancia al modelado de requerimientos para Líneas de Productos de Software.

5.1. Modelado de Requerimientos

Durante la fase de modelado de requerimientos, se desarrolla un modelo en el cual los requerimientos funcionales del sistema son definidos en términos de actores y casos de uso. Para cada caso de uso se desarrolla una descripción en prosa. Si los requerimientos no han sido bien entendidos, se puede elaborar un prototipo desechable para que ayude a clarificar los requerimientos.
El modelado de requerimientos consta de los siguientes elementos:

- Determinar el alcance de la Línea de Productos. Propósito: Se determina a alto nivel la funcionalidad, el grado de aspectos comunes y variables, y el probable número de miembros de la Línea de Productos.
- Modelado de Casos de Uso. Propósito: modelar el aspecto común y variable en el modelo de casos de uso. Para este propósito, PLUS proporciona un enfoque para modelado de casos de uso ‘kernel’, ‘optional’ y ‘alternative’, así como también un enfoque para modelar puntos de variación en casos de uso.
- Modelado de Características. Propósito: modelar las características de Líneas de Productos. El modelado de características es un concepto clave en las Líneas de Productos de Software. PLUS proporciona un enfoque para modelar características comunes, opcionales y alternativas; un enfoque para obtener como resultado el modelo de características a partir del modelo de casos de uso; y un enfoque para representar características con notación UML.

5.2. Proceso Evolutivo para el desarrollo de Líneas de Productos de Software

El proceso evolutivo para el desarrollo de Líneas de Productos de Software (ESPLEP) es el proceso mediante el cual PLUS es llevado a la práctica. Este proceso elimina la distinción tradicional entre desarrollo y mantenimiento de software. A cambio de eso, los sistemas evolucionan a través de diversas iteraciones. Por lo tanto, debido a que los nuevos sistemas, a menudo son derivados de los existentes, el proceso, toma una perspectiva que le permite el desarrollo de familias de productos.

ESPLEP consiste de dos procesos principales, los cuales son mostrados en la Figura 5.1.

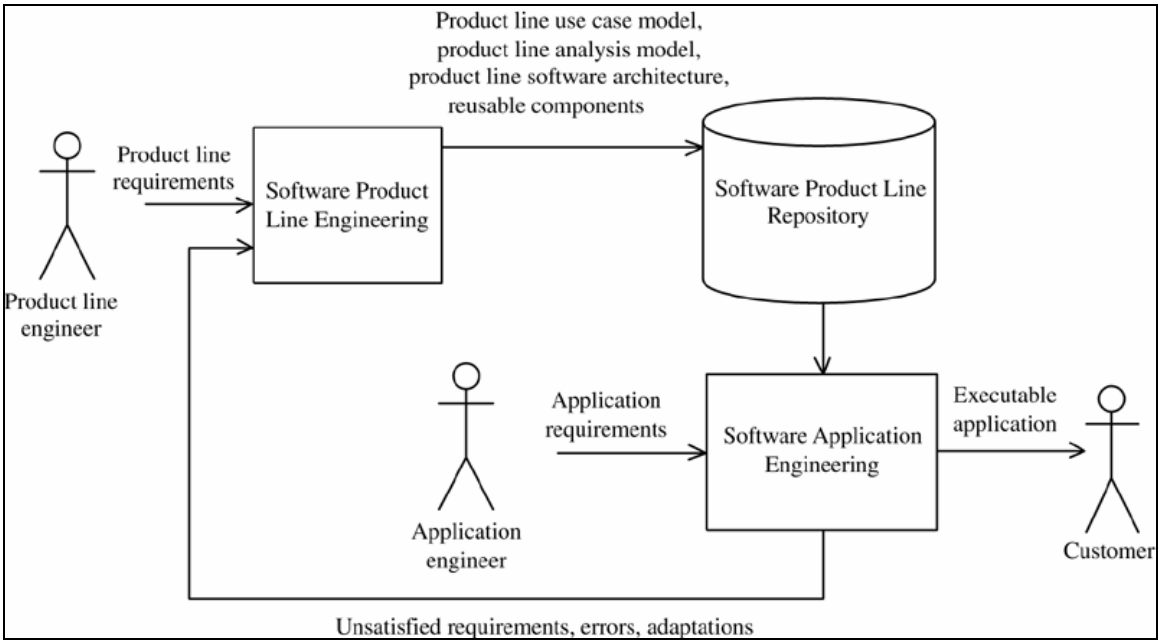

Figura 5.1. Procesos de ESPLEP [17].

Durante el proceso de Ingeniería de Líneas de Productos de Software, las partes común y variable en la Línea de Productos son analizadas a la luz de la totalidad de requerimientos de la Línea de Productos. Esta actividad consiste del desarrollo de un modelo de casos de uso, un modelo de análisis, un modelo de arquitectura y componentes reutilizables. También, se llevan cabo pruebas sobre los componentes desarrollados, así como también algunas configuraciones de la Línea de Productos. Todos los artefactos producidos son almacenados en un repositorio perteneciente a la Línea de Productos de Software.

En el proceso de Ingeniería de Aplicación, se desarrolla una aplicación individual la cual es miembro de la Línea de Productos. En lugar de comenzar desde cero, como usualmente se hace con sistemas individuales, los desarrolladores de la aplicación hacen uso de todos los artefactos desarrollados durante el proceso de Ingeniería de Líneas de Productos de Software. Dada la totalidad de los requerimientos de la aplicación individual, el modelo de casos de uso de la Línea de Productos es adaptado para extraer el modelo de casos de uso de la aplicación, así mismo la arquitectura de la Línea de Productos de Software es adaptada para extraer la arquitectura de la aplicación. Dada la arquitectura de la aplicación y los componentes apropiados del repositorio de la Línea de Productos, la aplicación ejecutable es desplegada.

5.3. Fases de Ingeniería de Líneas de Productos de Software

ESPLEP es un proceso de desarrollo de software altamente iterativo basado en el concepto de casos de uso. Durante el modelado de requerimientos, los requerimientos funcionales de la Línea de Productos son definidos en términos de actores y casos de uso. Durante el modelado de análisis, cada caso de la Línea de Productos es hecho para describir los objetos que participan en el caso de uso y sus interacciones. Durante el modelado de diseño, la arquitectura de software basada en componentes para la Línea de Productos es desarrollada. Las fases de ESPLEP para el proceso de Ingeniería de Líneas de Productos de Software son mostradas en la Figura 5.2. Todos los artefactos producidos en cada fase son almacenados en el repositorio de la Línea de Productos.

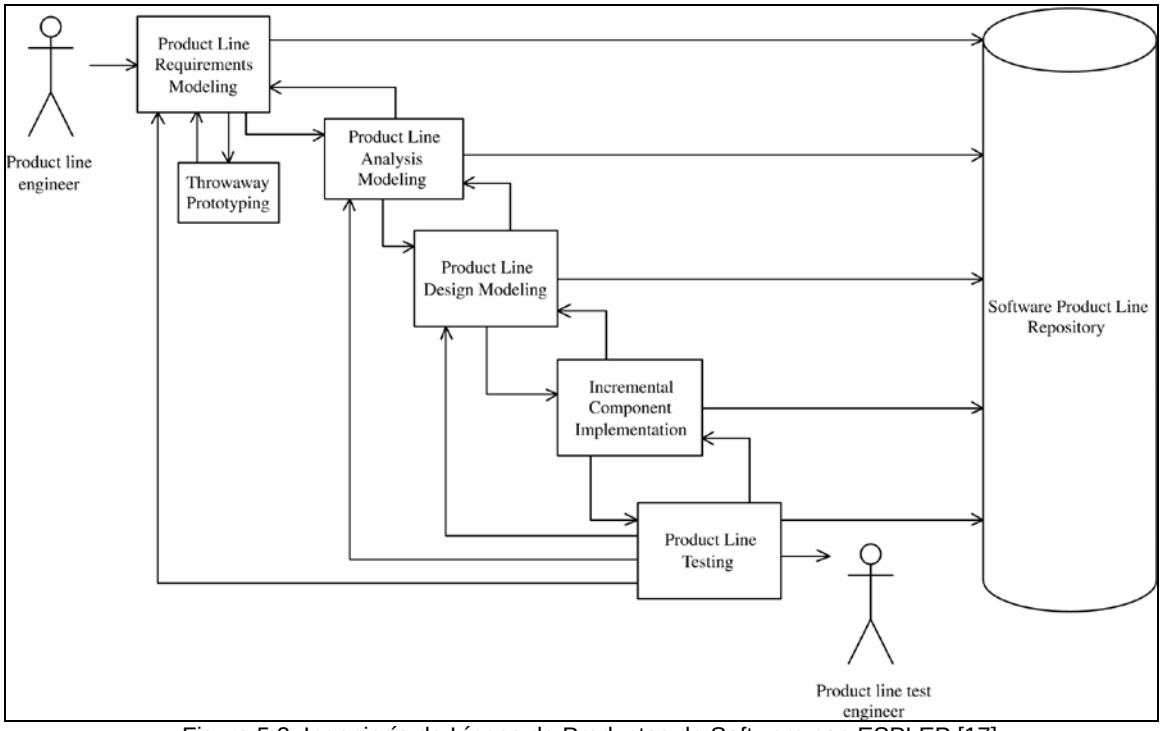

Figura 5.2. Ingeniería de Líneas de Productos de Software con ESPLEP [17].

5.3.1. Modelado de requerimientos de la Línea de Productos

Durante la fase de modelado de requerimientos de la Línea de Productos, un modelo de requerimientos consistente de un modelo de casos de uso y un modelo de características es desarrollado. Al igual que con sistemas individuales, el modelo de casos de uso define los requerimientos funcionales de la Línea de Productos en términos de actores y casos de uso. Sin embargo, el modelo de casos de uso para una Línea de Productos necesita ser extendido para modelar los aspectos comunes y variables en la Línea de Productos a través del desarrollo de los casos de uso 'kernel', 'optional' y 'alternative'. También se crea una descripción textual de cada caso de uso.
Para las Líneas de Productos de Software, un modelo de características es desarrollado. Una característica es un requerimiento o característica que es proporcionado por uno o más miembros de la Línea de Productos. En particular, las características son atributos que son usados para diferenciar entre los miembros de la Línea de Productos y por lo tanto, para definir la funcionalidad común y variable de una Línea de Productos de Software.

Las fases siguientes al modelado de requerimientos de la Línea de Productos no se abordarán en este documento debido a que se le proporcionará mayor importancia a las secciones concernientes con requerimientos del sistema.

5.4. Ingeniería de Aplicación de Software

Durante la Ingeniería de Aplicación de Software, la arquitectura de la Línea de Productos es adaptada y ajustada para obtener como resultado una aplicación de software dada, la cual es miembro de la Línea de Productos de Software. El crear la aplicación (derivación), involucra considerar la totalidad de los requerimientos de la aplicación individual, seleccionando aquellas características de la aplicación que concuerden con las características de la Línea de Productos; así mismo, las características seleccionadas son utilizadas para ajustar el modelo de casos de uso

de la Línea de Productos para dar como resultado el modelo de casos de uso de la aplicación; además, se ajusta la arquitectura de la Línea de Productos para obtener como resultado la arquitectura de la aplicación. Una vez dada la arquitectura de la aplicación, los componentes apropiados del repositorio de la Línea de Productos son instanciados, interconectados y desplegados. Una vista de alto nivel de la Ingeniería de Aplicación es mostrada en la Figura 5.3.

5.4.1. Fases en la Ingeniería de Aplicación de Software

Durante la Ingeniería de Aplicación de Software, se desarrolla una aplicación individual que es un miembro de la Línea de Productos de Software. Las fases del proceso de Ingeniería de Aplicación son mostradas en la Figura .

