Representación de la variabilidad de un sistema de venta de seguros en Internet mediante el marco de referencia de Bosch

Trabajo de grado, Maestría en Ingeniería

Juan Pablo Vergara Villarraga

Mayo de 2012

Asesor

Raquel Anaya de Páez

Departamento de Informática y Sistemas

Universidad EAFIT

Medellín, Colombia

Prefacio

"Ser sujeto de la humanidad nunca ha sido ni será jamás la única posibilidad que se le abre a la esencia recién iniciada del hombre histórico."

Martin Heiddegger.

Este trabajo de grado se realiza en el marco de la maestría en ingeniería cursada por Juan Pablo Vergara, Ingeniero de Sistemas egresado de la Universidad EAFIT con énfasis en Telemática y con algunos cursos en el pregrado del énfasis de Ingeniería de Software. Este trabajo de grado se realiza bajo la supervisión de PhD Raquel Anaya de Páez, perteneciente a la plata docente de la escuela de ingeniería de la Universidad EAFIT.

En 2006 yo, Juan Pablo Vergara, realicé mi período de práctica profesional en lo que hoy se conoce como Suramericana S.A., una empresa del sector asegurador y de seguridad social de Colombia y con recientes adquisiciones en países como Panamá, El Salvador y República Dominicana. A partir de ese momento y durante los últimos 6 años me he dedicado a la ejecución de proyectos de software tanto de mantenimiento como de construcción de productos de software a la medida para esta organización. Trabajar en proyectos de software empresarial para la compañía de seguros más grande del país me ha enfrentado con grandes retos tanto personales como técnicos los cuales se presentan en forma de necesidades complejas del negocio que deben ser satisfechas por piezas de software igualmente complejas. La inminente presencia del fallo, la alta rotación de los equipos de trabajo, los aspectos no técnicos de los proyectos y demás aspectos relacionados con proyectos de software de gran escala han alimentado constantemente mi interés y mi pasión por la disciplina de ingeniería de software de tal forma que decidí adelantar mi Maestría en Ingeniería con énfasis en Ingeniería de Software para la cual realicé el trabajo de grado presentado en este documento.

La elaboración de este trabajo se llevó a cabo entre Agosto de 2011 y Mayo de 2012; fue escrito con una dedicación de un tercio de tiempo diario, durante las noches, luego de mis jornadas laborales durante las cuales recolectaba información de valor para la elaboración del documento. Durante el año 2011 realicé una labor de conceptualización mediante cursos electivos con Raquel Anaya, mi asesora durante el programa de maestría, para en 2012 plasmar el conocimiento adquirido en un caso de estudio real. Escribir este documento ha sido una tarea enriquecedora, así como se espera que su lectura también lo sea. Asentar conocimientos recolectados y generados delinea un marco claro de lo aprendido en la medida en que permite organizar, en torno a un objetivo concreto, las diferentes referencias bibliográficas estudiadas.

Finalmente no me quiero dejar pasar por alto un agradecimiento a las personas que de alguna u otra manera estuvieron conmigo durante este proceso que me permite alcanzar una meta académica y personal. Una voz de aliento o una retroalimentación oportuna es invaluable cuando la mayor parte de mi tiempo personal estuvo dedicada a esta maestría durante los dos últimos años. Shin, Tita, Raquel, mil gracias.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

1. Introducción

1.1. Motivación

Cuando comencé a trabajar para Suramericana S.A. pude darme cuenta de primera mano que una gran parte de que las piezas de software del sistema principal transaccional de seguros se encuentra desarrollado en la base de datos. En una base de datos Oracle el lenguaje de programación principal es PL/SQL. El nombre de este lenguaje significa *Procedural Language / Structured Query Language*. La segunda parte del nombre del lenguaje es estándar y obedece a aquel lenguaje mediante el cual se ejecutan consultas y modificaciones de los datos en un modelo de datos relacional. La primera parte del nombre indica que el lenguaje ofrece la posibilidad de desarrollar programas que obedecen al estilo de programación estructurada en el cual toda la pieza de software está constituida por secuencias – instrucciones -, ciclos y decisiones. Si bien este lenguaje de programación ofrece a los programadores la posibilidad de hacer programas orientados a objetos, en Suramericana S.A. esta capacidad no es aprovechada completamente y casi la totalidad de la lógica de negocio de los sistemas transaccionales de seguros se encuentran desarrollados con piezas de software que obedecen a la programación estructurada exclusivamente. Adicionalmente, la existencia de múltiples productos de seguros y el cambio de personal en los equipos de desarrollo son los principales motivos para que la plataforma transaccional de seguros sea a la fecha una pieza de software monolítica y compleja en la que se mezclan características comunes de todos los productos y particularidades de cada uno de ellos.

Desde Enero de 2002 hasta Noviembre de 2011 mi trabajo consistió en realizar actividades de mantenimientos de todas las dimensiones sobre dicha plataforma. Desde cambios menores hasta cambios grandes que buscaban satisfacer necesidades funcionales de las diferentes unidades de negocio de seguros. Realizar dichas actividades de mantenimiento me tomaban una cantidad de tiempo considerablemente alta. En principio consideré que se debía a que no conocía el código fuente y por tanto debía ser cuestión de tiempo el conocer dicho activo de software y su organización para poder realizar cambios con agilidad. El tiempo pasó y lo que esperé que sucediera nunca sucedió; el tiempo que me tomaba implementar un cambio no disminuyó, no llegué a comprender el activo de software en su totalidad y lo peor de todo era que en algunas ocasiones los cambios generaban fallos en ambiente productivo. Consideré una posible falta de habilidades técnicas para lograr el objetivo pero luego de un proceso de verificación, encontré que esa no era la causa. Consulté con todos mis colegas en mi sitio de trabajo y la conclusión fue que comprender dicho activo de software se dificultaba básicamente porque todas las posibles particularidades de todos los productos estaban en un mismo activo, no estaban aisladas y no eran fácilmente identificables. Un cambio menor podría afectar el comportamiento en un producto que no se esperaba alterar. En ese momento comencé a preguntarme si existía alguna aproximación académica que me permitiera comprender el problema y sus posibles soluciones; inicialmente me aproximé al área de estudio de evolución de software para luego comprender, mediante la asesoría de Raquel Anaya, que el enfoque adecuado era abordar el tema desde el estudio de la variabilidad. Fue entonces cuando un problema real en la industria de software se convertía en objeto de estudio de este trabajo de grado.

Se encontró que el estudio de la variabilidad se desarrolla principalmente en las líneas de producto de software – LPS –. Luego de la inspección de referencias bibliográficas se concluyó que era posible adelantar un estudio de variabilidad por fuera del contexto de una LPS. Un producto específico también cuenta con puntos de variación y características variantes y los mecanismos de implementación de los puntos de variación pueden ser mapeados con mecanismos de variación documentados para las LPS. Así las cosas, se dio vía al estudio de variabilidad en un producto específico y no en una LPS, con la cual no se contaba.

Inicialmente se pensó en estudiar la variabilidad del activo de software descrito en los párrafos anteriores, sin embargo, con ánimos de realizar una estudio de variabilidad en un producto cuya

comprensión estuviera completamente al alcance en el tiempo estimado de elaboración de este trabajo se decidió hacer un análisis de variabilidad sobre un producto de software de menor proporción y del cual se pudiera tener pleno conocimiento. Se escogió un producto de software desarrollado entre 2010 y 2012 por Suramericana S.A. para la venta de seguros de automóviles sobre Internet para el cual se conocían todas las consideraciones de arquitectura y diseño.

Este documento tiene como principal motivación la comprensión de un aspecto de los productos de software denominado variabilidad. El manejo adecuado de la variabilidad permite que las diferentes piezas de software de un producto y las relaciones entre ellas sean identificables y modificables fácilmente. De esta forma se logra que el tiempo transcurrido entre la solicitud de modificación por parte del negocio y el transporte del cambio a un ambiente productivo ocurra en un menor tiempo comparado con un manejo inadecuado de la variabilidad.

1.2. Objetivos

Objetivo general

Analizar el nivel de variabilidad y las razones de ésta en una aplicación de software de Suramericana S.A. desarrollada bajo un esquema tradicional de desarrollo de software en el cual la variabilidad no es tenida en cuenta conductor de arquitectura.

Objetivos específicos

1. Determinar el nivel de variabilidad de interés en el producto de software Venta en Línea.

2. Determinar las características que dan razón del nivel de variabilidad de interés en el producto de software Venta en Línea.

1.3. Resumen

En este documento se presenta un estudio de la variabilidad de software en tanto que la capacidad que tiene un producto de software de ser extendido, modificado o personalizado. El estudio de variabilidad es llevado a cabo sobre un producto específico en lugar de ser ejecutado sobre una LPS como lo abordan en la gran mayoría de referencias bibliográficas. Este estudio se realiza de esta forma debido a que no se cuenta con una LPS sobre la cual se pueda realizar el estudio y debido a que luego de un trabajo de inspección del estado del arte del estudio de la variabilidad se concluyó que esta característica de las LPS puede ser estudiada en un producto específico, pues éste también cuenta con características como puntos de variación, variantes y mecanismos de variación.

La necesidad del estudio de la variabilidad sobre un producto de software nace cuando por lo que a priori se considera un mal manejo en la toma de decisiones con respecto a los comportamientos de la plataforma de seguros de Suramericana S.A., la ejecución de cambios en sus piezas de software constituye un proceso traumático en tanto que difícil de saber dónde realizarlo, difícil de probarlo y no ofrece garantías de que funcionará en los escenarios específicos que se quiere que se ejecute; esto hace que los tiempos de implementación de estos cambios sean demasiado altos y en suma hacen muy poco ilegible los componentes núcleo del sistema transaccional de seguros. La búsqueda de una perspectiva estructurada del problema y una posible solución resulta en lo que se documenta en este trabajo.

El marco teórico documentado en la sección 2 expone los conceptos necesarios para la comprensión del concepto de variabilidad. Estos conceptos fueron recopilados de las referencias bibliográficas buscadas en las diferentes fuentes digitales de la Universidad EAFIT entre las cuales se encuentran *Springerlink*,

ACM Digital Library e *IEEE Computer Society*. La sección 3 se describe el caso de estudio en el cual se enmarcará el estudio de variabilidad realizado. En esta sección se presenta la compañía Suramericana S.A. con su razón social, parte principal de su estructura organizacional y algunas cifras importantes acerca de sus resultados y presencia en varios países de Latinoamérica. También se expone la unidad de desarrollo de software de la organización llamada Gerencia de Desarrollo así como algunas de sus cifras importantes de tal modo que el lector se tenga una percepción cuantitativa de la compañía y pueda conocer la dimensión de la misma. En esta sección también se expone la necesidad de negocio que desemboca en la ejecución de un proyecto de software el cual tiene como principal entregable un producto de software llamado Venta en Línea Autos, cuyo objetivo es la venta de seguros de automóviles sobre Internet y sobre el cual se ejecutará el estudio y representación cuantitativa de variabilidad. En la sección 4 se realiza la identificación del tipo específico de variabilidad que se estudiará sobre el producto de software Venta en Línea para posteriormente en la sección 5 realizar una identificación de los puntos de variación del producto y sus respectivas variantes y brindar una descripción de cómo fueron implementadas en el producto de software lo que permite la clasificación de estas características de acuerdo al marco de referencia de Bosch para la representación cuantitativa de la variabilidad. Una vez clasificadas las características se procede con la ejecución de la medición de variabilidad para finalmente analizar los resultados en la sección 6. Las conclusiones acerca del estudio realizado y el planteamiento de trabajos futuros tienen lugar en la sección 7.

2. Marco Teórico

En esta sección se expondrá el marco teórico del estudio de la variabilidad. El objetivo de esta sección es incrementar el nivel de conocimiento del lector de los conceptos necesarios para la comprensión de qué es la variabilidad, cuáles son sus dimensiones, los mecanismos de materialización y de medición de esta característica de los productos de software. Con esta exposición se busca formar una base conceptual que permita la comprensión de la medición de variabilidad que se llevará a cabo en secciones posteriores de este trabajo.

2.1. Líneas de Producto de Software (LPS)

Con el objetivo de establecer el contexto adecuado, en esta sección se explorarán referencias acerca del concepto Línea de Producto de Software. Líneas de producto de software es un paradigma de desarrollo de familias de productos de software creado por el SEI (Software Engineering Institute) de la universidad Carnegie Mellon.

La exploración de la documentación del SEI y referencias bibliográficas adicionales conllevan a numerosas pero similares definiciones de lo que es una LPS. La primera definición histórica de LPS encontrada en la exploración bibliográfica realizada la brinda Brownsword y Clements en 1996 (Brownsword y Clements 1996) de la siguiente manera:

Una línea de producto es un conjunto de sistemas relacionados que está dirigido a un segmento de mercado. Construir una línea de producto a partir de un conjunto de elementos reutilizables comunes en lugar de construir cada sistema por separado potencializa el reuso.

Brownsword y Clements (Brownsword y Clements 1996) introducen con contundencia lo que serán las definiciones posteriores de LPS. Adicionalmente, desde este primer reporte técnico se introduce el concepto de *core asset,* el cual será traducido como *elementos reutilizable común*. Un elemento reutilizable común es un artefacto del proceso de desarrollo de software que permite ser modificado, adaptado o complementado para constituir un artefacto de software nuevo. Por artefacto se debe comprender cualquier entregable del proceso de desarrollo en todo su ciclo, así, son ejemplos de estos arquitecturas, diseños detallados, código fuente, productos empaquetados desplegables, planes de pruebas, casos de pruebas y demás.

Siendo un poco estrictos con la terminología, la definición brindada por Brownsword y Clements es un poco incipiente. Para 2002 el mismo Clements & Northrop (Clements y Northrop 2002) brindan una definición más rigurosa del concepto LPS que indica lo siguiente:

"Una línea de producto de software es un conjunto de sistemas basados en software que comparten un conjunto común y administrado de características que satisfacen las necesidades específicas de un segmento particular de mercado y que son desarrollados a partir de un conjunto común de activos núcleo de una manera prescrita".

Es evidente que la definición es más rigurosa y formal en (Clements y Northrop 2002) y se verá que alrededor de ésta giran las demás encontradas en otras referencias.

Bosch es un autor reconocido y frecuentemente citado en el estudio de la variabilidad en LPS. El objetivo de las publicaciones de Bosch no es definir ni teorizar acerca de las LPS (su tema central de investigación es la variabilidad en las LPS), sin embargo es posible encontrar referencias en sus artículos acerca de definiciones de LPS, como las siguientes: Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) indica que la idea principal de una LPS es desarrollar una infraestructura de reuso que soporte el desarrollo de una familia de productos de software. Asimismo

asemeja las LPS y la orientación a objetos en la medida en que ambas buscan flexibilidad y reuso (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002). Adicionalmente, Bosch determina en esta referencia cómo en una LPS la arquitectura de los sistemas de la familia es determinada en una etapa temprana del proceso de desarrollo pero los detalles de implementación de un producto específico de la familia son postergados hasta la etapa de implementación.

Finalmente y para no dejar por fuera de esta recopilación a las referencias adicionales encontradas, se hace referencia a una última definición encontrada, mucho menos rigurosa y formal pero que con un esfuerzo adicional del lector se podrá concluir que también gira en torno a la definición de Brownsword y Clements. De este modo, hacia 2009 se definía una LPS como una nueva metodología que desarrolla productos de software a partir de la configuración de otros productos de software existentes en un repositorio (Elfaki, Phon-Amnuaisuk y Kuan Ho 2009).

A modo de cierre se puede establecer como definición principal de una LPS la brindada por Brownsword y Clements (Clements y Northrop 2002) debido a su completitud y referencias que la citan.

2.2. Conceptos de Variabilidad

2.2.1. Variabilidad

El contexto principal en el que tenida en cuenta la variabilidad del software como objeto de estudio son las Líneas de Producto de Software (LPS). En este punto se considera prudente establecer cómo las referencias examinadas definen la variabilidad de software con el fin de que el lector y las secciones subsiguientes de este trabajo tengan estas definiciones como descripciones concretas del objeto de estudio.

Al ser las LPS un producto del SEI[1], la principal y más clara referencia acerca de la variabilidad la elabora Clements (Clements y Northrop 2002). En este texto, Clements aborda la variabilidad de una manera amplia y extensa. El autor nos brinda la siguiente definición de variabilidad:

"Variabilidad es la capacidad de un sistema, un activo, o un ambiente de desarrollo de soportar la producción de un conjunto de artefactos que se diferencian entre sí de una manera pre-planeada. Variabilidad es la capacidad de un activo núcleo de adaptarse a diferentes usos en diferentes contextos de productos pertenecientes al ámbito de la familia de productos. En esta definición se encuentra implícito el hecho de que las variaciones son planeadas en una línea de producción de software. Los desarrolladores de los activos núcleo han pensado en las consecuencias de las variaciones y, presumiblemente, los han restringido de tal forma que el activo núcleo resultante soporte los requerimientos futuros y ha tenido en cuenta el tiempo y presupuesto disponible para crear estos activos."

Bosch (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002), otro autor recurrente en temas de variabilidad es un poco menos estricto y define la variabilidad de la siguiente manera:

"Variabilidad es la capacidad que tiene un sistema para ser cambiado o personalizado[2]."

Bosch & Jaring (Bosch y Jaring, A taxonomy and hierarchy of variability dependencies in software product family engineering 2004) complementa la definición con lo siguiente:

[1] Software Engineering Institute
[2] Del inglés *customize.*

"Variabilidad es la capacidad que ofrece un sistema o artefacto de software para ser extendido, cambiado, personalizado o configurado para su uso en un contexto particular."

Las definiciones de Clements y Bosch están enmarcadas en el estudio de las LPS. Por fuera del contexto de las LPS se encuentran definiciones más generales y más antiguas de la variabilidad en los sistemas fuertemente basados en software como la de Coplien (Coplien, Hoffman y Weiss 1998), quien describe la variabilidad como aquello que solo es verdadero para algunos elementos de S, donde S es un conjunto de objetos. Se puede afirmar que las unidades del conjunto S en esta definición son productos de software y que *aquello que solo es verdadero* hace referencia a las *Variantes* de Clements & Northrop (Clements y Northrop 2002).

Las definición de Bosch & Jaring (Bosch y Jaring, A taxonomy and hierarchy of variability dependencies in software product family engineering 2004) será la adoptada en este trabajo como definición de variabilidad. Con base en esta definición y en las anotaciones que se realizarán a continuación en esta sección, se llevará a cabo el trabajo de medición de variabilidad propuesto en este documento.

Brownsword y Clements (Clements y Northrop 2002) (Brownsword y Clements 1996), Northrop (Northrop s.f.), Lauenroth & Pohl (Lauenroth y Pohl 2005) y Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) son autores que estudian la variabilidad en el marco de las LPS. Esto quiere decir, en principio, que los conceptos de variabilidad estudiados por estos autores son válidos únicamente para productos miembros de una familia generados bajo un enfoque de LPS (técnicas, herramientas, marco de trabajo y demás). Sin embargo la lectura de estos textos y el estudio cuidadoso de sus conceptos nos llevan a concluir que la Variabilidad puede ser objeto de estudio en productos de software que son similares pero que no necesariamente se construyeron en una LPS. Sin embargo, los conceptos de variabilidad expuestos por los autores permiten determinar que ésta puede ser estudiada en un producto de software que no haya sido generado en una LPS. En un producto de este tipo aquello que varía no sería variable entre diferentes productos sino características internas o propias de un solo producto de software.

Finalmente, después de haber examinado las diferentes definiciones de variabilidad, es importante resaltar dos características fundamentales y sus respectivas aclaraciones:

La variabilidad no sólo está presente en los artefactos ejecutables. Un producto de software está constituido por todos los entregables y artefactos de las diferentes fases del proceso de desarrollo utilizado para su construcción. Clements afirma que la variabilidad se presenta tanto en los elementos reutilizables comunes del producto como en su ambiente de desarrollo. Se concluye, a falta de especificación detallada (Clements y Northrop 2002), que el ambiente de desarrollo en esta definición constituye todo aquello diferente del elemento reutilizable común en tanto que unidad de software como unidad de despliegue en un ambiente de ejecución determinado. De manera complementaria, y como se expondrá en la sección *Dimensiones de la Variabilidad*, se hace referencia a Martínez-Ruiz (Martínez-Ruiz, y otros 2010), quien abarca el tema de variabilidad exclusivamente en el proceso de desarrollo de software.

Variabilidad de software no es lo mismo que Evolución de software. El término Evolución de Software y su relación con el término Mantenimiento de Software es expuesto de manera consistente por Chapin (Chapin, y otros 2001). De esta referencia se recalca que en el mejor y más crítico de los escenarios, el de mantenimiento de reglas de negocio en el código fuente, las actividades de mantenimiento se llevan a cabo de manera planeada. Esto no quiere decir que el software como producto (código) esté preparado para incluir variantes que constituyan los mantenimientos como tal. Por otra parte, de la definición de Clements (Clements y Northrop 2002) destaca el hecho de que las variaciones en los activos núcleo son **planeadas** y que por ende los arquitectos, diseñadores y desarrolladores de estos activos son conscientes de que vendrán cambios futuros en el activo en proceso

de desarrollo. De esta forma, las actividades de mantenimiento en un producto de software son cambios sobre el código fuente que en el mejor de los casos se realizan de manera planeada. Esto no significa que el código fuente esté preparado para dicha modificación. Las actividades de desarrollo de variantes en un producto de software el cual es miembro de una familia de productos no constituyen cambios sobre el elemento común reutilizable, por el contrario, dependiendo del mecanismo de variabilidad utilizado, no debe suponer ningún cambio en dicho elemento. Aclarado esto, se concluye que Variabilidad de software no es lo mismo que Evolución de Software. Para finalizar esta anotación, se hace referencia a Lauenroth & Pohl (Lauenroth y Pohl 2005) quienes relacionan los cambios en las variantes y la denominan *Variabilidad en el Tiempo*, la cual es diferente a la *Variabilidad en el Espacio*, objeto de estudio de este trabajo y definida por los mismos Lauenroth & Pohl (Lauenroth y Pohl 2005).

2.2.2. Procesos de definición de variabilidad

Los elementos reutilizables comunes hacen parte fundamental en los diferentes procesos de definición de variabilidad en una LPS. Northrop documenta (Northrop s.f.), como se puede ver en la Ilustración 1, los tres grandes procesos mediante los cuales se generan productos en una LPS.

Ilustración 1: Procesos principales en la construcción de productos de software mediante una LPS. Tomado de **(Northrop s.f.)**.

La variabilidad es definida por las restricciones, estrategias y consideraciones del negocio. Estos elementos de negocio son las condiciones particulares de la organización que determinan qué es común y qué es variable en un producto de software. La Ilustración 2 muestra de forma gráfica cómo aspectos de negocio generan una base de elementos reutilizables comunes a partir de los cuales se construirán nuevos elementos reutilizables.

La variabilidad se materializa en la construcción de productos específicos pues en es esta etapa de producción en la cual se desarrollan las variantes propias de cada producto. En la Ilustración 3 se puede observar cómo la variabilidad definida en el proceso *Desarrollo de elementos reutilizables comunes* se materializa mediante el desarrollo de nuevas variantes que combinadas con los activos núcleo constituyen productos de una familia de productos de software.

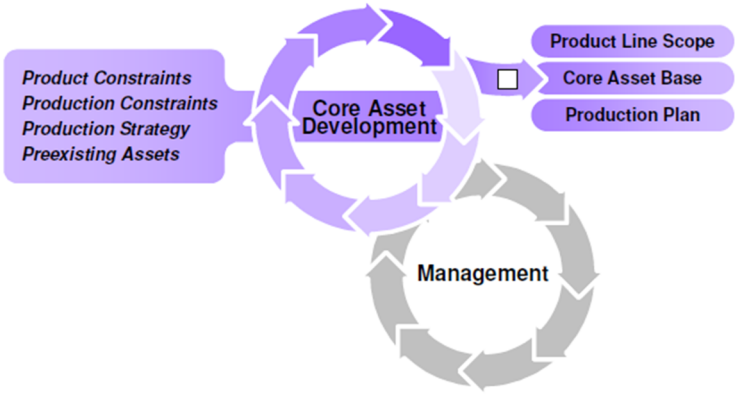


Ilustración 2: Representación gráfica de cómo las restricciones de negocio inciden en la producción de activos núcleo. Esto sucede en el proceso de construcción de activos núcleo según Northrop (Northrop s.f.).

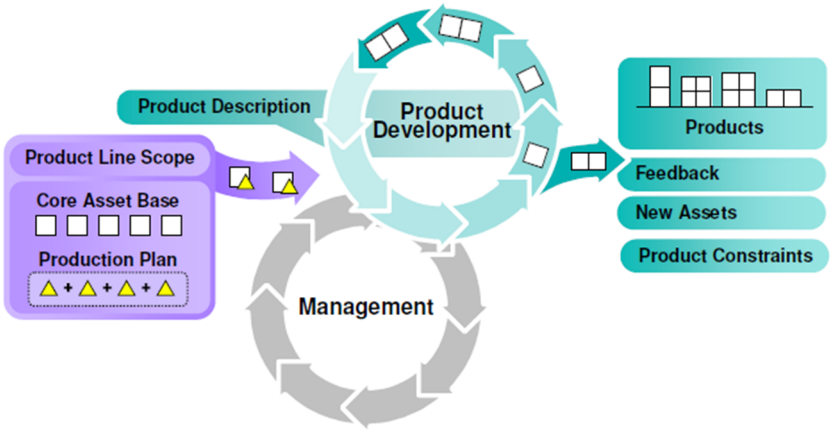


Ilustración 3: Representación gráfica de cómo en el proceso de Desarrollo de Producto se generan nuevos productos y se materializa la variabilidad según Northrop (Northrop s.f.).

La conclusión preliminar de acuerdo a Northrop (Northrop s.f.) con respecto a la variabilidad es que ésta nace en el proceso de *Desarrollo de Activos Núcleo* determinada por restricciones del negocio y que se materializa en el proceso de *Desarrollo de Productos* mediante la implementación de productos específicos.

Lauenroth & Pohl (Lauenroth y Pohl 2005), por su parte, describen dos procesos principales en el desarrollo de familias de productos con una LPS. Estos procesos principales son la *Ingeniería de Dominio* y la *Ingeniería de Aplicación*. Estos dos procesos son correspondientes con los anteriormente descritos por Northrop (Northrop s.f.) con respecto a la variabilidad y coinciden en que ésta es un aspecto clave de los artefactos de dominio. En este trabajo se determina que los artefactos de dominio de Lauenroth & Pohl (Lauenroth y Pohl 2005) se relacionan directamente con los activos núcleo de Northrop (Northrop s.f.). En la Ilustración 4 se puede ver gráficamente la separación entre los dos procesos y sus artefactos. Es importante notar la diferenciación gráfica que se hace entre los artefactos de dominio y los artefactos de aplicación, constituyendo estos últimos los productos específicos de una familia de productos.

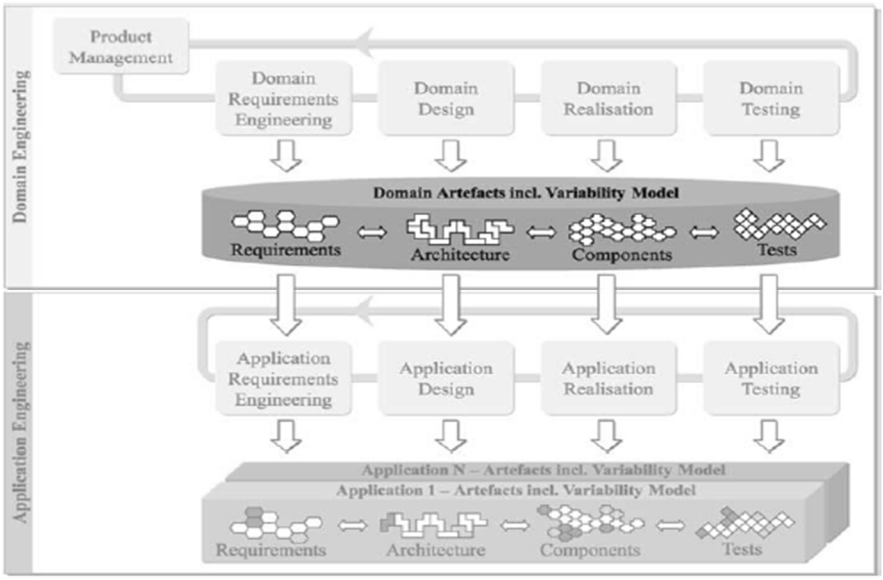


Ilustración 4: Representación gráfica de Lauenroth & Pohl de los dos procesos principales de una LPS: Ingeniería de Dominio e Ingeniería de Aplicación. Tomado de **(Lauenroth y Pohl 2005)**.

Así pues, queda establecida la relación entre LPS y Variabilidad. Los productos de una familia son constituidos por elementos reutilizables comunes y por variantes propias de cada producto. Las variantes se construyen en el proceso de ingeniería de aplicación de cada producto específico pero la planeación de su existencia es determinada en el diseño y construcción de los elementos comunes reutilizables. En este documento se acoge la terminología de Lauenroth & Pohl (Lauenroth y Pohl 2005) y por tanto se determina que la variabilidad nace en el proceso de Ingeniería de Dominio y se materializa en el proceso de Ingeniería de Aplicación. Los conceptos Característica y Variante serán expuestos en una sección posterior de este documento.

2.2.3. Punto de Variación

La primera definición encontrada en la literatura para el concepto de Punto de Variación la expone Jacobson (Jacobson, Griss y Jonsson 1997) indicando que se trata de la localización de uno o más puntos donde una variación puede ocurrir. El contexto de la definición de Jacobson es orientada a modelamiento en UML de productos de software.

Lauenroth & Pohl (Lauenroth y Pohl 2005) ubican estos puntos de variación en un contexto específico y los define como la representación de un sujeto de variabilidad. Un sujeto de variabilidad es aquello que responde a la pregunta **¿Qué es lo que varía en un producto de software?** La definición indica que un sujeto de variabilidad es un ítem variable del mundo real o una propiedad de dicho ítem. Un punto de variación, como ya se enunció, es la representación de estos sujetos de variación de Lauenroth & Pohl (Lauenroth y Pohl 2005).

Clements & Northrop (Clements y Northrop 2002) argumentan que es preferible utilizar el término *Parte Variable* en lugar de *Punto de Variación*, por dos razones: 1) una parte variable describe más que solo la ubicación de una variante, por ejemplo, el conjunto de mecanismos de variación que llevan a cabo estas variantes y 2) un punto de variación generalmente se utiliza para denotar variabilidad externa, es decir, aquella que se refiere a propiedades visibles de forma externa al activo de software y no tiene en cuenta la variabilidad en la estructura interna del activo.

La definición de Jacobson (Jacobson, Griss y Jonsson 1997) es la primera cronológicamente hablando. La definición de Lauenroth & Pohl (Lauenroth y Pohl 2005) es la más acertada y apropiada para la elaboración de este trabajo y por eso será adoptada como la definición de Puntos de Variación. Las anotaciones de Clements & Northrop (Clements y Northrop 2002) son válidas pero se prefiere comprender la definición de Puntos de Variación desde un punto de vista teórico y posteriormente abordar separadamente el tema de los mecanismos de variación, como se hará en la sección 2.2.5 de este documento.

2.2.4. Variantes y Características

2.2.4.1. Definiciones

Los valores que pueden tomar lugar en ese punto de variación son denominadas Variantes. Las definiciones más concretas de Variante las brinda Clements & Northrop (Clements y Northrop 2002) y Lauenroth & Pohl (Lauenroth y Pohl 2005). Para Clements & Northrop, una variante es la realización de una parte variable. Para Lauenroth & Pohl una variante es la representación de un objeto de variación, donde un objeto de variación es una instancia particular de un sujeto de variabilidad.

Acudiendo al mismo ejemplo de la producción de vehículos con transmisión mecánica o automática, el sujeto de variabilidad está constituido por la transmisión mientras que los diferentes objetos de variabilidad son la transmisión mecánica y la transmisión automática.

