# Universidad de las Ciencias Informáticas

## Facultad 6


# Título: Estrategia para la gestión de la variabilidad de los componentes desarrollados sobre la plataforma Atlas.

Trabajo de Diploma para optar por el título de Ingeniero Informático.

**Autor(es):** Cecilia Dupotey Tussén

Eduardo René Medina Pérez

**Tutor(es):** Ing. Yampier Medina Tarancón

Ing. Aimé Esther Guzmán Ramírez

Junio 2013

"Año 55 de la Revolución"

*Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte…*

*Ernesto Guevara de la Serna*

## DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

**Cecilia Dupotey Tussén**

_____

Firma del Autor

**Eduardo René Medina Pérez**

_____

Firma del Autor

**Ing. Yampier Medina Tarancón**

_____

Firma del Tutor

**Ing. Aimé Esther Guzmán Ramírez**

_____

Firma del Tutor

## DATOS DE CONTACTO

Tutor: Ing. Yampier Medina Tarancón

Graduado de Ingeniería en Ciencias Informáticas.

Dirección: Universidad de las Ciencias Informáticas (UCI).

Teléfono Oficina:

Teléfono Apto: 837-22-30

E-mail: ytarancon@uci.cu

Profesión: Ingeniero en Ciencias Informáticas.

Categoría docente: Instructor.

Años de graduado: 3

Tutor: Ing. Aimé Esther Guzmán Ramírez

Graduado de Ingeniería en Ciencias Informáticas.

Dirección: Universidad de las Ciencias Informáticas (UCI).

Teléfono Oficina:

Teléfono Apto: 837-31-63

E-mail: aguzman@uci.cu

Profesión: Ingeniero en Ciencias Informáticas.

Categoría: Especialista General.

Años de graduado: 2

**AGRADECIMIENTOS**

*Cecilia Dupotey Tussén*

*A mis padres por estar siempre a mi lado apoyándome en todas las decisiones que tomado.*

*A mi hija por existir y esperarme con ternura y amor en el transcurso de estos largos años.*

*A mi esposo Reinier por esperarme pacientemente.*

*A mis primos, en especial a Angelina por ser mi madre adoptiva en estos años.*

*A mi tía Siria por cuidar de mí.*

*A mis hermanos.*

*A todos los amigos que encontré en el camino, a Tamara, Doina, Aimé, Yadir, el Nene, Ulices y todos los dependientes del café.*

*Muchas gracias, además, a todas las personas que contribuyeron a que llegara con éxito hasta la meta.*

*Eduardo René Medina Pérez*

*A mis padres por estar siempre a mi lado.*

*A los amigos que me han dado ánimo, que me han brindado su cariño y amistad incondicional.*

*Muchas gracias, además, a todas las personas que contribuyeron a que llegara con éxito hasta la meta.*

**DEDICATORIA**

*Cecilia Dupotey Tussén*

*Dedico especialmente este trabajo a mis padres que tanto se han esforzado por mí, a mi esposo por estar siempre a mi lado y a mi hija Vania de las Mercedes por inspirarme a seguir hacia delante.*

*Eduardo René Medina Pérez*

*A mi familia por existir.*

*A mi sobrinita que amo.*

*A mi futura esposa.*

**RESUMEN**

La estrategia propone una guía para la gestión de la variabilidad en Líneas de Productos de Software (LPS), y con el objetivo de que los conceptos fueran aterrizados a un ejemplo real se consideró que la plataforma Atlas sería un buen punto de referencia ya que carece de algunos elementos teóricos relacionados con la gestión de la variabilidad que la pudieran dotar de capacidades para lograr mejores resultados con un funcionamiento semejante a una LPS. Se realiza un estudio acerca de las tendencias existentes presentadas por distintos autores para la gestión de la variabilidad en una LPS, en aras de lograr una propuesta de estrategia sustentada por fundamentos teóricos de actualidad.

La estrategia se dividió en dos fases: Ingeniería de dominio e Ingeniería de Aplicación. Durante la primera fase se realiza un análisis del domino donde se construyen los activos de software que soportan la línea que pueden ser algoritmos, componentes, especificaciones tecnológicas y de requisitos, planes, documentación y demás elementos con características reutilizables. La segunda fase se encarga directamente del proceso de desarrollo de las personalizaciones. En ambas fases se modela y analiza automáticamente la variabilidad a través de modelos y herramientas que según los autores consultados son elementos esenciales cuando se habla de LPS.

PALABRAS CLAVE:

Estrategia, gestión de la variabilidad, línea de productos de software, modelos de características, reutilización, variabilidad.

**ÍNDICE**

**ÍNDICE DE FIGURAS**

**ÍNDICE DE TABLAS**

**INTRODUCCIÓN**

Con el incremento de la demanda de software y su complejidad, es preciso buscar alternativas capaces de optimizar los procesos de desarrollo y aumentar la rentabilidad de cada producto desarrollado. Es por ello que muchos de los sistemas que se están realizando utilizan las Líneas de Productos Software (LPS) para poder alcanzar los niveles deseados de calidad y mejorar la productividad, ya que se basa en la creación de artefactos de alto nivel que son reutilizados durante el proceso de desarrollo de software.

Las LPS se definen como "*un conjunto de sistemas software, que comparten un conjunto común de características (features), las cuales satisfacen las necesidades específicas de un dominio o segmento particular de mercado, y que se desarrollan a partir de un sistema común de activos base (core assets) de una manera preestablecida*" (Clements, y otros, 2001). Con este nuevo enfoque de trabajo se reduce significativamente el tiempo de producción y los costes de desarrollo.

Al considerarse la reutilización como objetivo central en las LPS, toman peso adicional las actividades de integración de componentes y el mantenimiento de los repositorios de componentes, pues se gasta menos tiempo en desarrollar componentes nuevos y se dedica la mayor parte del tiempo a integrar los componentes desarrollados con anterioridad, según las distintas variantes definidas. Además, el desarrollo de los componentes específicos del nuevo producto es de acuerdo a la arquitectura de la LPS y a las especificaciones definidas en los mecanismos de variación que soporta.

La solución de muchos problemas requiere del acceso a diferentes tipos de información. Esta información puede ser representada geográficamente para un mejor entendimiento de los procesos en los que es manejada. Por tal razón surgen los Sistemas de Información Geográficos (SIG) que no son más que "*un sistema de hardware, software y procedimientos, diseñados para soportar la captura, el manejo, la manipulación, el análisis, el modelado y el despliegue de datos espacialmente referenciados (georeferenciados), para la solución de los problemas complejos del manejo y planeamiento territorial*" (Rhind, 1989).

El departamento Geoinformática de la Facultad 6, en la Universidad de las Ciencias Informáticas, se dedica al desarrollo de SIG, software para la minería y los recursos minerales. En dicho departamento se creó la Línea de Productos Software conocida como Aplicativos SIG buscando obtener los beneficios de los modelos de desarrollo de software basados en la reutilización. Si bien la línea ha sido

montada sobre la plataforma GeneSIG[1], que aporta los activos centrales; es importante explorar su funcionamiento basado en la plataforma Atlas[2], pues, ésta última aportaría mayor ligereza a los productos desarrollados por estar constituida por tecnologías ágiles.

En la LPS Aplicativos SIG se desarrollan las personalizaciones de una forma experimental, esto significa que aunque se ha montado la línea con resultados satisfactorios, por la puesta en el mercado de un significativo número de productos a la medida; al funcionamiento de la misma no se han incorporado algunos elementos teóricos relativos a la gestión de la variabilidad que la pudieran dotar de mayor capacidad para el éxito, pudiendo resultar en mayor productividad y menores tiempos de entrega.

Para llegar al estado de funcionamiento deseado de la línea, surge la necesidad de controlar las variaciones y configuraciones de los productos de software agrupados en familias. Teniendo en cuenta que, es primordial, haciendo uso de líneas de productos, crear diversos productos similares con el menor costo y tiempo posible. Además, se debe tener presente que la variabilidad es un mecanismo que nos permite generar distintas configuraciones de un mismo producto de software. Donde su principal problema radica en el instante donde debe modelarse y gestionarse dicha variabilidad cuando el número de puntos de variación se hace inmanejable. Esto es imprescindible para incrementar la productividad mediante una documentación precisa de las formas en las que pueden variar los componentes fundamentales y cómo se pueden utilizar en el desarrollo de productos específicos.

Se considera una limitante para la LPS, no disponer de un modelo de características para modelar su variabilidad. Considerando que trabajar orientado a características es una forma sencilla de representar propiedades de un producto.

Otros aspectos claves para el buen funcionamiento de la línea, tienen que ver con las herramientas y los procesos de apoyo que permiten gestionar y modelar la variabilidad. Pues, es muy difícil gestionar de forma manual grandes cantidades de puntos de variación, variantes e interdependencias en una LPS. Incidiendo en la necesidad de contar con herramientas que soporten dicha gestión y de este modo favorezcan el proceso de automatización de la línea.

[1] Plataforma Soberana GeneSIG proyecto mixto Geo Cuba - FAR- UCI para la personalización de SIG.

[2] Plataforma Atlas: estandarizada en los procesos de configuración y comunicación para la personalización de aplicaciones sustentada en tecnologías ágiles.

Otra dificultad lo constituye el hecho de que el Modelo de desarrollo basado en líneas de productos de software para SIG (Pantoja, 2012), que regula el funcionamiento de la LPS Aplicativo SIG y que es, en definitiva, el que se debe ejecutar para una LPS sobre Atlas, no está diseñado para realizar un tratamiento explícito en relación a la gestión de la variabilidad.

Por todos los argumentos abordados anteriormente se identificó como **problema a resolver:** las deficiencias en la gestión de la variabilidad de los productos desarrollados sobre una LPS basada en plataforma Atlas. Este problema se enmarca en el **objeto de estudio:** la gestión de la variabilidad en una LPS, que delimita el **campo de acción** los mecanismos de variación en la plataforma Atlas. Analizados los aspectos anteriores se establece como **objetivo general** desarrollar una estrategia para la gestión de la variabilidad de los componentes desarrollados sobre una LPS basada en plataforma Atlas.

Planteándose como **idea defender**: Con el desarrollo de una estrategia que aplique mecanismos de variación para la gestión de la variabilidad en una LPS desarrollada sobre la plataforma Atlas se facilitará la de personalización de los productos desarrollados.

Para cumplir con el objetivo trazado, es necesario llevar a cabo un conjunto de **tareas investigativas,** que permitirán, de manera sistemática y cronológica, avanzar en el desarrollo de la investigación:

1. Caracterizar la gestión de la variabilidad en las LPS.
2. Caracterizar la plataforma Atlas de acuerdo a su diseño arquitectónico.
3. Identificar los mecanismos de variación para LPS aplicables al desarrollo de productos sobre la plataforma Atlas.
4. Desarrollar una estrategia para la gestión de la variabilidad en una Línea de Producto de Software sobre la plataforma Atlas de acuerdo a los mecanismos de variabilidad definidos.
5. Realizar una herramienta de software que ilustre la aplicación de mecanismos de variación en la plataforma Atlas.
6. Validar los resultados mediante la utilización de un caso de estudio.

Entre los métodos de investigación que sostienen el presente trabajo se encuentran:

Dentro de los Métodos Teóricos:

**Método de análisis-síntesis:** se utilizó para el estudio de los conceptos empleados en la utilización del desarrollo de software basado en LPS, permitiendo la extracción de los elementos más importantes que se relacionan en este campo. Haciendo énfasis en cuanto a la reutilización del software y a la

gestión de la variabilidad.

**Análisis histórico- lógico:** se utilizó para conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a la ingeniería basada en reutilización; además de conceptos, términos y vocabularios propios del campo como componente, reutilización, activos, dominio, mecanismos de variación, puntos de variabilidad y variantes.

Dentro de los Métodos empíricos:

**Método de observación:** se emplea la observación natural en la recolección de información acerca de la plataforma Atlas, estudiando para ello como se realizan las personalizaciones SIG y dentro de esta actividad, enfocándose la investigación en las actividades que se llevan a cabo para realizar una correcta gestión de la variabilidad.

**Resultados esperados:**

Entre los resultados que se esperan de este trabajo de investigación están:

- ✓ Estrategia para la gestión de la variabilidad de los componentes desarrollados sobre la plataforma Atlas.
- ✓ La documentación que avala la investigación.

**Estructura de la investigación:**

La disposición de la información contenida en este documento está conformada por Resumen, Introducción, tres Capítulos (en los cuales se dejará constancia de toda la información referente a la investigación), Conclusiones, Recomendaciones, Bibliografía, Referencias Bibliográficas y Anexos. A continuación se expone una breve descripción de cada capítulo.

**Capítulo # 1 Fundamentación Teórica**: Se realiza una revisión bibliográfica sobre el tema de investigación para el estudio del estado del arte de la gestión de la variabilidad en Líneas de Producto Software, así como los conceptos fundamentales asociados.

**Capítulo # 2 Estrategia para la gestión de la variabilidad**: Se presenta la propuesta de la estrategia para la gestión de la variabilidad en la plataforma Atlas.

**Capítulo # 3 Validación de la estrategia**: Se muestra a través de un caso de estudio los resultados de aplicar la estrategia de gestión de la variabilidad, quedando evidencia de artefactos concretos relacionados con la utilización de mecanismos de variación y su soporte mediante la utilización de herramientas automáticas.

## CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

### Introducción

En este capítulo se presentará la fundamentación teórica de la investigación a través de la realización de un estudio sobre los antecedentes de las LPS; los conceptos fundamentales asociados al problema de investigación; lo referente a la gestión de la variabilidad y los modelos de características. Además, se hará un estudio de la arquitectura de la plataforma Atlas.

### 1.1. Antecedentes de las LPS

La reutilización de software ha sido uno de los enfoques mediante el cual las empresas han tratado las exigencias del entorno por alta productividad, alta calidad, costo razonable y rápida puesta en mercado de los productos de software; que se acrecientan ante la complejidad cada vez mayor de los negocios y los sistemas a integrar, el volumen y precisión de la información a procesar, la evolución acelerada de la tecnología de hardware y software unido a sus técnicas y plataformas de desarrollo que muchas veces implican recalificación constante de los profesionales. Siendo motivado además por el alto impacto que ejerce la industria del software en todas las esferas de la sociedad, pues sus aplicaciones influencian prácticamente en cualquier parte de la actividad humana, por lo cual es deseable garantizar y mantener una imagen y un ritmo que garantice el éxito de mercado.

Como se puede apreciar en la figura 1 los esfuerzos conocidos en este sentido datan desde la aparición de las subrutinas en 1960, los módulos en 1970, los objetos en 1980, los componentes en 1990 y los servicios en 2000, hasta llegar al muy novedoso enfoque de las líneas de productos de software a principios del 2000. Aunque existen indicios de líneas de productos de software desde los 70 por la mención que hace (Parnas, 1976) de los beneficios de tener una familia de sistemas compartiendo características entre ellos.

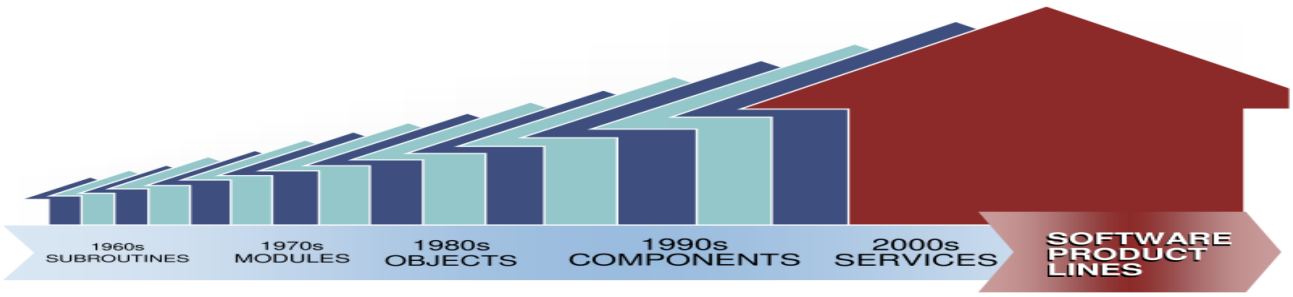


**Figura 1:** Evolución de la reutilización como estrategia de reutilización de software (Northrop, 2008).

### 1.2. Conceptos asociados al dominio del problema

#### Reutilización de software

"*La reutilización de software es el proceso de implementar o actualizar sistemas de software usando activos de software existentes*" (Sodhi, y otros, 1999).

Según Sommerville "*La reutilización es un enfoque de desarrollo de software que trata de maximizar el uso recurrente de componentes de software existentes*" (Sommerville, 2000).

Para (Krueger, 2006) la reutilización consiste en desarrollar elementos de software que puedan utilizarse más de una vez con la mínima cantidad de modificaciones, garantizando que al reutilizar un elemento de software, libre de defectos, implicará que el sistema que lo utilice no tendrá problema alguno en lo que respecta a dicho elemento.

Para Montilva la "Reutilización de software es el proceso de crear sistemas de software a partir de software existentes, en lugar de desarrollarlo desde el comienzo" (Montilva, 2006).

Estas definiciones consideran la reutilización como un enfoque o proceso de desarrollo de software. Su principal diferencia radica en el objeto de reutilización de elementos de software, componentes de software y activos. Implican la búsqueda de la calidad, el mínimo de trabajo, no reinventar la rueda; lo que evidentemente puede redundar en beneficios económicos.

**Líneas de Productos Software (LPS)**

Para (Clements, y otros, 2001) una línea de productos de software es un conjunto de sistemas intensivos de software que comparten un conjunto común y gestionado de características que satisfacen las necesidades específicas de un segmento de mercado particular o misión y que son desarrollados a partir de un conjunto común de activos centrales de una manera preestablecida. De ahí que la práctica de línea de productos de software consiste en el uso sistemático de activos centrales para ensamblar, instanciar, o generar múltiples productos que constituyen la línea de productos; comprende el rehúso estratégico y de granularidad alta[3].