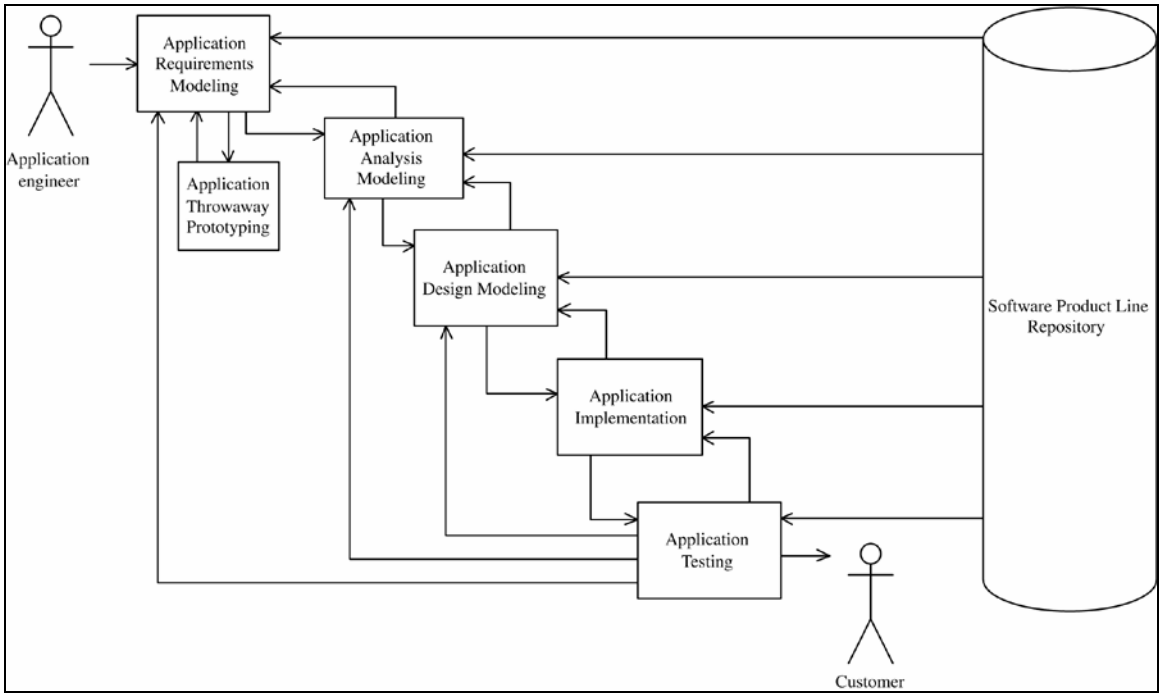

Figura 5.3. Fases de la Ingeniería de Aplicación [17].

5.4.1.1. Modelado de requerimientos de la Aplicación

Durante la fase de modelado de requerimientos de la aplicación, un modelo es desarrollado en el cual los requerimientos funcionales de la aplicación son definidos en términos de actores y casos de uso. Los requerimientos de la aplicación son buscados en el modelo de características de la Línea de Productos para determinar que características serán incorporadas a la aplicación. Una aplicación típica consiste de todas las características kernel (comunes) y algunas características opcionales y alternativas. Basándose en las características de la aplicación la tabla de dependencias características/casos de uso es analizada para determinar cuales casos de uso son necesarios para la aplicación y que variabilidad será insertada en los puntos de variación. Los casos de uso de la aplicación consistirán de todos los casos de uso 'kernel' de la Línea de Productos y una selección de casos de uso 'optional' y 'alternative'.

Las etapas posteriores al modelado de requerimientos de la aplicación no serán analizadas debido a que el presente documento se enfoca en los requerimientos.

5.5. Modelado de Líneas de Productos con UML

El campo de la reutilización de software ha evolucionado de componentes individuales hacia reutilización a gran escala con Líneas de Productos de Software. Los enfoque de modelado de software, el día de hoy, son ampliamente utilizados en el desarrollo de software y juegan un rol importante en las Líneas de Productos de Software.

PLUS es un método de diseño de software, el cual extiende los métodos de modelado basados en UML que son utilizados en sistemas individuales, para orientar a las Líneas de Productos de Software. Con PLUS, el objetivo es explícitamente modelar los aspectos comunes y variables en una Línea de Productos de Software.

5.6. Modelado de Casos de Uso para Líneas de Productos

En esta sección se muestra cómo el enfoque de Modelado de Casos de Uso puede ser extendido para modelar Líneas de Productos de Software.

Todos los requerimientos de software especificados para un sistema individual necesitan ser proporcionados por el sistema. Para una Línea de Productos de Software, el cual consiste de una familia de sistemas, únicamente algunos requerimientos son comunes para todos los miembros de la familia. Para especificar los requerimientos funcionales de la Línea de Productos es importante capturar aquellos requerimientos que son comunes para todos los miembros de la familia, así como los requerimientos ‘optional’ y ‘alternative’. Los requerimientos ‘optional’ son necesarios por únicamente algunos de los miembros de la Línea de Productos. Los requerimientos ‘alternative’ son manejados de manera diferente por distintos miembros de la Línea de Productos.

Al extender el enfoque de modelado de casos de uso, para especificar los requerimientos funcionales de Líneas de Productos de Software, es necesario definir diferentes tipos de casos de uso: ‘kernel’, los cuales son necesarios por todos los miembros de la Línea de productos; ‘optional’, los cuales son necesarios por solamente algunos miembros de la Línea de Productos; y ‘alternative’, donde diferentes casos de uso son necesarios para diferentes miembros de la Línea de Productos (estos casos de uso, son mutuamente exclusivos).

Los estereotipos UML son mecanismos de extensión estándar proporcionados por UML y utilizados para distinguir entre las diferentes clases de elementos de modelado Para los casos de uso de una Línea de Productos, los estereotipos «kernel», «optional», y «alternative» son utilizados para distinguir entre los casos de uso que siempre son requeridos, los casos de uso que algunas veces son requeridos y los casos de uso en los cuales se debe realizar una selección.

5.6.1. Modelando la variabilidad en Casos de Uso

Una forma de manejar la variabilidad de casos de uso en Líneas de Productos de Software es a través de los puntos de variación. Un punto de variación es una ubicación en un caso de uso donde un cambio puede tomar lugar.

Los puntos de variación en casos de uso pueden ser manejados de distintas formas, la que será utilizada en el presente documento será con las relaciones de ‘extend’ e ‘include’, tal como se describe en las siguientes secciones.

5.6.1.1. La relación de 'extend'

Un caso de uso puede volverse demasiado complejo si tiene muchas secuencias 'alternative', 'optional' y 'exceptional'. Una solución para este problema es dividir una secuencia de interacción 'alternative' u 'optional' en casos de uso separados. El propósito de este caso de uso nuevo es extender al caso de uso viejo, si es que las condiciones apropiadas se mantienen. El caso de uso que es extendido es llamado caso de uso base, y el caso de uso que hace la extensión es llamado caso de uso de extensión. Un ejemplo de esta relación es mostrado en la Figura 5.4.

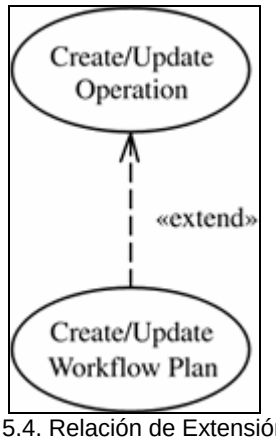

Figura 5.4. Relación de Extensión [17].

5.6.1.2. Puntos de Extensión en Líneas de Productos de Software

Los puntos de extensión son utilizados para especificar las ubicaciones precisas en el caso de uso base en las cuales las extensiones pueden ser agregadas. Un caso de uso de extensión puede ser extendido únicamente en estos puntos de extensión.

En una Línea de Productos de Software, un punto de extensión con un caso de uso de extensión puede ser utilizado para modelar una variación 'optional' (opcional) en la cual el caso de uso de extensión especifica la funcionalidad opcional. Si la condición de extensión es verdadera para un miembro de la línea de productos, entonces la funcionalidad opcional es proporcionada por ese miembro.

En Líneas de Productos de Software, un punto de extensión con múltiples casos de uso de extensión puede ser utilizado para modelar variaciones 'alternative', donde cada caso de extensión representa una alternativa diferente. Las condiciones de extensión son diseñadas tal que solamente una condición puede ser satisfecha, y por lo tanto un solo caso de uso de extensión seleccionado por cualquier miembro de la Línea de Productos. Un ejemplo de relación 'extend' y sus casos de uso de extensión es mostrado en la Figura 5.5**Figura** .

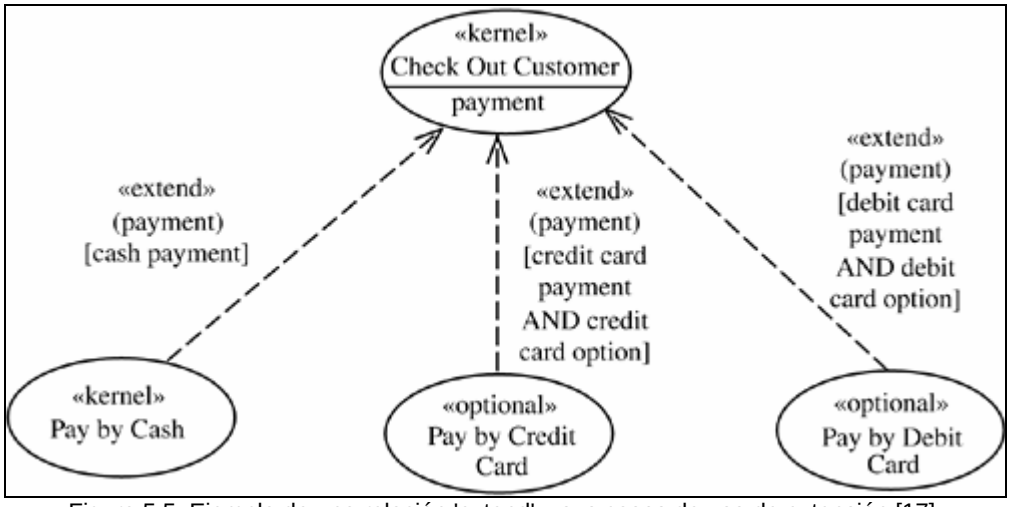

Figura 5.5. Ejemplo de una relación 'extend' y sus casos de uso de extensión [17].

5.6.1.3. La relación 'include'

La relación 'include' es utilizada en situaciones donde, después de que los casos de uso iniciales para una aplicación han sido desarrollados, se determina que secuencias de interacciones entre el actor y el sistema están presentes en varios casos de uso.

Estas secuencias de interacciones reflejan funcionalidad que es común a más de un caso de uso. Una secuencia común de interacciones puede ser extraída de varios de los casos de uso originales y pueden formar un nuevo caso de uso llamado caso de uso de inclusión. Un caso de uso de inclusión es usualmente abstracto, lo que quiere decir que no puede ser ejecutado por sí mismo. Un caso de uso abstracto debe ser ejecutado como parte de un caso de uso concreto (el cual es ejecutable).

La relación de 'include' puede ser utilizada para soportar casos de uso opcionales. El caso de uso de inclusión podría ser un caso de uso 'kernel'. Los casos de uso opcionales estarán "protegidos" por una condición de inclusión. Para un miembro dado de la Línea de Productos, si la condición de inclusión es satisfecha, entonces el caso de uso opcional estará disponible para ese miembro. Un ejemplo de un casos de uso 'kernel' y opcionales con relaciones de inclusión es mostrado en la Figura 5.6.

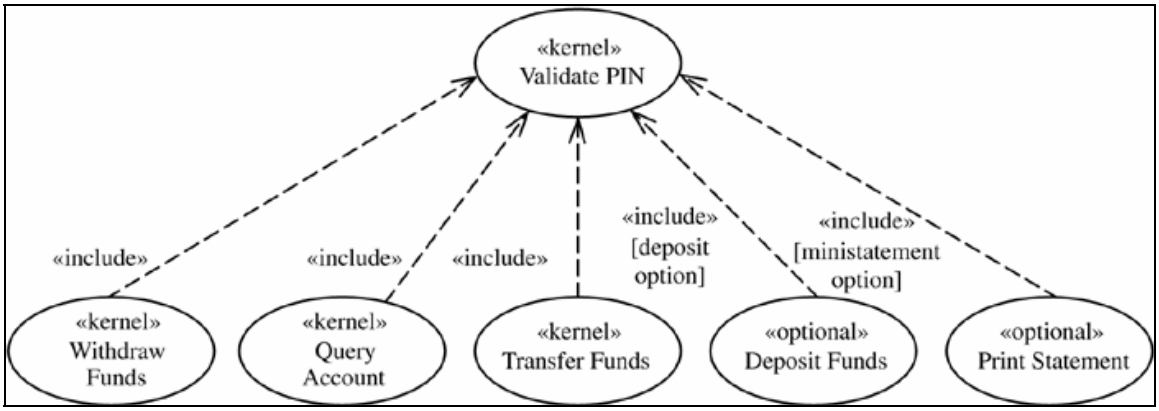

Figura 5.6 Ejemplo de casos de uso 'kernel' y opcionales con relaciones 'include' [17].