Bosch define el término Característica. Una Característica[3] es definida muy específicamente para sistemas de software de la siguiente manera (Bosch, Design & Use of Software Architectures - Adopting and Evolving a Product Line 2000):

"*Una característica es una unidad lógica de comportamiento la cual está especificada por un conjunto de requerimientos funcionales y de calidad".*

La diferencia con las definiciones de Clements y Lauenroth & Pohl se encuentra en que Bosch ubica las variantes en algo tan concreto como lógicas de comportamiento, es decir, características que finalizan siendo porciones de código fuente que ejecutan un comportamiento determinado dentro de una aplicación de software.

Adicionalmente, Bosch (Bosch, Design & Use of Software Architectures - Adopting and Evolving a Product Line 2000) clasifica las características de tal forma que es fácil encontrar a las Variantes en una de esas clasificaciones. La clasificación de las características según Bosch (Bosch, Design & Use of Software Architectures - Adopting and Evolving a Product Line 2000) es la misma realizada por Griss (Griss, Favaro y d'Alessandro 1998) con una categoría adicional (la de Características Externas). La siguiente es la clasificación expuesta en (Bosch, Design & Use of Software Architectures - Adopting and Evolving a Product Line 2000):

- Características externas
- Características obligatorias
- Características opcionales
- Características variantes

Las características variantes son aquellas que están presentes en un producto de forma variante, es decir, con comportamientos diferentes en escenarios o condiciones particulares dadas.

[3] Del inglés *Feature*.

Según lo anterior, las características variantes de Bosch (Bosch, Design & Use of Software Architectures - Adopting and Evolving a Product Line 2000) son los mismos puntos de variación de Clements & Northrop (Clements y Northrop 2002) y de Lauenroth & Pohl (Lauenroth y Pohl 2005) y se convertirán en el principal objeto de estudio en la elaboración de este trabajo debido a que es la capacidad de introducción de variantes (en su respectivo punto de variación) que posee un sistema específico.

2.2.4.2. Procesos de transformación de las variantes

Habiendo establecido una igualdad entre los conceptos *características variantes* y *variante* de los diferentes autores, se expondrán los procesos de transformación que pueden tomar éstas. Estas dimensiones obedecen a las diferentes etapas de un ciclo de desarrollo de software y son expuestas por Bosch (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002). La Ilustración 5 muestra cómo una característica (variante o no) nace siempre como una necesidad de un usuario y sufre transformaciones a lo largo del ciclo de desarrollo hasta convertirse en código en ejecución.

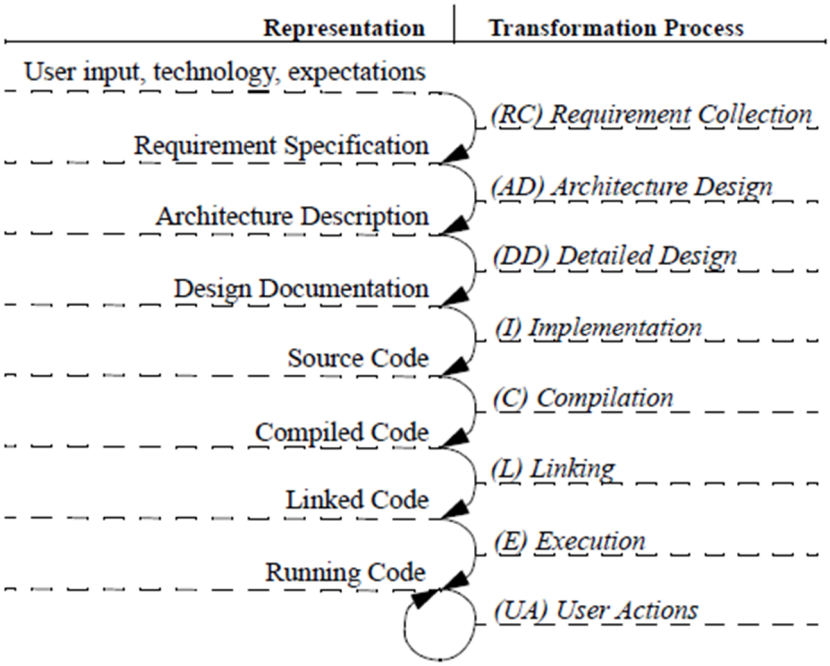


Ilustración 5: Procesos de transformación de una característica desde su nacimiento como necesidad de un usuario hasta ser código en ejecución. Tomado de (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002).

Lo más interesante de la Ilustración 5 es que siempre las variantes serán implementadas. Esto quiere decir que el sistema que requiera variabilidad en su comportamiento siempre lo logrará pues una necesidad de un usuario siempre deberá ser materializada en el comportamiento de la aplicación o del producto de software. Pero la pregunta que surge es ¿Cuál es la mejor manera de hacer llegar una necesidad de variabilidad de un usuario hasta ser código en ejecución de tal forma que se obtengan beneficios tales como calidad en la implementación y respuestas rápidas ante la necesidad de modificación del comportamiento? Esta pregunta queda abierta en este punto del documento y será resuelta para un producto específico de software, objeto de estudio de este trabajo.

Para ofrecer elementos de sustento a la respuesta, se deben exponer los conceptos de fase de introducción y fase de enlace de una característica en un producto de software. A continuación las definiciones.

2.2.4.3. Fase de introducción de una variante

La fase de introducción de una variante corresponde con el momento o proceso de transformación en el cual el mecanismo de variación de dicha variante está abierto para la definición de nuevas alternativas de comportamiento. Para comprender este concepto de fase de introducción es necesario comprender el concepto de mecanismo de variación, el cual es explicado en la sección *Mecanismos de Variación*. Se debe tener en cuenta que la **definición** de una alternativa de comportamiento es diferente de su **implementación**, así como definir una clase en un diagrama de clases UML no significa que dicha clase existirá en el producto de software. Sin embargo, es necesario explicar que existen ciertos mecanismos de variación en los cuales la definición y la implementación se realizan en la misma fase. Un ejemplo de este tipo de mecanismo de variación es *Condición en variable* (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) en el cual las variantes son definidas en el momento de la codificación mediante la inserción de bloques *if-else* en el código fuente del producto de software. En ese mecanismo de variación la fase de introducción de una nueva variante es la fase de implementación pues es en el momento de construcción en el cual se tiene la oportunidad de definir nuevas opciones de comportamiento. Otro ejemplo que sirve para exponer el concepto de fase de introducción es el del mecanismo *Implementación de componentes coexistentes*, el cual es generalmente materializado mediante patrones de diseño como *Strategy* (Gamma, y otros 1995); en este mecanismo, las variantes se pueden definir únicamente en tiempo de diseño. Las características que se sigan este patrón de variabilidad tienen fase de introducción en etapa de diseño.

2.2.4.4. Fase de enlace de una variante

La fase de enlace de una variante corresponde con alguno de los dos siguientes momentos:

1. El momento en el cual una variante es seleccionada para hacer parte de un producto específico.
2. El momento en el cual una variante es seleccionada para ser ejecutada en un producto específico.

La comprensión de la fase de enlace se habilita a partir de la exposición de los siguientes escenarios:

En el primer caso, un punto de variación puede contener múltiples variantes pero solo una de ellas estará presente en un producto específico de software; en este caso la fase de enlace corresponde al momento en el cual se selecciona dicha variante para que haga parte del producto final. En este caso la fase de enlace generalmente corresponde a las etapas de compilación, enlace o instalación.

En el segundo caso, cuando un punto de variación contiene múltiples variantes y todas ellas deben estar presentes para una posible selección para ejecución, la fase de enlace se refiere al tiempo en el cual el producto de software realiza una selección de estas variantes, generalmente, el tiempo de ejecución.

La selección de variantes no es esperada en todos los niveles de la Ilustración 5. Generalmente, los momentos de selección de variantes son las fases de compilación, enlace, instalación o ejecución del producto (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002).

Con respecto a la fase de enlace de variantes, es importante anotar que la actividad de enlace puede ser interna o externa. Cuando un sistema realiza el enlace de variantes de forma interna significa que el propio sistema cuenta con los mecanismos de enlazar una variable. Ahora, cuando el producto de

software requiere funcionalidades de un sistema externo para enlazar las diferentes variantes, se habla de un enlace externo.

2.2.5. Mecanismos de Variación

2.2.5.1. Definiciones

Ya se ha expuesto el concepto de LPS y cómo la variabilidad se estudia formalmente en ese enfoque de producción de sistemas de software. Adicionalmente, se han expuestos conceptos claves para la comprensión de la variabilidad como las definiciones en la literatura considerada, las variantes y características con sus procesos de transformación y sus diferentes fases en un producto de software, a saber: fase de introducción y fase de enlace. Ahora la pregunta que puede surgir es ¿Cómo se materializan todos estos conceptos de variabilidad en un producto de software?

Bachman & Clements (Bachman y Clements 2005) responden a esta pregunta indicando que son los mecanismos de variación los que permiten encapsular las partes variantes y seleccionar una adecuada en unas condiciones particulares. Bachman & Clements (Bachman y Clements 2005) exponen el concepto de mecanismo de variación de la siguiente manera:

Entiéndase un componente que hace parte de la base de activos núcleo de la LPS de préstamos bancarios. Este componente determina la cantidad máxima que puede pedir prestada una persona. Por regulaciones estatales, existe un estado donde el cálculo es un poco diferente al estándar. Este "poco diferente" indica que es posible reusar la mayor parte de la lógica estándar, pero que variará en algunos detalles de cálculo. El diseñador del activo núcleo debe entonces planear que el sistema debe calcular la cantidad máxima permitida de dos modos, estándar o por región específica. Para esto ha diseñado un *switch* el cual cuando está activado de un lado ejecuta la lógica estándar, pero cuando está activado para el lado contrario ejecutará la lógica estándar y la lógica específica de la región. A ese *switch* se le denomina *Mecanismo de Variación*.

Bachman & Clements (Bachman y Clements 2005) hacen énfasis en que la selección de un mecanismo de variación depende de varios elementos del equipo de trabajo como por ejemplo las habilidades con las que cuenten los diseñadores y desarrolladores de los productos así como también las restricciones de tiempo y presupuesto con las que cuente un proyecto al momento de seleccionar un determinado mecanismo. Así pues, si para implementar determinado mecanismo de variación es necesario tener mucha experiencia en desarrollo sobre bases de datos pero el equipo de desarrollo no es fuerte en ese aspecto entonces se debe desechar la posibilidad de selección de ese mecanismo. Así mismo, si la implementación de un mecanismo de variación toma más tiempo del disponible para la ejecución de un proyecto crítico para la organización entonces, en ese escenario, ese mecanismo también debe ser desechado.

A continuación se detallan los aspectos que deben ser tenidos en cuenta en el estudio y selección de un mecanismo de variación de acuerdo con Bachman & Clements (Bachman y Clements 2005):

- Las habilidades técnicas necesarias para la implementación del mecanismo.
- El costo de implementación del mecanismo.
- El costo y el tiempo necesarios para ejercitarse en el desarrollo del mecanismo.
- El grupo objetivo para el cual se está diseñando la adaptación del mecanismo de variación.
- El impacto del mecanismo de variación en la calidad del producto final, tales como impactos en los tiempos de respuesta o consumo de recursos como la memoria o procesador de la máquina.
- El impacto que genere la mantenibilidad del mecanismo.

2.2.5.2. Patrones de variabilidad y catálogo de mecanismos de variación de Bosch

Bosch (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) indica que los mecanismos de variación son aquellos que permiten a los arquitectos y diseñadores postergar decisiones acerca de qué variante seleccionar en un punto de variación específico en una fase específica del proceso de desarrollo. De modo adicional Bosch (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) indica que el uso de mecanismos de variación ha estado presente en los productos de software pero generalmente se tratan de soluciones ad-hoc.

El trabajo de Bosch (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) es de suma importancia en este marco teórico debido a que su inventario de mecanismos de variación es el más completo encontrado en las referencias inspeccionadas y por ende se convierte en la referencia a mostrar en este punto del documento. La Tabla 2 muestra los mecanismos de variación identificados por Bosch en un estudio de cuatro productos de software específicos. Estos mecanismos se muestras agrupados por lo que el autor denomina patrón de variabilidad. Un patrón de variabilidad no es más que la forma en la que se presenta la cantidad de variantes de las cuales dispone un punto de variación al momento de realizar una selección ante unas condiciones dadas. Los patrones de variación son expuestos en la Tabla 1.

Nombre del patrón	Descripción del patrón
Entidad Variante	Existen muchas variantes, pero una y solo una estará activa en el sistema de software.
Entidad Opcional	Es similar al patrón Entidad Variante, solo que siempre hay disponible una sola variante.
Entidades múltiples y coexistentes	El sistema en ejecución contiene varias variantes posibles. La decisión de cual escoger es postergada hasta el tiempo de ejecución.

Tabla 1: Patrones de variación de Bosch **(Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002)**. Tomado de **(Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002)**.

Nombre del patrón	Fase de enlace[4]	Fase de introducción	Nombre del mecanismo
Entidad Variante	Derivación de arquitectura	Arquitectura	Reorganización de

[4] Del inglés *Binding Phase*.

	del producto		arquitectura
			Componente de arquitectura variante
		Diseño detallado	Especialización de componente variante.
	Compilación	Implementación	Condición en constante
		Compilación	Superimposición de fragmento de código
	Enlace	Enlace	Reemplazo de binarios – Directivas de compilación
			Reemplazo de binarios - Físico
	Tiempo de ejecución	Arquitectura	Arquitectura centrada en infraestructura
		Implementación	Condición en variable
Entidad Opcional	Derivación de arquitectura del producto	Arquitectura	Componente de arquitectura opcional
		Diseño detallado	Especialización de componente opcional
	Compilación	Implementación	Condición en constante
	Tiempo de ejecución	Implementación	Condición en variable
Entidades multiples y coexistentes	Tiempo de ejecución	Diseño detallado	Implementación de componentes coexistentes.
			Especialización de

			componentes coexistentes.
		Implementación	Condición en variable

Tabla 2: Mecanismos de variación de Bosch **(Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002)**. Tomado de **(Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002)**.

A continuación se expondrá la intención de cada uno de los mecanismos de variación de la Tabla 2.

Reorganización de arquitectura. Soportar la arquitectura específica de muchos productos mediante la reorganización de la arquitectura de toda la línea de productos.

Componente de arquitectura variante. Soportar varios y diferentes componentes de arquitectura los cuales representan una misma entidad.

Arquitectura centrada en infraestructura. Hacer de las conexiones entre componentes una entidad de primera clase.

Especialización de componente variante. Ajustar la implementación de un componente a la arquitectura del producto específico.

Condición en constante. Soportar varias maneras de ejecutar un comportamiento. Solo uno de los comportamientos se presentará en un producto específico.

Condición en variable. Soportar varias maneras de ejecutar una operación, de las cuales solamente una tomará lugar en un momento dado. Sin embargo, brinda la oportunidad de escoger otra de las distintas maneras en tiempo de ejecución.

Superimposición de fragmento de código. Introducir nuevas consideraciones (comportamientos) en un sistema sin afectar directamente el código fuente.

Reemplazo de binarios – Directivas de compilación. Proveer al sistema con diferentes implementaciones de librerías subyacentes.

Reemplazo de binarios – Físico. Facilitar la modificación de un producto software en algún momento posterior a la implantación.

Componente de arquitectura opcional. Preparar al sistema para un componente que puede o no estar presente.

Especialización de componente opcional. Incluir o excluir partes de comportamiento de la implementación de un componente.

Implementación de componentes coexistentes. Soportar implementaciones concurrentes y coexistentes de un componente arquitectónico.

Especialización de componentes coexistentes. Soportar la existencia y selección de diferentes especializaciones que se encuentran dentro de una implementación de componente.

Cada uno de estos mecanismos de forma adicional a la intención que se ha expuesto en este documento está especificado de forma completa con su respectiva motivación, solución, ciclo de vida, consecuencias y ejemplos en (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002). En caso de requerir detalles adicionales acerca de estos mecanismos, se sugiere al lector la inspección de esta referencia específica para no incurrir en una transcripción literal del trabajo de Bosch en este marco teórico.

Finalmente, es necesario anotar que el inventario de mecanismos de variación de Bosch (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) es completo y abarca todas las posibilidades de inserción de variantes en un producto de software. Para efectos de la ejecución de la medición de variabilidad en secciones posteriores de este documento será la referencia de mecanismos de variación.

2.3. Dimensiones de la Variabilidad

Los productos de una familia de productos de software tienden a variar (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002). Los comportamientos de una aplicación que no sea concebida precisamente bajo el enfoque LPS también tienen a variar. Esta afirmación es amplia, pues supone que la variabilidad de un producto de software puede ser abordada desde múltiples puntos de vista como lo son el del usuario final, el punto de vista desde el proceso mediante el cual se desarrolló o desde una perspectiva más técnica que aborde el producto de software desde los artefactos de cada etapa del proceso de desarrollo. A partir de esta afirmación surgen las siguientes preguntas ¿Qué es lo que tiende a variar? ¿Cuál es el foco de estudio específico en la Variabilidad de Software?

Lauenroth & Pohl (Lauenroth y Pohl 2005) establecen tres preguntas para la comprensión y caracterización de las diferentes dimensiones de la variabilidad, éstas preguntas son

- ¿Qué es aquello que varía?
- ¿Por qué varía?
- ¿Cómo varía?

Del trabajo de Lauenroth & Pohl (Lauenroth y Pohl 2005) nos interesa en mayor parte las siguientes dimensiones documentadas:

2.3.1. Variabilidad espacial y temporal

La diferencia entre estos dos tipos de variabilidad se comprende mucho mejor mirando la Ilustración 6. En esta ilustración, Lauenroth & Pohl (Lauenroth y Pohl 2005) exponen como en una familia de productos de domótica se planea tener dos sistemas de seguridad en el ingreso al lugar de residencia. Uno de los productos, el de la línea económica, tendrá un control de acceso mediante tarjetas magnéticas. El otro producto, el de la línea profesional, contará con un control de acceso mediante un teclado. La figura ilustra cómo la Variabilidad en el Espacio es aquella en la que cobra sentido la variación de los diferentes sistemas de seguridad. La Variabilidad en el Tiempo, por su parte, hace referencia a los cambios que sufrirá cada tipo de control de acceso con el paso del tiempo. Aunque no es explícito en (Lauenroth y Pohl 2005) es posible identificar en el concepto de Variabilidad en el Tiempo el concepto de Mantenimiento y Evolución de software expuestos con detalle por Chapin (Chapin, y otros 2001).

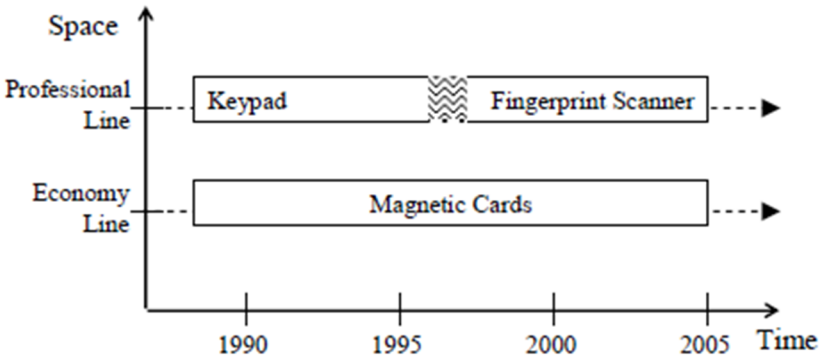


Ilustración 6: Diferenciación gráfica entre Variabilidad en Espacio y Variabilidad en Tiempo propuesta por Lauenroth & Pohl **(Lauenroth y Pohl 2005)**. Tomado de **(Lauenroth y Pohl 2005)**.

La variabilidad como objeto de estudio de interés es la Variabilidad en el Espacio (Lauenroth y Pohl 2005). Una vez establecido esto, los autores proceden con la definición de las siguientes dimensiones más específicas de variabilidad en el espacio en un producto de software:

2.3.2. Variabilidad externa

La variabilidad externa es la variabilidad de los artefactos de dominio que es visible a los usuarios del producto de software (Lauenroth y Pohl 2005).

2.3.3. Variabilidad interna

Así como la variabilidad externa es visible a los usuarios del sistema, la variabilidad interna es aquella que no es seleccionada por éstos y que es tenida en cuenta a nivel técnico tanto para lograr satisfacer las necesidades establecidas por la variabilidad externa como para acometer objetivos técnicos de cualidades sistémicas como la mantenibilidad y el rendimiento (Lauenroth y Pohl 2005).

2.4. Variabilidad por fuera del contexto de las LPS

Hasta este momento se han introducido definiciones de LPS y de conceptos de variabilidad. Se ha especificado cómo la variabilidad es objeto de estudio formal en el contexto de las LPS y que es en ese enfoque donde se debe ser riguroso con su manejo. Sin embargo, las necesidades de variación de los productos de software no nacen exclusivamente con el enfoque LPS. Referencias bibliográficas que se encuentran por fuera del nombre específico LPS como por ejemplo Gamma (Gamma, y otros 1995) y Jacobson (Jacobson, Griss y Jonsson 1997) ya tratan el tema de la variación en productos de software. El primero con la formalización de patrones de diseño como *Estrategy*, *Adapter* y *Facade* y el segundo con la introducción del concepto de Punto de Variación.

De igual forma, todos los trabajo de Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) (Bosch y Jaring, A taxonomy and hierarchy of variability dependencies in software product family engineering 2004) (Bosch, Design & Use of Software Architectures - Adopting and Evolving a Product Line 2000) pueden ser analizados y aplicados a un producto de software sin que se cumpla la precondición de que éste haya sido generado bajo un enfoque LPS o incluso de que el producto pertenezca a una familia de productos.

Los conceptos entonces de variabilidad, características y variantes, puntos de variación y mecanismos de variación aplican perfectamente a un solo producto de software y en esta medida la variabilidad puede ser estudiada en ese producto. De igual forma, como la variabilidad en una familia de productos puede ser medida, en un solo producto también puede ejecutarse esta tarea de medición. Con los resultados de esta medición pueden obtenerse conclusiones de las características técnicas del producto que dan cuenta del nivel de variabilidad de dicho sistema, repitiendo de nuevo que este sistema no es necesario que pertenezca a una familia de productos y mucho menos tiene que haber sido generado bajo un enfoque LPS.

Aclarado esto, se procede con la exposición, en la siguiente sección, de cómo se ejecuta la medición de la variabilidad en un producto de software específico de acuerdo con un marco de referencia definido por Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002).

2.5. Representación de la Variabilidad

Representar significa ser una imagen o símbolo de algo[5]. Cuando se habla de representar la variabilidad, se habla de proveer un símbolo que dé cuenta de ésta en un sistema de software, algo que dé cuenta de su presencia en un producto específico o en una familia de productos de software. Pero ¿Cómo lograr representar algo que se puede materializar de tantas formas desde una sentencia de control de flujo hasta una selección en tiempo de ejecución de una variante específica? Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) ofrece un marco de referencia para aprovechar las diferentes fases de las variantes (introducción y enlace) y en esos términos ofrecer una representación numérica de la variabilidad de un producto o familia de productos de software. En esta sección se expondrá el marco de referencia de Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) para la representación cuantitativa de la variabilidad de forma independiente al sistema en estudio.

2.5.1. El marco de referencia de Bosch

Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) desarrolló un marco de referencia para la representación cuantitativa de la variabilidad de un producto o familia de productos de software. Este marco de referencia es independiente de las características tecnológicas del sistema en evaluación, tales como la plataforma de desarrollo, el lenguaje de programación o los patrones de arquitectura y diseño utilizados. Así mismo, el marco de referencia se puede aplicar en sistemas de forma independiente a sus características funcionales. Lo anterior quiere decir que el marco de referencia puede ser aplicado para representar de forma cuantitativa la variabilidad en cualquier producto de software sin depender de sus características técnicas y funcionales; lo único que se debe tener definido de forma previa a la aplicación este marco son las características (tener en cuenta que característica en este contexto son los *features* del producto) y su clasificación; esta clasificación corresponde a la determinación de sus fases de introducción y de enlace. A partir de las fases de introducción y de enlace de las diferentes características del sistema (ver secciones *Fase de introducción de una* y *Fase de enlace de una*) se determina un número entre 0.5 y 24.5 que representará una variabilidad relativa del sistema. Estos valores corresponden a la variabilidad relativa mínima y máxima las cuales corresponden a un 2% y 100% respectivamente como cotas inferior y superior de los niveles de variabilidad de un producto de software de acuerdo con Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002). El proceso de obtención de dicho valor es presentado a continuación.

[5] Tomado del diccionario de la real academia de la lengua española.

Bosch define un espacio de variabilidad de dos dimensiones. Una dimensión es la fase de introducción de una característica y otra dimensión es la fase de enlace de la misma característica. De esta forma, el espacio de variabilidad queda como se describe en la Ilustración 7.

Las diferentes fases de introducción y de enlace se surgen de los diferentes procesos de transformación de un producto de software. Estas fases de transformación se presentan en la Ilustración 5.

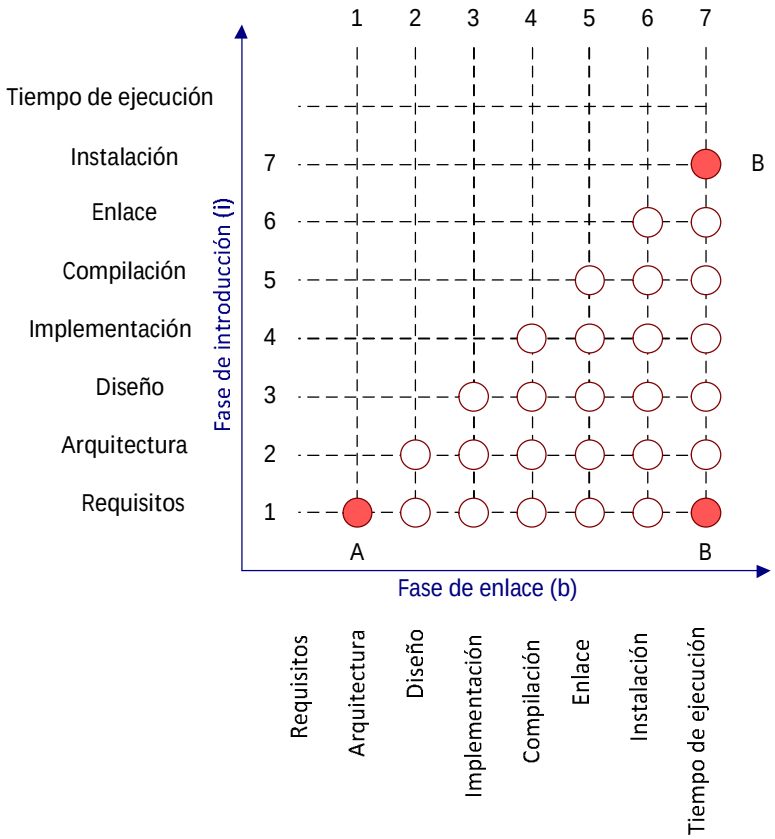


Ilustración 7: Espacio de variabilidad. Realizado a partir del trabajo de Bosch & Jaring **(Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002)**.

En este espacio de variabilidad se definen pares de variabilidad mediante un punto discreto en las coordenadas que representan la combinación de fase de introducción y de enlace específicas para una característica determinada. En la Ilustración 7 se muestran tres pares de ejemplo. De estos tres pares, el par A tiene menor variabilidad que el par C. De hecho, A es el par con menor variabilidad posible en todo el espacio de variabilidad. El par C por su parte tiene la mayor variabilidad posible en el espacio de variabilidad definido.

La magnitud de la variabilidad de un punto discreto en el espacio de variabilidad está determinada por el área que cubre el paralelogramo de base *i* y altura *b*. A este paralelogramo se le debe restar el área del triángulo formado por los puntos que no son posibles en el espacio de variabilidad. La Ilustración 8 ejemplifica como para un punto D ubicado de forma imaginaria en la fase de introducción 4 (Implementación) y la fase de enlace 7 (Tiempo de ejecución), el área que representa su magnitud es el área del cuadrado de área *i * b* menos el área del triángulo el cual corresponde a los pares [i,b] que no pueden existir en el espacio de variabilidad. Así las cosas, la magnitud de la variabilidad de un par [i,b] está dado por la Ecuación 1.

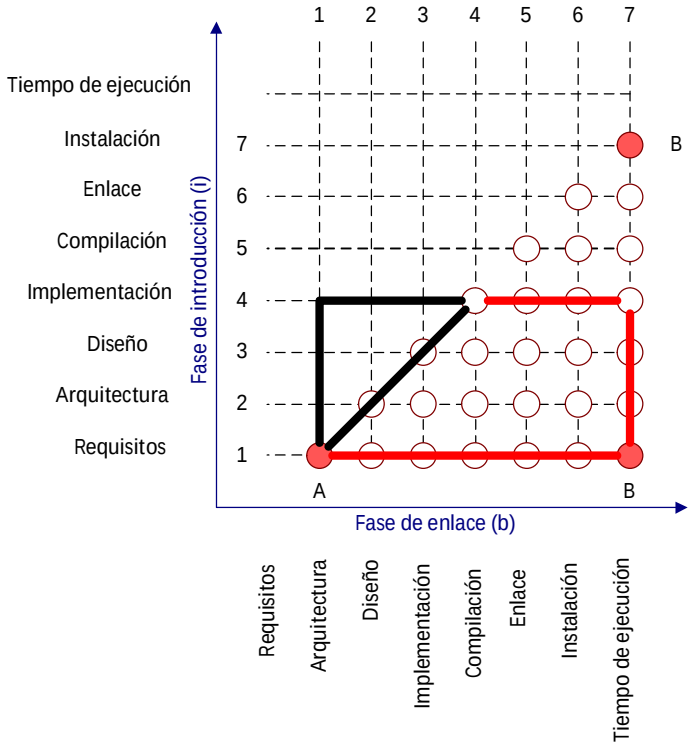


Ilustración 8: Magnitud de una coordenada específica.

$$A(i,b) = i * b - \frac{i^2}{2}$$

Ecuación 1: Magnitud de variabilidad de una característica con fase de introducción *i* y fase de enlace *b.* Tomado del trabajo de Bosch & Jaring **(Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002)**.

El valor mínimo de variabilidad en el espacio de variabilidad de Bosch es 0.5. Este valor se obtiene al calcular el valor de la Ecuación 1 con los menores valores posibles de i y b, el cual es 1 para ambas variables. El valor máximo de variabilidad en este espacio de variabilidad de Bosch en de 24.5. Este valor se obtiene de calcular el valor de la Ecuación 1 con los máximos valores posibles para i y b, el cual es 7 para ambas fases.

A los valores de variabilidad calculados mediante la Ecuación 1 se les denomina absolutos. A partir de ellos es posible obtener valores normalizados, los cuales corresponden a valores porcentuales de variabilidad. Siendo 24.5 el valor absoluto máximo posible, éste es el 100% de la variabilidad normalizada posible. El valor mínimo de variabilidad normalizada es del 2% el cual corresponde a la proporción que existe entre 0.5 y 24.5. La Ilustración 9 muestra cómo los valores de la variabilidad absoluta se incrementan en la medida en que la fase de introducción y la fase de enlace son mayores. Esta gráfica expone cómo el marco de referencia de Bosch determina que una característica es más variante en la medida en que su introducción y enlace en el producto se realiza en las etapas cercanas a su ejecución.