Una línea de producto de software consiste de una familia de sistemas de software que tienen una funcionalidad común y alguna funcionalidad variable. Donde la funcionalidad común descansa en el

[3] Aquí el rehúso es planificado, habilitado y exigido, lo contrario de rehúso oportunista. Los activos base incluyendo esos artefactos que en el desarrollo de software son más costosos de desarrollar desde cero tales como requerimientos, modelo de dominio, arquitectura de software, modelos de rendimiento, casos de prueba y componentes. Todos estos activos son diseñados para ser rehusados y están optimizados para usarse en más de un único sistema. El rehúso es comprensible, planificado y beneficioso (Clements, y otros, 2001).

uso recurrente de un conjunto de activos de software reutilizables por todos los miembros de la familia (Gomma, 2002).

Según (Pohl, y otros, 2005) la ingeniería de líneas de productos de software es un paradigma para el desarrollo de aplicaciones de software (productos de software y sistemas intensivos de software) usando plataformas[4] y personalización en masa[5]. Diferenciar aquí los sistemas intensivos de software como el software que es embebido, es decir, que se despliega generalmente como parte de piezas de hardware específicas. El desarrollo usando plataformas implica que como parte de la reutilización se planifica proactivamente la construcción de partes reusables y se rehúsa lo que ha sido construido para ello. La personalización en masa conlleva emplear el concepto de gestión de la variabilidad, por ejemplo, en base a las diferencias y a las similitudes en las aplicaciones de la línea de productos tiene que ser modelada de una manera común.

Para (Krueger, 2006) las líneas de productos de software se refieren a técnicas de ingeniería para crear un portafolio de sistemas de software similares, a partir de un conjunto compartido de activos de software, usando un medio común de producción.

Las líneas de productos posibilitan la creación de un conjunto de sistemas en un mismo dominio que comparten una serie de características comunes. Debido a que la variabilidad del software permite discriminar y configurar los productos software a partir de una arquitectura común. Una línea de productos consiste en un ciclo de vida dual, en la que en la primera fase (Ingeniería del Dominio) se construye una arquitectura común a la línea de productos y un conjunto de componentes reutilizables denominados "*core assets*". En la segunda fase (Ingeniería de Aplicaciones), se construyen los productos software a partir de la personalización de la arquitectura, mediante técnicas de variabilidad de software, y los componentes *core*, que constituyen gran parte de la funcionalidad del producto final. Está fundamentada en la reutilización de software y asume la existencia de una industria de partes de software (Barbosa León, 2010).

A las LPS se le atribuyen ventajas notables, no viene dado solo por razones de mejoras en la reutilización y las formas de producción. En (Montilva, 2006) se enumeran las siguientes ventajas:

[4] Una plataforma de software es un conjunto de subsistemas e interfaces que forman una estructura común a partir de la cual un conjunto de productos derivados pueden ser eficientemente desarrollados y producidos (Meyer, y otros, 1997).

[5] ¨Es la producción a gran escala de bienes personalizados según las necesidades individuales de los clientes¨ (Davis, 1987).

- Reducción del tiempo para el mercado.
- Incremento de la productividad.
- Reducción de costos en la producción.
- Mejor retorno de inversión.
- Aumento de la calidad de los productos.

**Variabilidad del software**

A través de la identificación y gestión de los aspectos comunes y las variaciones de un conjunto de artefacto de sistemas; se logra desarrollar sistemas intensivos en software utilizando plataformas y personalización en masa en una LPS. En el enfoque de una línea de productos, la variabilidad proporciona la flexibilidad necesaria para la diferenciación y diversificación de productos. La misma se refiere a la capacidad de un artefacto a configurar, personalizar, extender o cambiar para su uso en un contexto específico (Bachmann, y otros, 2005).

Para González- Baixauli se define la variabilidad como la habilidad de cambio o personalización de un sistema (González- Baixauli, y otros, 2004). Fabricia (Carneiro, 2011), define los conceptos fundamentales en el contexto de una LPS (ver anexo 4), entre ellos variabilidad, que son las "*propiedades que varían de un producto al otro en la misma familia".* Para (Barbosa León, 2010), la variabilidad nos permite discriminar y configurar los productos software a partir de una arquitectura común. De esta manera, resulta posible personalizar y adaptar los productos software y los componentes reutilizables para diferentes necesidades de los usuarios dentro del ámbito de la línea de productos.

En resumen, la variabilidad del software trata de identificar las partes comunes y variables de los sistemas para poder construir familias de sistemas relacionados y con características similares.

La variabilidad puede ser expresada en términos de características y es tratada a través de puntos de variabilidad plasmados en un Modelo de Características También, ésta es descrita en términos de puntos de variación y variantes. Donde un punto de variación, según Barbosa, es un área del sistema afectada por la variabilidad, mientras que una variante son las alternativas de un punto de variación. Para Jiménez, los puntos de variabilidad se definen como la localización en el software a cualquier nivel de detalle en el que se puede dar la variación.

Estas son técnicas que se utilizan para enlazar los modelos de características con los diferentes

niveles de diseño e implementación que se requieren en un sistema de LPS. Pueden introducirse en diferentes niveles de abstracción durante la construcción del software (Jiménez, 2010) entre los que se encuentran: la descripción de la arquitectura, las etapas de diseño, el código fuente, el código compilado y el código ejecutándose. Para cada nivel de abstracción, pueden, los puntos de variabilidad, encontrarse en uno de estos estados: implícito (cuando se introduce en un modelo de característica), diseñado (cuando se diseña una solución para manejar el punto de variabilidad) o enlazado (cuando un punto de variabilidad se enlaza a una variante).

Un punto de variabilidad, además, puede estar abierto o cerrado a extensión en cada uno de los niveles de variabilidad. Un punto abierto significa que se pueden añadir nuevas variantes, por el contrario, un punto cerrado, no se pueden añadir nuevas variantes. Si se quiere diseñar una LPS dinámica que evolucione durante el tiempo de ejecución, los puntos de variabilidad deben estar abiertos a nuevas variantes. Otro elemento de análisis dentro de la variabilidad es el análisis de variantes; en donde, a partir del modelado de la variabilidad, se analiza cual es la variante que mejor se adapta a las necesidades de un usuario o a un determinado producto de la familia. Por lo general se realiza de forma manual en cada punto de variabilidad, la parte que más convenga.

**Gestión de la variabilidad de software**

La configuración de aspectos variables en el software, es una tendencia que conduce a una situación donde la complejidad de manejar una gran cantidad de variabilidad se convierte en una preocupación básica que necesita ser tratada. Por tal razón las LPS toman en consideración la importancia de la gestión de la variabilidad del software.

La gestión de la variabilidad[6] abarca las actividades de representar de forma explícita la variabilidad de artefactos software en todo el ciclo de vida, gestionando dependencias entre las diferentes variabilidades y dando soporte al uso de las variabilidades para construir e incrementar una familia de sistemas de software (Schmid, y otros, 2004). Bosch, Sinnema, Deelstra, Florijn, al abordar el tema de la gestión refieren que se trata de tareas muy complejas y difíciles, las cuales deben ser apoyadas por técnicas , métodos y herramientas apropiados. Además, distinguen como característica principal en una LPS el identificar sistemáticamente y gestionar adecuadamente las variabilidades entre los

[6] Es lo que diferencia la ingeniería de LPS de la ingeniería de software tradicional. Se establece como uno de los pilares dentro de la concepción de LPS e implica una evolución en el proceso de desarrollo de software. Se define como la gestión de las diferencias entre los productos de una línea de productos y esencial a la hora de construir una línea de producto (Jiménez, 2010).

diferentes sistemas de una familia.

Juega un papel fundamental en las fases de Ingeniería de Dominio e Ingeniería de Aplicación. En la primera, se introducen todos los tipos de mecanismos de variabilidad en los artefactos software, por lo que en la segunda fase se puede hacer elecciones específicas (extensión, cambio, la personalización o configuración) que se adapten a la funcionalidad y la calidad requerida del producto que se está creando.

Por otro lado, la gestión de la variabilidad también tiene que tratar con un número de complicaciones que surgen durante la ingeniería de aplicaciones. En primer lugar, no todas las elecciones en los artefactos de la familia de productos se pueden hacer independientemente uno de otro. La elección de un componente articular, por ejemplo, puede requerir o excluir la presencia de otros componentes.

**Modelo de características**

Las características se definen en (Kim., y otros, 1998) como una abstracción funcional distintiva e identificable que puede ser empaquetada, probada, distribuida y mantenida. En (Bosch, 2000) es, "*una unidad lógica de comportamiento que se especifica por un conjunto de requisitos funcionales y de calidad*". En otras palabras, las características son abstracciones de los requisitos.

Las diferencias entre los productos pueden ser discutidas en término de características. Con el objetivo de poder describir todos los productos que una LPS es capaz de producir, surgen los modelos de características y son considerados una de las contribuciones más importante para el modelado de LPS. Además, representan gráficamente una línea de producto, a través de las posibles combinaciones entre las características.

Entre los modelos más utilizados para representar la variabilidad se encuentra el FODA, que tiene como objetivo principal la identificación de características relevantes de sistemas de software que pertenecen a un dominio. El modelado de características es una actividad, a través de la cual, se define un árbol donde quedarán especificados los elementos comunes y variables del dominio. Y además, el modelado está compuesto a su vez por dos elementos fundamentales: las características[7] y las relaciones existentes entre ellas (jerárquicas[8] o no jerárquicas). Los modelos de características

[7] Aquí las características están organizadas en una estructura jerárquica en forma de árbol y una de ellas es la raíz del árbol, representando al sistema como un todo.

[8] Es definida entre una característica padre y sus características hijas. Una característica hija solo puede hacer parte de los productos en los que la característica padre aparece.

organizan el conjunto de características jerárquicas mediante las siguientes relaciones:

- Obligatoria: indica que cuando la característica padre hace parte de un producto particular, la característica hija también debe hacer parte del producto.
- Opcional: indica que cuando la característica padre parte de un producto particular, la característica hija, puede o no, ser incluida en el producto.
- Alternativa: es la relación entre una característica padre y un conjunto de características hijas que indica que cuando la característica padre hace parte de un producto particular, sólo una de las características del grupo de hijas debe hacer parte del producto.

Y las relaciones no jerárquicas pueden ser de dos tipos:

- Excluye: una característica X excluye Y si significa que sí la característica X es incluida en el producto, la característica Y no debe ser incluida, y viceversa.

- Requiere: una característica X requiere Y significa que si la característica X es incluida en el producto, la característica Y también debe ser incluida, pero no viceversa.

Otros autores ((Griss, y otros, 1998); (Kim., y otros, 1998)) hacen referencia a los modelos Feature-RSEB[9], FORM[10], PLUSS[11] que constituyen, en general, extensiones del FODA. En todos ellos, las características capturan la parte común y variable organizándose por medio de diagramas jerárquicos, AND/OR. Donde en estos diagramas, los nodos AND dan la parte común y los OR la parte variable. De esta forma, son utilizados para definir la arquitectura de referencia del sistema y los componentes reutilizables instanciables durante el desarrollo de aplicaciones con variabilidad (Kang, y otros, 1990).

**Estrategia**

[9] Del acrónimo *Reuse-Driven Software Engineering Business,* traducido al español (Método de reutilización orientado a características con referencia específica al dominio), añade una nueva relación entre una característica padre y una característica hija; puede ser en forma de árbol o grafo.

[10] Del acrónimo *Feature-Oriented Reuse Method,* traducido al español (Método de Reuso Orientado a Características), considera la importancia de modelar la variabilidad en otras fases del desarrollo además de la ingeniería de requisitos.

[11] Del acrónimo *Product Line Use case modeling for System and Software engineering,* basado en *Feature-RSEB* donde se combinan los diagramas de características con los diagramas de casos de uso. Un conjunto de características compone un paquete de caso de uso y de esta forma es posible visualizar variantes en la especificación de caso de uso por medio de los modelos de características.

Una estrategia es un conjunto de acciones o pasos planificados que se llevan a cabo para lograr un determinado fin o una misión. En otras palabras constituye la ruta a seguir para alcanzar los propósitos, objetivos y metas planteados en corto, mediano y largo plazos.

### 1.3. Funcionamiento de LPS

El modelo básico de una LPS según Montilva , está agrupado en cuatro actividades fundamentales: la entrada de activos de activos software, el control de Modelos de Decisión ( se describen los aspectos variables y opcionales de los productos de la línea) y Decisiones de Productos (donde cada producto de la línea es definido por un conjunto de decisiones), el proceso de producción (donde se establece los mecanismos o pasos para componer y configurar productos a partir de los activos de entrada y las decisiones del producto se usan para determinar que activos de entrada utilizar y como configurar los puntos de variación de esos activos) y la salida de productos de software (el conjunto de todos los productos que pueden o son producidos por la línea de productos).

A diferencia, otros autores ((Díaz, 2010); (Clements, y otros, 2001); (Barbosa León, 2010); (Klaus, y otros, 2005)) agrupan las actividades de una LPS en dos procesos ingenieriles: Ingeniería de Dominio e Ingeniería de Sistemas[12]. Destacar que esta división no difiere de la anterior, se puede establecer una equivalencia entre las actividades de desarrollo de activos y la ingeniería de dominio y entre el desarrollo de productos y la ingeniería de sistemas (ver ilustración 2). Donde el primer proceso se encarga de la **Ingeniería de Dominio**. Este es el responsable de desarrollar los elementos comunes al dominio: estudiar el dominio, definir su alcance (requisitos) dentro del mercado objetivo de la LPS, definir las características, implementar los activos centrales reutilizables y su mecanismo de variabilidad, y establecer cómo es el plan de producción. El segundo proceso se encarga de la **Ingeniería de Producto**. Este es el responsable de desarrollar los productos para clientes concretos, a partir de los recursos basados no en los requisitos del dominio, sino en requisitos concretos de clientes. Para ello este segundo proceso utiliza recursos creados por el proceso anterior.

[12] Puede aparecer, además, como ingeniería de aplicación o como ingeniería de producto.

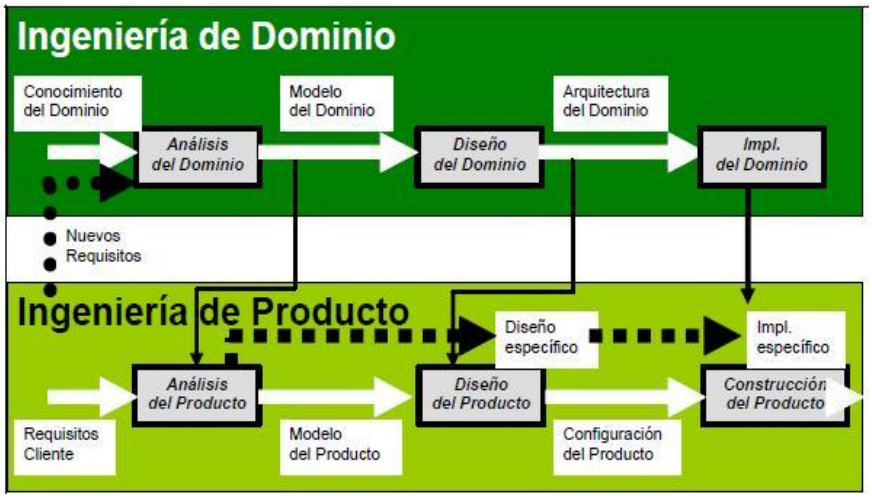


**Figura 2:** Procesos en Líneas de Procesos Software (Díaz, 2010).

Al analizar las definiciones de estos autores, se establece como la actividad de Ingeniería de dominio se retroalimenta, en todo momento, de la información que recoja del equipo de producto de software durante la producción de productos. Esto significa que durante la ingeniería de dominio se recogen iterativamente los requisitos comunes para toda la LPS. Estos requisitos se expresan típicamente en forma de características[13].

**Dominio y Familia de productos de software**

Lograr construir aplicaciones reutilizando componentes requiere un enfoque de desarrollo de familias de productos para un dominio específico, en lugar de un enfoque de construcción de aplicaciones independientes.

Un dominio según Northrop, "…satisface las necesidades específicas de un segmento de mercado o misión…". Es definido como "*un bloque de conocimiento especializado, un área de experticia o un conjunto de funcionalidades relacionadas*" (Northrop, 2002).

Una familia de productos de software (LPS) se define como "un conjunto de sistemas de software que satisfacen necesidades específicas de un segmento del mercado" (Garcés, y otros, 2007).

Una familia de productos es: "el conjunto de productos diferentes que pueden ser producidos desde un diseño común, activos compartidos y mediante un proceso de ingeniería de aplicaciones" (García, y otros, 2002).

[13] Estas características ayudan no solo a que los clientes puedan distinguir unos productos de otros, sino que dirigen así mismo el desarrollo de código que implemente también esas características.

Los miembros de la familia comparten aspectos comunes tales como:

- Un diseño arquitectónico común.
- Un conjunto de componentes reutilizables.
- Capacidades y servicios comunes.
- Tecnologías comunes.

### 1.3.1. Activos de software reutilizables

En las LPS se les llama activos a un producto de software diseñado expresamente para ser utilizado múltiples veces en el desarrollo de diferentes sistemas o aplicaciones, es decir que se reutilizan a lo largo de la vida de la línea (Montilva, 2006). Los activos no son solamente componentes ejecutables. También podemos encontrar otros artefactos como son:

- Un componente de software.
- Una especificación de requisitos.
- Un modelo de negocios.
- Un algoritmo.
- Un patrón de diseño.
- Una arquitectura de dominio.
- Un esquema de base de datos.
- Una especificación de prueba.
- La documentación de un sistema.
- Un plan.

**Componentes de software**

Se utiliza la misma definición de componente de Clemens Szyperski, “un componente de software es una unidad de composición con interfaces especificadas contractualmente y solamente dependencias explícitas de contexto” (Szyperski, 2002).

Para (Brown, 2000) un componente de software reutilizable es “una pieza de software funcional que es liberada independientemente de otras y que proporciona acceso a sus servicios a través de sus interfaces”. Puede ser liberado desplegado e instanciado, independiente de otros.