6. Otros métodos para LPS

A pesar de ser un área relativamente joven, la Ingeniería de LPS ha tenido grandes contribuciones, pues hasta la fecha han surgido varios “frameworks” para asistir la compleja tarea de adoptar una LPS. En este capítulo se describirán brevemente algunos de los más relevantes, entre ellos están:

- Software Product Line Practice Framework (SEI)
- FORM
- FAST
- KobrA
- Síntesis
- Pulse
- Sherlock
- Odyssey DE

6.1. Software Product Line Practice Framework

El SEI reunió en este framework las actividades y prácticas esenciales para producir los core assets y los productos así como para la orquestación del proceso. El framework describe las áreas prácticas específicas para Ingeniería de Software, Gestión Técnica y Gestión Organizacional. Un área práctica está formada por un conjunto de actividades que una organización debe dominar para poder adoptar la Ingeniería de LPS. Dentro del framework cada práctica contiene los siguientes puntos:

- Una descripción introductoria de la práctica
- Aspectos de la práctica que son peculiares a las líneas de productos
- El cómo esta práctica es aplicada al desarrollo de los core assets
- El cómo esta práctica es aplicada al desarrollo de los productos
- Prácticas específicas en esta área práctica
- Riesgos en esta práctica
- Referencias

Como se mencionó, son tres categorías en las que caen estas áreas prácticas del framework. A continuación se listarán las prácticas asociadas para cada una de estas categorías.

6.1.1. Áreas prácticas de Ingeniería de Software

Estas prácticas son necesarias para aplicar la tecnología necesaria para crear y evolucionar los core assets y los productos desarrollados. Estas áreas prácticas son:

- Definición de la Arquitectura
- Evaluación de la Arquitectura
- Desarrollo de Componentes
- Utilización de COTS
- Minería de artefactos existentes
- Ingeniería de Requerimientos
- Integración de Sistemas de Software
- Pruebas de Software
- Entendimiento de los Dominios Relevantes

6.1.2. Áreas prácticas de Administración Técnica

Éstas son prácticas administrativas para diseñar o ingeniar el desarrollo y la evolución de los core assets y los productos de la familia. Estas áreas prácticas son:

- Administración de la Configuración
- Colección de datos, métricas y seguimiento
- Análisis para crear/comprar/minar/encomendar
- Definición de Procesos
- Alcance
- Planeación Técnica
- Administración de Riesgos Técnicos
- Soporte de herramientas

6.1.3. Áreas prácticas de Administración Organizacional

Estas prácticas se refieren a los aspectos del negocio que son visibles a nivel organizacional y no a nivel proyecto. Las áreas prácticas son:

- Construcción del Caso de Negocio
- Administración de la Interfase con el Cliente
- Desarrollo de la Estrategia de Adquisición
- Financiamiento
- Puesta en marcha e Institucionalización
- Análisis del Mercado
- Operaciones
- Planeación Organizacional
- Administración de Riesgos Organizacionales
- Estructura de la Organización
- Predicción de Tecnología
- Capacitación

Debido a la limitación de este trabajo de investigación no se describen exhaustivamente cada una de las prácticas listadas, por lo que si el lector se interesa en profundizar en este framework puede referirse a [3].

6.2. FORM

El método FORM (Feature-Oriented Reuse Method) desarrollado en Pohang University of Science and Technology se enfoca principalmente al análisis y modelado de similitudes y diferencias en términos de características de las cuales se desarrolla la arquitectura y los componentes de la LPS. Este método extiende al método FODA con la principal diferencia en que FORM se basa en una perspectiva de mercadotecnia para realizar el análisis y diseño mencionados. FORM se divide principalmente en dos actividades: desarrollo de assets (que es lo mismo que la Ingeniería de Dominios) y el desarrollo de productos (análogo a la Ingeniería de Aplicaciones). Si se quiere profundizar más en este framework véase [11].

6.3. FAST

El método FAST (Family-Oriented Abstraction, Specification, and Translation) es un método desarrollado por David Weiss y que es utilizado en Lucent Technologies como parte de un proceso de ingeniería de dominios. FAST se basa en una técnica analítica conocida como análisis de similitudes para decidir cómo deberían ser los miembros de una familia o dominio. Dicho análisis de similitudes es llevado a cabo a través de un conjunto de reuniones con los expertos del dominio, donde cada miembro es experto en uno o más aspectos de la familia y el moderador debe ser un experto en FAST. El proceso de análisis de similitudes está organizado en las siguientes etapas: preparar, planear, analizar, cuantificar y revisar. Para mayores detalles acerca de este método lea [12] y [13].

6.4. KobrA

El método KobrA (Komponentenbasierte Anwendungsentwicklung) fue desarrollado como parte de un proyecto fundado por el ministerio de educación e investigación de Alemania con la colaboración del Fraunhofer Institute for Experimental Software Engineering (IESE) en Kaiserslautern Alemania, y otras tres organizaciones, siendo el IESE el principal responsable del proyecto. KobrA aprovecha los beneficios del desarrollo basado en componentes a través del ciclo de vida y permite que la mejora significativa de la reutilización de componentes. Para esto KobrA se basa en los enfoques de model-driven, UML para la representación de los componentes y una línea de productos para el desarrollo y evolución. Para profundizar en este método véase [14].

6.5. Synthesis

Synthesis proporciona un 'framework' para desarrollo de Líneas de Productos el cual puede ser personalizado de acuerdo al contexto organizacional en el que se esté utilizando [20]. Las organizaciones pueden adoptar Synthesis de forma total o parcial dependiendo del nivel de reutilización que deseen alcanzar. Las actividades de Synthesis se agrupan en dos fases principales: Ingeniería de Dominio e Ingeniería de Aplicación. En la Figura 6.1 se muestran las actividades involucradas en la fase de Ingeniería de Dominio. Las actividades iniciales tales como Definición del Dominio (Domain Definition), Especificación del Dominio (Domain Specification) y Verificación del Dominio (Domain Verification) son responsables de identificar y categorizar los requerimientos de la Línea de Productos, así como a los potenciales miembros de la familia. En la fase de Ingeniería de Aplicación es donde se desarrollan aplicaciones individuales.

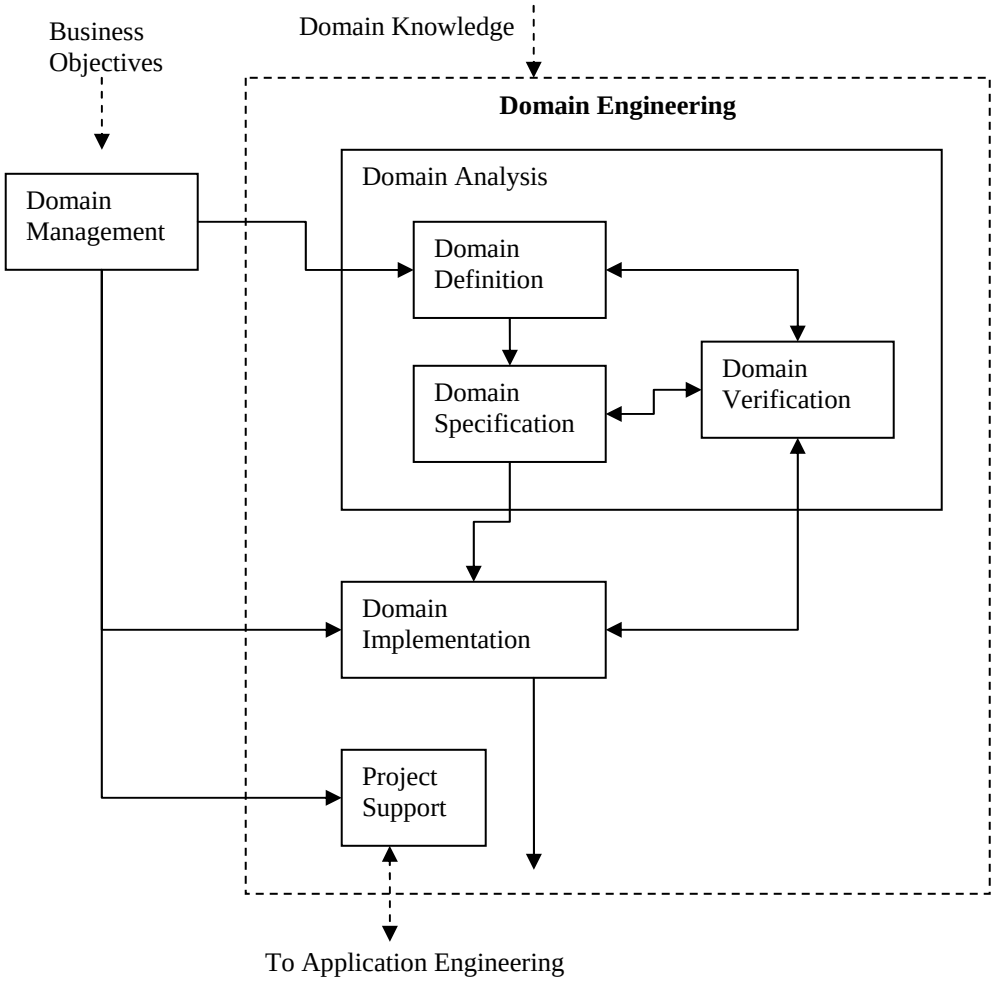

Figura 6.1. Actividades de la Ingeniería de Dominio en Síntesis [20].

Synthesis proporciona un 'framework' para las actividades de Ingeniería de Requerimientos de las Líneas de Productos desarrolladas bajo este método. Esto nos sugiere que las etapas de Definición del Dominio, Especificación del Dominio, Verificación del Dominio y Administración del Dominio corresponden a las actividades de Captura, Análisis, Especificación y Administración de la Ingeniería de Requerimientos; sin embargo, Synthesis proporciona poca información acerca de técnicas concretas que pudieran ser utilizadas para llevar a cabo las actividades de la Ingeniería de Requerimientos.

6.6. PuISE (Product Line Software Engineering)

PuLSE es un método para desarrollo de Líneas de Productos que se puede adaptar a diversos contextos empresariales [22]. PuISE tiene tres elementos básicos: Fases de Despliegue, Componentes Técnicos y Componentes de Soporte. Las Fases de Despliegue son pasos lógicos que describen las actividades necesarias para definir y desarrollar una familia productos. Estas actividades están clasificadas en tres etapas llamadas Inicialización de PuISE, Construcción de la Infraestructura PuISE y Uso de la Infraestructura PuISE. En la etapa de Inicialización, se define un

proceso PulSE, el cual es adaptado al dominio y contexto organizacional actual. En la etapa de Construcción de la Infraestructura PulSE se definen y modelan las características de la línea de productos, lo cual brinda las bases para el posterior desarrollo de una arquitectura reutilizable. En la etapa Uso de la Infraestructura PulSE, la arquitectura de la línea de productos es instanciada para el desarrollo de aplicaciones individuales. Los Componentes Técnicos incluyen el conocimiento técnico requerido para llevar a cabo varias actividades PulSE. Los Componentes de Soporte proporcionan las directrices requeridas para resolver cuestiones no técnicas tales como problemas organizacionales, evolución de procesos, entre otras.

Similar a Synthesis, PuLSE proporciona un 'framework' para desarrollo de Líneas de Productos de software. A pesar de que propone varias etapas para el desarrollo de una Línea de Productos, no tiene un proceso sistemático de Ingeniería de Requerimientos, lo cual permite a las organizaciones decidir qué técnicas y herramientas serán utilizadas para cada actividad. A pesar de esto, utiliza una técnica llamada 'Product Map', la cual liga las características de la línea de productos con productos miembros en la forma de una matriz. Usando una evaluación subjetiva sobre la importancia y beneficio de mercado, ciertas características y funciones de beneficio son calculadas. Estas funciones son entonces, evaluadas para escoger los requerimientos de la línea de productos así como los productos miembro. 'Product Map' es una excelente técnica que puede ser utilizada para identificar las características de una línea de productos y miembros potenciales. PuLSE proporciona poca información acerca de la especificación, verificación y seguimiento de requerimientos [18].

6.7. Sherlock

Sherlock es un método de Análisis de Dominio e Ingeniería orientado a objetos [24]. En Sherlock, a través de cinco pasos básicos se desarrolla un 'framework' de dominio. Estos pasos son: Definición del Dominio, Particularización del Dominio, Alcance del Dominio, Modelado del Dominio y Desarrollo del 'Framework' de Dominio. Durante la Definición del Dominio, Particularización del Dominio y Alcance del Dominio, se recolecta la información de fuentes diversas, se definen las características comunes y variables y se identifican a los potenciales miembros de la familia. En el paso de Modelado del Dominio, los requerimientos para cada miembro de la familia y los requerimientos comunes son analizados utilizando modelos orientados a objetos. En el paso de Desarrollo del 'Framework' de Dominio, un 'framework' de dominio es desarrollado utilizando varios modelos de proceso y diversos modelos arquitectónicos. El paso de Definición del Dominio es equivalente a la captura de requerimientos, los cuales son recolectados a partir de expertos del dominio, análisis de mercados y otras fuentes. El análisis de mercado es una técnica excelente para identificar las tendencias actuales y futuras debido a que esto tendrá gran influencia sobre las características y selección de productos de la familia. A pesar de que Sherlock tiene actividades de requerimientos, no utiliza alguna técnica específica para identificar requerimientos comunes y variables. Los requerimientos de la línea de productos y potenciales productos miembro, son escogidos acorde a las estrategias organizacionales. Una vez que los productos y los requerimientos han sido identificados, son analizados utilizando técnicas de análisis orientado a objetos. Sherlock proporciona herramientas de soporte para la administración de cada actividad.

