Ontología para la gestión unificada de variantes y versiones de productos

Sonzini María Soledad[1,2], Vegetti Marcela[1]

[1] INGAR, Instituto de Desarrollo y Diseño, Avellaneda 3657, Santa Fe, Arg.
[2] Universidad Nacional de La Rioja, Luis M. de la Fuente S/N, La Rioja, Arg.
[1]{ssonzini, mvegetti}@santafe-conicet.gob.ar

Resumen. El objetivo de este trabajo es presentar una ontología para gestionar la variación temporal de una familia de productos a través de versiones. La propuesta permite identificar los puntos variantes, la causa, el tiempo de validez y el control de la propagación/ impacto de los cambios. Es una ontología genérica que puede ser integrada con distintos modelos de representación de variantes de productos. A fin de validar la propuesta, se muestra la integración de la ontología de versiones propuestas con la ontología PRONTO (PRoduct ONTOlogy) [1] para la gestión de variantes de familias de productos.

Palabras Claves: Variabilidad, Ontología, Versiones, Familia de productos.

1 Introducción

La importancia de gestionar la información de los productos en todas las fases de su ciclo de vida, está en la ocurrencia de un evento de cambios en una determinada etapa, el cual podría propagarse y afectar la consistencia e integridad de la información en otras etapas. Pohl y colab. en [2], sostienen que es fundamental hacer una distinción entre variabilidad en el tiempo y variabilidad en el espacio. La primera de ellas se define como "la existencia de diferentes versiones de un producto que es válido en diferentes tiempos", denotando su evolución. Por el contrario, la variabilidad en el espacio se define como "la existencia de un producto en diferentes formas en un mismo tiempo". Esta dimensión abarca de manera simultánea el uso de diferentes variantes de productos 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 y que pueda ser aplicado conjuntamente con los modelos de representación de variantes existentes. De esta manera se podría gestionar simultáneamente ambos tipos de variabilidad.

Un concepto primordial para este trabajo, es el concepto de "ontologías", que se han propuesto como una herramienta de integración semántica en el contexto de la Web Semántica [3]. Una ontología es un modelo formal que representa explícitamente el conocimiento consensuado de un dominio [4]. Es por ello, que las

ontologías se definen para establecer un vocabulario común, sin ambigüedades entre diferentes áreas de una misma organización o entre organizaciones diferentes.

Con el fin de contar con un modelo formal para la gestión integrada de variantes y versiones, así como un vocabulario común para la representación de los cambios que modifican la información de productos durante su ciclo de vida, este artículo propone una ontología, desarrollada en OWL[1]. La propuesta define conceptos genéricos, que pueden ser especializados utilizando entidades de modelos de gestión de variantes existentes. En particular, este trabajo muestra cómo el modelo propuesto puede extenderse con conceptos de la ontología de productos PRONTO, para gestionar de manera conjunta las variantes y las versiones. Asimismo, se presenta un conjunto de reglas, a través del lenguaje SWRL (Semantic Web Rule Language)[2], que permiten inferir nuevo conocimiento. 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 de mencionadas. En la sección 3, se presenta el modelo conceptual genérico para gestionar la variabilidad temporal, su implementación en OWL y la definición de un conjunto de reglas de inferencia. La Sección 4 introduce un caso de estudio sencillo a fin de validar la propuesta. Finalmente, se presentan las conclusiones y trabajos futuros.

2 Gestión de variabilidad

En diferentes investigaciones se ha tratado el tema de la variabilidad de familias de productos, sin embargo aún existen cuestiones sin resolver, tal como la gestión de dependencias entre puntos de variación y variantes. Esta situación ha impulsado algunos estudios sobre las relaciones que existen (no siempre explícitas) entre los diferentes elementos afectados (puntos de variación) a lo largo del ciclo de vida de un producto. Sin embargo, a pesar de que existen diversas propuestas para el manejo de las variabilidades en el espacio y en el tiempo, no se ha encontrado soluciones que aborden simultáneamente estas dos problemáticas.