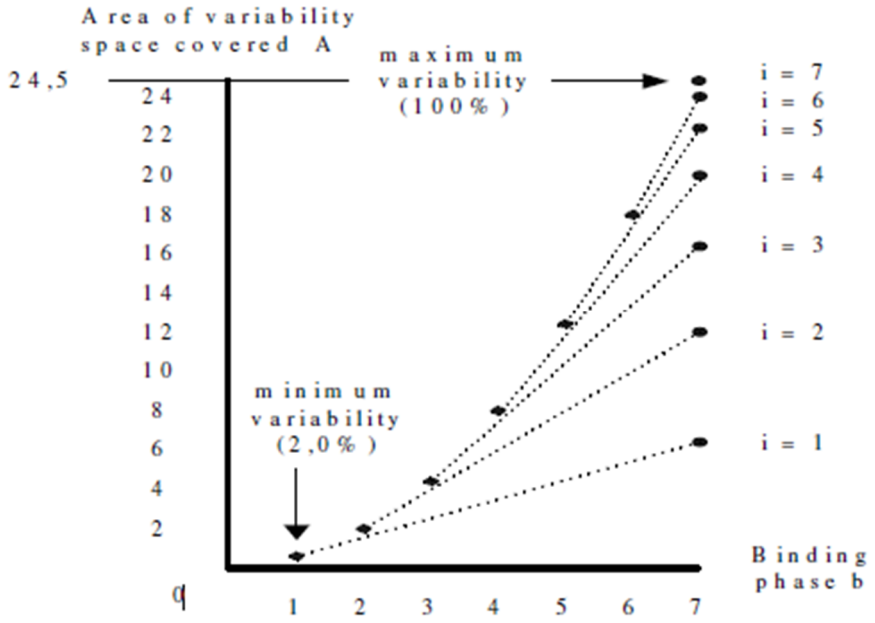


Ilustración 9: Acumulación de espacio de variabilidad cubierto por cada fase de introducción *i*. Tomado del trabajo de Bosch & Jaring **(Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002)**.

En resumen, lo que se debe realizar de acuerdo con Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) para determinar una representación cuantitativa de la variabilidad de un producto de software es lo siguiente:

- Identificar las características variantes del producto de software.
- Identificar sus fases de introducción y enlace.
- Obtener su representación cuantitativa de variabilidad.
- Normalizar esta representación mediante la obtención de un valor porcentual.

Una vez se obtienen estos resultados se puede proceder con un análisis de las características del sistema que dan cuenta del porcentaje de variabilidad obtenido como resultado.

2.5.2. Modelado de variabilidad

Un modelo es definido como un arquetipo o punto de referencia para ser imitado o reproducido[6]. El objetivo de un modelo es facilitar la comprensión y el estudio de algo complejo, aplicando el principio de abstracción para destacar los elementos importantes. En las referencias examinadas se encuentran diversos mecanismos de modelado de la variabilidad. Estos mecanismos permiten exponer la variabilidad de un producto de software –estrictamente de una LPS- mediante elementos gráficos que habilitan la visibilidad de las características variables y sus posibles valores o variantes así como restricciones de existencia de dichas características en un producto de software. Los principales mecanismos encontrados en la literatura para el modelado de variabilidad son FODA (Kang, y otros 1990) y OVM (Lauenroth y Pohl 2005) aunque también se encuentran múltiples opciones adicionales con diversas características particulares de modelamiento de variabilidad (Elfaki, Phon-Amnuaisuk y Kuan Ho 2009) (Robak y Pieczynski 2003) (Dhungana 2006) (Bu-Qing, Bing y Qi-Ming 2009) (Ye y Liu 2005) (Sarinho y Apolinario 2010)..

[6] Tomado del diccionario de la real academia de la lengua española.

El foco de este trabajo no es el modelado formal de variabilidad, sin embargo es importante resaltar la posibilidad que se genera de llegar hacia la implementación de un producto de una familia a partir de un modelo de características como un diagrama FODA. En la gran mayoría de referencias inspeccionadas se encontraron ejemplos básicos los cuales modelan una cantidad de características que son mínimas comparadas con el número real de características con el que puede contar un sistema o familia de sistemas real. Sin embargo, el trabajo de Antkiewicz y Czarnecki (Czarnecki y Antkiewicz 2004) expone una herramienta de modelado de características en diagramas FODA para mostrar la variabilidad de sistemas reales con cientos y hasta miles de características. El trabajo de Antkiewicz y Czarnecki es de interés especial debido a que adicionalmente al modelado se enfocan en la generación de piezas de software ejecutable a partir de dichos modelos.

Un análisis comparativo de los principales lenguajes de modelado de variabilidad se puede encontrar en el trabajo de Sinnema y Deelstra (Sinnema y Deelstra 2007).

3. Definición del caso de estudio

3.1. Visión general de la compañía

3.1.1. Razón social y filiales

Suramericana S.A. es una empresa subsidiaria del Grupo de Inversiones Suramericana. En ella se agrupan inversiones en seguros y seguridad social. Su objetivo básico es satisfacer integralmente las necesidades de protección y ahorro de la comunidad, entregando las mejores soluciones para sus clientes mediante servicios diferenciados, innovación, personalización de productos y servicios, tecnología de vanguardia y con el compromiso de colaboradores altamente calificados.

Suramericana S.A. es una compañía que se compone de 12 filiales en 4 países de Latinoamérica. La Ilustración 10 muestra el número de filiales por país y por tipo de negocio de Suramericana S.A. De esta ilustración se puede observar que tanto el negocio de seguros como el de seguridad social, de acuerdo al número de filiales, son de igual importancia y representación en la compañía. El negocio de seguros, sin embargo, es aquél con el que se tiene presencia en demás países de Latinoamérica.

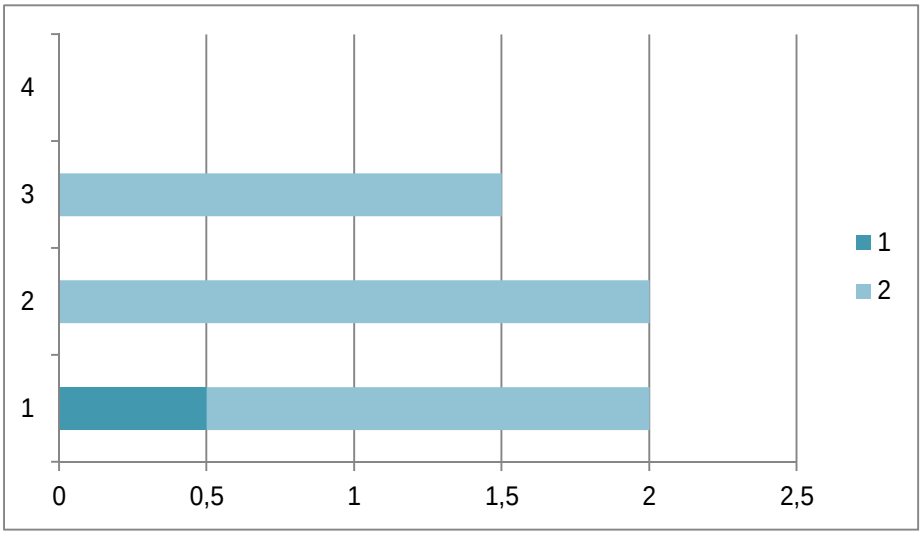


Ilustración 10: Filiales de Suramericana S.A. por país y por negocio.

Las razones sociales de seguridad social y las de seguros fueron tres empresas independientes hasta 2007. Mediante la sinergia de estas compañías nace lo que hoy se conoce como Suramericana S.A. En este trabajo nos enfocaremos en aspectos y necesidades específicas de un proyecto en la razón social de seguros, por lo que en este documento se hablará de Suramericana S.A. como la compañía de seguros, aislada de las de seguridad social.

3.1.2. Estructura organizacional

En la Ilustración 11 se muestra la estructura de la compañía de seguros y los departamentos de tecnología que atienden todas las necesidades de software y hardware de la organización. La vicepresidencia de seguros cuenta con cuatro grandes gerencias orientadas a dirigir los focos principales de la compañía de seguros. Estos focos son a) soluciones orientadas a individuos y familias (vida, salud, exequias), b) soluciones orientadas a empresas (cumplimiento, maquinaria, arrendamiento), c) soluciones orientadas al aseguramiento de vehículos y d) soluciones orientadas a grandes grupos de personas como el personal de una compañía. Estas gerencias son las encargadas de definir los diferentes productos de seguros de la compañía y velar por su sostenibilidad como productos de seguros. Son estas unidades de negocio las principales áreas usuarias y patrocinadoras de los diferentes proyectos de software que se ejecutan en la compañía. Por su parte, las unidades de tecnología se encuentran bajo la vicepresidencia administrativa y hacen parte de lo que en Suramericana S.A. se denomina Unidad de Servicios Compartidos. Esta unidad fue conformada con el fin de reunir en la vicepresidencia administrativa todas aquellos servicios que son de necesidad común a todas las diferentes compañías de la organización (incluyendo las de seguridad social). Esta unidad de servicios compartidos tiene una fuerte orientación al servicio y se caracteriza por ofrecer soluciones mejores que las que ofrece el mercado tanto en costo como en calidad. Así por ejemplo, la Gerencia de Desarrollo de TI es la encargada de ofrecer soluciones de software e inteligencia de negocios a toda la organización y velar por que estas soluciones sean de la mejor calidad en la medida que satisfacen y sobrepasan las necesidades de las diferentes unidades de negocio y apoyen su direccionamiento estratégico.

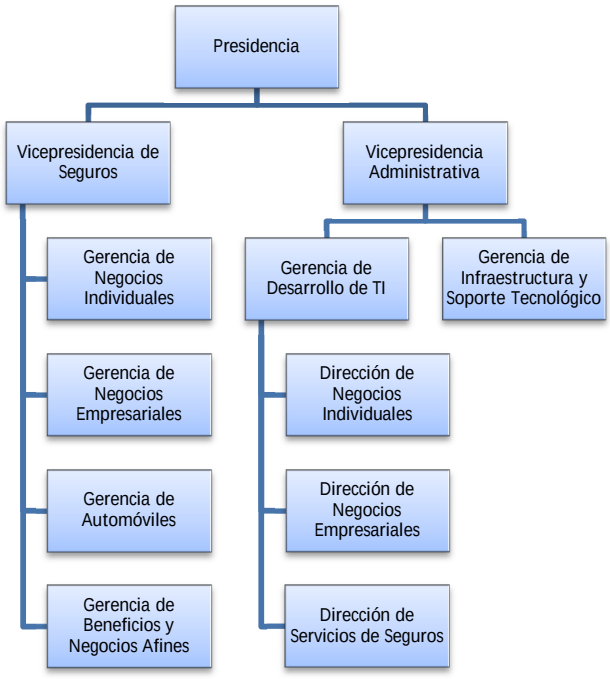


Ilustración 11: Estructura de la compañía de seguros y de las unidades de tecnología

En la Ilustración 11 se muestran, bajo la Gerencia de Desarrollo de TI, tres diferentes direcciones. Estos tres equipos (de un total de 10 equipos en esta gerencia) son los encargados de ejecutar los proyectos de software que satisfacen las necesidades de las cuatro gerencias de negocios de la vicepresidencia de seguros. Detalles adicionales sobre la estructura y la Gerencia de Desarrollo serán expuestos en la siguiente sección.

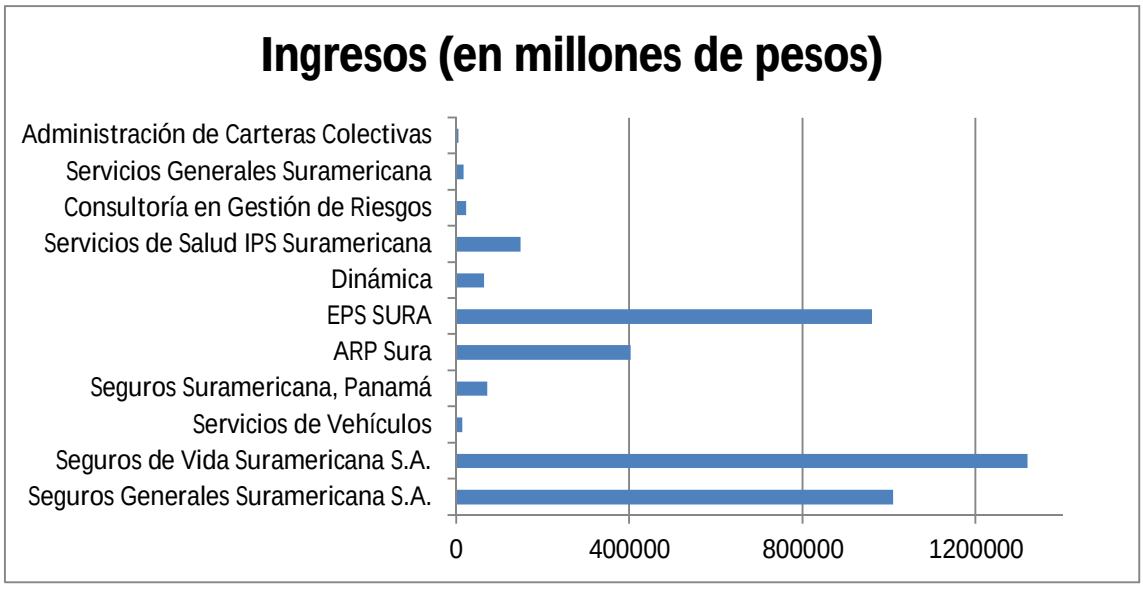
Es importante aclarar que la estructura organizacional de Suramericana S.A. es más amplia, pero para efectos de este trabajo, solo se documentan las estructuras estrictamente necesarias para dar contexto al problema y al estudio a realizar.

3.1.3. Cifras importantes

Para brindar una mirada cuantitativa de Suramericana S.A. (en su negocio de seguros) se ofrecerán algunas cifras de tal forma que el lector se forme una idea acertada de la dimensión de la compañía para la cual se realizará el estudio de variabilidad en uno de sus productos de software. Es importante aclarar que a la fecha de elaboración de este trabajo se cuentan con reportes cuantitativos de Suramericana S.A. a Diciembre de 2010; dado que las adquisiciones en Centro América se concretaron en el año 2011, aún no están disponibles cifras como su número de empleados y cifras financieras.

Suramericana S.A. es una compañía de 10967 empleados a nivel nacional y 1027 empleados en la filial de Panamá. Las compañías de seguros cuentan con 2 millones de clientes[7] Su operación se lleva a cabo mayormente en Colombia, pero como lo muestra la Ilustración 10, cuenta con operaciones en 4 países de Latinoamérica. Suramericana S.A. agrupa sus negocios en cuatro categorías principales: seguros individuales, seguros de automóviles, seguros generales y rentas vitalicias. Cada una de estas líneas cuenta en su interior con diversos productos los cuales se materializan para los clientes en pólizas de seguro.

Para brindar una dimensión global del alcance financiero de Suramericana S.A. y sus filiales de seguros, la Ilustración 12 expone los ingresos por filial para el año contable 2010 [8].



[7] Número de clientes de las compañías de seguros vigente a Abril de 2012. No incluye el número de asegurados el cual asciende a 4 millones. Se considera cliente aquel que paga por una póliza.
[8] Tomado de www.suramericana.com

Ilustración 12: Ingresos por filial de Suramericana S.A. del año contable 2010.

En esta ilustración se puede observar cómo las filiales de seguros son las compañías que generan mayores ingresos para Suramericana S.A. Sin embargo, en el sector de seguros hay que tener en cuenta aspectos como la siniestralidad, el resultado técnico y el resultado financiero, así como los gastos administrativos y demás costos, los cuales, luego de ser deducidos de los ingresos, permiten obtener el resultado neto de cada compañía. La Ilustración 13 muestra el resultado neto de cada filial para el año contable 2010.

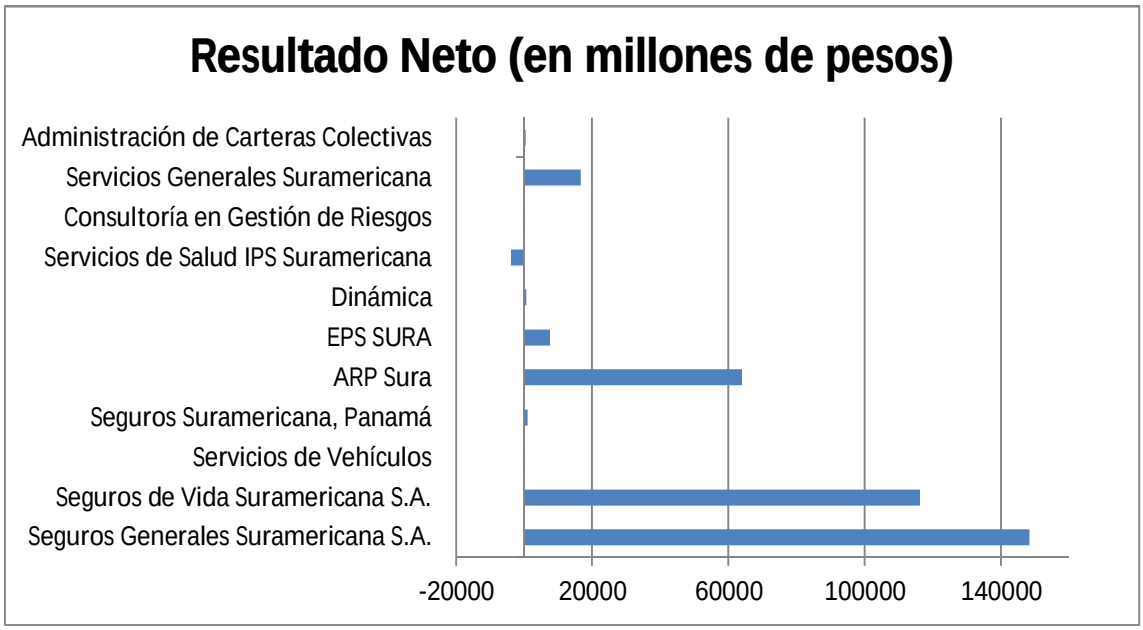


Ilustración 13: Resultado neto por filial de Suramericana S.A. del año contable 2010.

En la compañía de Seguros de Vida la siniestralidad y demás aspectos financieros y técnicos del seguro hacen que el resultado neto sea menor que el de la filial de Seguros Generales. Seguros Generales Suramericana S.A. desarrolla productos de seguros en los cuales aquello que es objeto de ser asegurado no es una persona sino un bien material; en esta compañía se desarrollan los diferentes productos de seguros de Automóviles (entre otros), dominio para el cual se desarrolló el producto de software sujeto de ser estudiado en este trabajo con respecto a su variabilidad.

3.2. Gerencia de Desarrollo de TI

En esta sección del documento se pretende brindar información de la Gerencia de Desarrollo de TI de Suramericana S.A. con el fin de brindar un contexto general de la unidad organizacional en la que se desarrollan aplicaciones en la compañía. Se brindará información acerca de la misión y objetivos de la gerencia así como el portafolio de servicios de la misma.

3.2.1. Descripción, misión y objetivos

La Gerencia de Desarrollo de TI es una gerencia que depende de la Vicepresidencia Administrativa de Suramericana. Esta gerencia hace parte de la unidad de servicios compartidos de la organización en conjunto con otras gerencias que prestan servicios transversales a todas las organizaciones de Suramericana S.A.

La Gerencia de Desarrollo de TI tiene la misión de planear, impulsar y coordinar el proceso de desarrollo de sistemas de información de acuerdo con el plan estratégico, los requerimientos de las diferentes áreas usuarias y el presupuesto aprobado para soportar los procesos del negocio, tanto operativos como de toma de decisiones, buscando fortalecer el negocio a través de la tecnología y el suministro de información.

El objetivo general de la Gerencia de Desarrollo de TI es generar e implementar soluciones integrales de software e información que faciliten a Suramericana S.A. realizar eficientemente la gestión de los negocios y la innovación.

3.2.2. Portafolio de servicios

A disposición de las diferentes unidades de negocio, la Gerencia de Desarrollo de TI ofrece el catálogo de servicios que se especifica a continuación:

Nombre del servicio	Descripción del servicio
Desarrollo de Aplicaciones	Desarrollo de nuevas aplicaciones para soportar procesos de negocio.
Atención de requerimientos sobre aplicaciones	Modificaciones y adiciones a la funcionalidad de una aplicación ya existente.
Atención de fallas de aplicaciones	Corrección de fallas o mal funcionamiento de una aplicación existente.
Asesoría y suministro de información	Brindar asesoría en el manejo de la herramienta de inteligencia de negocios. Creación y modificación de reportes adicionales a los sistemas analíticos existentes. Entrega de información solicitada por el negocio para análisis particulares o requerimientos legales. Desarrollo de nuevos sistemas de información analíticos para soportar la toma de decisiones del negocio. Atención y explicación de las cifras de negocio presentadas en los sistemas de información analíticos.

Tabla 3: Portafolio de servicios de la Gerencia de Desarrollo de TI.

3.2.3. Estructura interna

La Ilustración 11 muestra como la Gerencia de Desarrollo de TI tiene en su estructura un conjunto de direcciones. En total son diez direcciones encargadas de atender todas las necesidades de software e información de Suramericana S.A. En esa ilustración solo se muestran las direcciones encargadas de atender los proyectos de software de las unidades de negocios de seguros. Las demás direcciones se omiten de la ilustración debido a que no son de interés para la elaboración de este trabajo.

Las direcciones de desarrollo están a cargo de su respectivo director. Este director tiene a su cargo un equipo de personas cuyo cargo es Analista de Desarrollo y cuyas funciones van desde la atención a los usuarios de las unidades de negocio, la comprensión de sus necesidades y educción de requisitos, pasando por la definición de la arquitectura de las aplicaciones, su diseño detallado, implementación, pruebas e implantación. En muchos casos también son los encargados de coordinar estas actividades con proveedores externos tanto de análisis, diseño y construcción como de pruebas funcionales y no funcionales. Los directores de desarrollo son los encargados de coordinar el trabajo y las prioridades de los analistas a su cargo, procurando orientar la ejecución de los proyectos hacia la consecución de los objetivos estratégicos de la compañía.

Finalmente, se anota que el perfil de los analistas de desarrollo no se limita a personas con excelentes capacidades técnicas sino que adicionalmente deben ser personas orientadas al servicio, capaces de escuchar a los usuarios y proponer las mejores soluciones de tecnología de acuerdo a los recursos (tiempo y presupuesto) disponibles.

3.2.4. Cifras importantes

Para finalizar la descripción de la Gerencia de Desarrollo de T.I. de Suramericana se brindarán algunos datos cuantitativos que permitan al lector obtener una idea de la magnitud de la Gerencia de Desarrollo y su proporción con respecto a las demás unidades de Suramericana S.A.

La Gerencia de Desarrollo de T.I. es una gerencia de 120 empleados sin tener en cuenta los proveedores que prestan servicios para el desarrollo de aplicaciones en modalidad de *outsourcing*. Contando el personal en outsourcing, el número de personas que trabajan en esta gerencia asciende a 260 empleados.

Estos 260 empleados se encuentran agrupados en 10 direcciones que atienden las diferentes unidades de negocio tanto de seguridad social como de seguros. Como ya se ha mencionado, en la Gerencia de Desarrollo son tres las direcciones que trabajan atendiendo las necesidades y proponiendo soluciones de software para los equipos de negocio de seguros.

Unidad de negocio	**# de empleados en el negocio**	**Dirección en la gerencia de Desarrollo**	**# de empleados en la Gerencia de Desarrollo**	**Proporción**
Gerencia de Automóviles	400	Servicios de Seguros	16	4%

Gerencia de Negocios Individuales	80	Negocios Individuales	17	21.25%
Gerencia de Negocios Empresariales	50	Negocios Empresariales	16	32%
Gerencia de Beneficios y Negocios Afines	66	Negocios Empresariales y Servicios de Seguros	7	10.6%
Total	596		56	9.4%

Tabla 4: Relación de empleados de las unidades de negocio con respecto los equipos de trabajo en la Gerencia de Desarrollo.

Las cifras de las Tabla 4 dan cuenta de la proporción que representa los equipos de desarrollo sobre unidades de negocio de seguros de Suramericana S.A. El número de empleados de las unidades de negocio no tiene en cuenta a la fuerza de ventas pues ésta no está dividida por unidades de negocio y adicionalmente no generan necesidades y requisitos para los equipos de desarrollo. Todas sus necesidades son canalizadas a través de una dirección al interior de la Gerencia de Desarrollo llamada Dirección de Servicios Comerciales.

La ausencia de datos de otras compañías que permitan realizar una comparación inhabilita la asignación de una calificación de la dimensión de la Gerencia de Desarrollo en Suramericana S.A. Un juicio basado en la percepción permitiría la calificación de la dimensión como normal, en tanto que no es tan grande ni tan pequeña como para estar subdimensionada o sobredimensionada.

3.3. Descripción del sistema en evaluación

Ya se ha expuesto el marco general de la compañía Suramericana S.A. y cómo las unidades de negocio de seguros se apoyan en la Gerencia de Desarrollo de TI para la construcción de productos de software que apoyen su operación. En esta sección del documento se brindará información acerca del producto de software sobre el que se realizará la medición de la variabilidad. Se expondrán detalles desde la necesidad del negocio hasta la propuesta arquitectónica y las tecnologías utilizadas para su implementación.

3.3.1. Necesidad del negocio

La Ilustración 11 expone la existencia de la Gerencia de Automóviles como una unidad de negocio patrocinadora de los diferentes proyectos de software implementados por la Gerencia de Desarrollo de TI. Esta gerencia, hacia finales de 2009, planteó la idea inicial de construir una plataforma de venta de seguros sobre Internet. Este sitio, básicamente, capturaría los datos de un vehículo (modelo, marca, cilindraje, entre otros), calcularía un valor de prima, solicitaría el pago por algún mecanismo de pago electrónico al cliente y procedería con la expedición de la póliza de vehículo y finalmente la entrega vía email al cliente de los documentos relacionados con la compra. En ese momento nació la idea de negocio de hacer un sitio transaccional de Venta en Línea para clientes.

Una vez nace esta idea se plantean las siguientes inquietudes:

- ¿Cómo se mercadea un sitio de venta de seguros en Internet?
- ¿Algún competidor ya está vendiendo pólizas de vehículos en Internet?
- ¿Cómo está el mercado latinoamericano y mundial con respecto a la venta de seguros en Internet?
- ¿Qué es necesario técnicamente para proveer una solución de software que apoye el proceso de venta de seguros de vehículos sobre Internet?

Las respuestas se dieron después de una gran cantidad de validaciones entre diferentes equipos, entre ellos, los siguientes:

- Gerencia de Automóviles.
- Gerencia de Beneficios y Grupos Afines.
- Gerencia de Mercadeo.
- Gerencia de Desarrollo de TI.
- Gerencia de Infraestructura y Soporte Tecnológico.
- Gerencia de Asuntos Legales.
- Gerencia de Gestión Humana.
- Consultores externos expertos en venta de seguros en Internet a nivel mundial.
- Proveedores externos de usabilidad de aplicaciones de software.
- Proveedores externos de diseño gráfico web.

Luego de todas las validaciones necesarias, se encontró que la mejor estrategia para desarrollar el sistema consistía en hacer una primera versión de la aplicación orientada a satisfacer una necesidad de la Gerencia de Beneficios y Grupos Afines la cual consistía en ofrecer a los empleados de Suramericana una herramienta de software para que ellos mismos realicen la inclusión de sus vehículos en las pólizas colectivas de automóviles que Suramericana S.A. ofrece para sus empleados. Esta decisión se tomó debido a que las sugerencias de los consultores siempre estuvieron orientadas a no salir en falso directamente a Internet debido a que un error en ese entorno afectaría la imagen de Suramericana S.A. de forma considerable y disminuiría la probabilidad de salir al mercado con una nueva línea de negocio confiable e innovadora .

Esta primera fase del proyecto, orientada a los empleados internos como cliente del producto, permitió afinar detalles tanto de usabilidad como de cualidades sistémicas con el objetivo de salir al mercado abierto con la menor cantidad de defectos posibles. Una vez finalizada esta fase, se complementaría el producto de software construido con las especificaciones de negocio propias para que la funcionalidad pueda satisfacer las necesidades de un navegante que desee comprar una póliza de automóviles en Internet con Suramericana S.A.

Es muy importante hacer claridad que el proyecto, que se denomina *Venta en Línea*, no solo es un proyecto de software sino que constituye un proyecto estratégico que involucra los siguientes retos:

- Diseño del proceso de negocio que tiene por objetivo vender pólizas de vehículos por un canal diferente al tradicional, es decir, sin la intervención de un intermediario.

- Diseño de estrategias de mercadeo para ofrecer de una forma clara la idea de compra de un seguro en un canal inexplorado como es Internet en el mercado de seguros a nivel regional y nacional.

- Diseño e implementación de una solución de software no tradicional en la medida en que debe estar orientada a clientes finales y no a usuarios internos conocedores de productos de seguros los cuales son generalmente los usuarios de las aplicaciones de software que son desarrollados por las direcciones de seguros de la Gerencia de Desarrollo de TI.

Todo este trabajo se centra en Venta en Línea como un proyecto de desarrollo de software y no abarca el estudio de ningún otro aspecto diferente al producto de software analizado, diseñado e implementado para satisfacer la necesidad de negocio expuesta en esta sección.

A manera de conclusión de detallan entonces los siguientes aspectos importantes del producto de software Venta en Línea Autos:

- Se realiza en dos fases. La primera fase es para clientes internos (empleados) y la segunda fase es para clientes externos (navegantes en Internet).

- El producto se construyó en un proyecto llamado Venta en Línea, el cual, a la fecha de elaboración de este trabajo se encuentra en los procesos de certificación funcional y técnica.

- Independiente de los resultados obtenidos con respecto a la variabilidad del producto de software en estudio, éste (el producto) no se modificará en caso de que las conclusiones así lo sugieran. Esto quiere decir que con este trabajo se realizará una labor de observación y posterior caracterización de la variabilidad, sin que los resultados de esta caracterización desemboquen en la modificación del sistema para mejorar los niveles de variabilidad, si ese fuera el caso.

- Los resultados del estudio de variabilidad sobre el producto de software Venta en Línea ayudarán a determinar la potencialidad que tienen los diversos sistemas software de Suramericana S.A., en el dominio de seguros, de ser producidos con una LPS. Las características comunes y variantes de estos sistemas conocidas hasta el momento permiten intuir que dicho estudio se podría realizar.

3.3.2. Arquitectura de la solución

En esta sección se expondrán los aspectos de arquitectura relevantes para la construcción del producto de software que soporta al proyecto Venta en Línea en Suramericana S.A. Se expondrán las cualidades

sistémicas que habilitan el diseño arquitectónico realizado para el producto y se expone así mismo el estilo arquitectónico seleccionado y detalles acerca de ésta.

El propósito de esta sección es brindar una visión de alto nivel de la arquitectura del sistema del producto de software sobre el que se realizará, en secciones posteriores, la medición de la variabilidad. De esta forma se podrá saber de qué tipo de sistema se está ejecutando la medición de variabilidad.

El producto de software sobre el que se realizará la representación de variabilidad se desarrolló utilizando el modelo 4+1 vistas definido por Krutchen (Kruchten 1995) y aplicando un enfoque de desarrollo basado en la arquitectura (Bass y Kazman 1999). En este enfoque se otorga igual prioridad a los requisitos arquitectónicos y a los requisitos funcionales. Teniendo en cuenta los requisitos arquitectónicos, se obtienen los drivers de arquitectura, los cuales servirán de guía para el diseño y posterior realización y mantenimiento de la arquitectura del sistema. En las secciones posteriores se especificarán los drivers de arquitectura que guían el diseño del producto y posteriormente se provee una descripción del estilo arquitectónico de la solución.

En vista de que el sistema se desarrolló bajo un enfoque de desarrollo orientado por la arquitectura y sobre dicho sistema se realizará el estudio de la variabilidad es posible plantear desde este punto del documento la siguiente pregunta:

¿Las características arquitectónicas del producto de software implementado y/o sus mecanismos de implementación favorecen la variabilidad del producto software?

3.3.3. Drivers de arquitectura

Tomando como catálogo las características de calidad de software especificadas por la norma ISO 25000 (International Organization for Standarization 2005), se seleccionaron como conductores de arquitectura aquellos que determinaron las decisiones de arquitectura del producto de software, los especificados en la Tabla 5.