Al desarrollar componentes se debe tener en cuenta que los mismos sean aptos para aplicaciones de LPS, evitando de este modo, que el cliente quede insatisfecho y que los productos no tengan la

capacidad de cambiar rápidamente. Tener en cuenta, además, que los componentes no sólo deben cumplir con los requisitos de comportamiento y calidad, sino, que también, debe ser de manera planificada de antemano para permitir a los constructores de productos crear instancias de forma rápida y fiable en las formas correctas para productos concretos. Un componente que no se pueda adaptar cuando se necesite, es el resultado de la selección de forma incorrecta del mecanismo de variación y el exceso de variabilidad puede impedir que los mismos puedan entenderse suficientemente bien como para ser utilizados con eficacia. Las características esenciales de un componente de software reutilizable se detallan en el anexo 1.

### 1.3.2. Arquitectura de software en LPS

La disciplina de la arquitectura de software fue establecida durante la última década del siglo XX. Específicamente en la década de 1990, Dewayne Perry y Alexander Wolf preveían que esta década sería la "*década de la arquitectura de software*" (Perry, y otros, 1992). Es una disciplina todavía joven dentro de la ingeniería de software. A pesar de esto, ha adquirido gran importancia tanto en proyectos de sistemas tradicionales como en las LPS; debido a que ha sido fuertemente desarrollado durante las últimas décadas. Según varios autores se define como: "*la organización fundamental de un sistema dada por sus componentes, las relaciones entre ellos y el ambiente, y los principios que orientan su diseño y evolución*"(IEEE, 2000).

Otros autores((Perry, y otros, 1992)) suelen dar más detalles en sus definiciones; y se concentran en caracterizaciones de los artefactos que se involucran en la arquitectura de software.

En muchos casos, los autores ((Fielding, 2000);(Webber, y otros, 2010)), utilizan "vistas" para representar varios aspectos de la arquitectura del sistema y poder analizarlo respecto a un grupo de atributos deseados.

Para (Córdova Bayas, 2012), la arquitectura de una LPS es una arquitectura genérica, es decir, que describe la estructura de toda la familia de productos y no solamente la de un producto en particular, además, captura los aspectos comunes y variables de una familia de productos de software. Esto se explica, cuando los aspectos comunes de la arquitectura son capturados por los componentes de software que son comunes a toda la familia y los aspectos variables de la arquitectura son capturados por los componentes de software que varían entre los miembros de la familia. También es denominada como arquitectura de dominio.

En la arquitectura se concentran muchas propiedades que exhiben todos los productos de la línea.

También en ella se regula como pueden variar los productos: cuales son las posibilidades de variación y que es fijo en todos los productos. Además ,se puede observar que uno de los elementos distintivos de la ingeniería para líneas de productos software es la atención que se le brinda a la gestión de la variabilidad.

Al revisar diferentes literaturas como son ((Bass, y otros, 1997); (Clements, y otros, 2005) ;(Zubrow, y otros, 2003)) se establece un consenso sobre que la arquitectura es uno de los activos imprescindibles dentro de en una LPS. Esta indica los elementos de diseño que son comunes y también cómo se soporta la variabilidad de cada producto. Da forma a todos los productos y regula la variabilidad de forma manejable. "*De cierta manera, la arquitectura evoluciona a la par de los activos del dominio e incluye componentes que sean necesarios para garantizar la variabilidad. También debe asegurar que existan mecanismos de variabilidad en los productos*" (Krueger, 2006).

**Mecanismos de variabilidad**

De acuerdo con (Mikael Svahnberg and Jan Bosch, 2003), los mecanismos de variabilidad son: "*técnicas disponibles para introducir variabilidad en la línea de producto de software*".

Clements (Clements, 2006), define como mecanismos de variabilidad: a las construcciones que llevan a cabo las variaciones a nivel de los artefactos. Además, plantea que se han publicado catálogos de estos mecanismos, que han llegado a convertirse en una variedad confusa y que pueden estar organizados por niveles.

- Nivel de requerimientos (modelos de características, extensiones de los casos de uso).
- Nivel de aplicación (configuradores o generadores de programas).
- Arquitectura (plug-in o replicación o reemplazo de componentes).
- Nivel de diseño (aspectos).
- Nivel de implementación (herencia o parametrización).
- Tiempo de ejecución (programación reflectiva o condicional).

A través de los mecanismos de variabilidad pre planificados, cada producto es formado por los componentes aplicables tomados del repositorio de activos comunes, personalizándolos según sea necesario. Estos mecanismos de variabilidad se pueden clasificar de acuerdo con el punto en la línea de ciclo de vida del producto en el que se llevan a cabo la resolución de la variabilidad, que también se conoce como el binding-time.

- **Herencia**: la especialización se lleva a cabo mediante la modificación o no adición a las

definiciones existentes. Se realiza en el momento de la definición de la clase, ejemplo: Suplantación de un método heredado de una clase en un componente.

- **Framework**: en este mecanismo la variabilidad está documentada habitualmente como patrones de diseño ya que los framework pueden interpretarse como colecciones de las implementaciones de ciertos patrones de diseño
- **Puntos de extensión**: un uso de un sistema puede ser definido mediante la adición de otro uso. Se realiza en tiempos de los requisitos. Ejemplo: se agrega nueva funcionalidad o comportamiento a un componente.
- **Parametrización**: una definición funcional está escrita en términos de elementos no unidos, que se suministra al uso real que se haga de definición. Se realiza en tiempo de implementación del componente. El comportamiento de un componente puede ser parametrizado a tiempo de diseño y definido a tiempo de implementación. Ejemplos: Macros o "*Templates".*
- **Configuración**: un recurso separado, tales como archivo, se utiliza para el componente especializado. Se realiza anterior al tiempo de ejecución. Selección o no de los componentes de la arquitectura que pudieran conformar el nuevo producto.
- **Selección a tiempo de compilación**: la implementación de una funcionalidad es seleccionada, entre varias posibles, al momento de la compilación del componente o de la aplicación.
- **Usos**: un uso de un sistema puede ser definido mediante la adición o la definición de otro uso. Se realiza en tiempo de los requisitos.
- **Planilla de instancias**: una especificación de tipo está escrita en términos de elementos no unidos, que se suministran al uso real que se haga de la especificación. Se realiza en tiempo de implementación del componente.
- **Generación**: una herramienta produce definiciones de entrada de usuarios. Se realiza antes o durante el tiempo de ejecución.

### 1.4. Modelado de la variabilidad

En cuanto a las técnicas que se utilizan para el estudio de la variabilidad, el principal esfuerzo recae en encontrar las partes comunes y variables dentro de los sistemas (González- Baixauli, y otros, 2004).Dicha tarea se define como modelado de la variabilidad y esta es la propiedad que diferencia el desarrollo de líneas de producto del desarrollo de software tradicional. Entre las técnicas más

conocidas para modelar la variabilidad de LPS se encuentran los modelos de características, que están mayormente orientadas hacia el diseño. Los métodos más utilizados son los basados en características como FODA[14] como una forma de describir una LPS.

**Otras técnicas de modelado de la variabilidad en LPS**

**OVM[15]**: Es una metodología para el modelado de la variabilidad en la LPS, que propone un modelo de variabilidad separado de los artefactos de línea de producto. Solo las características variables son documentadas en el modelo y cada una de ellas está relacionada con el respectivo artefacto de software que la realiza. Posee dos elementos de primera clase: punto de variación y variante. Donde en OVM, un punto de variación es la definición de lo que varía. Además, indica el componente que varía o la propiedad del componente que varía y la variante; define cómo el punto de variación varía e indica las posibles instancias de un ítem variable o de la propiedad del ítem que varía. En la relación entre los elementos punto de variación y variantes, cada punto de variación debe estar asociado a por lo menos a una variante y cada variante a un punto de variación. Esta relación puede ser de tres tipos: obligatoria, opcional o alternativa.

**COVAMOF**: "*es un framework[16] para el modelado de LPS que modela la variabilidad en términos de puntos de variación y dependencias en diferentes niveles de abstracción*" (Klaus, y otros, 2005). La jerarquía de LPS es modelada en tres niveles de abstracción: características, arquitectura e implementación. Cada punto de variación ofrece un conjunto de opciones que pueden ser del tipo: variante o valor. La opción del tipo valor se usa para representar parámetros para una determinada propiedad del producto, mientras la opción variante puede ser tipo de dato cualquiera. COVAMOF define punto de variación como la representación del sitio en el que se puede hacer una elección en una línea de producto. Donde los puntos de variación pueden ser de cinco tipos:

- Valor: las opciones de este punto de variación son un valor dentro de un rango definido.
- Opcional: cero o una de las variantes hacen parte del producto.
- Alternativo: exactamente una de las variantes hacen parte del producto.
- Opcional variante: cero o más de las variantes hacen parte del producto.

[14] Del acrónimo *Feature Oriented Domain Analysis,* traducido al español como: Análisis del Dominio Orientado a Características. Primer modelo de característica propuesto en 1990 por (Kang, y otros, 1990) y ha sido utilizado para extraer las similitudes y variabilidades de un dominio.

[15] Del acrónimo *Orthogonal Variability Modeling,* traducido al español como Modelado de la Variabilidad Ortogonal. Metodología propuesta (Eriksson, y otros, 2005) como parte del proyecto PRIME llevado a cabo por el grupo de investigación Software "*Systems Engineering (SSE)"*de la Universidad de Duisburg-Essen.

[16]*Framework:* marco de trabajo.

- Variante: una o más de las variantes hacen parte del producto.

**K. Schimid e I. John**: proponen una técnica para el modelado de la variabilidad en la que las características variables son capturadas en modelos de decisión. Se caracteriza por ser independiente de notación, es decir, toda la información sobre la variabilidad está separada de los artefactos de software de la línea de producto, y por modelar la variabilidad en un modelo integrado abarcando todas las fases de desarrollo de software (Schmid, y otros, 2004). Un modelo de decisiones describe los efectos de la variabilidad y los mismos son:

- Interacciones. Un mecanismo para describir interacciones entre las decisiones.
- Relaciones. Un mecanismo para describir las relaciones entre los puntos de variaciones en los artefactos y las decisiones específicas.
- Tipos de variaciones. Un conjunto común de diferentes tipos de selectores, por ejemplo, opcional y alternativo.
- Mapeo Específico. Un mapeo acompañado del tipo de selector para una notación específica, el que expresa los puntos de variaciones entre los artefactos.

**DecisionKing**: es una herramienta para el modelado de la variabilidad propuesta por (Dhungana, y otros, 2007) para diferentes dominios y organizaciones. Los artefactos y las decisiones son los elementos principales del modelo. Los puntos de variaciones son capturados como decisiones y las dependencias entre los artefactos y decisiones son modelados de forma explícita. Entre los artefactos se han definido dos tipos de dependencias: estructural donde siempre que un artefacto hace parte de otro artefacto o contribuye de alguna manera para su existencia; y funcional cuando un artefacto requiere otro artefacto para su funcionamiento, ejemplo cuando un componente sirve de entrada para otro componente.

Las decisiones pueden ser vistas como puntos de variaciones en los artefactos. Las cuales son relacionadas unas con las otras por medio de dos tipos de dependencias: jerárquica donde las decisiones deben ser tomadas de acuerdo con cierta orden y dependencia lógica, donde las consecuencias de tomar una decisión, particularmente, son reglas de negocios que deben ser chequeadas antes y después de una decisión.

**VSL**[17] (Benavides Cuevas, 2005): propuesta para el modelado de LPS en la cual se propone un modelo general para la variabilidad, en el que considera que la variabilidad en líneas de producto debe ser abordada en dos niveles de abstracción: el nivel de especificación, donde se tratan las

[17] *Variability Specification Language.*

variabilidades visibles al cliente, sin tener en cuenta los detalles de su realización. En este nivel se documenta de forma abstracta las características de la línea de producto y se define la variabilidad como elemento central; y el nivel de realización, donde la variabilidad especificada en el nivel de especificación es analizada y tratada. El elemento central es el punto de variación, el cual representa el sitio en el artefacto donde se va a realizar la variabilidad expresada en la especificación. En los puntos de variación también se describen las acciones que deben ser llevadas a cabo cuando una opción es hecha.

Se asocia también a los puntos de variación los mecanismos de software, como selección, generación y sustitución. Los puntos de variaciones se separan en dos tipos: estáticos; que consideran la variabilidad *pre-runtime*[18], y dinámicos; que consideran la variabilidad *runtime*[19].

La comunicación entre los dos niveles se da por medio de la relación "*implementa*". Esta relación entre el elemento variable y el punto de variación describe como el punto de variación debe ser configurado basado en las variantes elegidas. Está basado en el lenguaje XML y puede ser considerado como un lenguaje macro. No posee una notación gráfica para representar los elementos del modelo. Especifica en los artefactos los puntos de variaciones y la variabilidad que afecta a este punto.

**¿Qué técnica utilizar para modelar la variabilidad?**

Actualmente hay varias maneras de modelar la variabilidad, sin que ninguna prevalezca por encima de las otras todavía. Estas técnicas se dirigen a apoyar la gestión de la variabilidad durante la obtención de productos. Tienen por objeto representar la variabilidad en una familia de productos, de tal manera que los ingenieros de software pueden manejar la variabilidad más fácilmente durante la derivación producto. Tienen similitudes y diferencias en cuanto a los aspectos fundamentales que son importantes en el manejo de la variabilidad, es decir, en términos de conceptos de modelado y en términos de las herramientas que los apoyan. Estas diferencias hacen que cada técnica de variabilidad sea adecuada para una situación en particular. Varios autores ((Becker, 2002); (Capilla & Dueñas, 2002); (Krueger, 2002); (Salicki & farcet, 2002)), al analizar la relación entre variabilidad y característica han llegado a la conclusión de que la variabilidad se puede identificar más fácilmente si el sistema se modela utilizando el concepto de característica.

[18]pre-runtime: antes de ser compilado.

[19]Runtime: en ejecución, corriendo.

Teniendo en cuenta este criterio, se determinó utilizar en la investigación, para modelar la variabilidad, el FODA que es el método más extendido. Con él, se representan las relaciones a través de las características de sistema.

### 1.5. Herramientas para el modelado de características

La variabilidad en LPS es cada vez mayor, los modelos de variabilidad pueden tener miles de características. Además, tales características presentan dependencias complejas entre ellas. Por lo tanto, un apoyo automático se hace necesario para tratar dichos modelos. El objetivo del análisis automático es auxiliar a los analistas, diseñadores y técnicos en el desarrollo de la línea de productos. Este análisis se lleva a cabo por medio de operaciones que se ejecutan sobre el modelo que representa la línea de productos.

Para llevar a cabo este análisis automático es necesario contar con alguna herramienta que agilice este proceso. Hasta la fecha, existen multitud de herramientas para el modelado de características, la mayoría fueron realizadas por distintas universidades como proyectos de investigación y publicadas hace poco tiempo. La mayoría coincide en permitir crear modelos de características, otros permiten hacer solo configuraciones gráficamente, algunos solo validan los modelos, cualquiera de los casos estas herramientas presentan alguna que otra limitación.

**FeatureIDE**

FeatureIDE (Kastner, y otros, 2009) está desarrollado por la University of Magdeburg de Alemania. Es un IDE basado en Eclipse que integra AHEAD, FeatureC ++ y herramientas de FeatureHOUSE como herramientas de composición y compiladores entre otros. Es tanto un editor gráfico como textual, puesto que la construcción de los modelos de características está basada en la gramática GUIDLS. Permite configuraciones y la creación de restricciones avanzadas. Utiliza como resolutor SAT. Como defectos podemos citar que no dispone de características clonables, ni la posibilidad de referencias entre características y que no dispone de atributos.

**S2T2**

S2T2 (Botterweck, y otros, 2009) es el resultado de la investigación de las líneas de productos software por Lero (Centro de investigación de Ingeniería del Software de Irlanda). Es una aplicación de Java Open- Source independiente que permite configuraciones y comprobación de restricciones básicas. El modelo está basado en una gramática y utiliza como resolutor SAT. Sorprendentemente no dispone de un editor gráfico para modelar, únicamente permite realizar configuraciones para un

modelo textual previamente dado. No dispone de características clonables, referencias, atributos o restricciones avanzadas.

**Requiline**

Requiline (Thomas von der Maben and Horst Lichter, 2003) está desarrollada por el grupo de investigación “Software Construction 13” de la Universidad RWTH Acchen de Alemania. El proyecto está actualmente abandonado desde 2005. Es una aplicación para Windows que requiere de la conexión con una base de datos. Es gratis y de baja facilidad de uso. El editor gráfico está basado en la notación FORM. Dispone de un validador de restricciones consistente (aunque no permite restricciones avanzadas), gestor de usuarios con diferentes vistas y una interfaz XML. Las características disponen de atributos y se pueden realizar configuraciones entre otras propiedades. Es muy completa en algunos aspectos pero ha dejado de lado algunas características realmente importantes, como la usabilidad o algunos aspectos relacionados con el modelado de características.

**FAMA**

FAMA (Trinidad, y otros, 2008) está desarrollado por un equipo de la Universidad de Sevilla. Permite el análisis automatizado de los modelos de características integrando algunos resolutores más comúnmente propuestos (BDD, SAT, CSP), siendo esta una característica poco común. Dispone de un plugin de Eclipse gráfico desarrollado en EMF bajo la licencia Open- Source. FAMA está trabajando conjuntamente con pure::Variants para integrar sus herramientas. El proyecto funciona actualmente y la última versión se lanzó en agosto del 2009. Entre las ventajas que brinda el framework FAMA se encuentran su base formal, que garantiza una semántica fiable; su abstracción que permite el análisis de modelos distintos de los modelos de características; su capacidad de analizar modelos de características extendidos, en lo que se incluyen atributos; y el uso de varios resolutores distintos en su implementación. Por el contrario, las restricciones que permite son básicas (solamente implicación y exclusión).