A pesar de lo anterior, Sherlock no define técnicas concretas para la especificación, verificación y seguimiento de requerimientos.

6.8. Odyssey DE

Odyssey DE es una técnica de Análisis de Dominio e Ingeniería orientada a objetos, la cual utiliza técnicas de desarrollo de software basado en componentes [25]. Odyssey DE tienen cuatro fases

de desarrollo: Análisis de Viabilidad del Dominio, Análisis del Dominio, Diseño del Dominio e Implementación del Dominio. En el Análisis de Viabilidad del Dominio, se realiza un análisis de costo-beneficio para determinar la factibilidad del desarrollo de la Línea de Productos. En la fase de Análisis del Dominio, la familia de productos es descrita y su alcance es determinado. En las fases de Diseño e Implementación del Dominio, la arquitectura de la familia de productos es diseñada e implementada. Un aspecto único de Odyssey DE es que sus componentes interactúan a través de protocolo CORBA. De esta forma, cualquier modelo de dominio legado o componentes residentes en ambiente distinto pueden ser fácilmente integrados.

Odyssey DE tiene varias actividades que son equivalentes a las actividades de Ingeniería de Requerimientos. El análisis de factibilidad, captura y análisis de requerimientos son llevados a cabo en las fases de Análisis de Viabilidad del Dominio y Análisis del Dominio, sin embargo, Odyssey DE, no proporciona técnica alguna para ejecutar el análisis de factibilidad. Utiliza la técnica de modelado de características de FODA para identificar los requerimientos comunes y variables. No tiene una técnica para la selección de potenciales miembros, así como de sus características. Los requerimientos son modelados utilizando técnicas de modelado orientado a objetos tales como diagramas de casos de uso y diagramas de clase. Odyssey DE utiliza ciertos patrones de diseño para especificar conceptos del dominio. También proporciona mecanismos de seguimiento entre los requerimientos y otros artefactos, utilizando herramientas de soporte, sin embargo, la técnica utilizada para esto no es especificada.

PARTE III

En esta segunda parte se describe el trabajo de investigación realizado. Es en esta parte donde se muestran los artefactos obtenidos dentro de la fase de Ingeniería de Requerimientos de la Ingeniería de Dominios de la LPS a implantar en la Gerencia de Desarrollo de Software del CIMAT. Además, se describe el árbol de características obtenido el cual tuvo como objetivo fungir como guía para la obtención de los artefactos de requerimientos (casos de uso y requerimientos funcionales) y del cual se espera se obtengan los artefactos de las fases posteriores dentro de la Ingeniería de Dominios, los cuales se proponen como posibles trabajos futuros.

Esta parte consta principalmente de tres capítulos:

El capítulo 7 describe el dominio de los sistemas de optimización que serán parte de la LPS a implantar, así como también se mencionan algunos antecedentes relevantes de la Gerencia de Desarrollo.

En el capítulo 8 se describe y explica el árbol de características obtenido a partir del análisis de dos sistemas de optimización ya existentes dentro de la gerencia. A partir de este árbol se definirán los artefactos de requerimientos comunes a los sistemas analizados.

En el capítulo 9 se describe el análisis efectuado al dominio del problema, así como los resultados obtenidos (el modelo de casos de uso y requerimientos funcionales), los cuales fueron verificados por el Gerente del Departamento mencionado.

En el capítulo 10 se describe el modelo de casos de uso y los requerimientos funcionales para uno de los productos de la LPS, tomando como base el documento generado en el capítulo 9.

7. Antecedentes del problema

Como se ha comentado, este trabajo de investigación ha sido realizado para la Gerencia de Desarrollo de Software (GDS) del CIMAT. Dicha gerencia se ubica dentro de la Coordinación de Servicios Tecnológicos (CST) y forma parte de los laboratorios de consultoría de los que consta el CIMAT, en la figura 7.1 (tomada del sitio web del CIMAT) se muestran las gerencias que forman parte de la CST. El desarrollo de software (servicio ofrecido por la GDS) forma parte de los servicios que ofrece el laboratorio de computación, siendo principalmente los siguientes:

- Desarrollo de tecnología
- Desarrollo de software
- Transferencia de tecnología
- Asesorías, cursos de actualización y capacitación

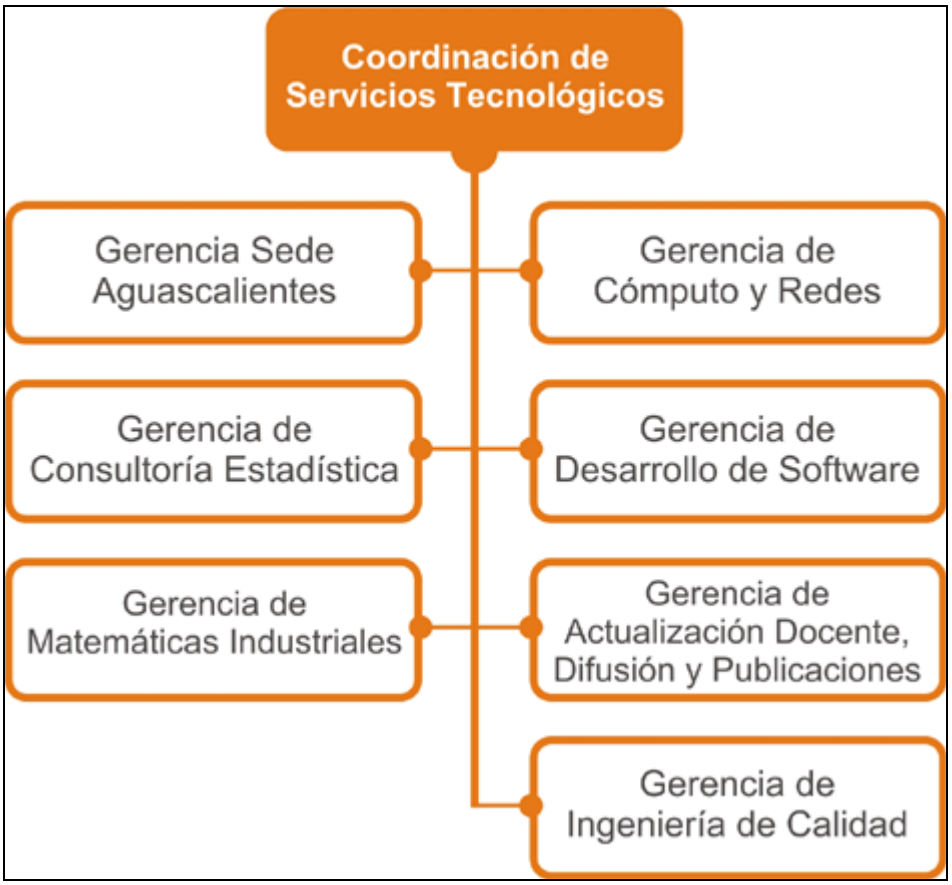

Figura 7.1. Gerencias de la Coordinación de Servicios Tecnológicos del CIMAT.

La GDS es dirigida por el M.C. Maximino Tapia Rodríguez el cual se encarga de coordinar los proyectos solicitados por empresas que requieren apoyo del CIMAT así como de reunir al personal adecuado para desarrollar dichos proyectos. Entre los tipos de proyectos más solicitados se encuentran los sistemas de optimización, para los cuales el CIMAT cuenta con el personal ad-hoc para realizarlos.

Cuando una empresa solicita al CIMAT el desarrollo de uno de estos sistemas, le informa a la GDS los requerimientos de su problema donde dichos requerimientos varían en cuanto a forma pues a

veces les proporcionan solamente algunos artículos con los algoritmos que desean que sean implementados u otras veces incluso les proporcionan los parámetros del problema en sí. Después de esto, el Gerente busca al personal adecuado para la realización del proyecto, entre ellos, busca la asesoría de alguno de los investigadores del área que proporcionen el conocimiento para la mejor elección de los algoritmos a implementar, luego asigna a alguien del personal para tal implementación, siendo frecuentemente los mismos alumnos o exalumnos de los postgrados los asignados para dicha tarea.

Entre los tipos de métodos de optimización implementados para dichos sistemas se encuentran principalmente métodos determinísticos y métodos heurísticos. Dentro de los determinísticos hasta ahora solo se han utilizado el algoritmo de Gradiente y el algoritmo de Programación Cuadrática Secuencial. Dentro de los heurísticos se han utilizado principalmente algoritmos evolutivos y recocido simulado.

Debido a la demanda de este tipo de sistemas, se identificó la necesidad y oportunidad de implantar una LPS para de alguna manera producir este tipo de sistemas de una manera más rápida, confiable, en el menor tiempo posible y con un bajo costo de desarrollo. Con esto se motivó a que este trabajo se enfocará a dicha necesidad, pues se identificaron que dichos sistemas tenían varias partes en común y para la realización de cada sistema se desarrollan desde cero dichas partes similares.

En los siguientes capítulos se describen el modelo de características del problema y el modelo de casos de uso del problema. En el modelo de características se identificaron las partes comunes y variables de los sistemas hasta ahora generados para a partir del mismo seguir con la fase de requerimientos.

8. Modelado de características

Este capítulo se enfoca a describir el árbol de características obtenido a partir de los sistemas de optimización desarrollados en la GDS. La generación de este árbol de características fue realizada bajo la especificación del método FODA, específicamente la fase de modelado del dominio.

Para generar tal árbol, primero se definió el conjunto de sistemas que formarán parte de la familia y puesto que dos éstos ya existían se utilizó una mezcla del enfoque de adopción extractivo con el enfoque proactivo (descritos en el capítulo 2) con los cuales se obtuvieron las características comunes y variables entre los sistemas de la familia listando anteriormente las características de los que ya existían.

Los sistemas identificados como parte de la familia, fueron los sistemas de optimización desarrollados para las empresas Prolec y Condumex, y el sistema que está en desarrollo para Comex. Dichos sistemas fueron desarrollados con distintos tipos de algoritmos, principalmente: algoritmos evolutivos, algoritmos de recocido simulado y algoritmos de optimización determinísticos. Aunque solo se tuvo la oportunidad de ver la ejecución de uno de ellos, las características de los demás fueron obtenidas a través de entrevistas con el gerente del departamento.

8.1. Características de los sistemas existentes

Antes de mostrar el árbol de características, se listarán las características de los dos sistemas existentes, para mostrar las que son similares y las que son variables entre ellas y tener evidencia del porqué fue formado el árbol de la manera en la que está.

Para la empresa Prolec, se han desarrollado sistemas de optimización con algoritmos evolutivos con las siguientes características:

- Entrada de datos:
    - a través de una base de datos.
- Salida de datos:
    - por pantalla y a una base de datos.
- Ejecución del proceso de optimización:
    - Iniciar, pausar y detener.
- Monitoreo del proceso de optimización:
    - modo gráfico.
- Problema de optimización que se trata.
- Algoritmo:
    - Función objetivo
    - Parámetros del algoritmo
        - Parámetros de algoritmo evolutivo
        - Parámetros de recocido simulado
    - Conjunto de restricciones
    - Algoritmo Heurístico
        - Algoritmo evolutivo
        - Recocido Simulado
- Postprocesamiento

Para la empresa Condumex, se desarrollaron sistemas de optimización con métodos determinísticos con las siguientes características:

- Entrada de datos:
    - a través del teclado
- Salida de datos:
    - por pantalla y a un archivo
- Ejecución del proceso de optimización:
    - iniciar y detener
- Monitoreo del proceso de optimización:
    - blink
- Problema de optimización que se trata
- Algoritmo:
    - Función objetivo
    - Parámetros del algoritmo
        - Parámetros de Gradiente
    - Conjunto de restricciones
    - Algoritmo Determinístico
        - Gradiente

El sistema para la empresa Comex está aún en desarrollo por lo que no se recolectaron las características del mismo.

8.2. Árbol de características

Ahora se mostrará el árbol de características generado a partir de las características mostradas y algunas características pensadas de manera proactiva. Para poder mostrarse el árbol se dividió en las figuras 6.1 y 6.2 debido a su extensión.

En la figura 6.1 se muestra la parte del árbol que tiene a las características principales de entrada de datos, el problema en sí y el algoritmo de solución.

Como se puede ver en dicha figura, la entrada de datos es múltiple porque se pensó proactivamente en que se pudieran introducir datos de más de un problema y que éstos fueran ejecutados con filosofía FIFO (primero en llegar, primero en procesar). Esta entrada de datos es obligatoria, es decir, que todos los sistemas tendrán un módulo o componente para entrada de datos. Esta entrada, es especializada (líneas punteadas) en tres características alternativas, es decir, que sólo una de ellas podrá ser implementada en un sistema de la familia. Todo esto significa que cada miembro de la familia tendrá una y solamente una de las siguientes tres entradas de datos: por medio del teclado ó por medio de un archivo ó por medio del acceso a una base de datos.