Característica	Justificación
Usabilidad	Dado que la solución busca ofrecer la posibilidad de comprar un seguro en Internet sin la participación de un intermediario, el producto de software debe ser usable en la medida en que debe ser claro para un usuario final qué producto está comprando, qué características tiene el seguro que desea comprar y qué opciones de funcionalidad tiene este usuario final de cara al producto de software. Debe ser un producto orientado a usuarios no conocedores de productos de seguros que puedan comprender de forma ágil el producto y los pasos de compra.
Eficiencia	Todas las operaciones, tanto funcionales como de presentación deben ejecutarse de forma eficiente. La eficiencia hace referencia al consumo adecuado de recursos para efectuar una operación específica. En este caso, el recurso más crítico de cara al usuario final es el tiempo de respuesta del producto de software. Es claro que el producto de software en análisis debe tener tiempos adecuados de respuesta en las operaciones tanto

	funcionales (cálculos, consultas, transacciones etc) como de presentación (interfaz gráfica). En la documentación detallada del sistema se establecen tiempos de respuesta específicos máximos para cada transacción de tal forma que esta característica se pueda garantizar.
Funcionalidad	Que el sistema haga lo que debe hacer es fundamental en cualquier producto de software. Esta es una necesidad obvia en todo sistema. Sin embargo, pocos diseñan sus arquitecturas para buscar satisfacer esta característica. El producto de software del proyecto Venta en Línea mediante la ejecución de una disciplina de desarrollo de software basado en componentes establece lineamientos desde su arquitectura para que se pueda cumplir esta característica.
Mantenibilidad	La experiencia del arquitecto y el equipo de diseñadores del producto de software del proyecto Venta en Línea hacen que preparar el software para que las modificaciones solicitadas y la corrección de errores se realicen de manera ágil y en puntos específicos del producto es clave para el éxito en el largo plazo para las aplicaciones en Suramericana S.A. Los usuarios se reconocen como variables en la medida en que constantemente están solicitando modificaciones sobre las aplicaciones y para satisfacer esta necesidad de cambio constante se debe preparar la aplicación para que soporte de forma adecuada esta característica.

Tabla 5: Características principales de calidad que orientan la definición de arquitectura y diseño del producto de software del proyecto Venta en Línea.

3.3.4. Estilo arquitectónico del producto

El catálogo POSA (Buschman, y otros 1996) especifica diferentes patrones de arquitectura de productos de software. En Suramericana S.A. la gran cantidad de sistemas de software obedecen al patrón *Layers* (por capas). Esto se debe a que los propósitos de los sistemas empresariales de Suramericana S.A. se satisfacen con sistemas organizados de esa manera, por capas. En la etapa de elaboración de arquitectura del software del proyecto Venta en Línea, se realizó un análisis detallado de lo siguiente para tomar la decisión del estilo arquitectónico de la solución:

- Necesidades del negocio y su proceso de venta de seguros en Internet.
- Restricciones de tiempo para implementación del sistema de software.
- Disponibilidad de recursos de infraestructura (servidores, licencias etc) en Suramericana S.A.

El primer punto especificado fue el prioritario al momento de realizar la arquitectura del sistema. Los puntos siguientes a éste son mencionados debido a que fueron tenidos en cuenta pero con menor prioridad que el primero. La Ilustración 14 muestra las capas que guían la arquitectura del producto de software del proyecto Venta en Línea.

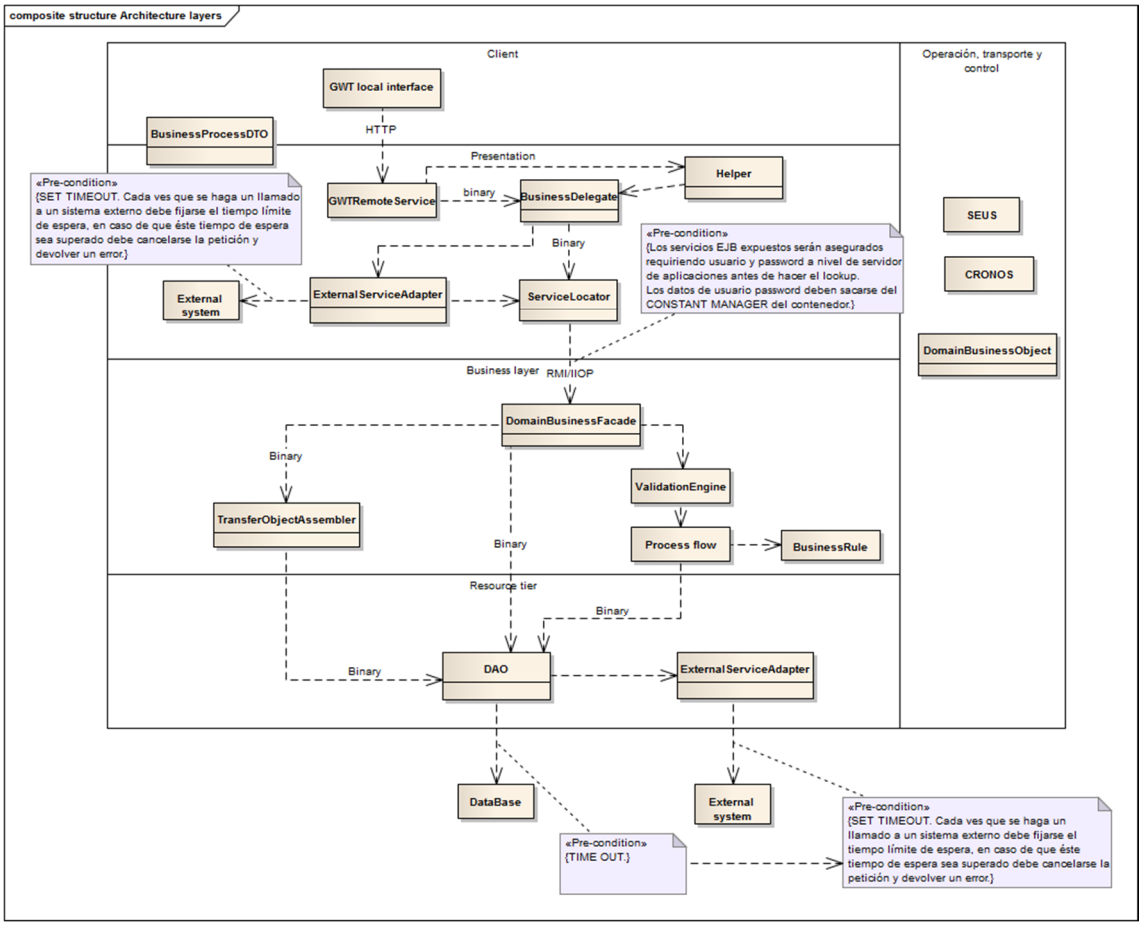


Ilustración 14: Capas de la aplicación Venta en Línea

En la siguiente sección del documento se especificará cómo los diferentes elementos de cada capa habilitan la satisfacción de las diferentes características que constituyen los drivers de arquitectura del sistema.

3.3.5. Mecanismos de implementación de los conductores de arquitectura

Por *mecanismo de implementación* se desea exponer aquellos elementos tangibles del producto de software que hacen que se puedan cumplir efectivamente las características que constituyen los drivers de arquitectura. En la Tabla 6 se expone la relación entre los diferentes elementos presentes en las diferentes capas de la arquitectura del sistema y los drivers de arquitectura del mismo.

Driver de arquitectura	Mecanismo relacionado	Observaciones
Usabilidad	Separación de la capa de presentación en unidad de despliegue independiente.	Separa como artefactos independientes la interfaz gráfica de usuario y la capa de negocio. El punto de contacto entre estas capas son operaciones de negocio claras y estrictas. De esta forma el equipo de desarrollo de la interfaz

		gráfica se concentra en implementación de aspectos de diseño y usabilidad establecidos por los respectivos equipos de definición de estos temas.
	Tecnología de presentación acorde a la necesidad.	La tecnología de implementación seleccionada (ver sección *Tecnologías seleccionadas*) habilita la construcción de sistemas más amigables al usuario y más usables.
Eficiencia	Peticiones entra la capa de presentación y la capa de negocio vía peticiones asíncronas (AJAX).	AJAX permite la comunicación de la interfaz gráfica con la capa de negocio gracias al uso de JavaScript y XML y no mediante cargas completas de páginas como se hace en un enfoque tradicional. Esto significa que el usuario final esperará menos tiempo para que una página se cargue y se pinte y una interacción más fácil con la página. De esta forma, el usuario puede continuar observando la interfaz gráfica mientras ésta se comunica de forma asíncrona con la capa de negocio (Heilmann 2006)
	Comunicación con la capa de negocio vía RMI-IIOP	RMI-IIOP permite la comunicación remota de forma nativa en aplicaciones J2EE. Se seleccionó este mecanismo de comunicación debido a que algunos componentes ya estaban desarrollados en Java y debido a que este protocolo garantiza un menor tiempo de respuesta comparado con SOAP-HTTP. La comunicación se realiza siempre mediante RMI-IIOP entre la capa de presentación y la capa de negocio. Sin embargo, si algún sistema externo expone sus servicios mediante SOAP-HTTP, el patrón Fachada en el componente *ExternalServiceAdapter* permite adaptar la invocación de Web Services externos y sus respuestas ser entregadas siempre a la capa de presentación mediante interfaces remotas EJB.
	Control de tiempo de ejecución de funcionalidades en sistemas externos.	Para poder garantizar que una operación de negocio se ejecuta en un tiempo específico o menor a éste, se implementó un mecanismo de *time-out* para cada invocación. Si un sistema externo o una operación de negocio no entregan una respuesta en este tiempo la comunicación con ese componente se cortará y se entregará un mensaje de error al usuario final.

		Esto se realiza de esta manera pues se consideró preferible dar una respuesta de error a un usuario final a dejarlo esperando por una respuesta de la cual no se tiene garantía de si llegará o no.
Funcionalidad	Aislamiento de las operaciones de negocio vía el patrón Fachada (Gamma, y otros 1995).	De cara a las necesidades del sistema, las operaciones de negocio deben ser claras y tener un propósito específico. Es por esto que la capa de presentación siempre interactuará con la capa de negocio mediante fachadas de negocio (**ExternalServiceAdapter**) que garantizan que operaciones complejas se llevan a cabo bajo un contrato claro y estricto. Estas fachadas hacen parte de la arquitectura de referencia del sistema y son de estricto cumplimiento en toda relación de la capa de presentación con la capa de negocio.
	Uso de JUnit como framework de pruebas unitarias.	Las operaciones de grano grueso especificadas como métodos en las clase fachadas de negocio deben ser tan estrictas y claras que sus resultados deben poder ser probados de forma unitaria. En este proyecto se hizo uso de JUnit para garantizar los resultados esperados de estas fachadas de negocio.
Mantenibilidad	Aislamiento de las operaciones de negocio vía los patrones Fachada y Adaptador (Gamma, y otros 1995).	Este mecanismo habilita la detección de errores y la capacidad de pruebas estrictas sobre una operación de negocio determinada. Si es posible saber qué hacer específicamente cada operación es entonces posible saber dónde se está presentando un error y también es posible garantizar su calidad funcional de manera previa a una liberación a un ambiente de aseguramiento de calidad y posterior paso a ambiente productivo.
	Uso de Drools JBPM para la orquestación de operaciones de negocio de grano grueso.	En el proyecto se utilizó este motor de procesos para implementar la secuencia de tareas de un proceso de negocio como un proceso de corta duración. De esta forma se hace más comprensible y legible para los encargados de realizar actividades de comprensión y mantenimiento del producto final. Estos flujos se pueden observar de forma gráfica en la notación BPMN.

	Uso de Drools Expert para la evaluación de reglas de negocio.	Las reglas de negocio escritas en un lenguaje dedicado exclusivamente a la escritura de reglas acerca la definición de las mismas a un lenguaje más natural en el cual se habilita la facilidad de comprensión, cambio y evolución de las mismas en escenarios cambiantes.

Tabla 6: Mecanismos de implementación de conductores de arquitectura

3.4. Tecnologías seleccionadas

En esta sección se brindan datos finales del **producto de software** sobre el que se realizará la medición de variabilidad mediante la especificación de las diferentes tecnologías utilizadas para la construcción del sistema. Con esto se pretende brindar datos concretos sobre cómo está implementada la aplicación. La Tabla 7 da cuenta de estas tecnologías en las diferentes capas de la aplicación.

Capa	Tecnología seleccionada	Versión
Presentación	Google Web Toolkit (GWT)	2.3
Negocio	J2EE	2
	Java	1.6
	Drools Expert	5.3.0
	Drools JBPM	5.3.0
	Hibernate	3.3.2
Integración	EJB	3.0
	Web Services	SOAP1.2
	JDBC	
Persistencia	Oracle	11g

Tabla 7: Tecnologías de implementación de la aplicación en estudio.

4. Identificación de variabilidad objetivo

El objetivo de esta sección es establecer el tipo de variabilidad que se representará cuantitativamente en el producto de software Venta en Línea Autos descrito en secciones anteriores. Adicionalmente, se establecerán los motivos por los cuales se selecciona un tipo específico de variabilidad para su representación. La necesidad de esta sección surge debido a que la variabilidad está presente en cualquier producto de software de múltiples maneras y con múltiples propósitos y es necesario acotar qué tipo específico de variabilidad se estudiará y cuál es la motivación de su representación cuantitativa para habilitar la obtención de conclusiones claras y el establecimiento de posibles trabajos futuros.

Lo primero que se realizará será especificar los motivos concretos que soportan la necesidad de representar la variabilidad en el producto de software específico Venta en Línea. Estos motivos ya han sido establecidos de forma general en la sección *Motivación* pero a continuación se especificará con más detalle por qué es necesario realizar una representación de variabilidad del producto de software seleccionado. Con los argumentos ya establecidos, se facilita la selección de una variabilidad específica a ser representada.

4.1. Motivos de estudio de la variabilidad

La motivación principal para estudiar de forma explícita la variabilidad en el producto Venta en Línea es la disminución del tiempo que le toma al equipo de la Gerencia de Desarrollo de TI encargado del producto de software para implementar una nuevo comportamiento en aquellos puntos donde, desde una etapa temprana (análisis) se estableció que habría variación en etapas futuras, es decir, con el producto ya en operación.

Para darle un poco más de contexto a esta necesidad, es apropiado indicar que actualmente los artefactos de implementación (componentes, librerías, código fuente y compilado) invocan finalmente procedimientos almacenados en la base de datos. Estos procedimientos almacenados han sido modificados durante los últimos doce (años) por múltiples equipos conformados por un gran número de personas, con diferentes niveles de habilidad para el análisis, diseño y construcción de software y sometidos a diferentes niveles de presión por parte de las unidades de negocio que varían en el rango de bajo hasta los más altos. Esto ha llevado a que los productos de software principales de Suramericana S.A. se aproximen a un anti-patrón denominado por Foote y Yoder (Foote, Harrison y Rohnert 2000) como *Big Ball Of Mud* o *Gran Bola de Lodo*.

Cuando un producto de software se aproxima a este anti-patrón o de hecho cumple con todas las características para ser clasificado de esta forma, se presentan las siguientes consecuencias principales:

- Se dificulta la comprensión del código debido a su ilegibilidad.
- No se diferencian los diferentes tipos de comportamientos existentes. Todos conviven en unas pocas unidades de software de forma interrelacionada y desorganizada.
- La implementación de un nuevo comportamiento o la modificación de alguno ya existente no garantiza que se mantenga el correcto funcionamiento del producto como una unidad holística.

Estas tres situaciones, de forma global, desembocan en la siguiente:

- La implementación de un nuevo comportamiento o la modificación de alguno ya existente toma un tiempo considerablemente alto.

Si se lograra construir un producto de software que fuera legible, con los diferentes comportamientos diferenciados de forma clara y que estos comportamientos se puedan verificar y certificar de forma aislada e integrada al producto final de forma ágil, entonces el tiempo de implementación de un nuevo comportamiento, o la modificación de uno ya existente se podría lograr en un tiempo menor. Por tal motivo se podría reducir el tiempo de liberación de esta necesidad de funcionalidad de forma más rápida; de esta forma, el usuario final contaría con el cambio en un menor tiempo en ambiente productivo.

La pregunta que podría surgir al lector luego de lo recién descrito es ¿Y qué beneficios se obtienen si se logra implementar un comportamiento nuevo o una modificación a uno ya existente en un menor tiempo? La respuesta es simple: si se logra llevar productos en un menor tiempo posible a un ambiente producto, la organización será más competitiva, debido a que podrá mejorar la oferta a sus clientes finales en tiempos cortos y podrá alejarse de sus competidores de forma rápida, ofreciendo mejores productos de forma anticipada.

Es importante entender la relación de proporcionalidad directa entre las características de los productos de software y la capacidad de competencia de la compañía. Si los productos de software apoyan directamente la operación de ventas y producción de la organización entonces su capacidad de variabilidad impactará directamente la capacidad de competir de la organización.

Ahora, habiendo comprendido lo anterior, es importante recordar que la variabilidad de un producto de software *es la capacidad que ofrece un sistema o artefacto de software para ser extendido, cambiado, personalizado o configurado para su uso en un contexto particular* (Bosch y Jaring, A taxonomy and hierarchy of variability dependencies in software product family engineering 2004) **.** Para construir un producto de software con estas características se deben conocer los puntos del producto en los que existirá variación y debido a los puntos previamente expuestos, estos puntos de variación son exclusivamente de negocio y no de aspectos técnicos.

El producto de software Venta en Línea, como su nombre bien lo describe, en un sistema que apoya un proceso de venta de seguros y tiene que ver de primera mano con la capacidad de Suramericana S.A. de competir con otras aseguradoras en la venta de seguros en Internet. Es por esto que es de interés conocer la capacidad de variabilidad de este producto de software. Esta capacidad de variabilidad es posible representarla cuantitativamente y a partir de esta representación cuantitativa determinar si el producto favorece o no la capacidad de competir de Suramericana S.A. en el mercado de venta de seguros de autos en Internet.

Se debe dejar claro que la variabilidad no fue un conductor de la arquitectura del sistema ni de los sistemas externos que se construyeron para el funcionamiento del sistema completo, mucho menos de los sistemas ya existentes y que se reusaron para la ejecución de funcionalidades necesitadas por el producto Venta en Línea. Esto quiere decir que la necesidad de variabilidad del sistema no existió explícitamente ni de forma oficial en las diferentes etapas del proceso de desarrollo del sistema. Esto, sin embargo, no significa que el sistema no tenga necesidades de variabilidad y que los mecanismos empleados para atender la mantenibilidad del sistema no atiendan a su vez la mantenibilidad del mismo. Los usuarios de las unidades de negocio realizaron las definiciones y a partir de éstas los arquitectos y diseñadores del producto realizaron una propuesta de arquitectura y diseño que posteriormente fue implementada. Estos artefactos son los que serán analizados a la luz del marco de referencia de Bosch para determinar si las decisiones de arquitectura, diseño y construcción aportan valor o restan valor a la variabilidad global del sistema.

4.2. Clasificación de la variabilidad en estudio

La selección de la variabilidad objetivo se realizará en tres pasos, descritos a continuación:

1. Selección del tipo de variabilidad de acuerdo a la clasificación expuesta en la sección *Dimensiones de la Variabilidad*.
2. Determinación de los subtipos de variabilidad posibles de ser representados de acuerdo a la variabilidad seleccionada en el punto 1.
3. Selección del subtipo de variabilidad de los identificados en el punto 2.

Los 3 pasos enumerados previamente se abarcarán brindando una clasificación textual de las diferentes clasificaciones de la variabilidad y una descripción final, también textual, de la variabilidad seleccionada de acuerdo a los motivos expuestos en la sección anterior.

La Tabla 8 muestra la clasificación de los tipos de variabilidad definidos previamente (ver sección *Dimensiones de la Variabilidad*) y adiciona una columna en la que se especifica una clasificación adicional para el producto de software Venta en Línea. La columna *Clasificación* especifica valores para la variabilidad externa en el contexto del sistema objeto de estudio, los cuales serán detallados más adelante en esta sección. La variabilidad interna no cuenta con clasificación adicional debido a que la definición ofrecida por Lauenroth & Pohl (Lauenroth y Pohl 2005) es considerada suficiente. La variabilidad temporal, como es determinado también por Lauenroth & Pohl (Lauenroth y Pohl 2005), se relaciona con actividades de mantenimiento de los productos de software; los diferentes tipos de mantenimiento se clasifican con completo detalle en el trabajo de Chapin (Chapin, y otros 2001).

Tipo de Variabilidad	**Subtipo de Variabilidad**	**Clasificación**	**Descripción resumida**
Espacial	Externa	Del proceso de venta soportado por el producto.	Variación en los pasos a seguir en el proceso de negocio de venta de seguros de automóviles.
		Del comportamiento de los componentes que ejecutan las tareas del producto Venta en Línea.	Variación en los comportamientos de las piezas de software que soportan un proceso particular y estático de venta de seguros de automóviles.
	Interna	No hay clasificación adicional. La definición de (Lauenroth y Pohl 2005) es suficiente.	Ver (Lauenroth y Pohl 2005).
Temporal	Ver clasificación de tipos de mantenimiento de software en (Chapin, y otros 2001)	Ver clasificación de tipos de mantenimiento de software en (Chapin, y otros 2001)	Ver (Chapin, y otros 2001).

Tabla 8: Tipos de variabilidad posibles de ser estudiados en el producto de software Venta en Línea.

Antes de documentar los detalles de las clasificaciones de la variabilidad Espacial e Interna, es propio hacer referencia a la guía que proveen Lauenroth & Pohl (Lauenroth y Pohl 2005) para determinar si una característica de un producto de software se clasifica como externa o interna. La primera guía para la clasificación consiste en determinar si la característica en estudio es de interés para el usuario/cliente del sistema; si es de interés para él que el producto tenga una característica o no la tenga, entonces esa característica debe ser clasificada como externa. Por otra parte, si la característica no es de interés para el usuario en absoluto pero esa misma característica presenta variación entonces debe ser clasificada como interna.

Lauenroth & Pohl (Lauenroth y Pohl 2005) presentan también la pirámide de variabilidad, en la cual se muestra la presencia de la variabilidad externa e interna en los diferentes niveles de abstracción de un producto de software.

La Ilustración 15 muestra cómo (mediante el grosor de la franja vertical gris) la variabilidad externa está presente en su totalidad en artefactos de altos niveles de abstracción de un producto de software y disminuye su presencia en artefactos de niveles de abstracción más bajos. Los autores son enfáticos en aclarar que esta disminución se presenta debido a que en bajos niveles de abstracción las decisiones de manejo de variabilidad giran casi en su totalidad en aspectos de implementación y no de negocio.

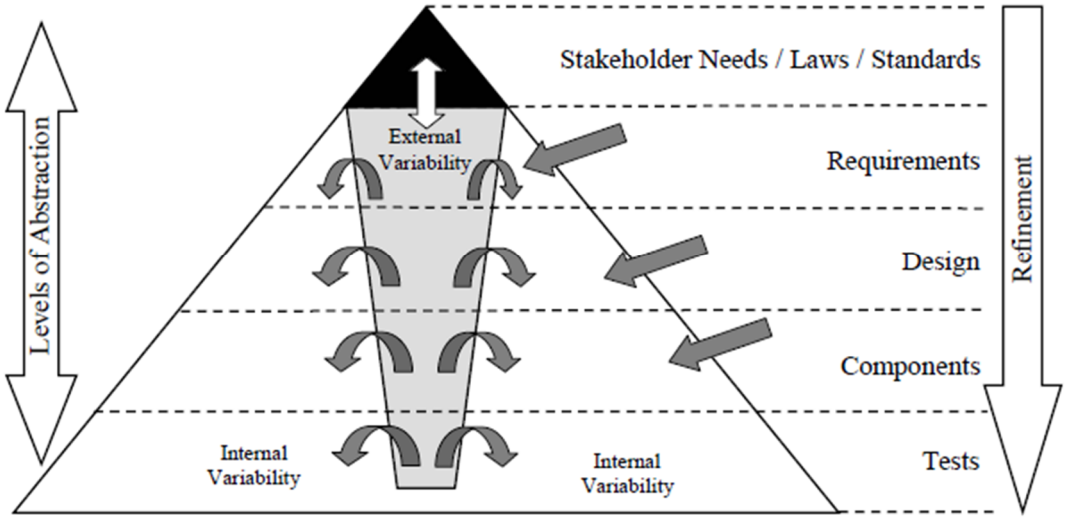


Ilustración 15: Presencia de la variabilidad interna y externa en los diferentes niveles de abstracción de un producto de software. Tomado de Lauenroth & Pohl **(Lauenroth y Pohl 2005)**.

¿Por qué entonces se clasifica al comportamiento de los componentes que ejecutan las tareas del producto Venta en Línea como sujetos de Variabilidad Externa cuando la pirámide indica gráficamente que está más próxima de la variabilidad interna? Esta pregunta se resuelve mirando la porción de la franja gris que se encuentra presente en el nivel de abstracción de los Componentes. Es claro que en la implementación de los componentes las decisiones de variabilidad giran en su mayor parte en torno a cuestiones técnicas como por ejemplo cuál protocolo utilizar para la comunicación con sistemas externos, qué mecanismos de validación de precondiciones, post-condiciones e invariantes utilizar y demás, pero también hay decisiones de alto nivel de abstracción qué tomar como por ejemplo ¿Cómo garantizar que cuando el usuario/cliente solicite un comportamiento nuevo y este esté dentro de las condiciones de posible cambio pactadas, se pueda implementar de forma ágil? ¿Cómo garantizar que el equipo de ingenieros que asuma la responsabilidad del producto encuentre claro dónde implementar un nuevo comportamiento o modificar uno ya existente? Estas cuestiones son las que permiten ubicar a los comportamientos de los componentes del producto Venta en Línea como sujeto de estudio de

variabilidad externa debido a que un software preparado correctamente para el cambio en puntos de variación claros y definidos garantiza que los nuevos comportamientos sean implementados ágilmente. De igual forma, que un nuevo equipo de trabajo comprenda rápidamente dónde hacer un cambio solicitado permite que las solicitudes de cambio de estos comportamientos, o implementación de algunos nuevos, lleguen más rápidamente a ambiente productivo comparado con un software que no está preparado para el cambio en puntos de variación explícitos. Esto, como ya se ha mencionado previamente, habilita la competitividad de Suramericana S.A. en el sector de venta de seguros de vehículos en Internet.

Para comprender mejor las dos categorías adicionales expuestas en la **Tabla 8** enmarcadas en la variabilidad externa, se brinda una explicación de ellas a continuación:

Variabilidad del proceso de venta soportado por el producto

Un proceso es definido como el *"conjunto de las fases sucesivas de un fenómeno natural o de una operación artificial"*[9] En una definición más cercana a la ingeniería de software, en el contexto de BPM se encuentra que un proceso de negocio es el conjunto de actividades que se deben realizar en un orden determinado para lograr un objetivo de negocio. El proceso de negocio especificado en un primer nivel se muestra en la Ilustración 16. Este proceso de negocio tiene como objetivo realizar la expedición de una póliza de seguro de automóviles en el sistema de Suramericana S.A. capturando como primer paso los datos del vehículo a asegurar y verificando si el vehículo debe asistir a una revisión técnica para evaluación previa a la expedición del seguro. Con respecto a la Ilustración 16 se aclara que no obedece a una notación BPMN debido a que el sistema no fue desarrollado como un sistema formalmente orientado a procesos (BPM) y que al ser un diagrama de primer nivel no proporciona información detallada del proceso sino que por el contrario expone el proceso en sus grandes pasos.

Este proceso de negocio especifica los pasos a seguir para vender un seguro de autos y puede ser susceptible de variación debido a que la unidad de negocio encargada de definir el proceso le puede agregar, eliminar o modificar pasos y objetivos, alterando de esta forma la complejidad del proceso como tal. De forma independiente a lo que hagan los componentes que ejecutan las tareas de los diferentes pasos o cómo lo hacen internamente, el proceso como tal puede perfectamente hacer las veces de sujeto de variación, definido por Lauenroth & Pohl en (Lauenroth y Pohl 2005).

Variabilidad del comportamiento de los componentes que ejecutan las tareas del producto Venta en Línea

La mejor manera de exponer esta clasificación de variabilidad Espacial y Externa es estableciendo una definición común para el término *comportamiento*. La referencia bibliográfica que apoya directamente la tarea de encontrar una comprensión común de este término es la de Gamma en (Gamma, y otros 1995) en la exposición del patrón *Estrategia*. Este patrón tiene como propósito "*definir una familia de algoritmos, encapsular cada uno de éstos y hacerlos intercambiables, permitiendo que un algoritmo varíe independientemente de los clientes que lo usan"* (Gamma, y otros 1995). Estos algoritmos pueden ser entendidos como comportamientos, en la medida en que consisten en una serie de pasos que se ejecutan para obtener un resultado en un tiempo finito.

La variación en comportamientos del producto de software Venta en Línea se puede manifestar en términos de reglas de negocio como las que se especifican a continuación:

Selección de productos a cotizar

Que de acuerdo a las preferencias de asegurabilidad del cliente, el sistema presente cotizaciones correspondientes a dichas preferencias. Las preferencias deben poder ser variables y no estáticas, es

[9] Tomado del diccionario de la real academia de la lengua española.

decir, la unidad de negocio podrá definir nuevas opciones de preferencias de tal forma que el usuario pueda tener más o menos opciones en un momento dado.

Tarifación por factores

Que de acuerdo a las características del vehículo, no solamente a su zona de circulación, se puedan especificar incrementos o decrementos en los valores de las primas del seguro. Esto quiere decir que el negocio puede agregar, eliminar, modificar o combinar a su parecer estas características y definir nuevos factores de incremento o decremento de las primas.

Calcular la prima de un seguro

Que de acuerdo a las características de la póliza, el cálculo del valor de prima se realice de diferentes formas, siempre definidas por la unidad de negocio.

Calcular la bonificación comercial de un cliente

Que de acuerdo a la definición de la unidad de negocio, se debe soportar la ejecución de diferentes maneras de calcular una bonificación comercial.

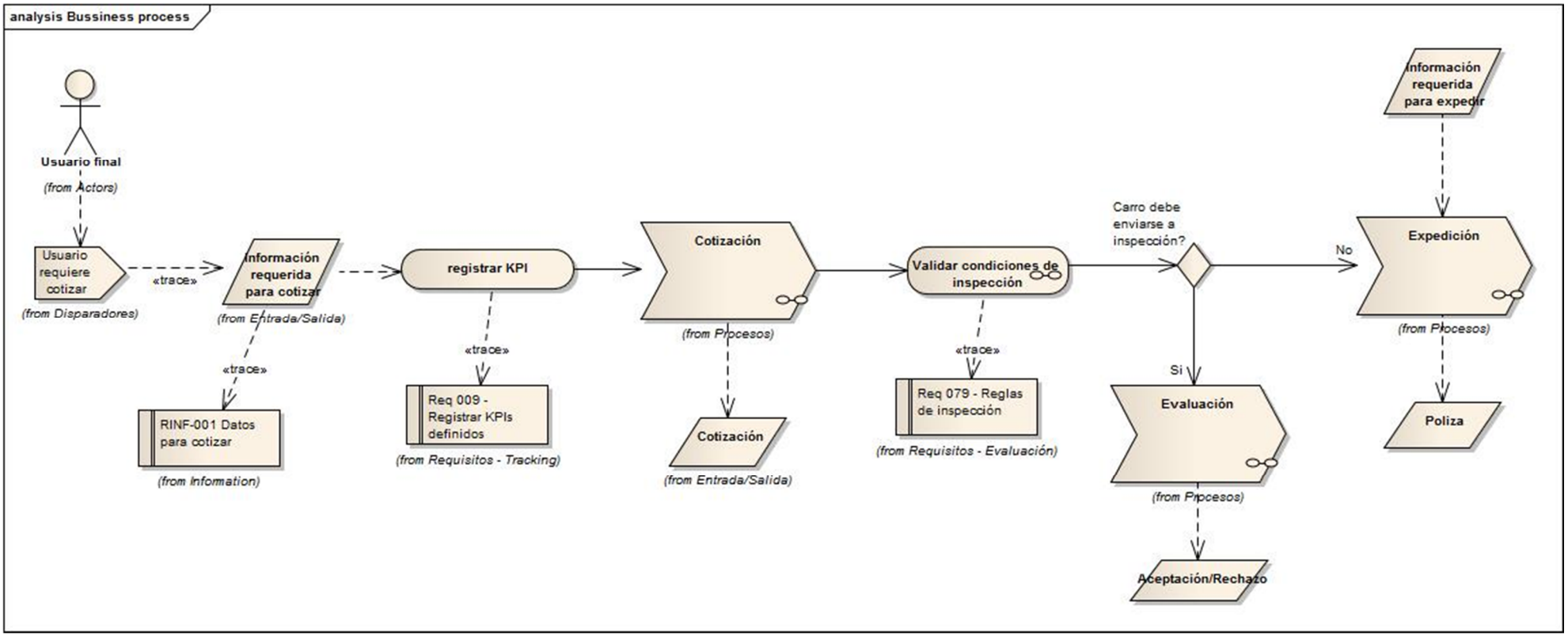


Ilustración 16: Proceso de negocio de alto nivel para la venta de un seguro de autos por Internet en Suramericana S.A.