Luego de analizar las herramientas anteriores, se culminó hacer uso de FAMA por las diversas ventajas que ofrece. Como información adicional, en el sitio de FAMA se puede encontrar información especializada además de contar con el criterio y la ayuda de especialistas. Para su descarga se puede acceder al sitio en el cual aparecen todas sus versiones (FAMA, 2010). FAMA Framework (FAMA-FW)(Benavides, 2008) se ha desarrollado con el propósito de que otras personas integren sus técnicas de razonamiento automático.

El proceso general para automatizar el análisis de los modelos que describen las LPS empleando FAMA se esboza en la Figura1. Inicialmente, el modelo que describe la línea de productos es traducido en una representación lógica, como por ejemplo, lógica proposicional (LP), programación con restricciones (PR), lógica descriptiva (LD) o múltiples paradigmas (MP) luego se realizan las operaciones a través de los resolutores (Sat4j[20], JavaBDD[21], JaCoP[22]) que incluye el plug-in y de esta forma se obtienen los resultados contenidos en el modelo (Carneiro, 2011).


**Figura 3:** Proceso para el análisis automático de modelos de características.

**Operaciones ejecutadas por FAMA (software para el análisis automático de los modelos de características):**

- **Determinan si un producto es válido para una LPS**: Esta operación toma como entrada un modelo característico y un producto (un conjunto de características) y retorna un valor determinando si el producto pertenece al modelo o no.
- **Determinan si un modelo es válido**: Esta operación toma como entrada un modelo y retorna un valor determinando si tal modelo es válido o no. Un modelo es válido cuando representa por lo menos un producto. Un modelo de característica solo puede ser inválido cuando se hace el uso incorrecto de las relaciones no jerárquicas entre las características, generando de esta manera contradicciones en el modelo.
- **Obtienen todos los posibles productos**: Esta operación toma como entrada un modelo y retorna todos los productos válidos representados por dicho modelo. [ejemplo Px= {Atlas. Navegación, Localizar Persona, Calcular Área...}]

[20] http://www.sat4j.org

[21] http://javabdd.sourceforge.net

[22] http://www.cs.lth.se/home/Radoslawzymanek/

- **Determina si dos modelos son equivalentes**: Esta operación toma como entrada dos modelos y retorna un valor determinando si los modelos son equivalentes. Dos modelos de características son equivalentes si representan el mismo conjunto de productos.
- **Obtiene las características básicas**: Esta operación toma como entrada un modelo y retorna el conjunto de características que aparecen en todos los productos.
- **Obtiene las características variantes**: Esta operación toma como entrada un modelo y retorna el conjunto de características que no aparecen en todos los productos.
- **Calcula en número de productos**: Esta operación toma como entrada un modelo y retorna el número de productos que este representa.

En la tabla 1, verticalmente se organizan las operaciones de análisis que se han identificado en la literatura. Las celdas que poseen la señal "/" indican que la propuesta de la columna provee soporte a la operación de la fila. Una señal "−" indica que la propuesta no suporta la operación. Una señal "∼" indica que aunque la propuesta no trate de forma explícita la operación, se supone que podrá soportarla. De esta forma queda demostrado que los modelos de características analizados automáticamente con FAMA brindan la mayor cantidad de funcionalidades (Carneiro, 2011).

| | Fan et al. [29] | Wang et al. [61] | Mannion [38] | Zhang et al. [64] | Sun et al. [54] | Batory [3] | Metzger et al. [41] | Benavides et al. [57] | FAMA [8] |
|---|---|---|---|---|---|---|---|---|---|
| | DL | | PL | | | | | CP | MP |
| Product válido | ≈ | ✓ | ✓ | ≈ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Modelo válido | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Todos los productos | - | - | ≈ | ≈ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Modelos equivalentes | - | - | ≈ | ≈ | ✓ | ≈ | - | ≈ | ✓ |
| Características básicas | - | - | ≈ | ≈ | ≈ | ≈ | ✓ | ≈ | ✓ |
| Características variantes | - | - | ≈ | ≈ | ≈ | ≈ | - | ≈ | ✓ |
| Número de productos | - | - | ✓ | ≈ | ≈ | ≈ | ≈ | ✓ | ✓ |
| Potenciales productos | - | - | ≈ | ≈ | ≈ | ≈ | - | ✓ | ✓ |
| Filtra productos | ≈ | ≈ | ≈ | ≈ | ≈ | ✓ | - | ✓ | ✓ |
| Commonality | - | - | ≈ | ≈ | ≈ | ≈ | - | ✓ | ✓ |
| Características muertas | - | - | ≈ | ✓ | ≈ | ≈ | ✓ | - | ✓ |
| Optimización | - | - | - | - | - | - | - | ✓ | ✓ |
| Explanaciones | ≈ | ✓ | ≈ | ≈ | ✓ | ≈ | - | - | ✓ |
| Explanaciones correctivas | - | - | - | - | - | - | - | - | ≈ |
| Reducción de productos | ≈ | ≈ | ≈ | ≈ | ≈ | ✓ | - | - | ≈ |
| Propagación de decisiones | - | - | ≈ | ✓ | ≈ | ✓ | - | ≈ | ≈ |
| Simplificación | - | - | ≈ | ✓ | ≈ | ≈ | - | - | ≈ |
| Fusión | - | - | ≈ | ≈ | ≈ | ≈ | - | - | ≈ |
| Modelos de características | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| OVM | - | - | - | - | - | - | ✓ | - | ≈ |

**Tabla 1**: Resumen de las propuestas para el tratamiento automático de modelos de características (Carneiro, 2011).

### 1.6. Plataforma Atlas

La plataforma Atlas es una herramienta para la personalización de Aplicativos SIG con tecnologías ágiles que basa su funcionamiento en técnicas de mapeo por caché y configuración por Plug-in de interfaz (Pantoja, 2012). La plataforma utiliza el estilo Modelo-Vista-Controlador en su arquitectura debido a la dinámica, robustez y rapidez que brinda, generalizadas en sistemas a gran escala. Se

determinó en el proyecto que la metodología de desarrollo a utilizar es RUP[23] y como servidor de mapas y cumpliendo con las exigencias de la universidad para el desarrollo de software libre, se determinó que MapServer resulta el adecuado para el desarrollo de la plataforma; siendo Apache el servidor web escogido por ser libre.

### 1.7. Análisis de algunas soluciones basadas en LPS

**Modelo de Desarrollo para LPS en Centros de Producción UCI**

El Modelo de Desarrollo para LPS en Centros de Producción, fue propuesto por Henrik Pestano Pino en el año 2010 como trabajo de diploma para optar por el título en máster en Gestión de Proyectos Informáticos; con el objetivo de aplicarse como esquema de trabajo en la UCI. En el mismo se analizaron las principales tendencias en Europa y América tomado los aspectos positivos para conformar un modelo basado en los principios de LPS, arquitectura de empresas y mejora continua.

Su esquema de trabajo está determinado por seis unidades de desarrollo que recoge cada uno de los escenarios claves de los proceso de elaboración de software en una línea. Está soportado por los principios propuestos por los modelos del SEI (Software Engineering Institute), TWIN y PMI (Project Management Institute) y adopta en cada una de sus áreas de procesos claves las mejores prácticas y tendencias mundiales sobre esta rama. La principal limitante para su aplicación en una línea de desarrollo de SIG es la generalidad productiva que abarca y establecimiento de una Familia de Productos como expansión conceptual.

**Modelo de desarrollo basado en líneas de productos de software para Sistemas de Información Geográfica sobre la base de la Plataforma GeneSIG.**

El Modelo de desarrollo basado en líneas de productos de software para Sistemas de Información Geográfica sobre la base de la Plataforma GeneSIG, fue presentado por Yoenis Pantoja Zaldívar en el año 2012 como trabajo de diploma para optar por el título de máster en Gestión de Proyectos Informáticos; con el objetivo de desarrollar SIG sobre la base de GeneSIG. Se toman como punto de partida los elementos esenciales de algunos de los principales modelos de desarrollo basados en LPS

[23] Proceso Unificado Ágil. Es la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas, tiene como objetivo garantizar un desarrollo eficiente y robusto. Es un proceso dirigido por casos de usos, centrada en la arquitectura, iterativo e incremental. Define quién está haciendo, qué está haciendo, cuando lo hace y como lo hace, utiliza para ello el Lenguaje de Modelado Unificado (UML) (Jacobson, 2004).

existentes como son PRAISE[24], CAFE[25], TWIN[26]…, así como la experiencia de trabajo en la línea Aplicativos SIG. Toma como referencia el entorno conceptual propuesto por (Pestano Pino, 2010) y se complementa con técnicas y metodologías para el desarrollo de SIG. Contempla la conceptualización de cinco elementos esquemáticos basados en los principios de LPS: Dominio, Familia de productos, Arquitectura, Activos de software y Modo de producción.

El esquema de trabajo implantado por el equipo de proyecto basa su funcionamiento en cinco actividades principales llevadas a cabo por los mismos miembros del proyecto en cada iteración (ver figura 2). El modelo combina los principios de LPS y la mejora continua de sus procesos mediante una adaptación de CMMI. Al tratar el tema de la variabilidad, solamente se hace referencia al uso de los modelos de características empleando el método FODA, por lo que como principal limitante se identificó la gestión parcial de la variabilidad en el modelo propuesto.

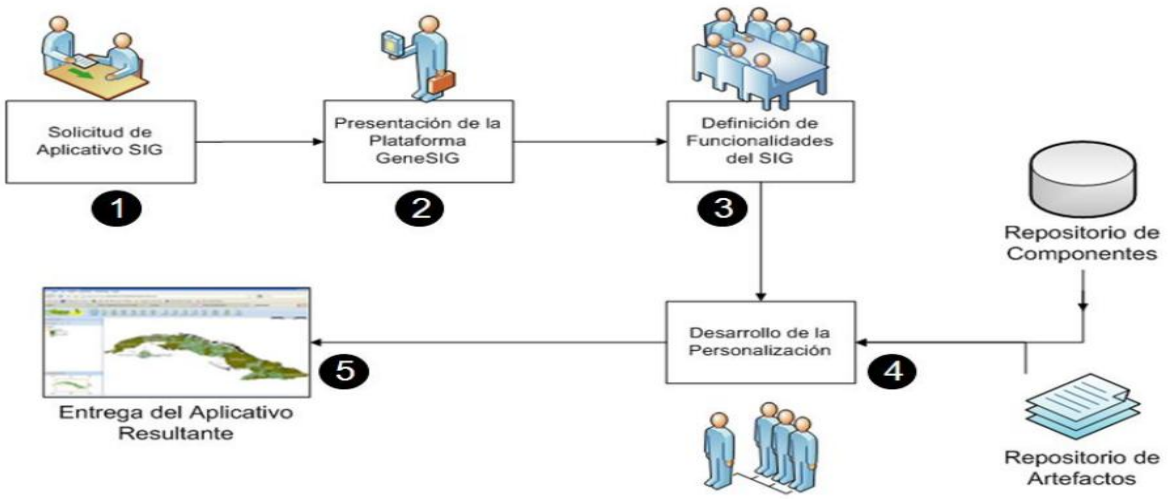


**Figura 4:** Procesos de personalizaciones sobre GeneSIG (Pantoja, 2012).

[24] *Product-line Realization and Assessment in Industrial Settings* definido en el *European Software Institute* (ESI) entre los años 1998 y 1999. Constituyó una referencia para el paradigma de desarrollo de familias de productos en Europa.

[25] *From Concepts to Application in System-Family Engineering*. Cuarto proyecto fomentado por el ESI basado en metodologías ágiles y con un esquema superior a los otros en cuanto a reutilización y gestión de componentes y/o activos de software.

[26] *Twin life cycle* o ciclo de vida gemelo. Provee una visión mucho más amplia y completa del desarrollo de software basado en componentes y su representación divide el proceso de desarrollo de software en dos grandes grupos paralelos: Ingeniería de Dominios e Ingeniería de Aplicaciones.

**Conclusiones del capítulo**

Dentro de los puntos más importantes de este capítulo se encuentra el proceso de gestión de la variabilidad donde se abordaron aspectos fundamentales para desarrollar una línea de productos como son la separación en un sistema de los aspectos comunes y variables (puntos de variación y variantes) los cuales dan la posibilidad de ampliar la gama de posibles productos. También resultó de mucha utilidad el estudio del modelado de la variabilidad lo cual facilita un mayor control cuando se utilizan modelos de características los cuales pueden ser útiles a los analistas, diseñadores, programadores y gestores de la línea de productos. Al investigar acerca de las soluciones existentes se concluye que existen modelos que fomentan el paradigma de LPS como el modelo propuesto por el MSc. Yoenis Pantoja en el año 2012, aunque todavía carece de una completa gestión de la variabilidad lo cual es un elemento inseparable para el éxito de una LPS.

Del estudio de las distintas formas de modelar la variabilidad se concluyó que los modelos de características son el artefacto adecuado, los cuales separan los elementos obligatorios y opcionales de la línea, además necesitan de herramientas que permitan analizarlos automáticamente dentro de los que se encuentra el Plug-in para Eclipse FAMA el cual ofrece información precisa para la toma de decisiones. Al analizarse el diseño arquitectónico de la plataforma Atlas v2.0 se conocieron las herramientas y tecnologías utilizadas para el desarrollo de dicho sistema, lo cual, es de vital importancia a la hora de comprender la plataforma y añadir nuevas herramientas que la puedan dotar de elementos que aumenten su productividad.

## CAPÍTULO 2: ESTRATEGIA PARA LA GESTIÓN DE LA VARIABILIDAD

### Introducción

Todo proyecto dedicado a la producción de software se ve en la necesidad de aplicar guías para facilitar y agilizar el proceso de desarrollo ya que, con el paso del tiempo, los productos software se vuelven cada vez más complejos, por lo tanto, todo su desarrollo se torna más complicado. Es por tal razón que se han concebido estrategias para simplificar, de alguna forma, las tareas al momento de desarrollar software a gran escala. En este capítulo se propone una estrategia para la gestión de la variabilidad, su descripción y una explicación detallada acerca del proceso de creación de nuevos productos derivados en la plataforma Atlas.

### 2.1. Descripción de la estrategia

La estrategia se dividirá en dos fases (Ingeniería de dominio e Ingeniería de aplicación) que coincidirán con los dos procesos básicos establecidos para las líneas de productos de software (figura 5) unido a los cuales se definen 7 actividades principales dirigidas a la gestión de la variabilidad.

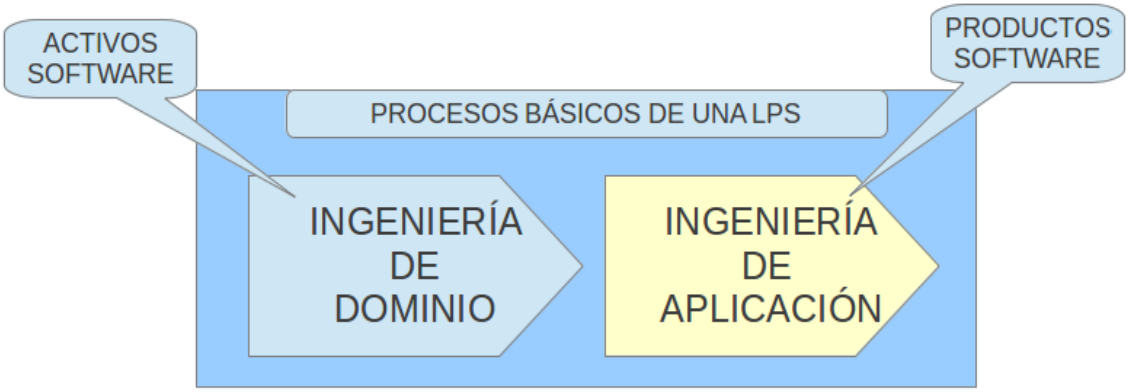


**Figura 5:** Procesos básicos de una LPS (Montilva, 2006).

Teniendo en cuenta el ciclo de vida de RUP las actividades se incluirán como lo muestra la figura 6.

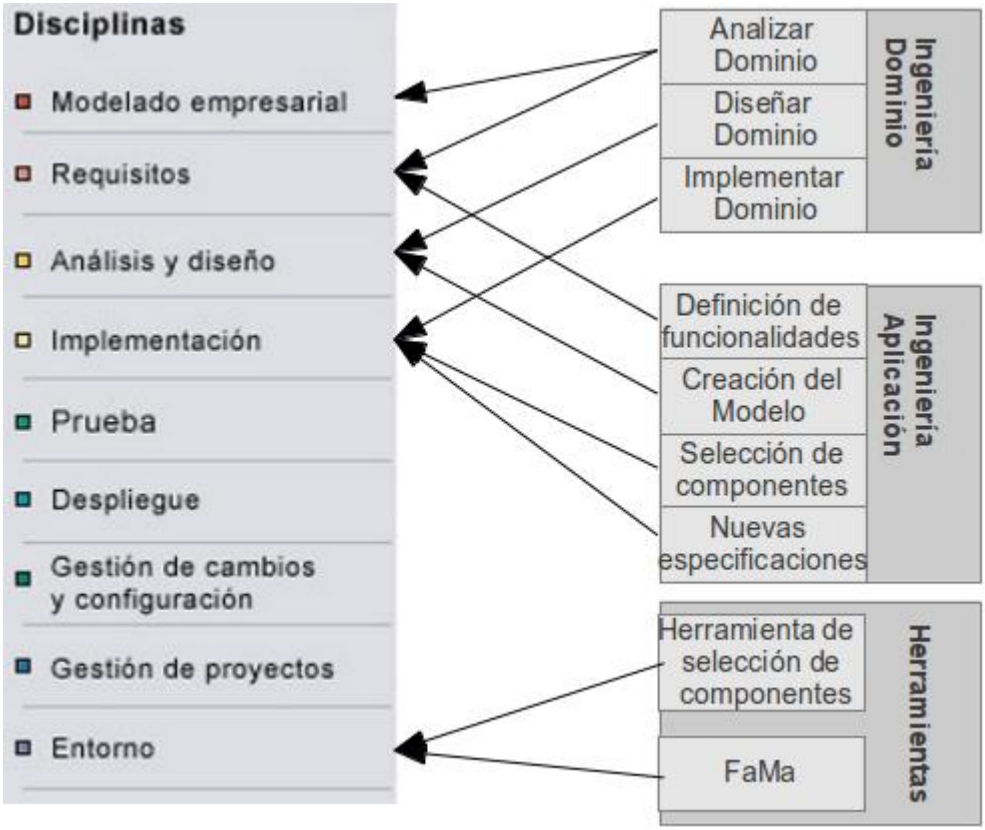


**Figura 6:** Actividades integradas a RUP.

Con la estrategia se busca responder a las siguientes interrogantes sobre la gestión de la variabilidad: ¿qué hay que hacer?, ¿cómo?, ¿quién?, ¿con qué herramienta?, ¿cuáles son los resultados esperados?

