Modelo Conceptual para la Gestión de Variabilidad Temporal de Familias de Productos

Resumen: *Dada la naturaleza de diversos eventos que modifican la información de una familia de productos, la gestión de ciclo de vida de productos o PLM (Product Lifecycle Management) requiere de soluciones robustas que registren los cambios ocurridos y que gestionen la información de familia de producto. El objetivo del trabajo es presentar un modelo conceptual para gestionar la variabilidad en el tiempo de una familia de productos, manteniendo la integridad y la consistencia de los modelos de datos involucrados. Esta conceptualización sirve como herramienta para referenciar las versiones de una familia de productos, recuperar información y representar los cambios. La propuesta se aplicará a dos modelos de representación de productos de diferentes dominios, tales como: PRONTO (PRoduct ONTOlogy) y el Modelo de Características (Feature Model - FM).*
Palabras Claves: *Variabilidad, Gestión de Versiones, Familia de Productos, Ontologías.*

Abstract: *Given the nature of several events that modify product family information, PLM (Product Lifecycle Management) requires robust solutions to record changes and manages product family information. The object of this article is to present a conceptual model to manage product family variability over time, maintaining integrity and consistency in the involved product data models. This conceptualization serves as a tool for referencing product family versions, retrieving information, and representing change events. The proposal will be applied to two product models which belong to different domains such as: PRONTO (Product Ontology) and Feature Model (FM) with the aim of representing changes and versions.*
Keywords: *Variability, Version Management, Product Family, Ontologies.*

María S. Sonzini[1,2], Marcela Vegetti[1], Horacio Leone[1]

[1]INGAR, Instituto de Desarrollo y Diseño (CP3000) Tel. +54 0342 4535568.
Avellaneda 3657, Santa Fe, Argentina
[2]Universidad Nacional de La Rioja (CP5300)
Luis M. de la Fuente S/N, La Rioja, Argentina
Mail: ssonzini@santafe-conicet.gob.ar - mvegetti@santafe-conicet.gob.ar - hleone@santafe-conicet.gob.ar

INTRODUCCIÓN

La gestión del ciclo de vida de producto (PLM) es una actividad que permite gestionar de modo holístico la información relacionada a un producto durante su creación, desarrollo, madurez, hasta su final (Matsokis and Kiritsis, 2010). Es importante gestionar la información de los productos en todas las fases de su ciclo de vida, debido a que un cambio en el mismo durante una determinada etapa, podría propagarse y afectar la consistencia e integridad de la información en el resto de las fases. Los sistemas de información que implementan PLM requieren soluciones para representar los modelos de datos de producto de forma consistente, con el fin de compartir y operar con otras organizaciones, procesos, etapas del ciclo de vida y stakeholders (personas o entidad concernidas en las actividades). Durante su ciclo de vida es común que los productos sufran cambios en su diseño y/o configuración debido a cuestiones comerciales, de mercado o avances tecnológicos. Esta capacidad de cambio de un producto recibe el nombre de variabilidad y permite que un producto sea extendido, transformado, personalizado o configurado para su utilización en un dominio particular (Asikainen et al., 2006).

Pohl et al. (2006) sostienen que es fundamental hacer una distinción entre Variabilidad Temporal y Variabilidad Espacial. La Variabilidad en el Tiempo es definida como "la existencia de diferentes versiones válidas de un producto en diferentes instantes", este tipo de variabilidad denota la evolución definiendo puntos de variación que ayudan a mantener el control del impacto de pequeños cambios. La Variabilidad Espacial es definida como "la existencia de un producto en diferentes formas en un mismo instante". Es decir, abarca de manera simultánea el uso de distintas variantes de un producto que coexisten en un mismo instante de tiempo. Es importante mencionar que los métodos utilizados actualmente en la gestión de Variabilidad Espacial, no pueden aplicarse del mismo modo para la gestión de Variabilidad Temporal. Por esto, surge la necesidad de definir un mecanismo apropiado para administrar los cambios en el tiempo, que pueda ser aplicado conjuntamente con los modelos de representación de variantes existentes. De esta manera se podría gestionar simultáneamente ambos tipos de variabilidad.

Este artículo propone un modelo de datos para la gestión de versiones (Variabilidad Espacial) y la representación del cambio ocurrido en la información de productos durante su ciclo de vida (Variabilidad Temporal), a través de conceptos genéricos que pueden ser especializados, independientemente del modelo de productos y de administración de variantes utilizado. El artículo se organiza de la siguiente forma: la Sección 2 introduce el estado del arte acerca de la gestión de variabilidad en las dos dimensiones mencionadas. Luego, en la Sección 3, se definen los conceptos fundamentales sobre la Variabilidad Temporal, los cuales se integran en un modelo conceptual genérico y se describen los conceptos básicos de la propagación de los cambios y su clasificación. La Sección 4, se introduce un caso de estudio sencillo a fin de validar la propuesta. Finalmente, se presentan las conclusiones y trabajos futuros.

ESTADO DEL ARTE EN LA GESTIÓN DE VARIABILIDAD

Diversas investigaciones se han enfocado en la gestión de la variabilidad de familias de productos. Entre estos aportes se destacan las propuestas de Garcés et al. (2007), Männisto T. (2000), Estublier y Casallas (2005), Vegetti et al. (2011) y Kang et al. (2003), las cuales se enfocan en el análisis de los aspectos variables y en el desarrollo de estrategias para gestionar esas variaciones.

