https://creativecommons.org/licenses/by/4.0/

DEL ESQUEMA DE FÁBRICAS DE SOFTWARE A UNA ARQUITECTURA DE REFERENCIA POR MEDIO DE LAS TRANSFORMACIONES DE MODELOS (IMFS & VCD)

From software factories schema to an architecture of reference, using model transformations (MESF & DQV)

ALEIDYS S. ARRAIZ GOICOCHEA[1], VICTOR A. ESTELLER L.[2]

Recibido:01 de junio de 2018. Aceptado:07 de junio de 2018

DOI: http://dx.doi.org/10.21017/rimci.2018.v5.n10.a45

Resumen

Las Fábricas de Software (FS) representan un enfoque que fomenta la reutilización para la construcción de Familias de Sistemas de Software (FSS), se basan en un Esquema y una Plantilla de FS. En el Esquema se desarrollan las Arquitecturas de las Líneas de Productos de Software (ALPS) que son arquitecturas iniciales en las que se incorporan los aspectos relativos a la calidad que deben estar presente en los productos derivados, esto a través de la Vista de Calidad del Dominio (VCD), que plantea la integración de un modelo de calidad. Este trabajo propone un proceso denominado IMFS&VCD para la incorporación de los principios de la Ingeniería de Modelos (IM) para la creación de modelos, utilizando transformaciones y siguiendo un enfoque Top-Down para la obtención de una Arquitectura de Referencia (AR). El proceso planteado se lleva a cabo en la fase de Análisis del Dominio (AD) en el contexto de la Ingeniería del Dominio (ID), contemplando y describiendo las actividades, artefactos y herramientas de soporte que lo sustentan.

Palabras clave: Fábricas de Software, Familias de Sistemas de Software, Ingeniería de Modelos, Reutilización, Transformaciones, Vista de Calidad del Dominio.

Abstract

Software Factories (SF) represent an approach that encourages reuse for the construction of Software System Families (SSF), are based on a Scheme and a FS Template. In the Scheme the Product Lines Architecture (PLA) is developed which is an initial architecture in which the quality aspects that must be present in the derived products are incorporated, this through the Quality View of the Domain (QVD), which proposes the integration of a quality model. This work proposes a process called MESF&QVD for the incorporation of the principles of Model-Driven Engineering (ME) for the creation of models, which uses transformations, and these are described to obtain a Reference Architecture (RA), with a top -down in early stages. The process is carried out in the Domain Analysis (DA) stage in the context of Domain Engineering (DE), contemplating and describing the activities, artifacts and support tools that support it.

Keywords: Software Factories; Software Systems Families; Model-Driven Engineering; Reuse; Transformations; Domain Quality View.

1 Ingeniero de Sistemas de la Universidad Bicentenaria de Aragua (UBA), egresada de la Universidad de Oriente (UDO) Núcleo Anzoátegui. Actualmente Tesista Doctoral en Ciencias de la Computación en la Universidad Central Venezuela (UCV), Caracas. Campos de investigación: la Ingeniería de Software y Desarrollo Industrial de Software. Departamento de Informática, U.P.T.P. "Luis Mariano Rivera", Carúpano, Venezuela. Correo electrónico: aleidysarraiz@gmail.com

2 Licenciado en Computación en la Universidad de Carabobo (UC), Magister en Tecnología Educativa de la Universidad Nacional Experimental de las Fuerzas Aéreas (UNEFA), Doctor en Computación de la Universidad Central de Venezuela (UCV). Campos de investigación: desarrollo y evaluación de software educativo, tecnología de computación y líneas de producción de software en la Facultad de Ingeniería de la Universidad de Carabobo. Departamento de Computación, Facultad de Ingeniería, Universidad de Carabobo, Valencia, Venezuela. Correo electrónico: vesteller@gmail.com

I. Introducción

Las fábricas del Software (FS), representan una de las alternativas más vanguardistas en lo que al desarrollo de sistemas de software (SS) se refiere, en especial si se considera su abordaje de industrialización para producir las Familias de Sistemas de Software, que son un conjunto de aplicaciones o productos que poseen una arquitectura y características comunes y están referidas a un dominio en particular, en una familia a los productos individuales o específicos se les conoce como miembros. Un *dominio* representa el conjunto mínimo de propiedades que describen con precisión una familia de problemas para los cuales se requieren aplicaciones.

Es fundamental que el desarrollo de los productos que se obtienen de las Familias de Sistemas de Software sea de calidad, aspecto que las FS no definen de forma concreta y que se aborda en el trabajo publicado por Arraiz y Losavio (2015)[1], aunado a esto es importante destacar que desde la aparición del enfoque de FS, se complementan los procesos de las Líneas de Productos de Software (LPS)[2], siendo las LPS sistemas con características comunes, que permiten configurar, a partir de elementos reutilizables o existentes (activos), sistemas extensibles, procesos y contenidos para el desarrollo y mantenimiento de las variantes de un producto de software con el objetivo de sacar el mayor provecho posible de los elementos comunes y así reducir el tiempo, esfuerzo y complejidad de construir y mantener los productos de software. La diferencia principal entre LPS y FS es que este último enfoque de desarrollo se basa en la incorporación de un *Esquema* y una *Plantilla* para establecer los activos de Software, dicha Plantilla depende del dominio y se instancia para obtener los productos concretos. De forma más detallada lo anterior significa que la plantilla posee las configuraciones validas probadas para la construcción de un miembro específico perteneciente a una familia lo cual facilita el desarrollo. El alcance de las FS es mucho mayor cuando se utiliza con la Ingeniería de Modelos (IM) también conocida como Desarrollo de Software Dirigido por Modelos (DSDM) que es un enfoque clave porque utiliza modelos en varios de los pasos del ciclo de vida de desarrollo de software: modelos de requisitos, modelos de análisis y diseño, y modelos de código, lo que facilita la comprensión para el desarrollo de software. Estos modelos también se consideran activos y son reutilizables para cualquier otro dominio con características similares.

Uno de los elementos más relevantes en el desarrollo de software bajo el enfoque de FS es el *Esquema*, que define los puntos de vistas que son necesarios para la construcción de un tipo de producto de software o miembro perteneciente a una Familia de Sistemas de Software. Estos definen cómo construir y utilizar una vista; la información que debe aparecer en ella; y estas a su vez se definen como una instancia de ellos que describen ciertos aspectos de un SS, tales como la arquitectura del negocio, de los datos, de la aplicación y el de tecnología entre otros, es decir, a diferencia de las LPS viene a representar un contenedor de las configuraciones validas para los productos específicos. En detalle, una vista es una representación de todo el sistema de software desde una determinada perspectiva establecida por los involucrados en el desarrollo de la Familia de Sistemas de Software[3]. Cada uno de estos puntos de vista son el inicio para identificar el Núcleo (*del Inglés Core)* de artefactos que permitirá el camino más eficiente para producirlos porque contiene los mecanismos para expresar las vistas, en otras palabras referenciándolos su despliegue depende de los componentes estructurales proporcionados por las vistas en el *Esquema* que posteriormente se implementan en la *Plantilla,* que no es más que la colección de todos los activos definidos en el.

El *Esquema* contiene los requisitos comunes que poseen los distintos productos de la Familia de Sistemas de Software y estos se establecen desde las etapas tempranas del desarrollo específicamente en el Análisis del Dominio (AD) de Ingeniería del Dominio (ID), en la LPS contemplada en la FS. Es importante destacar que el desarrollo de las LPS generalmente, sigue dos procesos esenciales, Ingeniería de Dominio e Ingeniería de Aplicación (IA)[2]. En el primero, se construyen los activos centrales o fundamentales, entendiéndose por activos (*del inglés assets*) los componentes o productos de software diseñados para ser utilizados en el desarrollo de diferentes sistemas o aplicaciones; es decir, todos los artefactos que se generen del proceso de desarrollo de software.

En IA, se deriva cada producto específico a partir de los activos creados. La ID captura información y representa el conocimiento sobre un dominio determinado, con la finalidad de crear activos de software reutilizables en el desarrollo de cualquier nuevo producto de software se compone como toda disciplina de las siguientes fases: análisis del dominio, diseño e implementación[2]. La etapa de análisis es ''el proceso donde la información utilizada en el desarrollo del sistema de software es identificada, capturada, y organizada para que resulte reutilizable (para crear activos) en la construcción de nuevos productos»[2]. En esta investigación se sigue sólo lo correspondiente al proceso de ID específicamente para la fase o etapa de AD.

Por otra parte, es importante destacar que existen diferentes enfoques para definir y documentar las arquitecturas de software, sin embargo, todos esos enfoques pueden ser clasificados usando dos dimensiones ortogonales: dirección y énfasis; estas alternativas de dirección son top-down y bottom-up y las alternativas de énfasis son: infraestructura y aplicación. La dirección top-down (también en la literatura conocida como descendente o proactiva): está basada en los requisitos de la arquitectura, se define a partir del conocimiento del dominio para obtener la infraestructura común requerida para una familia de SS, a través de las iteraciones de refinación de procesos[4]. El enfoque top down se enmarca desde la ID ya que hace énfasis primero en la infraestructura común y se puede realizar como un desarrollo de forma incremental, que es lo considerado en esta investigación. Además, el hecho de tomar en cuenta requisitos globales del dominio en la VCD para el enfoque es de mayor relevancia para la construcción de una AR más Flexible y realista lo cual se traduce en una ventaja positiva para la utilización de este método.