Existe un conjunto de propuestas que se concentran en la gestión de la variabilidad temporal de conjuntos de datos de familias de productos en el dominio de la industria de software. Entre estas investigaciones, Männisto [5] propone una metodología para modelar la evolución de familias de productos, incluyendo los aspectos temporales y los mecanismos necesarios para la representación de datos. Estublier y Casallas en [6], 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. Sjoberg [7] sostiene que para obtener herramientas sofisticadas y poder predecir las consecuencias de los cambios, es necesario identificar un conjunto de cuestiones, tales como: que objetos fueron modificados, en qué instante de tiempo ocurre, cómo fueron cambiados estos objetos

[1] http://www.w3.org/2007/09/OWL-Overview-es.html
[2] http://www.w3.org/Submission/SWRL/

y de qué modo se registra este acontecimiento. En cuanto a la variabilidad en el espacio, es posible encontrar propuestas tanto en dominios relacionados con la manufactura como en la industria del software. Por cuestiones de espacio, en este artículo se introducirán sólo los conceptos de PRONTO [1]. 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 un modelo conceptual [3] que describe dos jerarquías: la jerarquía estructural (SH - Structural Hierarchy) para representar la información concerniente a las partes que participan en la manufactura de un producto final, y la jerarquía de abstracción (AH - Abstraction Hierarchy) que permite la representación de información no estructural de productos en diferentes niveles de abstracción. La AH consta de 3 niveles: Familia (*Family*), Conjunto de Variantes (*VariantSet*) y Producto (*Product*), los cuales se relacionan entre sí mediante la asociación "*memberOf*". La SH considera dos tipos de relaciones de estructuras, que se especializan en "*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. Es importante aclarar que existe una SH para cada uno de los niveles de la AH y las relaciones que la constituye se infieren a partir de las entidades y relaciones definidas de manera explícita en PRONTO. A continuación se introducen brevemente los mismos.

Una familia representa un conjunto de productos que son similares y puede ser simple (*SFamily*), o compuesta (*CFamily*). Esta última tiene al menos una estructura (*Structure*) asociada, que puede ser: de composición (*CStructure*) o de descomposición (*DStructure*). Los componentes de una *CStructure* y los derivados de una *DStructure* se asocian a dicha estructura por medio de dos tipos de asociaciones: *CRelation* y *DRelation*, respectivamente. En ambos casos, estas relaciones están vinculadas con las cantidades de cada componente o derivado que se necesitan o derivan de la estructura involucrada, a través de las asociaciones *quantityPerUnit* y *productionFactor* [3]. 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* y puede definir cambios (*ChangeSet*) sobre la misma. Cada uno de estos cambios, se aplican a una relación de la estructura (*affectedRelation*). Existen diferentes tipos de cambios que pueden aplicarse. Su descripción está fuera del alcance de este trabajo. Para más detalles sobre los mismos ver [1].

El nivel más bajo 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*,

[3] Ver el modelo conceptual en el repositorio: https://sites.google.com/site/ontoversioningrepository/pronto---product-ontology

PRestriction) que permiten limitar las entidades que deben o no pueden estar juntas en una misma SH en cada uno de los niveles propuestos.

3 Ontología de Versiones

En la Fig. 1 se introducen los conceptos fundamentales de la ontología de versiones propuesta. Uno de estos es el concepto de *ProductConcept*, que representa el elemento cuyas versiones se van a gestionar. El conjunto de versiones es representado con el concepto de *History*. Este concepto mantiene las versiones bajo la forma de una secuencia expresada mediante las relaciones *firstVersion* y *previous*, las cuales denotan la evolución en el tiempo y permiten reconstruir la versión actual de un producto a partir de las sucesivas versiones. La relación *firstVersion* indica la versión inicial, y cada versión (excepto la primera), es relacionada con su predecesora a través de la asociación *previous*. Una versión (*Version*) representa la configuración del elemento en un período de tiempo en el cual ésta es válida. Este período, abarca desde el instante en que se crea la versión (indicado por *DateTime*), hasta el instante en el que se crea la siguiente versión. El concepto de *Specification* representa el conjunto de información acerca de las causas que motivaron la generación de la versión, las cuales podrían ser una actualización tecnológica, nuevos requerimientos de los usuarios, modificaciones legales, etc. Además, mediante este concepto, es posible registrar información acerca del responsable que registra la nueva versión.