Lo anterior deja ver cómo a la unidad de negocio no le interesa si, por ejemplo, el cálculo *i* de la bonificación comercial se ejecuta en una clase mientras que el cálculo *j* se realiza en otra clase; al negocio lo que le interesa es que ellos puedan definir diferentes formas de calcular una bonificación comercial y que el sistema lo soporte. Al negocio, en definitiva, le interesa que el sistema, **en determinados puntos se comporte de una manera o de otra**. Allí es donde se conjugan los conceptos de Componente y Variabilidad Externa. Lo que al negocio le interesa, a la Gerencia de Desarrollo de TI también le interesa; lo que al primero le interesa en términos de negocio, al segundo le interesa en términos de artefactos de software. Es muy importante comprender que las necesidades del negocio acerca de la variación en comportamiento se traducen en necesidades de variación en el comportamiento de los componentes que obtienen los resultados esperados por el negocio. Allí es donde nace la segunda categoría de la variación Espacial y Externa descrita en la Tabla 8.

Una vez expuesta la clasificación de las diferentes categorías de la variabilidad y de acuerdo con los motivos descritos en la sección previa se determina que la variabilidad que se representará cuantitativamente será la **variabilidad del comportamiento de los componentes que ejecutan las tareas del producto Venta en Línea** la cual, a su vez, es una categoría descrita como Espacial y Externa. Las razones por las cuales la variabilidad en el proceso de negocio de venta de seguros de autos sobre Internet se excluye del estudio de variabilidad recaen en que la aplicación no es fuertemente orientada por procesos, es decir, Venta en Línea no constituye una solución de software inscrita en el marco de soluciones BPM y se ha considerado que este tipo de variabilidad recae en su totalidad en ese campo de estudio; adicionalmente, las referencias bibliográficas apoyan en su totalidad el estudio de variabilidad en la segunda categoría, razón por la cual se cuenta con mayor referencias bibliográficas que sustenten el proceso de representación a realizar. No está demás indicar que la variabilidad interna queda descartada, puesto que los intereses de estudio de la variabilidad se ubican exclusivamente en aquello que es de interés para el usuario y cliente del producto de software Venta en Línea, como ya fue justificado.

5. Identificación y clasificación de características

En esta sección del documento se identificarán concretamente las características del producto de software Venta en Línea que corresponden a aquellas cuya variabilidad debe ser representada y estudiada de acuerdo a los motivos y al tipo de variabilidad expuestos en la sección anterior. La identificación de características se realizará mediante un nombre y una descripción textual de su propósito en el producto de software. Para cada característica se indicará por qué constituye cada una de ellas un punto de variación; esto se hará identificando qué es lo que varía y cuáles son las diferentes variantes que toman lugar en ella. Una vez descritas las características, se realizará una exposición de cómo están implementadas cada una de ellas en el producto de software Venta en Línea. Para facilitar la lectura de esta sección del documento se expondrá a manera de resumen, una tabla con todas las características variantes la cual especifica el patrón de variación y el mecanismo de variación de cada característica. En las secciones posteriores se brindará el detalle de cada característica.

La labor de clasificación de cada característica, consiste, de acuerdo al marco de referencia de Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002), en determinar la fase de introducción y la fase de enlace de ésta. Es necesario referenciar en este punto las definiciones de estas fases, las cuales tomaron lugar en la sección ***El marco de referencia de Bosch***. Esta clasificación será utilizada en la siguiente sección de este trabajo para ejecutar la medición de variabilidad mediante el marco de referencia de Bosch.

El proceso seguido para la obtención de estas características variantes obedeció al seguimiento pantalla a pantalla de todos los flujos posibles de la aplicación. El autor de este trabajo ejecutó el rol de arquitecto y diseñador líder del producto de software por lo que conocía de antemano tanto los posibles flujos de pantallas como las características en potencia a ser estudiadas. Este conocimiento previo permitió un análisis mucho más detallado y profundo del producto pues el tiempo necesario para la contextualización y conocimiento de la aplicación ya había sido invertido durante el desarrollo del proyecto.

Finalmente, se considera necesario hacer una observación con respecto a la descripción de las características variantes que se presentarán a continuación. De acuerdo con los métodos encontrados para documentación de variabilidad expuestos de forma breve en la sección ***Modelado de variabilidad***, se considera que la forma ideal de presentar una vista general de las características variables y sus respectivas variantes del producto de software Venta en Línea es mediante una opción formal de modelado de variabilidad; sin embargo, la existencia de muy pocas herramientas de uso libre para la realización de estos modelos en sistemas industriales sumada con la dificultad que representa la lectura de un diagrama tan extenso como el que resultaría anularon la posibilidad de presentar dicho modelo en este trabajo.

Se resalta que la búsqueda de herramientas de modelado de variabilidad mediante FODA permitió encontrar a FeatureIDE (FeatureIDE s.f.) y FeaturePlugin (Czarnecki y Antkiewicz 2004). Con la primera se realizaron pruebas de modelado de características del producto en estudio en este trabajo con el resultado ya documentado: un diagrama extenso e ilegible. La herrmienta FeaturePlugin fue construida para el modelado de sistemas y familias de sistemas industriales con gran número de características. Su uso, sin embargo, no fue posible debido a la imposibilidad de su descarga y uso.

5.1. Resumen de clasificación de las características variantes

De forma previa a la descripción detallada de las características variantes del producto de software Venta en Línea, se expone la Tabla 9, la cual resume los patrones y mecanismos de variación de las características expuestas y la Tabla 10 que resume la fase de introducción y enlace de cada una de estas características.

Característica	Patrón de variación	Mecanismo de Variación
Selección de combos de sistemas de gas	Entidades múltiples y coexistentes	Tabla de decisión de BRMS
Selección de combos de blindaje	Entidades múltiples y coexistentes	Tabla de decisión de BRMS
Selección de combos de accesorios	Entidades múltiples y coexistentes	Tabla de decisión de BRMS
Selección de planes a cotizar	Entidades múltiples y coexistentes	Tabla de decisión de BRMS
Presentación de coberturas	Entidad Variante	Implementación de componentes coexistentes.
Lógica de tarifación	Entidades múltiples y coexistentes	Condición en variable
Cálculo de tasas de coberturas	Entidades múltiples y coexistentes	Tabla de decisión de BRMS
Determinación de capacidad de deducción por nómina	Entidad Variante	Condición en variable

Tabla 9: Resumen del patrón y mecanismo de variación de cada característica del producto de software Venta en Línea de Suramericana S.A.

Característica	Fase de introducción	Fase de enlace
Selección de combos de sistemas de gas	Implementación	Tiempo de ejecución
Selección de combos de blindaje	Implementación	Tiempo de ejecución
Selección de combos de accesorios	Implementación	Tiempo de ejecución
Selección de planes a cotizar	Implementación	Tiempo de ejecución
Presentación de coberturas	Diseño	Tiempo de ejecución
Lógica de tarifación	Implementación	Tiempo de ejecución
Cálculo de tasas de coberturas	Implementación	Tiempo de ejecución
Determinación de capacidad de deducción por nómina	Implementación	Tiempo de ejecución

Tabla 10: Resumen de la fase de introducción y fase de enlace de cada característica variante en el producto de software Venta en Línea de Suramericana S.A.

A partir de la identificación de los patrones de variación y las fases de introducción y enlace de cada característica es posible proceder con la representación cuantitativa de la variabilidad del producto de software Venta en Línea. Esto se realizará luego de realizar una descripción detallada de cada característica variante en las secciones siguientes.

5.2. Identificación de las características variantes

5.2.1. Selección de datos del vehículo

Para realizar la cotización de un seguro de un vehículo es necesario conocer los datos del vehículo para el cual se desea obtener una cotización. Datos como la zona de circulación del vehículo, la marca, la línea, el estado del vehículo (si es nuevo o usado), entre otros datos más. La Ilustración 17 muestra cómo la aplicación captura estos datos del vehículo, los cuales son comunes a todos los vehículos que pueden obtener una cotización en el sistema de software en estudio.

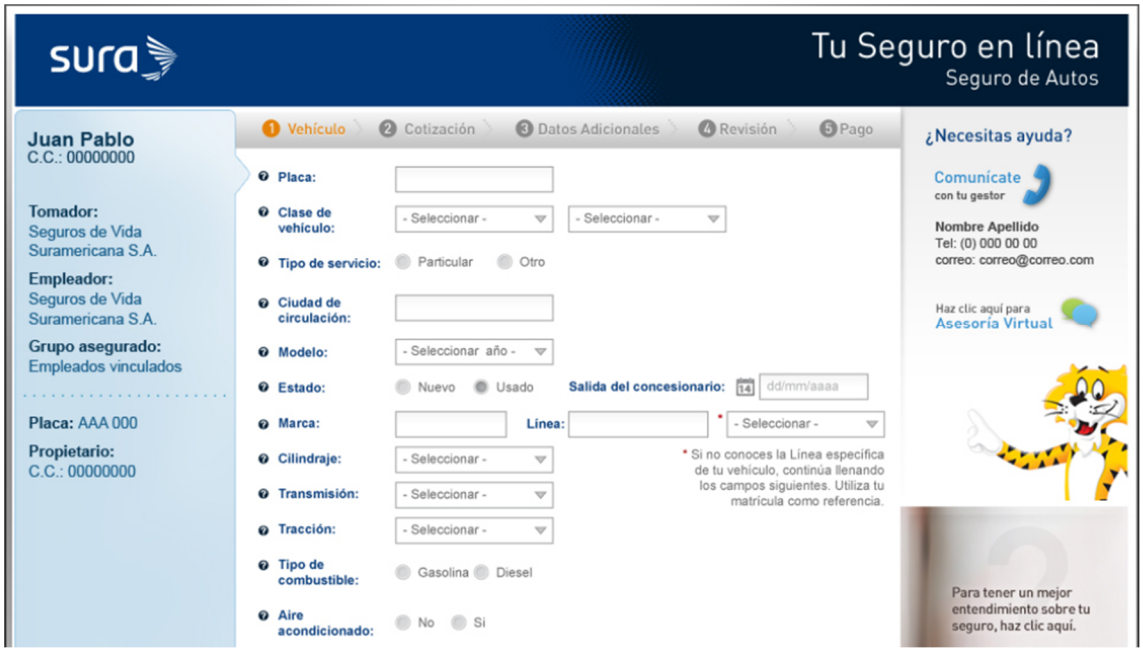


Ilustración 17: Pantalla de captura de datos del vehículo del producto de software Venta en Línea Autos.

De forma adicional, existen algunos datos adicionales que un usuario debe ingresar en caso de que el vehículo los tenga, los cuales son el sistema de gas, el nivel de blindaje del vehículo y los accesorios (radio, rines, pantallas y demás). Debido a que estos datos se obtienen, en el proceso tradicional de venta, cuando el vehículo se dirige a un centro de inspección y en este medio de compra el cliente diligencia estos datos antes de ir a un centro de inspección, la Gerencia de Automóviles, encargada de la definición del producto y del proceso para la venta en Internet, se vio enfrentada a resolver el reto de definir cómo presentar las opciones de sistemas de gas, niveles de blindaje y listado de accesorios a los usuarios finales del sistema. La solución fue acudir a los expertos en inspecciones del equipo de la Gerencia de Automóviles quienes definieron cuáles eran las opciones más frecuentes para estos tres puntos de variación. La necesidad del negocio fue entonces que se construyera la aplicación con estos valores pero que se tuviera en cuenta que en la medida en que se conocieran valores más precisos para éstos campos se requeriría que se pudiera implementar sin afectar la estructura del sistema ni que existiera la necesidad de realizar cambios en el núcleo del sistema. Esto es una muestra evidente de un punto de variación, en el cual el sujeto de variación es el valor da cada categoría y las variantes, los diferentes valores y formas de presentarlas al usuario. En las siguientes secciones se presentarán detalles de cada uno de ellos.

5.2.1.1. Selección del sistema de gas

El sistema de gas es aquél que proporciona energía al motor mediante la combustión de gas natural vehicular. El sistema de gas generalmente es adicionado al vehículo después de la compra. El sistema de gas tiene un costo y es susceptible de ser asegurado, al ser parte del vehículo.

Sin intentar explicar con detalles exhaustivos las razones de por qué es necesario conocer el valor del sistema de gas de un vehículo, solamente se menciona que el valor de este sistema de conversión a gas debe ser tenido en cuenta en el valor asegurado del vehículo pues el propietario de éste estará interesado en que Suramericana S.A. le responda, en caso de un siniestro, no solo por el valor comercial de su vehículo sino también por esta modificación que él realizó posteriormente.

¿Cómo definió presentar Suramericana S.A. este valor? Mediante un concepto denominado "combo de gas" el cual asociará bajo un nombre el valor del sistema de gas del vehículo. Dependiendo del tipo de vehículo (automóvil o campero) se presentan listados diferentes de combos de los cuales el usuario debe seleccionar uno específico. Además de depender del tipo de vehículo, los valores de la lista no deben ser el valor del combo como tal sino una presentación nemotécnica del combo. La Ilustración 18 muestra cómo el sistema de software Venta en Línea muestra los combos de gas. El valor del combo deberá presentarse en la sección de valores asegurados al momento que el usuario realice una selección. La sección de valores asegurados se muestra en la Ilustración 19.

Ilustración 18: Presentación de los sistemas de gas en el producto de software Venta en Línea.

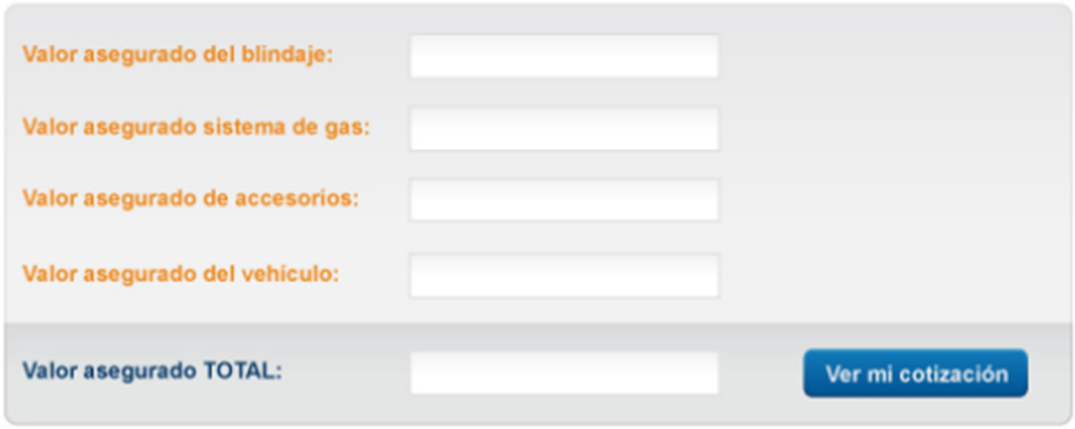


Ilustración 19: Sección de resumen de valores asegurados en el producto de software Venta en Línea.

A simple vista, la lista de valores de los combos de gas no es nada diferente a una lista de valores cuyos elementos dependen de un valor previamente ingresado por el usuario. Esto es correcto y es un ejemplo claro de que la variabilidad está presente en aspectos tan sencillos como éste en un producto de software. ¿Qué es aquello que varía? es la pregunta que plantea Lauenroth & Pohl (Lauenroth y Pohl 2005) para definir un punto de variación. La respuesta es: **los elementos de la lista de los combos de gas** ¿Qué valores puede tomar aquello que varía? es la siguiente pregunta planteada por los autores para identificar las diferentes variantes. La respuesta es: valores definidos por la unidad de negocio para cada tipo de vehículo.

Los combos de blindaje son una primera característica variante del producto de software Venta en Línea.

5.2.1.2. Selección del nivel de blindaje

Un vehículo blindado es aquel que ha sido modificado para interponer en su latonería y ventanas una capa de un material resistente a las balas y artefactos contundentes. De forma similar a lo que ocurre con el sistema de gas previamente expuesto, es el propietario del vehículo quien realiza la modificación haciendo efectivo un pago por este sistema de blindaje. De acuerdo con esto, el propietario de un

vehículo blindado debe indicar a Suramericana S.A. que su vehículo cuenta con este mecanismo de protección de tal forma que el valor asegurado del vehículo se incremente conforme el valor de éste.

Existen diferentes tipos de blindaje, con diferentes valores cada uno. Estos tipos se denominan niveles y básicamente corresponden a diferentes grosores de la capa protectora que conforma el blindaje. De forma adicional, el valor del blindaje depende del año de instalación; así, un blindaje más viejo es menos costos que un blindaje instalado más recientemente. Esto se debe a un factor de depreciación determinado por los expertos en inspecciones de vehículos de Suramericana S.A. Finalmente, una última variable índice en la determinación del valor del blindaje y es el tipo de vehículo; esto quiere decir que los valores de los blindajes son diferentes para vehículos y para camperos.

La Ilustración 20 muestra como el producto de software permite al usuario seleccionar diferentes tipos de blindaje, de acuerdo a un valor previamente seleccionado, el cual es el tipo de vehículo.

Ilustración 20: Selección del nivel de blindaje en el producto de software Venta en Línea Autos de Suramericana S.A.

De igual forma, como con el sistema a gas, estamos frente a una lista de selección en un formulario en la cual, sus valores dependen de un valor previamente ingresado por el usuario. También como con la característica anterior, el equipo de trabajo de la unidad de negocio de Automóviles, solicitó que se pudieran definir nuevos valores de blindaje sin que existiera la necesidad de realizar cambios en el núcleo del sistema para que éstos tomaran efecto en el sistema de software.

Las respuestas a las preguntas planteadas por Lauenroth & Pohl para determinar puntos de variación y sus respectivas variantes son en este caso: las opciones de la lista de los niveles de blindaje de acuerdo al tipo y modelo del vehículo (en el caso de la pregunta acerca del punto de variación) y los diferentes valores definidos por la Gerencia de Automóviles (en el caso de la pregunta acerca de las variantes).

El valor del blindaje no se muestra, como con la característica anterior, directamente en el valor de la lista; este se muestra también en el cuadro resumen de valores asegurados que se muestra en la Ilustración 19.

5.2.1.3. Selección de accesorios

Los accesorios de un vehículo son aquellos aditamentos los cuales han sido adicionados por el propietario del carro y generalmente constituyen lujos del vehículo. Estos accesorios son aquellos equipos o aditamentos que no son de fábrica que han sido adicionados al vehículo de forma particular por el propietario. Equipos de sonido, radios, rines de lujo, pantallas, alarma y demás constituyen la lista de accesorios que frecuentemente presentan los vehículos que desean ser asegurados en Suramericana S.A.

Si el propietario desea que Suramericana S.A. asuma responsabilidad de asegurabilidad sobre estos accesorios, es necesario que el potencial cliente especifique el valor de los accesorios del vehículo de tal forma que estos pasen a ser parte del valor asegurado y por ellos se cobre una prima adicional.

Estos accesorios, como sucede con el sistema de gas y el nivel de blindaje, son examinados en el momento de la inspección en el proceso de venta tradicional. Una vez más, la gerencia de automóviles se decidió por presentar los denominados *combos de accesorios*, los cuales son una agrupación de los accesorios más comunes presentados por los vehículos en los procesos de inspección tradicionales. En este caso, a diferencia de los dos anteriores, sí se muestra un valor asegurado de los accesorios y adicionalmente se muestra como un *tooltip* los accesorios que el combo cubre. En caso de un siniestro, al cliente se le responderá financieramente por el valor de accesorios indicado en esta opción. Finalmente, se especifica que el conjunto de opciones ofrecidas al usuario final del sistema depende del tipo de vehículo (automóvil o campero), el cual es un dato que se diligencia previamente en la misma pantalla.

La Ilustración 21 muestra cómo el producto de software Venta en Línea Autos ofrece al usuario final diferentes opciones de combos de accesorios. Estas opciones dependen del tipo de vehículo diligenciado previamente.

Ilustración 21: Selección del valor del combo de accesorios en el producto Venta en Línea Autos.

Esta característica constituye un punto de variación adicional en la aplicación en la sección de captura de datos de vehículo. Aquello que varía es los elementos del conjunto de opciones que se deben presentar al usuario final con respecto a los combos de accesorios. Los diferentes valores que toma son las diferentes opciones que define la unidad de negocio para cada tipo de vehículo.

5.2.2. Selección de los productos a cotizar

La unidad de negocio de Automóviles de Suramericana S.A. diseñó tres planes de póliza los cuales constituyen la oferta de seguros de autos de la compañía a los clientes. Estos planes se diferencian entre sí por las coberturas que ofrecen y adicionalmente, por las características de las coberturas en cada producto. Para ser un poco más claros y dado que esta explicación será de utilidad para una de las características posteriores, se aclara que una cobertura en una póliza de seguro se puede entender como la protección que brinda el seguro al vehículo y a sus ocupantes. Así por ejemplo, la cobertura de Pérdida Total por Hurto protege al vehículo en caso de que sea sujeto de un hurto del automóvil; a su vez, la cobertura Pérdida Parcial por Hurto cubre los robos parciales como por ejemplo farolas, espejos, llantas y demás que puedan ser sustraídos del vehículo. Los tres productos mencionados anteriormente se denominan Plan Básico, Plan Clásico y Plan Global. En ese orden, una póliza de plan básico ofrece coberturas básicas que cualquier asegurado debe tener. El Plan Clásico ofrece todas las coberturas posibles pero las cláusulas del contrato de seguro hacen que la prestación de los servicios no sea la máxima posible por parte de Suramericana S.A. aun cuando es un plan de las mejores características. El Plan Global, por su parte, es el mejor plan de póliza que ofrece Suramericana S.A. pues ofrece todas las coberturas y éstas con todas las características posibles de servicio por parte de la compañía. Un ejemplo de diferencia entre el Plan Clásico y el Plan Global reside en la distancia en la que una grúa puede asistir a un vehículo asegurado; en el plan Clásico una grúa asistirá a un vehículo máximo a 40

kilómetros por fuera del área metropolitana de Medellín mientras que en Plan Global esta distancia es mucho mayor.

La combinación de coberturas y planes genera un conjunto de posibilidades de productos de cardinalidad muy grande. Esto quiere decir que si a un usuario se le presentan todas las opciones de configuración de planes esta persona se podría confundir. La necesidad de la unidad del negocio radicaba en ofrecer cotizaciones a los usuarios finales solamente de los planes que aplican para ellos y que satisfacen sus necesidades de asegurabilidad. La pregunta que surge entonces es ¿Cómo determinar qué productos aplican para cada usuario? La primera aproximación se resolvió mediante el diseño de un cuestionario el cual es de opcional respuesta por el usuario final y cuyas respuestas son el insumo para la determinación de qué productos cotizar y posteriormente mostrar en pantalla al usuario. La Ilustración 22 muestra algunas preguntas del cuestionario definido por la Gerencia de Automóviles mientras que la Ilustración 23 expone cómo se muestran las diferentes cotizaciones de los diferentes planes al usuario final.

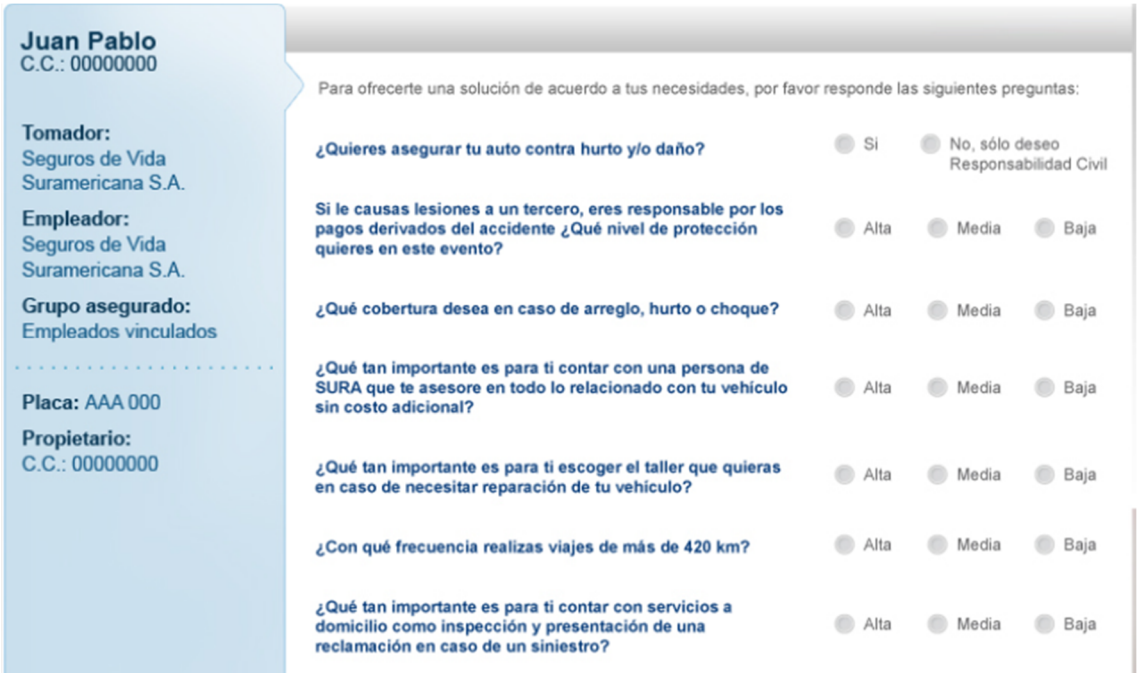


Ilustración 22: Cuestionario que puede responder el usuario final para la determinación de sus necesidades de asegurabilidad y posterior selección de planes a cotizar.

Ilustración 23: Presentación de cotización de diferentes planes al usuario final en el producto de software Venta en Línea.

Una abstracción inicial para solucionar esta necesidad consistía en seleccionar los planes que se debían cotizar basados en las respuestas a las preguntas del cuestionario. Sin embargo, la necesidad evolucionó para indicar que no solamente se debían tener en cuenta estas respuestas; estas iban a ser solamente criterios iniciales pero posteriormente se determinarían criterios adicionales como por ejemplo la marca del vehículo para la cual se cotizaría el seguro, esto último debido a que no tenía mucho sentido que a un automóvil de gama alta se le presentaran ofertas de cotización de los planes básicos. La abstracción evolucionó también entonces para convertirse en la selección de los planes a cotizar de acuerdo a criterios genéricos los cuales podían ser determinados por el área de negocio sin que su implementación requiriera modificar el núcleo del sistema.

En esta característica del producto de software se encuentra un punto de variación adicional a los ya descritos previamente ¿Qué es aquello que varía? el conjunto de planes para los cuales se debe realizar y presentar una cotización al usuario final ¿Cuáles son los posibles valores que puede tomar este conjunto? Aquellos que defina el negocio de acuerdo a la dinámica del sistema en la medida en que se efectúen ventas de los diferentes planes.

5.2.3. Presentación de coberturas

En una póliza de seguro se contratan una o más coberturas. Estas coberturas son las que protegerán a aquél bien asegurado, en el caso del producto en estudio, un vehículo. Cada unidad de negocio es responsable de diseñar las coberturas de acuerdo a la estrategia de la compañía. En el caso de automóviles, el producto cuenta con doce coberturas diferentes, las cuales protegen al vehículo en aspectos diversos como el hurto, los daños, los accidentes al conductor, asistencia en viaje, entre otros.

Si bien las coberturas tienen una estructura común, al momento de presentarlas al usuario en una pantalla de cotización se deben tener en cuenta detalles que son diferentes entre ellas, de tal forma que al usuario final le queden claras las características de cada cobertura. Las ilustraciones que se muestran a continuación exponen tres diferentes coberturas del producto de automóviles en su representación en pantalla en el producto de software Venta en Línea. A continuación se expondrá de forma breve las diferencias entre ellas de tal forma que queden claras al lector.

La Ilustración 24 muestra cómo se presenta la cobertura Responsabilidad Civil en la pantalla de cotizaciones de Venta en Línea. Si bien se trata de una sola cobertura, el valor asegurado de ésta consta de cuatro partes diferentes, los cuales corresponden al de *Daños a bienes de terceros*, *Lesiones Personales*, *Asistencia Jurídica*, y *Exceso de RC*. Ninguna otra cobertura divide su valor asegurado de esta manera lo que hace la presentación de esta cobertura algo específico. De forma adicional, el título en el valor asegurado, por norma legal, debe ser *Límite Asegurado* y no *Valor Asegurado*. Esto quiere

decir que en caso de una reclamación, el valor máximo por el que Suramericana S.A. como aseguradora realiza el pago es el valor descrito. Poner como título *Valor Asegurado* indicaría que en caso de una reclamación, Suramericana S.A. pagaría siempre este valor, lo cual no es cierto.

La Ilustración 25 da cuenta de cómo se presenta en la pantalla de cotizaciones las coberturas de Daños al vehículo. Si bien la presentación es muy similar a la presentación de la cobertura de Responsabilidad Civil explicada previamente, esta cobertura cuenta con características particulares que hacen que su manejo sea también particular. Primero que todo, para estas coberturas, el título del valor asegurado sí debe ser la cadena *Valor Asegurado*; ahora bien, el valor asegurado siempre debe llevar la constante *"valor comercial"* esto debido a que en caso de una reclamación, el valor que Suramericana S.A. pagará será sobre el valor comercial del vehículo a la fecha de la reclamación y no el valor de compra ni el valor comercial a la fecha de cotización ni adquisición del seguro. De forma adicional, las coberturas de daños son dos (Pérdida Total por Daños y Pérdida Parcial por Daños) a diferencia de la cobertura de Responsabilidad Civil y otras más en las que solamente se presenta una sola cobertura.

Ilustración 24: Presentación de la cobertura Responsabilidad Civil en el producto Venta en Línea de Suramericana S.A.

Ilustración 25: Presentación de las coberturas de daños en el producto de software Venta en Línea de Suramericana S.A.

La Ilustración 26, finalmente, muestra cómo se presenta la cobertura Asistencia en la pantalla de cotizaciones del producto de software Venta en Línea. En este caso, a diferencia de los expuestos previamente, no se muestra ningún título para el valor asegurado, pues éste no existe en esta cobertura. De forma particular para esta cobertura, solamente se muestra el tipo de asistencia que se contrataría en la póliza correspondiente a dicha cotización. En el ejemplo de la Ilustración 26 ambas cotizaciones muestran la descripción de una asistencia de tipo Global.