El *Esquema* presenta la estructura de una FS. Además, desarrolla la Arquitectura de la Línea de Productos (ALPS) que es una arquitectura genérica instanciable, también denominada Arquitectura del Dominio o Arquitectura de la Familia Sistemas de Software, para esta investigación considerando la termología que se establece en el enfoque se concuerda denominarla ALPS, esto basado en el modelo que describe todos los elementos de las FS, por lo cual puede apoyar la construcción arquitectural no sólo de un sistema, sino de muchos, tiene que ser coherente, modular y extensible para toda la Familia; mientras que la Arquitectura de Referencia (AR) es una plantilla o esqueleto genérico para derivar productos concretos de una Familia de Sistemas de Software; y en las FS se define como una instancia de la ALPS, es decir, se considera que esta proporciona un mayor nivel de detalle respecto a los elementos orientados al producto específico para su desarrollo. Se puede contrastar esta definición señalando la ALPS como meta modelo que deriva a la AR que sería el modelo.

Al mismo tiempo, en el *Esquema* es donde se incorpora la Vista de Calidad del Dominio (VCD)[5], y se realiza la selección del estilo arquitectónico para construir las Arquitecturas Candidatas (AC) de la ALPS. La AC es la arquitectura inicial de alto nivel donde se describen los elementos importantes de la Familia de Sistemas de Software, que será objeto de numerosos cambios durante el desarrollo. Otro punto a considerar, son los requisitos comunes que se generan desde el modelo de negocios y que también incluye a los requisitos variables para todos los productos de software y se desarrollan a partir de la ALPS (ver Fig.1).

En lo que respecta al Núcleo Central de los Activos de software contenido en el *Esquema* se encuentra constituido por los activos funcionales y los no funcionales. Los activos funcionales son aquellos principales percibidos de manera explícita por los involucrados en el desarrollo de software y los no funcionales son aquellos que implícitamente están asociados a los funcionales[6], estos por ende comprenden el dominio de los Productos de Software y contienen los Requisitos de Calidad (RC), que son componentes presentes en el software sujeto a ser medido que están asociados a los Requisitos No funcionales (RNF); y se consideran los que son relevantes para la AR[7], que se debe generar a través de la asociación los puntos de vistas, y las vistas desde los intereses de los involucrados en el desarrollo de software con la correspondiente representación y documentación para integrar las FS con la VCD[1], y así establecer las garantías de que los Productos de Software de la familia a construir con este enfoque cubran los aspectos relativos a los estándares de calidad que para

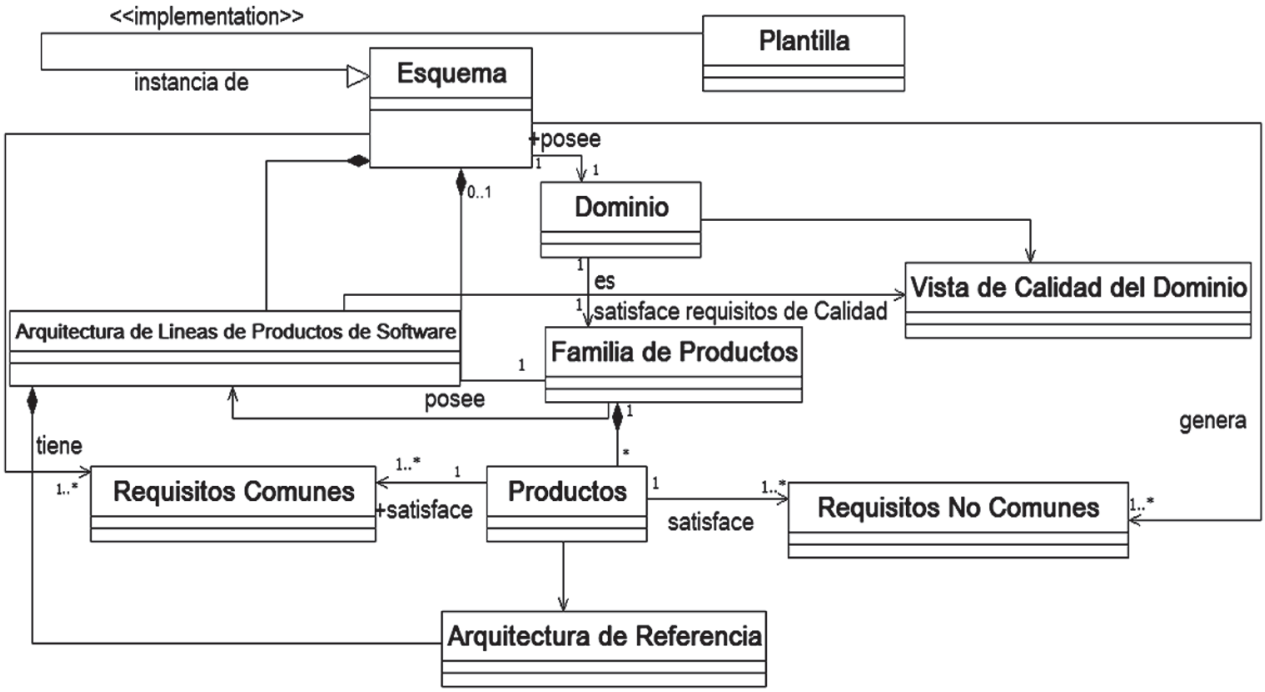

Fig. 1. Modelo FS asociado a VCD[1]

este trabajo es el ISO/IEC 25010[6] (ver Fig. 2), los aspectos relativos a dicha integración se profundizan y se incorporan otras actividades en esta investigación.

De acuerdo a lo explicado previamente tomando como base esta concepción del *Esquema* con VCD, y considerando que las FS están soportadas por la IM, surge abordar en este trabajo la premisa de cómo se pueden realizar transformaciones de modelos a partir de estos enfoques, para obtener la AR que será posteriormente implantada en la Plantilla de la FS y que servirá para crear un producto especifico desde el AD en el marco del proceso de ID.

La IM ofrece y está estrechamente ligada a los Lenguajes Específicos de Dominio *(DSLs por sus siglas en inglés*), y los modelos han sido usados para documentar las arquitecturas de los s*istemas o productos de software* y sus estructuras por mucho tiempo. En general, las FS utilizan los conceptos fundamentales de la IM, como son: la abstracción y la reutilización de los modelos que para el enfoque de FS representan activos de software[8]; no hay cambios importantes porque estos conceptos están relacionados con la terminología básica utilizada por el estándar MDA *(del inglés Model Driven Architecture)* de la OMG (*Object Management Group)*, que es equivalente al enfoque general del DSDM o de la IM. Es así como en[8] señalan que *MDA* establece tres niveles de abstracción sobre los que se diseña e implementa un sistema de software: *Computational Independent Model* (*CIM*), *Platform Independent Model* (*PIM*) y *Platform Specific Model* (*PSM*).

Cabe destacar en referencia a MDA que los perfiles UML (*del inglés Unified Modeling Language*) son usados ampliamente para crear esos modelos, un perfil de UML representa una extensión de un subconjunto de UML orientada a un dominio particular, utilizando los conceptos que incorpora el mecanismo de extensión de UML: estereotipos, restricciones y valores etiquetados. Como resultado se tiene una variante de UML para un propósito específico. En la actualidad existen innumerables perfiles adoptados por la OMG para diversas plataformas.

En lo que respecta a las transformaciones entre modelos que destaca MDA, son procesos que convierten los modelos de un sistema a otro modelo