El *qué hay que hacer* se logra mediante la organización de un conjunto de actividades relacionadas que según transcurre el ciclo de vida de la línea de productos de software, guía a los trabajadores hacia las metas de la gestión de la variabilidad.

El *cómo* indica las tareas específicas, que pueden variar su implementación de acuerdo a la metodología que se utiliza en un proyecto específico de construcción de activos de software o de productos personalizados dentro de la línea. Es, cómo llevar a artefactos concretos los propósitos indicados en el qué de la estrategia, según las restricciones de la metodología.

El *quién* hace referencia a los trabajadores encargados de llevar a cabo las distintas actividades propuestas en el qué y concretadas en el cómo. Estos trabajadores son los responsables de obtener los resultados esperados apoyándose en las herramientas establecidas.

Las *herramientas* buscan proveer un entorno automático o semiautomático para la realización de las

actividades propuestas por el qué, concretadas en el cómo, que facilite el trabajo de los trabajadores y permitan obtener parte de los resultados esperados.

Los *resultados esperados* son los artefactos que se obtienen de la ejecución de las actividades por los trabajadores, utilizando las herramientas según sea previsto y de acuerdo a determinadas restricciones de la metodología de desarrollo.

### 2.2. Estrategia para la gestión de la variabilidad

El modelo propuesto por el MSc. Yoenis Pantoja Zaldívar en conjunto con la metodología de desarrollo RUP, brindan las actividades complementarias a la estrategia para la gestión de la variabilidad que se desea proponer y sirven como procesos de apoyo a la estrategia pues indican cómo llevar a cabo determinadas actividades de ingeniería de LPS e ISW[27], cuestiones a las que se recurre indistintamente como parte del desarrollo de activos y productos de software.

[27] Ingeniería de software

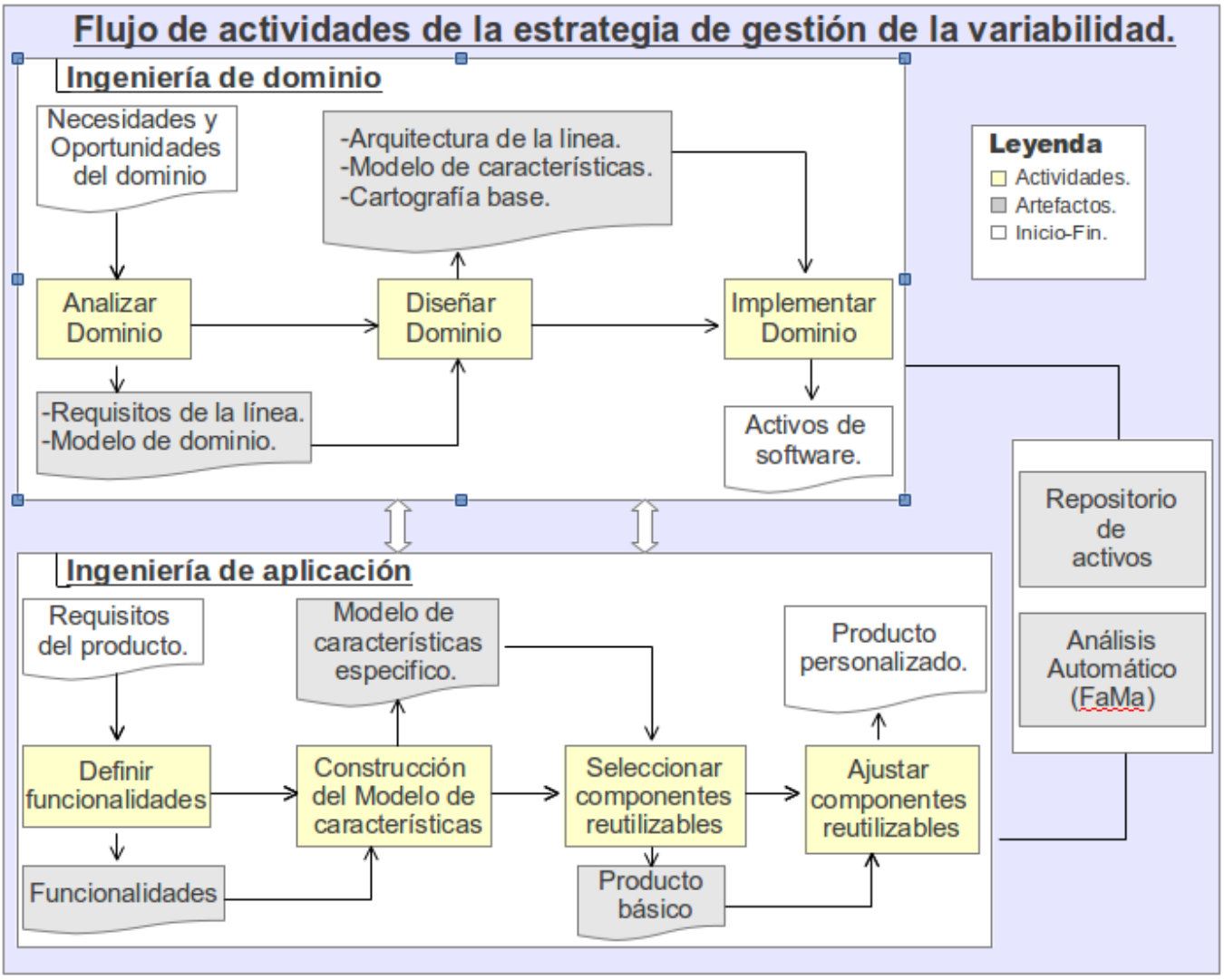


**Figura 7:** Representación gráfica de la estrategia de gestión de variabilidad.

**Fase # 1 Ingeniería de dominio**

Para la gestión de la variabilidad como parte de la ingeniería del dominio se toma en cuenta el peso que tienen los modelos de características para representar la variabilidad de la línea, siendo identificado como un mecanismo de variación utilizado a nivel de requerimientos (Clements, 2006). Durante la implementación del dominio se aconseja la utilización de otros mecanismos de variación, según sea necesario, que permitan poblar la arquitectura y su evolución mediante la incorporación de nuevas variantes.

**Actividad No. 1: Análisis del dominio.**

Esta actividad implica analizar las necesidades y oportunidades del dominio de los sistemas de información geográfica, en su forma más general, pudiendo llegar a determinarse dominios más específicos como el de los SIG para la salud o el de los SIG para el transporte. Esta termina con la identificación de los requisitos de la línea para el dominio en cuestión. Usualmente implica un gran esfuerzo al inicio de la creación de la línea de productos, dónde se construyen los activos que formarán parte de la misma.

Partiendo del hecho de que la estrategia está pensada para la creación de una línea cuyos activos de software fundamentales son tomados de Atlas, la cual ya tiene creado los principales componentes que se necesitan para la construcción de sus personalizaciones, la estrategia se enfocará en esta actividad en el análisis del dominio cuando se solicite la creación de nuevos componentes o la actualización de los ya existentes, para evitar que se implemente una funcionalidad desarrollada con anterioridad o que no sea parte del dominio de la línea.

**Actividad No. 2: Diseño del dominio**

Luego de un correcto análisis del dominio y conocido los principales activos que conformarán la línea de productos se procede a diseñar el dominio. Pero la estrategia se enfocará en la representación del dominio a través del modelo de características gestionando la variabilidad. Hay muchas semejanzas entre los conceptos de arquitectura de software y modelo de características. La arquitectura de software se puede definir como una descomposición de un problema en términos de componentes o módulos. De la misma forma, el modelo de características puede ser definido como una descomposición de un problema en términos de unidades lógicas de comportamiento o características. Parece intuitivo para hacer una correspondencia directa entre características y componentes (Jiménez, 2010).Para diseñar los modelos de características se transforma la arquitectura teniendo en cuenta:

- Componentes opcionales que son requeridos por algunos miembros de la familia de productos.
- Componentes comunes a todos los miembros de la familia de productos.
- Componentes variantes de los cuales algunos miembros de la familia de productos emplean distintas versiones.

**Descripción del modelo de características de la línea.**

Para posibilitar una mejor comunicación entre los involucrados y facilitar la interpretación y uso del modelo de características de la línea se procede a su descripción. Es una actividad que se ejecuta

además cada vez que ocurre algún cambio en el modelo que implique la actualización de su descripción.

El modelo de características de la figura 8 describe en forma de árbol la arquitectura de ATLAS a un alto nivel donde la raíz representa el sistema como un todo. La aplicación está formada por cuatro secciones principales (*Cálculos, Localizaciones, Interfaz y Navegación*) las cuales son opcionales excepto la sección de Navegación que es obligatoria, lo que indica que tiene que estar presente en cualquier nuevo producto.

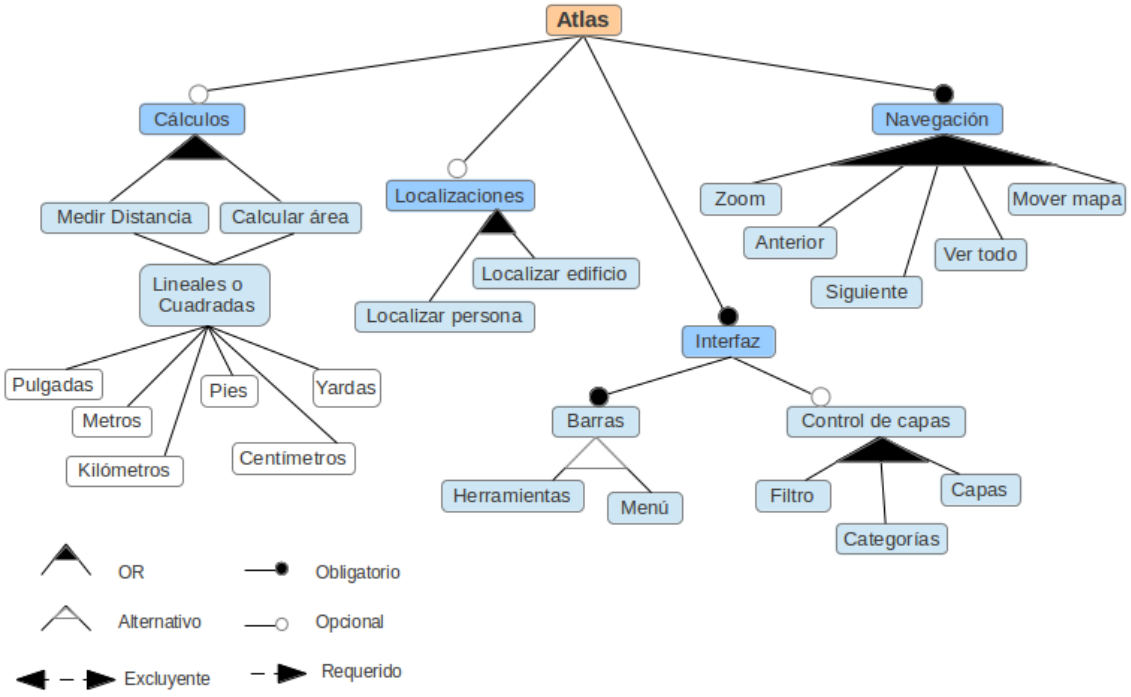


**Figura 8:** Arquitectura de Atlas descrita en características.

Como se puede observar la relación de *Navegación* con sus características hijas es de tipo *OR* que indica que pueden estar presentes una, dos o todas las características a la vez. En la relación *Interfaz-Barras* se puede ver una relación obligatoria lo cual indica, que *Barras* debe estar presente de alguna forma pero como la relación de *Barras* con sus características hijas es alternativa esto indica que solo estará presente una de las dos barras (*Menú* o *Herramientas*).

**Análisis de los modelos de características con el uso de FAMA.**

Después de creado el modelo característico que representa la línea se procede a su validación y análisis automático a través de la herramienta FAMA. En la medida en que crece la línea de productos

de software se dificulta el manejo de las características que forman parte de la misma así como de los productos que se pueden formar, por solo citar algunas dificultades. Es por ello que resulta beneficioso incorporar herramientas que automaticen determinados análisis sobre la línea, como los que se indican a continuación con el uso de FAMA.

**Funcionalidades de FAMA aplicables en el diseño del dominio.**

- **Determinan si un modelo es válido**: Esta operación toma como entrada un modelo y retorna un valor determinando si tal modelo es válido o no. Un modelo es válido cuando representa por lo menos un producto. Un modelo de característica solo puede ser inválido cuando se hace el uso incorrecto de las relaciones no jerárquicas entre las características, generando de esta manera contradicciones en el modelo.
- **Obtienen todos los posibles productos**: Esta operación toma como entrada un modelo y retorna todos los productos válidos representados por dicho modelo. [ejemplo Px = {Atlas. Navegación, Localizar Persona, Calcular Área...}]
- **Obtiene las características básicas**: Esta operación toma como entrada un modelo y retorna el conjunto de características que aparecen en todos los productos.
- **Obtiene las características variantes**: Esta operación toma como entrada un modelo y retorna el conjunto de características que no aparecen en todos los productos.
- **Calcula en número de productos**: Esta operación toma como entrada un modelo y retorna el número de productos que este representa.

**Actividad No. 3: Implementación del dominio.**

Una vez creado y validado el modelo de características donde se encuentran los principales activos que conforman la LPS, el equipo de desarrolladores se dará a la tarea de implementar y almacenar los componentes necesarios para la creación de los productos. Los desarrolladores llevarán a cabo esta tarea en dos momentos, inicialmente cuando se crean los componentes por primera vez, y el segundo momento cuando los activos necesitan ser modificados o actualizados.

Durante la implementación del dominio se procede a la incorporación de los mecanismos de variación a la arquitectura de la línea, que la dotará de mayores capacidades de variabilidad y por tanto de construcción de productos.

| Actividad | Objetivos | Ejecutores | Salida |
|---|---|---|---|
| **Análisis del dominio.** | Conocer las necesidades y oportunidades que se puedan tener en un dominio o sector del mercado, lo cual será posible después de una correcta recopilación de información de alta calidad. | Jefe de la línea.<br>Analista principal.<br>Arquitecto de software. | Modelo de dominio.<br>Diccionario de datos. |
| **Diseño del dominio.** | Diseñar una arquitectura flexible que pueda ser genérica y entendible para clientes o abstracta y específica para el equipo de desarrollo. | Arquitecto de software.<br>Jefe de desarrollo.<br>Cartógrafos. | Repositorio de activos.<br>Arquitectura base de la línea.<br>Modelo de características.<br>Estándares de codificación.<br>Cartografía base. |
| **Implementación del dominio.** | Implementar todos los componentes que fueron identificados en el análisis de aspectos los cuales serán la base de las personalizaciones. | Jefe de desarrollo.<br>Desarrolladores. | Activos de software. |

**Tabla 2:** Establecimiento de roles y objetivos por actividad (Fase Ingeniería de dominio)

**Fase # 2 Ingeniería de Aplicación.**

En la fase de ingeniería de aplicación se tendrá en cuenta la entrada de los requisitos funcionales del producto a construir como parte de la instanciación de la arquitectura de la línea. Luego estos requisitos que organizados permitirán la creación de un modelo de características específico el cual será evaluado por la herramienta FAMA y una vez asegurada la validez del modelo, se procederá a conformar la nueva personalización a través de la herramienta de selección de características la cual fue creada con el objetivo de automatizar el proceso de integración de los componentes que formaran

parte del nuevo producto.

**Actividad No. 1: Definición de funcionalidades**

Esta actividad se centra en la descripción de los requerimientos funcionales con los que debe contar la nueva personalización. Estos requerimientos son los encargados de definir el comportamiento que debe tener el nuevo producto, o sea las condiciones o capacidades que este debe cumplir. El artefacto que se utiliza para el levantamiento de los requisitos es el artefacto “Especificación de Requisitos” definidos en el expediente de proyecto v 2.0.

Aquí se genera una tabla compuesta por cuatro columnas (*Requerimientos, Módulos, Características Asociadas, Características Definidas*) donde quedarán recogidos los requisitos ubicados en su módulo correspondiente y con las características que dan solución al requisito. Es decir el requisito “Calcular distancia entre dos puntos” quedará recogido en la columna requisitos correspondiente al módulo Cálculo y en la columna Características Definidas quedara recogida la característica que da solución al requisito en este caso calcular área. Esta tabla brindará a todo el equipo de trabajo una mejor visión de lo que podría ser el nuevo producto y sobre todo será una guía básica para el desarrollador, lo que se traduce en la trazabilidad entre estos elementos durante el ciclo de vida. La tabla es la entrada principal para la construcción del modelo de características del producto. Ver ejemplo en Tabla 4.

| Requerimientos | Módulos | Características Asociadas | Características Definidas |
|---|---|---|---|
| | **Navegación** | **--Ver todo**<br>**--Zoom**<br>**--Mover** | |
| | **Cálculo** | **--Medir distancia**<br>**--Calcular Área** | |
| | **Localización** | **--Localizar una persona en el mapa.**<br>**--Localizar un edificio en el mapa.** | |
| | **Interfaz** | **--Barra de herramientas.**<br>**--Panel** | |

**Tabla 3:** Requerimiento Módulo Característica.

### Actividad No. 2: Construcción del modelo de características específico

Una vez determinados los principales requisitos del producto entra a jugar un papel muy importante el modelo de características a través del cual se determinarán aspectos como: ¿Existen suficientes características descritas como para satisfacer todos los requerimientos de la aplicación? ¿Se deberán incluir nuevas características al modelo?