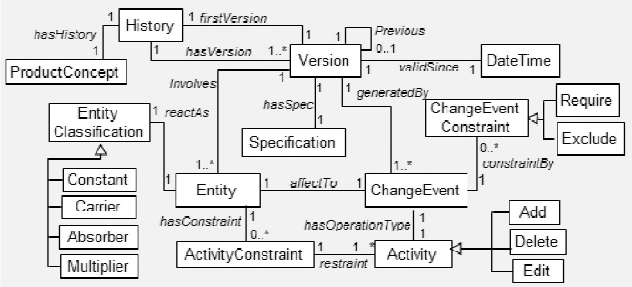

Fig. 1. Modelo Conceptual para la gestión de variabilidad temporal

El origen de una versión está dado por la ocurrencia de un evento de cambio, el cual se representa mediante el concepto de *ChangeEvent* y se vincula a ésta a través de la relación *generatedBy*. Los eventos de cambios afectan a una entidad (*Entity),* a través de una actividad (*Activity*). Una *Entity* es un concepto abstracto que debe ser extendido para representar los elementos de un modelo de productos específico que serán afectados por los cambios. Es decir, una entidad podría ser una relación, un atributo, una restricción, u otro elemento del modelo de variabilidad espacial de productos que es extendido.

En el modelo propuesto, las actividades que generan cambios son: el agregado (*Add*) y la eliminación (*Delete*) de un elemento, así como la modificación (*Edit*) de

atributos. 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,* el cual se vincula a una entidad mediante la relación *hasConstraint* y permite restringir las operaciones *Add*, *Delete* o *Edit* que puede aplicarse sobre la misma.
En el contexto de la información de familias de producto, una entidad forma parte de una estructura y un cambio en ella podría afectar a otras entidades de la estructura, produciéndose una serie de nuevos eventos de cambios, también conocida como propagación de cambios. Para controlar este efecto, se introduce el concepto de *ChangeEventConstraint*, el cual especifica las restricciones asociadas a las reacciones de los cambios, es decir: si la ocurrencia de un evento de cambio requiere (*Require*) o no (*Exclude*) que se produzca un nuevo evento de cambio.

Considerando el impacto que tiene un evento de cambio sobre una entidad específica, el modelo conceptual considera la clasificación de entidades (*EntityClassification*) introducida por Ecker y colab. en [8], donde una entidad puede clasificarse como: i) Absorbente (*Absorber*), si absorbe un número de cambios mayor de los que pueden generar y transmitir; ii) Portadora (*Carrier*), cuando puede absorber el mismo número de cambios que el que transmite a otras entidades; iii) Multiplicadora (*Multiplier*) si genera un número de cambios mayor al número de cambios que puede absorber; y iv) Constante (*Constant*), si no absorbe ni transmite 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 responder a una serie de preguntas, tales como: ¿Qué elementos fueron afectados en la nueva versión?, ¿Cómo se vieron afectados estos elementos?, ¿Cuándo comienza a ser válida la nueva versión?, ¿Por qué se realizaron los cambios?, ¿Cómo se registran estos eventos?, ¿Cómo reacciona una entidad ante un evento? Y ¿Qué nuevos eventos se generan a partir de ésta última?

La propuesta de 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. De este modo, se obtiene la gestión simultánea de las dimensiones de variabilidad espacial y temporal de una Familia de Productos. La ontología propuesta se ha implementado en lenguaje OWL mediante la herramienta Protégé en su versión 5.0. Inicialmente, se definió un namespace con el prefijo *ovm,* para contener todos los identificadores únicos de los elementos de la ontología. Una vez definido el espacio de nombres, se identificaron todos los conceptos y se clasificaron para organizarlos en una estructura jerárquica, Para cada entidad se especifican un conjunto de expresiones (axiomas) para verificar que la información del dominio que representa, sea consistente y correcta, tal como: "una entidad *ProductConcept* debe estar asociado a una entidad *History*". Seguido de esta clasificación, se definieron los tipos de datos asociados a cada entidad, y las propiedades que vinculan los términos de un dominio con los términos de un rango.