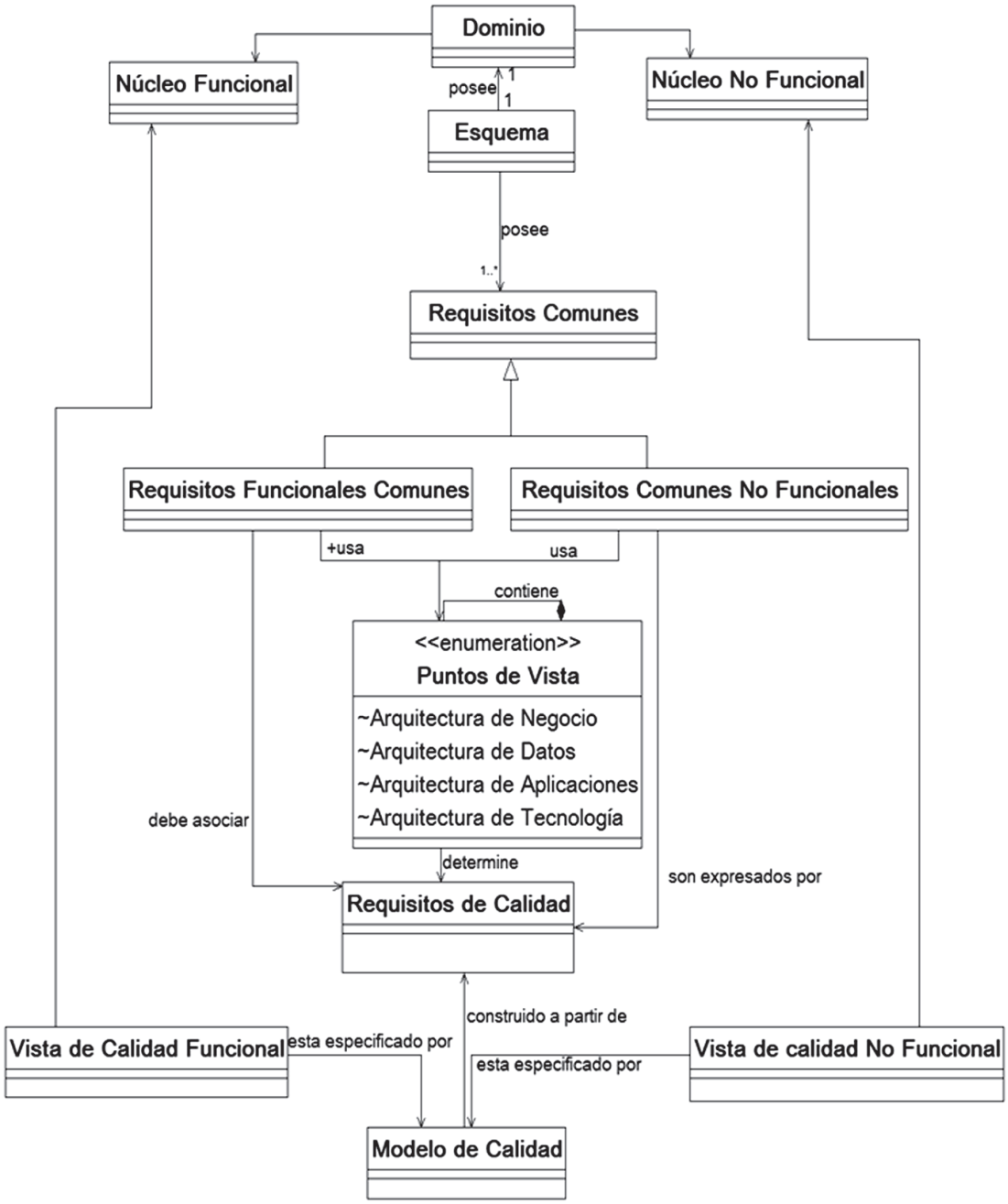

Fig. 2. Modelo de Activos Funcionales y No Funcionales de VCD para Esquema de FS[1]

del mismo sistema. Esto se realiza a través del establecimiento de un conjunto de reglas que describen cómo un modelo expresado en un lenguaje origen puede ser transformado en un modelo en un lenguaje destino. Así pues, para definir las transformaciones entre modelos es necesario:

- Seleccionar el tipo de transformación y el nivel de abstracción que requiere para el EFS incluyendo su VCD asociada.
- Identificar el lenguaje de definición de transformaciones más adecuado a utilizar.
- Seleccionar la herramienta MDA que permita implementar los modelos y las transformaciones de forma automática.

En MDA el proceso de desarrollo de software es de transformación iterativa e incremental, lo que se aplica también a las FS, donde cada paso

transforma un CIM en PIM del sistema considerado, a otro nivel en un PSM del nivel siguiente. Esto se realiza hasta que se alcanza la implementación final del sistema, con la peculiaridad de que cada PSM que es resultado de una transformación puede convertirse en el PIM para la transformación siguiente, es decir, se convierte en el modelo inicial para realizar la nueva transformación, con respecto a otra plataforma o nivel de abstracción.

Tanto los enfoques FS como IM ofrecen ventajas y desventajas para el desarrollo de software según lo evidenciado en estudios realizados en[9]. Sin embargo, mientras que la IM se centra principalmente en el modelado, en contraste, el enfoque de FS, concreta la definición de productos de software basados en Familias de Sistemas de Software, definiendo puntos de vistas desde los cuales los miembros de una familia de productos son modelados[9]. La IM favorece ampliamente la reutilización, ya que los modelos se pueden reutilizar una vez definidos; éstos, así como las transformaciones entre ellos, pueden también ser considerados como activos de software dentro del proceso de desarrollo con FS.

El objetivo principal de este trabajo es presentar el enfoque FS con VCD en el contexto de la IM, especificando su modelo y también el proceso IMFS&VCD, para realizar las transformaciones del *Esquema* con VCD y así lograr la obtención de la AR en el contexto del AD de la ID.

Además de esta introducción, este trabajo se encuentra estructurado de la siguiente forma: en la sección II se describen los elementos generales de las FS (*Esquema y Plantilla*), se contextualiza el *Esquema* en la etapa de AD de la ID y se describe el enfoque de FS con VCD. La sección III, aborda las transformaciones de modelos para *el Esquema*, se analizan los tipos y niveles de abstracción para las transformaciones, se presenta el modelo de FS con VCD que incorpora las transformaciones. En la sección IV, se presenta el proceso que se propone IMFS&VCD, detallando las actividades, artefactos de entradas y salidas, y las herramientas de soporte que lo sustentan. Por último, en la sección V se presentan las conclusiones y perspectivas futuras derivadas de la presente investigación.

II. Fábricas de software, vista de calidad del dominio e ingeniería de modelos

Las FS, promueve un enfoque de desarrollo basado en las LPS que fomenta la reutilización de activos de software, específicamente su modelo de procesos contempla de acuerdo a[10], tres ideas claves: un *Esquema*, una *Plantilla* y *un entorno de desarrollo extensible*.

El *Esquema*: es un modelo interpretado por seres humanos y herramientas que describen los productos de trabajo y flujos utilizados para obtener los activos para la promulgación de estos, de una familia específica de SS en un dominio dado. La *Plantilla* representa la instanciación o implementación del *Esquema*; básicamente, es la colección de todos los activos definidos por los *puntos de vistas* que lo conforman. Entendiendo por *puntos de vistas* el abordaje de las incumbencias, preocupaciones o intereses de los diferentes participantes *(del inglés stakeholders)* involucrados en el desarrollo, la descripción de diferentes aspectos de la arquitectura y el diseño de una línea de productos; además, proporcionan una separación de intereses y están organizados jerárquicamente; constituyen activos que pueden dividirse en varias categorías. El *entorno de desarrollo extensible*: se encuentra representado por un conjunto de componentes, herramientas y procesos específicos para crear un producto de software. Es decir, proporciona los elementos para manejar y automatizar fácilmente las tareas de la fase de desarrollo del producto concreto.

En el *Esquema* se desarrolla el proceso de ID de la LPS, y tal como lo indica la definición representa un modelo en el cual se describen los elementos conceptuales presentes para el AD, que posteriormente serán implementados en la *Plantilla* para obtener productos de software específicos. Como ya se ha analizado, el *Esquema* está organizado en base a los puntos de vistas cada uno de ellos describe el sistema desde diferentes perspectivas, como por ejemplo, el comportamiento en tiempo de ejecución, estructura lógica y ejecución, estructura de los componentes, o la distribución física en los nodos de la red, cada punto de vista describe algunos aspectos de un sistema y cómo se abordan las inquietudes de los interesados o involucrados. A partir de la definición de estos elementos del

Esquema se incorporó la VCD como un punto de vista que debe aplicarse para cada vista definida por los involucrados para garantizar que los productos de software que se obtengan en la FS contemplen los RC.

La VCD permite la trazabilidad de los Requisitos Funcionales (RF) con los RNF también conocidos como RC y está representada por un modelo de calidad del dominio construido utilizando escenarios de calidad, considerando no sólo los RNF sino también los RF, ya que contribuye a identificar características de variabilidad. Este enfoque realiza una clara distinción entre los RF y los RNF lo cual facilita la construcción de la ALPS, de una FS, tal como se puede observar en la Fig 3. Tomada de[1].

En la figura anterior la VCD representa un punto de vista más, que debe ser considerado primordialmente para justificar y respaldar que la producción de software en las FS contemple los aspectos relativos a la calidad, esto permite reforzar que el enfoque estudiado es el único en la creación de una metodología para una Familia de Sistemas de Software con una arquitectura específica en un dominio particular de la industria. El *Esquema* con la VCD en sí mismos representan un modelo derivado donde se garantiza que el conocimiento del dominio capturado en el *Esquema* puede estar disponible para transformaciones de modelos bajo la IM, y otros procesos, que se establecen en este trabajo.

La IM[9] hace frente a la creciente complejidad del software, sin embargo se deben introducir mayores niveles de abstracción durante el proceso de desarrollo, las tecnologías de IM combinan abstracciones específicas de dominio (descritas mediante metamodelos que expresan la ontología del dominio) que permiten definir la estructura de la aplicación, su comportamiento, requisitos para dominios particulares, motores de transformación y generadores que procesan determinados aspectos de los modelos para sintetizar varios tipos de artefactos[11].

Un metamodelo es una especificación para la cual el sistema estudiado se representa a través de un cierto lenguaje de modelado[12]. Éste a su vez, está conformado por un conjunto de posibles modelos que conforman el respectivo metamodelo, incluyendo la sintaxis abstracta, representada por una o más sintaxis concretas y que satisfacen una semántica dada. Además, lo pragmático de los lenguajes de modelado es que dan soporte y guían el desarrollo, así se presenta la relación entre metamodelos y modelos tal como se indica en la figura 4 (ver Fig.4).