Con respecto a la característica que representa al problema de optimización, puede verse que esta característica es obligatoria para todos los sistemas. Esta característica no fue definida a mayor detalle puesto que el problema de optimización de cada sistema es demasiado particular.

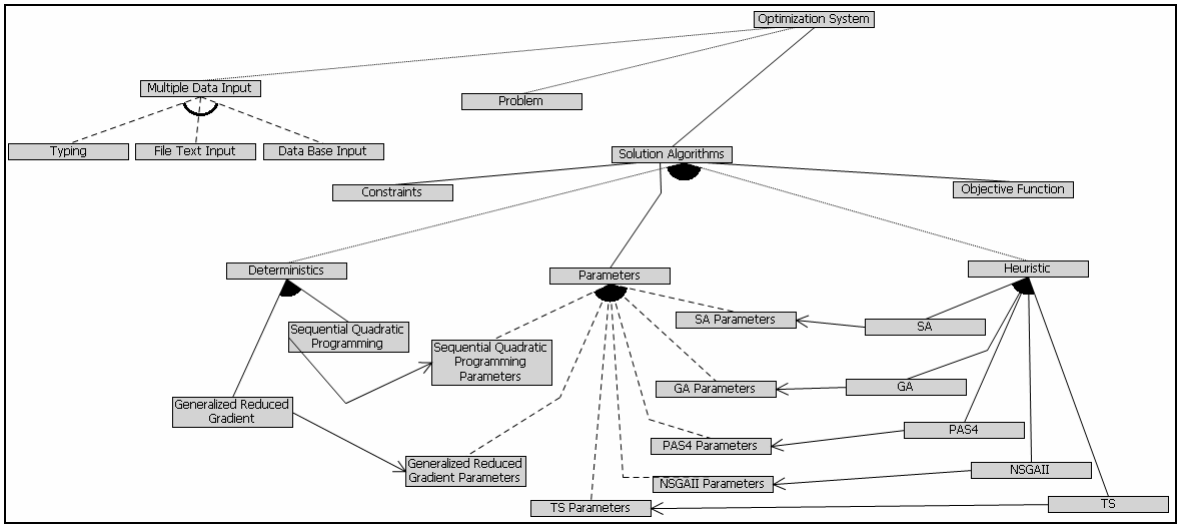

Figura 8.1. Primera parte del árbol de características para la LPS.

Con respecto a la característica que representa al algoritmo de solución del problema, se puede ver que es una característica obligatoria para todos los sistemas de la familia pues representa el componente principal, y está compuesta de otras características más que también son obligatorias tales como:

- Dos características relacionados con el símbolo “OR” que representan los tipos de algoritmos que se utilizarán para resolver el problema de optimización, significando que al menos uno de los dos tipos debe ser utilizado pero dando la posibilidad que se utilicen algoritmos de ambos tipos. Estos dos tipos son algoritmos determinísticos y algoritmos heurísticos representados por las características con mismo nombre. Como se puede ver cada una de estas características a su vez están compuestas de más características representando a los algoritmos en sí que se utilizarán para los problemas de utilización; éstos están relacionados con el símbolo “OR” significando que al menos uno debe ser seleccionado pero de igual manera se da la posibilidad de utilizar más de uno. Dentro del tipo de algoritmos determinísticos se tienen a los algoritmos: gradiente reducido y programación cuadrática. Dentro del tipo de algoritmos heurísticos se tienen a los algoritmos: evolutivos (GA), recocido simulado (SA), búsqueda tabú (TS), PAS4 y NSGAII (siendo estos dos últimos desarrollados originalmente en el CIMAT).

- Una característica que representa la función objetivo que es la solución buscada dentro del problema de optimización. Esta característica es obligatoria dentro de la característica que representa al algoritmo de solución al ser parte de ésta última.

- Una característica que representa las restricciones del problema, la cual también es obligatoria pues forma parte del algoritmo de solución, lo que significa que esta característica será parte de todos los sistemas de la familia.

- Una característica que representa los parámetros del o los algoritmos elegidos para la solución del problema, como se puede ver, esta característica se especializa en un conjunto de tipos de parámetros relacionados con el símbolo “OR” el cual significa que debe ser seleccionado al menos uno de ellos pudiendo ser varios. Esto es debido a que de acuerdo al número de algoritmos que se hayan elegido para atacar el problema es el número de tipos de parámetros que se utilizarán.

Se pueden ver también unas líneas que van desde los algoritmos hasta los tipos de parámetros. Estas líneas son implicaciones o reglas llamadas “requires” en FODA que como se explicó en el

capítulo 4 significan que depende del algoritmo que se utilice es el tipo de parámetros que se debe utilizar. Por ejemplo, si se utilizan algoritmos evolutivos implica que forzosamente se utilicen parámetros para algoritmos evolutivos.

En la figura 6.2 se muestra la segunda parte del árbol de características, que como se puede ver, tiene las características principales de: salida de datos, procesamiento, monitoreo y postprocesamiento.

Con respecto a la característica que representa a la salida de datos puede verse que es una característica obligatoria (pues todos los sistemas deben tener un modo de arrojar la solución o soluciones encontradas. Esta característica se especializa en tres características, siendo una de ellas obligatoria: la salida de datos a pantalla, lo cual significa que todos los sistemas deberán arrojar los resultados a la pantalla. Además, se especializa en dos características más: la salida a un archivo y la salida a una base de datos; éstas dos características están relacionadas con el símbolo “OR” lo cual significa que al menos una de ellas debe ser seleccionado pudiendo seleccionarse ambas. De esta manera un sistema de la familia arrojará los resultados a pantalla y además arrojará los resultados a un archivo o a una base de datos.

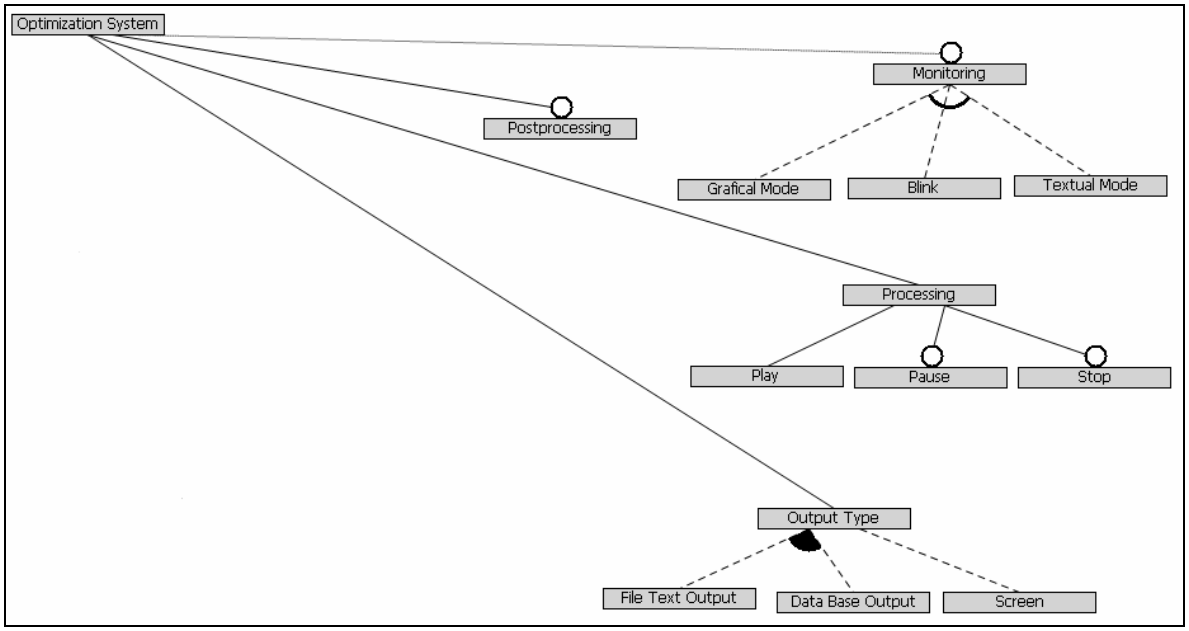

Figura 8.2. Segunda parte del árbol de características para la LPS.

La característica que representa a la ejecución del o los algoritmos (procesamiento) es una característica obligatoria la cual está compuesta de tres características: el inicio de la ejecución, la pausa o paro temporal de la ejecución y el paro final de la ejecución. Como se puede ver el inicio de la ejecución es obligatoria lo que significa que todos los sistemas tendrán algún modo de interacción para iniciar la ejecución del algoritmo de solución; y la pausa y paro total son características que tienen el símbolo de “opcional” indicando que pueden ser seleccionadas o no.

La característica que representa al monitoreo de la ejecución del algoritmo es una característica opcional, puesto que para algoritmos determinísticos cuyo tiempo de ejecución es irrisorio no es necesaria. Sin embargo, para los sistemas con algoritmos heurísticos cuyo tiempo de ejecución tiende a ser demasiado largo es necesaria y para eso, esta característica es especializada en tres características alternativas: monitoreo gráfico ó monitoreo textual ó simplemente un indicador de

que el algoritmo está trabajando. Esto significa que para los sistemas que si cuentan con la característica de monitoreo, sólo uno de los tres puede ser seleccionado.

Por último, la característica de postprocesamiento es opcional porque puede ser seleccionada o no para los sistemas de la familia. Esta característica está incluida en el árbol porque representa una manera más ad-hoc de visualizar las soluciones al usuario, pues en ocasiones es complicado entender un conjunto de números.

En este capítulo se proporcionó el árbol de características para la LPS a implantar. Este árbol es relevante porque es la base para el resto de las etapas de la LPS como se verá en el siguiente capítulo donde se describen los requerimientos para la LPS y su relación con dicho árbol. Las fases posteriores a la fase de requerimientos quedan fuera del alcance de este trabajo como ya se ha comentado.

9. Requerimientos de la LPS

9.1. Introducción

9.1.1. Propósito

Este documento de SRS describe los requerimientos funcionales de la LPS para los sistemas de optimización de la Gerencia de Desarrollo de Software del CIMAT. En lo subsiguiente, se hará referencia a la Línea de Productos de Software para los sistemas de optimización de la Gerencia de Desarrollo de Software del CIMAT únicamente como LPS de optimización y a la Gerencia de Desarrollo de Software del CIMAT como Gerencia de Desarrollo SW.

La LPS de optimización especificada en el presente documento corresponde a la versión 1.0.

Este documento representa la base para la LPS de optimización, por lo que podrá ser refinado cuando se utilice para especificar los requerimientos funcionales de alguno de los miembros de la LPS optimización.

9.1.2. Alcance del Proyecto

La LPS de optimización permitirá crear miembros (también llamados productos individuales) a partir de un conjunto de artefactos reutilizables. Dado lo anterior, la implantación de una LPS de optimización permitirá a la Gerencia de Desarrollo SW desarrollar sistemas de optimización de manera más rápida, confiable y económica, ya que gracias a la LPS de optimización se reutilizarán componentes complejos, y permitirá contar con un repositorio de artefactos que estarán listos para generar nuevos miembros de la familia.
A partir de los artefactos reutilizables con los que cuenta la Línea de Productos deberá ser posible crear un nuevo miembro de la familia con la funcionalidad básica. Adicionalmente, si se desea que el nuevo miembro de la familia cuente con características o funcionalidad extra, deberá especificarse en el documento SRS correspondiente a dicho miembro.

El presente documento SRS es el primero de dichos artefactos reutilizables, ya que constituye la base para los documentos SRS de miembros futuros de la LPS de optimización.

9.2. Descripción General

9.2.1. Perspectiva de la Línea de Productos

Un miembro de la LPS de optimización es un sistema individual que tiene la capacidad de resolver un problema de optimización aplicando diversos algoritmos a un conjunto de entrada de datos, de tal forma que puede producir resultados visibles y útiles los cuales representan la mejor solución a dicho problema. Es por ello, que el CIMAT, al contar con la LPS de optimización puede atender la demanda de este tipo de servicios solicitados por las empresas que así lo requieran, en un lapso de tiempo ‘corto’, con calidad superior y aplicando del menor esfuerzo.

9.2.2. Clases de Usuarios

- Cliente. Representado por la Gerencia de Desarrollo. Su principal interés en la creación de la LPS de optimización radica en la capacidad de desarrollar rápidamente sistemas individuales utilizando la LPS de optimización.
- Investigador. En colaboración con la Gerencia de Desarrollo, proporciona conocimiento para la resolución de los problemas de optimización de las empresas que solicitan los servicios del CIMAT.