Con el fin de definir el comportamiento, la semántica de las relaciones e inferir nuevo conocimiento por medio de la deducción, se definió un conjunto de reglas SWRL. Una regla SWRL tiene dos partes, el antecedente y el consecuente. En este sentido, si todos los conceptos atómicos en el antecedente de una regla son verdaderos, entonces la consecuencia debe ser verdadera también. En la Fig. 2, se ilustran algunas reglas para inferir nuevo conocimiento, tal como: clasificar una entidad en base a su reacción ante la ocurrencia de un evento de cambio que la afecta (Reglas 2 y 3) [4], conocer qué entidades están involucradas en una versión (Regla 4), o inferir la propiedad *hasVersion* a partir de la propiedad *firstVersion* (Regla 1).

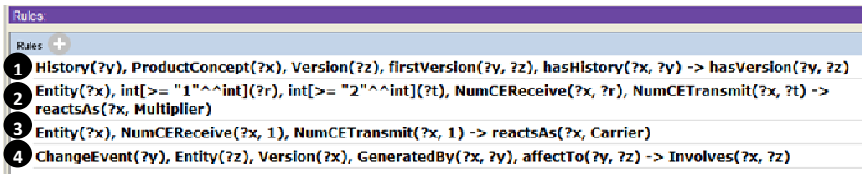

Fig. 2. Reglas de Inferencia en SWRL

Una vez que la información acerca del cambio es capturada y formalizada, es posible formular consultas para obtener y manipular datos almacenados en un formato de tripleta por medio de la utilización de un lenguaje de consultas SPARQL[5]. Este lenguaje permite responder el conjunto de preguntas mencionado los párrafos anteriores, proporcionando un conocimiento apropiado para la gestión de cambio dentro de la información de una familia de producto durante su ciclo de vida. Estas consultas se describen y se demuestran en el caso de estudio de la siguiente sección.

4 Gestión de versiones de la familia de productos *PhoneSE*

Para validar la propuesta se extiende la ontología de versiones en PRONTO, tomando como base, un sencillo ejemplo acerca de un producto ficticio basado en la telefonía celular, con el objetivo de representar los eventos de cambios y la propagación de los mismos.

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 *PhoneSE*, 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. Por cuestiones de espacio, para analizar el impacto y la propagación de

[4] Ver reglas de inferencia en: https://sites.google.com/site/ontoversioningrepository/ontology-versioning

[5] http://www.w3.org/TR/rdf-sparql-query/

cambios, este caso se basa en un único requerimiento que surgen durante la etapa de diseño: "Incorporar una cámara digital frontal de 2Mp res 1733x1155".

A partir de esta breve descripción, los cambios en una familia de productos puede ocurrir en cualquiera de los niveles propuestos por PRONTO: *Family*, *VariantSet* y *Product*. Dado que en el nivel más abstracto, las modificaciones están dadas en la estructura de los productos que forman la familia. En el nivel intermedio pueden ser cambiados la selección de componentes, así como las restricciones entre conjuntos de variantes, y en el nivel más bajo puede verse afectada la especificación de los productos concretos. Sin embargo, un cambio en un cierto nivel, puede generar nuevos cambios en otro nivel. Por esto, se propone la especialización de la entidad *ProductConcept* en los niveles: *Family*, *VariantSet* y *Product*, y para cada uno de estos niveles, se identifican los elementos susceptibles de ser modificados y se representan a través de la especialización de los conceptos *Entity*, *ChangeEvent* y *ActivityConstraint* [6]

Así, por ejemplo en el nivel de Familia, la entidad *Entity* se especializa en *CRelation/DRelation*, *FRestricion* y *QuantityPerUnit/ ProductionFactor*. Estas entidades son afectadas respectivamente por las especializaciones de *ChangeEvent FRestrictionCE*, *CRelationCE/ DRelationCE* y *QuantityPerUnitCE*. En 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, *QuantityPerUnit/ ProductionFactor* pueden ser afectados únicamente por un tipo de operación *Edit* (*QuantityPerUnitAC*).