Cabe destacar que si bien es cierto que las FS se consideran un enfoque de desarrollo dirigido por modelos propuesto por Microsoft, no contrapone la propuesta de IM llevada a cabo a través de las

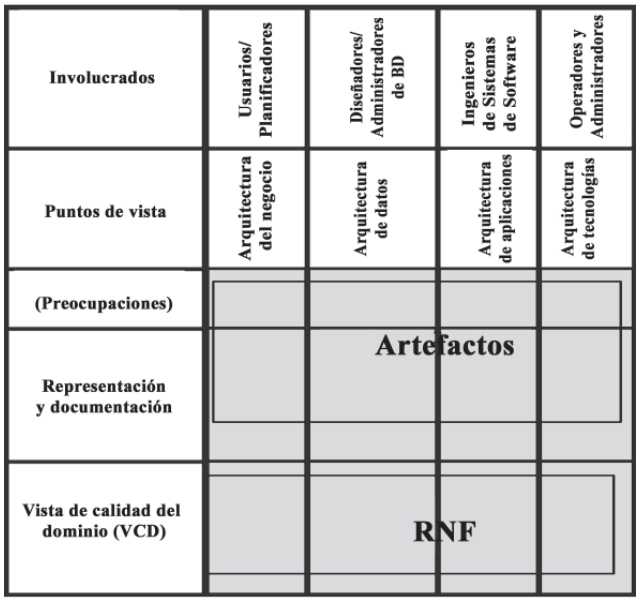

Fig.3. Asociación del EFS con la VCD[1].

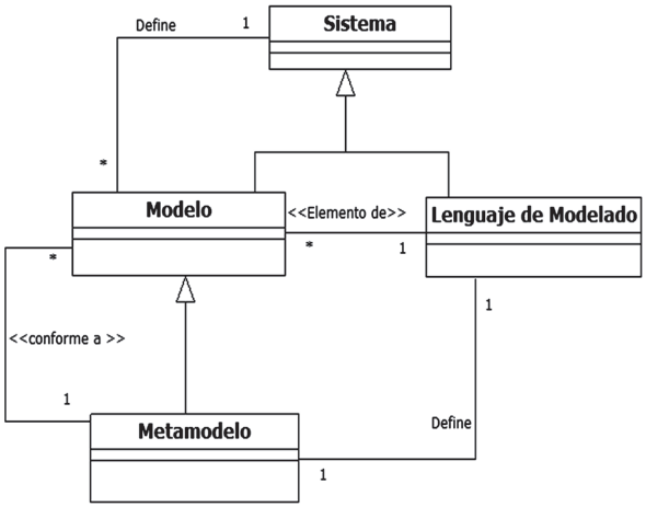

Fig. 4. Relación entre Metamodelo y Modelo[9]

otras corrientes, como por ejemplo MDA de la OMG, de hecho estas se pueden complementar y es lo que se aborda en la investigación, que las FS trasciendan de manera general independientemente del origen y sean promovidas en la comunidad de tecnologías libres.

Sin embargo, se debe analizar y especificar el nivel de abstracción más adecuado y para ello se toman en cuenta las siguientes consideraciones[12]:

- Para las FS no se establece claramente el nivel de abstracción.

- Tampoco se definen las disciplinas de Ingeniería de Software que las soportan y que conformarían el modelo de procesos detallado.

- No se establece el tipo de transformación de modelos que pueden abarcar, pero si definen como lenguajes para el modelado los DSLs, y como lenguaje de meta modelado UML, se puede aplicar para cualquier dominio y pueden ser soportadas por diversas herramientas.

III. Las transformaciones de Ingeniería de Modelos para el esquema de fábricas de software

El enfoque de IM hace uso de los lenguajes de modelado para especificar los modelos y obtener el nivel de abstracción para el desarrollo de las aplicaciones finales. Un Producto de Software tal y como se define en las FS es un sistema compuesto por integración de plataformas, artefactos derivados de las transformaciones, artefactos elaborados directamente por los desarrolladores y de manera eventual modelos ejecutados en un contexto particular de plataforma de software, tomando en consideración que tanto las aplicaciones de software, artefactos, plataformas y modelos conforman un sistema.

Las transformaciones son procesos para convertir un modelo a otro en un mismo sistema[7], ya que en ellas se establece un conjunto de reglas que solo pueden basarse en la construcción de metamodelos y esta es la principal propuesta del Enfoque de IM. Entonces para definir una transformación se debe tomar en cuenta lo siguiente:

- Las plataformas de software comprenden un conjunto integrado de elementos computacionales que permiten desarrollar tipos de Productos de Software, frecuentemente estos elementos poseen diferentes funcionalidades a través de los distintos mecanismos de reutilización y extensibilidad.

- La generación de artefactos que pueden construirse durante el proceso y el tiempo de desarrollo, mientras que otros pueden realizarse en tiempos de ejecución.

- Dos tipos de transformaciones son consideradas en el enfoque de IM, una es model-to-text transformations (M2T) la cual permite generar o producir artefactos de software usualmente: código fuente, XML y otros archivos de texto. La otra model-to-model transformations (M2M) que permite transformar modelos en otros modelos que abarcan diversos aspectos del dominio para satisfacer las distintas necesidades de los involucrados (del inglés stakeholders) en el desarrollo, cabe destacar que estas transformaciones son especificadas por distintos lenguajes propuestos por diferentes paradigmas de modelado como QVT, Acceleo, ATL3 VIATRA, DSLTrans, entre otros.

En general el desarrollo bajo el enfoque de IM sigue un proceso de abstracción donde se conjugan dos perspectivas con una marcada diferencia:

(1) Considerando los aspectos relativos al dominio de la aplicación, para este caso será el de la Familia de SS en contexto de desarrollo de las FS.

(2) Haciendo referencia a los aspectos de interés para los usuarios finales, debido a esto el proceso se enfatiza básicamente en la representación más adecuada de ese Dominio, para poder transformar esa representación en la etapa de implementación con una tecnología y herramientas concretas[12].

Se enfatiza en el proceso de ID, ya que por lo general el dominio de un problema nada tiene que ver con la tecnología final de software a emplear, por ello la IM permite una separación conceptual entre la especificación funcional de los SS y su

implementación, en la que los modelos y las transformaciones pueden desarrollarse de manera independiente, adquiriendo una mayor importancia en todo el proceso de desarrollo de software[13].

Con respecto a los niveles de abstracción se abordan los promovidos por la IM, en el estándar conocido como MDA, ya que ésta se organiza en Modelo Independiente de Computación (por sus siglas en inglés CIM), Modelos Independientes de la Plataforma (por sus siglas en inglés PIM), Modelos Específicos de la Plataforma (por sus siglas en inglés PSM), y las transformaciones que se puedan generar entre estos modelos.

Una forma para facilitar las transformaciones entre modelos es la identificación de elementos en este caso en el Esquema que es donde se desarrollan el Núcleo de Activos del Dominio (Core Assets), que deben transformarse de una manera concreta para una plataforma destino. Se debe contemplar también el Lenguaje de Especificación de Dominio (por sus siglas en inglés DSL) que se debe usar en el contexto que indica el marco conceptual de las FS.

Es por ello por lo que el objetivo primordial de la IM es la creación de un producto de software que forma parte de una Familia de Sistemas de Software, a través de una o más transformaciones, dicho producto puede ser una aplicación completa o cualquier activo de software que pueda ser usado para la construcción de otros productos pertenecientes a la Familia de SS de un dominio estudiado.

A Continuación, se muestra el modelo para definir las transformaciones en el marco del enfoque con VCD, donde se expresa que ALPS, para las FS, como ya se definió anteriormente; está soportada por un DSL, destacando que la ALPS debe ser siempre relativa a una plataforma para la ejecución del sistema (ver Fig. 5).

El conjunto de productos que se crean a partir de la ALPS debe ser totalmente reutilizables para cualquier otro producto de la Familia de Sistemas de Software. Es por ello por lo que la ALPS en el contexto de la FS, debe ser lo suficientemente flexible para que permita la representación de la variabilidad entre los diversos productos que conforman la FSS, por lo cual es considerada la Arquitectura del Dominio en el contexto del desarrollo con el enfoque de FS como se ha analizado previamente.

Entonces, para esta investigación, y de acuerdo a las consideraciones abordadas y analizadas respecto a la IM, el proceso que se propone contempla lo siguiente:

1. Tipo de Transformación: se selecciona el tipo de transformación de acuerdo a lo analizado anteriormente, que para este trabajo es model-to-model (M2M). También llamada transformación horizontal. Permite cambiar la estructura de la información, pero se mantiene en el mismo nivel de abstracción. Se utiliza para la evolución de los modelos donde se puede distinguir entre refinamiento y optimización, esto contribuye al establecimiento de un modelo que garantice el cumplimiento de la calidad en la AR seleccionada de acuerdo a las especificaciones del dominio demandadas para la familia de productos. Las optimizaciones cambian la estructura de la información con el fin de mejorar algunas de las características de un modelo.

2. Reglas de Transformaciones: Una regla de transformación según lo señalado en[15] consta de dos partes: un lado izquierdo y un lado derecho. El lado izquierdo accede al modelo fuente también denominado origen, mientras que el lado derecho se expande en el modelo destino. Para este trabajo se utilizan ambos, con las transformaciones de modelo de los elementos usando patrones en términos gráficos como el de BPMN, y su equivalencia con los elementos de UML, ya que estos se pueden representar usando sintaxis abstracta o concreta del modelo origen o lenguaje del modelo destino, y la sintaxis puede ser textual y /o gráfica.

3. Nivel de Abstracción: Alto (aspectos relativos al dominio). Modelo Independiente de Computación CIM a Modelo Independiente de Plataforma PIM. Este nivel es el que permite caracterizar el dominio del cual se extrae toda la información.