- Estudiante. Colabora en la Gerencia de Desarrollo de Software desarrollando los sistemas de optimización. Se ve afectado con la implantación de la LPS de optimización ya que su ciclo de desarrollo deberá adaptarse a la LPS de optimización.
- Usuario. Es aquella entidad que recibe un producto individual generado a través de la LPS de optimización como respuesta a la solicitud de la construcción de un sistema para la resolución de un problema de optimización.

9.2.3. Ambiente de Operación

El ambiente de operación para el cual serán generados los sistemas individuales corresponde a un sistema operativo Windows funcionando sobre procesadores compatibles con Intel. Los ambientes de operación no tendrán herramientas de desarrollo instaladas, por lo que un miembro debe funcionar totalmente independiente.

El lenguaje de programación utilizado para el desarrollo será C++ Builder.

Los miembros de LPS de optimización pueden tener distintos tipos de entradas de datos tales como archivos de texto y bases de datos.

9.2.4. Suposiciones y Dependencias

No especificadas.

9.3. Características de Sistema

9.3.1. Cargar datos de entrada

9.3.1.1. Descripción.

La LPS de optimización debe proveer la característica de sistema a través de la cual se alimente o proporcione información al programa para que tenga datos sobre los cuales realizar el proceso de optimización. Estos datos se detallarán a nivel de miembro de la LPS de optimización.

9.3.1.2. Secuencias de Estímulos/Respuestas

Estímulo: El usuario solicita al sistema que desea capturar datos para iniciar un nuevo proceso de optimización.
Respuesta: El sistema, dependiendo del tipo de entrada, despliega un mecanismo para que el usuario especifique los datos correspondientes. Este mecanismo de entrada será definido por el miembro de la LPS de optimización.
Estímulo: El usuario introduce los datos correspondientes y le indica al sistema que ha finalizado de realizar la captura.
Respuesta: El sistema guarda la información y queda en espera de que el usuario proporcione otra petición.

9.3.1.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

9.3.2. Poner en marcha el proceso de optimización

9.3.2.1. Descripción.

La LPS de optimización debe proveer la característica de sistema a través de la cual el usuario pondrá en marcha el proceso de optimización.

9.3.2.2. Secuencias de Estímulos/Respuestas

Estímulo: Después de que el usuario ha realizado la carga de datos o ha pausado un proceso en marcha, solicita al sistema que desea iniciar el proceso de optimización.
Respuesta: El sistema inicia el proceso de optimización indicando de alguna forma que se encuentra en proceso.

9.3.2.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

9.3.3. Detener el proceso de optimización

9.3.3.1. Descripción.

La LPS de optimización debe proveer la característica de sistema a través de la cual el usuario podrá detener el proceso de optimización.

9.3.3.2. Secuencias de Estímulos/Respuestas

Estímulo: Después de que el usuario ha puesto en marcha el proceso de optimización y antes de que dicho proceso finalice, solicita al sistema que desea detenerlo.
Respuesta: El sistema detiene el proceso de optimización de forma repentina.

9.3.3.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

9.3.4. Pausar el proceso de optimización

9.3.4.1. Descripción.

Dependiendo de método que se esté utilizando en el proceso de optimización, la LPS de optimización debe proveer la característica de sistema a través de la cual el usuario podrá pausar el proceso de optimización.

9.3.4.2. Secuencias de Estímulos/Respuestas

Estímulo: Después de que el usuario ha puesto en marcha el proceso de optimización y antes de que dicho proceso finalice, solicita al sistema que desea pausarlo.

Respuesta: El sistema pausa el proceso de optimización y queda en espera de que el usuario le solicite reanudarlo mediante la característica que le permite ponerlo en marcha, la cual fue descrita en el punto 3.2.

9.3.4.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

9.3.5. Monitoreo del proceso de optimización

9.3.5.1. Descripción.

Dependiendo de método que se esté utilizando en el proceso de optimización, la LPS de optimización debe proveer la característica de sistema a través de la cual el usuario podrá monitorear el proceso de optimización.

9.3.5.2. Secuencias de Estímulos/Respuestas

Estímulo: Después de que el usuario ha puesto en marcha el proceso de optimización y antes de que dicho proceso finalice, solicita al sistema que desea monitorearlo.
Respuesta: El sistema muestra las soluciones óptimas que hasta el momento ha encontrado.

9.3.5.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

9.3.6. Captura de resultados del proceso de optimización

9.3.6.1. Descripción.

La LPS de optimización debe proveer la característica de sistema a través de la cual el usuario podrá capturar los resultados de salida una vez que el proceso de optimización termine.

9.3.6.2. Secuencias de Estímulos/Respuestas

Estímulo: Después de que el proceso de optimización ha finalizado el usuario le indica al sistema que desea capturar los resultados de salida.
Respuesta: El sistema le indica al usuario que especifique el método de salida, el cual podrá a archivos de texto, bases de datos o a pantalla únicamente.
Estímulo: El usuario especifica el método de salida.
Respuesta: El sistema genera la salida dependiendo del método seleccionado por el usuario. En sistema queda en espera de que el usuario seleccione otra instrucción.

9.3.6.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

9.4. Modelo de Casos de Uso

Las características de sistema descritas en la sección anterior, son demasiado 'finas' por lo que el mapeo con los casos de uso de forma directa es posible. El modelo de casos de uso presentado en esta sección utiliza las extensiones definidas por PLUS para el modelado de las características comunes y variables.

9.4.1. Funcionalidad principal

Los casos de uso que permiten la carga de datos e iniciar el proceso de optimización forman parte del 'core' ya que todos los miembros de la LPS de optimización lo incluirán como parte de su funcionalidad básica. El caso de uso que permite el monitoreo del proceso de optimización es 'optional' debido a que esté será incluido en algún miembro dependiendo del tipo de algoritmos de optimización que se utilicen. Al igual que los casos de uso que permiten la carga de datos e iniciar el proceso de optimización, el caso de uso que permite la captura de resultados también forma parte del 'core' ya que es una funcionalidad obligatoria para todos los miembros de la LPS de optimización. Los casos de uso 'core' de la LPS de optimización se muestran en la Figura 9.1.

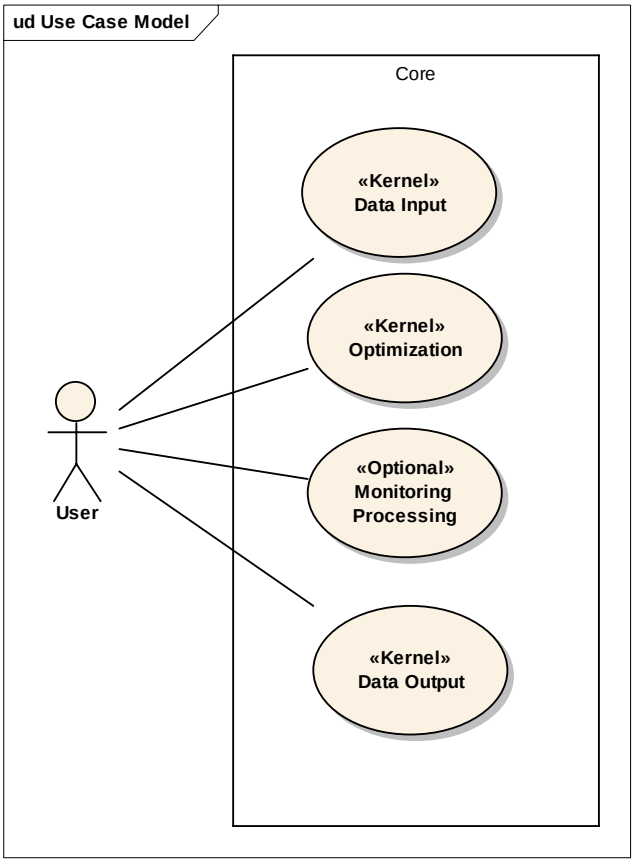

Figura 9.1. Core de la LPS de optimización.

9.4.1.1. Carga de datos

La carga de datos, como se mencionó anteriormente forma parte del 'core' de la LPS de optimización. Adicionalmente, la carga de datos debe permite varios métodos de ingreso de datos como son: por medio de archivos de texto, bases de datos o de forma manual. El caso de uso correspondiente a la Carga de datos es mostrado en la Figura 9.2.

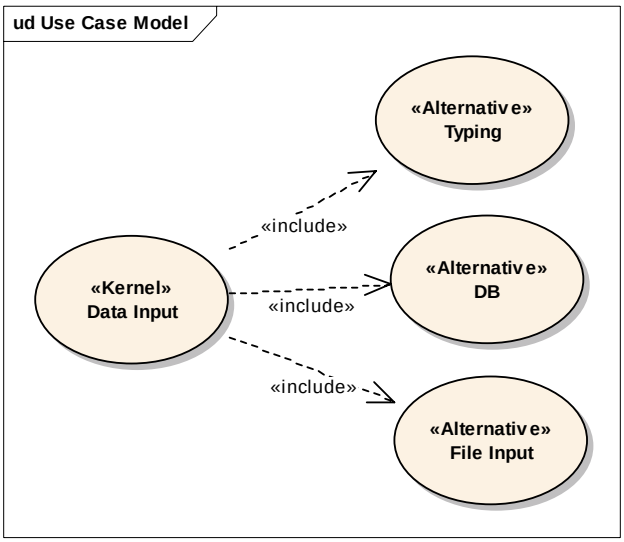

Figura 9.1. Carga de datos.

9.4.1.2. Iniciar proceso de optimización

El caso de uso que permite iniciar el proceso de optimización extiende a otros tres casos de uso más:

En el caso de que se ha realizado la carga de datos o después de que se ha pausado el proceso se le permite al usuario iniciar el proceso ('playing'). El caso de uso que permite detener proceso es 'optional' ya que dependerá de los algoritmos de optimización que se estén utilizando en el miembro de la LPS de optimización. El caso de uso que permite pausar el proceso de optimización al igual que el caso de uso que permite detenerlo es 'optional', ya que su presencia en el miembro de la LPS de optimización dependerá de los algoritmos de optimización que utilice. El caso de uso correspondiente al Inicio del proceso de optimización se muestra en la Figura 9.3.

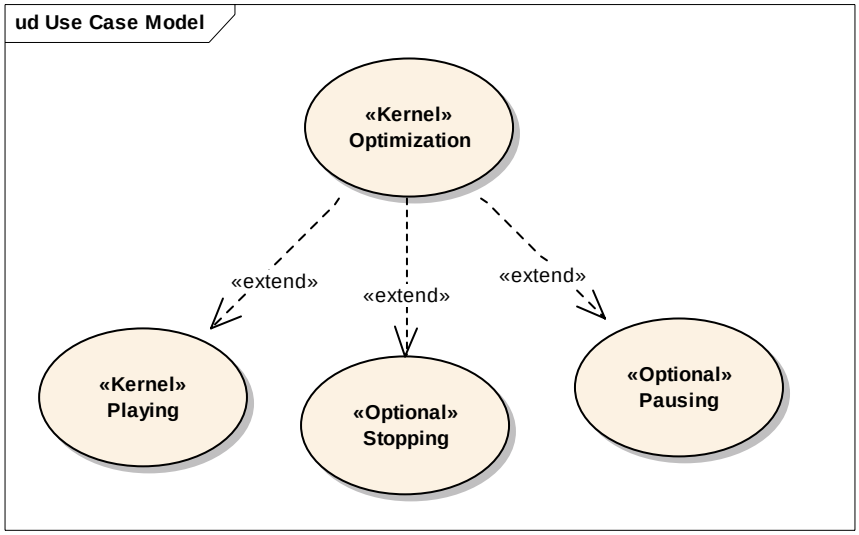

Figura 9.2. Inicio del proceso de optimización.

9.4.1.3. Monitoreo del Proceso de Optimización

El caso de uso que permite el monitoreo del proceso de optimización es 'optional' ya que depende de lo algoritmos que utilice el miembro de la LPS de optimización para que se encuentre presente. Adicionalmente, si el caso de uso es incluido, podrán seleccionarse los parámetros de configuración del monitoreo, ya que podrán existir diversos mecanismos mostrarlo: gráfica, blink, barra de progreso. Este caso de uso, es mostrado en la Figura 9.4.

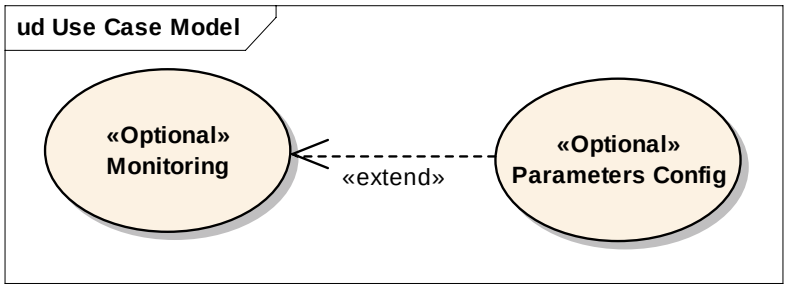

Figura 9.3. Monitoreo del proceso de optimización.