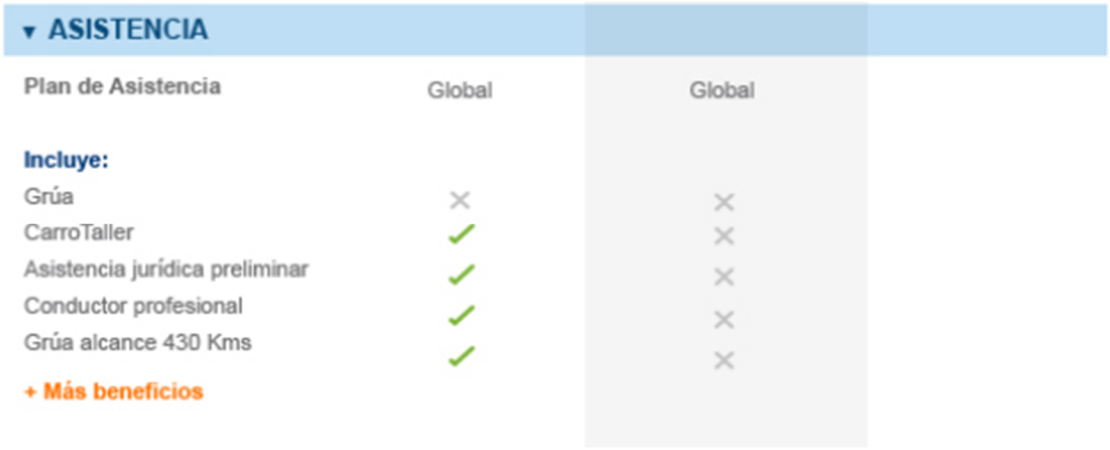


Ilustración 26: Presentación de la cobertura de asistencia en el producto de software Venta en Línea de Suramericana S.A.

La presentación de coberturas debe considerarse como un punto de variación adicional del producto de software Venta en Línea en el cual aquello que varía para cada cobertura es la forma de presentación en pantalla y las diferentes variantes las constituyen las diferentes maneras que se han ejemplificado en esta sección y las cuales están definidas e implementadas en su totalidad en el producto de software Venta en Línea para el conjunto completo de coberturas del producto automóviles.

5.2.4. Tarifación

La tarifación es el proceso mediante el cual se calcula el valor de prima de una cotización de un seguro. El valor de prima es el valor a pagar por el cliente para contratar un seguro con una compañía aseguradora, en este caso, Suramericana S.A. Los encargados de definir qué cálculos se deben realizar para obtener el valor de la prima de un seguro se llaman *actuarios* y pertenecen a cada unidad de negocio en la compañía. Así las cosas, la Gerencia de Automóviles como unidad de negocio encargada del producto de Automóviles cuenta con actuarios encargados de definir los insumos de información y los cálculos necesarios para obtener el valor de prima de una póliza de autos.

En el trabajo de identificación de características del producto de software Venta en Línea que constituyan puntos de variación se encuentra que en la tarifación existen dos puntos de variación importantes, los cuales se explican a continuación.

Para comprender estos puntos de variación es necesario tener en cuenta los siguientes puntos:

- Una póliza agrupa a uno o más vehículos
- Un vehículo contrata una o más coberturas.
- La prima para un vehículo es la suma de la prima de sus coberturas.
- La prima de una póliza es la suma de las primas de sus vehículos.

Esto deja ver que la unidad de tarifación es la cobertura. Ahora bien ¿Cómo se obtiene el valor de prima de una cobertura? Sin entrar en detalles técnicos, sucede de la siguiente manera:

Existen dos tipos de coberturas, unas cuya prima se determina a partir del valor asegurado del vehículo y otras cuya prima es calculada sin tener en cuenta el valor asegurado del vehículo. La prima de las primeras se calcula multiplicando el valor asegurado del vehículo por una tasa; la prima de las segundas es determinada por la Gerencia de Automóviles. Tanto las tasas como las primas varían de acuerdo a las características del vehículo; esto hace entonces que una misma cobertura, por ejemplo Responsabilidad Civil, tenga un valor para un vehículo que circula en Medellín y tenga otro valor para un vehículo que circula en Bogotá. De igual manera, esta misma cobertura tiene un valor diferente para vehículos más antiguos que para vehículos de modelos más recientes. El primer caso de variación gira en torno a una excepción a esta descripción mientras que el segundo se relaciona con el manejo de variación en los precios de las coberturas con respecto a las características del vehículo.

5.2.4.1. Tarifación por tasa única y tarifación por tasas de cobertura

La descripción general brindada previamente acerca de la tarifación por coberturas tiene una excepción. En algunos casos, sobretodo en negociaciones grandes, no se asignan valores de tasas y primas por cobertura. En estos casos especiales lo que hace la Gerencia de Automóviles es definir una única tasa para toda la póliza. Esta tasa será multiplicada por el valor asegurado del vehículo y ese, finalmente, será el valor de prima para cada vehículo.

El producto Venta en Línea también opera sobre pólizas con este tipo de tarifación por lo que ante una póliza de este tipo el producto de software debe seleccionar un comportamiento específico de tarifación y ejecutarlo con el objetivo de obtener el valor de prima adecuado.

En este punto se evidencia un nuevo punto de variación. Aquello que varía es el cálculo a realizar para obtener el valor de prima de un vehículo. Los diferentes valores que puede tomar son el tradicional y el excepcional, denominados en el negocio como ***tarifación por tasas por cobertura*** y ***tarifación por tasa única*** respectivamente.

5.2.4.2. Cálculo de tasas de coberturas

En el comportamiento tradicional de tarifación por coberturas explicado al inicio de esta sección se expuso cómo existen dos tipos de cobertura: aquellas que se tarifan mediante la multiplicación del valor asegurado por una tasa y aquellas a las que se asigna un valor de prima. Adicionalmente se expuso que los valores de las tasas y de las primas de estas coberturas varían de las características del vehículo.

Al momento de comenzar el desarrollo del producto de software Venta en Línea se encontró que la consulta de las tasas y primas de coberturas de acuerdo a las características del vehículo ya se encontraba implementada y que por tanto se reusaría el componente que tenía esta responsabilidad. Sin embargo, se encontró con el hecho de que dicha implementación no tenía tiempos de respuesta adecuados y por lo tanto no estaba alineado con el driver de arquitectura **Desempeño** seleccionado para el producto de software (ver sección *Drivers de arquitectura*). De esta forma se decidió construir un componente que permitiera consultar las tasas y las primas de las coberturas a partir de las características del vehículo a asegurar. Este componente debía satisfacer las siguientes características específicas de variabilidad:

- Se debe poder adicionar el cálculo de primas por cobertura. Esto quiere decir que en cualquier momento el cálculo de prima de una cobertura puede ser definido de forma diferente de aquellos ya establecidos.

- Las características del vehículo que determinan la tasa o el valor de prima de cada cobertura pueden variar en el tiempo. Esto quiere decir que si actualmente solamente se tienen como modificadores de la tasa o prima la ciudad de circulación y la antigüedad del vehículo, el componente debe poder soportar que el equipo de negocio determine nuevos factores de modificación. Incluso, debe soportar combinaciones de estos factores satisfaciendo así las necesidades de los actuarios para calcular precios de las coberturas de acuerdo a las condiciones del mercado de seguros en el país.

El segundo punto puede resultar confuso y por tal motivo se procederá con un ejemplo muy breve para aclararlo. A continuación el ejemplo:

Un automóvil que desee contratar la cobertura de Responsabilidad Civil debe pagar un valor de prima, por dicha cobertura, de $100.000. Sin embargo, según un análisis actuarial, se ha determinado que si el vehículo circula en la ciudad de Medellín, este valor de prima debe ser incrementado en un 10%. De forma adicional, si el vehículo es último modelo obtendrá una reducción en el valor de la prima de un 10%. Ahora, si el vehículo circula en la ciudad de Bogotá, el valor de la prima se incrementará en un 40%. Así las cosas, la prima para el vehículo de la ciudad de Medellín sería de $99.000 mientras que la prima para la misma cobertura para el automóvil de la ciudad de Bogotá sería de $126.000.

Más adelante en el tiempo, la Gerencia de Automóviles solicita lo siguiente: *"Los vehículos de la ciudad Medellín con cero años de antigüedad* ***y de la marca XYZ*** *tendrán un incremento adicional del 40%"* El producto de software ha sido solicitado para soportar estas variaciones sin tener que modificar el núcleo del sistema.

En esta característica la necesidad del manejo de variabilidad es evidente. La variabilidad en esta característica debe dividirse en los dos siguientes puntos de variación de mayor nivel de detalle:

- Variabilidad en el cálculo a realizar por cobertura.
- Variabilidad en el manejo de factores modificadores de las tasas o primas de coberturas.

Es importante tener en cuenta esta división interna pues serán tenidas en cuentas como características independientes al momento de realizar la medición y análisis de resultados en secciones posteriores.

5.2.5. Determinación de capacidad de deducción por nómina

Una vez el usuario final toma una decisión acerca del producto que desea comprar procede con el diligenciamiento de datos adicionales necesarios para la expedición del seguro. Finalmente se realiza el pago en la pantalla de pago. Es necesario recordar en este punto que el producto de software Venta en Línea fue planeado para construirse en dos fases; la primera fase está destinada a ser usuarios por empleados de la compañía mientras que la segunda fase está destinada a navegantes en Internet. En la primera fase del producto, el único medio de pago disponible es el descuento por nómina del valor de la prima. De esta forma, en el pago de cada quincena la compañía procede a descontar el valor de prima correspondiente para ese período. Esto se repite hasta el final de la vigencia del seguro.

El producto de software Venta en Línea, sin embargo, debe ejecutar una validación sobre la capacidad de deducción por nómina del empleado. Esto quiere decir que el sistema debe tener en cuenta si el empleado, con su nómina y sus deducciones, puede asumir tranquilamente la deducción del valor de prima de la póliza del vehículo. En caso de que su capacidad de deducción tolere la deducción del valor de prima, el sistema continuará con la expedición de la póliza; en caso contrario, el sistema deberá interrumpir el proceso dando aviso al empleado de que no puede continuar con el proceso de expedición.

La unidad de negocio encargada de hacer la definición acerca de cuándo un empleado puede asumir la deducción por nómina del valor de la prima de la póliza no fue en este caso la Gerencia de Automóviles sino la Gerencia de Gestión Humana la cual es la encargada de todos los temas salariales de la compañía. Esta unidad de negocio planteó los cálculos a realizar para determinar cuándo es posible y cuándo no lo es proceder con la deducción de la nómina de un empleado. En esta definición se hizo una especificación de variabilidad en el cálculo. Por razones de confidencialidad se debe limitar la exposición a que para ciertos cargos se debe ejecutar un cálculo especial y para el resto de los cargos se debe ejecutar un cálculo tradicional.

Esta característica denota variabilidad evidentemente. Aquello que varía es el cálculo a realizar para determinar si el valor de la prima puede ser deducido de la nómina de un empleado. Las variantes en este punto de variación las constituyen los dos diferentes tipos de cálculos, a saber: el cálculo para los empleados de los cargos específicos y el cálculo tradicional para el resto de los cargos.

5.3. Descripción de implementación de las características variantes

Para poder realizar una clasificación de las características que cuentan con variabilidad expuestas en la sección anterior, es necesario conocer con un nivel de detalle adecuado cómo están implementadas en el sistema. A continuación se presentará la forma como fueron implementadas estas características en el producto de software Venta en Línea de tal modo que en la siguiente sección se pueda determinar para cada una su fase de introducción y de enlace. Debido a que algunas de las características fueron implementadas haciendo uso de un componente denominado *Configurador de Objetos*, se hace necesario hacer una introducción de este componente en esta sección. Así pues, se procederá inicialmente con la descripción de la funcionalidad de este componente y algunas características técnicas del mismo para posteriormente describir cada una de las características documentadas.

5.3.1. Configurador de Objetos

5.3.1.1. Descripción de funcionalidad

El configurador de objetos es un componente desarrollado por Suramericana S.A. para satisfacer la necesidad inicial de determinar qué planes se deben cotizar ante los datos ingresados por un usuario del producto de software Venta en Línea. Este componente permite la asociación de criterios de selección específicos con los planes que se deben cotizar y presentar al cliente en pantalla. Así las cosas, este componente permite determinar, por ejemplo, que para un usuario al que solamente le interesa asegurar su vehículo por Responsabilidad Civil y cuyo vehículo es un automóvil entonces el producto de software le cotizará y presentará en pantalla planes del plan básico de pólizas de autos; así también, permite determinar que para un usuario que desea coberturas contra daños y hurto y cobertura de la mejor calidad, se deben cotizar y presentar en pantalla las diferentes variaciones del plan global.

En este punto es muy importante tener en cuenta la referencia a la necesidad de variación con respecto a los criterios de selección de planes planteada por la Gerencia de Automóviles expuesta en la sección anterior. Se debe recordar que de antemano se sabe que la Gerencia de Automóviles variará la asociación de criterios con los diferentes planes de tal forma que si para un momento dado los criterios *tipo de vehículo* y *necesidad de cobertura contra daño y hurto* determinan que se debe seleccionar el *Plan Básico* en un momento posterior y una vez realizada la debida solicitud a la Gerencia de Desarrollo de TI, estos mismos criterios deben poder determinar que también se debe cotizar el *Plan Básico + Asistencia.* De forma adicional, la Gerencia de Automóviles también ha solicitado que ante una necesidad de adición o eliminación de criterios, la implementación del cambio se pueda realizar de una forma ágil; así por ejemplo, si en un momento los únicos criterios que existen son los ya mencionados en

un momento posterior, y luego de radicar una solicitud a la Gerencia de Desarrollo de TI, puedan haber dos nuevos criterios llamados *necesidad de asistencia en viaje* y *frecuencia de viajes de más de 400 km.*

Una primera aproximación de solución a esta necesidad puede ser mediante una tabla en la base de datos de la aplicación en la cual las columnas sean los criterios de selección y así hacer la consulta de los planes que se deben cotizar. Los siguientes puntos permitieron no seleccionar esta solución:

- Para satisfacer la necesidad de variabilidad en los criterios, la implementación sería mediante la adición o eliminación de columnas de dicha tabla. La adición y eliminación frecuente de columnas en un modelo de datos no es considerada una buena práctica en la Gerencia de Desarrollo de TI debido tanto a motivos de gobierno sobre la modificación de modelos de datos como a motivos técnicos que apuntan a la degeneración del desempeño en términos de tiempos de ejecución y consumo de memoria física en tablas con modificaciones frecuentes.

- De realizarse en una tabla con la columnas como criterios (como lo sugeriría un enfoque tradicional) la escritura de la consulta debe ser inevitablemente una consulta en SQL. Si se adicionan o eliminan criterios, como bien lo sugiere la necesidad de variabilidad de los criterios, la implementación sería mediante la edición de la sentencia SQL de consulta. Esto trae varias desventajas entre las cuales se encuentra un posible detrimento del desempeño cuando el número de criterios crezca considerablemente.

- De necesitarse la asociación de criterios para selección de planes de otros productos diferentes a Automóviles en proyectos futuros de la estrategia de Venta en Línea (entiéndase la construcción de productos de software para la venta de seguros de Hogar o Salud por Internet), existirían tres alternativas: 1) crear una tabla para el nuevo producto (p.ej Hogar) lo cual acarrearía las dos consecuencias previamente expuestas, 2) Adicionar columnas para los criterios específicos del nuevo producto lo cual desnormalizaría el modelo de datos, y 3) generar desde un comienzo una tabla con nombres de columnas genéricos de tal forma que una columna tenga un significado en un producto y otro completamente diferente en otro producto. A esto se le denominó *Abismo Semántico* al interior del equipo de trabajo, pues se genera un abismo entre el artefacto de software (en este caso la tabla) y lo que éste significa para las diferentes unidades de negocio.

La decisión tomada para dar solución a este escenario propuesto fue utilizar el motor de reglas de negocio *Jboss Drools. Drools* es un BRMS[10] de código abierto que presta las funcionalidades de motor y repositorio de reglas. En términos generales, un BRMS consiste en una herramienta que permite administrar la lógica de decisión referente a aspectos de negocio en los productos de software de una organización. Las preguntas inmediatamente siguientes a esta definición de alto nivel son ¿Qué es *lógica de decisión*? Y ¿Qué significa *administrar* la lógica de decisión?

Si bien lo más apropiado en este trabajo, el cual no tiene dentro de su alcance la exposición precisa de un BRMS, sería ofrecer un ejemplo que dé a conocer sus capacidades funcionales, se ha optado por proveer una definición general y precisa de estos dos conceptos y posteriormente realizar una ilustración mediante un ejemplo. Así pues, se debe entender que *lógica de decisión* consiste en aquellas reglas que rigen la toma de decisiones de ejecución de los sistemas de software operacionales de una organización y por *administrar la lógica de decisión* se debe entender la capacidad de definir, desplegar, ejecutar, monitorear y mantener dichas reglas (Hegde y Wall 2009). Para ofrecer un ejemplo claro que toma como objeto de estudio al producto de software Venta en Línea, se tomará como caso de estudio el escenario en el cual se debe seleccionar un plan a cotizar a partir de determinados criterios de selección, determinados por el negocio.

[10] Business Rules Management System

Considere nuevamente la característica descrita en la sección 5.2.2. En dicho escenario, el negocio puede establecer lo siguiente:

Si el vehículo que el usuario final desea asegurar es un automóvil ***y***

al usuario final le interesa asegurar su vehículo con coberturas contra daños y hurto ***y***

al usuario final le interesa un nivel alto de cobertura en caso de un percance en un viaje

entonces *se debe ofrecer una cotización del producto Plan Autos|Sura Global al 90%.*

Es claro que implementar un componente o servicio o en general una pieza de software que realice estas validaciones no representa ninguna complejidad pero ¿Qué sucede cuando las condiciones o las consecuencias de dicha validación cambian constantemente? Inevitablemente se llegará con el paso del tiempo a una pieza de software compleja y poco legible como consecuencia de la codificación de esta lógica de decisión. Ahora ¿Qué pasa si otro producto de software necesita realizar estas mismas validaciones de selección de producto y no tiene en cuenta que este producto de software ya las implementó en un componente interno? Se construirá una nueva pieza de software que hace lo mismo pero con abstracciones propias del nuevo producto que la requiera.

Un BRMS permite la escritura de esta lógica de decisión, o "reglas" como se denomina en el vocabulario de los BRMS, en un lenguaje específico destinado para ello; este lenguaje permite que esta lógica de decisión sea escrita en términos de negocio y no en términos técnicos como lo son por ejemplo las sentencias de control de flujo –sentencias *if* -, tradicionalmente utilizadas para dar solución a este tipo de escenarios. De forma adicional, una solución de BRMS permitirá el almacenamiento centralizado de estas reglas, de tal forma que si un nuevo producto de software las necesita, entonces fácilmente las puede acceder y ejecutar haciendo de esto un reuso efectivo, pues solamente se escribieron las reglas una vez, pero fueron utilizadas por diferentes clientes.

En el caso específico de *JBoss Drools*, las reglas se pueden escribir de tres maneras diferentes pero equivalentes entre sí: 1) En el lenguaje de escritura de reglas de este BRMS llamado *MVEL*, 2) Mediante tablas de decisión las cuales se especifican en hojas de cálculo de Excel y 3) En lenguajes específicos de dominio. Cualquiera de estas tres formas de escribir reglas es mucho más cercana a los términos del negocio que cualquier implementación en un lenguaje de programación tradicional.

¿Cómo se enlaza lo descrito acerca del BRMS y la selección de productos que se deben cotizar y mostrar en pantalla? En este caso se diseñó una solución para la selección de productos mediante una tabla de decisión en la cual cada columna, menos la de la extrema derecha, corresponde con un criterio de selección de plan. En la Ilustración 27 se muestra parcialmente esta tabla de decisión. En esta ilustración se muestran, por efectos de comodidad en la visualización, solamente tres columnas con criterios de selección. Para comprender cómo funciona esta tabla de decisión, es necesario primero dar una mirada a la Ilustración 28 para observar como por pantalla se solicita al usuario final una serie de respuestas a preguntas claves que permiten al sistema tomar la decisión acerca de qué plan cotizar y presentar.

10			
11	criterioComparacion.criterio7=="$param"	criterioComparacion.criterio8=="$param"	criterioComparacion.criterio9=="$param"
12	Hurto y/o daño	Protección terceros	Cobertura arreglo, choque
13	No, sólo deseo responsabilidad civil		
14			
15			
16			
17	Si	Alta	
18			90%
19			100%
20			
21			90%

Ilustración 27: Criterios de la tabla de decisión de selección de productos a cotizar del producto de software Venta en Línea.

Una vez se tienen las respuestas del usuario, éstas se entregan al BRMS para que ejecute las reglas contenidas en la tabla de decisión. Cada respuesta del cuestionario corresponde a una columna de la tabla de decisión. En la Ilustración 27 se muestran las columnas pobladas que corresponden a las tres primeras preguntas mostradas en la Ilustración 28.

Ilustración 28: Pantalla de respuestas de cuestionario para indicar al sistema los insumos que se deben tener en cuenta al momento de seleccionar los planes a cotizar.

El plan que se debe cotizar se identifica mediante un número único de plan el cual es especificado en la columna más a la derecha de la tabla de decisión. La Ilustración 29 muestra cómo se especifica el plan que se debe seleccionar mediante el identificador único de objeto a seleccionar.

criterioComparacion.criterio10=="$param"	criterioComparacion.criterio11=="$param"	consultaObjeto.addIdObjetoSeleccionado($param);
Facilitador asesor	Eleccion de taller	Escoger objeto
		15146
		15222
		15298
		15464
Media	Media	15633
		15981
Alta	Alta	16329
		16781
		17233
		15146

Ilustración 29: Columna de selección de plan a cotizar en el Configurador de Objetos.

El objetivo inicial de este componente llamado Configurador de Objetos era la selección de planes a cotizar a partir de criterios de negocio. Sin embargo, en etapa de diseño detallado del componente, el equipo de trabajo pudo darse cuenta que el componente podía prestar una funcionalidad general que permitiera realizar una selección de un objeto a partir de un conjunto de criterios. Para que esta funcionalidad pudiera ser general se necesitaba que tanto los criterios como los objetos seleccionados tuviera un nivel de abstracción tal que cualquier aplicación cliente que necesitara esa funcionalidad la pudiera utilizar. El diseño obedeció a esta necesidad y se desarrollaron tanto criterios genéricos como objetos genéricos.

Para brindar una perspectiva un poco más técnica del Configurador de Objetos, la Ilustración 30 muestra los diferentes componentes internos del sistema. La interface *ObjectBuilderFacade* provee la operación que permite a partir de un conjunto de criterios seleccionar un conjunto de objetos que deban ser seleccionados de acuerdo a las reglas de negocio de selección. Esta interface ejecuta las reglas de selección en el BRMS y posteriormente le solicita al componente *ObjectBuilder* el ensamblaje de los objetos correspondientes. Lo que hace el *ObjectBuilder* es consultar la estructura del objeto en base de datos (sus atributos y sus valores) y crear una instancia del tipo de dato diseñado para tal fin. Es la aplicación cliente quien debe conocer de antemano la estructura de los objetos que serán retornados para poder accederla y proceder con los objetos según su necesidad.

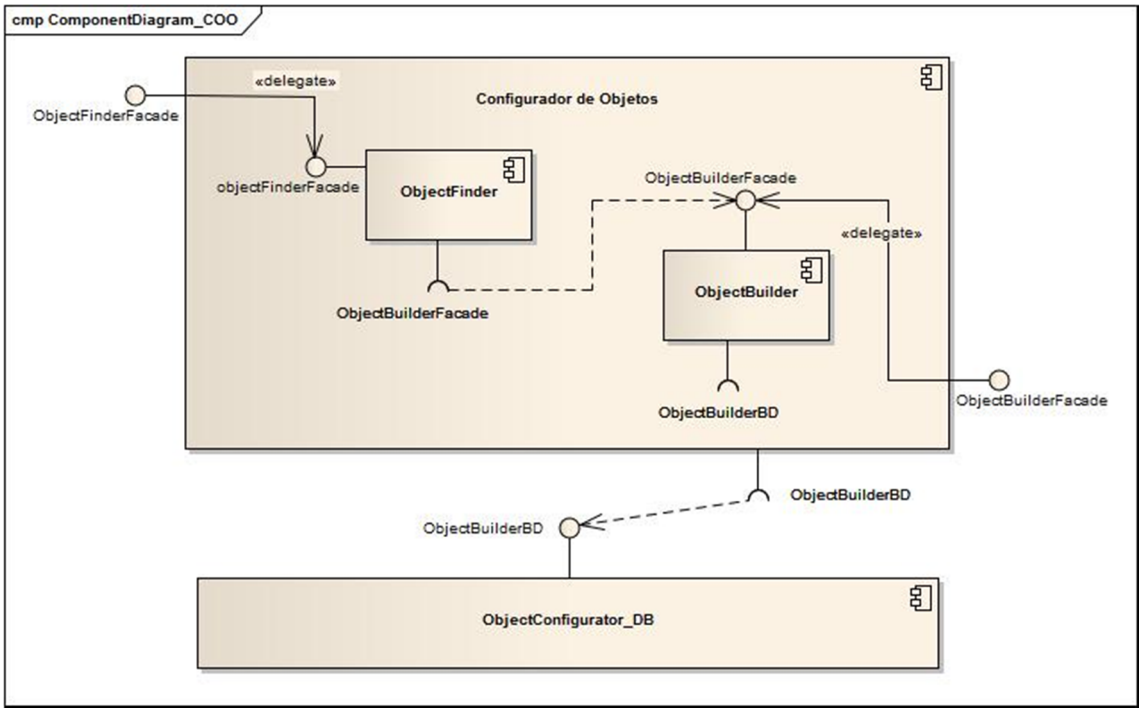


Ilustración 30: Diagrama de componentes de alto nivel del Configurador de Objetos.

Los objetos instanciados por el componente *ObjectBuilder* son una construcción que combina el patrón *Value Object* (Fowler 2002) y *Composite* (Gamma, y otros 1995). Siguiendo el patrón *Composite* se logra que los objetos retornados tengan atributos que bien pueden ser de tipos de datos nativos y también puedan tener atributos tipados con el mismo tipo de dato del objeto que los encapsula. Una estructura compuesta de este tipo tiene la capacidad de representar cualquier entidad del negocio. Siguiendo el patrón *Value Object* se logró que los objetos simplemente tengan responsabilidades de acceso a sus datos; estos objetos no basan su igualdad en la identidad sino en la igualdad de sus atributos. No ahondaremos en la explicación de los patrones de diseño en los que se basó la solución debido a que su documentación y ejemplificación es extensa en la literatura (Fowler 2002) (Gamma, y otros 1995).

Como anotación adicional acerca de estos objetos que se relacionan a un conjunto de criterios que permiten su selección se debe decir que si bien permiten representar cualquier entidad de negocio, su intención no es sustituir los modelos de dominio específicos de cada producto de software; el nivel de abstracción con el que se construyó esta estructura busca ofrecer un servicio que pueda ser usado por todas las aplicaciones independiente de las clases de su modelo de dominio; es responsabilidad de la aplicación cliente conocer la estructura de lo que en el Configurador de Objetos matricularon como objetos y posteriormente realizar la traducción a los objetos propios de su dominio.

Para finalizar la exposición del configurador de objetos se deben enunciar las siguientes ventajas concretas en comparación con un enfoque tradicional de construcción en base de datos:

- La creación de columnas en la tabla de decisión consiste en la modificación de un archivo de Excel. No se necesita ejecutar una sentencia DDL (Data Definition Language) como sucede en una solución mediante un modelo relacional en base de datos. Si el negocio requiere un nuevo criterio de selección la solución se reduce a poblar la tabla de decisión con el nuevo criterio y garantizar que la aplicación cliente proveerá dicho criterio.

- Un cliente nuevo debe generar una tabla de decisión nueva lo cual consiste simplemente en crear una tabla de Excel con sus **criterios propios**. Esto elimina el *Gap Semántico* que se crearía en una solución de base de datos al tener columnas con nombres genéricos.

- El algoritmo en el cual se basa el BRMS para evaluar la coincidencia de criterios se especializa en buscar un resultado a partir de un conjunto de hechos dados. El algoritmo, llamado algoritmo de RETE, tiene como principal característica que no evalúa aquello que no debe evaluar. En una solución de base de datos es inevitable que crecer el número de columnas (en caso de que estas hagan parte de los criterios de búsqueda) y el número de registros afecte el rendimiento de la búsqueda; el algoritmo de RETE, por su parte, es independiente del número de reglas que existan y del número de condiciones que tenga una regla dada.

La desventaja principal que presenta el componente de búsqueda de objetos a partir de un conjunto de criterios dados se encuentra en el consumo de memoria RAM de la máquina donde se ejecute el componente. *JBoss Drools* es una solución que al implementar el algoritmo de RETE provee soluciones muy eficientes en términos de tiempos de respuesta sacrificando memoria para poder obtener resultados en tiempos muy cortos.

5.3.1.2. Fases de introducción y enlace

Todas las características del producto de software Venta en Línea Autos que se apoyen en Venta en Línea para habilitar su variabilidad tienen como fase de introducción la fase de implementación y como fase de enlace a la fase de tiempo de ejecución.

La aplicación Configurador de Objetos utiliza, como ya se explicó, tablas de decisión de *Drools* para habilitar la variabilidad; la variabilidad en este caso debe ser vista desde dos perspectivas:

- En tanto que la capacidad que tiene el cliente de seleccionar un conjunto de objetos u otro a partir de unos criterios dados. Es decir, que el conjunto de objetos seleccionados pueda variar.

- En tanto que la capacidad que tiene el cliente de seleccionar un conjunto de objetos dados a partir de un conjunto de criterios variable. Es decir, que el conjunto de criterios de selección pueda variar.

El mecanismo de variación utilizado – tablas de decisión – no se encuentra en el catálogo de mecanismos de variación de Bosch (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) y se considera que puede ser una entrada válida para el catálogo. Este mecanismo debe ser embebido en el patrón de variabilidad denominado *Entidades Múltiples Coexistentes*, también definido por Bosch (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002).

Desde el punto de vista en el cual lo que necesita es variabilidad en el conjunto de criterios de selección, la fase de introducción es la fase de implementación. Suponga que se necesita incrementar el número de criterios con dos nuevos criterios definidos por la unidad de negocio. El Configurador de Objetos soporta un crecimiento hasta 20 criterios genéricos sin necesidad de intervenir el componente internamente. Hasta la fase de implementación el componente acepta la introducción de nuevas variantes.

Ahora, en el caso en el cual las variantes son los objetos a seleccionar, la fase de introducción también es la fase de implementación. La tabla de decisión en el BRMS Drools es un artefacto ejecutable por lo que ingresar reglas –como filas- en la tabla de decisión es una actividad de implementación la cual constituye la inserción de nuevas variantes en tanto que nuevos objetos candidatos a ser retornados por el componente.

La fase de enlace definitivamente es la fase de tiempo de ejecución pues es en ese tiempo en el cual en el BRMS determina que para unos criterios entregados por el software cliente se deben, efectivamente, evaluar esos criterios y obtener un conjunto de objetos a ensamblar y retornar.

5.3.2. Selección de datos del vehículo

Para la implementación de las tres características relacionadas con la selección de datos de vehículo que constituyen puntos de variación en el producto de software Venta en Línea se hizo uso del configurador de objetos. Debido a que ya se ha realizado una descripción de este componente, en cada característica que se utilice se mostrará solamente las características particulares de la tabla de decisión respectiva.

5.3.2.1. Selección del sistema de gas

Para la selección del sistema de gas se hizo uso del componente Configurador de Objetos. La tabla de decisión configurada hace uso de un solo criterio el cual corresponde al tipo de vehículo. La Ilustración 31 muestra esta tabla de decisión en la cual se hace uso de un solo criterio para crear reglas que a partir del tipo de vehículo (codificado para evitar errores de digitación) determinan la lista de combos de gas a mostrar en pantalla. En el caso particular de la ilustración, para un automóvil (tipo de vehículo igual a 1) se crearon cuatro reglas, para seleccionar cuatro combos de gas diferentes; para una camioneta (tipo de vehículo igual a 2), la tabla de decisión muestra la creación de 6 reglas, correspondientes a un único combo definidos por el negocio para dicho tipo de vehículo. La razón para la que exista el registro 6 veces en la tabla de decisión obedece a la ejecución de un caso de prueba en el cual se deseaba verificar que el mismo combo se presentaba en pantalla tantas veces como estuviera matriculado en la tabla de decisión.