Para la gestión de Variabilidad Temporal en el dominio de la industria de software, Männisto T. (2000) propone una metodología para modelar la evolución de familias de productos, incluyendo mecanismos necesarios para la representación de datos con sus aspectos temporales. Estublier y Casallas (Estublier and Cassallas, 2005) introducen tres dimensiones ortogonales para la gestión de versiones: histórica, lógica y cooperativa; haciendo referencia a la evolución de un objeto en el tiempo, la coexistencia de múltiples variantes de un objeto y la cooperación con actividades de forma concurrente en un mismo instante de tiempo. En cuanto a las propuestas que gestionan la Variabilidad Espacial, tanto en dominios relacionados con las manufacturas industriales como en la industria del software, en este artículo se introducirán sólo los conceptos fundamentales de dos de ellas por cuestiones de espacio: PRONTO (Vegetti et al., 2011) y el Modelo de Características (Kang et al., 2003). Además, serán utilizadas con el fin de validar el modelo de gestión de Variabilidad Temporal propuesto en este artículo. PRONTO es una ontología que permite representar datos de productos en diferentes niveles de abstracción y en distintos dominios de la industria. Para ello, define una jerarquía estructural (SH - Structural Hierarchy) que representa de forma eficiente la información concerniente a las partes y componentes que participan en la manufactura de un producto final y una jerarquía de abstracción (AH - Abstraction Hierarchy) que permite la representación de información no estructural de productos en diferentes niveles de abstracción, así como la representación de procesos de agregación y desagregación de información entre estos niveles. La Fig. 1 introduce los conceptos definidos en PRONTO. La AH consta de 3 niveles: nivel de Familia (Family), nivel de Conjunto de Variantes (VariantSet) y nivel de Producto (Product). Estos tres niveles se relacionan entre sí mediante la asociación memberOf, indicando que las entidades comprendidas en un nivel inferior, son miembros de una instancia de abstracción de un nivel superior. La SH considera dos tipos de relaciones de estructuras, que se especializan como componentOf, para aquellas estructuras que relacionan al producto con sus partes componentes, y derivateOf, para aquellas estructuras que enlazan al producto con sus derivados constituyentes. Esto quiere decir que un producto puede tener otros productos como componentes o derivados siempre en el mismo nivel de abstracción, lo cual permite representar las diferentes listas de materiales de un producto, conocidas como BOM (Bill Of Materials).

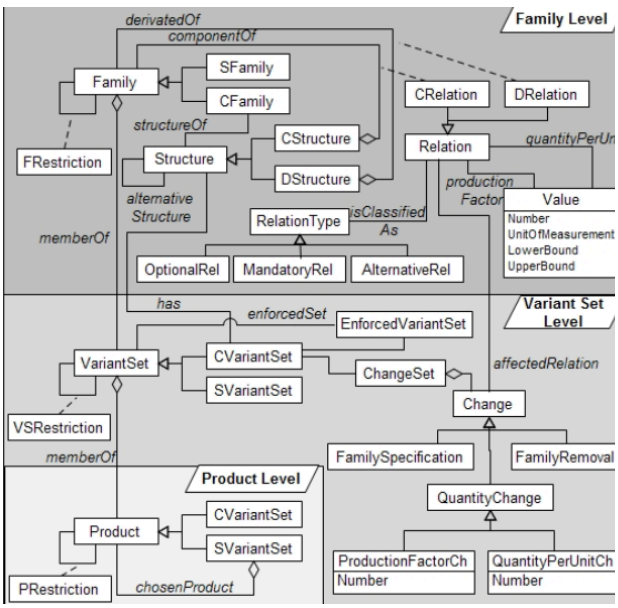

Fig.1: Modelo conceptual de la propuesta PRONTO.

Una familia representa un conjunto de productos que son similares en cuanto a su información y su estructura. Una familia puede ser compuesta (CFamily) o simple (SFamily), dependiendo de si tiene o no una estructura (Structure) asociada. Una familia compuesta

tiene al menos una estructura (Structure) asociada. Estas estructuras pueden ser de composición (CStructure), donde el producto es obtenido por el ensamblado de productos como partes componentes, o de descomposición (DStructure), donde es posible obtener varios productos derivados por medio del proceso de desensamblar un producto. Los componentes de una CStructure y los derivados de una DStructure se asocian a dicha estructura por medio de dos tipos de relaciones: CRelation y DRelation, respectivamente. En ambos casos, estas relaciones están vinculadas con las cantidades de cada componente o derivado que necesitan o derivan de la estructura involucrada (ver las asociaciones quantityPerUnit y productionFactor en la Fig. 1). Además, cada relación se clasifica como:

- Opcional (OptionalRel): el componente o derivado puede estar o no presentes en una estructura.

- Obligatoria (MandatoryRel): el componente o derivado debe estar presente en la estructura.

- Alternativa (AlternativeRel): el componente o derivado debe ser seleccionado de un conjunto de familias, pero sólo un miembro de dicho conjunto debe ser parte de la estructura.

Por su parte, el nivel de Conjunto de Variantes modela un subconjunto de miembros de una familia que comparten una estructura común y/o tienen características similares. Un conjunto de variantes puede ser simple (SVariantSet) o compuesto (CVariantSet) dependiendo de si es miembro de una Familia Simple o Compuesta. Un Conjunto de Variantes Compuesto, se asocia a la estructura de la familia que es compartida por todos los miembros del conjunto por medio de la relación Has. Un conjunto de variantes compuesto puede definir cambios (ChangeSet) sobre la estructura seleccionada de la familia. Los cambios, que pueden ser aplicados a una relación de la estructura (affectedRelation) se especializan en FamilyRemoval, FamilySpecification y QuantityChange, el cual a su vez, se especializa en quantityPerUnitCH y productionFactorCH para representar modificaciones en los valores asociados a quantityPerUnit y productionFactor de la relación afectada. FamilyRemoval representa la exclusión de la relación affected-Relation de la estructura. FamilySpecification está relacionado a los cambios en la elección de una sola alternativa de la estructura asociada.

Por su parte, el menor nivel de abstracción de la AH, el nivel de Producto, representa productos que tienen existencia física. Del mismo modo que los niveles anteriores, un producto puede ser simple (SProduct) o compuesto (CProduct). En este último caso, la relación ChosenProduct permite identificar los componentes o derivados concretos de un CProduct. A fin de controlar la construcción de jerarquías estructurales válidas, PRONTO especifica tres tipos de restricciones (FRestriction, VSRestriction, PRestriction) que permiten limitar las entidades que deben o no pueden estar juntas en una misma SH a nivel de Familia, Conjunto de Variantes y/o Productos.

Por su parte, Kang et al. (2002) desarrolla un método eficiente y efectivo para identificar y organizar los elementos variables y elementos comunes entre productos o líneas de producto de software, mediante la definición de un Modelo de Características (FM - Feature Model). Se considera una característica a un aspecto distintivo de un sistema, que es visible a todos los stakeholders. El FM posee una relación estructural denominada “consiste de” (consistOf), para representar una estructura lógica de características, donde las características padres se relacionan con las características hijas, formando una estructura de árbol. Los tipos de relaciones de un FM pueden clasificarse como sigue:

- Obligatoria: para indicar que una característica hija es requerida.

- Opcional: para indicar que una características puede ser requerida o no.

- Selección (OR): indica que al menos una característica hija debe ser seleccionada.

- Alternativa (XOR): indica que una única característica hija debe seleccionarse.