9.4.1.4. Captura de resultados

El caso de uso que permite la captura de resultados forma parte de la funcionalidad básica (común) de la LPS de optimación. Así mismo, la salida por pantalla también se parte del 'core', en tanto que son opcionales las salidas por Base de datos y Archivos de texto. El caso de uso correspondiente a la Captura de resultados es mostrado en la Figura 9.5.

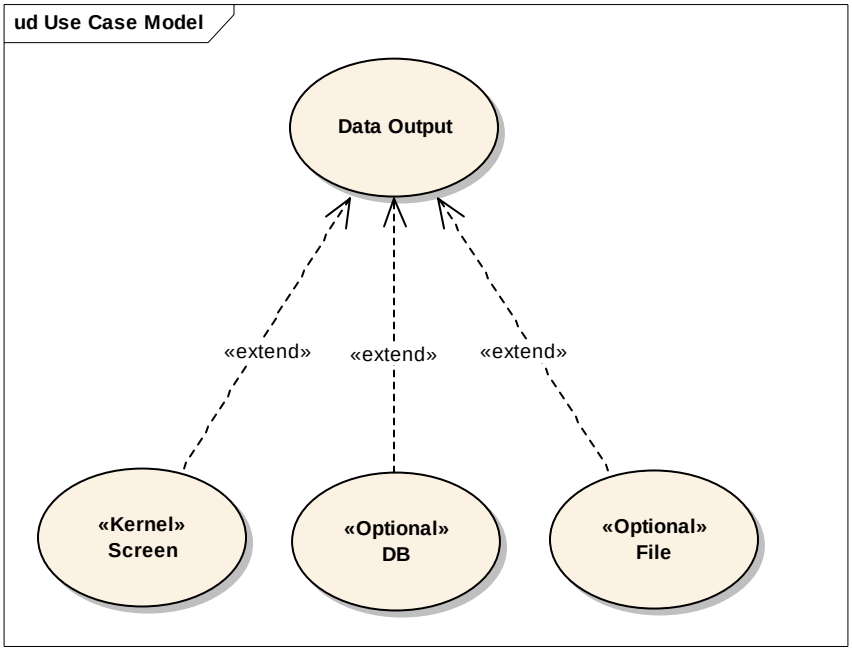

Figura 9.5. Salida de datos.

9.5. Otros requerimientos

No especificados.

10. Requerimientos de uno de los productos de la LPS

10.1. Introducción

10.1.1. Propósito

Este documento de SRS describe los requerimientos funcionales de la aplicación Cables Super Conductores, la cual es miembro de la Línea de Productos de Software para los sistemas de optimización de la Gerencia de Desarrollo de Software del CIMAT. En lo subsiguiente, se hará referencia a la Línea de Productos de Software para los sistemas de optimización de la Gerencia de Desarrollo de Software del CIMAT únicamente como LPS de optimización y a la Gerencia de Desarrollo de Software del CIMAT como Gerencia de Desarrollo SW.

La LPS de optimización especificada en el presente documento corresponde a la versión 1.0.

La versión de la Aplicación Cables Super Conductores corresponde a la versión 1.0.

Este documento representa la especificación de requerimientos de aplicación, por lo que incluye características que aplican únicamente a este miembro. Por lo tanto, no es recomendable utilizar el presente documento como base para la especificación de otros miembros de la LPS de optimización.

10.1.2. Alcance del Proyecto

Cables Super Conductores es un miembro de la LPS de optimización el cual permite obtener una solución óptima del uso de materiales para la construcción de cables. Debido a que Cables Super Conductores es miembro de la LPS de optimización, ya contiene toda la funcionalidad principal (‘core’), por lo que únicamente habrá que definir de manera más concreta aquellos aspectos que así lo requieran.

10.2. Descripción General

10.2.1. Perspectiva de la Línea de Productos

Se desea que Cables Super Conductores sea una aplicación que auxilie en el diseño y construcción de cables de tal forma que se obtenga un ahorro en el material utilizado. Para esto, se utilizan los algoritmos de optimización con el método de gradiente. Debido a la naturaleza de estos algoritmos y la cantidad de los datos involucrados, el proceso de optimización no dura más de tres minutos.

10.2.2. Clases de Usuarios

- Cliente. Representado por la Gerencia de Desarrollo. Su principal interés en la creación de la LPS de optimización radica en la capacidad de desarrollar rápidamente sistemas individuales utilizando la LPS de optimización.
- Investigador. En colaboración con la Gerencia de Desarrollo, proporciona conocimiento para la resolución de los problemas de optimización de las empresas que solicitan los servicios del CIMAT.
- Estudiante. Colabora en la Gerencia de Desarrollo de Software desarrollando los sistemas de optimización. Se ve afectado con la implantación de la LPS de optimización ya que su ciclo de desarrollo deberá adaptarse a la LPS de optimización.
- Usuario. Es aquella entidad que recibe un producto individual generado a través de la LPS de optimización como respuesta a la solicitud de la construcción de un sistema para la resolución de un problema de optimización.

10.2.3. Ambiente de Operación

El ambiente de operación en el cual será operado el sistema Cables Super Conductores corresponde a un sistema operativo Windows funcionando sobre procesadores compatibles con Intel. Los ambientes de operación no tendrán herramientas de desarrollo instaladas, por lo que Cables Super Conductores deberá funcionar totalmente independiente.

El lenguaje de programación utilizado para el desarrollo será C++ Builder.

La entrada de datos a Cables Super Conductores será realizada a través del teclado.

10.2.4. Suposiciones y Dependencias

No especificadas.

10.3. Características de Sistema

10.3.1. Cargar datos de entrada

10.3.1.1. Descripción.

Cables Super Conductores debe proveer la característica de sistema a través de la cual se alimente o proporcione información al programa para que tenga datos sobre los cuales realizar el proceso de optimización. Estos datos son mostrados en la Figura 10.1.0.1.

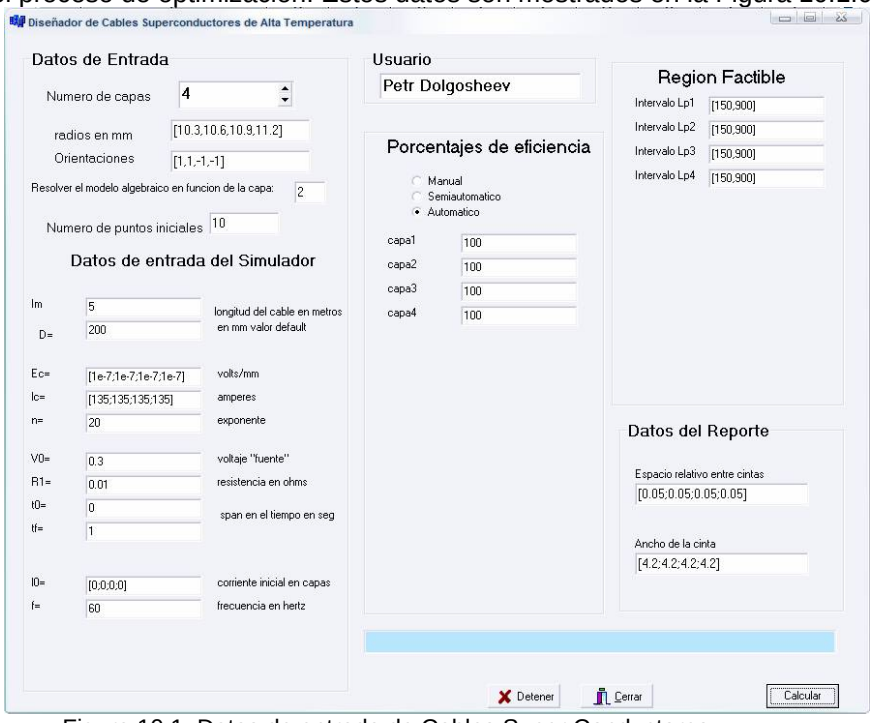

Figura 10.1. Datos de entrada de Cables Super Conductores.

10.3.1.2. Secuencias de Estímulos/Respuestas

Estímulo: El usuario solicita al sistema que desea capturar datos para iniciar un nuevo proceso de optimización.
Respuesta: El sistema despliega una forma de datos para que el usuario introduzca los datos.

Estímulo: El usuario introduce los datos correspondientes y le indica al sistema que ha finalizado de realizar la captura.
Respuesta: El sistema guarda la información y queda en espera de que el usuario proporcione otra petición.

10.3.1.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

10.3.2. Poner en marcha el proceso de optimización

10.3.2.1. Descripción.

Cables Super Conductores debe proveer la característica de sistema a través de la cual el usuario pondrá en marcha el proceso de optimización.

10.3.2.2. Secuencias de Estímulos/Respuestas

Estímulo: Después de que el usuario ha realizado la carga de datos o ha pausado un proceso en marcha, solicita al sistema que desea iniciar el proceso de optimización.
Respuesta: El sistema inicia el proceso de optimización indicando de alguna forma que se encuentra en proceso.

10.3.2.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

10.3.3. Detener el proceso de optimización

10.3.3.1. Descripción.

Cables Super Conductores debe proveer la característica de sistema a través de la cual el usuario podrá detener el proceso de optimización.

10.3.3.2. Secuencias de Estímulos/Respuestas

Estímulo: Después de que el usuario ha puesto en marcha el proceso de optimización y antes de que dicho proceso finalice, solicita al sistema que desea detenerlo.
Respuesta: El sistema detiene el proceso de optimización de forma repentina.

10.3.3.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

10.3.4. Captura de resultados del proceso de optimización

10.3.4.1. Descripción.

Cables Super Conductores debe proveer la característica de sistema a través de la cual el usuario podrá capturar los resultados de salida una vez que el proceso de optimización termine.

10.3.4.2. Secuencias de Estímulos/Respuestas

Estímulo: Después de que el proceso de optimización ha finalizado el usuario le indica al sistema que desea capturar los resultados de salida.
Respuesta: El sistema le indica al usuario que especifique el método de salida, el cual podrá a archivos de texto, bases de datos o a pantalla únicamente.
Estímulo: El usuario especifica el método de salida.
Respuesta: El sistema genera la salida dependiendo del método seleccionado por el usuario. En sistema queda en espera de que el usuario seleccione otra instrucción.

10.3.4.3. Requerimientos funcionales

Debido al nivel de granularidad de las características del sistema, no se detallarán los requerimientos funcionales tanto para la LPS de optimización como para sus miembros. Por lo tanto se realizará un mapeo directo de las características de sistema con los casos de uso.

10.4. Modelo de Casos de Uso

Las características de sistema descritas en la sección anterior, son demasiado 'finas' por lo que el mapeo con los casos de uso de forma directa es posible. El modelo de casos de uso presentado en esta sección utiliza las extensiones definidas por PLUS para el modelado de las características comunes y variables.

10.4.1. Funcionalidad principal

Los casos de uso que aplica para la aplicación Cables Super Conductores son aquellos que forman únicamente parte del 'core' y son los siguientes:

- Cargar datos
- Proceso de optimización
- Salida de datos

Los principales casos de uso de la aplicación se muestran en la Figura 10.2.

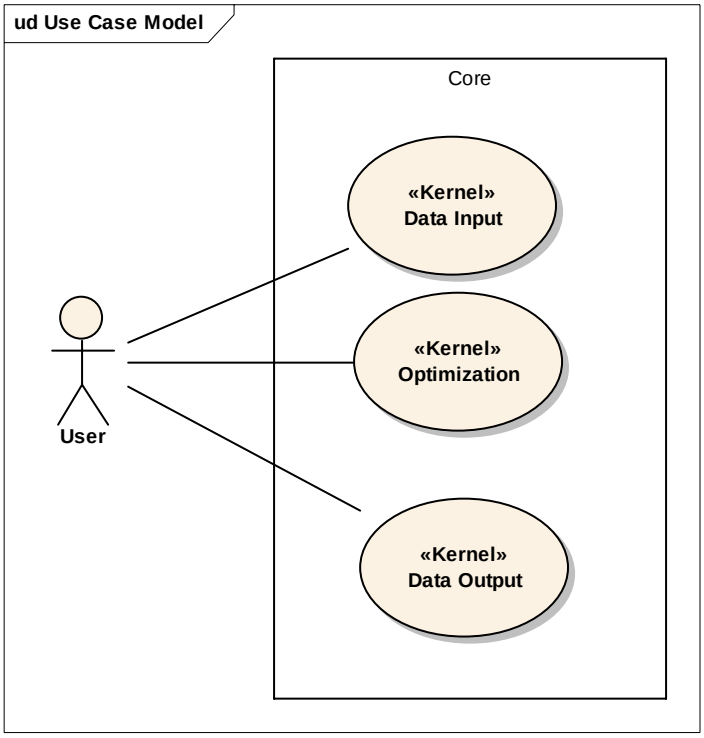

Figura 10.2. Casos de uso de la aplicación Cables Super Conductores.