4. Estándar de IM: MDA que implementa el enfoque de modelos con sus especificaciones

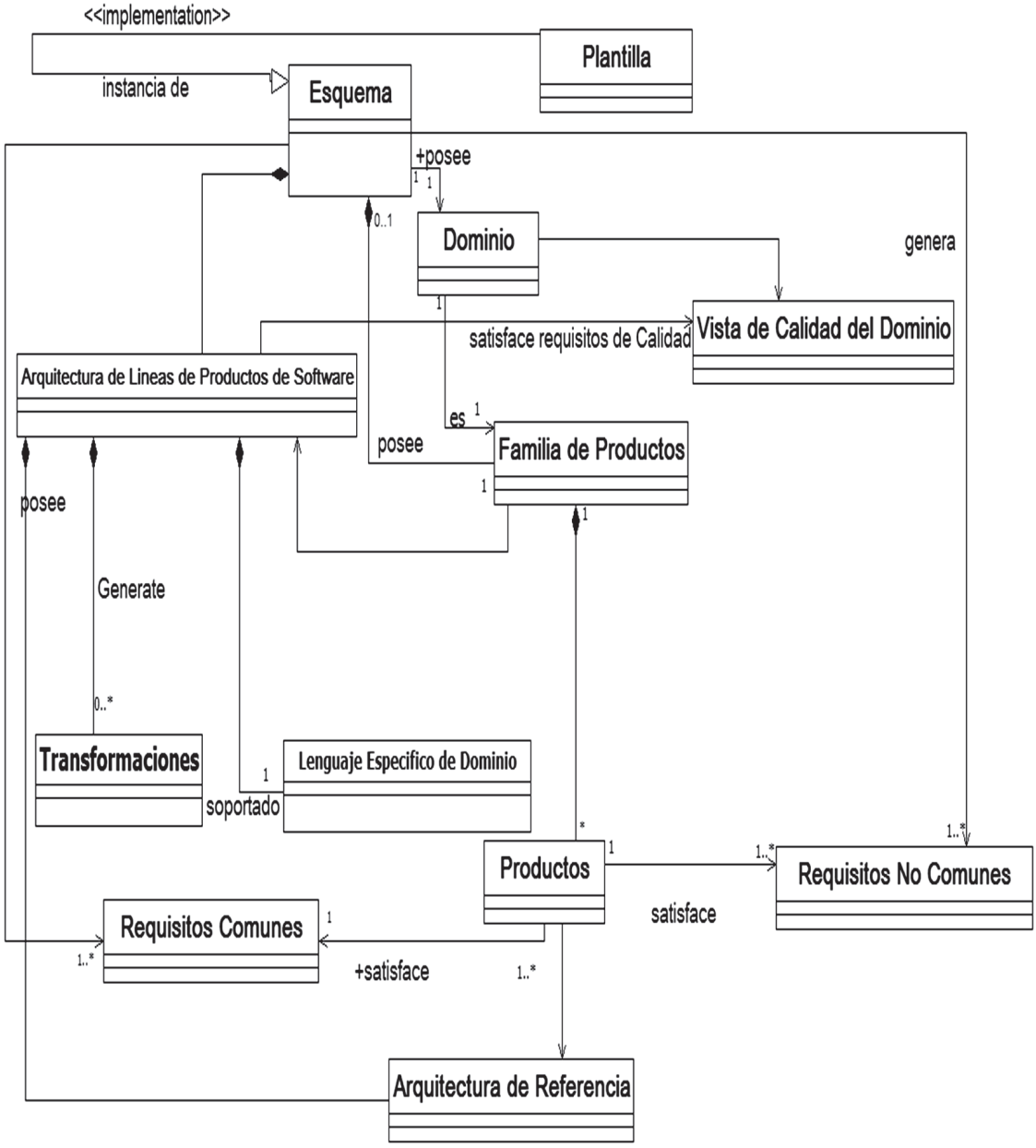

Fig. 5. Modelo de FS con VCD para las Transformaciones.

de Consulta/Vista/Transformación, conocidas como el Lenguaje QVT. Además de ser compatible con UML por ser ambas propuestas de la OMG.

Es importante considerar que el estándar MDA tal como evidencia en[15], acerca de la clasificación de las alternativas de enfoques de IM, donde se señala que permite realizar implementaciones en altos niveles de abstracción y abarca disciplinas de Ingeniería de Software que van desde el análisis y diseño hasta la implementación , maneja los tres niveles definidos de abstracción PIM-PSM-CIM, también refiere como lenguaje de modelado a UML y perfiles de UML, asocia dos tipos de transformación que se pueden usar para el enfoque M2M y M2T, y como lenguajes para realizar dichas transformaciones QVT, como lenguaje de metamodelado sugiere como alternativas MOF, EMOF, UML. Se puede aplicar para cualquier dominio y de manera general como herramientas que pueden soportar todo el proceso se encuentran:

EMF (del inglés Eclipse Modeling Framework) y para las FS Microsoft Visual Studio.

MDA, también aporta una mejor portabilidad debido a la separación que hace del conocimiento del dominio y la trazabilidad hacia una tecnología de implementación específica, aumenta la productividad, mejora de la calidad debido a la reutilización de los componentes de software o activos y por supuesto mejora consistencia y trazabilidad entre los modelos y el código[16].

Otro aspecto de relevancia que se ha estudiado, por ejemplo, en[16], es a pesar de que el nivel CIM está contemplado en la fundamentación teórica de MDA, este no muestra detalles de las estructuras de los Sistemas de Software, viene a representar el Modelo de Dominio y permite modelar los requisitos de los Sistemas de Software. Por lo cual, resulta de especial utilidad cuando se crea un vocabulario común al Dominio de las aplicaciones. Puede utilizar un lenguaje para modelar procesos de negocios que no tiene que ser precisamente UML, aunque éste pudiera ser perfectamente derivado usando MOF (del inglés MetaObject Facility). El CIM se manifiesta a través de los procesos de negocios y deja establecidas las relaciones entre trabajadores humanos o no, sus interacciones, responsabilidades y roles.

La mayoría de los trabajos orientan sus investigaciones con MDA al caso de las transformaciones del PIM a PSM y de PSM a código, para lo cual existen varias alternativas de herramientas que la realizan. Sin embargo, es indispensable señalar que la definición de CIM y su transformación al PIM ha sido la menos estudiada, debido a la naturaleza de los modelos involucrados en las etapas iniciales de los procesos de desarrollo. Se parte de la idea de que las transformaciones tienden cada vez más hacia una homogenización para la información a plasmar en CIM y/o PIM, como también las transformaciones entre estos modelos. Es por ello el interés de considerar este aspecto en esta investigación.

Para realizar una transformación entre modelos se debe contar con información de los metamodelos que describen las representaciones de cada elemento y definen las restricciones que estos deben cumplir. De esta manera entonces se asume que una transformación es una correlación entre modelos representados por metamodelos.

También es indispensable señalar que para utilizar el enfoque MDA, se resaltan algunas características que se deben tomar en cuenta para realizar las transformaciones de manera rigurosa, que son: el ajuste de las transformaciones, la trazabilidad, la consistencia incremental y la bidireccional[17].

El ajuste de las transformaciones permite el control sobre el proceso, se definen manualmente los elementos del modelo que va a ser sometidos a las reglas de transformación, para complementar este método, se asignan condiciones a cada regla que contiene la descripción para aplicarla, y éstas se pueden parametrizar para cambiar la transformación. La trazabilidad, define que se pueda conocer los elementos fuentes que se generaron en el elemento destino. La consistencia incremental se utiliza cuando se requiere refinar algún elemento fuente y esto implica la regeneración del elemento destino. La bidireccionalidad, señala que las transformaciones se realicen de acuerdo con una única regla, lo cual contribuye a su operación en ambas direcciones fuente y destino.

En el contexto de este trabajo se abarca el desarrollo de Familias de Sistemas de Software con el enfoque de FS incorporando los aspectos de calidad que garanticen desde el estudio del dominio que los Productos de Software desarrollados sean confiables, se aborda la IM desde el modelo CIM, pues este nivel es el menos explorado en la comunidad académica.

El CIM es primordial para estrechar las diferencias entre los expertos del dominio y los requisitos, por una parte, y los que construyen y diseñan artefactos de software por la otra parte.

IV. Proceso para realizar las Transformaciones de Ingeniería de Modelos para el esquema de Fábricas de Software con Vista de Calidad del Dominio IMFS &VCD

En esta sección se describe detalladamente el proceso IMFS& VCD cada uno de sus pasos y/o actividades para la obtención de la AR, a partir del EFS con VCD y la incorporación de

transformaciones con el enfoque de IM, utilizando lineamientos del estándar MDA.

Considerando lo señalado anteriormente y los estudios analizados en[18], una de las formas de representar el CIM es el Modelo y Notación de Proceso de Negocios (*del inglés BPMN*), y se adopta para modelar el CIM de una Familia de SS. BPMN es ampliamente utilizado en la comunidad empresarial, científica y académica para el modelado de los procesos básicos de negocios lo que abarca su análisis, especificación y diseño.