En el nivel de conjunto de variantes (*VariantSet*), se identifica las entidades *VSRestriction* y *Change* como especializaciones de *Entity*. Estas entidades son afectadas respectivamente por los siguientes eventos 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*. En la Fig. 3.a se representa el ejemplo descripto, instanciando el modelo de PRONTO, donde la familia de productos *PhoneSE* posee una estructura de composición (*SEStructure),* la cual se vincula con sus partes componentes mediante instancias de la clase *CRelation.* Para facilitar la comprensión, únicamente se muestra el componente *Processor* con la relación de composición *CR1* y su atributo *quantityPerUnit* (*CR1Value*). En el nivel de conjunto de variantes se identifica la variante *SEMH,* miembro de la Familia *PhoneSE,* que especifica su estructura de composición a través de la relación *has. SEMH* restringe los conjuntos de variantes que pueden usarse como componentes a los conjuntos de variantes: *Qualcomm msm 8227*, *Lateral Camera 5Mp, FlashLED* y *SESO,* miembros de *Processor, Camera, CameraFlash* y *SO,* respectivamente. En la figura se representa únicamente el

[6] Ver diagrama en el repositorio en:
https://sites.google.com/site/ontoversioningrepository/ontology-versioning

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. Este conjunto de instancias se traduce a individuos, poblando la ontología de versiones, las cuales constituyen las versiones iniciales de los 3 niveles: *PhoneSEv1, SEMHv1, SEMH401v1.*

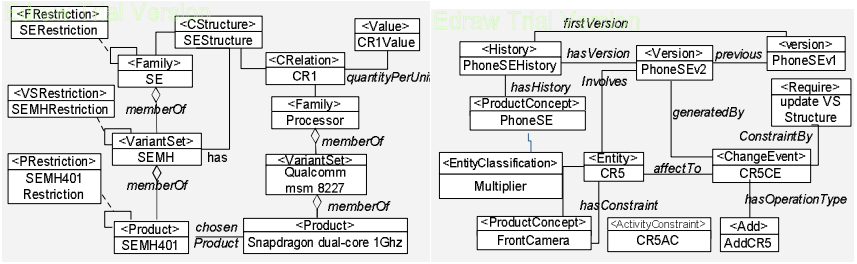

Fig. 3. a) Representación en PRONTO. b) Instanciación de la ontología de versiones.

Para dar respuesta al requerimiento mencionado, en la Fig. 3.b se representan los individuos que intervienen en la gestión del cambio que afectan a la familia de productos. Este requerimiento implica un evento de cambio (*CR5CE*), para agregar (*AddCR5*) una relación de composición *CR5* que vincule al nuevo componente *FrontCamera.* Para esta situación se genera una nueva versión de la familia (*PhoneSEv2*) formando parte del historial *PhoneSEHistory,* y se vincula con su versión previa *PhoneSEv1*. El evento de cambio *CR5CE* posee una restricción asociada (*updateVSStructure*) para indicar que una nueva versión de conjunto de variante debe inferir la estructura generada para incluir el nuevo componente. Dada la situación anterior, se genera una nueva versión de *SEMH (SEMHv2)* para incluir el componente *FrontCamera 2Mp,* mediante la selección de la nueva estructura de la familia a la que pertenece. La nueva versión del conjunto de variantes*SEMHv2*, es compatible con la versión inicial del producto *SEMH401* (*SEMH401v1).* Sin embargo, para incluir el componente *FrontCamera 2Mp 1733x1155px*, es necesario inferir la nueva estructura a nivel de producto, donde la relación *chosenProduct (chosenProduct5)* asocia el nuevo componente. Este evento, denominado *chosenProductCE5*, es el responsable de generar una nueva versión de producto *SEMH401v2*. Para esta situación el cambio impacta en el nivel de familia en la entidad *CR5*, y ésta reacciona generando un número mayor de cambios de los que recibe, los cuales se propagan a los niveles inferiores y se clasifica como una entidad multiplicadora de eventos. Esta clasificación se infiere de la regla 2 de la Fig. 2. Sobre el conjunto de información que representa la ontología de versiones, es posible ejecutar una seria de consultas en SPARQL, para obtener las respuestas de las preguntas de la sección 3. En la Tabla 1 se presentan un conjunto reducido de consultas y sus resultados, con respecto a la nueva versión que se generó de la familia

PhoneSE. Más consultas pueden verse en el repositorio[7]. De este modo, es posible conocer los componentes afectados en la nueva versión, cuál fue la causa del cambio, como éste afectó a la entidad modificada y la reacción de esta última, en cuanto a si absorbió, propagó o no el evento de cambio.