10.4.1.1. Carga de datos

La carga de datos, como se mencionó anteriormente forma parte del 'core' de la LPS de optimización. Para Cables Super Conductores, la carga de datos es realizada de forma manual. El caso de uso correspondiente a la Carga de datos de la aplicación se muestra en Figura 10.3.

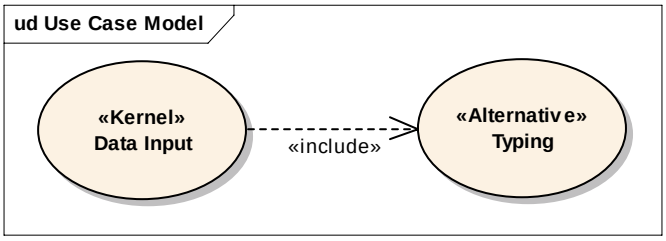

Figura 10.3. Carga de datos.

10.4.1.2. Iniciar proceso de optimización

El caso de uso que permite iniciar el proceso de optimización extiende a otros tres casos de uso más. Para el caso de Cables Super Conductores estarán disponibles los casos de uso que permite iniciar y detener el proceso de optimización. El caso de uso correspondiente al Inicio del proceso de optimización es mostrado en la Figura 10.4.

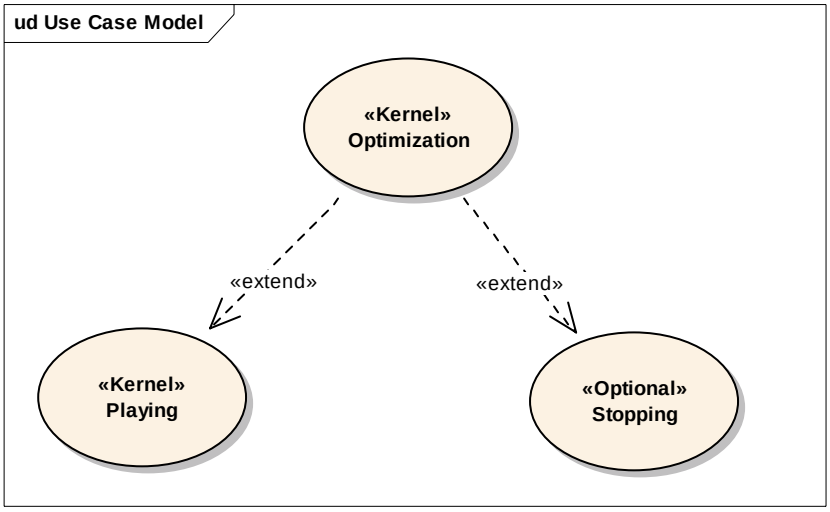

Figura 10.4. Inicio del proceso de optimización.

10.4.1.3. Captura de resultados

Para Cables Super Conductores, el método de captura de resultados será realizado a través de la pantalla, en la cual se mostrará una gráfica con las soluciones óptimas y distintas configuraciones de materiales; o adicionalmente, en caso de ser requerido, los resultados podrán ser almacenados en un archivo en formato seleccionado por el usuario. El caso de uso correspondiente a la Captura de resultados se muestra en Figura 10.5.

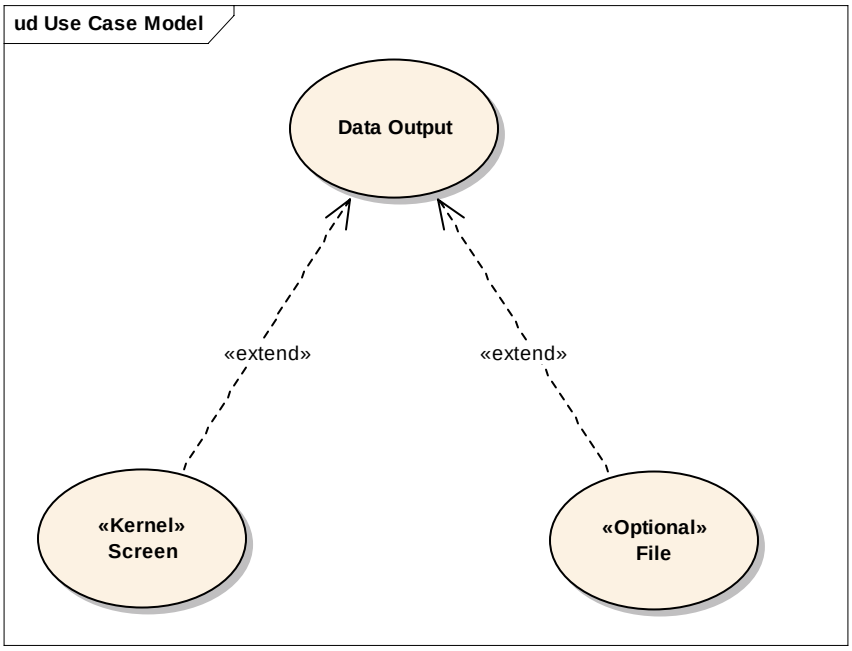

Figura 10.5. Salida de datos.

10.5. Otros requerimientos

No especificados.

PARTE IV

Esta es la última parte de este trabajo donde se presentan las conclusiones y las oportunidades de continuación para trabajos futuros.

En el capítulo 7 se presentan las conclusiones, las cuales son esenciales en cualquier trabajo de investigación; y en el capítulo 8 se presentan los posibles trabajos futuros que pueden desencadenarse a partir de este tema.

11. Conclusiones

En este trabajo se mostró que la parte fundamental para la implantación de una LPS es el desarrollo de los artefactos que fungirán como core assets de la familia. Esta es una ardua tarea pues dependiendo del enfoque de adopción varía la técnica a llevar a cabo. Lo que si es esencial, es la obtención del árbol de características en el cual se encuentran modeladas las similitudes y las variabilidades de los productos que forman parte de la familia y partir de él se deben llevar a cabo las fases que dicta el modelo cascada pero desde el ámbito de la Ingeniería de Dominios.

Este trabajo, tuvo como alcance solo la etapa de requerimientos, en la cual se produjeron artefactos de esta etapa conteniendo las similitudes y partes variables de los sistemas analizados.

Se espera que la implantación de la LPS sea un hecho en un futuro no muy lejano y que pueda ser realizada con éxito. Y es razonable pensar que mientras se produzcan nuevos sistemas de optimización se realice una retroalimentación al árbol de características actual de modo que éste se adecue más al nuevo conjunto de sistemas parte de la familia. Por lo tanto se puede decir que una LPS puede verse como un proceso iterativo en el cual cada vez que un nuevo miembro se integra a la familia se debe realizar un análisis para corroborar que los core assets sigan representando las similitudes y variabilidades reales de la familia.

Por último, como se mostró en este trabajo, la implantación de una LPS requiere una gran inversión económica, de recursos humanos, y sobre todo de esfuerzo. Sin embargo, los beneficios son enormes y hacen que valga la pena tal inversión.

12. Trabajos futuros

A pesar de que hay muchos trabajos de investigación reportados en esta área, aún hay muchísima madera que cortar, y este trabajo no es la excepción.

En este trabajo solo se abarcó la fase de requerimientos, la cual no es suficiente para la implantación exitosa de la LPS dentro del CIMAT. Se requieren esfuerzos adicionales para finiquitar esta tarea llevando a cabo las fases posteriores para complementar el conjunto de artefactos que fungirán como core assets, entre ellos la arquitectura, la implementación de los componentes así como la batería de pruebas para asegurar la calidad de los productos generados. De hecho, este trabajo surgió con el objetivo de servir como preámbulo a una tesis doctoral actualmente en desarrollo aquí en el CIMAT

Se espera este trabajo sirva como motivación para la continuación del propósito planteado. Una de las motivaciones esenciales, es que en México no se ha difundido este fascinante tema y ojalá que en un futuro próximo empiece a tener más auge a nivel investigación y que las organizaciones lo consideren para llevarlo a la práctica, ya que de implantarlo con éxito se está seguro no se arrepentirán.

Referencias

[1] Brooks II, F.P. “No Silver Bullet: Essence and Accidents of Software Engineering”. IEEE Computer, Vol. 20, No. 4 (Apr. 1987), pp. 10-19.

[2] Parnas, D.L. “On the Design and Development of Program Families”. IEEE Transactions on Software Engineering, SE-2:1-9, March 1976.

[3] Clements, P. & Northrop, L. “Software Product Lines: Practices and Patterns”. Reading, MA: Addison-Wesley, 2002.

[4] DeRemer, Frank & Kron, Hans H. “Programming-in-the-Large versus Programming-in-the-Small,” IEEE Trans. on Software Engineering, SE-2, 2, June 1976

[5] Clements, P. & Krueger, C. “Point/Counterpoint”. IEEE Software. July-August 2002.

[6] Ferguson, P. et al. “Software Process Improvement Works! (CMU/SEI-99-TR-027, ADA371804)”. Pittburgh, PA: Software Engineering Institute, Carnegie Mellon University, November 1999.

[7] Goldenson, D. & Herbsleb, J. “After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success (CMU/SEI-95-TR-009, ADA302225)”. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, August 1995.

[8] Zahran, S. “Software Process Improvement: Practical Guidelines for Business Success”. Reading, MA: Addison-Wesley, 1997.

[9] Kang, K. et al. “Feature-Oriented Domain Analysis (FODA) Feasibility Study, Technical Report CMU/SEI-90-TR-21”. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, November 1990.

[10] Chen P. “The Entity-Relationship Model – Toward an Unified View of Data”. ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, Pages 9-36.

[11] Kang, K. et al. “Feature-Oriented Product Line Engineering”. IEEE Software. July-August 2002.

[12] David, M.Weiss. “Commonality Analysis: A Systematic Process For Defining Families”. Lucent Technologies Bell Laboratories. 1997.

[13] Weiss, D.M. & Lai, C.T.R. “Software Product-Line Engineering: A Family-Based Software Development Process”. Reading, MA: Addison-Wesley, 1999.

[14] Atkinson, C. et al. “Component-based Product Line Engineering with UML”. Reading, MA: Addison-Wesley, 2002.

[15] Software Engineering Institute (SEI): "A Framework for Software Product Line Practice" - Version 4.2. Carnegie Mellon University.

[16] Software Product Lines: 9th International Conference. Springer. 2005. Hafedh Mili, Ali Mili, Sherif Yacoub, Edward Addy. Reuse-Based Software Engineering: Techniques Organization, and Controls. John Wiley. 2002.

[17] Hassan Gomaa. "Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures". Addison Wesley. 2004.

[18] Chetana Kuloor, Armin Eberlein. "Requirements for Software Product Lines". The University of Calgary. 2002.

[19] Karl E. Wiegers. "Software Requirements: Practical techniques for gathering and managing requirements through the product development cycle".

[20] Software Productivity Consortium Inc. "Synthesis: Reuse-Driven Software Processes Guidebook". Reuse-Driven Development Environment (RDE) Product Line Project, Software Productivity Consortium. 1996.

[21] Gupta,Neeraj.K, Lalit Jategaonkar Jagadeesan, eleftherios E.Koutsofios, David M. Weiss. Auditdraw: "Generating Audits the FAST Way". Lucent Technologies/Bell Laboratories Software Production Research Dept, AT&T Research, Network Services Research Lab, New Jersey. U.S.A. 1995.

[22] Joachim Bayer, Oliver Flege, Peter Knauber, Roland Laqua, Dirk Muthig, Klaus Schmid, and Tanya Widen. "PuLSE: A Methodology to Develop Software Product Lines". Fraunhofer Institute for Experimental Software Engineering, Germany and Lucent Technologies Software Product Line Engineering Laboratories, U.S.A. May 1999.

[23] Colin Atkinson, Joachim Bayer, and Dirk Muthig. "Component Based Product Line Development – The KobrA Approach". Fraunhofer Institute for Experimental Software Engineering, Germany. 1999.

[24] Succi,G. SENG609.08 "Class notes on Sherlock: An Object Oriented Framework for Domain Analysis and Engineering". The University of Calgary, Canada. 1999.

[25] Regina, M.M Braga., Claudia,M.L.Werner., and Marta,Mattoso. "Odyssey: A Reuse Environment based on Domain Models". Computer Science Department, Federal University of Rio de Janeiro, Brazil. 2001.

[26] Sommerville, I. Kotonya, G. "Requirements Engineering: Processes and Techniques". John Wiley & Son Ltd. 1998.

[27] Macaulay, L. A. "Requirements Engineering". Springer- Verlag. 1996.

[28] Jacobson Ivar, Griss Martin, Jonsson Patrik. "Software Reuse Architecture, Process and Organization for Business Success". ACM Press New York. 1997.