El FM posee dos tipos de restricciones. El primer tipo indica que una característica requiere la selección de otra. En tanto, el segundo tipo especifica que dos características no pueden ser parte de una misma característica padre.

A partir del análisis de las diferentes propuestas, aún existen cuestiones que merecen un estudio más profundo, tal como la gestión de dependencias entre puntos de variación y variantes, despertando así el interés de analizar los vínculos que existen (explícitas e implícitas) entre los diferentes elementos afectados (puntos de variación) a lo largo del ciclo de vida de un producto. Asimismo, no se han encontrado soluciones genéricas que aborden de forma simultánea las dos dimensiones en diferentes dominios de aplicación.

MODELO CONCEPTUAL PARA GESTIONAR LA VARIABILIDAD TEMPORAL - VERONTO

Teniendo en cuenta la necesidad de gestionar de manera conjunta las dos dimensiones de la variabilidad, esta sección presenta un modelo conceptual que permite la gestión de la Variabilidad Temporal y que puede ser extendido con modelos de productos que gestionan la Variabilidad Espacial, tal como los presentados en el punto anterior. En la Fig. 2 se introducen los conceptos fundamentales de la propuesta. Entre estos, el concepto ProductConcept, que permite representar la información del producto, cuyas versiones se van a gestionar. El concepto History, reúne el conjunto de versiones, representando la evolución de la información de producto en distintos instantes de tiempo. Cada versión temporal (Version) representa la configuración del producto que es válida en un período de tiempo comprendido desde el instante en que se crea la versión (indicado por DateTime), hasta el instante de creación de una nueva versión temporal. La relación firstVersion, entre las entidades History y Version, permite obtener la versión inicial de ProductConcept. La entidad Specification captura un conjunto de datos adicionales acerca de la versión generada, tal como los motivos de los cambios generados y la identificación de la persona responsable del registro de la versión. Además, cada versión reúne información concerniente a los eventos de cambios (ChangeEvent) que la generaron. Cada evento de cambio afecta a una entidad (Entity) del modelo de producto a través de una actividad (Activity).

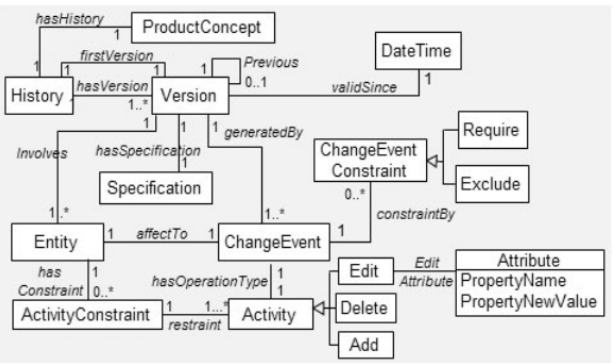

Fig. 2: Modelo Conceptual para la gestión de variabilidad temporal - VERONTO.

Una entidad es un concepto abstracto que debe ser extendido para representar los elementos del modelo de Variabilidad Espacial utilizado que pueden ser afectados por los cambios, por ejemplo: un compo-

nente, una relación, una restricción, un atributo, etc. Asimismo, un evento de cambio se relaciona con la actividad que ocasiona el cambio. Las actividades básicas consideradas por el modelo son: el agregado (Add) y la eliminación (Delete) de una entidad, así como la modificación (Edit) de un atributo (Attribute), identificando su nombre y el nuevo valor.

Dependiendo del dominio de aplicación, no todas las entidades pueden ser afectadas por todas las clases de actividades. Por tal motivo, se define el concepto de ActivityConstraint asociado a una entidad mediante la relación hasConstraint. Este nuevo concepto permite restringir (restraint) las operaciones Add, Delete o Edit, que pueden afectar a una entidad específica. En la propuesta, cada versión temporal, excepto la inicial, es relacionada con su predecesora a través de la asociación previous. De este modo, es posible reconstruir la versión actual de un producto a partir de su versión inicial.

En el contexto de la información de producto, una entidad forma parte de una estructura y, un cambio en ella podría afectar a otras entidades de dicha estructura, produciéndose una serie de nuevos eventos de cambios. Para representar esta situación, también conocida como propagación de cambios, se introduce el concepto ChangeEventConstraint. Este concepto permite especificar las restricciones asociadas a las reacciones que genera un evento de cambio sobre una entidad, es decir: si éste requiere (Require) o no (Exclude) la ocurrencia de nuevos eventos de cambios que afectan a otras entidades. Ecker et al (2004) propone una clasificación de entidades basada en cuatro tipos de reacciones ante la ocurrencia de eventos de cambio:

- Constantes: son aquellas entidades que no se ven afectadas por un cambio, es decir no absorben ni transmiten cambios a otras entidades, permaneciendo en su estado en todo momento.

- Absorbentes: son las entidades que absorben un número de cambios mayor de los que pueden generar y transmitir.

- Portadoras: son las entidades que pueden absorber el mismo número de cambios que el que transmiten a otras entidades.

- Multiplicadoras: entidades que generan un mayor número de cambios de los que pueden absorber, generando nuevos eventos de cambio.

En el ámbito de la industria, se requiere una gestión eficiente de los cambios con el fin de predecirlos y evitar que impacten de forma negativa en los costos, tiempos o recursos asignados durante el proceso de fabricación de un producto. Por esta situación, se considera que los conceptos descriptos permiten identificar los componentes afectados, cómo fueron modificados por un tipo de actividad, la causa de los cambios y el instante en que una versión comienza a ser válida. Además, es posible analizar la evolución de una familia de productos mediante el conjunto de versiones temporales que se generaron durante su ciclo de vida.

De este modo, la propuesta que se presenta en este trabajo, es un modelo genérico donde los conceptos ProductConcept, ChangeEvent, Entity y ActivityConstraint, pueden ser extendidos por los modelos de gestión de Variabilidad Espacial (PRONTO y FM). Logrando así, obtener la gestión simultánea de las dos dimensiones de variabilidad: Temporal y Espacial, de la información de una Familia de Productos.