Se determina si las características de la línea cubren las del producto, es decir, si realmente existen características suficientes para satisfacer al nuevo producto. Esto quiere decir que el producto solicitado forma parte de nuestra línea de productos, de lo contrario si las características de nuestro modelo resultan ser insuficientes el equipo deberá retornar a la fase de ingeniería de dominio ya que el producto solicitado no pertenece al conjunto de productos posibles. Como salida de la actividad se obtiene el modelo de características del producto, el cual facilita la identificación de las necesidades del usuario, representa las características de los productos y sus relaciones, y facilita la identificación de los componentes reutilizables. La figura 9 es una representación visual del modelo de características de un producto hipotético.

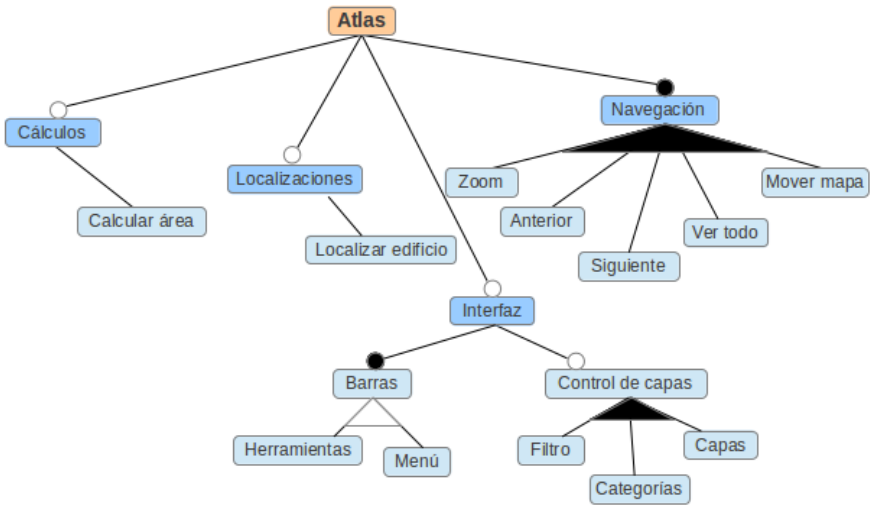


**Figura 9:** Modelo de características específico que representa un producto.

### Análisis de los modelos de características con el uso de FAMA

Después de creado el modelo de características especifico, vuelve a ser necesario el uso de la herramienta FAMA ya que cuando el producto solicitado tiene muchas características se dificulta su manejo, incluso muchos de los análisis que realizan los analistas y arquitectos pueden ser engorrosos

y consumir tiempo valioso. Es por ello que resulta beneficioso incorporar herramientas que automaticen determinados análisis del producto que se construye, como los que se indican a continuación con el uso de FAMA.

Tener en cuenta que para poder realizar estos análisis en FAMA es necesario tener digitalizados el modelo de características que representa la línea (tabla 5) y el producto que se creará.

| Relación entre características | Significado de la relación |
|---|---|
| **%Relationships** | Palabra reservada para definir la estructura de árbol y tipos relaciones. |
| **;** | Palabra reservada que indica el fin de una relación |
| **A, B, C, D** | Variables que representarán las características. |
| **A: B C D;** | **A** es la raíz del modelo si es la primera característica definida.<br>**B, C** y **D** son hijas de la característica **A** |
| **A: [B];** | **B** representa una característica opcional que puede estar presente o no. |
| **A: B;** | **B** representa una característica obligatoria. |
| **A: [1,2] {B C};** | Relaciones con cardinalidad de 1…n<br>**A: [1,2] {B C} A** puede contener a **B** o **C** ó **BC** |
| **A: [1,1] {B C};** | Relaciones con cardinalidad de 1…1<br>**A: [1,1] {B C} A** puede contener a **B** o **C.** |
| **%Constraints** | Palabra reservada para representar relaciones de exclusión e inclusión entre características. |
| **A EXCLUDES B** | Si **A** es seleccionada **B** no puede ser incluida. |
| **A INCLUDES B** | Si **A** es seleccionada **B** debe ser incluida. |

**Tabla 4:** Proceso de digitalización de los modelos de características.

**Funcionalidades utilizables de FAMA para el análisis automático de modelo de características.**

- Determinan si un producto es válido para una LPS: Esta operación toma como entrada el modelo de características de la línea y un producto (un modelo de características del producto a construir) y retorna un valor determinando si el producto pertenece al modelo o no.

- Determina si dos modelos son equivalentes: Esta operación toma como entrada dos modelos y retorna un valor determinando si los modelos son equivalentes. Dos modelos de características son equivalentes si representan el mismo conjunto de productos.

**Actividad No. 3: Selección de componentes reutilizables para producto.**

En esta actividad el desarrollador se encarga de identificar los componentes reutilizables requeridos según el modelo de características del producto que fue validado en la actividad anterior. Es automatizada con la herramienta de integración de componentes que implementa el mecanismo de configuración, el cual se basa en la selección o no de los componentes, antes de la compilación. La herramienta construye el producto solicitado a partir de la selección de los componentes requeridos por el modelo de características del producto. Termina con la creación de una carpeta que contendrá la nueva personalización.

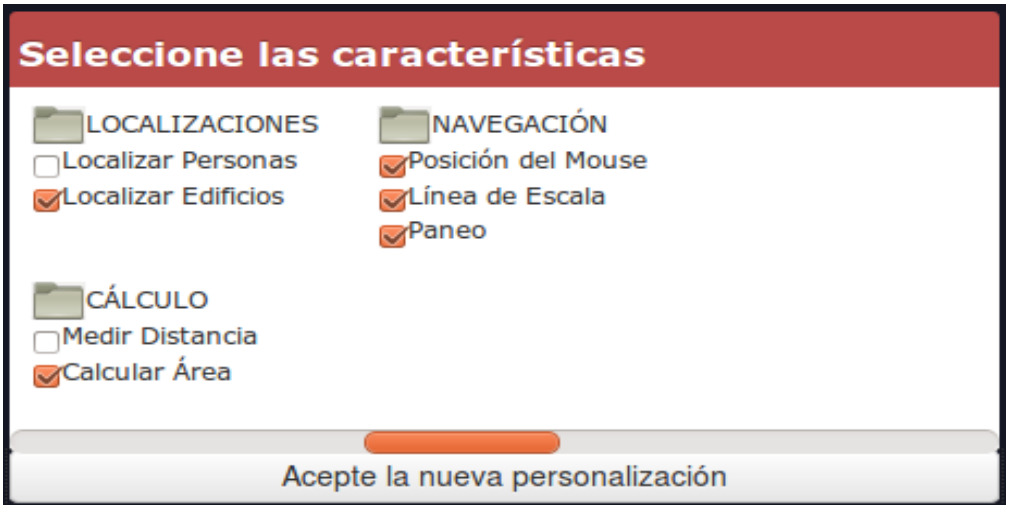


**Figura 10:** Herramienta para la integración de componentes.

**Actividad No. 4: Adecuar los componentes a las nuevas especificaciones.**

Una vez concluidas las actividades anteriores se puede decir que existe un producto funcional pero debido a que los componentes están programados de forma genérica, se debe adecuar los componentes seleccionados a las exigencias específicas del nuevo producto como etiquetas, redefinir validaciones de entrada de datos y estilos de diseños.

| Actividad | Objetivos | Ejecutores | Entrada | Salida |
|---|---|---|---|---|
| Definición de funcionalidades. | Dejar plasmado de forma clara las necesidades específicas de un cliente determinado. | Analista | ------- | Requisitos funcionales |
| Creación del modelo de características. | Llevar a las manos del desarrollador las necesidades del cliente en un leguaje a bajo nivel. | Analista | Requisitos funcionales | Modelo de características específico |
| Selección de componentes del nuevo producto. | Sustituir la creación manual de las nuevas personalizaciones. Facilitando en trabajo a través de la selección de características. | Desarrollador | Modelo de características específico | Nuevo producto |
| Adecuación de los componentes a las nuevas especificaciones. | Ofrecer al cliente una vista más personalizada de su producto. | Desarrollador | Nuevo producto | Producto personalizado |

**Tabla 5:** Establecimiento de roles y objetivos por actividad (Fase Ingeniería de Aplicación).

Se representa además un repositorio de activos que independientemente de la actividad y de la fase en cuestión se utiliza ya sea para obtener un activo que se necesite, para subir un activo nuevo o actualizar un activo modificado.

### 2.3. Descripción de la herramienta desarrollada

**Nombre de la herramienta:** Selección de componentes

**Lenguaje de programación utilizado:** HTML (del lado del cliente) y PHP (del lado del servidor)

Esta herramienta tiene como objetivo crear un nuevo producto utilizando los activos de Atlas. Es una representación visual para la selección de las características aunque por dentro lo que realmente sucede es que los componentes de la arquitectura que pudieran conformar el nuevo producto, son integrados entre ellos utilizando el mecanismo de variabilidad "mecanismo de configuración". Este

mecanismo brinda la posibilidad de seleccionar los componentes en una etapa previa al tiempo de compilación, aunque en este caso no utiliza un archivo que guarda las configuraciones como suele hacerse, sino que las ejecuta directamente al código.

Con la utilización de la herramienta se automatiza el proceso de creación de los productos ya que sustituye el trabajo manual que se hacía, que consistía básicamente en copiar o eliminar lo que se necesitaba o no, de un lugar a otro.

Al final se obtiene una personalización que contiene las características seleccionadas y analizadas en las diferentes fases de la estrategia propuesta.

**Conclusiones del capítulo**

Durante el desarrollo de este capítulo se realizó el planteamiento de la estrategia que se propone para guiar el proceso de personalización de productos derivados de la Plataforma Atlas. Al analizar la variabilidad se hace imprescindible la utilización de herramientas que automaticen el proceso pues se dificulta tratar con modelos que representen cientos de características por lo que se consideró relevante la utilización de la herramienta FAMA que constituye un soporte para el análisis automático de los modelos de características contribuyendo. Por otra parte resultó de gran utilidad la inclusión de la herramienta encargada de la integración de componentes que contribuye al proceso de creación de personalizaciones.

## CAPÍTULO 3: VALIDACIÓN DE LA ESTRATEGIA

### Introducción

En este capítulo se abordan los resultados obtenidos después de haber aplicado la propuesta a un caso de estudio. El enfoque principal consiste en asegurar que la estrategia es adecuada para la gestión de la variabilidad de los componentes desarrollados sobre la plataforma Atlas.

### 3.1. Alcance del modelo propuesto

La investigación se enmarca en la gestión de la variabilidad a través de los modelos de características, donde se agrega un perfil tecnológico para el posterior desarrollo de una LPS sobre la Plataforma Atlas. La estrategia propuesta permitirá profundizar en la adquisición de conocimientos para la gestión de la variabilidad en una LPS sobre Atlas, a través de guías que facilitan y agilizan el proceso de desarrollo, aunque la validación se enmarca en un caso de estudio, también se concibió para cualquier tipo de personalización a partir de los activos de Atlas.

### 3.2. Definición de características

En esta actividad a partir de los requerimientos funcionales solicitados para la nueva personalización o producto del caso de estudio, que fueron extraídos previamente del documento de “Especificación de requisitos” del expediente de proyecto v 2.0, se genera el artefacto mostrado en la tabla 6. Aquí se seleccionan las características que se asocian al requisito solicitado. Por ejemplo, para el requisito ¨Calcular el área de una superficie en el mapa¨ se obtiene que la característica que se corresponde sea ¨Calcular área¨, desechándose la característica de ¨Medir distancia¨. Este resultado es esencial para la construcción del modelo de características específico del producto y posteriormente para identificar los componentes necesarios.

| Requerimientos | Módulos | Características Asociadas | Características Definidas |
|---|---|---|---|
| --Navegar por el mapa | **Navegación** | **--Ver todo.**<br>**--Zoom.**<br>**--Mover.** | --Ver todo.<br>--Zoom.<br>--Mover. |
| -Calcular el área de una superficie en el mapa. | **Cálculo** | **--Medir distancia.**<br>**--Calcular Área.** | -Calcular Área. |
| -Localizar edificio en el | | **--Localizar persona.** | -Localizar edificio. |

| | | | |
|---|---|---|---|
| mapa. | **Localización** | **--Localizar edificio.** | |
| -Mostrar la barra de herramientas.<br><br>-Mostrar panel. | **Interfaz** | **--Barra de herramientas.**<br><br>**--Panel.** | --Barra de herramientas.<br><br>--Panel. |

Tabla 6: Especificación de requisitos.

### 3.3. Construcción del modelo de características

En esta actividad se crea el modelo de características específico que se corresponde con las exigencias de la nueva personalización, cuyas características fueron identificadas en la actividad anterior, partiendo del modelo de características que representa toda la línea de productos y desechando las características que no fueron seleccionadas. La figura 9 representa el modelo de características del producto solicitado y la figura 10 la digitalización de ese modelo en la herramienta FAMA, dónde se le pueden realizar análisis de interés.

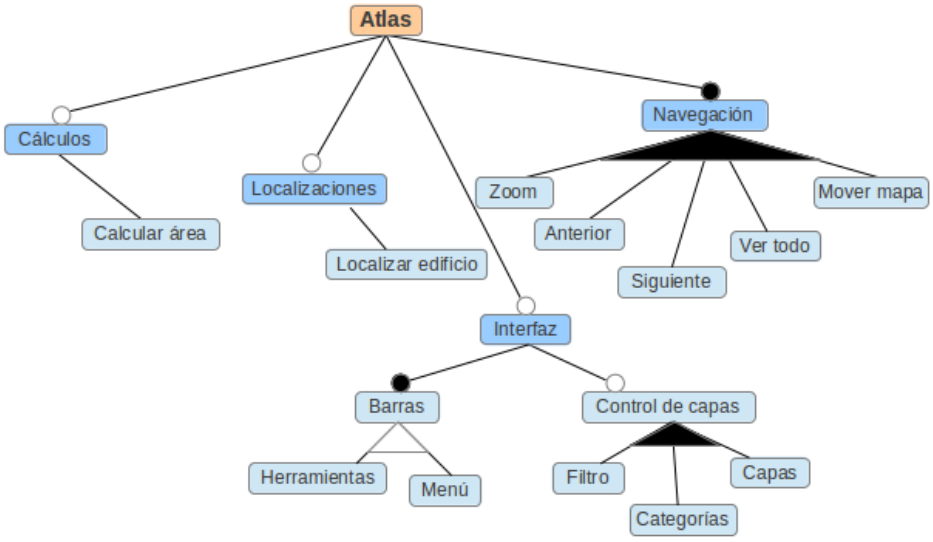


Figura 11: Modelo de características de la nueva personalización.

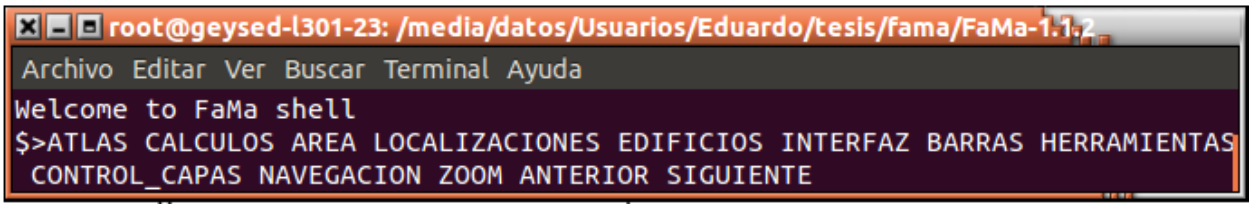


**Figura 12:** Digitalización de la nueva personalización.

Si bien es importante trabajar con el modelo de características específico, es necesario partir del modelo de característica que representa la línea de productos, a fin de determinar si el modelo de características específico pertenece a la línea. Es por ello que se utiliza el modelo de característica de la línea representado en la figura 11 y digitalizado para ser cargado por la herramienta FAMA, según se muestra en la figura 12, a partir de lo cual se realizan operaciones sobre éste como parte de análisis automático.

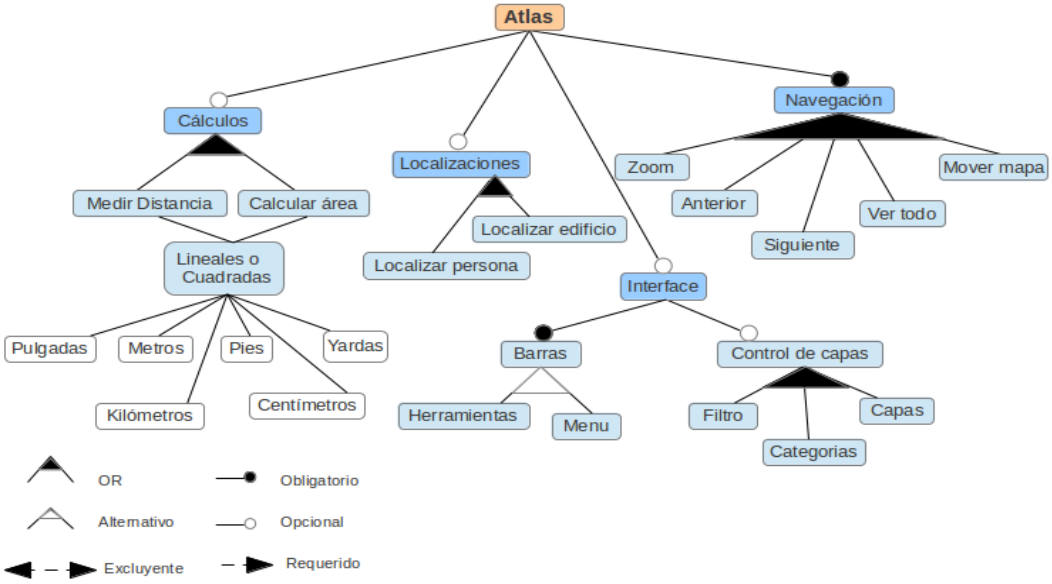


Figura 13: Modelo de características de la LPS.