Table 1. Consultas en SPARQL

Pregunta	¿Qué elementos fueron afectados en la versión PhoneSEVersion2?
Consulta	`SELECT ?entity WHERE { ovm:PhoneSEVersion2 ovm:Involves ?entity }`
Resultado	Entity CR5
Pregunta	¿Por qué se aplicaron los cambios en la versión PhoneSEVersion2?
Consulta	`SELECT ?specification ?description WHERE { ?specification ovm:Description ?description. ?specification ovm:isSpecificationOf ovm:PhoneSEVersion2 }`
Resultado	specification | description SpecVersion2 "Se agrega la Familia FrontCamera como componente de la estructura SEStructure"
Pregunta	¿De qué modo se afectaron a los elementos en la versión PhoneSEVersion2?
Consulta	`SELECT ?change ?activity ?entity ?date ?datetime WHERE { ovm:PhoneSEVersion2 ovm:GeneratedBy ?change. ?change ovm:affectTo ?entity. ?change ovm:hasOperationType ?activity. ovm:PhoneSEVersion2 ovm:ValidSince ?date.?date ovm:datetime ?datetime } ORDER BY ASC (?date)`
Resultado	change | activity | entity | date AddCR5CE Add CR5 DTVersion2
Pregunta	¿Cómo se registran las versiones de la Familia PhoneSE?
Consulta	`SELECT ?productConcept ?history ?version WHERE {ovm:PhoneSE ovm:hasHistory ?history. ?history ovm:hasVersion ?version} ORDER BY ASC (?version)`
Resultado	history | version PhoneSEHistory PhoneSEVersion1 PhoneSEHistory PhoneSEVersion2
Pregunta	¿Cómo se clasifica la entidad afectada en la versión PhoneSEVersion2?
Consulta	`SELECT ?change ?entity ?classification WHERE { ovm:PhoneSEVersion2 ovm:GeneratedBy ?change. ?change ovm:affectTo ?entity. ?entity ovm:reactsAs ?classification. }`
Resultado	change | entity | classification AddCR5CE CR5 Multiplier

5 Conclusión

La propuesta de este trabajo se basa en el uso de ontologías para tratar la variabilidad temporal de modelos de producto, en término de versiones, manteniendo su coherencia y consistencia. Diferentes modelos de representación de familias de productos, que manejen la variabilidad en el espacio, pueden ser extendidos por esta propuesta, y así gestionar los cambios durante su ciclo de vida. En base a esto, el

[7] https://sites.google.com/site/ontoversioningrepository/ontology-versioning

desarrollo de la propuesta de este trabajo permitió representar la gestión de versiones de las familias de productos, extendiendo a PRONTO, junto a la definición de reglas de inferencia SWRL que permitieron inferir nuevo conocimiento sobre un caso concreto. Además, se escribieron consultas en SPARQL dando como resultado información respecto a los cambios, al registro de versiones y a la reacción de los elementos afectados por los cambios. De esta forma se logra validar la propuesta y, desde una perspectiva funcional, se logra extender la representación de información de familia de productos. Con el fin de mejorar la interpretación y la visualización de imágenes, se generó un repositorio de datos para acceder a más detalles del contenido de este trabajo.

A futuro, se pretende extender la propuesta teniendo en cuenta la propagación de los cambios en diferentes niveles de PRONTO. Asimismo, se realizarán actividades para complementar la validación de la propuesta y se trabajará en la identificación de patrones de comportamientos repetitivos a fin de lograr predecir las consecuencias de los cambios para una gestión eficiente de estos.

6 Agradecimientos

Se agradece el apoyo brindado por estas instituciones: CONICET, Univ. Tecnológica Nacional (PID 25-O156 y PID 25-O144) y Universidad Nacional de La Rioja.

Referencias

1. M., Vegetti, H., 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)
2. 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)
3. Shadbolt, N., W. Hall and T. Berness-Lee, (2006). The Semantic Web Revisited. IEEE Intelligent Systems, May-Jun.
4. Brandt, S.C., J. Morbach, M. Miatidis, M. Theißen, M. Jarke and W. Marquardt, 2008. An Ontology-Based Approach to Knowledge Management in Design Processes. Computers and Chemical Engineering, 32, 320-342.
5. 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)
6. 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)
7. Sjoberg D. Managing Change in Information Systems: Technological Challenges. Departament of Informatics, University of Oslo. N-0316 Oslo, Norway. (1995)
8. 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)