MODELO CON VARIABILIDAD ESPACIO TEMPORAL DE UN PRODUCTO

En la presente sección se describe un ejemplo sencillo sobre un producto ficticio basado en la

telefonía celular, con el objetivo de representar los eventos de cambios y la propagación de los mismos en los modelos de productos: PRONTO y FM, expresándolos en sus extensiones temporales. Como ya se mencionó previamente, estos enfoques representan la variabilidad espacial de familias de productos. Para la fabricación de un teléfono celular, se requiere una configuración adecuada para el proceso productivo, generada durante la etapa de diseño. La configuración se integra a partir de varias partes componentes del producto. Para simplificar la explicación se consideran sólo 4 componentes: procesador, sistema operativo, cámara digital lateral y flash. De este modo, la versión inicial de la familia de productos denominada SE, se compone de un procesador chipset Qualcomm msm 8227 CPU Snapdragon dual-core 1Ghz, una cámara digital lateral 5Mp res 2592x1944, un Flash LED con 0.5 d/s (disparos por segundo) y un sistema operativo denominado SESO v.2.3. Para analizar el impacto y la propagación de cambios, se introduce un conjunto de requerimientos adicionales a los satisfechos por la configuración previamente descripta, que surgen durante la etapa de diseño:

- Requerimiento 1: Incorporar una cámara digital frontal de 2Mp res 1733x1155.

- Requerimiento 2: Incorporar una unidad más del componente Flash.

- Requerimiento 3: Actualizar la versión del Sistema Operativo SESO v.2.3, por la versión SESO v.4.0.

A partir de estos requerimientos, el ejemplo es modelado con las extensiones de PRONTO y FM, las cuales se obtienen de especializar el modelo conceptual VERONTO (Fig. 3) en los modelos resultantes se analiza la variabilidad temporal mediante la representación de los cambios y la propagación de los mismos. Los cambios en una familia de productos pueden afectar a cualquiera de los niveles propuestos por PRONTO: Family, VariantSet y Product. Así, por ejemplo, en el nivel más abstracto (Family) las modificaciones están dadas en la estructura de los productos que forman la familia. En el nivel intermedio (VariantSet) pueden ser cambiados la selección de componentes, así como las restricciones entre conjuntos de variantes, y en el nivel más bajo (Product) puede verse afectada la especificación de los productos concretos. Sin embargo, un cambio en un cierto nivel, puede generar nuevos cambios en otro nivel. Para el primer caso, cambios en el nivel Family, es posible gestionar las versiones de cada nivel de forma independiente, mientras que en el segundo caso, cambios en el nivel VariantSet, se gestionan las versiones incluyendo las dependencias afectadas en diferentes niveles mediante la definición de restricciones. Por lo tanto, para integrar el modelo de Variabilidad Temporal con PRONTO, se especializa la entidad ProductConcept en los niveles: Family, VariantSet y Product, y para cada uno de estos, se especializan las entidades descriptas en la Sección "Modelo conceptual para gestionar la variabilidad temporal – VERONTO", en los elementos correspondientes a cada nivel de la AH.

Gestión de Variabilidad Temporal en PRONTO

En la Fig. 4 se representa la especialización de los conceptos Entity, ChangeEvent y ActivityConstraint

Fig.3: Especialización de VERONTO y FM en PRONTO.

para representar los cambios en cada uno de los mencionados niveles. En la notación empleada en la Fig. 4, en la parte superior de cada rectángulo se indica el concepto que la entidad está especializando. Así, por ejemplo, en el nivel de Family, la entidad Entity se especializa en CRelation/ DRelation, FRestriction y Value. Estas entidades son afectadas respectivamente por las especializaciones de ChangeEvent, FRestrictionCE, CRelationCE/DRelationCE y ValueCE. Cada evento de cambio afecta a la entidad correspondiente a través de una actividad. Para controlar el tipo de operación aplicable, cada Entity tiene asociada una restricción. Por lo tanto, para el nivel de familia, se definen un conjunto de restricciones que especifican que sólo las actividades Add y Delete, pueden afectar a FRestriction (FRestrictionAC) y a una CRelation (CRelationAC). En contraste, Value pueden ser afectados únicamente por el tipo de operación Edit (ValueAC).

En el nivel de conjunto de variantes (VariantSet), se identifica las entidades VSRestriction y Change, ambas son especializaciones de Entity. Estas entidades son afectadas respectivamente por los siguientes eventos

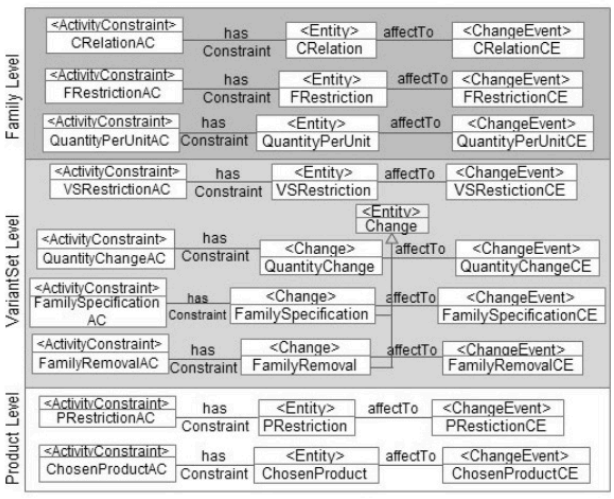

Fig. 4: Entidades de Cambio de PRONTO.

de cambios: VRestrictionCE, FamilySpecificationCE, FamilyRemovalCE y QuantityChangeCE. Además, cada entidad tiene asociada una restricción que controla el tipo de operación que puede afectarla, tal como: VSRestrictionAC, QuantityChangeAC, FamilySpecificatonAC y FamilyRemovalAC.

En el nivel de Producto, las entidades que pueden ser modificadas son: PRestriction y la relación chosenProduct, que se reifica en una entidad, dado que es susceptible a los cambios en el modelo. Estas entidades pueden ser afectadas por los eventos de cambio PRestrictionCE y ChosenProductCE, respectivamente. Asimismo, estas entidades están limitadas por las restricciones PRestrictionAC y ChosenProductAC.