	B	C	D
3		RuleSet	co.com.techandsolve.objectsconfigur
4			
5		Import	java.math.BigDecimal;import co.com.
6		Variables	
7		Notes	
8		RuleTable PlantillasCotizacion	
9		CONDITION	ACTION
10		consultaObjeto:ConsultaObjeto	
11		criterioComparacion.criterio1==$param	consultaObjeto.addIdObjetoSeleccionado($param);
12	Combos de GAS	Clase de vehiculo	Escoger articulo del kit
13	COMBO 1 AUTOMOVIL	1	4850
14			4800
15	COMBO 2 AUTOMOVIL	1	4804
16			4808
17	COMBO 3 CAMIONETA	2	4559
18			4559
19			4559
20	COMBO 4 CAMIONETA	2	4559
21			4559
22			4559

Ilustración 31: Tabla de decisión del Configurador de Objetos para la selección de la lista de combos de gas de acuerdo con el tipo de vehículo que se cotiza.

Debido a que la necesidad de variabilidad se ubica en la selección de diferentes combos, en esta sección no documentaremos la estructura compuesta que describe al combo de gas como tal. Estos combos de gas, como bien se explicó en la sección anterior, están descritos en una estructura que obedece a los patrones *Value Object* y *Composite* los cuales se identifican con un número único de objeto, ubicado en la columna más a la derecha de la tabla de decisión.

Dado que este punto de variación se implementó con el Configurador de Objetos, su fase de introducción es la fase de implementación y la fase de enlace es la fase de tiempo de ejecución.

5.3.2.2. Selección del nivel de blindaje

El punto de variación de selección de combos de blindaje también fue implementado haciendo uso del Configurador de Objetos. En este caso se utilizaron dos criterios: la clase de vehículo (codificada nuevamente para evitar errores de digitación y favorecer la estandarización) y los años de uso del vehículo. El objeto a seleccionar corresponde con un combo de blindaje que tiene un nombre y un valor determinado. Como en el caso anterior, la estructura de este objeto no será detallada en este trabajo debido a que allí no se encuentra la necesidad de variación en el producto de software Venta en Línea.

La Ilustración 32 da cuenta de la existencia de esta tabla de decisión.

	B	C	D	E
3		RuleSet	co.com.techandsolve.objectsconfigurator.core.objectsfinder.businessrule	
4				
5		Import	java.math.BigDecimal;import co.com.techandsolve.objectsconfigurator.cc	
6		Variables		
7		Notes		
8		RuleTable PlantillasCotizacion		
9		CONDITION	CONDITION	ACTION
10		consultaObjeto:ConsultaObjeto		
11		criterioComparacion.criterio1=="$param"	criterioComparacion.criterio2 == "$param"	consultaObjeto.addIdObjetoSeleccionado($p aram);
12	Combos de blindaje	Clase de vehiculo	Años de uso	Escoger articulo del combo
13	NIVEL BASICO II (ESPESOR 13 MM)	1	-1	5018
14			0	4563
15			1	4563
16			2	4563
17			3	5012
18			4	4563
19			5	4600
20			6	4563
21			7	4563
22			8	4563
23			9	4563
24			10	4563

Ilustración 32: Tabla de decisión para la selección de un grupo de combos de blindaje de acuerdo al tipo de vehículo a cotizar y su antigüedad.

Nuevamente, al ser una característica implementada mediante el Configurador de Objetos, su fase de introducción es la fase de implementación y su fase de enlace es la fase de tiempo de ejecución.

5.3.2.3. Selección de accesorios

La selección los diferentes combos de accesorios dependiendo del tipo de vehículo a cotizar en Venta en Línea se implementó, al igual que las características anteriores, con el Configurador de Objetos. En este caso, solo hacía falta un criterio, el cual es el tipo de vehículo a cotizar. Dado un tipo de vehículo se puede conocer el conjunto de combos de accesorios que se deben presentar en pantalla para la selección por parte del usuario final del sistema.

La estructura de un combo de accesorios describe su valor en pesos y los diferentes accesorios que contiene. Así como en las características anteriores, la estructura de este objeto específico no se detallará debido a que no afecta la exposición de la variabilidad de esta característica. La Ilustración 33 es un ejemplo de la tabla de decisión para la selección de combos de accesorios por tipo de vehículo.

9		CONDITION	ACTION
10		consultaObjeto:ConsultaObjeto	
11		criterioComparacion.criterio1==$param	consultaObjeto.addIdObjetoSeleccionado($param);
12	Combos de accesorios	Clase de vehiculo	Escoger articulo del combo
13			4550
14			4606
15	COMBO#1	1	4611
16			4550
17	COMBO#2	2	4611
18			4550
19			4606
20			4611
21			4606
22	COMBO#3	3	4611
23			4550
24			4606
25			4611
26			4606
27	COMBO#4	4	4611

Ilustración 33: Tabla de decisión para la selección de los combos de accesorios según el tipo de vehículo a cotizar en el producto de software Venta en Línea.

Esta característica también tiene su fase de introducción en la fase de implementación y su fase de enlace en la fase de tiempo de ejecución. Esto debido a que hereda las características de fases de introducción y enlace del componente Configurador de Objetos.

5.3.3. Selección de los productos a cotizar

La selección de los planes a cotizar y presentar en pantalla constituyó la primera motivación para la creación del componente Configurador de Objetos. La implementación de esta selección se realizó mediante la creación de reglas de selección de planes en una tabla de decisión. Esta tabla de decisión para la implementación de este punto de variación es la tabla más grande y compleja de todo el producto de software; de los veinte criterios de evaluación que ofrece el Configurador de Objetos, la tabla de decisión de esta característica utiliza 19. Los criterios utilizados para la selección de productos son los siguientes:

1. Si el contrato es colectivo y la forma de tarifación es por *tasa única*.
2. Si el vehículo es de una marca y línea específica.
3. Si el vehículo circula en una zona específica del territorio nacional.
4. Si el vehículo es financiado.
5. Si el usuario final tiene bonificación mayor a un porcentaje determinado.

6. Si el contrato es colectivo entonces para ciertos números de póliza se asocian algunos planes especiales.
7. Si el usuario final desea asegurar si vehículo contra daño y hurto.
8. Si al usuario final le interesa en un nivel alto, medio o bajo la protección de terceros en caso de un accidente.
9. Si al usuario final le interesa proteger el vehículo para cubrir gastos por arreglos en caso de choque.
10. Si al usuario final le interesa contar con un facilitador asesor en el sitio del accidente.
11. Si al usuario final le interesa elegir un taller de su preferencia en caso de necesitar algún arreglo.
12. Si el usuario final realiza con frecuencia viajes de más de 420 kilómetros.
13. Si al usuario final le interesa que le realicen inspecciones del vehículo a domicilio.
14. Si al usuario final le interesa contar con un facilitador SURA en el sitio del accidente.
15. Si al usuario final le gustaría contar con un vehículo de reemplazo en caso de que no pueda utilizar el propio en caso de un accidente.
16. Si el vehículo es un automóvil o una motocicleta.
17. Si el vehículo es de servicio público o particular.
18. Criterio técnico de la Gerencia de Automóviles.
19. Criterio técnico de la Gerencia de Automóviles.

Debido a la extensión de esta tabla de decisión es imposible mostrarla como ilustración tal como se ha realizado en las características previas. Se debe documentar, adicionalmente, que así como esta tabla de decisión es la más extensa en cuanto a cantidad de criterios, la complejidad de la estructura objetual compuesta que describe los planes a cotizar es la estructura más compleja que se encuentra matriculada en el Configurador de Objetos. Esta estructura describe todas las coberturas del plan y para cada cobertura indica sus valores de deducible y mínimo deducible; adicionalmente agrupa los diferentes valores agregados de cada cobertura y de todo el plan, de tal forma que puedan ser comparados gráficamente en pantalla por el usuario final.

Como los demás puntos de variación del producto de software en estudio que se implementaron en el Configurador de Objetos, éste tiene su fase de introducción en la fase de implementación y su fase de enlace en el tiempo de ejecución.

5.3.4. Presentación de coberturas

El punto de variación correspondiente a la presentación de coberturas fue diseñado e implementado obedeciendo el patrón *Strategy* (Gamma, y otros 1995). Este patrón tiene como propósito "*definir una familia de algoritmos, encapsular cada uno de éstos y hacerlos intercambiables, permitiendo que un algoritmo varíe independientemente de los clientes que lo usan"* (Gamma, y otros 1995). En este caso específico, los diferentes algoritmos corresponden con las diferentes formas de presentar una cobertura o un conjunto de ellas en la pantalla. La Ilustración 34 muestra un diagrama de clases con el patrón *Strategy* implementado para dar solución al punto de variación correspondiente con esta característica.

Este mecanismo de variación es documentado por Bosch y categorizado como *Implementación de Componentes Coexistentes* (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002). En esta misma referencia, a este mecanismo de variación se le asigna la fase de introducción en la fase de diseño y su fase de enlace en el tiempo de ejecución. La razón por la cual su fase de introducción es el diseño radica en que el mecanismo está abierto para nuevas variantes solamente hasta la etapa de diseño del producto y en ninguna etapa posterior adicional.

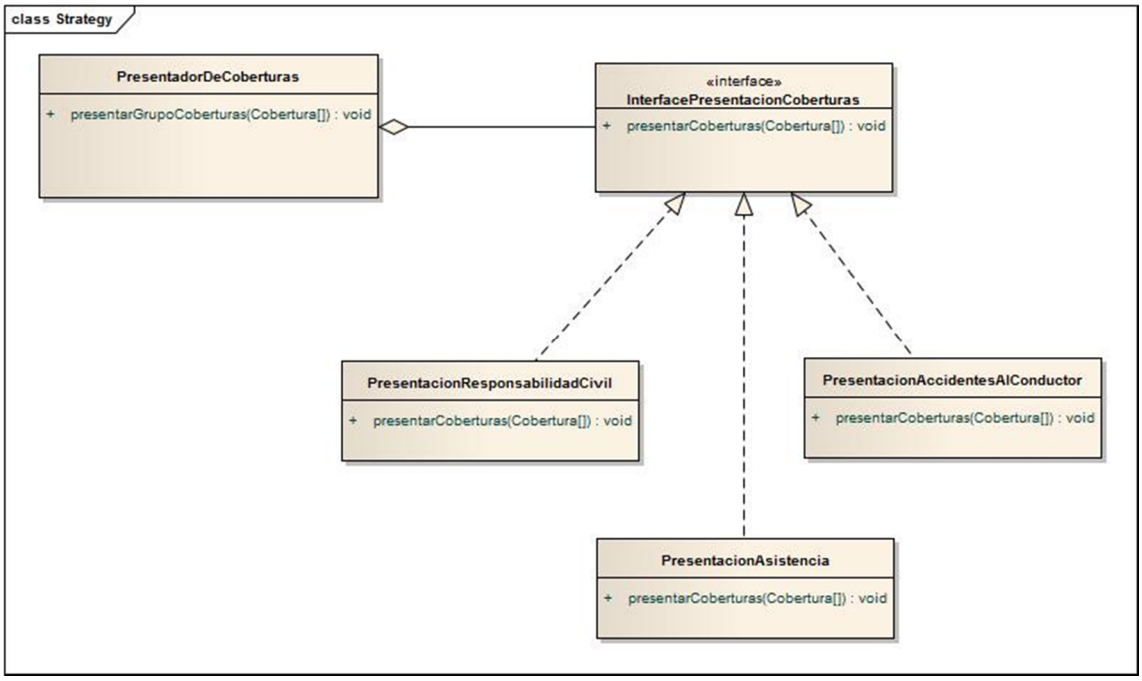


Ilustración 34: Diagrama de clases del patrón *Strategy* para la presentación de coberturas en el producto de software Venta en Línea.

5.3.5. Tarifación

5.3.5.1. Tarifación por tasa única y tarifación por tasas de cobertura

El producto de software Venta en Línea hace uso de un componente de software de Suramericana S.A. mediante el cual se obtiene el valor de prima de una póliza de seguro de automóviles. Para hacer uso de este componente es necesario indicar por parámetro si se trata de una tarifación tasa única o de tarifación por coberturas. El componente, internamente, realiza la selección de la lógica a ejecutar para tarifar de una u otra manera.

Esta selección de comportamiento se realiza mediante sentencias de control de flujo – *if-else* – directamente en el código de la aplicación. Este mecanismo de selección es documentado por Bosch como *Condición en Variable* y pertenece al patrón *Entidad Variante* (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002).

Este mecanismo de variación acepta la inserción de nuevas variantes en tiempo de implementación por lo que es esta la fase de introducción de las diferentes variantes; dado que éstas son enlazadas en tiempo de ejecución es ésta su fase de enlace.

5.3.5.2. Cálculo de tasas de coberturas

La necesidad de variación en esta característica, como ya se describió previamente, se encuentra en:

- Posibilidad de inserción el cálculo de una nueva cobertura.
- Posibilidad de adición o eliminación de los factores que intervienen en el cálculo de la prima de una cobertura.

En este caso también se hizo uso de *Drools* en una solución de software desarrollada para efectuar estos cálculos. La solución de software fue diseñada para permitir la inserción de nuevas coberturas sin

intervención del núcleo del sistema; adicionalmente se diseñó para que, también sin modificar el núcleo, se pudiera adicionar, eliminar y combinar factores como el negocio lo necesitara.

Para satisfacer la necesidad de variación de factores, para cada cobertura se debe desarrollar una tabla de decisión por cada factor; todas estas tablas de decisión, por organización del sistema deben estar agrupadas en un mismo archivo de Excel.

	RuleTable PrimaPuraRC		
	CONDITION	CONDITION	ACTION
	Cobertura	ListaCaracteristicasAutos	
	(strOpcion == "$param") && (producto == "RC")	$fechaConsulta: feTarifa && eval($fechaConsulta.after(new SimpleDateFormat("yyyyMMdd").parse($1))) && eval($fechaConsulta.before(new SimpleDateFormat("yyyyMMdd").parse($2)))	insert(new PrimaPura("RC", "$param"));
Base pricing rules	**opcion**	**fealta,febaja**	**Prima**
Prima Pura	1	"20120101","20121231"	59769,00
	2		96000,00
	3		107692,00
	12		120563,00
	4		159531,00
	13		187704,00
	5		208509,00
	14		224182,00
	6		237621,00
	7		250170,00
	15		266349,00

Ilustración 35: Tabla de decisión para el factor Prima Pura de la cobertura Responsabilidad Civil del producto de software Venta en Línea de Suramericana S.A.

	RuleTable Factor Clase RC		
	CONDITION	CONDITION	ACTION
	ListaCaracteristicasAu	ListaCaracteristicasAutos	
	strTipoVehiculo == "$param"	$fechaConsulta: feTarifa && eval($fechaConsulta.after(new SimpleDateFormat("yyyyMMdd").parse($1))) && eval($fechaConsulta.before(new SimpleDateFormat("yyyyMMdd").parse($2)))	insert(new FactorClase("RC", "$param"));
Base pricing rules	**clase**	**fealta,febaja**	**Factor**
AUTOMOVILES	1	"20120101","20121231"	0,7554
CAMPEROS Y PICKUPS	2		1,2526
MOTOS < 125 CC	3		0,4320
MOTOS 125 - 250 CC	4		0,1964
MOTOS < 250 CC	5		0,4720
MOTOCARRO	6		0,4468
REMOLQUES	7		
UTILITARIOS	8		1,4379
CAMION MEDIANO	9		1,4536
CAMION PESADO	10		3,1544

Ilustración 36: Tabla de decisión para e factor Clase de Vehículo de la cobertura Responsabilidad Civil del producto de software Venta en Línea de Suramericana S.A.

La Ilustración 35 y la Ilustración 36 son muestras de dos de las cinco tablas de decisión que corresponden a los factores de la cobertura Responsabilidad Civil del producto de automóviles de Suramericana S.A. ¿Qué sería entonces necesario si se quisiera definir un nuevo factor para esta cobertura? En tiempo de implementación se reduce a la creación de una nueva tabla de decisión que determine para los características del vehículo qué factor seleccionar.

Estos factores no tendrían sentido si no son incluidos en una fórmula de tarifación. Se recuerda una vez más que la variación en esta fórmula de variación también hace parte de las necesidades de variación de esta característica. Para darle solución a este punto, en el producto de software Venta en Línea se implementó un archivo de reglas escritas en el lenguaje de *Drools* llamado *MVEL* el cual define una regla en la que se ejecuta la tarifación de cada cobertura para la que se hayan obtenido factores. Así las cosas, si nace un nuevo factor en una cobertura ya existente, la modificación de la fórmula de tarifación consiste en la modificación de la regla correspondiente con dicha cobertura. En el caso de una cobertura completamente nueva, se deben implementar las diferentes tablas de decisión para cada uno de los factores y se debe agregar una regla al archivo de reglas de tarifación para que, a partir de los factores, obtenga el valor de prima de esa nueva cobertura.

La Ilustración 37 muestra una de las reglas escritas en el archivo de reglas de tarifación. En esa ilustración se muestra la regla correspondiente a la tarifación de la cobertura Pérdida Total por Daños del producto de Automóviles. Por razones de confidencialidad, Suramericana S.A. solicita ocultar detalles de la fórmula de tarifación.

```
8  rule "RN_cotizacion_PTD"
9      when
10         coberturas: ListaCoberturas()
11         $cobertura : Cobertura(producto == "PTD")
12         Tasa(producto == "PTD", $tasa :  valor)
13         FactorZona(producto == "PTD",$zona : valor)
14         FactorGasto(producto == "PTD", $factorGasto : valor)
15         FactorDeducible(producto == "PTD", $deducible : valor)
16         FactorGrupoTarifa(producto == "PTD", $grupoTarifa : factor)
17         Bonificacion(producto == "PTD", $bonificacion : valor)
18     then
19         CoberturaConTasa coberturaPTD = new CoberturaConTasa($cobertura.getStrCdGarantia()
20
21         float ptdTasa = ($[redacted]) / (1-$[redacted]) / (
22         //ptdTasa = ptdTasa * 100;
23         coberturaPTD.setBigdTasa(new BigDecimal(ptdTasa));
24         coberturaPTD.setProducto($cobertura.getProducto());
25         coberturas.addCobertura(coberturaPTD);
26     end
27
```

Ilustración 37: Regla de tarifación para la cobertura Pérdida Total Daños del producto de automóviles en el producto de software Venta en Línea de Suramericana S.A.

Las reglas de negocio, bien escritas en MVEL o en tablas de decisión, están abiertas para la inserción de variantes en tiempo de implementación y por este motivo la fase de introducción de las características que se apoyen en ellas para su construcción es la fase de implementación. De forma adicional, la decisión de ejecutar o no una regla se posterga hasta el tiempo de ejecución por lo que es la fase de tiempo de ejecución la que se determina como fase de enlace de estas características.

Con respecto al patrón de variabilidad de esta característica, para cada cobertura existen muchos factores que interfieren en su fórmula de tarifa. Por su parte, para una cotización de pueden cotizar más de una cobertura. En sendos casos el patrón de variabilidad corresponde a *Entidades Múltiples y Coexistentes*.

5.3.6. Determinación de capacidad de deducción por nómina

La determinación de capacidad de deducción por nómina fue implementada en el sistema de nómina de Suramericana S.A. Este desarrollo tuvo como entregable un procedimiento almacenado en la base de datos de nómina la cual es invocado por el producto de software Venta en Línea. Dicha invocación se hace mediante la implementación de un *ExternalSystemAdapter*, elemento importante de la arquitectura del producto y que se describió en la sección 3.3.2.

Realizando una inspección en este procedimiento almacenado, se encuentra que la decisión de ejecutar el cálculo para empleados o para asesores está basado en una sentencia de control de flujo *if-else* en el código fuente.

Dado a que para un empleado solamente se ejecuta una sola de las opciones (empleado o asesor), este punto de variación obedece al patrón *Entidad Variante.* Adicionalmente, el mecanismo de variación es determinado como *Condición en Variable* dado que este es equivalente a la selección de comportamiento mediante sentencias de control de flujo en el código fuente.

6. Ejecución de la medición

En esta sección se realizará una representación cuantitativa de la variabilidad del producto de software Venta en Línea de Suramericana S.A. Esta representación cuantitativa se posibilita debido a que ya se han expuesto conceptualmente las características del producto que constituyen puntos de variación y se han descrito detalles de su implementación los cuales denotan aspectos importantes para la medición como su fase de introducción y su fase de enlace. En la sección 6.1 se realizará la obtención de la cifra concreta mientras que en la sección 6.2 se realizará un análisis del resultado.

6.1. Obtención de medida de variabilidad

El objetivo de la medición de la variabilidad es conocer un valor cuantitativo para la variabilidad de un producto. Esto significa que mediante una medida concreta se pueda describir qué tan preparado está un producto de software para aceptar nuevas variantes en puntos que de antemano se sabía que presentarían variación.

La ***Ecuación 1*** permite obtener la magnitud de variabilidad de una característica con fase de introducción i y fase de enlace j. Esta magnitud se obtiene a partir de la medición del área que cubre dicha combinación de fases en el espacio de variabilidad propuesto por Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002).

Para realizar la medición de variabilidad de un producto de software utilizando esta ecuación, es necesario en primera instancia conocer el conjunto de los diferentes pares [fase de introducción, fase de enlace] de las características variantes del producto de software. Posteriormente, es necesario saber en qué proporción están presentes en el sistema cada uno de estos pares; conocer estas proporciones permitirá obtener la medición total de variabilidad del producto como lo especifica la ***Ecuación 2*****.**

$$Vt = \sum_{i=1}^{n} Vi * Pi$$

Ecuación 2: Magnitud de Variabilidad Total ($\boldsymbol{Vt}$) calculada a partir de la medida Variabilidad de las Características con un conjunto de fase de introducción y enlace único ($\boldsymbol{Vi}$) ajustada por el porcentaje de presencia ($\boldsymbol{Pi}$) de dicho conjunto en el sistema.

Para obtener la variabilidad total de un producto de software es necesario entonces:

1. Conocer los diferentes conjuntos de fases de introducción y enlace de las características variables del producto.

2. Obtener el porcentaje de presencia de estos conjuntos en todo el producto.

3. Obtener la medida de variación para cada una de las combinaciones [fase de introducción, fase de enlace] mediante la ***Ecuación 1***.

4. Realizar el cálculo de variabilidad total mediante la ***Ecuación 2*** a partir de los resultados obtenidos en los pasos anteriores.

El paso 1 ya se ha realizado en la sección anterior. La Tabla 10 muestra los pares [fase de introducción, fase de enlace] de cada característica variante del sistema. La Tabla 11 indica las proporciones de las diferentes combinaciones de fases de introducción y enlace de las características variantes del producto de software Venta en Línea.

Fase de introducción	Fase de enlace	Proporción	Porcentaje
Implementación	Tiempo de ejecución	7 de 8	87.5%
Diseño	Tiempo de ejecución	1 de 8	12.5 %

Tabla 11: Porcentajes de combinación de las fases de introducción y las fases de enlace de las características con variabilidad en el producto de software Venta en Línea de Suramericana S.A.

El marco de referencia para la medición de variabilidad sugiere la representación gráfica de estas proporciones de tal forma que se tenga una representación adicional a la cuantitativa, de la variabilidad del producto. Esta representación gráfica corresponde a la ubicación de puntos en el espacio de variabilidad en el cual se relacionan en el eje x del plano las diferentes fases de introducción y en el eje y las diferentes fases de enlace. En dicho espacio de variabilidad se da cuenta de las combinaciones de fases de las características del producto en estudio mediante la ubicación de puntos en las diferentes combinaciones; el tamaño de los puntos debe representar visualmente en qué proporción se presentan esas combinaciones. La Ilustración 38 muestra el espacio de variabilidad del producto de software en estudio con dos puntos graficados, uno de mayor dimensión correspondiente a la presencia de la combinación ***[implementación, tiempo de ejecución]*** en un 87.5% de las características variantes y otro para la combinación ***[diseño, tiempo de ejecución]*** presente en un 12.5% de las características variantes del producto de software Venta en Línea.

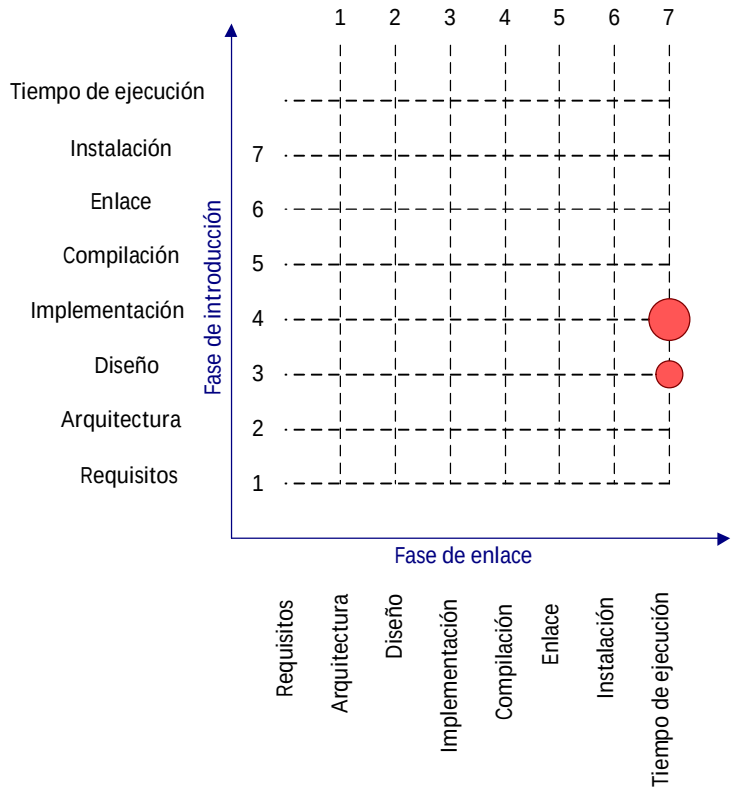


Ilustración 38: Espacio de variabilidad para el producto de software Venta en Línea de Suramericana S.A.

Habiendo obtenido los diferentes conjuntos de fase de introducción y fase de enlace, habiendo obtenido el porcentaje de presencia de cada combinación diferente en el producto y habiendo representado gráficamente estas combinaciones y porcentajes en el espacio de variabilidad, el paso que se debe realizar a continuación es la obtención de la medida de variabilidad de cada par ***[fase de introducción, fase de enlace]*** mediante la ***Ecuación 1***. La Tabla 12 muestra los resultados de la Ecuación 1 para cada combinación de fase de introducción y fase de enlace de los mecanismos de variación encontrados en el producto estudiado. En esta tabla se puede ver tanto el valor de variabilidad absoluta como el valor de la variabilidad normalizada (porcentual).

Fase de introducción	Fase de enlace	Medición de variabilidad absoluta	Medición de variabilidad normalizada
Implementación (i=4)	Tiempo de ejecución (b=7)	20	81.63%
Diseño (i=3)	Tiempo de ejecución (b=7)	16.5	67.34%

Tabla 12: Medida de variabilidad para cada combinación de fase de introducción y fase de enlace.

Cada uno de los valores de variabilidad obtenidos y expuestos en la Tabla 12 corresponde a un Vi de la ***Ecuación 2*** mientras que cada porcentaje expuesto en la Tabla 11 corresponde al respectivo Pi de la misma ecuación. En vista de que en este punto se cuenta con todos los parámetros para ejecutar el cálculo de variabilidad para todo el producto se procede con la ejecución de éste mediante la ***Ecuación 2.*** A continuación se expone el cálculo realizado:

$$Vt = \sum_{i=1}^{n} Vi * Pi = (20)*(0.875) + (16.5)*(0.125) = \mathbf{19.5625} \cong \mathbf{80\%}$$

Finalmente, y debido a que la variabilidad máximo posible en el marco de referencia de Bosch es de 24.5, se procede con la representación del resultado en proporción a este resultado máximo. El resultado absoluto de variabilidad para el producto Venta en Línea de Suramericana S.A. es de 19.5625 correspondiente a un resultado relativo normalizado del 80%.

En la sección posterior se expondrá un análisis detallado de este resultado.

6.2. Análisis de resultados

El análisis de los resultados obtenidos en la sección anterior se realizará en los siguientes frentes:

- Análisis de las causas que generan el resultado obtenido.
- Evaluación de los posibles resultados ante variaciones en los mecanismos de variación.

Primero que todo se debe aclarar qué significa el 80% obtenido como medida de la variabilidad del producto de software Venta en Línea de Suramericana S.A; para esto es necesario exponer qué significan los valores extremos de la medición.

En el marco de referencia de Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002), los valores extremos de la medición de variabilidad son 2% y 100%. El menor valor significa que el producto de software en estudio se encuentra preparado para cambios en los puntos de variación existentes en etapas muy tempranas de su ciclo de vida. Si un producto de software obtiene este valor en la medición de variabilidad entonces todos sus puntos de variación, sin excepción, son implementados con mecanismos de variación que aceptan nuevas variantes únicamente en etapa de requisitos y se materializan como tal en etapa de arquitectura. Por el contrario, si para un producto de software se obtiene una medida de variabilidad igual al 100% entonces todos sus puntos de variación, sin excepción, fueron implementados con mecanismos de variación los cuales aceptan la definición de nuevas variantes incluso en el momento de la instalación del producto y la selección de éstas sucede en tiempo de ejecución.

En el primer caso, los mecanismos de variación se caracterizan por tener fases de introducción y de enlace en etapas muy tempranas del ciclo de desarrollo - requisitos y arquitectura respectivamente – lo que quiere decir que el artefacto ejecutable del producto de software no será variable como pieza de software en ejecución sino únicamente en sus artefactos de documentación. Aquello que se ejecuta, en este escenario, no contiene en sí características variantes.

En el segundo caso, por el contrario, los mecanismos de variación sí permiten que el artefacto ejecutable del producto de software sea variable. Al permitir definir nuevas variables hasta el momento de la instalación y sobretodo realizar una selección en tiempo de ejecución, estos mecanismos hacen del producto de software un producto completamente variable pues las diferentes variantes son independientes entre sí y pueden pasar a ser parte del producto - en tanto que software en ejecución - de forma transparente.

Lo que se acaba de describir permite concluir que la medida de variabilidad obtenida mediante el marco de referencia de Bosch describe qué tan variable es un producto en la medida en que éste tiene la capacidad de aceptar nuevas variantes y seleccionar estas variantes para su ejecución en momentos cercanos al tiempo de ejecución del producto de software. Si un producto de software tiene capacidades de postergar la introducción y selección de variantes entonces es más variable pues el producto de software aporta verdadero valor en tiempo de ejecución.

Habiendo expuesto esto, se hace necesario establecer un parámetro para poder comparar cuándo un resultado de la medida de variación es o no aceptable. En el orden de ideas expuesto, la variabilidad en un producto de software cobra verdadero valor en tanto en que las variantes – tanto en su introducción como en su enlace – constituyen piezas de software ejecutable. Así las cosas, la combinación de tiempos más temprana para que una variante haga parte de un producto de software es la de diseño-compilación. Se podría considerar la etapa más temprana para la fase de enlace la del *tiempo de implementación* sin embargo, existen mecanismos que habilitan la transformación de artefactos de diseño en piezas de software ejecutables; con respecto a la fase de enlace, como se enunció en secciones previas, la etapa más temprana en la que se esperan enlaces de variantes es la de *compilación*. Para esta combinación de fase de introducción y fase de enlace se puede ejecutar el cálculo de variabilidad mediante la ***Ecuación 1*** obteniendo como resultado un valor absoluto igual a 7.5, el cual corresponde a un valor relativo normalizado igual al 30.6% de variabilidad. En este trabajo se ha determinado este valor como el mínimo aceptable para considerar a un producto como variable. Adicionalmente, para el espacio de variabilidad entre el 30.6% y el 100% se determinarán tres secciones correspondientes a niveles bajo, medio y alto de variabilidad como lo muestra la Ilustración 39. Estas tres secciones corresponden de la división en tres secciones iguales del espacio comprendido entre el valor mínimo y el máximo de variabilidad. Productos de software que obtengan una medición por debajo del 30.6% se consideran entonces como ***no variables***.