El CIM no tiene ninguna vinculación con conceptos computacionales dado que posee un nivel de abstracción más alto, para realizar la transformación se adopta la propuesta señalada en[19], donde se describe la correspondencia entre artefactos de un modelo origen CIM hacia un modelo destino PIM. A continuación se presenta cada paso con las actividades, y todos los artefactos de entrada y de salida para el proceso IMFS& VCD:

Paso 1. Estudio y conocimiento del Dominio:

***Entradas*:** *Descripción de forma textual y gráfica del problema en el contexto de la Familia de Sistemas de Software, para la elaboración de un modelo de negocios específico del dominio.*

Actividad 1.1. *Se deben especificar los procesos inherentes al negocio siguiendo lineamientos de la Gestión de procesos de Negocios de una organización que comprenden: Descubrimiento, Análisis, Desarrollo, Monitoreo y Optimización[*20]. *Esto con la finalidad de identificar la información del dominio, para comprenderla, especificando todos los detalles de Requisitos Tecnológicos y estableciendo las Reglas de Negocio (RN), lo que permitirá su análisis, modelado y asociación con RF y RNF.*

Salidas: *Modelo de Negocio, utilizando la Notación de Modelado de Procesos de Negocio BPMN. Este modelo se considera como el modelo origen para iniciar el proceso de transformación utilizando la visión del enfoque de IM. También se obtienen los RF y RNF y las RN del dominio perteneciente a la Familia de Sistemas de Software.*

Paso 2. Descripción para las Transformaciones de Modelos

Entradas: *Modelo de Negocio especificado en BPMN (Modelo Origen), Reglas de Negocio, RF y RNF del dominio.*

Actividad 2.1. *Identificar las Reglas de Transformaciones (RT) gráficas de BPNM a UML que se utilizan para el subproceso de transformaciones, considerando que ambos lenguajes son de notación gráfica lo cual permite definir de una forma clara las Reglas de Transformación (RT).*

Actividad 2.2. *Describir las RT tomando en consideración las siguientes P**remisas** para establecer la correspondencia, que se establece entre modelo origen para obtener el modelo destino, para pasar del nivel CIM al PIM, con tipo de transformación M2M:*

P1. *Las RT involucran un artefacto de metamodelo BPMN y uno o más artefactos del metamodelo UML.*

P2. *Se consideran transformaciones 1 a 1 (elemento simple de BPMN a elemento simple de UML) o transformaciones 1 a N (elemento simple de BPMN a elemento compuesto o conjunto de elementos compuestos de UML).*

P3. *Se utiliza el metamodelo correspondiente a los diagramas de Casos de Uso UML, ya que este representa el modelo de comportamiento, en sincronía con el comportamiento que establecen los diagramas de procesos de negocio en BPMN.*

Actividad 2.3.*Construir tabla contentiva de especificaciones gráficas de los diagramas de procesos de negocio en BPMN hacia los diagramas de casos de uso UML.*

Actividad 2.4. *Construir el Modelo de casos de uso resultante (Modelo Destino) con todos los Diagramas de casos de uso correspondientes a la descripción de dominio de la Familia de Sistemas de Software estudiado, una vez que se hayan aplicado las RT. Esta actividad se debe realizar de la siguiente forma:*

Identificar actores y su correspondencia derivada de las transformaciones de Diagrama de Procesos de Negocio (DPN).

Identificar escenarios de Casos de Uso, derivados de los DPN.

Identificar relaciones de actores y casos de uso derivadas de los DPN.

Salidas: *Modelos de Casos de Uso (Modelo Destino), resultante de la aplicación de las RT.*

Paso 3. Descripción de RF y RNF para establecer el Modelo VCD de la Familia de Sistemas de Software.

Entradas: *Modelo de Casos de Uso (Modelo resultante del paso anterior).*

Actividad 3.1*. Se deben definir los activos funcionales del dominio:*

Actividad 3.2*. Especificar los principales RF comunes del dominio de la familia de Sistemas de Software.*

Actividad 3.3*. Especificar los RNF, Señalando y describiendo detalladamente los RNF que influyen de acuerdo a la familia de sistemas de software estudiada.*

Actividad 3.4. *Documentar los RF y RNF a través del ERS (Especificación de Requisitos de Software), documento para la especificación basado en el estándar IEEE-830-1998.*

Actividad 3.5. *Consolidar los activos funcionales del dominio de la Familia de Sistemas de Software.*

Salidas*: ERS de la Familia de Sistemas de Software.*

Paso 4. Especificación de Vista de Calidad Funcional del Dominio (VCD).

Entradas: *ERS de la Familia de Sistemas de Software.*

Actividad 4.1. *Expresar cada RF del núcleo funcional de activos, en términos de características, subcaracterísticas y atributos de calidad en concordancia con el estándar de ISO/IEC 25010.*

Actividad 4.2*. Elaborar listado de los RC de acuerdo a cada RF del dominio estudiado, esto complementa los activos funcionales del dominio.*

Actividad 4.*3. Especificar el Modelo de Calidad Funcional (MCF) del dominio de la familia.*

Salidas: *Tabla contentiva de los RF, características, subcaracterísticas y atributos de calidad asociados al dominio de la Familia de Sistemas de Software, Listado de RC de acuerdo a cada RF asociado. Modelo de Calidad Funcional del Dominio de la Familia de Sistemas de Software.*

Paso 5. Descripción de los RNF para definir el Núcleo de Activos No Funcionales

Entradas: *ERS de la Familia de sistemas de software*

Actividad 5.1. *Identificar RNF derivados del dominio estudiado.*

Actividad 5.2*. Conformar el Núcleo de Activos No Funcionales del dominio.*

Salidas*: Listado de RNF del núcleo de activos no funcionales.*

Paso 7. Descripción y Especificación de la Vista de Calidad No Funcional del Dominio

Entradas: *Listado de RNF del núcleo de activos no funcionales.*

Actividad 7.1. *Especificar la Vista de Calidad del Dominio No Funcional (VCDNF) de acuerdo a cada RNF del núcleo de activos no funcional.*

Actividad 7.2*. Definir cada RNF en términos de características, subcaracterísticas y atributos de acuerdo al estándar ISO/IEC 25010.*

Actividad 7.3*. Consolidar el Modelo de Calidad No Funcional del Dominio de la Familia de Sistemas de Software.*

Salidas: *Tabla de RNF en términos de características, subcaracterísticas y atributos de calidad del dominio de la familia.*

Paso 8. Descripción de estilos/soluciones arquitecturales basado en las vistas y puntos de vistas del Esquema

Paso 6. Descripción y Especificación de VCD del Esquema de la Familia de Sistemas de Software

Entradas: *Tabla de RNF en términos de características, subcaracterísticas y atributos de calidad del dominio de la familia.*

Actividad 6.1. *Utilizando el estándar para la descripción arquitectural y desarrollo de vistas y puntos de vistas IEEE-1471-2000, se seleccionan las vistas y puntos de vistas en el desarrollo de la Familia de Sistemas de Software, estas contienen los estilos/soluciones de acuerdo a*

cada perspectiva y a las preocupaciones o incumbencias de los involucrados, para ello se utiliza la tabla de asociación del Esquema de FS con VCD, tal como se describe en la sección II de este trabajo y se ejemplifica en la Fig. 4.

***Actividad 6.2**. De acuerdo a la actividad anterior, se seleccionan los RNF más influyentes de forma directa para la definición de la AR desde la ALPS.*

***Actividad 6.3**. Incluir en la tabla de asociación de Esquema de FS con VCD la representación y documentación de cada estilo arquitectónico de acuerdo a las preocupaciones o incumbencias en correspondencia con cada RNF definido.*

***Salidas:** Tabla de vistas y puntos de vistas del Esquema con VCD (ver Fig.4).*

Paso 7. Descripción de estilos/soluciones arquitecturales basado en las vistas y puntos de vistas del Esquema

***Entradas:** Tabla de vistas y puntos de vistas del Esquema de FS con VCD.*

***Actividad 8.1.** Seleccionar estilos/soluciones arquitecturales de acuerdo a lo obtenido en el paso anterior, Tabla de Asociación del EFS con VCD, representado en la Fig.4, considerando las vistas y puntos de vistas de los involucrados.*

***Actividad 8.2**. Expresar los estilos/soluciones arquitecturales de cada vista y punto de vista con sus correspondientes RC, de acuerdo a ISO/IEC 25010.*

***Salidas:** Tabla de estilos/soluciones arquitecturales.*

Paso 9. Definir la AC para la Familia de Sistemas de Software

***Entradas:** Tabla de estilos/soluciones arquitecturales.*

***Actividad 9.1**. Construir la AC de acuerdo a las combinaciones de estilos/soluciones derivadas del Esquema de FS con VCD, que permita satisfacer las características requeridas presentes, luego del análisis detallado.*

***Actividad 9.2**. Se debe evaluar cada estilo/solución planteado. Para esta actividad se debe realizar una matriz para la verificación de la satisfacción de las características del Dominio por cada estilo/solución seleccionado. Si existiera más de uno que satisfaga el mayor número de características señaladas, se debe acudir a la experticia del arquitecto de software en soluciones ya probadas anteriormente.*

***Actividad 9.3**. Se concreta la AC con la combinación de los estilos/soluciones resultado de la matriz de verificación aplicada y se representa a través de un diagrama de componentes de UML.*

***Actividad 9.4.** Se construye el Modelo de calidad asociado a la AC, esto se realiza extrayendo los RC que satisfacen la AC de la tabla donde se asocia el EFS con VCD.*

***Salidas:** AC, Modelo de Calidad de la Arquitectura del EFS con VCD.*

Paso 10. Construcción del Modelo de Calidad del Dominio completo de la Familia de Sistemas de software

***Entradas:** AC, Modelo de calidad de la arquitectura de la Familia (contiene los RC que satisfacen la AC), Modelo de Calidad de la Arquitectura del EFS con VCD (contiene los RC que satisfacen la AC).*