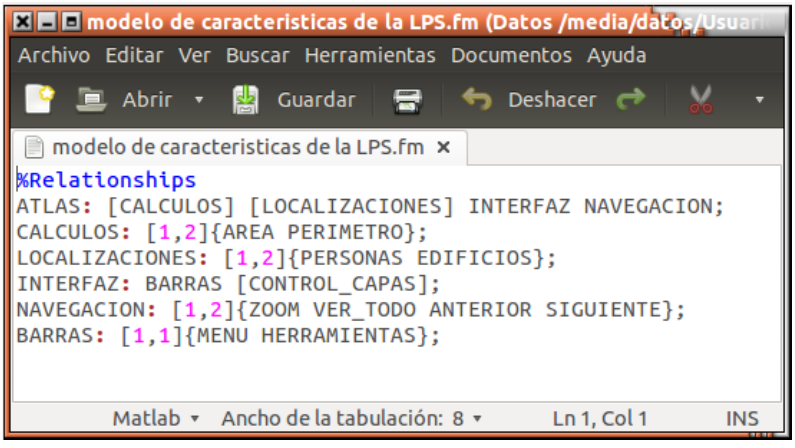


Figura 14: Digitalización del modelo de características de la LPS.

### 3.4. Análisis automatizado con la herramienta FAMA

Aquí se tienen en cuenta los procedimientos a realizar en la solución propuesta, primeramente se procede a acceder al directorio donde se ubica la carpeta **FaMa-1.1.2** que contiene la versión 1.1.2 del FAMA FW y posteriormente, mediante el comando ***java -jar lib/FaMaSDK-1.1.1.jar***, se inicializa el mismo y se muestra el mensaje "**Welcome to Fama Shell"** como lo muestra la figura 13**.**

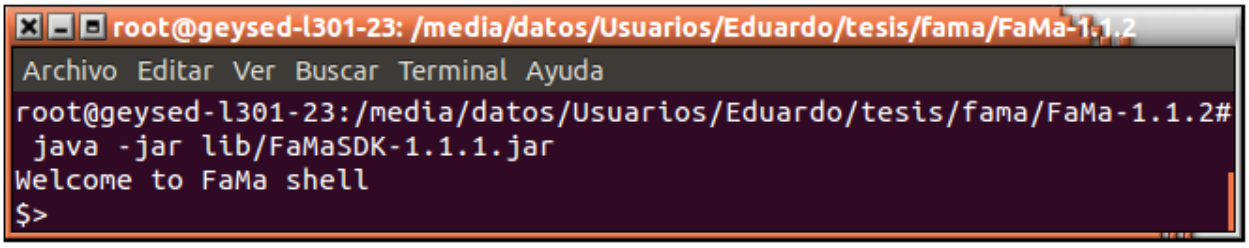


Figura 15: Pasos para inicializar FAMA.

Una vez iniciado FAMA, se procede a cargar el modelo de características de la LPS mediante el comando **load fm-samples/LPS_model.fm** y de haber realizado la operación satisfactoriamente se mostrara el mensaje **"Feature model loaded",** como lo indica la figura 14.

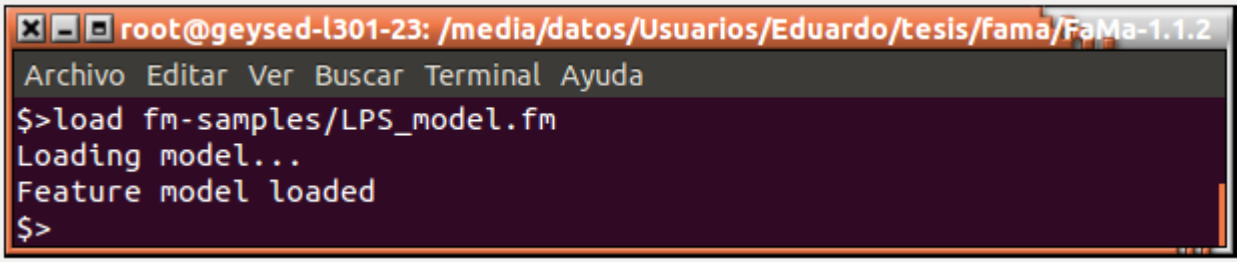


Figura 16: Pasos para cargar el modelo de características.

Seguidamente se procede a verificar si el producto seleccionado por el usuario es válido, para ello se introduce el comando **valid-product** seguido de la sintaxis del producto digitalizado.

Para calcular la variabilidad de la LPS, se emplea el comando **variability** y posteriormente se mostrará el valor como lo indica la figura 15.

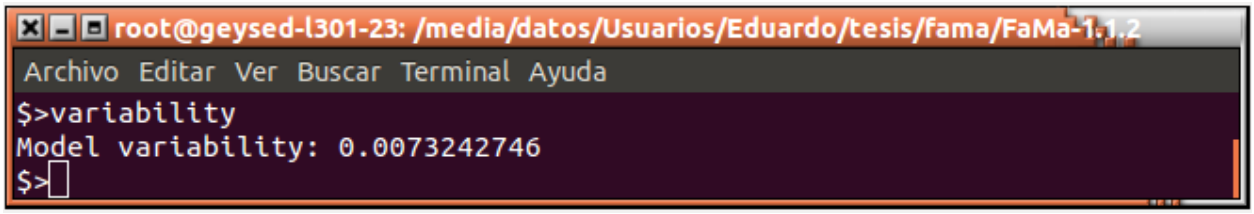


Figura 17: Funcionalidad para calcular la variabilidad de la LPS.

Si se quiere determinar la cantidad de veces que está presente una determinada característica en los productos de la LPS, se emplea el comando **commonality,** seguido del nombre de la característica como lo muestra la figura 16.

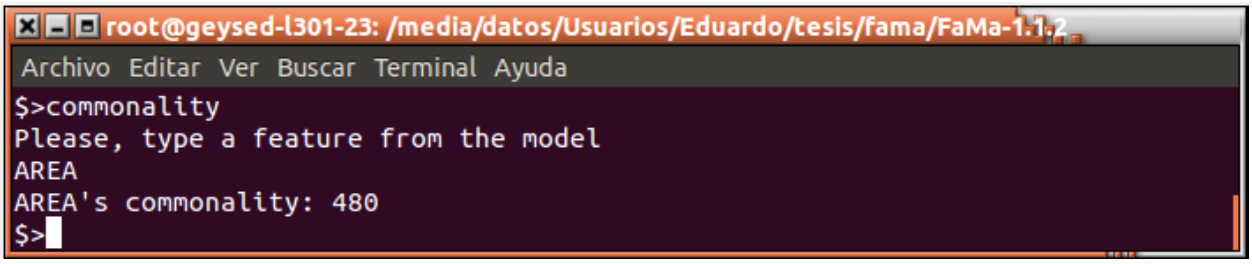


Figura 18: Funcionalidad para calcular la comunalidad en una LPS.

Seguidamente se procede a verificar si el producto seleccionado por el usuario es válido, para ello se introduce el comando **valid-product** seguido la sintaxis del producto digitalizado.

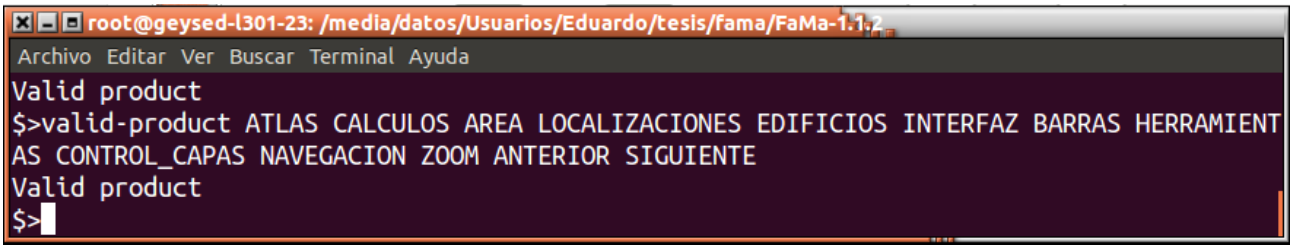


Figura 19: Funcionalidad para verificar la validez del producto.

### 3.5. Selección de componentes reutilizables

Teniendo en cuenta que la entrada de esta actividad son las características que conformarán el nuevo producto, el proceso de integración de componentes se traduce en que a partir de las características que formarán parte del nuevo producto, se obtienen los componentes necesarios. En la interfaz de la herramienta son seleccionadas estas características y posteriormente al presionar el botón nombrado “Acepte la nueva personalización“ se creará el nuevo producto según se muestra la figura 18.


Figura 20: Herramienta para la integración de componentes.

Posteriormente se crea en el directorio **www** del servidor web utilizado una carpeta nombrada **Personalización** que contendrá el nuevo producto integrado por los componentes seleccionados, como lo muestra la figura 19.

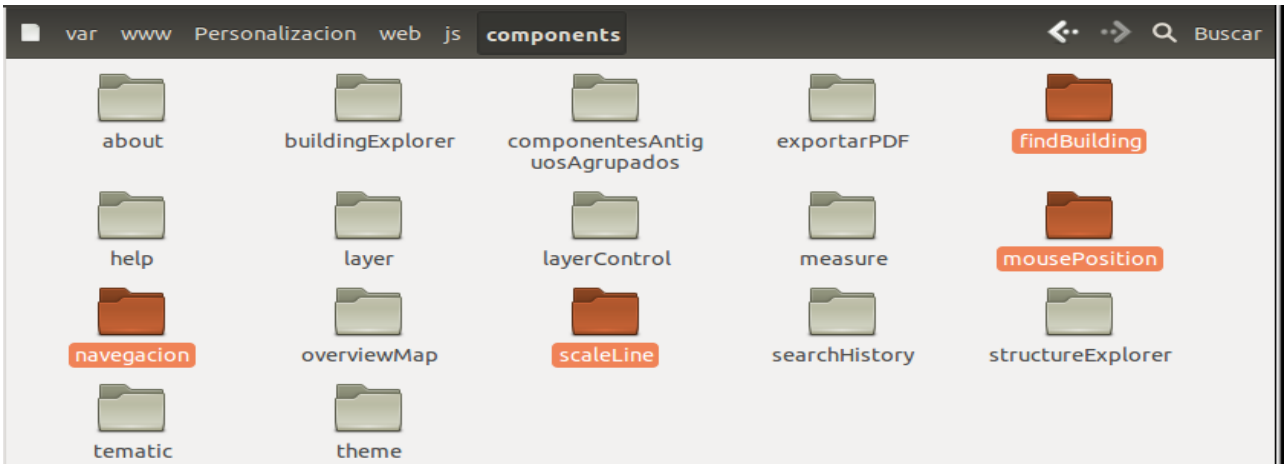


Figura 21: Ubicación de los componentes en la nueva personalización.

**Nuevo producto personalizado**

Una vez adecuado el producto creado en la actividad anterior a las nuevas especificaciones se obtendrá la nueva personalización tal como lo muestra la figura 20 para el caso de estudio.

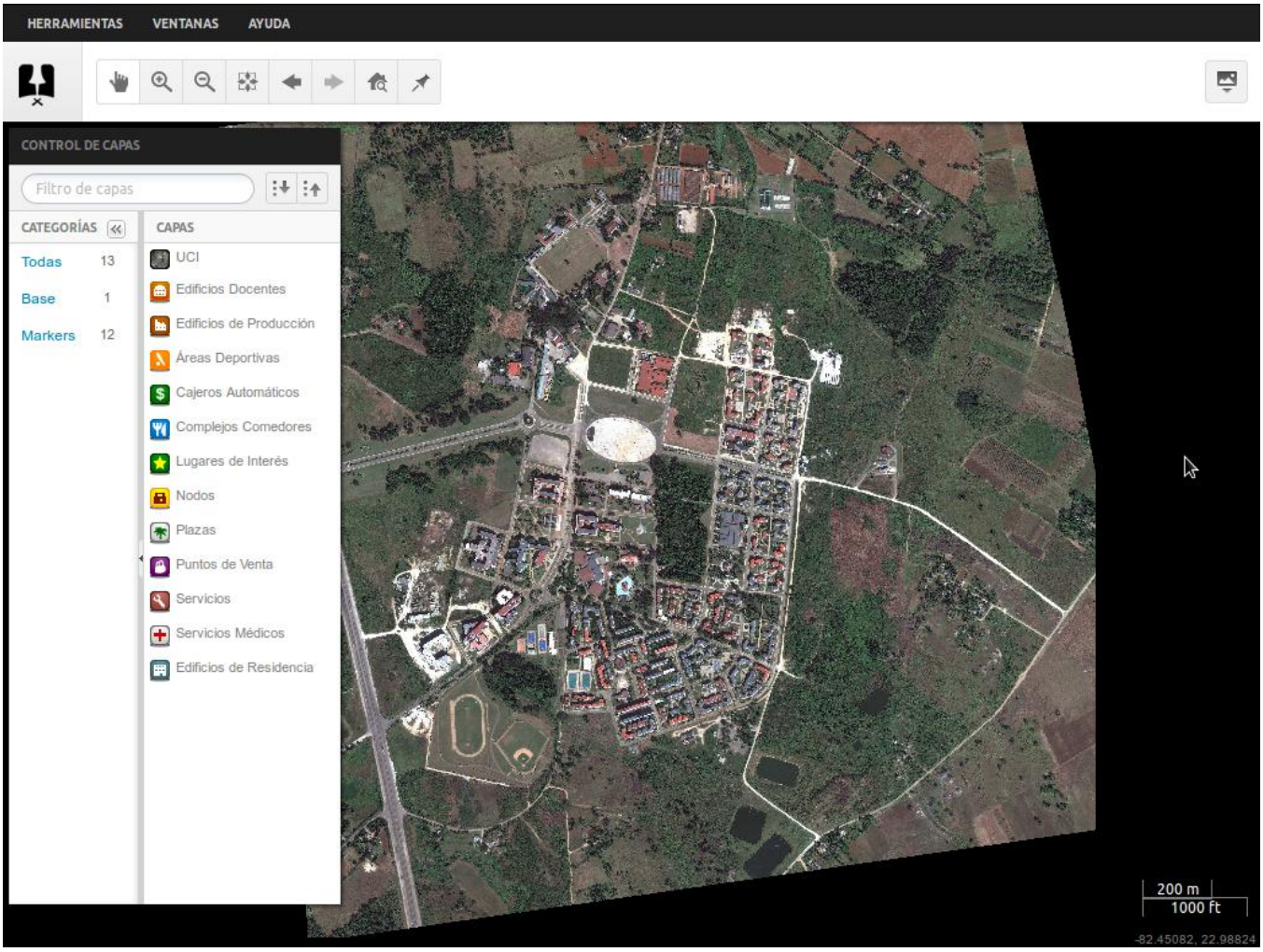


Figura 22: Nueva personalización.

**Conclusiones del capítulo**

Con la validación de la estrategia a través de los pasos seguidos en el caso de estudio se logró de gestionar la variabilidad. Además, el uso de la herramienta de selección de componentes posibilitó el proceso de integración de componentes, que se traduce en las características que formarán parte del nuevo producto, necesarios para la personalización.

## CONCLUSIONES

Mediante el desarrollo de la presente investigación fueron obtenidos numerosos conocimientos sobre la gestión de la variabilidad y su aplicación en líneas de productos proporcionando así, los fundamentos necesarios para el análisis y la selección de los medios que conformarían la estrategia a desarrollar. Es fundamental mencionar que con el estudio realizado se ha llegado a las siguientes conclusiones:

- El estudio de los conceptos fundamentales relacionados con el dominio del problema, así como la descripción de elementos importantes para la gestión de la variabilidad, permitieron tener un mayor entendimiento del tema, se reconoció la importancia de una correcta gestión de la variabilidad y se establecieron las bases para saber cómo realizar la estrategia resultante.
- Se escogió la herramienta FAMA para el análisis automático de los modelos de características, escogido como modelo de variabilidad.
- Al analizarse el diseño arquitectónico de la plataforma Atlas v2.0 se conocieron las herramientas y tecnologías utilizadas para el desarrollo de dicho sistema, lo cual, es de vital importancia a la hora de comprender la plataforma y añadir nuevas herramientas que la puedan dotar de elementos que aumenten su productividad.
- Se planteó la estrategia para guiar el proceso de personalización de productos y se desarrolló la herramienta de selección de componentes, dándosele cumplimiento al objetivo de la investigación.

**RECOMENDACIONES**

Con el cumplimiento de los objetivos planteados en la investigación, se recomienda para garantizar la total eficiencia de la estrategia:

- ✓ La aplicación de la estrategia a la plataforma Atlas con el objetivo de comprobar la eficiencia de la misma.
- ✓ Capacitar a todos los roles involucrados en los procesos de la estrategia, para garantizar la efectividad de su implantación.
- ✓ Desarrollar todos los componentes de Atlas con los mismos niveles de independencia que los componentes agrupados en la herramienta de selección de componentes.
- ✓ Desarrollar una herramienta que integre todas las actividades y herramientas que se proponen en la estrategia con el objetivo de aumentar los niveles de automatización.

## BIBLIOGRAFÍA Y REFERENCIAS BIBLIOGRÁFICAS

**Álvarez, Carlos. 1995.** *Metodología de la Investigación Científica.* Santiago de Cuba: Centros de Estudios de Educación Superior "Manuel Grain", 1995. : s.n., 1995.

**Bachmann, F y Clements, P. 2005.** Variability in Software Product Lines, Software Engineering Institute. [aut. libro] F Bachmann y Paul Clements. Pittsburg, USA : s.n., 2005.