La división de este espacio de variabilidad en tres secciones de igual rango obedece a una aproximación ad-hoc en lugar de obedecer a una división de acuerdo a datos históricos de mediciones de variabilidad. Se sugiere la ejecución de mediciones adicionales, para otros productos software de la compañía, utilizando este mismo marco de referencia, con el propósito de calibrar dichos segmentos del espacio de variabilidad.

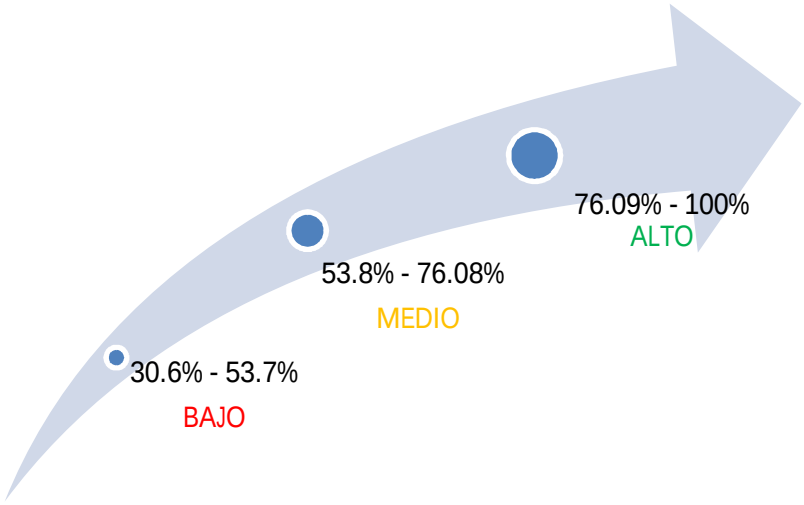


Ilustración 39: Clasificación de porcentajes de variabilidad en el espacio de variabilidad de Bosch para el producto de software Venta en Línea.

Dado que el producto de software estudiado obtuvo un resultado del 80% para la medición de variabilidad y de acuerdo con la clasificación propuesta para el espacio de variabilidad de Bosch, Venta

en Línea se caracteriza por tener un ALTO nivel de variabilidad. Significando esto que el producto se encuentra preparado para introducir cambios – nuevas variantes – en los puntos de variación indicados en secciones anteriores de forma rápida, dado que no es necesario modificar el núcleo del producto para la adición de dichas variantes.

Niveles altos de variabilidad reducen el *tiempo al mercado*[11] de nuevas características de un producto de software. Esto se debe a que un nivel alto de variabilidad se logra a partir de la identificación adecuada de puntos de variación y su implementación mediante mecanismos de variación que permiten incluir y enlazar variantes sin modificar el núcleo del producto lo cual habilita la agilidad en el desarrollo de nuevas variantes. Si una aplicación tiene un nivel alto de variabilidad significa que llevar una nueva variante al producto de software se hace de manera más rápida que en un producto con un nivel de variabilidad inferior. Por lo anterior se puede deducir que la inclusión de una nueva variante en alguno de los puntos de variación identificados en el producto de software Venta en Línea se realiza de forma rápida. Este documento no tiene especificado dentro de su alcance la medición de tiempos de inclusión de nuevas variantes para corroborar esta hipótesis; la ejecución de un estudio que permita corroborar esta hipótesis, es sugerido como trabajo futuro.

Es interesante conocer los motivos por los cuales el producto de software Venta en Línea alcanza una medida de variabilidad del 80%. Para hacer este análisis es necesario recordar que las características con variabilidad en el producto de software Venta en Línea fueron implementadas mediante tres mecanismos de variación denominados: *Tabla de decisión en BRMS*, *Condición en Variable E Implementación de componente coexistente* (ver Tabla 9); estos mecanismos obtuvieron una medida relativa normalizada de variabilidad igual a 81.63% para los dos primeros y 67.34% para el último. Los dos primeros mecanismos obtienen una medida de variabilidad igual debido a que su combinación de fase de introducción y fase de enlace es la misma para ambos. Los dos primeros mecanismos de variación, de acuerdo con los rangos descritos por la Ilustración 39 favorecen enormemente la variabilidad del producto dado que los puntos de variación implementados con esos mecanismos son clasificados como altamente variables. Sin embargo, es necesario hacer un análisis un poco más detallado de cada uno de ellos y observar su relación con el impacto en la variabilidad del producto.

Los dos mecanismos de variación que aportan en mayor medida al resultado obtenido son aquellos que tienen por fase de introducción el tiempo de implementación y como fase de enlace el tiempo de ejecución. Sin embargo existe una diferencia muy importante entre ellos y consiste en la visibilidad que brindan sobre el punto de variación. Implementar un punto de variación con el mecanismo *Condición en Variable* significa que se están utilizando sentencias de control de flujo para la selección de variantes. La Ilustración 40 muestra un ejemplo de este mecanismo de variación en el producto Venta en Línea; en este ejemplo, se toma una decisión de comportamiento basado en si el modo de tarifación de la póliza es de *tasa única*. Al ser el código fuente un artefacto del software que solamente conoce y comprende con detalle el equipo de desarrolladores, la visibilidad para el resto del equipo del manejo de esta variabilidad es mínima. Adicionalmente, los términos en los que se especifica el mecanismo pueden llegar a no ser legibles para un equipo de desarrollo nuevo que requiera hacer inclusión de nuevas variantes sobre esta característica del producto; en este caso, por ejemplo, puede no ser muy claro de dónde proviene el valor de la variable *str_excepciones* sobre la cual se evalúa su valor para proceder con el comportamiento variable.

[11] Término conocido en la industria como *Time to market*

```
else if ( str_SNNuevaTarifacion.equals( SI ) && str_excepciones.equals( TASA_UNICA ) ) {
    try {

        SearchRateService srs = new SearchRateService();
```

Ilustración 40: Ejemplo de mecanismo de variación *Condición en Variable* en el producto Venta en Línea

En la Ilustración 35, por su parte, se muestra cómo se implementó una característica variante mediante el mecanismo de variación *Tabla de decisión en BRMS*. Lo verdaderamente importante de este mecanismo es que obedece también a una implementación con la estructura *si-entonces* pero en este caso, el artefacto de software no manifiesta explícitamente la existencia de estos condicionales. Una tabla de decisión como mecanismo de variabilidad posee la cualidad de expresar la variabilidad en un lenguaje de más alto nivel que el código fuente que finalmente se compilará y ejecutará; una tabla de decisión, sin embargo, continúa siendo el artefacto construido por el equipo de desarrollo, pues será el BRMS quien al final traducirá las reglas en código ejecutable. Al ser escrita la tabla de decisión en un lenguaje de alto nivel mucho más cercano al negocio, no es el equipo de desarrollo el único integrante del equipo de trabajo quien conoce cómo se implementa la variabilidad sino todo el equipo completo está en la capacidad de conocer las decisiones que tomará el producto de software bajo unas condiciones dadas. Esto significa, en conclusión, que una tabla de decisión aporta el mismo nivel de variabilidad con la gran virtud de no sacrificar la visibilidad de la misma.

En este punto es posible hacer una comparación con la cualidad sistémica *rendimiento* la cual indica generalmente, en los sistemas que la requieren, que se deben obtener tiempos de respuesta en la ejecución de las funcionalidades especificadas menores a un parámetro establecido. En estos casos siempre se evalúan aspectos como el ancho de banda de la conexión que se utilizará para la invocación de procedimientos remotos, el lenguaje de programación en el que se construirá la solución, el tipo de mensajes que se enviarán entre sistemas y demás aspectos que se acotan para poder lograr los tiempos de respuesta solicitados. En esta evaluación es común comparar el rendimiento de un lenguaje orientado a objetos con el rendimiento de un lenguaje de script y la respuesta a la que se llega en algunos casos es que aquél lenguaje-plataforma de programación que permite lograr menores tiempos de respuesta desfavorecen la mantenibilidad del producto de software pues ejecutar actividades de mantenimiento evolutivo, correctivo o preventivo será más difícil que si se hace sobre otro más orientado a favorecer estas cualidades. Con la variabilidad y específicamente con el mecanismo *Condición en variable* sucede lo mismo: favorece enormemente la variabilidad pero desmejora considerablemente la mantenibilidad del producto de software.

Ahora bien ¿Qué sucedería con la variabilidad del producto de software si los mecanismos de variación no hubieran sido los seleccionados sino otros diferentes? Existen dos escenarios claros que podrían modificar los resultados obtenidos. El primer escenario posible consiste en que todos los mecanismos de variación tengan fases de introducción y enlace cuya combinación generen un resultado inferior al obtenido para el producto en estudio. En este escenario la mejor combinación que logra un resultado inferior al obtenido para Venta en Línea es el par ***[diseño, instalación]*** la cual obtendría un resultado absoluto de 13.5 equivalente a un resultado relativo normalizado del 55.1% el cual está alejado en un 12.14% del menor resultado obtenido para el producto Venta en Línea con el mecanismo de variación *Implementación de Componentes Coexistentes* el cual tiene fase de introducción el diseño y la fase de enlace el tiempo de ejecución.

La Ilustración 41 muestra las dos combinaciones de fases de introducción y enlace que más se aproximarían al menor resultado de Venta en Línea como producto de software. La combinación adicional se trata de la fase de introducción *arquitectura* con fase de enlace *tiempo de ejecución*. Para

esta combinación se obtiene un valor absoluto de variabilidad igual a 12 correspondiente a un valor relativo normalizado del 48.9%.

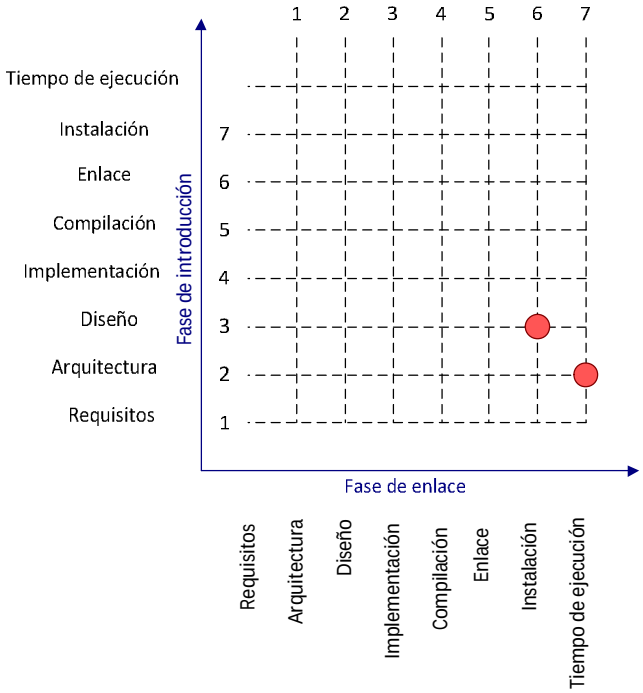


Ilustración 41: Combinaciones de fase de introducción y de enlace que generarían el mayor valor de variabilidad menor que el obtenido para el producto Venta en Línea.

En ambas combinaciones, los resultados obtenidos (55.1% y 48.9%) clasificarían al producto en estudio con un nivel medio de variabilidad, de acuerdo con la clasificación de porcentajes propuesta en la Ilustración 39.

Con respecto a los mecanismos de variación que cumplan con dichas combinaciones de fases de introducción y enlace, es necesario observar que en la categorización de mecanismos de variación de Bosch (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) no se encuentra ningún mecanismo caracterizado con estas dos fases. Adicionalmente, en otros proyectos de software liderados por el autor de este trabajo tampoco se encuentran presentes mecanismos de variación que cumplan con estas características. Por tal motivo no es posible documentar en este documento qué propiedades del producto de software en estudio lo harían menos variable de lo que ya es.

El segundo escenario de resultados de variabilidad diferentes es más interesante porque consiste en encontrar mecanismos de variación que hagan más variable al producto en estudio de lo que es actualmente. Siguiendo el mismo procedimiento del escenario anterior, se deben encontrar las combinaciones de fases de introducción y enlace que generen una medición de variabilidad mayor al 80%. La Ilustración 42 muestra cuáles son las diferentes combinaciones de fases de introducción y enlace que deben tener los mecanismos de variación para que el producto de software Venta en Línea fuera más variable.

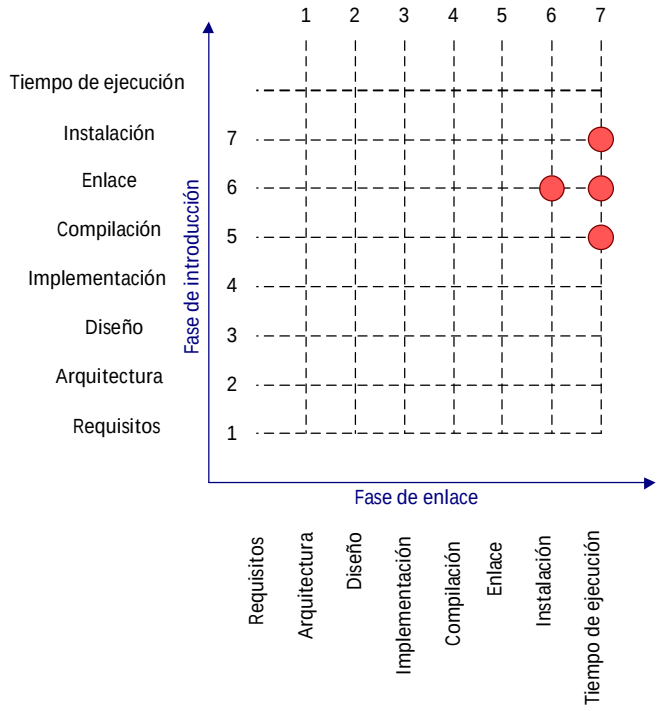


Ilustración 42: Combinación de fases de introducción y enlace de los mecanismos de variación que harían más variable al producto de software Venta en Línea.

En el inventario de mecanismos de variación de Bosch tampoco se encuentran mecanismos que se caractericen por tener las fases de introducción y enlace necesarias en este escenario. Análisis realizados por el autor de este documento no arrojan ningún resultado que permita identificar mecanismos de variación que cumplan con estas combinaciones de fases de introducción y enlace. Estas combinaciones, sin embargo, deben continuar presentes en el espacio de variabilidad pues corresponden a las fases de transformación de todo producto de software y por tanto es posible que existan mecanismos aún no implementados que posean estas características; dichas transformaciones son expuestas en la Ilustración 5.

Este análisis de resultados finaliza resumiendo que encontrar mecanismos de variación que desmejoren la variabilidad en estudio del producto de software Venta en Línea no llevaría a resultados tan interesantes como encontrar mecanismos de variación que mejoren o incrementen la variabilidad del producto de software estudiado. Adicionalmente, para mejorar el nivel de variabilidad obtenido se requiere de la existencia de mecanismos de variación que aún no han sido documentados en las referencias bibliográficas examinadas. Finalmente, se debe resaltar la presencia de un mecanismo de variación no documentado en el inventario de mecanismos tomado como referencia, el cual ha sido nombrado *Tabla de decisión en BRMS* y cuenta con fase de introducción la implementación y con fase de enlace el tiempo de ejecución logrando con esto que el producto alcance un nivel alto de variabilidad pues la medida de variabilidad de este mecanismo es alta y la mayor parte de las características evaluadas fueron implementadas en el producto utilizando dicho mecanismo.

7. Conclusiones

En este trabajo se estudió un aspecto importante de los productos de software denominado variabilidad. Este concepto fue definido concretamente como la capacidad que tiene un activo de software de ser extendido, modificado o personalizado de forma planeada (Bosch y Jaring, A taxonomy and hierarchy of variability dependencies in software product family engineering 2004) . El estudio de la variabilidad en este trabajo consistió en la exposición de los conceptos principales de la variabilidad como puntos de variación, variantes, fases de introducción y enlace de una variante, mecanismos y patrones de variación y las diversas dimensiones en las que se puede estudiar la variabilidad como la variabilidad espacial, temporal, interna y externa. Posteriormente se procedió con la exposición del marco de referencia de Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) el cual permite la obtención de una medida cuantitativa de la variabilidad de una línea de producto de software. Si bien este marco de referencia está documentado para que sea usado en la obtención de una medida cuantitativa de la variabilidad de una LPS, en este trabajo fue utilizado para la medición de variabilidad de un producto de software de Suramericana S.A. en lugar de una LPS por dos motivos principales: (a) se considera que las características del marco de trabajo se adaptan a la necesidad de medición de variabilidad en un producto;(b) la organización no posee un enfoque orientado a LPS en el que la variabilidad es un elemento de primera clase. El uso de dicho marco en el producto de software denominado Venta en Línea arroja las conclusiones que se describen a continuación.

La variabilidad, como la arquitectura de software, siempre se encuentra presente en los productos de software. Independiente de la forma como se maneje, modele o represente, en un producto de software, éste generalmente deberá tomar decisiones acerca de cómo comportarse de acuerdo a condiciones bien de negocio –variabilidad externa- o condiciones técnicas – variabilidad interna- y generalmente existirá la necesidad de extender o modificar las funcionalidades de acuerdo a las necesidades anticipadas del negocio . La existencia de la toma de decisiones con respecto a la selección de comportamientos o variación en los comportamientos núcleo habilita la existencia de la variabilidad en un producto específico y por ende se habilita su estudio y análisis. De aquí que la variabilidad no constituye una característica exclusiva de las familias de productos sino que, como ya se ha mencionado, existe en productos independientes.

Para poder estudiar la variabilidad de un producto, se hace necesario realizar una taxonomía de los diferentes tipos de variabilidad presentes en dicho producto y determinar con claridad cuál de esos tipos de variabilidad se desea estudiar. Esta clasificación y selección permite aislar la variabilidad de interés de aquellas cuyo estudio no agrega valor a los objetivos que se buscan alcanzar. En este trabajo se identificó como variabilidad objetivo aquella que es espacial, externa y correspondiente con el comportamiento de los componentes que implementan el proceso de venta de seguros sobre Internet de Suramericana S.A. Otra alternativa era el estudio de la variabilidad en el proceso mismo de venta de seguros, la cual se descartó debido a que el sistema estudiado no fue desarrollado mediante una metodología BPM o con tecnologías o herramientas de desarrollo de aplicaciones orientadas por procesos y a que las referencias bibliográficas no se concentran en el estudio de variabilidad en procesos de negocio.

Con respecto al método utilizado –marco de referencia de Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) - para obtener la medida cuantitativa de variabilidad del producto de software estudiado, se puede concluir que fue posible su utilización a pesar de haber sido diseñada para la medición de variabilidad en líneas de producto de software en lugar de productos específicos. No fue necesario realizar algún acondicionamiento especial al espacio de variabilidad o de los indicadores utilizados para ejecutar la medición sobre el producto estudiado. Se sugiere la documentación explícita de lo que en este documento se denominó ***Ecuación 2*** pues fue

necesario hacer un proceso de ingeniería inversa a partir de los cálculos realizados en (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002).

Los resultados arrojados por la medición de variabilidad en el producto de software Venta en Línea determinan un 80% de variabilidad del producto. Debido a ausencia de datos históricos de mediciones de variabilidad en otros productos similares no es posible determinar con bases concretas la magnitud de este resultado. Debido a esto, se dividió de manera ad-hoc el espacio numérico de posibles resultados en tres rangos iguales correspondientes a los valores Alto, Medio y Bajo. De acuerdo con la clasificación de los resultados planteada, el producto de software Venta en Línea posee un nivel de variabilidad Alto específicamente en el tipo de variabilidad estudiada.

El resultado obtenido del 80% de variabilidad del producto objeto de estudio, se considera muy buen resultado, especialmente teniendo en cuenta que durante la construcción del producto no consideró la variabilidad de manera explícita tal como se hace en un enfoque LPS. Este resultado obtenido se debe a que el 87.5% de los puntos de variación del producto de software estudiado se implementaron haciendo uso de dos mecanismos de variación que cuentan con fase de introducción el tiempo de implementación y con fase de enlace el tiempo de ejecución. El resultado de la medición de variabilidad se incrementa debido a que estas fases constituyen fases tardías del proceso de desarrollo lo cual se ve recompensado en el marco de referencia de Bosch. Uno de estos mecanismos ya había sido documentado por este mismo autor (Bosch, Svahnberg y van Gurp, On the Notion of Variability in SPL 2002) y denominado *Condición en Variable;* el 25% de las variaciones fueron debidas a este mecanismo. El segundo mecanismo de variación, por otra parte, no se encontró documentado como tal y constituye un aporte al estudio de variabilidad; este mecanismo se documentó en este trabajo como *Tabla de Decisión en BRMS*; el 62.5% de los puntos de variación se implementaron con este mecanismo.

Es muy importante resaltar que si bien los dos mecanismos cuentan con las mismas fases de introducción y enlace y en esta medida aportan en igual medida cuantitativa al resultado final de variabilidad del producto, el mecanismo *Tabla de decisión en BRMS* se diferencia enormemente porque da mayor visibilidad al artefacto de software que finalmente se ejecutará: Mientras que la comprensión del código fuente está al alcance del equipo de desarrollo o de personal con habilidades específicas de programación, las variantes desarrolladas mediante *Tablas de decisión en BRMS* son visibles a cualquier participante del proyecto debido a que están escritas en un lenguaje mucho más cercano al negocio que el código fuente "tradicional", sin perder la cualidad de ser un artefacto ejecutable del producto de software que las contiene.

Es interesante observar como la selección de variantes mediante sentencias de control de flujo *if-else* aporta en gran medida a la variabilidad de un producto. Al comenzar la ejecución de esta medición se consideraba de antemano que esta práctica haría menos variable a un producto pero se han encontrado resultados que contradicen aquella percepción inicial. De aquí es necesario resaltar que en el manejo de variabilidad, de forma similar a como se hace en la disciplina de arquitectura de software, se deben sacrificar algunas cualidades sistémicas para satisfacer algunas otras. En el caso específico del mecanismo de variación *Condición en Variable* se debe reconocer que si bien la variabilidad se verá favorecida enormemente, la visibilidad de las variantes disminuirá proporcionalmente. Es apropiado indicar que un producto de software que decida utilizar este mecanismo de variación asume conscientemente el riesgo de perder visibilidad de los puntos de variación y sus respectivas variantes lo cual resultará perjudicial en la medida en que pase el tiempo y el equipo de desarrollo cambie, como es natural en todos los equipos de este tipo. Artefactos como modelos UML que den fe de la existencia de los puntos de variación implementados con el mecanismo *condición en variable* son sugeridos para mitigar este riesgo.

De acuerdo con lo anterior, se concluye que cualquier sistema de software está preparado para variar, dado que todo producto de software, finalmente, es ejecutado a partir de su código fuente y no de los demás artefactos que soportan la existencia de dicho código. Ahora bien, el marco de referencia de Bosch recompensa de la misma manera a un producto de software que varía mediante sentencias de

control de flujo *if-else* de una manera desorganizada y poco legible y a un producto de software que decide utilizar este mismo mecanismo pero de una manera elegante y disciplinada. Esto constituye uno de las principales objeciones, sino la principal, con respecto al marco de referencia de Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002), pues puede resultar que un producto de software con activos de software legados que implementan comportamientos muy acoplados y poco cohesivos pero que varían mediante sentencias de control de flujo es altamente variable.

Para la exposición de los puntos de variación del producto de software estudiado se inspeccionaron referencias bibliográficas que arrojan como los principales mecanismos de modelado de variabilidad a FODA (Kang, y otros 1990) y a OVM (Lauenroth y Pohl 2005). Sinnema (Sinnema y Deelstra 2007), por su parte, realiza un trabajo comparativo de mecanismos de modelado muy claro, el cual también fue revisado en la inspección realizada. El modelado de características permite obtener una perspectiva visual, en primera instancia, de los aspectos comunes y variables de un producto de software. El resultado obtenido en el modelado de características para el producto de software estudiado no fue referenciado en este trabajo dada la dificultad de navegar, comprender y leer debido a su gran tamaño. Se sugiere como trabajo futuro profundizar en la inspección del estado del arte acerca del modelado de variabilidad en escalas industriales de tal forma que dé soluciones a los problemas de manipulación y legibilidad de grandes modelos.

8. Trabajo futuro

Se intuye que el estudio de la variabilidad en el contexto de procesos de negocio implementados en un BPMS o mediante alguna otra tecnología puede llevar a resultados interesantes debido a que las unidades de negocio especifican procesos que no son estáticos y que de acuerdo a ciertas condiciones se ejecutan ciertas actividades de negocio u otras. El estudio podría orientarse al descubrimiento y categorización de patrones y mecanismos de variación desde ese punto de vista de un producto de software. La ejecución de un estudio de este tipo se plantea como trabajo futuro.

Con respecto al nivel *alto* de variabilidad obtenido para el producto de software estudiado, es posible plantear la hipótesis acerca de que los tiempos al mercado de nuevas variantes en los puntos de variación identificados y descritos en este documento son menores a aquellos que tomaría implementar las mismas variantes si los mecanismos de variación hubieran sido diferentes (de menor nivel de variabilidad). Debido a que está por fuera del alcance de la realización de este trabajo la verificación de dichos tiempos al mercado, se plantea como trabajo futuro la verificación y comparación de estos tiempos para la corroboración de esta hipótesis.

En vista de que el marco de referencia de Bosch (Bosch y Jaring, Representing Variability in Software Product Lines A Case Study 2002) no ofrece una clasificación en rangos de los posibles resultados, los niveles establecidos de manera Adhoc en este trabajo, son un referente de partida que debe ser refinado. Se sugiere como trabajo futuro la calibración de estos rangos de variabilidad por medio de estudios similares que realicen la medición en otros aplicativos de Suramericana S.A. y que permitan el refinamiento de los rangos establecidos.

Adicionalmente a lo anterior, se sugiere una modificación al marco de referencia de Bosch. Esta modificación consiste en tener en cuenta la existencia de factores de corrección tal como la visibilidad de los mecanismos de variación. Estos factores permitirían diferenciar de forma cuantitativa los mecanismos *Condición en variante* y *Tabla de decisión en BRMS*.

Finalmente, se plantea como trabajo futuro el abordaje de la variabilidad en los diferentes productos de software de seguros de Suramericana S.A. tanto para su representación cuantitativa – como se ha

realizado en este trabajo – como para considerar la seria posibilidad de implantar en la compañía un enfoque de desarrollo de productos mediante una LPS. Las características de los productos –de seguros y de software- actuales y las necesidades de las unidades de negocio dejan entrever desde un análisis inicial superficial, y a la luz de las referencias estudiadas, que cuenta con todas las características para que un enfoque de este tipo sea implantado como una estrategia prometedora de la Gerencia de Desarrollo de TI.

9. Referencias

Bachman, Felix, y Paul Clements. *Variability in Software Product Lines.* SEI, 2005.

Bass, Len, y Rick Kazman. *Architecture-Based Development.* SEI Library, 1999.

Bosch, Jan. *Design & Use of Software Architectures - Adopting and Evolving a Product Line.* Addison-Wesley, 2000.

Bosch, Jan, Mikael Svahnberg, y Jilles van Gurp. *On the Notion of Variability in SPL.* Proceedings. Working IEEE/IFIP Conference on Software Architecture, 2002.

Bosch, Jan, y Michael Jaring. «A taxonomy and hierarchy of variability dependencies in software product family engineering.» Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004.

Bosch, Jan, y Michael Jaring. «Representing Variability in Software Product Lines A Case Study.» (Lecture Notes in Computer Science) 2379 (2002).

Brownsword, Lisa, y Paul Clements. *A Case Study in Successful Product Line Development.* CMU/SEI-96-TR-016, 1996.

Bu-Qing, Cao, Li Bing, y Xia Qi-Ming. «A Process-Driven and Ontology Based Software Product Line Variability Modeling Approach.» Grid and Cooperative Computing, 2009. GCC '09. Eighth International Conference on, 2009.

Buschman, Frank, Regine Meunier, Hans Rohnert, y Peter Sommerland. *Pattern Oriented Software Architecture, Volume 1: A System of Patterns.* Willey & Sons, 1996.

Chapin, Ned, Joanne E Hale, Khaled Md. Khan, Juan F Ramil, y Wui-Gee Tan. «Types of software evolution and software maintenance.» (Journal of Software Maintenance and Evolution: Research and Practice) 13 (2001).

Clements, Paul, y Linda Northrop. *Software Product Lines: Practices and Patterns.* Addison-Wesley, 2002.

Coplien, James, Daniel Hoffman, y David Weiss. «Commonality and Variability in Software Engineering.» (IEEE Software), nº 6 (1998).

Czarnecki, K, y M Antkiewicz. «FeaturePlugin: Feature Modeling Plug-In for Eclipse.» Vancouver: OOPSLA'04 Eclipse Technology eXchange (ETX) Workshop, 2004.

Dhungana, D. «Integrated Variability Modeling of Features and Architecture in Software Product Line Engineering.» Automated Software Engineering, 2006. ASE '06. 21st IEEE/ACM International Conference on , 2006.

Drools, Business Logic integration Platform. s.f. http://www.jboss.org/drools (último acceso: 17 de Febrero de 2012).

Elfaki, Abdelrahman Osman, Somnuk Phon-Amnuaisuk, y Chin Kuan Ho. «Modeling Variability in Software Product Line using First Order Logic.» (Seventh ACIS International Conference on Software Engineering Research, Management and Applications) 2009.

FeatureIDE. s.f. http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/ (último acceso: 18 de 04 de 2012).

Foote, Brian, Neil Harrison, y Hans Rohnert. *Pattern Languages of Program Design 4 (Software Pattern Series).* Addison-Wesley, 2000.

Fowler, Martin. *Patterns of Enterprise Application Architecture.* Addison-Wesley, 2002.

Gamma, Erich, Richard Helm, Johnson Ralph, y John Vlisides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Reading, Massachusets: Addison Wesley Longman, 1995.

Griss, M.L, J Favaro, y M d'Alessandro. «Integrating feature modeling with the RSEB.» (Software Reuse, 1998. Proceedings. Fifth International Conference on) 1998.

Hegde, Mahesh, y Janet K. Wall. «Effective Application Modernization with Business Rule Management System.» 2009.

Heilmann, Christian. *Beginning JavaScript with DOM Scripting and Ajax.* New York, New York: Springer-Verlag, 2006.

International Organization for Standarization. *ISO/IEC 25000.* 2005.

Jacobson, Ivar, Martin Griss, y Patrick Jonsson. *Software Reuse. Architecture, Process and Organization for Business Process.* New York, NY: ACM Press, 1997.

Kang, Kyo, Sholom Cohen, James Hess, William Novak, y A Spencer Peterson. «Feature-Oriented Domain Analysis (FODA) Feasibility Study.» Pittsburgh, 1990.

Kruchten, Philippe. «Architectural Blueprints - The "4+1" View Model of Software Architecture.» (IEEE Software), nº 6 (1995).

Lauenroth, Kim, y Klaus Pohl. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer, 2005.

Martínez-Ruiz, T, F García, M Piattini, y J Münch. «Modelling software process variability: an empirical study.» (IET Software) 2010.

Northrop, Linda. *Software Product Lines Essentials.* s.f.

Robak, S, y A Pieczynski. «Employing fuzzy logic in feature diagrams to model variability in software product-lines.» Engineering of Computer-Based Systems, 2003. Proceedings. 10th IEEE International Conference and Workshop on the, 2003.

Sarinho, V.T., y A.L. Apolinario. «Combining feature modeling and Object Oriented concepts to manage the software variability .» Information Reuse and Integration (IRI), 2010 IEEE International Conference on, 2010.

Sinnema, M., y S. Deelstra. «Classifying variability modeling techniques.» (Elsevier) 2007.

Ye, H., y H. Liu. «Approach to modelling feature variability and dependencies in software product lines.» Software, IEE Proceedings - Volume: 152 , Issue: 3, 2005.