***Actividad 10.1**. Esta actividad consiste en plasmar a través de una tabla las características y subcaracterísticas de calidad asociadas al Núcleo Funcional y No Funcional tanto del modelo de arquitectura de la familia (basado en la AC), como el modelo de calidad de arquitectura del EFS con VCD, es decir, la suma de ambos modelos permite establecer de manera definitiva los RNF globales pertenecientes al dominio de la Familia y que son en su mayoría derivados del Núcleo no Funcional.*

***Actividad 10.2**. Se selecciona las características de calidad asociadas al Núcleo Funcional, se debe tomar en cuenta en este tarea que no se dupliquen dichas características, también se deben revisar las sub-características por cada característica del estándar ISO/IEC 25010.*

***Actividad 10.3.** Se clasifican y catalogan todas las características de calidad a través de una tabla donde se detallan Requisitos, estilos/soluciones, características y sub-características, asociadas al Núcleo No Funcional que representan los RNF globales del dominio estudiado.*

***Salidas:** Modelo de Calidad del Dominio.*

Paso 11. Construcción de la AR de la Familia de Sistemas de Software

Entradas: *Modelo de Calidad del Dominio, AC.*

Actividad 11.1*. Para definir la AR: se debe tomar en cuenta todos los componentes establecidos en la AC, así como también todas las características resultantes asociadas al dominio de la familia estudiado. Esta actividad va en concordancia con lo extraído y derivado del modelo de calidad del Dominio basado en la VCD y el conocimiento obtenido en los pasos anteriores:*

Actividad 11.1.1*. Se analizan los componentes de la AC y se verifica la satisfacción de los RNF y RF, con las soluciones seleccionadas.*

Actividad 11.1.2*. Se construye una tabla para verificar y contrastar componentes de la AC, los RNF y RF satisfechos.*

Actividad 11.1.3. *Se selecciona el estilo/solución que satisface todos los RC y se genera el diagrama de componentes UML para la AR resultante.*

Salidas*: AR.*

Para complementar el proceso descrito en los pasos anteriores, se debe considerar que en cuanto al lenguaje de transformación, se sugiere QVT (*del inglés Query/Views/Transformations*) que es un conjunto estándar de lenguajes para concebir las transformaciones de modelos definido por la OMG, y que se utiliza en el marco del estándar MDA, sus siglas son el acrónimo de transformaciones, vistas y consultas. El planteamiento de QVT, se basa principalmente en la definición de un lenguaje para consultas sobre los modelos MOF, es decir, la búsqueda de un estándar para generar vistas que revelen aspectos específicos de los sistemas modelados y finalmente, la definición de un lenguaje para la descripción de transformaciones de modelos MOF.[23]. ver Fig.6).

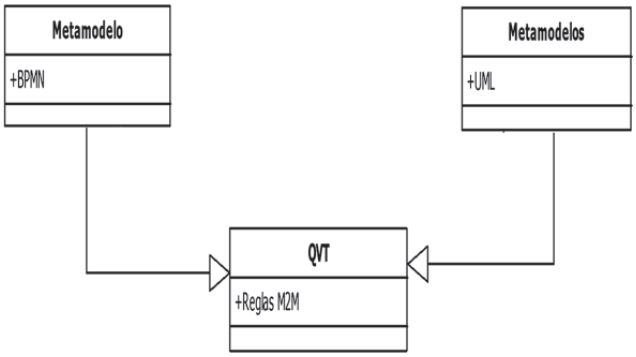

Fig. 6. Implementación con el Lenguaje de Transformaciones QVT para el Proceso IMFS& VCD

V. Conclusiones y Perspectivas Futuras

Cabe destacar que en el contexto de las FS, y en el marco del proceso presentado en este trabajo, los DSLs son usados para describir una parte del software en una notación que utiliza conceptos del dominio en diferentes niveles de abstracción, no sólo los conceptos a nivel de código. En este caso BPMN representa el DSL, ya que permite manejar los conceptos que se establecen en el problema del dominio que se desee estudiar, como pueden ser los actores, objetos de negocios y transacciones[22].

Los DSLs, son las herramientas que permiten abstraer la complejidad de cualquier dominio estudiado, y esto lo hace cuando proveen conceptos y reglas expresados directamente de ese dominio del problema. Los conceptos principales suelen tener su representación en la notación de lenguaje (gráfico o textual), otros conceptos se pueden representar mediante conexiones o propiedades[23]. Algunas de las ventajas que ofrecen los DSL se sintetizan a continuación:

- Los DSLs permiten expresar soluciones usando los términos y el nivel de abstracción apropiado para el dominio del problema. En consecuencia, los mismos expertos de dominio pueden comprender, validar, modificar y a menudo desarrollar programas en DSL.

- Es código auto-documentado.

- Los DSLs mejoran la calidad, productividad, confianza, mantenibilidad, portabilidad y reusabilidad de las aplicaciones.

- Los DSLs permiten validaciones a nivel del dominio. Mientras las contrucciones del lenguaje estén correctas, cualquier sentencia escrita puede considerarse correcta.

Desde la perspectiva de los desarrolladores, los modelos son las primeras clases de artefactos en el desarrollo de cualquier proyecto, incluyendo los que implican el desarrollo a gran escala para una Familia de Sistemas de Software, y en la actualidad se habla de Modelado Especifico de Dominio (*del inglés DSM*) que representa un enfoque adicional que complemente la visión propuesta por la IM.

En el proceso de IMFS& VCD los DSLs son esenciales para la construcción de modelos, es por ello que se utilizan y son parte importante para aplicar las transformaciones de modelos desde el mismo estudio y conocimiento del dominio[24].

La ID abarca la utilización de estos DSLs, y ejercen también un rol fundamental para la construcción de la fase de AD y para la obtención de una AR altamente extensible que permita cubrir las características comunes y variables que demande el desarrollo de los productos específicos de una FSS y detalladamente en el contexto objeto de este trabajo que es el enfoque de FS, considerando todos los aspectos inherentes a la calidad.

Para este proceso IMFS&VCD se consideraron artefactos que se encuentran enmarcados en el proceso de ID específicamente para la fase de AD, ya que se enfoca el estudio desde el problema de dominio a examinar, bien se ha tratado en diversos trabajos[25]; y en la introducción del presente, que la ID tiene como finalidad construir analizar y catalogar componentes de software que se aplican con el objetivo de reutilizar para crear y su AD identifica los dominios, los delimita y permite encontrar variables concordantes.

Esta información es capturada mediantes los modelos que se utilizan en el momento en el cual se crean los artefactos como los componentes reutilizables, si bien es cierto, que existen técnicas ampliamente difundidas para realizar los procesos que involucran los procesos de AD como FORM (*del inglés(Feature Oriented Method),* FAST (del inglés Family-Oriented Specification and Traslation) y COPA (del inglés Component-Oriented Platform Architecting), entre otros[26][27][28], que han sido métodos usados para el desarrollo en el contexto industrial de las Familias de Sistemas de Software, pero ninguno trata explícitamente los aspectos de calidad, es decir, no especifican estándares que garanticen las propiedades de los productos obtenidos, punto que se abarca en el presente trabajo.

El proceso IMFS&VCD que se describe en esta investigación resulta innovador al incluir la VCD basada en el estándar ISO/IEC 25010[6] y al reforzar los avances en la obtención de una AR que ha sido débilmente tratada en el contexto que se estudia de las FS, la mayoría de las técnicas ampliamente difundidas han sido utilizadas para el diseño de arquitecturas de sistemas simples y no para diseñar y obtener las arquitecturas de las Familias de Sistemas de Software como es el objetivo fundamental de las FS.

Aunado a lo anterior es importante señalar que UML, es usada en el modelado de los artefactos ya que proporciona amplias ventajas en el manejo de los aspectos comunes y variables de los activos de software tal como se indica en[2].

También se debe considerar que para MDA la perspectiva de UML es central porque muchas de las herramientas que le dan soporte son basadas en UML y sus perfiles, por ello la OMG ha ido realizando adaptaciones en los últimos años en el contexto de UML 2.0, como lo son el reforzamiento de su infraestructura que internamente define formalmente a UML lo cual es un requisito para el enfoque de la IM, específicamente en lo que se refiere a la transformación de los modelos y la posterior generación de códigos.

Las extensiones, perfiles y estereotipos representan mecanismos que permiten la adaptación del lenguaje a un dominio en particular, a su vez es importante destacar en UML sus perfiles, correspondientes para el modelado en MDA. Algunas de las ventajas que se señalan de la utilización de UML en el contexto de la IM son de acuerdo a[29]:

- La separación de sintaxis concreta y abstracta.
- La extensibilidad.
- La independencia de plataformas.
- La estandarización.

Entonces a través de UML se realiza el metamodelado de los artefactos, que no es más la descripción formal de la posible estructura de los modelos en un nivel de abstracción alto, esto define las construcciones de los lenguajes de modelado y sus relaciones así como las restricciones y las

reglas que se aplican al modelado, entonces el lenguaje de modelado permite definir la sintaxis abstracta y la semántica estática de cualquier lenguaje de modelado.

Importante es considerar que en el metamodelado sólo se obtienen la sintaxis abstracta ya que el aspecto concreto obedece a la implementación de estos metamodelos y modelos, ya en una plataforma determinada, por ello se debe distinguir que en el contexto de la IM y en la disciplina de ID se trabaja bajo la abstracción.[2].