Para estudiar la propagación de los eventos de cambios, en la Fig. 5 se representa el ejemplo descripto, mediante la instanciación del modelo conceptual de PRONTO. En dicha figura, se observa

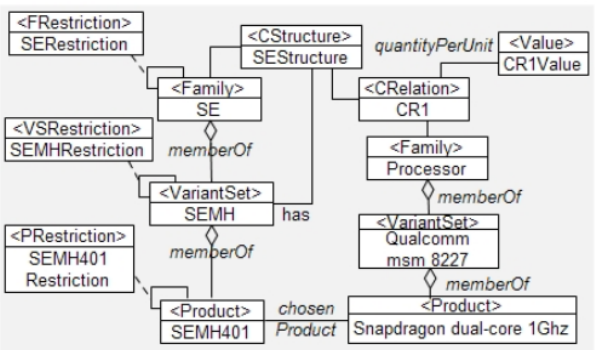

Fig.5: Representación del Producto en PRONTO.

que la familia de productos SE posee una estructura de composición (SEStructure), la cual se vincula con sus partes componentes mediante instancias de la clase CRelation. Para facilitar la comprensión de la Fig. 5, se muestra únicamente el componente Processor con la relación de composición CR1 y su atributo quantityPerUnit (CR1Value). De manera similar se especifican los

componentes restantes, descriptos en el ejemplo.

En el nivel de conjunto de variantes se identifica la variante SEMH, miembro de la Familia SE, que especifica su estructura de composición a través de la relación has. SEMH restringe los conjuntos de variantes que pueden usarse como componentes, que en este caso son: Qualcomm msm 8227, Lateral Camera 5Mp, FlashLED y SESO, miembros de de Processor, Camera, CameraFlash y SO, respectivamente. En la Fig. 5 se representa únicamente el conjunto de variante miembro de Processor (Qualcomm msm 8227). A su vez, en el nivel de Producto, se identifica al producto concreto SEMH401 miembro de SEMH, del cual infiere la estructura de composición y selecciona como componentes concretos del mismo a los siguientes productos: Snapdragon dual-core 1GHz, Camera lateral 5Mp res 2592x1944px, Flash Led 0.5d/s y SESO v2.3, miembros de los conjuntos de variantes antes mencionados.

La Fig. 6 muestra la aplicación de los conceptos propuestos para representar los cambios que se generan al atender el Requerimiento 1 sobre la familia de productos del caso de estudio representada con

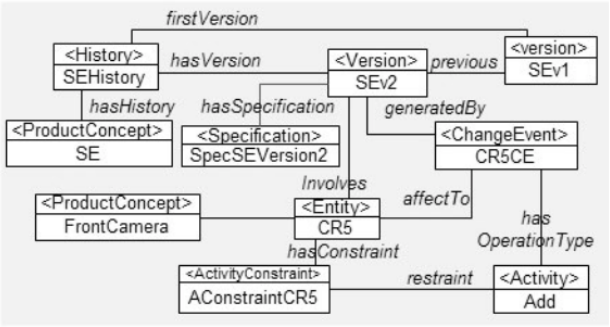

Fig.6: Instanciación de la ontología de versiones.

PRONTO. Este requerimiento se especifica mediante un evento de cambio (CR5CE), para agregar (Add) "una cámara digital frontal de 2Mp res 1733x1155", lo cual incorpora una relación de composición CR5 que vincula al nuevo componente, a la familia FrontCamera. FrontCamera se clasifica con el estereotipo ProductConcept, dado que es una familia que se incorpora como parte componente a la estructura SEStructure de la familia SE. Este evento genera una nueva versión de la familia SE (SEv2). En la Fig. 6, se ilustra el historial de la familia SE por medio de SEHistory, la cual involucra las versiones SEv1 y SEv2 de la familia.

El impacto del evento de cambio CR5CE sobre la entidad CR5, no genera nuevos eventos que se propaguen ya sea en el mismo nivel o en los niveles de conjunto de variantes y producto. De este modo, la entidad CR5 se clasifica como una entidad de tipo Absorbente. Otras entidades de esta categoría son FRestriction, VSRestriction y PRestriction, dado que un cambio que impacta en cada una de ellas, no se transmite a otros niveles o entidades. Es decir, se puede incorporar o eliminar una restricción sin afectar otras entidades. A diferencia de la situación presentada con el primer requerimiento, los eventos de cambios generados para resolver el segundo requerimiento: "Incorporar una unidad más del componente Flash", permiten observar el efecto de propagación.

Como puede verse en la Fig. 7, el segundo de los requerimientos genera un evento de cambio (ValueCR3CE) que impacta en la entidad ValueCR3 para editar el valor Number del atributo AttributeValueCR3 a 2, indicando que se requieren 2 unidades del componente Flash para la configuración del producto. Este evento genera una nueva versión a nivel de Familia (SEv3) y se propaga al siguiente nivel ocasionando una nueva versión del conjunto de variantes, SEMHv2. Esta propagación está indicada por la restricción EditFlashQPUChConstraint, que vincula los eventos de cambio ValueCR3CE y FlashQPUChCE. Este último evento, indica que se debe editar el valor de la propiedad Number de la entidad FlashQ-

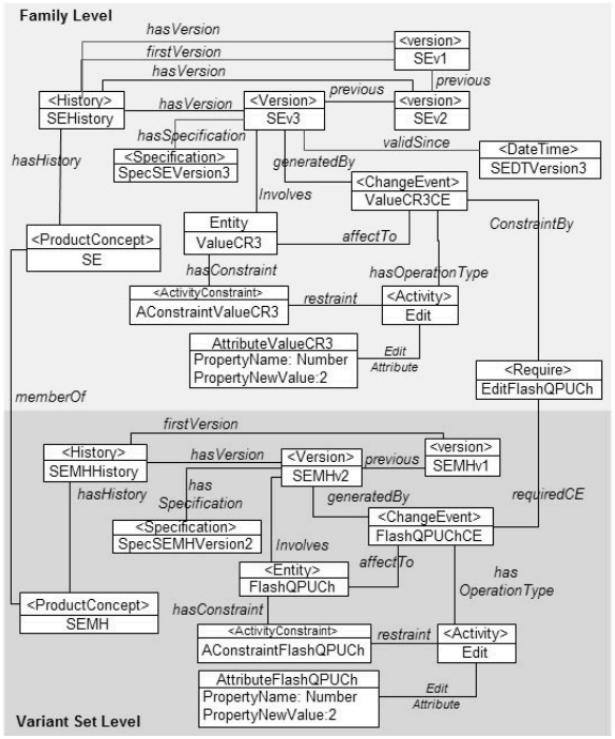

Fig.7: Instanciación de los eventos de cambio del Requerimiento 2.

PUCh, con el nuevo valor 2 (AttributeFlashQPUCh). Dado que en el nivel de productos el número de componentes requeridos se infiere desde la estructura de la familia, la propagación no continúa hacia el nivel más bajo. Por lo tanto, la versión de producto actual es compatible con la nueva versión del conjunto de variantes.