**Barbosa León, Alberto. 2010.** *Testing de Servicios Web para líneas de Productos Software.* Ingeniería Informática, Universidad Rey Juan Carlos. 2010. págs. 7-10, Trabajo de fin de carrera.

**Bass, Len, y otros. 1997.** Product line practice workshop report.Reporte técnico CMU/SEI-97-TR-003, Software Engineering Institute.Carnegie Mellon University . *Product line practice workshop report. .* Pittsburgh : Pennsylvania 15213, 1997.

**Bayas, Córdova y Raúl, Marcos. 2012.** *Líneas de Prodcutos de software. Facultad de Ingeniería de Sistemas, Escuela Politécnica Nacional Campus "José Rubén Orellana", Universidad Politécnica SALESIANA., Agosto 2012.* Quito : Ladrón de Guevara E11-253 : s.n., 2012.

**Becker. 2002.** "Comprehensive Variability Modeling to faciliate Efficient Variability Treatment", in Proceedings of the 4th International Workshop on Product Family Engineering . [aut. libro] M Becker, y otros. Germany : Springer Verlag, 2002.

**Benavides Cuevas, David. 2005.** *Técnicas avanzadas para el razomaniento automático sobre modelos de características.* Sevilla, 41012. España : The Distributed Group, 2005.

**—. 2005.** *Técnicas avanzadas para el razomaniento automático sobre modelos de características.* Sevilla, 41012. España : The Distributed Group, 2005.

**Benavides, D. 2008.** *Fama framework In software product line conference tool demostrations.* 2008.

**Benavides, David. 2005.** *Técnicas avanzadas para el razomaniento automático sobre modelos de características.* Sevilla, 41012. España : The Distributed Group, 2005.

**Benavides, David, Trinidad, Pablo y Ruiz-Cortés, Antonio. 2005.** Automated Reasoning on Feature Models. In Advanced Information Systems Engineering: 17th International Conference CAiSE 2005. *Automated Reasoning on Feature Models.* s.l. : Springer-Verlag, 2005, Vol. 3520 of Lecture Notes in Computer Sciences, págs. 491-503.

**Bosch, J. 2000.** Design & use of Software Architectures: Adopting and evolving a product-line approach. s.l. : Addison-Wesley, 2000.

**Botterweck, Goetz, Janota, Mikolás y Schneeweiss, Denny. 2009.** A design of a configuratable feature modle configurator.Third International Workshop on Variability Modelling of Software- Intensive Systems, Seville , Spain, juanuary 28-30,2009 . Universitat Duisburg-Essen : ICB Research Report, 2009.

**Brown, A W. 2000.** *Latge-scale Component- Based Development.* Prentice-Hall : s.n., 2000.

**Capilla & Dueñas. 2002.** "Modelling Variability with Features in Distributed Architectures", in Proceedings of the 4th International Workshop on product Family Engineering. [aut. libro] R Capilla y J C Dueñas. Germany : Springer Verlag, 2002.

**Carneiro Roos, Fabricia.** Análisis automático de líneas de producto software usando distintos modelos de variabilidad. *Communicating the variability of a software-product family to customers. Software and System Modeling.*

**Carneiro, Fabricia. 2011.** *Análisis automático de LPS usando distintos modelos de variabilidad.* 2011.

**Chen, Lianping, Ali, Muhammad y Shull, Forrest. 2010.** *Managing Variability in Software Product Lines.* s.l. : IEE Software- SOFTWARE, 2010. ISSN: 0740-7459.

**Clements, P. y Northrop, Linda. 2001.** *Software Product Lines: Practices and Patterns.* 2001. Addison- Wesley.

**Clements, Paul C. 2006.** Managing Variability for Software Product Lines: Wor King with Variability Mechanisms. 10th International Software Product Line Conference. s.l. : IEEE Computer Society, 2006.

**Clements, Paul C., y otros. 2005.** Project management in a software product line organization. s.l. : IEEE Software, 2005.

**Córdova Bayas, Marcos Raúl. Agosto 2012.** *Líneas de Prodcutos de software.* Facultad de Ingeniería de Sistemas, Escuela Politécnica Nacional Campus "José Rubén Orellana", Universidad Politécnica SALESIANA. Quito : Ladrón de Guevara E11-253, Agosto 2012.

**—. 2012.** *Líneas de Prodcutos de software.* Facultad de Ingeniería de Sistemas, Escuela Politécnica Nacional Campus "José Rubén Orellana", Universidad Politécnica SALESIANA. Quito : Ladrón de Guevara E11-253, 2012.

**Dhungana, Deepak, Grunbacher, Paul y Rabiser, Rick. 2007.** DecisionKing: A flexible and extensible tool for integrated variability modeling. In Proceedings of the First International Workshop on Variability Modelling of Software-Intensive Systems . *DecisionKing: A flexible and Extensible Tool for Integrated Variability Modeling.* s.l. : VaMoS 2007, 2007, págs. 119-127.

**Díaz, Óscar . 2010.** Líneas de Producto Software. [aut. libro] Óscar Díaz y Salva Trujillo. [ed.] J. Garzás M.G.Piattini. *"Fábricas de Software: experiencias, tecnologías y organización".* s.l. : Ra-MA, 2010.

Ecured. [En línea] http://www.ecured.cu/index.php/Sistema_de_Informaci%C3%B3n_Geogr%C3%A1fica#Definici.C3.B3n.

Ecured. [En línea] http://www.ecured.cu/index.php/Sistema_de_Informaci%C3%B3n_Geogr%C3%A1fica#Definici.C3.B3n.

**Eriksson, M,, Borstler, J. y Borg, K. 2005.** *The pluss approach-domain modeling with features, use case and use case realizations. In software Product Lines. s.l. : Springer-Verlag, 2005, págs. 33-44.* 2005.

**FAMA. 2010.** FAMA. [En línea] 2010. http://code.google.com/p/famats/ .

**—. 2010.** FAMA. [En línea] 2010. http://code.google.com/p/famats/ y http://code.google.com/p/famats/downloads/list.

**Fielding, Roy Thomas. 2000.** Architectural Styles and the Design of Network-based Software Architectures. Tesis doctoral, University of Califormia. Irvine : s.n., 2000.

**Gacek, Cristina y OTROS. Marzo 2001.** *Successful Software Product Line Development in a Small Organization.* Alemania : IESE-Report, Marzo 2001.

**Garcés, Kelly, y otros. 2007.** Administración de Variabilidad en una línea de producto basada en modelos. *Variability Management in a Model-Driven Software Product Line.* Universidad de los Andes , Bogotá : Kelly Garcés et al., revista Avances en Sistemas e Informática, 2007, Vol. 4.

**Garcés, Kelly, y otros. 2007.** *Variability Management in a Model-Driven Software Product Line.* 2007.

**García, F. J., y otros. 2002.** *Líneas de Producto, Componentes, Frameworks y Mecano.* Universidad de Valladolid. España : s.n., 2002.

**González- Baixauli, Bruno, Laguna, Miguel A y Sampio do Prado Leite, Julio Cesar. 2004.** *Visual Variability Analysis with Goal Models. Sept. 2004.* Kyoto, Japan. : IEEE Computer Society., 2004.

**González Cento, Ing. Orlando Enrique. Marzo 2009.** *Estrategia y Plataforma Informática para la Gestión y Desarrollo de un proyecto Productivo.* Valencia : s.n., Marzo 2009.

**Griss, L y Favaro. 1998.** *Integrating feature modeling with the RSEB. In Proceedings of the Fifth International Conference on Software.* Canada : s.n., 1998.

**Hernández León, Rolando Alfredo y Coello González, Sayda. 2011.** *El Proceso de Investigacón Científica.* Ciudad de la Habana : Editorial Universitaria del Ministerio de educación Superior, 2011. IBSN 978-959-16-1307-3.

**IEEE. 2000.** 2000.

**Jacobson, Ivar. 2004.** El Proceso Unificado de Desarrollo de software. s.l. : Addison Wesley, 2004.

**Jiménez Méndez, Alberto.** Gestión de l avariabilidad en líneas de Producto Software.

**Jiménez, Alberto. 2010.** *Gestión de la variabilidad en Líneas de Producto Software.* Universidad de Sevilla : s.n., 2010.

**Kang, K.C., y otros. 1990.** *"Feature-Oriented Domain Analysis(FODA). Feasibility Study".* Pennnsylvania 15213 : Technical Report , 1990. CMU/SEI-90-TR21 (ESD-90-TR-222).

**Kang, KC y otros. 1990.** *Feature-Orinted Domain Analysis .Feasibility Study.* Pennsylvania 15213 : s.n., 1990.

**Kastner, Christian, y otros. 2009.** FeatureIDE: A tool framework for feature-oriented software development. In ICSE, pages 611-614. Alemania : IEEE, 2009.

**Kim., Jaejoon, y otros. 1998.** FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. 1998.

**Kim., Jaejoon, y otros. 1998.** FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. s.l. : Annals of Software Engineering, 1998.

**Klaus, Pohl, Gunter, Bockle y Frank, J. van der Linden. 2005.** Software Product Line Engineering: Fundations, Principles and Techniques. Berlín. Alemania : Springer-Verlag, 2005.

**Krueger, C W. 2002.** " Easing the Transition to Software mass Customization", in Proceedings of the 4th International Workshop on Product Family Engineering. " *Easing the Transition to Software mass Customization".* germany : Springer Verlag, 2002.

**Krueger, Charles W. 2006.** New methods in software product line practice. Communications of the ACM. s.l. : ISSN 0001-0785, 2006.

**L. Griss, Martin, Favaro, John y d´Alessandro, Massimo. 5-5-1998.** Integrating feature modeling with the RSEB. *In Proceedings of the Fifth International Conference on Software Reuse, Victoria, BC.* Vancouver, BC,Canadá : s.n., 5-5-1998, págs. 76-86.

**Laqua, Roland. 2002.** "Concepts for a Product Line Knowledge Base & variability". Proceedings of NetObjectDays 2002. s.l. : Erfurt, 2002.

**León, Barbosa. 2010.** *Testing de Servicios Web para líneas de Productos Software. Ingeniería Informática, págs. 7-10, Trabajo de fin de carrera.* Universidad Rey Juan Carlos : s.n., 2010.

**Liu, D. y Mei, H. 2004.** *From requirements to software architecture: A feature oriented mapping approach. In: In Proc. of the 2nd Int. Software Requirements to Architectures Workshop, ACM Press (2004) 69–76.* 2004.

**M, Eriksson, J., Borstler y K, Borg. 2005.** The pluss approach-domain modeling with features, use case and use case realizations. *In software Product Lines.* s.l. : Springer-Verlag, 2005, págs. 33-44.

**Marco, Sinnema, y otros. august 2004.** COVAMOF: A framework for modeling variability in software product families. *In Proceedings of the Third International Software Product Lines Conference ( SPLC 2004).* s.l. : Springer-Verlag Lecture Notes in Computer Science (LNCS 3154), august 2004, págs. 197-213.

**Mendez, Alberto Jimenez. 2010.** *Gestión de la variabilidad en Líneas de Producto Software.* 2010.

**Mikael Svahnberg and Jan Bosch. 2003.** Issues Concerning Variability in Software Product Lines, volume June of 146. Lecture Notes in Computer Science. 2003.

**Montilva, J. 2006.** *Desarrollo de Software Basado en Líneas de Productos Software.* 2006.

**Montilva, J. A. 2006.** Desarrollo de Software Basado en Líneas de Productos Software. IEE Computer Society Region9 Capitulo Argentina Programa DVP. 2006.

**Moreno Lazo, Luis Grabiel. 2012.** *Diseño de una Arquitectura Base para la Plataforma ATLAS v2.0.* Ciudad de la Habana, 2012 : s.n., 2012.

**Northrop, L M. 2002.** *SET Software Product Line Tenets. 2002.* s.l. : IEEE Software, 2002. págs. 32-40.

**Northrop, L. M. 2002.** *SET Software Product Line Tenets.* 2002. págs. 32-40. IEEE Software.

**Northrop, Linda y Clements, Paul. 2007a.** A framework for software product line practice. [En línea] 2007a. http://www.sei.cmu.edu/productlines/frame_report/.

**Pantoja, Yoenis. 2012.** *Modelo de desarrollo basado en LPS para SIG sobre la plataforma GeneSIG.* 2012.

**Parnas, D. 1976.** *On the Design and Development of Program Families.* 1976.

**Perry, Dewayne E y Wolf, Alexander L. 1992.** *Foundations for the study of software architecture. ACM SIGSOFT.Octubre 1992.* s.l. : Software Engineering Notes, 1992.

**Pestano Pino, Henrik. 2010.** Propuesta de modelo de desarrollo para líneas de productos de software en centros de producción. Tesis de Maestría. Universidad de las Ciencias Informáticas. La Habana. Cuba : s.n., 2010.

**Ramírez Mastrapa, Ing. Yudenia. 2008.** Estrategia de integración para el proyecto de transformación del sistma de identificación, migración y control de extranjeros de la República Bolivariana de Venezuela. *Tesis de maestría en Gestión de Proyetcos Informáticos.* Cuba : s.n., 2008.

**—. Diciembre 2008.** Estrategia de integración para el proyecto de transformación del sistma de identificación, migración y control de extranjeros de la República Bolivariana de Venezuela. *Tesis de maestría en Gestión de Proyetcos Informáticos.* Cuba : s.n., Diciembre 2008.

Revista Vinculando. [En línea] [Citado el: 26 de 01 de 2013.] www.vinculando.org/educacion/sistemas_de_informacion_geograficos_petroleros..

**Rhind, David. 1989.** Ecured. [En línea] 1989. http://www.ecured.cu/index.php/Sistema_de_Informaci%C3%B3n_Geogr%C3%A1fica#Definici.C3.B3n) . .

**Salicki & farcet. 2002.** "Expression and usage of the Variability in the Software Product Lines", in Proceedings of the 4th Internatinal Workshop on product Family Engineering. [aut. libro] S Salicki y N Farcet. Germany : Springer Verlag, 2002.

**Sánchez, A. 2004.** *Fuentes de Información para la Inteligencia Competitiva en I+D. 2004.* 2004.

**Schmid, K y I, John. 2004.** "A customizables approach to full lifecycle variability management" Sciencie of Computer Programming. 2004.

**Schmid, Klaus y John, Isabel. 2004.** A customizable approach to full-life cycle variability management. s.l. : Elsevier North- Holland, Inc. Amsterdam, The Netherlands, 2004, Vols. 53 Issue 3, December 2004, págs. 259-284.

**Sinnema, M y Deelstra, S. 2007.** Industrial validation of COVAMOF. Journal of Systems and Software, 2007. 2007.

**Sodhi, J. y Sodhi, P. 1999.** *Software reuse: Domain analysis and design process.* New York : McGraw-Hill, 1999.

**Sommerville. 2010.** 2010.

**Sommerville, Ian. 2000.** *Software engineering. 6ta edición.s.l. agosto 2000.* 6ta edición. s.l. : Addison-Wesley Pub Co, 2000.

**Szyperski, Clemens. 2002.** *Component software: Beyond object-oriented programming.* 2da edición. s.l. : Addison-Wesley Pub Co, 2002.

**Thomas von der Maben and Horst Lichter. 2003.** Requiline: A requirements engineering tool for software product lines. In Frank van der Linden,PFE, volumen 3014 of Lecture Notes in Computer Sciencie, pages 168-180. Alemania : Springer, 2003.

**Trinidad, Pablo, y otros. 2008.** FAMA framework. In SPLC, pages 359. s.l. : IEEE Computer Society, 2008.

**Webber, Jim, Parastatidis, Savas y Robinson, Ian. 2010.** REST in Practice. O´Reilly Media : ISBN, 2010.

**Zuani, Elio Rafael de. 2005.** *Introducción a la administración de organizaciones.* s.l. : Valletta Ediciones SRL, Business & Economics, 2005.

**Zubrow, Dave y Chastek, Gary. 2003.** Measures for software production lines. s.l. : Software Engineering Institute, Carnegie Mellon University,CMU/SEI-2003-TN-031, 2003.

**GLOSARIO DE TÉRMINOS**

**Característica:** Representación abstracta de un determinado producto.

**COVAMOF:** Framework para el modelado de la LPS para modelar la variabilidad en términos de puntos de variación y dependencias entre diferentes niveles de abstracción.

**Core assets:** Activos centrales.

**COTS:** Components "Off-The-Shelf", módulo de software que se puede escoger "de la estantería", listo para su empleo: unidad de reuso.

**Estrategia:** Arte para dirigir, guiar un asunto. Según la Real Academia Española.

**FAMA:** *Framework_for the Automated Analysis of Feature Models, framework* o marco automatizado para el análisis de modelos de características.

**FODA:** *Feature- Oriented Domain Analysis.*Análisis del dominio orientado a características*.* Primer modelo característico para extraer las similitudes y variabilidades de un dominio.

**FORM:** Feature Oriented Reuse Method es una extensión del método FODA para el modelado de la variabilidad en otras fases del desarrollo.

**Gestión de la variabilidad:** Se refiere a las actividades de evaluar y representar la variabilidad en los artefactos de software, establecer y gestionar dependencias entre diferentes variabilidades y dar soporte al uso de las variabilidades para construir e incrementar una familia de sistemas de software.

**GEYSED:** Centro de Desarrollo Geoinformática y Señales Digitales ubicado en la facultad 6.

**LPS:** Línea de Productos de Software.

**Modelos de características:** técnica para modelar variabilidad en una LPS. Representación gráfica de una línea de producto en términos de características y relaciones entre ellas.

**OVM:** Ortogonal_ Variability Modeling, modelado de la variabilidad ortogonal. Metodología para el modelado de la variabilidad en LPS.

**Reutilización:** Acción y efecto de reutilizar.

**SIG:** Sistemas de Información Geográficos.

**Variabilidad:** Cualidad de las cosas que tiende a cambiar o transformarse según la Real Academia Española.