El enfoque de las FS aporta gran variedad de posibilidades respecto al desarrollo cada vez más profundizado de Familias de Sistemas de Software y su utilización en combinación con la IM fomentan la reutilización de activos de software que se derivan en el proceso de la ID y para los cuales es posible garantizar que serán incorporados en la creación de productos de software de calidad, porque se incluye la visión de los estándares de calidad como el propuesto por la ISO/IEC 25010[6] desde etapas tempranas del desarrollo para los Sistemas de Software.

En este trabajo se presentó el proceso IMFS & VCD como una perspectiva para el desarrollo de Familias de Sistemas de Software, basado en el enfoque de FS, donde en el *Esquema* se desarrolla la ID. La IM es incorporada desde una visión de calidad del dominio, para poder obtener la AR en cualquier problema de dominio estudiado en el marco del desarrollo industrial de software y más específicamente para la construcción de los productos de software asociados.

Se presentó el modelo de FS con VCD incluyendo las transformaciones de modelos que son generadas y soportadas desde la ALPS, esto debido a que es en ella donde se desarrollan los activos funcionales y no funcionales de software totalmente reutilizables que constituyen el modelo de calidad, por lo cual se realizó de manera conceptual, ya que esto permitió describir los elementos que conforman el *Esquema* integrado con los elementos de la VCD. Lo cual representa un aporte más de esta investigación a la línea estudiada referida al enfoque de FS.

Otra de las contribuciones que aporta este trabajo entonces, es que IMFS&VCD es un proceso basado en el enfoque top-down para definir arquitecturas, lo cual permite establecer el desarrollo de forma incremental partiendo del estudio y conocimiento del dominio del problema que se va abordar, siguiendo esta premisa es que describe una propuesta paso a paso que va para obtener la AR incluyendo la VCD aplicada al *Esquema* de FS, donde a través del análisis de vistas y puntos de vistas de la descripción arquitectural basada en el estándar IEEE-1471-2000[21], que permite establecer los activos de software para implementar el Esquema, y además todo esto se plantea en el marco de la IM, que aporta los procedimientos de transformaciones necesarias para complementar y refinar los modelos derivados de este, con el fin último de conformar un proceso completo de la fase AD de la ID, en el contexto de las FS. En esta investigación se describen las actividades, tareas/artefactos/entradas/salidas, disciplinas y herramientas para el soporte de todo el proceso IMFS&VCD dentro de la etapa de AD del proceso de ID de la LPS.

IMFS & VCD permite que desde el enfoque top-down, con las ventajas ya señaladas del mismo se logre la obtención de una AR, documentada con actividades basadas en artefactos formales, bien estructurados y se sugiere que sean soportados por diversas herramientas que sustentan todo el proceso, específicamente la utilización de EMF y su Plugin base para UML2 recomendados ampliamente por ser de código de abierto y poseer características y configuraciones para el modelado de FS. De igual modo se toman los basamentos de MDA, para incorporar el proceso de transformaciones de modelos, las cuales se desarrollan en un alto nivel de abstracción y se propone la utilización del lenguaje QVT para describir estas transformaciones que son de tipo M2M.

Como futura investigación en esta línea, se planteará el caso de estudio completo para validar el proceso IMF&VCD propuesto, cuyo dominio de estudio es la FSS de Sistemas Integrados de Salud (SIS) tomando como referencia el caso de la industria venezolana de software que ha sido ya caracterizado en otros trabajos de acuerdo a lo señalado en investigación realizada por[30] que se enfatiza en el manejo de las Historias Clínicas que es el conjunto de documentos con datos, valoraciones e informaciones sobre la situación y evolución clínica de un paciente a lo largo del proceso asistencial y debe ser de carácter confidencial y se debe garan-

tizar la integridad de la información que contiene, entonces para los SIS se convierten en Historias Clínicas Electrónicas (HCE) que son uno de los productos derivados de la familia y es de vital importancia que todos los PS pertenecientes a el dominio de los SIS contemplen el manejo de las HCE ya que no tendría sentido si no se pueden compartir formatos digitales entre diferentes instituciones de salud por ejemplo en el contexto venezolano y con el enfoque de FS.

Referencias

[1] A. Arraiz, F. Losavio, A. Matteo, "Esquema para fábricas de software con un modelo de vista de calidad del dominio," Tercera conferencia nacional de computación, informática y sistemas/CoNCISa /isbn: 978-980-7683-01-02. Universidad de Carabobo, Valencia, Venezuela, octubre, 2015.

[2] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns, Addison Wesley. 2001.

[3] The Open Group. Developing Architecture Views. Introduction. http: www.opengroup.org. 2002.

[4] J. Herrera, F. Losavio, O. Ordaz, "Product Line Scoping for Healthcare Information Systems Using the ISO/IEC 26550 Reference Model". IV Simposio Científico y Tecnológico en Computación / SCTC 2016 / ISBN: 978-980-12-8407-9 Universidad Central de Venezuela, Caracas, Venezuela. 2016.

[5] F. Losavio and A. Matteo, "Reference Architecture Design Using Domain Quality View", Journal of Software Engineering & Methodology, vol. 3, no. 1. 2013.

[6] ISO/IEC 25010 System/Software Product Quality Standard. Secretarriat: Canadá (SCC). 2010

[7] Th. Stahl and M. Völter, "Model Driven Software Development," Techonology, Engineering, management Wiley Publishing, Indianapolis, IN 46256. pp. 180-205. 2006.

[8] P. Amaya, C. González y J.M. Murillo, *Separación de Aspectos en MDA: Una aproximación basada en múltiples vistas*. Quercus Software Engineering Group. Departamento de Informática. Universidad de Extremadura. 2005.

[9] J. Greenfield y K. Short, "Moving to Software Factories". Microsoft Corporation. 2004.

[10] A. Arraiz, F. Losavio, A. Matteo, "Revisión sistemática de fábricas de software e ingeniería de modelos considerando requisitos de calidad,". Venezuela (SCTC). 2014.

[11] M.L. Alvarez, "Desarrollo metodológico de sistemas de control aplicando ingeniería conducida por modelos". Actas de las XXXV Jornadas de Automática, Valencia, ISBN-13: 978-84-697-0589-6. Comité Español de Automática de la IFAC (CEA-2014). 2014.

[12] A. Da Silva, "Model-driven engineering: A survey supported by theunified conceptual model." journal homepage: www.elsevier.com/locate/cl 2015.

[13] M. Piattini y J. Garzas, Fábricas de Software: experiencias, tecnologías y organización. Alfaomega Grupo Editor, S.A. de C.V. México. 2007.

[14] Anliewicz, M., Czarnecki, K. "FeaturePlugin: Feature modeling plug-in for Eclipse". Proceeding of then OOPSLA. 2004.

[15] K. Czarnecki y S. Helsen, "Classification of Model Transformation Approaches," OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture. 2003.

[16] S. Martinez, L. Lamont, R. Moreno y otros, "Análisis de la Transformación de Modelo CIM a PIM en el Marco de Desarrollo de la Arquitectura Dirigida por Modelos (MDA)," Revista Politécnica Vol. 36, No. 1, septiembre 2015.

[17] C. Ariste, J. Ponisio, L. Nahuel y otros, "Diseñando Transformaciones de Modelos CIM / PIM:desde un enfoque de negocio hacia un enfoque de sistema". ASSE 16º Simposio Argentino de Ingeniería de Software, 2015.

[18] J. Rodíguez y J. García, "Ingeniería de modelos con MDA. Estudio comparativo del Optimalj y ArcStyler," Facultad de informática de la universidad de Murcia, España, 2004.

[19] 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. IEEE.SA standars board.

[20] P. Sánchez, Desarrollo de software con aspectos dirigido por modelos.[Documento en línea]. Disponible en: en:http://www.dsi.uclm.es/personal/lenanavarro/dsoa/papersCR/Sanchez_ desarrollo. pdf[Consulta: 2015, marzo 29]. 2005.

[21] IEEE 1471-2000 IEEE.SA standars board, IEEE Recomemended Practice Aschitectural Description of software intensive Systems 2000.

[22] M. Danielle, A. Arsaute y otros, "Transformación de modelos aplicada a la definición genérica de Casos de Uso utilizando QVT (Query/View/Transformation) y RTG (Reglas de Transformación de Grafos) , Universidad Nacional de Río Cuarto, Facultad de Ciencias Exactas, Físico-Químicas y Naturales,Departamento de Computación Argentina. 2005.

[23] C. Naveda, A. Cortez y otros, "Lenguaje Específico de Dominio para Aplicaciones de Negocios," Universidad de Mendoza, Argentina, 2013.

[24] Object Management Group (OMG). Unified Modeling Language Superstructure, version 2.0 (formal/05-07-04).

[25] K. Pohl, G. Bockle & F. van Der Linden, "Software product line engeneering: foundations, principles and techniques". Springer. 2005.

[26] L. Chung & S. Supakkul, "Integrating FRs and NFRs: a use case and goal driven approach" 2nd ICSE, pp 30-37. 2004.

[27] A. Van Lamsweerde, "From systems goals to software architecture. Schhol on formal methods, 25-43. 2003.

[28] M. Matinlasi, "Comparison of software product line architecture design methods: COPA, FAST, FORM, Korba and QADA 26th International conference on software engeneering pp. 127-136. 2004.

[29] D.S. Frankel, "Model driven architecture", Wisley publishing Inc. 2003.

[30] F. Losavio, O. Ordaz and I. Santos, "Proceso de Análisis del Dominio Ágil de Sistemas Integrados de Salud en un Contexto Venezolano". Enl@ce Revista Venezolana de Información, Tecnología y Conocimiento, 12 (1), 101-134. 2015.