A fin de analizar el impacto de los cambios a nivel de productos, se debe examinar la aplicación del modelo para representar el Requerimiento 3: Actualizar la versión del Sistema Operativo SESO v.2.3 por

```
Value(?o) ∧ ChangeEvent(?oc) ∧ affectTo(?o,?oc)
∧ edit (?ot)∧ hasOperationType(?oc,?ot) ∧
QuantityPerUnitCh(?n) ∧ ChangeEvent(?m)
→ affectTo(?m,?n)
```

Fig.8. Regla de propagación de ChangeEvent para la entidad Value.

la versión SESO v.4.0.Este requerimiento se representa mediante dos eventos de cambios, que generan una nueva versión de producto, SEMH401v2. El primer cambio (chosenProduct8CE) afecta a la relación de composición ChosenProduct8, mediante la actividad Delete. Este cambio implica eliminar de la estructura el vínculo del producto con el componente SESO v2.3. Este evento requiere de un nuevo cambio, chosenProduct9CE, para vincular el nuevo componente SESO v4 mediante la entidad ChosenProduct9. Para este caso, se considera que los cambios son compatibles con las entidades de los niveles superiores y es válida su aplicación. Además, se clasifica la entidad afectada (ChosenProduct8) como Portadora, dado que recibe y transmite el mismo número de eventos.

En la Fig.9 (a) se representa la propagación de cambios que se dan en un mismo nivel y aquellos que se transmiten entre niveles de la AH de PRONTO, lo cual ocurre al introducir los cambios necesarios para satisfacer los Requerimientos 1, 2 y 3. En las situaciones analizadas la propagación no se transmite en sentido inverso, es decir desde el nivel de producto hacia el nivel de familia. A continuación, para analizar la compatibilidad de las versiones generadas, en la Fig.9 (b) se representa los puntos donde se verifica la compatibilidad de versiones mediante un conjunto de líneas dirigidas discontinuas. Tal es el caso de la versión de familia SEv2, donde se evalúa si los cambios introducidos para satisfacer el Requerimiento 2 permiten mantener el mismo conjunto de variantes que poseía la versión inicial SEMHv1. Esta verificación se representa en la Fig. 9(b) a través de la flecha de trazos número 1. En esta situación se observa que los cambios que dieron origen a SEv2 son compatibles con el conjunto de variantes de la familia inicial, SEMHv1, dado que la estructura es inferida a través de la relación has, incluyendo las modificaciones incorporadas en el nivel de Familia.

Otra situación se produce con los cambios introducidos para satisfacer el Requerimiento 2, lo cual da lugar a la versión de familia SEv3. En este caso es necesario generar una nueva versión (indicado en la Fig.9 (b) con el número 3) del conjunto de variantes SEMH (SEMHv2), dado que el evento de cambio se propaga afectando a una entidad de tipo Change (QuatityPerUnitCh). Este análisis de compatibilidad se representa en el instante 3 de la Fig.9 (b), indicado por el número 2. Este evento de cambio no se propaga al siguiente nivel (nivel de producto) dado que las cantidades requeridas residen en el nivel de Variantes y son inferidas a través de la estructura, como se mencionó anteriormente. Las líneas dirigidas continuas con puntas de flechas rellenas, representan la compatibilidad entre versiones. Así, la versión SEMHv2 es compatible con la versión de productos SEMH401v1.

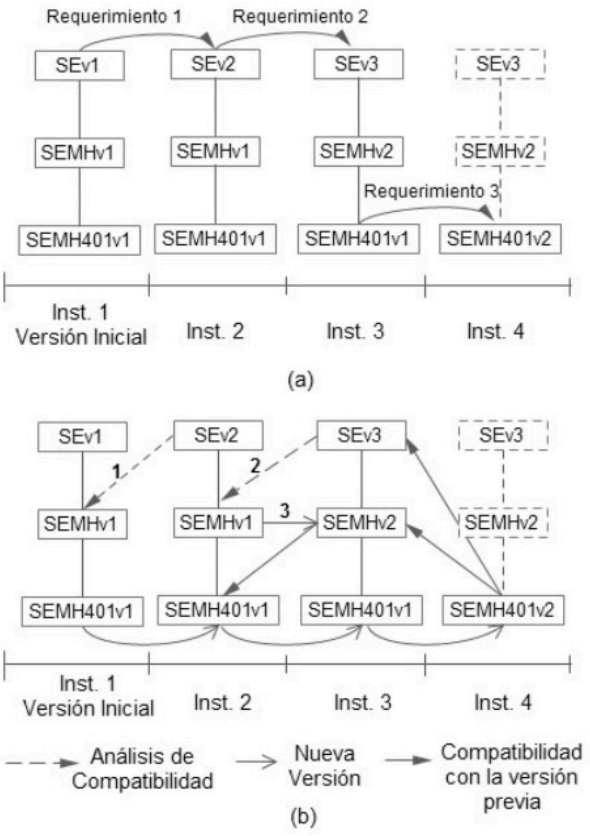

Fig.9: Propagación de Cambios en PRONTO.

Los cambios introducidos para satisfacer el Requerimiento 3 dan lugar a la nueva versión de producto SEMH401v2 (Fig. 9(a)). En el instante 4, el análisis de compatibilidad se desarrolla de forma ascendente, dando como resultado la compatibilidad de la nueva versión de productos SEMH401v2 con las versiones de familia y conjunto de variantes SEv3y SEMHv2, respectivamente. Los cambios introducidos no se propagan hacia arriba (nivel Variante o nivel Familia), por lo que las versiones generadas son compatibles con las versiones correspondientes a los niveles superiores.

Gestión de Variabilidad Temporal en el Modelo de Características - FM

En esta sección se representa la gestión de versiones de productos representados por medio de FM. Como se mencionó al comienzo del artículo, FM es un modelo más sencillo que PRONTO y utiliza una estructura de árbol compuesto de un conjunto de características (lo que podría asociarse al concepto de Familia en PRONTO), relaciones y restricciones, para representar la variabilidad espacial de familias de productos. En la Fig. 10 se identifican las entidades mencionadas por medio de la especialización del concepto ChangedEntity en: Feature, Relationship y Restriction, dado que son conceptos susceptibles de ser modificados. Una relación se clasifica en: ConsistOf, Optional, Mandatory, Selection y Alternative, y una restricción en: Require o Exclude. Una restricción del tipo Require indica que una característica necesita de otra característica, mientras que una restricción Exclude expresa

que la selección de una característica excluye a otra. Para representar los eventos de cambio que afectan a las entidades, se especializa ChangeEvent en las entidades FeatureCE, RelationshipCE y RestrictionCE. A su vez, RelationshipCE se especializa en ConsistOfCE, MandatoryCE, OptionalCE, AlternativeCE y SelectionCE. Finalmente, la entidad RestrictionCE se especializa en RequireCE y ExcludeCE. Para simplificar el diagrama de la Fig. 10, no se muestra en la misma, las especializaciones de RelationshipCE y RestrictionCE.

Con el fin de analizar la propagación de los cambios en FM, en la Fig.11 (a) se representa un conjunto de características en una estructura de árbol, para describir el caso basado en la telefonía

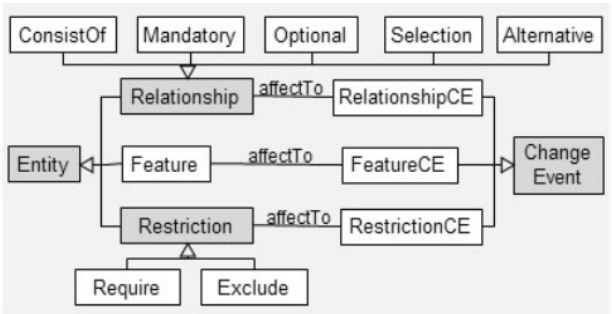

Fig.10: Entidades de Cambio de FM.

celular descripto anteriormente. La versión inicial se compone de una característica raíz para representar la familia SE y se compone de las características hijas Processor, Camera, CameraFlash 0.5 d/s y SO, siendo estas características obligatorias para la configuración del producto. La característica Processor consiste de la característica Snapdragon dual-core 1Ghz. La característica Camera consiste de Lateral Camera 5Mp res 2592x1944 y la característica SO consiste de SESO v2.3.

Para analizar la propagación de los eventos de cambios en FM extendido, primero se considera el Requerimiento 1: "Incorporar una cámara digital frontal de 2Mp res 1733x1155", el cual genera una nueva versión de la familia SEv2 (ver Fig.11 (b)). El evento de cambio ConsistOfCE1, elimina la entidad ConsistOf1, la cual vincula las características Camera (Feature1) y Lateral Camera 5Mp res 2592x 1944 (Feature2) y genera tres nuevos eventos de cambio:

1. FeatureCE3, agrega la característica FrontCamera2Mp 1733x1155 (Feature3),

2. MandatoryCE1, agrega una relación de obligatoriedad (Mandatory1) para vincular las entidades Feature1 con Feature2,

3. MandatoryCE2, agrega otra relación de obligatoriedad entre Feature1 y Feature3.

La versión SEv1 se representa en el modelo propuesto en la Fig.12, donde el árbol de características es un individuo de la clase ProductConcept y se lo denomina SE. Este individuo tiene un historial (SEHistory) que se compone de 2 versiones (SEv1 y SEv2). La nueva versión (SEv2) es generada por un evento de cambio (ConsistOfCE1) que afecta a la entidad ConsistOf1, mediante una operación de tipo Delete (Delete_ConsistOf1). Este evento de cambio posee una restricción que genera nuevos eventos de cambio para agregar dos relaciones de tipo obligatoria (MandatoryCE1 y MandatoryCE2) y una característica (FeatureCE3), los cuales afectan las entidades: Mandatory1, Mandatory2 y Feature3, respectivamente.

Los requerimientos 2 y 3, no se representan en la ontología de gestión de versiones, por cuestiones de espacio. Sin embargo estos requerimientos tienen una representación similar al del Requerimiento 1. El impacto del evento de cambio ConsistOfCE1 sobre la entidad ConsistOf1 genera un número mayor de eventos de cambio, por lo que se clasifica como un evento multiplicador. El Requerimiento 2 introduce un cambio de tipo Portador y consiste en eliminar la característica Snapdragon Dual Core 1Ghz (Feature4) y agregar la característica Adreno Dual Core

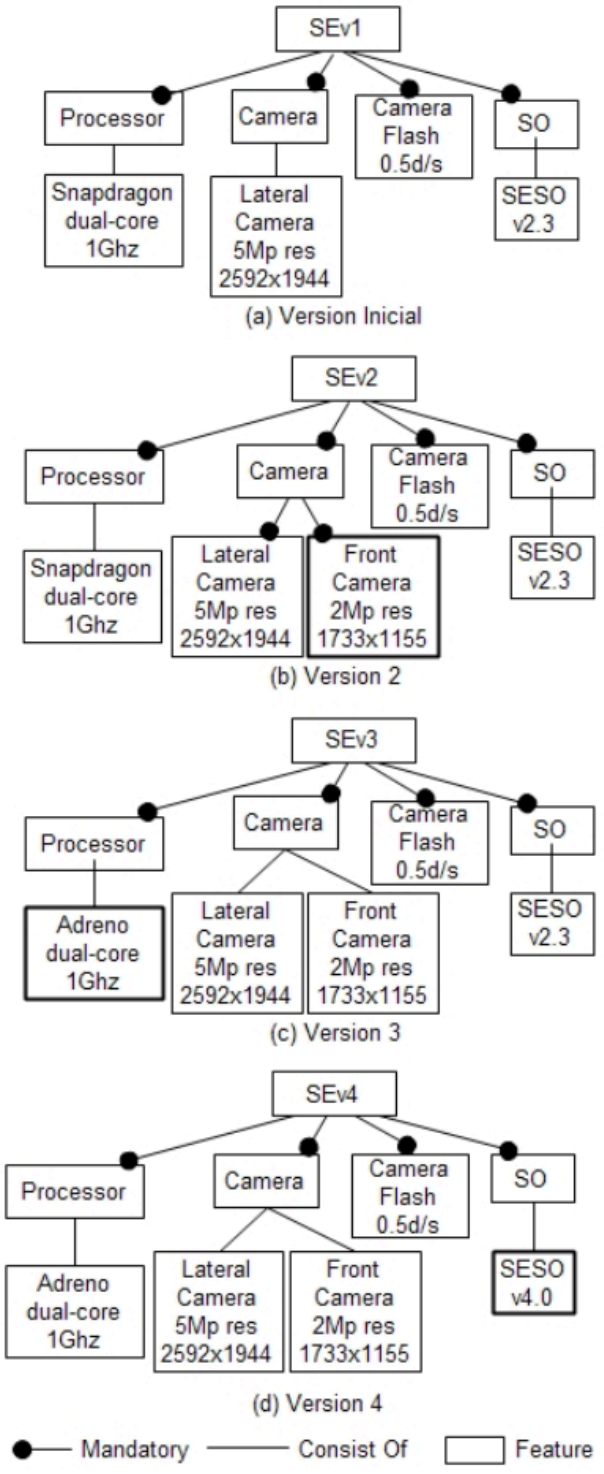

Fig.11: Propagación de Cambios en FM.

1Ghz (Feature5), generando la versión 3 de la familia SE (ver Fig.11 (c)). Finalmente, el Requerimiento 3 genera una nueva versión de familia SEv4 (Fig.11 (d)), donde el evento se clasifica como portador y requiere eliminar la característica SESO v2.3 y agregar una nueva característica SESO v4.0.

La propagación de los cambios puede controlarse mediante instancias de la entidad ChangeEventConstraint, indicando que se requiere un nuevo evento de cambio. Es importante mencionar que la familia de producto en FM se representa íntegramente como instancia de ProductConcept, a diferencia de PRONTO que cada nivel de la AH son individuos diferentes de ProductConcept.

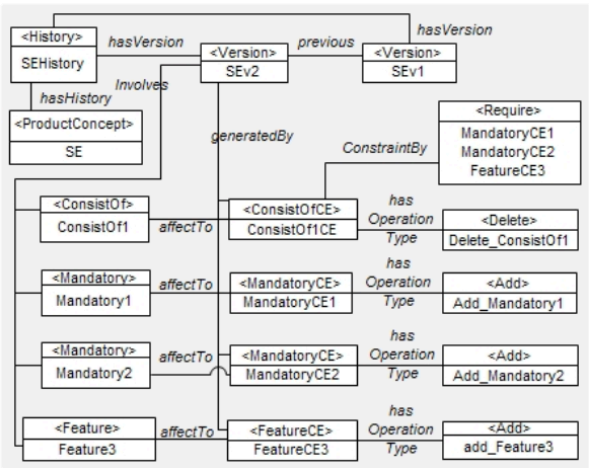

Fig.12: Propagación de Cambios en FM.

CONCLUSIONES

El desarrollo del trabajo permitió analizar las diferentes propuestas de gestión de variabilidad de familias de productos, tanto en modelos para empresas de manufactura como en la industria del software. Se utilizaron los modelos de PRONTO y FM, los cuales poseen un conjunto conceptos vinculados entre sí, que además permiten representar la propagación de los eventos de

cambio. Estos eventos pueden comportarse e impactar de diversas formas sobre el conjunto de entidades de los modelos. Por esta razón, es importante estudiar el comportamiento de los cambios para gestionarlo de forma eficiente y mantener la consistencia de la representación de la información de productos en el tiempo y espacio. En este trabajo se presenta el modelo VERONTO que permite extender los modelos de producto, que solo representan variabilidad espacial, con la gestión de variabilidad temporal. A partir de esto, es posible resaltar un conjunto de características de VERONTO. La primera de ellas, es que su modelo conceptual se construye mediante conceptos genéricos de modo que su implementación sea independiente de los modelos de productos y del dominio de aplicación. Además, esta característica de extensibilidad, permite gestionar de forma simultánea, las dos dimensiones de variabilidad de una familia de producto. Finalmente, la formalización de la propuestas a través de un lenguaje de ontologías, tal como OWL, proporciona una representación del modelo de modo que puede ser compartido con una o más ontologías, con el fin de lograr un vocabulario común que gestionar la variabilidad temporal de productos.

Como trabajos futuros, se propone extender VERONTO con reglas que incrementen la robustez de los modelos de productos generados a partir de la verificación de la consistencia y la compatibilidad de versiones. Además, se implementará la propuesta en otros casos de estudio, con el fin de identificar comportamientos repetitivos de los eventos de cambios y de esta forma, definir patrones de comportamientos para una gestión eficiente de los cambios. Esta gestión deberá conducir a la predicción de los efectos de tales cambios y sus dependencias, con el fin de administrarlos mediante un proceso de gestión de cambios adecuado.

AGRADECIMIENTOS

Este trabajo ha sido financiado en forma conjunta por CONICET, la UTN (PID 25-O156) y la Universidad Nacional de La Rioja. Se agradece el apoyo de estas instituciones.

REFERENCIAS

Matsokis A. Kiritsis D. An ontology-based approach for products lifecycle management. Computer in Industrial 61 787-797. (2010)

Asikainen T., Mannisto T., Soininen T. Kumbang: A domain ontology for modelling variability in software product families. Advances engineering informatics. (2006)

Pohl K., Bockle G., Van Der Linden F. Software Product Line Engineering. Foundations, principles, and Techniques. ISBN-10 3-540-24372-0 Springer Berlin Heidelberg New York. (2006)

Garces, K., Parra, C., Arboleda, H., Yie, A., Casallas, R.: Administración de Variabilidad en una Línea de Producto Basada en Modelos. In Proceedings of the Congreso Colombiano de Computación, Bogotá, Colombia. (2007)

Männistö T, A Conceptual Modelling Approach to Product Families and their Evolution. Acta Polytechnical Scandinavica, Mathematics and Computing Series. No. 106, ISSN 1456-9418. (2000)

Estublier J. and Casallas R. Three dimensional versioning. ICSE SCM-4 and SCM-5 Workshops Selected Papers. Software Configuration Management. Volume 1005, issue 1995, pp 118-135.ISBN 978-3-540-60578-2 (2005)

Vegetti, M., Leone, H., Henning, G. PRONTO: An ontology for comprehensive and consistent representation of product information. Engineering Applications of Artificial Intelligence 24 (8), pp. 1305-1327. (2011)

Kang K.C. Lee K., Lee J.: Feature Oriented Product line Software Engineering: Principles and guidelines. Domain-Oriented Systems Development: Practices and Perspectives. Cap. 2. ISBN 0203711874. (2003)

Kang K., Lee J., Donohoe P.: Feature-Oriented Product Line Engineering. IEEE software 19. (2002)

Ecker C., Clarkson J.P., Zanker W. Change and Customization in complex engineering domains. Research in Engineering Design. Volume 15, Issue 1 pp 1-21. (2004)

FECHA DE PRESENTACIÓN: 30/06/16
FECHA DE APROBACIÓN: 25/10/16